Contents

List of Figures

List of Tables

THE UNIVERSITY OF AUCKLAND
SCHOOL OF GRADUATE STUDIES

Co-Authorship Form

Graduate Centre
ClockTower – East Wing
22 Princes Street, Auckland
Phone: +64 9 373 7599 ext 81321
Fax: +64 9 373 7610
Email: postgraduate@auckland.ac.nz
www.postgrad.auckland.ac.nz

This form is to accompany the submission of any PhD that contains research reported in published or unpublished co-authored work. **Please include one copy of this form for each co-authored work.** Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work.

Chapter 4 was extracted from a co-authored work. Chapter 4 is entitled "Using Ensembles For Web Effort Estimation: A Replication".

The paper in question is called "Using Ensembles For Web Effort Estimation". The paper was published for the 2013 ACM/IEEE Symposium on Empirical Software Engineering and Measurement (ESEM). One part of this paper dealt with a replication experiment. The second part dealt with using the Scott-Knott algorithm and the StatREC tool.

I performed the research and wrote the sections dealing with the replication experiment. This is the only part of the paper I have used in my thesis (in Chapter 4). Professor Lefteris Angelis and Dr. Nikolaos Mittas performed the research and wrote the sections dealing with using the Scott-Knott algorithm and the StatREC tool. I have not used their work in my thesis.

Nature of contribution by PhD candidate	Replicating an experiment done using ensembles for effort estimation, on Web project data in the Tukutuku dataset. The experimental work and writing on this was done by me.
Extent of contribution by PhD candidate (%)	50%

CO-AUTHORS

Name	Nature of Contribution
Nikolaos Mittas	Research done using the Scott-Knott algorithm and using the StatREC tool. Writing the sections dealing with this in the paper (remaining 50% of the paper).
Lefteris Angelis	Research done using the Scott-Knott algorithm and using the StatREC tool. Writing the sections dealing with this in the paper (remaining 50% of the paper).

Certification by Co-Authors

The undersigned hereby certify that:

❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and

❖ in cases where the PhD candidate was the lead author of the work that the candidate wrote the text.

Name	Signature	Date
Nikolaos Mittas	[illegible]	4/09/2015
Lefteris Angelis	[illegible]	4/09/2015
		Click here
		Click here

Last updated: 25 March 2013

1 Introduction

The following thesis provides an in-depth look into the use of ensembles for Web effort estimation. This introductory chapter starts with a discussion of why research in the domain of Web effort estimation is important. A summary of the main scientific contributions made by this thesis follows, and the chapter concludes with a discussion of how this thesis is organized.

1.1 Motivation

Effective resource management is crucial for successful software development. It enables managerial decisions to be made with regards to cost, quality, and scheduling tradeoffs inherent in the software lifecycle [14, 5]. Mendes described resources as being any "factors such as cost, effort, quality, "problem size" that have a bearing on a project's outcome" [24]. Resource estimation has traditionally centered on the use of one or more size measures (e.g. lines of code or function points) as key determinants to software development effort, and hence resource requirements.

In comparison to the traditional software industry, Web development is a relatively new and rapidly growing industry, with e-commerce alone weathering the recession and growing by around 19% worldwide in 2013, with similar growth in 2014[1]. This would make

[1]`http://www.emarketer.com/Article/Worldwide-Ecommerce-Sales-Increase-Nearly-20-2014/1011039`

research geared towards enabling Web development companies to make more efficient managerial decisions worthwhile.

Simply porting over existing software resource estimation methodologies would not be adequate as Web development differs from general software development in numerous ways [40, 25]. In summary, Web applications vary widely in terms of structure and implementation from project to project, often make use of numerous non-code elements (e.g. multimedia objects) and may have to work with different (and possibly incongruous) legacy systems [40]. Web development also requires a wider skill set, with graphic designers and writers being involved in the development process in addition to IT professionals [25]. Web projects tend to have a shorter development time with Reifer coining the phrase "quick-to-market software" to describe them [34].

A more detailed look into the differences between Web development and general software development and hence the need for different estimation methodologies is provided in [30, 35].

Various Web resource estimation techniques have been investigated in the literature, with a focus on effort estimation [3]. There is however no consensus as to which effort estimation technique is the best. Accuracy results have been found to vary widely depending not only on the estimation technique used, but also on the dataset investigated, and the validation procedure and accuracy measure/s utilized.

In this thesis we will investigate using ensembles of effort estimation techniques to address this issue, with the aim of demonstrating that they can be used to provide consistently accurate effort estimates. While research has been done to show that ensembles can be used for effort estimation in the general software domain [21], we are interested in a more in depth look at ensemble effort estimation with Web project data.

1.2 Scientific Contributions

The primary scientific contributions made by this thesis are summarized as follows:

1. We carried out a systematic literature review of Web resource estimation with the aim of establishing the current state of the art, and documenting any existing research gaps in this domain [3]. We followed an established review protocol designed to comprehensively identify, evaluate and interpret all relevant research. To our knowledge, a systematic literature review that looks at Web resource estimation in its entirety has not been done previously.

2. The literature review findings provided motivation to use ensembles for Web effort estimation, a novel approach in this particular domain. We replicated the methodology of a study showing ensembles are effective for general software effort estimation,

using Web project data from the Tukutuku dataset [4].

3. We expanded on the replicated methodology, introducing the use of bootstrap aggregation, commonly referred to as bagging, to the process of ensemble creation. This too is novel in the area of Web effort estimation. Bagging enables multiple experimental runs to be performed using a single dataset. We investigated three variants of bagging using 10 experimental runs for each variant, to provide a clearer picture of ensemble Web effort estimation performance.

4. Using ensemble results obtained from the bagging experiments, we broke down ensemble performance via a mathematical formalization of the accuracy-diversity trade-off. This allowed us to quantify the relationship between ensemble error, ensemble diversity, and the accuracy of the component classifiers. The insight obtained from the analysis of this relationship is valuable for creating effective ensembles.

All of the above contributions collectively further the general understanding of ensemble behaviour and performance, when used for Web effort estimation. Practitioners will be able to use these contributions to implement effective effort estimation ensembles, while researchers can build on them to further research into this domain.

1.3 Organization

The remainder of the thesis is organized as follows:

Chapter 2 presents a systematic literature review of Web resource estimation, detailing both the review process and its findings. We conclude this chapter with a discussion of what can be done to address the research gaps revealed by this review, ultimately providing the motivation for our line of research.

In Chapter 3 we provide the background information required for our research. This chapter describes the dataset used, the Tukutuku dataset, which contains data specific to Web effort estimation collected from 195 Web projects from several international companies. We conclude the chapter with a discussion of what ensembles are, how effective ensembles are created, and how they can be used for Web effort estimation.

Chapter 4 is the first of three chapters covering our experimental work. In this chapter we present a replication of a study, one that had shown ensembles were effective for general software effort estimation, using the methodology laid out in this study with Web project data from the Tukutuku dataset. We conclude that ensembles are indeed effective for Web effort estimation and end the chapter with a discussion of the improvements we will implement to this methodology for further research.

In Chapter 5 we discuss using our updated methodology to investigate using bagging with ensembles for Web effort estimation. We provide results for 10 experimental runs carried out for each of the three variants of bagging investigated. We once again show that ensembles are effective for web effort estimation, and consistently so, performing well over all runs and types of bagging.

Chapter 6 analyzes the results presented in the previous chapter in terms of the accuracy-diversity trade-off. The chapter begins with a discussion of the accuracy-diversity trade-off, as well as the mathematical formalization of this trade-off we will use in our analysis. We show why increasing the number of component classifiers an ensemble has may not result in an improvement in performance, and why in certain situations, it can even worsen performance.

Chapter 7 concludes the thesis with a summary of our findings, as well as a discussion of directions that can be taken for future research.

2 Systematic Literature Review

In Chapter 1 we discussed how effective resource management is crucial for successful software development. We also presented the difference between Web development and general software development. Considering the importance that Web development plays in today's industry and its difference from general software development, a detailed insight into Web resource estimation would be valuable. To this end, a systematic literature review would be essential in establishing the current state of the art as well as document existing gaps in the domain.

In this chapter we present a systematic literature review (SLR) of Web resource estimation that is geared at "identifying, evaluating, and interpreting all available research" [17] relevant to resource estimation for Web development. Research that has been done on estimating any factor that has "a bearing on a project's outcome", per the definition Mendes provides for resources [24] will be considered. Despite our research being focused on development effort estimation, we feel that documenting research on resource estimation in its entirety, will provide us information on the datasets, predictors, and estimation techniques used in research that is closely related, and therefore relevant to ours.

The remainder of this chapter is organized as follows: Section 2.1 describes the steps involved in the SLR process. Section 2.2 discusses the SLR findings, followed by a discussion of the results and any research gaps in Section 2.3. Section 2.4 concludes the chapter with a presentation of possible avenues for future research identified by the SLR.

2.1 Systematic Literature Review Protocol

The purpose of a SLR is to comprehensively identify, evaluate and interpret all research relevant to the research questions the review is to address [17]. The following section details the research questions central to this review, as well as the process followed to identify the relevant studies required to do so. This protocol is based on the guidelines published by Kitchenham in [17].

2.1.1 Research Questions

Formulating the research questions that a SLR will address is the first step in the review process [17]. The research questions determine which primary studies are selected, the data to be extracted from these selected studies, and how this data is to be analyzed so that the research questions can be answered.

One approach to formulating research questions is to use the PICOC criteria specified by Petticrew and Roberts [33], which structures research questions according to five attributes: population, intervention, comparison, outcome and context. However, since the focus of this literature review is not to compare interventions, the comparison attribute will not be utilized and hence only the population, intervention, outcome and context (PIOC) attributes of the research questions are shown in Table 2.1.

Table 2.1: Research questions as structured by the PIOC criteria.

Population	Web development projects
Intervention	Methods/techniques used for Web resource estimation, resource predictors considered, and characteristics of the datasets worked on.
Outcome	The effectiveness of the method/technique used for Web resource estimation, i.e. its accuracy [26].
Context	Within the domain of Web development with a focus on empirical studies.

Therefore in order to identify and evaluate all the research done on Web resource estimation, the research questions addressed by our SLR are as follows:

Question 1

What methods and techniques have been used for Web resource estimation?

Question 1a

What metrics have been used to measure estimation accuracy?

Question 1b

What (numerical) accuracy did these various methods/techniques achieve?

Question 2

What resource facets (e.g. effort, quality, size) have been investigated in research on Web resource estimation?

Question 2a

What resource predictors have been used in the estimation process?

Question 2b

At what stage are these resource predictors gathered?

Question 3

What are the characteristics (single/cross-company, student/industry projects) of the datasets used for Web resource estimation?

2.1.2 Search Strategy

The process of identifying primary studies needs to be rigorous and unbiased. In order to minimize researcher bias a pre-defined search strategy was required, and involved the following steps:

1. Identifying search terms to be used in the search process. These were identified using the PIOC attributes detailed in Table 2.1, and from subject headings/keywords used by related articles and journals. Synonyms, alternate spellings, and abbreviations of any search terms identified were also considered.

2. Once the search terms were identified, they were compiled into a search string that would be used in the search process. This was done using the Boolean operators OR and AND. The OR operator was used to group the various forms (e.g. synonyms and alternate spellings) of individual search terms. The AND operator was then used to link the different search terms into a single search string.

The resulting search string is shown in Figure 2.1

(Web **OR** hypermedia **OR** net-centric)
AND
(resource **OR** cost **OR** effort **OR** maintenance **OR** maintainability **OR** quality **OR** reliability)
AND
(estimation **OR** prediction **OR** forecasting **OR** calculation)

Figure 2.1: Final search string.

2.1.3 Search Process

With the search string compiled we began our search process, which was split into a primary and secondary search phase.

Primary search phase

This phase involved identifying and searching through primary sources of relevant literature using our search string. These sources include online databases, search engines, and grey literature (e.g. PhD theses and technical reports). Given that resource estimation for Web development is the focus of this literature review, and that the World Wide Web started as a CERN project in 1989 with the first Web browser Mosaic appearing in 1993 [2], the primary search phase only considered literature published from 1990 (inclusive) to February 2012. The list of primary sources is given in Table 2.2 along with the number of search results and number of relevant papers (see subsection 2.1.4). These resources were recommended by the University of Auckland Library website as resources relevant to Computer Science.

It is important to note that each primary source has its own procedure for entering a search query, with different databases using different keywords for data fields and operators. Therefore our search string in Figure 2.1 had to be tailored to each particular primary source. Initially we used our search string on full text. This however led to thousands of results being returned. We eventually restricted our search to titles and abstracts (and depending on the search engine, keywords).

Table 2.2: Summary of search results.

Database name/ search engine	**Number of search results**	**Number of unique search results**	**Number of selected articles**
Inspec	202	84	14
IEEE Explore	60	60	28
ACM Digital library	96	51	30
Scopus	180	103	12
Springer-Link	20	10	4
ScienceDirect	16	5	0
Web of Science	102	16	3
Computer Database	12	9	1
Current Contents	40	3	0
ProQuest Computing[1]	20	16	0
CiteSeerX[2]	66	40	6
Total	814	397	98

[1]Includes ProQuest Theses and Dissertations.

[2]`http://citeseerx.ist.psu.edu/`

Secondary search phase

The purpose of the secondary search phase is to ensure that the primary search phase has not missed any relevant literature. Our secondary search phase entailed reviewing the references for selected primary studies in order to identify any additional relevant articles. The secondary search phase and the study selection process (discussed in subsection 2.1.4) are iterative in nature, and were repeated until no new literature was found.

2.1.4 Study Selection

Study selection involved assessing the primary studies identified in order to select those that best addressed our research questions.

Inclusion and exclusion criteria for study selection

Studies were selected for the SLR if they met the following inclusion criteria:

1. The study looks at resource estimation within the domain of Web development. Studies can consider any facet of resource estimation, for example, effort estimation.
2. The study describes the methodology, metrics, and datasets used for resource estimation.
3. The study provides an empirical basis for its findings.

In terms of exclusion criteria, studies were excluded if they:

1. Did not focus on estimating a resource factor that is relevant to Web development.
2. Did not provide an empirical basis for their findings.

Selection process

Using the inclusion and exclusion criteria, the primary studies identified by the search phase were screened. Their titles and abstracts were extracted and compiled into a list, and for those that were found relevant, a hardcopy was retrieved. In the situation that the title and the abstract were not sufficiently detailed to determine a study's relevance, a hardcopy was retrieved and used to make a decision. At this stage of the selection process 98 studies were deemed relevant (see Table 2.2 for further details). Each of the 98 studies was assigned a study id, beginning with the letter "S" followed by a numeral between 1 and 98.

In the final selection process, the hardcopies retrieved previously were analyzed in detail, and if a study was still found to be relevant at this stage, it was added to the final reference library for the SLR. After completing the final selection process, a further 21 studies were excluded:

- 7 studies did not focus on estimating a resource factor relevant to Web development (exclusion criterion 1).
- 7 studies did not provide an empirical basis for their findings (exclusion criterion 2).
- 2 studies met both exclusion criteria.
- 4 studies were duplicates of other studies in the reference library, in which case only the most comprehensive study was selected.
- 1 study was not published in English despite what was indicated when it was retrieved during the primary search phase.

The remaining 77 selected studies were used in the secondary search process which led to the inclusion of a further 7 studies. To distinguish studies identified in the secondary search phase from those identified in the primary search phase, they were assigned a study id consisting of the letter "E" followed by a numeral between 1 and 7, bringing the total number of studies in the final reference library for the SLR to 84. A list of all 84 studies is provided in Appendix A.

2.1.5 Study Quality Assessment

A quality assessment checklist was defined to provide a means to quantitatively assess the quality of the evidence presented by these studies. The conclusions drawn from a SLR are only as strong as the evidence they are based on, so compiling an appropriate checklist to assess study "quality" is important [17]. As such, the checklist was not meant to be a form of criticism of any researchers' work.

Table 2.3 details the quality assessment checklist used to evaluate the primary studies. This checklist was adapted from those compiled by Kitchenham [17], with each question utilizing the same three point answer scale, with a "Yes" being worth 1 point, "No" being worth 0 points, and "Partially" being worth 0.5 points. A primary study could thus score between 0 and 12, with the higher the overall score a study obtains, the greater the degree with which this study addresses our research questions. We selected the first quartile (i.e. 3) to act as a cutoff point, with any study scoring 3 or below being excluded from our final reference library. None of the 84 primary studies selected fell into this category.

Table 2.3: Quality assessment checklist for primary studies, adapted from [17].

No.	Question	Answer
1	Are the research aims clearly specified?	Yes/No/Partially
2	Was the study designed to achieve these aims?	Yes/No Partially
3	Are the prediction techniques used clearly described and their selection justified?	Yes/No/Partially
4	Are the variables considered by the study suitably measured?	Yes/No/Partially
5	Are the data collection methods adequately detailed?	Yes/No/Partially
6	Is the data collected adequately described?	Yes/No/Partially
7	Is the purpose of the data analysis clear?	Yes/No/Partially
8	Are the statistical techniques used to analyze the data adequately described and their use justified?	Yes/No/Partially
9	Were potential confounders suitably controlled for in the analysis?	Yes/No/Partially
10	Are the study findings credible?	Yes/No/Partially
11	Are negative results (if any) presented?	Yes/No/Partially
12	Do the researchers discuss any problems with the validity/reliability of their results?	Yes/No/Partially

2.1.6 Data Extraction

The data extraction process involved identifying and recording all the relevant information from the primary studies required to answer the research questions. This was therefore performed for all 84 articles in the final reference library. Data extraction of a subset of these articles was also performed by one of our supervisors and used for comparison to ensure consistency. Results were recorded in a form created specifically for this purpose (as seen in Appendix B), with a separate file being used for each study. In the situation where data was difficult to understand or not clearly detailed in the study, the main author of the study was contacted for clarification.

2.2 Systematic Literature Review Findings

The next phase of the SLR process involved compiling the data extracted from the primary studies in order to address each of the research questions. Data synthesized for each question was tabulated to facilitate any future analysis required. As the tables derived during this phase of the review process are too large to be practical for publication, we have summarized the results in the subsections that follow.

Table 2.4: Methods/techniques used in Web resource estimation[3].

Estimation Technique	Study ID	Percentage (%)
Case based reasoning (CBR)/analogy	S4, S5, S7, S8, S14, S16, S17, S21, S22, S24, S25, S34, S37, S39, S42, S43, S46, S48, S51, S54, S55, S65, S66, S67, S72, S76, S80, S90, S91, S93	35.7
Stepwise regression	S6, S7, S9, S10, S11, S15, S17, S21, S24, S25, S32, S34, S40, S42, S44, S48, S51, S52, S54, S55, S65, S66, S67, S72, S76, S81, S90, S91, E2	34.5
Linear regression	S6, S7, S9, S10, S11, S15, S17, S21, S27, S28, S31, S37, S41, S42, S62, S74, S80, S82, S93, E1	23.8
Bayesian networks	S51, S52, S53, S65, S66, S72, S83, S97, E7	10.7
Classification and regresion trees (CART)	S17, S25, S42, S51, S54	6.0
Support vector regression	S71, S72, S84, S90, S91	6.0
Expert judgment	S27, S28, S32, S34, S40	6.0
Web-COBRA	S27, S75, S77, S93	4.8
Custom	S26(Chilean Web Application Development Effort Estimation), S35, S56, S57, S58 and S69 (Content Management System Effort Estimation Model), S70, S87 (modified versions of WEBMO–WEBMO+ and Vector Prediction Model–VPM+), S92 (Web component model), S96, S98	13.1
Mean Estimation	S25, S44, S46, S48, S53, S54, S55, S65, S66, S67, S75, S76, S77, S81, S82, S91, S93	20.2
Median estimation	S32, S34, S40, S44, S46, S48, S53, S54, S55, S65, S66, S67, S75, S76, S77, S81, S82, S91, S93	22.6
Other	Non-linear regression (S2, E1), function point counts (S30), fuzzy analogy (S43), fuzzy least squares regression (S45), fuzzy radial basis function neural networks (S50), WEBMO (S63), radial basis function neural networks (S64, S89), Tabu search (S85), average unit cost model (E1), use case points (E3), generalized linear model (E5, E6), and hybrid models (CART/linear regression and CART/analogy S42, CART/analogy, CART/stepwise regression S76)	19.0
No estimation technique	S3, S19, S47, S49, E4	6.0

2.2.1 Question 1

Question 1 looks at what methods/techniques have been used for Web resource estimation, the accuracy measures used to evaluate these techniques, and the numerical accuracy achieved. Table 2.4 summarizes the various techniques that have been used for Web resource estimation. These include expert judgment, algorithmic techniques (e.g. linear and stepwise regression), and machine learning techniques (CBR, CART and Bayesian networks). Certain techniques can fall into more than one category, for example Web-

[3]The category "Other" is used to encompass estimation techniques that have been investigated infrequently, being seen in no more than 2 studies. The percentages in the final column do not add up to 100% since a single study may consider more than one estimation technique.

Table 2.5: Performance measures used in Web resource estimation[4].

Accuracy Measure	Study ID	Percentage (%)
MMRE	S4, S7, S8, S10, S11, S14, S16, S17, S21, S22, S24, S25, S27, S28, S30, S31, S32, S34, S37, S39, S40, S41, S42, S43, S44, S46, S48, S50, S51, S52, S53, S54, S55, S58, S62, S63, S64, S65, S66, S67, S69, S71, S72, S74, S75, S76, S77, S81, S82, S84, S85, S87, S89, S90, S92, S93, S98, E1, E2	70.2
Pred(25)	S4, S14, S16, S17, S21, S22, S24, S25, S27, S28, S31, S32, S34, S37, S39, S40, S41, S42, S43 (Pred(0.20)), S44, S46, S48, S50, S51, S52, S53, S54, S55, S58, S62, S64, S65, S66, S67, S69, S70, S71, S72, S74, S75, S76, S77, S81, S82, S84, S85, S89, S90, S92, S93, S96, E1, E2	63.1
MdMRE	S7, S14, S16, S17, S21, S25, S27, S28, S32, S34, S40, S41, S42, S44, S46, S48, S51, S52, S53, S54, S55, S62, S65, S66, S67, S71, S72, S74, S75, S76, S77, S81, S82, S84, S85, S90, S93	44.0
Boxplots of residuals	S6, S9, S15, S16, S17, S21, S24, S25, S27, S32, S39, S41, S42, S48, S52, S54, S62, S65, S66, S71, S72, S74, S76, S82, S84, S85, S90	32.1
Boxplots of z	S25, S41, S42, S52, S65, S66, S71, S72, S84, S90	11.9
MEMRE	S25, S52, S65, S66, S71, S72, S84, S85, S90	10.7
MdEMRE	S52, S65, S66, S71, S72, S84, S85, S90	9.5
MRE	S8, S30, S35, S70, S96	6.0
Other	S26 (expert evaluation), S28 (boxplots of MRE), S34 and S40 (mean absolute residuals, and median absolute residuals), S91 (median of absolute residuals)	6.0
Accuracy measure not used	S2, S3, S5, S19, S45, S47, S49, S56, S57, S80, S83, S97, E3, E4, E5, E6, E7	20.2

COBRA uses expert judgment in conjunction with an algorithmic model. CBR, stepwise regression and linear regression (both simple and multiple) have been used the most frequently in 35.7%, 34.5% and 23.8% of studies respectively. 6% of the primary studies did not use an estimation technique because they did not evaluate an estimation model.

Mean and median estimation represent the simplest estimation scenarios, where the mean and median measure of the resource facet in question, obtained from past projects, is used as an estimate. They are often utilized as benchmarks against which other techniques are compared, the reasoning being if the estimation technique in question is not superior to either the mean or median estimate, there is no point in using it.

Tables 2.5 and 2.6 address the final aspects of the first research question and deal with the accuracy measures used to evaluate Web resource estimation techniques, and the accuracy which these measures obtained respectively.

The absolute residual (a residual is the difference between the estimate and the actual

[4]The category "Other" is use categorize performance measures that have been used infrequently, being seen in no more than 2 studies. The percentages in the final column do not add up to 100%, since a single study may use several accuracy measures.

Table 2.6: Performance obtained by the six most frequently used effort/cost estimation techniques.

Estimation technique	**Accuracy achieved (%)**
CBR	MMRE: 7.00 – 14430.99 MdMRE: 6.00 – 5146.52 Pred(25): 0.00 – 100.00
Stepwise regression	MMRE: 1.50 – 2.62E+12 MdMRE: 0.62 – 5668.56 Pred(25): 0.00 – 100.00
Linear regression	MMRE: 1.50 – 110.00 MdMRE: 0.62 – 100.00 Pred(25): 40.00 – 100.00
Bayesian networks	MMRE: 34.26 – 3731.00 MdMRE: 27.42 – 805.00 Pred(25): 0.00 – 33.33
Mean	MMRE: 31.64 – 31208.52 MdMRE: 25.61 – 8781.81 Pred(25): 0.00 – 49.00
Median	MMRE: 32.25 – 32542.41 MdMRE: 23.00 – 9160.36 Pred(25): 0.00 – 66.67

value) forms the basis for all the numerical measures of performance. The MRE (magnitude of relative error) is calculated by considering the absolute residual relative to the actual value. The mean and median MRE (MMRE and MdMRE) along with Pred(25) (the percentage of estimates with an MRE of 25% or less) were proposed by Conte et al. [10], and are the most frequently used measures of accuracy being seen in 70.2%, 44% and 63.1% of the primary studies respectively. MMRE and Pred(25) are frequently used in conjunction with each other (86% of the time), which is not particularly surprising as they are considered to be two of the most widely used accuracy measures in software estimation [19].

The absolute residual can also be considered relative to the estimate (estimation magnitude of relative error or EMRE), and the associated mean and median EMRE values, proposed by Kitchenham et al. [19], have been used in 10.7% and 9.5% of the primary studies respectively. Whereas accuracy measures using MRE are sensitive to overestimates, accuracy measures using EMRE are sensitive to underestimates [11]. Appendix C provides the mathematical formulation for all these numerical performance measures.

Boxplots are graphical representations of accuracy, and are typically used to compliment numerical accuracy measures: by providing a graphical representation of the distribution of residuals (or z, or MRE), boxplots enable a visual comparison of different estimation techniques and may also help explain the values obtained by numerical accuracy measures [19].

Due to the amount of data collected, presenting all the findings for numerical accuracy achieved for all estimation techniques here, is not possible. Instead Table 2.6 summarizes the accuracy achieved by the six most frequently used estimation techniques. Accuracy was also restricted to that measured by MMRE, MdMRE, and Pred(25) which are the three most commonly used accuracy measures. Lastly, only effort/cost estimation is considered by this table due to the lack of accuracy findings for quality and maintenance effort estimation (see section 2.2.2). For the complete findings please refer to Appendix E.

What stands out immediately from Table 2.6 is the incredibly wide range of accuracy values for all three accuracy measures, for all estimation techniques. If the thresholds suggested by Conte et al. [10] for a good estimation technique are used (MMRE and MdMRE $\leq 25\%$, and Pred(25) $\geq 75\%$), then only CBR, stepwise regression and linear regression achieve this, although based on the ranges seen accuracy varies widely from excellent to extremely poor.

The amount of variation is likely due to the fact that the estimation accuracy achieved by a particular technique is dependent upon a large number of factors including, what measure of accuracy is used, the dataset characteristics (academic/industry, single-/cross-company see section 2.2.3), the particular configuration of the estimation technique used, the cross-validation technique used, and even the software used to execute the technique. As no fixed standard exists in terms of what technique should be used (or how it is used) in a particular estimate, different studies using the same technique will naturally produce quite different results.

This also makes comparing different techniques difficult. For example looking at Table 2.6, it seems that linear regression has produced the best results overall, based on the ranges provided for MMRE, MdMRE, and Pred(25). However if the studies that use linear regression (listed in Table 2.4) are cross-referenced with dataset characteristics (listed in Table 2.10, section 2.2.3), it can be seen that when effort/cost estimation is in question, linear regression has only been used on academic and single-company industry datasets, where more favorable estimation results have been obtained.

2.2.2 Question 2

Data collected for the second research question will provide information as to which areas of the Web resource estimation domain have been studied, what predictors are considered the most important, and when in the development cycle they are gathered. This data is divided into three tables, with Table 2.7 dealing with the facets of resource estimation investigated, Table 2.8 dealing with the related predictors, and Table 2.9 dealing with when they were gathered.

It is immediately obvious from Table 2.7 that the majority of research in the field of

Table 2.7: Resource facets investigated.

Resource facet investigated	Study ID	Percentage (%)
Design	S19, S47, E4	3.6
Quality	S37, S80, E3	3.6
Maintenance	S45, S49, S57, S98, E1	6.0
Size	S30	1.2
Cost/Effort	All remaining studies	85.7

Web resource estimation has focused on development cost/effort[5] estimation, with 85.7% of the primary studies selected by this SLR focusing on this area. Design effort estimation is related to cost/effort estimation with studies S19, S47, and E4 looking at estimating the effort for just the design phase of the development process.

In terms of quality estimation, studies S37 and S80 characterized software quality into four factors; testability, error proneness, reliability and fault tolerance, and looked at estimating various metrics associated with these factors. Study E3 on the other hand focused on test effort estimation which was considered part of testability by studies S37 and S80. It is interesting to note only study S37 offered any form of accuracy assessment (MMRE and Pred(25)) as seen in Table 2.5.

Studies S45, S49, S57, S98 and E1 all consider maintenance effort estimation. Studies S45 and E1 are related in that they used the same dataset of 15 maintenance tasks on a single Web application. Only two studies offered any form of accuracy assessment with study S98 using MMRE and study E1 using MMRE and Pred(25) accuracy measures.

Study S30 looked at estimating Web application size using four different variations of function point counting. As mentioned previously, size is regarded as a key determinant in development effort estimation and hence resource requirements.

Looking at the percentages in the rightmost column of Table 2.7, it can be seen that they add up to 100% (barring a slight rounding error). This indicates that all the primary studies in the SLR have focused on a single facet of Web resource estimation.

Table 2.8 lists the different predictors that have been used for resource estimation. These predictors have been categorized as size measures, complexity measures, cost drivers, Tukutuku measures, and other measures (a category used for predictors that have only been used infrequently). Only a single study, E4, did not investigate any predictors, as it was an exploratory study designed to investigate a series of hypothesis on design effort estimation as opposed to evaluating the estimation process itself.

Size measures are unsurprisingly the most frequently used resource predictors, being seen in 58 out of the 84 studies (69%). Size measures can be furthered categorized into length measures (e.g. page count and media count), functionality measures (e.g.

[5]Cost and effort have been used interchangeably in the primary studies.

Table 2.8: Resource predictors investigated[6].

Resource Predictors	**Study ID**	**Percentage (%)**
Size: Length	S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S14, S15, S16, S17, S19, S21, S25, S27, S28, S37, S39, S42, S47, S48, S49, S57, S62, S70, S75, S76, S80, S83, S87, S92, S93, S96, S97, S98, E2, E5, E6, E7	50.0
Size: Functionality	S2, S9, S15, S26, S28, S30, S31, S35, S41, S42, S45, S46, S48, S56, S57, S58, S62, S63, S69, S70, S74, S77, S82, S93, S96, E1, E3	32.1
Size: Reusability	S4, S5, S6, S7, S8, S10, S11, S14, S17, S19, S21, S25, S39 S47, S83, E2, E5, E6	21.4
Complexity	S3, S4, S5, S6, S7, S8, S9, S10, S11, S14, S15, S16, S17, S19, S21, S25, S35, S37, S39, S47, S49, S57, S80, S97, S98, E2, E5, E6, E7	34.5
Cost Drivers	S26, S27, S35, S56, S58, S63, S69, S75, S77, S83, S87, S93, S98, E1, E6, E7	19.0
Tukutuku	S22, S24, S32, S34, S40, S43, S44, S46, S48, S50, S51, S52, S53, S54, S55, S64, S65, S66, S67, S71, S72, S81, S84, S85, S89, S90, S91	32.1
Other	Separation of concerns (S37 and S80), quality measures (S57), reliability measures (S98)	4.8
No predictors Investigated	E4	1.2

COSMIC functional size units - Cfsu), and reusability measures (e.g. reused page count and reused media count). It seems that while reusability measures are not as frequently used as length measures, whenever they are used, they are always used in conjunction with length measures.

Complexity measures have also frequently been used as resource predictors (in 34.5% of the primary studies). Complexity measures include connectivity, application structure, and in the case of studies dealing with design estimation (S19 and S47), quality estimation (S37 and S80), and maintenance estimation (S49), cohesion and coupling measures. Complexity measures are often used in conjunction with size measures.

Cost drivers are any predictors that do not characterize the size or complexity of a Web

[6]The percentages in the final column do not add up to 100% as a study may investigate more than one type of predictor.

Table 2.9: Stage at which resource predictors are gathered.

Stage	**Study ID**	**Percentage (%)**
Early	S2, S16, S19, S26, S30, S31, S35, S41, S46, S47, S48, S58, S69, S82, S92, E1, E3, E4	21.4
Late	S2, S15, S16, S27, S28, S49, S57, S70, S80, S82, S96, S97, S98	15.5
Not specified	Remaining 56 studies	66.7

project (e.g. development team size and experience), and are seen in 19% of the primary studies. They are also always used in conjunction with size measures. For example studies S27, S75, S77, and S93 use Web-COBRA for development effort/cost estimation. Web-COBRA uses costs drivers to build a causal model relating the cost drivers to development effort/cost. These causal relationships are then quantified through expert judgment, and used along with a length (Web Objects) or functional (Cfsu) size measure to generate an estimate.

Tukutuku is a dataset of information on Web hypermedia and software applications from companies around the globe [29]. Each project in the dataset is characterized by a set of 25 variables related to Web applications and their development, of which length and reusability measures and cost drivers have been used as predictors in resource estimation (in 32.1% of the primary studies). The Tukutuku dataset is discussed in Chapter 3, while a more detailed look at the Tukutuku variables used as resource predictors is provided in Appendix D.

With regards to when in the development (or maintenance) cycle the resource predictors are gathered, Table 2.9 shows that two thirds of the primary studies surprisingly do not specify this information. Out of the 28 studies that do, 18 gather resource predictors at an early stage (e.g. after requirements analysis), 13 at a late stage (i.e. after implementation), and 3 investigate predictors collected both early on and after implementation. There is however a general consensus that for resource predictors to be useful, they should be able to be gathered as early in the development (or maintenance) cycle as possible.

2.2.3 Question 3

The final question is directed at investigating the characteristics of the datasets used in Web resource estimation. We are interested in whether academic or industry datasets were considered, and if they were industry datasets, whether the Web project data came from a single company or from several companies (cross-company data).

Table 2.10: Domain of the dataset used.

Domain	**Study ID**	**Percentage (%)**
Industry	S2, S22, S24, S27, S28, S30, S32, S34, S35, S37, S40, S42, S43, S44, S45, S46, S48, S49, S50, S51, S52, S53, S54, S55, S56, S57, S58, S62, S63, S64, S65, S66, S67, S69, S70, S71, S72, S74, S75, S76, S77, S81, S82, S83, S84, S85, S87, S89, S90, S91, S92, S93, S96, S97, S98, E1, E3, E7	69.0
Academia	S3, S4, S5, S6, S7, S8, S9, S10, S11, S14, S15, S16, S17, S19, S21, S25, S26, S31, S39, S41, S47, S58, S69, S87, E2, E4, E5, E6	33.3
Not specified	S80	1.2

It can be seen from Table 2.10, that most Web resource estimation has been done using industry datasets (69% versus 33.3% for academic datasets). Table 2.11 shows that there is almost an even split between cross-company and single-company industry datasets (53.4% and 50% respectively). A small selection of primary studies consider both industry and academic datasets (S58, S69, and S87). In fact studies S58 and S69 use both single- company and cross-company datasets. Only a single study (S80) did not specify the domain of its dataset. Note that the percentages in the final column of both tables do not add up to 100% as a study may investigate more than one type of dataset.

Looking at the studies that use cross-company industry data, it can be seen that the majority of these datasets (80.6%) come from the Tukutuku dataset. This is because as far as we know, Tukutuku is a unique repository of cross-company Web project data.

2.3 Discussion

The results of the SLR address three areas of Web resource estimation research, as defined by our research questions:

1. The techniques used for Web resource estimation and their accuracy.
2. What resource facets have been investigated and the predictors considered.
3. The characteristics of the datasets used in the empirical research.

We can see that a variety of techniques have been used in Web resource estimation, including expert judgment, various algorithmic and machine learning techniques, as well

Table 2.11: Type of industry dataset used.

Type	**Study ID**	**Percentage (%)**
Cross-company	S22, S24, S32, S34, S37, S40, S43, S44, S50, S51, S52, S53, S54, S55, S56, S58, S63, S64, S65, S66, S67, S69, S71, S72, S81, S83, S84, S85, S89, S90, S91	53.4
Single-company	S2, S27, S28, S30, S35, S42, S45, S46, S48, S49, S57, S58, S62, S69, S70, S74, S75, S76, S77, S82, S87, S92, S93, S96, S97, S98, E1, E3, E7	50.0

as those that fall into more than one category. Estimation accuracy forms the basis for evaluating these techniques, and a number of numerical and graphical measures of accuracy have been used, most of which are based on the concept of the absolute residual.

Despite the number of estimation techniques investigated, there does not appear to be any guidelines as to which techniques to use in a particular estimation scenario, how to configure these techniques, or what accuracy measures to use to evaluate them. Consequently we have obtained an incredibly wide range of accuracy results from excellent to extremely poor. Given the fact that estimation accuracy is dependent on these factors (amongst others), it was not possible to use the results in a meta-analysis directly comparing the different techniques. As such, practitioners will find the lack of resource estimation guidelines a formidable hurdle to overcome.

In terms of the second research question, the SLR demonstrates that in the domain of Web resource estimation, work has been done on effort/cost, design, quality, maintenance, and size estimation. The focus has primarily been on development effort/cost estimation with only 14.4% of the primary studies selected by this SLR dealing with other resource facets. If we take into account that design effort estimation is a subset of development effort estimation, and that paper S30 looked at size estimation as "fundamental" for development effort estimation, then this percentage drops to 9.6%.

These results are not surprising given that resource estimation has typically focused on development effort estimation. What is surprising is that out of the three studies dealing with quality estimation, only one provided an accuracy assessment, and out of the five studies dealing with maintenance estimation, only two provided an accuracy assessment. This lack of evaluation limits the usefulness of these studies for practitioners looking to undertake quality or maintenance estimation.

There does not seem to be any research in the field of Web resource estimation that considers more than a single resource facet simultaneously. Given how size, quality, development and maintenance are related to each other in software development in general, research into investigating a more comprehensive model of Web resource estimation would prove useful to practitioners.

As mentioned previously Web resource estimation has traditionally viewed size measures as key predictors of resource requirements [14]. This still holds true with length, reusability, and functionality size measures being seen in 69% of the selected studies. If you take into account the fact that size measures are included amongst the Tukutuku variables, then every single primary study that investigates resource predictors (i.e. all studies but E4), considers size measures as predictors of resource estimation.

For practitioners looking to select resource predictors, most research done into Web resource estimation selects predictors on the basis that they are correlated with whatever resource facet is being estimated. Certain estimation techniques like Bayesian nets and Web-COBRA however, utilize predictors that have a cause and effect relationship with the resource facet being estimated. These predictors are usually elicited via input from a domain expert.

Industry datasets are more frequently used than academic datasets for empirical research into Web resource estimation. These industry datasets can be categorized into single company datasets and cross-company datasets according to whether they use data from one or several companies respectively. Estimates from single company datasets appear to be superior to those from cross-company datasets, which corresponds to findings from prior research that has been done on single versus cross-company estimates, in both Web and general software resource estimation [18]. Single company datasets are smaller than their cross-company counterparts, of which the Tukutuku dataset is the largest and most often used.

There are a few issues encountered during the SLR process that are worth mentioning. Tailoring the search string for every search engine can be time consuming. Different search engines also offer different levels of utility, which can make certain searches more difficult than they should be. The databases/search engines also tend to overlap significantly resulting in numerous duplicate search results that need to be weeded out. Lastly, whilst quality checklists are useful in evaluating the extent to which studies answer SLR research questions, because they follow a general guideline they sometimes, in our opinion, do not provide an accurate indication of study quality. For example certain questions in our quality checklist were difficult to answer as they were often not directly addressed by the studies.

2.4 Conclusion

This chapter presents a systematic review on Web resource estimation. The primary search phase returned 397 unique results, of which 98 were selected. Of these 98, 21 were excluded upon detailed analysis leaving 77 studies. These 77 studies were then used in the secondary search phase resulting in the retrieval of 7 more studies, bringing the final total of selected studies to 84. Relevant data was extracted from these studies and then synthesized in order to answer our research questions, which aimed to establish the current state of the art in Web resource estimation, as well as any possible research gaps.

We have found that there is no gold standard when it comes to resource estimation techniques. Accuracy results vary widely being dependent on numerous factors including choice of accuracy measures. We have also discovered that most work on resource estimation has focused on development effort/cost estimation, with little done on areas like quality or maintenance effort estimation. As far as we know, no work has yet been done where Web resource estimation has encompassed more than just a single resource facet. Size measures have traditionally been key resource predictors in software estimation in general, with Web resource estimation being no different. Every single study in this SLR that investigated resource predictors used a size measure of some sort. Empirical research into Web resource estimation has favored the use of industry datasets over academic datasets. Of these industry datasets, single company datasets seem to produce superior estimates than cross-company datasets, although a direct comparison between the accuracy findings obtained by studies in this SLR is not possible.

Given the number of estimation techniques available and the fact that based on accuracy results, there is no consensus as to which are the best, we suggest further exploration of existing techniques. This would involve research into what technique to choose in a particular estimation scenario, how to configure the various estimation techniques (e.g. choice of numerical parameters), the experimental setup to validate the choice of techniques (e.g. the type of cross-validation used), as well as the accuracy measures to use in this validation. The end result would be a set of guidelines to help practitioners looking to undertake Web resource estimation.

Kocaguneli et al. [21] have taken another approach to this problem by considering effort estimates generated by ensembles of estimation techniques. This involves combining the estimates of a number of individual estimation techniques. They found that ensembles of the best techniques outperformed all solo techniques and more importantly, maintained their performance over 20 datasets and several accuracy measures. Given our access to Web project data related to effort estimation from the Tukutuku dataset, we decided that we would use this approach as a foundation with which to build our research on. Naturally as the work in [21] was in the general software domain, this will first involve verifying whether ensembles of estimation techniques are effective on Web project data by doing a replication study using the Tukutuku dataset. If this replication is successful we will then expand on the approach used in [21] based on the replication findings. Before discussing the replication it is important to have a clear understanding of the Web project data in Tukutuku that the research will be dealing with, as well as with the concept of using ensembles of machine learning techniques. These topics will be described in greater detail in the following chapter.

3 Background

In Chapter 2 we presented a systematic review on Web resource estimation. We demonstrated that the Tukutuku dataset has provided most of the cross-company data used in Web effort estimation research, making it a vital resource for our research area. We also discussed, in Section 2.4, that our research would focus on using ensembles for Web effort estimation.

In the following chapter we will provide the required background on both the Tukutuku dataset and the concept of ensembles. We will start this chapter with a discussion of the dataset, from the process of its creation to its contents as relevant to our research. We will then conclude this chapter with a discussion of ensembles from a machine learning perspective, looking at how they relate to the domain of Web effort estimation.

3.1 Tukutuku Dataset

Tukutuku is a dataset of project data from Web applications developed by companies around the globe [29]. It was developed as part of the Tukutuku Benchmarking Project, whose aim was to compile data related to Web project development that could be used to benchmark productivity, and develop effort estimation models.

This dataset currently has data on 195 projects, with each project in the dataset being characterized by a set of 25 variables related to Web applications and their development. This is important, as research has shown that using metrics tailored for Web projects results in superior estimates than when using general software development metrics like function points [36, 35]. As far as we know in this regard, Tukutuku is a unique repository of cross-company Web project data.

The groundwork for the Tukutuku Benchmarking Project was laid down by a three part investigation into Web size metrics [29]:

1. A survey labelled S1 whose aim was to identify size metrics and cost factors related to Web development. These size metrics and cost factors had to be able to be measured early in the development process.

2. A case study to validate the findings of S1.

3. A second survey (S2) whose aim was also to validate the findings of S1.

As such we will first discuss the initial survey S1 as well as its validation (case study and S2), before moving on to the Tukutuku Benchmarking Project. We will complete our discussion of the Tukutuku dataset with a description of its characteristics.

3.1.1 The S1 Survey

The S1 survey used online forms for Web project price quotations, to identify size metrics and cost factors that could be measured early in the development cycle and then used for Web effort estimation [29]. Data from 133 such forms was collected and divided into six categories. Three of these categories directly focused on size measures and cost factors: Web application "static metrics" (those used to size applications), Web application "dynamic metrics" (those related to application functionality) and cost drivers (described in section 2.2.2). Web project metrics included measures that had a bearing on a project's contigency and/or profit costs (e.g. budget available for the project). Attributes that characterised a Web company (e.g the company's customer base) were grouped under Web company metrics, while attributes dealing with the final appearance of the Web application (e.g. CSS styles to be used) were grouped under Web interface style metrics. The percentage of companies requiring each metric in their online quotation form was calculated and used as a way of ranking these metrics in terms of importance.

The total number of Web pages was the most commonly used size metric, being required in 70% of the online price quotation forms [29]. Application features and functionality were also commonly required information, requested by 66% of the companies. A single cost driver was identified, categorizing projects into new projects or enhancements and was used by 31% of the companies. Metrics related to Web interface style however were not commonly used (5%–17% of forms). Web project metrics and some of the Web company metrics were seen as being important to project pricing rather than effort estimation. To address this a case study and a further survey were organized to validate the metrics collected by S1 and identify only those directly relevant to Web effort estimation.

3.1.2 Case Study

The aim of the case study was to use an experienced Web development company to evaluate the metrics identified in S1, focusing on those important to Web effort estimation [29]. The results from S1 were presented to one of the company's directors, who was asked to identify which of these metrics the company utilized in their effort estimation process, taking into account the rankings seen in S1 (i.e. the percentage of companies that required a particular metric).

There was a general agreement with regards to the Web size measures identified by S1; a few more size measures were added and some of them were re-arranged (in terms of importance) [29]. More metrics related to features and functionality were added, and more importantly, a complexity level was added to each feature/function to take into account differing levels of implementation difficulty. The case study deemed none of the Web interface style metrics important for Web effort estimation. As with S1, Web project metrics and some of the Web company metrics were seen as being important to project pricing rather than specific to effort estimation.

3.1.3 The S2 Survey

The S2 survey was also performed to validate the findings of S1 [29]. S2 was carried out via phone interview with representatives from 32 New Zealand Web companies. Nine questions were asked related to Web application static metrics, Web application dynamic metrics, company demographics, pricing procedure, factors used for effort estimation in addition to the static and dynamic metrics (cost drivers, Web project metrics and Web company metrics), when in the development cycle the effort estimate is generated, and whether effort estimates were modified during the development process.

The Web application static metrics used to size applications, identified by S1 and the case study, were split into four categories[29]. Every company surveyed in S2 used all four categories of size metrics for effort estimation. In terms of Web application dynamic metrics related to application functionality, the Web companies either provided a list of features/functionality for clients to choose from or asked clients for required features/functionality without a list, providing suggestions where necessary. Only 22% of the Web companies surveyed took into account the complexity of features/functions required. Three cost drivers were identified as additional factors used for effort estimation. The first of these, differentiating between new and enhancement projects, was also identified by S1 and the case study. The two new cost drivers considered development team experience and development tools used. None of the Web project metrics, Web company metrics or Web interface style metrics were identified as being relevant to effort estimation.

3.1.4 Tukutuku Benchmarking Project

The Tukutuku Benchmarking Project involved the collection of Web project data from development companies worldwide to be used in the creation of Web effort estimation models, and to benchmark Web company productivity [29]. Web forms were used in the data gathering process and these were created utilizing the information obtained from the two surveys and case study discussed previously.

As both the case study and some of the Web companies surveyed in S2 took into account the complexity of features/functionality being implemented, this was also incorporated into the data collection forms [29]. Web companies filling in these forms would indicate whether a project's features/functions were implemented via "black box reuse, reuse with adaptation, or new development". Each feature/function would also be categorized as being either "high effort" or "low effort". For clarification, companies would provide the number of person hours they thought were representative of high effort for the development or adaptation of a feature/function.

In all 25 variables were used to characterize a Web application and its development process [27]. These included size measures like the number of Web pages and images, measures related to features/functionality taking into account implementation complexity (e.g. number of reused high effort features/functions), and cost drivers (e.g. team experience with the development language used). A complete list of these 25 variables and their description is provided in Appendix F.

It is important to note that the data obtained for the Tukutuku Benchmarking project was not measured using automated tools [29]. To distinguish between guesstimates and more accurate data recording, the Web forms also gathered information on how the companies collected effort data. It was found that data for at least 77.6% of Web projects in the Tukutuku dataset was collected using timesheets that measured hours worked per project per day/week, or hours worked per project task per day. Naturally the use of timesheets is not a guarantee of absolute accuracy with regards to the effort data collected, and this remains a weakness of the Tukutuku dataset.

3.1.5 Tukutuku Dataset Characteristics

The Tukutuku dataset is a cross-company dataset; currently Tukutuku has data on 195 Web projects developed by 51 Web companies from around the globe. As discussed previously, each project is characterized by 25 variables related to the Web application and its development. Of these 25 variables we will only consider the 15 numerical variables.

Table 3.1: The numerical Tukutuku variables

Variable	**Description**
nLang	Number of different development languages used.
DevTeam	Size of a project's development team.
TeamExp	Average team experience with the development language(s) used.
TotWP	Total number of Web pages (new and reused).
NewWP	Total number of new Web pages.
TotImg	Total number of images (new and reused).
NewImg	Total number of new images created.
Fots	Number of features reused without any adaptation.
HFotsA	Number of reused high-effort features/functions adapted.
Hnew	Number of new high-effort features/functions.
TotHigh	Total number of high-effort features/functions.
FotsA	Number of reused low-effort features/functions.
New	Number of new low-effort features/functions.
TotNHigh	Total number of low-effort features/functions.
TotEff	Actual total effort in person hours to develop an application.

Based on previous effort estimation research done using the Tukutuku dataset [27, 28], along with a discussion with one of the creators of the dataset, we believed that using the categorical variables:

- Would not be of value for this analysis.
- Would require the use of dummy variables reducing the degrees of freedom for analysis.

A description of these numerical Tukutuku variables is provided in Table 3.1 and their summary statistics in Table 3.2. The first 14 of these numerical variables (i.e. all the variables apart from `TotEff`) will act as the independent variables in the estimation process, with the dependent variable being the estimated total effort, in person hours, required to develop an application (which we will refer to as `EstEff`). Looking at the independent variables it can be seen that they consist of size measures (e.g. `TotWP`), measures related to features/functionality (e.g. `Fots`) and cost drivers (e.g. `DevTeam`). Complexity is taken into account for measures related to features and functionality with metrics for high effort features/functions (e.g. `Hnew`) and low effort features and functions (e.g. `New`). Distinction is also made between features and functions that were implemented from scratch (referred to as “new”) or those “reused” with or without adaptation from existing functionality. Comparison between `EstEff` and `TotEff` will be used to evaluate estimation performance.

Table 3.2: Summary statistics of the numerical Tukutuku variables

Variable	Mean	Median	Std. Dev	Min	Max
nLang	3.9	4	1.4	1	8
DevTeam	2.6	2	2.4	1	23
TeamExp	3.8	4	2.0	1	10
TotWP.	69.5	26	185.7	1	2000
NewWP	49.5	10	179.1	0	1980
TotImg	98.6	40	218.4	0	1820
NewImg	38.3	1	125.5	0	1000
Fots	3.2	1	6.2	0	63
HFotsA	12.0	0	59.9	0	611
Hnew	2.1	0	4.7	0	27
TotHigh	14.0	1	59.6	0	611
FotsA	2.2	0	4.5	0	38
New	4.2	1	9.7	0	99
TotNHigh	6.5	4	13.2	0	137
TotEff	468.1	88	938.5	1.1	5000

3.2 Ensembles

Creating an ensemble involves combining the estimates from a number of individual estimation techniques, examples of which are seen in Table 2.4 (see section 2.2), with the aim of obtaining a single, more accurate estimate.

Before we discuss ensembles we need to understand how a learning problem like Web effort estimation is represented. In machine learning, effort estimation would be considered to be a supervised learning problem; one that involves learning a function from a set of examples where both the inputs and outputs are known [37]. The following subsections describe what a supervised learning problem entails, before going on to discussing how to create effective ensembles, and why ensembles work. We will end this section with a discussion of ensembles as they relate to Web effort estimation.

3.2.1 Supervised Learning

The set of examples a supervised learner receives is called a training set. Each example within the training set is a tuple $\langle x, y\rangle$ where x is referred to as the input and y the output or outcome [12]. The input x is usually a vector $\langle x_1, x_2, x_3, \ldots, x_n\rangle$, where x_j is referred to as the jth feature of x. The output y can either be obtained from a set of discrete classes (a classification problem) or from the set of real numbers (a regression problem).

The true function F, that maps x to y is not known to the learner [15]. However, the learner can use a training set to produce a function C, known as a classifier, that approximates F. The classifier C can then be used to make predictions $\hat{y}$ from new values of x.

3.2.2 Creating An Effective Ensemble

An ensemble of classifiers is created by combining the predictions of individual classifiers to provide improved predictions for unseen examples [12, 15]. Combination schemes for classifiers include weighted or unweighted majority voting (for a classification problem), and weighted or unweighted averaging (for a regression problem) [6].

In order for an ensemble of classifiers to be effective, its component classifiers need to be both accurate and diverse [12]. This can be illustrated using a simple classification problem as an example: There are two possible classes, present in roughly equal amounts, for the outcome y. For a classifier to be considered accurate it would need to have an error rate less than that obtained through random guessing. In this particular scenario this would mean an error rate of less than 0.5. For a set of classifiers to be diverse they would need to make different errors from the same input data x. If for example there were three classifiers that were identical, if one classifier makes an error in classification, then the remaining two classifiers would also make the same error, as would the ensemble when majority voting is used. If however the three classifiers were not identical (i.e. made different errors from the same input data x), then it would be plausible for one classifier to incorrectly classify the output while the remaining two classifiers correctly classify the output. The resulting ensemble would thus also classify the output correctly through majority voting.

With the above scenario, if the classifiers share the same error rate e, and if they make errors independently from each other, then the probability that a classifier will make an incorrect classification can be modelled with a Binomial distribution, with parameters n and p being equal to the number of classifiers and $1 - e$ respectively [12]. The probability that an ensemble from these classifiers makes an incorrect classification would then be equivalent to the probability that more than half of the classifiers make an error. If the individual classifiers are accurate, i.e. their $e < 0.5$, then this probability would be smaller than the probability that a single classifier makes an error. In other words, the ensemble will be more accurate than its individual classifiers. If, on the other hand, the classifiers are not accurate and have $e > 0.5$, the resulting ensemble would be less accurate than its individual classifiers.

3.2.3 Why Do Ensembles Work?

To understand the reasons why ensembles work, it is necessary to view the process of learning as a search; a learner can be seen as using training data to search through the space of possible classifiers for one that best approximates the true function F between the inputs x and the outputs y [12]. Dietterich categorises these reasons as being "statistical", "representational", and "computational":

- **Statistical:** To illustrate the statistical reason for why ensembles work, let us consider a learner that has to produce a classifier from insufficient training data. This limits how much of the space of possible classifiers the learner can search through, to find one that best approximates the true function F. Many different classifiers can be produced by the learner, depending on the training data it is given, but since this data is not sufficient for a complete search, which of the many classifiers produced is the "right" one? When a set of accurate classifiers created by the learner is used to construct an ensemble, their individual predictions are "averaged out" resulting in a classifier that better approximates F, and thus produces more accurate predictions than the individual classifiers [12].

- **Representational:** Depending on the characteristics of the learner as well as the amount of training data it has access to, a learner may only be able to produce a finite set of classifiers; none of which may be representative of the true function F. By combining classifiers, for example those produced by different learners, the resulting ensemble classifier may be able to represent a larger selection of functions, providing a better approximation of F and hence more accurate predictions [12].

- **Computational:** Even if there was sufficient data to search through the space of all possible classifiers and find the optimum one, i.e. one that best approximates F, this would be a computationally expensive task [12]. A number of learners, for example neural nets and decision trees, use local search to find a classifier, and these can get stuck in local optima. With an ensemble, learners can be run using different starting points for their local search. The resulting classifiers when combined in an ensemble, provide a better approximation of F and therefore more accurate predictions.

3.2.4 Ensembles For Web Effort Estimation

We will now discuss ensembles as they relate to Web effort estimation. As mentioned in the previous section, Web effort estimation is a supervised learning problem. The inputs x are Web project characteristics or those related to its development, that play a role in estimating development effort. A list of such characteristics, known as resource predictors, can be see in Table 2.8 (see Section 2.2). These include measures of size and complexity as well as cost factors. In terms of our research we are using the Tukutuku dataset of Web project data covered in the previous section. The first 14 of 15 numerical Tukutuku variables listed in Table 3.1 will be used as inputs for our effort estimation problem.

With regard to Web effort estimation, the output y would naturally be the development effort required. As this is a real number, effort estimation is a regression problem. Development effort in person hours is the 15th numerical Tukutuku variable in Table 3.1.

A learner will therefore be given a training set consisting of Web project data from the Tukutuku dataset containing both the inputs, numerical measures of size, complexity and cost drivers, and the output, development effort required. With such a training set, the learner produces a classifier that can be used to estimate development effort when given inputs from new Web projects. Learners that have been previously used in Web effort estimation include case-based reasoning, stepwise and linear regression, and classification and regression trees. Table 2.4 in Section 2.2 provides a comprehensive list of these learners.

In terms of our research, we replicated the work done by Kocaguneli et al. [21], creating ensembles of classifiers using data from the Tukutuku dataset. Insufficient data is an issue within the Web effort estimation domain; given the statistical and representational reasons why ensembles work, we investigate whether ensembles also prove effective here. The learners used, as well as how their classifiers are combined, will be discussed in the next chapter.

3.3 Conclusion

This chapter provides the background information required for our research. We first discussed the Tukutuku dataset; a cross-company dataset of Web project data that we will use in our research.

The Tukutuku dataset was driven by an initiative to identify size metrics and cost factors to be used in effort estimation, that could be measured early in the development cycle. After two surveys and a case study involving Web companies from around the globe, a set of 25 variables was chosen. These variables are related to the characteristics of a Web project (measures of size and functionality), as well as its development process (cost drivers). Of these 25 variables only the 15 numerical variables will be utilized in our research, and a discussion of these variables and their summary statistics is provided.

As we will be using this Tukutuku data with ensembles of effort estimation techniques, we concluded this chapter with a discussion on the concept of ensembles. In order to achieve this we consider Web effort estimation as a supervised learning problem detailing how such a problem is approached. Given training data consisting of inputs x and outputs y, both of which are known, a learner produces a classifier C that can then use inputs from unseen examples to predict their corresponding outputs. In effort estimation terms, resource predictors like size measures are the inputs, and development effort required the output.

An ensemble combines the predictions made by several classifiers into a single prediction. Classifier predictions are combined via weighted or unweighted voting or averaging, depending on whether the learning problem is a classification one or a regression one (as is the case with effort estimation). For an ensemble to be effective, its constituent classifiers must be both accurate and diverse. Ensembles may be effective due to reasons that are statistical, representational and computational.

The next chapter will describe the replication process and its results and lay the groundwork for our subsequent research.

4 Using Ensembles For Web Effort Estimation: A Replication

As discussed in Chapter 2, despite the number of effort estimation techniques (learners) that have been investigated, there is no consensus as to which is the best. Accuracy results were found to vary widely depending not only on the learner used, but also on the dataset investigated and the validation procedure and performance measure/s utilized. One possible approach to this issue is to use estimates generated by ensembles of prediction techniques. This involves combining the predictions of a number of individual estimators. Kocaguneli et al. [21] demonstrated that a set of individual learners can be divided into a "minority" subset of "superior" learners, and a "majority" subset of "inferior" learners. They found that ensembles made from these superior learners outperformed all individual learners, and more importantly, maintained their performance over 20 datasets and several accuracy measures.

The study described in [21] was done in the domain of general software development. In Chapter 1, we discussed how this differs from Web development in a number of areas including the personnel involved in the development process, the intrinsic characteristics of the software, and the audience for which they are developed [40, 25]. Due to these differences it is not possible to assume that the findings of this previous study can be generalized to Web effort estimation. We have therefore decided to replicate this study to investigate whether such ensembles of learners would be as effective for Web effort estimation, which has not been done before. The remainder of this chapter is organized as follows: Section 4.1 provides details on the original study by Kocaguneli et al. [21]. Section 4.2 discusses our replication, and Section 4.3 details our results. Section 4.4 concludes the chapter with a discussion of our findings and comments on areas where we can extend and/or improve on the methodology utilized in the original study.

4.1 The Original Study

In order to provide context for our replication the following section will discuss the original study upon which our replication is based [9], focusing specifically on its research aim, methodology and findings.

4.1.1 Research Aim

When it comes to software effort estimation there is no consensus as to which of the numerous estimation techniques investigated is the best (i.e. most accurate). Kocaguneli et al. [21] aimed to demonstrate that a better option than looking for a single best learner, is to consider estimates generated by ensembles of multiple learners; specifically ensembles of "superior" learners as we will now discuss in more detail.

4.1.2 Methodology

The research methodology used can be summarized into the following three steps [21]:

- A large number of effort estimation techniques were run.
- These learners were evaluated and then sorted using a predefined set of performance measures.
- Ensembles were built using a subset of the best learners.

The effort estimation techniques investigated in [21] were categorized into "solo" learners, and ensembles of these solo learners referred to as "multimethods". A solo learner is run on a dataset by first pre-processing the data (referred to as the pre-processing option), followed by generating effort estimates on this data by running the learner using leave-one-out cross-validation. Ten pre-processing options and nine learners were used, giving a total of 90 solo learners. These learners were selected on the basis that they had been previously investigated in the software engineering effort estimation literature, and that they made different assumptions about the data. Table 4.1 provides a summary of these pre-processors and learners.

Table 4.1: List of pre-processing options and learners used.

Pre-processing Options	**Learners**
No pre-processor	Analogy with 1 nearest neighbour
Normalization	Analogy with 5 nearest neighbours
Natural logarithm	Stepwise regression
Principal component analysis	Classification and regression trees with pruning
Sequential forward selection	Classification and regression trees without pruning
Stepwise regression	Neural net with two hidden layers
Equal width discretization into 3 bins	Simple linear regression
Equal width discretization into 5 bins	Principal component regression
Equal frequency discretization into 3 bins	Partial least squares regression
Equal frequency discretization into 5 bins	

Ensembles combine the estimates provided by two or more solo learners into a single estimate [21]. Three simple combination schemes were utilized whereby a multimethod estimate was obtained by taking the mean, median, and inverse rank weighted mean (IRWM) of the estimates obtained from the solo learners that make up the ensemble. A discussion of how the IRWM is calculated is provided in Appendix G

The 90 solo learners were evaluated using a set of performance measures, obtained from error measures, that provide information on the quality of a prediction. The error measures utilized were the absolute residual (AR), the magnitude of relative error (MRE), the estimation magnitude of relative error (EMRE[1]), the balanced relative error (BRE) and the inverted balanced relative error (IBRE). Section 2.2.1 provides a discussion of these error measures apart from the latter two which have not seen prior use for Web effort estimation. Appendix C provides the mathematical basis for all of these error measures.

Distributions for these five error measures were compiled for each of the 90 solo learners during the evaluation process. From these five error distribution, seven performance measures were calculated: the mean absolute residual (MAR), mean and median MRE (MMRE and MdMRE), the percentage of estimates with an MRE of 25% or less (PRED(25)), the mean MER (MMER), the mean BRE (MBRE) and the mean IBRE (MIBRE) [21]. These performance measures were selected on the basis that they had been previously used in the literature on effort estimation. Table 4.2 provides a summary of the error distributions and their associated performance measures.

The performance of every solo learner was evaluated over 20 general software effort estimation datasets of variable size and feature count. All 20 datasets are publically available from the PROMISE repository [32], and a detailed table describing their characteristics can be found in [21]. The evaluation process was held as a round-robin, with each of the 90 solo learners being compared to the other 89, using the five error distributions and their seven associated performance measures, over all 20 datasets. As seen in

[1]Kocaguneli et al. [21] referred to this as the magnitude of error relative to the estimate or MER

Table 4.2: Error distributions and their associated performance measures.

Error Distribution	**Performance Measures**
AR	MAR
MRE	MMRE MdMRE PRED(25)
MER	MMER
BRE	MBRE
IBRE	MIBRE

Figure 4.1 the error distributions for two learners i and j were compared using a Wilcoxon nonparametric statistical test with 95 percent confidence. If for a particular error distribution E there is no statistical difference between E_i and E_j, then the tie counts for i and j are updated. If however, there is a significant difference between E_i and E_j, then the performance measures are compared and the learner with the "better" performance measure has its win count updated, and the other learner has its loss count updated. As the performance measures are based on error measures, a smaller value is considered "better" with the exception of PRED(25), where a higher percentage of estimates with an MRE of 25% or less is considered better. Thus at the end of the evaluation process the performance of every solo learner was summarized as "win-tie-loss" statistics [21].

The win, tie and loss counters were then used to identify the best solo learners, i.e., the ones with which to build the effort estimation ensembles. The 90 solo learners were first ranked according to their number of losses - from best (least number of losses) to worst (most number of losses). The solo learners were then also ranked according to their number of wins and number of wins−losses. These additional rankings were used to assess ranking instability by enabling the calculation of the maximal change in rank, δr, of a solo learner [21]. For example, if a solo learner was ranked third for losses, fifth for wins and seventh for wins−losses, then $\delta r = 4$.

```
if WILCOXON($E_i$, $E_j$, 95) is not significant
   $tie_i$ = $tie_i$ + 1
   $tie_j$ = $tie_j$ + 1
else
   if better($E_i$, $E_j$)
      $win_i$ = $win_i$ + 1
      $loss_j$ = $loss_j$ + 1
   else
      $win_j$ = $win_j$ + 1
      $loss_i$ = $loss_i$ + 1
```

Figure 4.1: Comparing learners i and j, where E_i and E_j are the respective error distributions.

It was found that the top 13 solo learners, as ranked by their number of losses, also had low δr values [21]. In addition, these solo learners had noticeably lower δr values than the solo learners that followed. These learners were therefore deemed "superior" and were used to build ensembles. A total of 12 ensembles were made using the top two, top four, top eight and top thirteen solo learners, with individual estimates being combined in one of three ways - mean, median and IRWM, as mentioned previously. The 12 ensembles were then ranked along with the solo learners using the same evaluation process.

4.1.3 Results

Three major findings were made when all 102 learners (90 solo learners + 12 ensembles) were evaluated:

- When ranked based on number of losses, it was found that the top ten learners were all ensembles. The remaining 2 ensembles fell within the top 20 learners.
- The top ranking ensemble (top 13 solo learners combined using IRWM) was found to have the smallest δr value (i.e. smallest ranking instability).
- Ensembles performed better than the solo learners with regards to the MdMRE performance measure.

4.2 Our Replication

We performed a dependent replication [39] with the goal of investigating whether the methodology used in [21] would result in similar findings when evaluated with Web project data. To this end we used:

- The same 90 solo learners.
- The same seven performance measures.
- The same round-robin evaluation using leave-one-out cross-validation.
- The same three combination schemes for building ensembles.

We were fortunate to be able to consult with the first author of [21] and be able to use the same code for the nine learners and most of the pre-processors. Whatever we had to implement was done with strict adherence to the methodology described in [21] with the author of the original study, once again, kindly clearing up any queries we had. All of our programming was done on the 64 bit Windows version of MATLAB R2012a.

In terms of Web project data we did not have access to 20 datasets, which was the number of datasets used by the original study. Instead we employed the Tukutuku dataset which we have discussed in detail in Chapter 3 using only its numerical variables.

Table 4.3: Ranking based on losses, of the top 16 and top 13 solo learners and their related δr values, as obtained by our replication study and the original study respectively

Our Replication				**Original Study**			
Rank	δr	Pre-processing option	Learner	Rank	δr	Pre-processing option	Learner
1	0	PCA	CART (yes)	1	8	Normalization	CART (yes)
2	0	PCA	CART (no)	2	6	Normalization	CART (no)
3	12	Stepwise regression	CART (yes)	3	6	None	CART (yes)
4	12	Stepwise regression	CART (no)	4	9	None	CART (no)
5	0	Natural logarithm	Analogy – 5NN	5	5	Natural logarithm	CART (yes)
6	2	Stepwise regression	Analogy – 5NN	6	4	Natural logarithm	CART (no)
7	6	Equal frequency – 5 bins	CART (yes)	7	5	Stepwise regression	CART (yes)
8	6	Equal frequency – 5 bins	CART (no)	8	6	Stepwise regression	CART (no)
9	6	Equal frequency – 5 bins	Analogy – 5NN	9	6	Sequential forward selection	CART (yes)
10	4	Stepwise regression	Analogy – 1NN	10	5	Sequential forward selection	CART (no)
11	5	None	CART (yes)	11	5	Stepwise regression	Analogy – 1NN
12	5	None	CART (no)	12	4	Natural logarithm	Analogy – 1NN
13	4	Normalization	CART (yes)	13	5	Stepwise regression	Analogy – 5NN
14	4	Normalization	CART (no)	**CART(yes)/CART(no)** – CART with or without pruning			
15	7	Natural logarithm	CART (yes)	**PCA** – Principal component Analysis			
16	7	Natural logarithm	CART (no)	**1NN/5NN** – 1 or 5 nearest neighbours			

4.2.1 Building our ensembles

As discussed previously, the original study identified the top 13 solo learners (as ranked by losses) as being superior and used them to build ensembles. These learners were characterised by low δr values such that even if these learners changed rank "by their maximum δr, then they would still be performing better than most of the other 90 methods" [21].

When we ran the 90 solo learners on the Tukutuku dataset, we found that we obtained a similar ranking of learners whether using wins, losses, or wins–losses. We however did not find a marked increase in δr values separating a set of highly ranked learners from the rest. In fact the δr values we obtained were generally lower than those obtained in [21]. We believe the reason for this is that we were only able to run these solo learners on a single dataset, Tukutuku. On the other hand in [21], these solo learners were run on 20 diverse datasets making ranking instability a greater issue.

In order to identify our set of superior solo learners with which to build ensembles, we decided to look for the smallest subset of solo learners that share the highest rankings whether using wins, losses, or wins−losses. This would be consistent with the methodology in [21], where the top solo learners would perform better than the other solo learners even if they changed rank by their maximum δr. In doing so we identified 16 solo learners that we deemed superior.

Table 4.3 contains a list of these superior solo learners, ranked in ascending order according to their number of losses, along with their δr values. We have also included the top 13 solo learners identified in [21] for the sake of comparison. It can be seen that the two sets of learners are not dissimilar: ten of the top 13 solo learners identified by the original study appear in our top 16 solo learners. Appendix H contains the rankings of all solo learners investigated.

We used these 16 solo learners to build 15 ensembles using the top two, top four, top

Table 4.4: Ranking based on losses of the top 17 learners; solo and ensemble.

Rank	**δr**	**Preprocessing option/ combination scheme**	**Learner**
1	9	Mean	Top 4
2	9	Median	Top 4
3	6	Mean	Top 8
4	4	IRWM	Top 4
5	1	IRWM	Top 8
6	2	IRWM	Top 12
7	0	Mean	Top 16
8	5	IRWM	Top 16
9	8	Median	Top 8
10	8	Mean	Top 12
11	6	Median	Top 12
12	5	Median	Top 16
13	1	Principal component analysis	CART(yes)
14	1	Principal component analysis	CART(no)
15	1	Mean	Top 2
16	1	Median	Top 2
17	1	IRWM	Top 2

eight, top twelve and top sixteen solo learners, with individual estimates obtained using the mean, median and IRWM combination schemes. As in [21], the 15 ensembles were then ranked along with the solo learners using the evaluation process already discussed.

4.3 Results

We will present our results in both tabular and graphical form. Table 4.4 focuses on the top 17 learners, solo and ensemble, ranked according to their number of losses. The complete list of rankings is provided in Appendix H. This represents the smallest subset of Web effort estimation techniques whose membership does not change regardless of type of ranking used (i.e. wins, losses, and wins−losses). In other words, these 17 learners outperform all other learners even when their rank is changed by their δr value.

Figure 4.2 graphs the δr values for all 105 learners (90 solo + 15 ensemble learners). The x-axis is used to display the ranking of the learners based on their number of losses. The y-axis is used to display their δr values. We will use the graph to see if there is a pattern to the δr values as we go from the top ranked learners to the bottom ranked learners. The dotted vertical line separates the δr values of the top 17 learners discussed in the previous paragraph with the rest of the learners.

From Table 4.4, it can be seen that 15 out of the top 17 learners are ensembles. This coincides with the findings made in [21] where the top 10 learners were also ensembles. Our findings however, do diverge from those in [21] in terms of δr values. They found

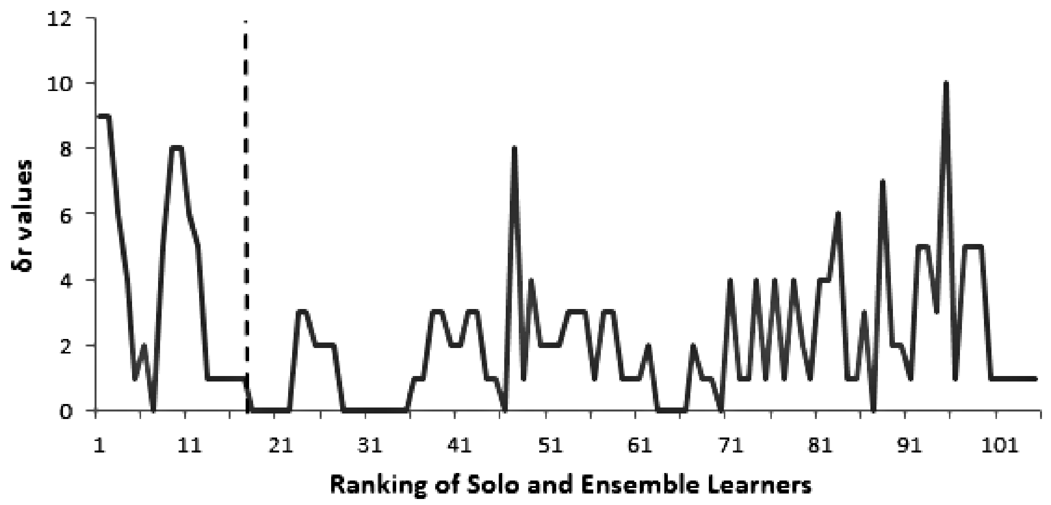


Figure 4.2: Graph showing the δr values for all learners, solo and ensemble. The dotted line separates the top 17 learners from the rest.

that the highest ranked ensemble, which was ranked first overall, had the lowest δr value of any learner. This is not the case with our highest ranked ensemble (which was also ranked first overall). We in fact did not find a pattern between δr values and learner rank, with our top 17 learners having some of the highest δr values as seen in Figure 4.2. This is not to say that these learners are not superior. As already pointed out, our top 17 learners (of which 15 are ensembles) outperform all other learners, even when taking into account these δr values. It is also important to note that as before, our δr values are generally lower than those seen in [21]. Once again, we believe that this is due to the fact that we have only used a single dataset as opposed to the 20 datasets used in [21], making ranking instability less of an issue for us.

We could analyze the performance of the learners in more detail by considering the wins, losses, and tie counts separately. Figure 4.3 graphs these values for all 105 learners. Note that the sum of wins, losses and ties for each technique is equal to 728: 104 comparisons x 7 performance measures. It is immediately obvious that there is a considerable difference in terms of win and loss values between the best learners (those with the highest rankings) and the worst learners (those with the lowest rankings). This gives credence to the ranking seen in Figure 4.2, and is also consistent with the findings made in [21].

Lastly, as in [21], we compared the MdMRE values of all learners. Figure 4.4 graphs the range of MdMRE values obtained by both ensembles and solo learners. It can be seen that ensembles generate lower MdMRE values than their solo counterparts, and more importantly, avoid producing the exceedingly large MdMRE values seen with some solo learners. These findings are consistent with those seen in [21].

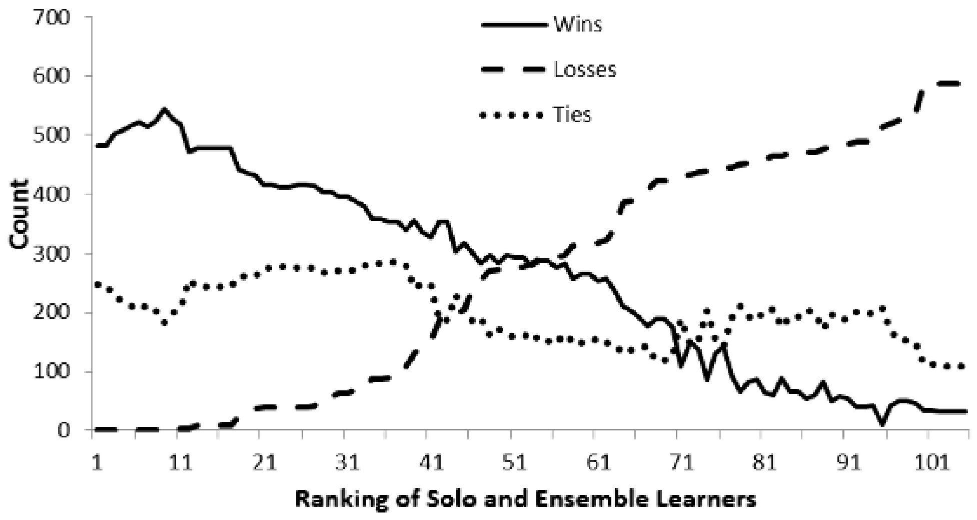


Figure 4.3: A graph of wins, losses, and ties for all learners. For each learner the sum of wins, losses and ties would be equal to 104 x 7 = 728.

4.4 What Next?

As seen in the previous section, the results obtained from using the methodology in [21] to create ensembles for Web effort estimation have been promising. We will now lay down the ground work for our subsequent research into this domain, by looking at how to improve and expand on the approach we have replicated.

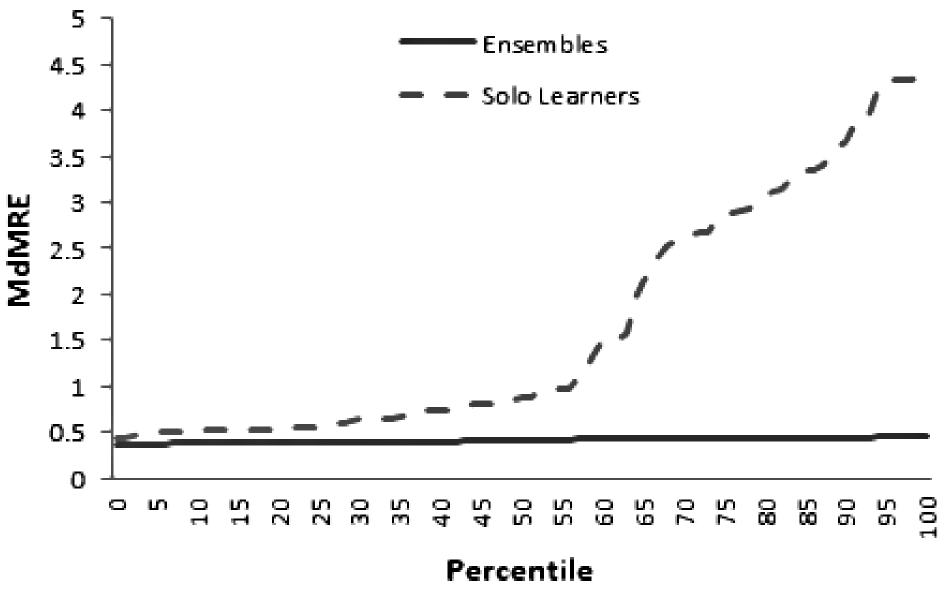


Figure 4.4: The range of MdMRE values obtained by the solo and ensemble learners.

4.4.1 Performance Measures

In Section 4.1.2 we have described the seven performance measures (and their five underlying error distributions) that were used to evaluate both the solo learners and the ensembles. Looking at Table 4.2 it can be seen that the MRE error distribution has three associated performance measures; all other error distributions have only one associated performance measure. This means that a significant difference in MRE distributions has a larger impact on the "win–tie–loss" counts than a significant difference involving the other error distributions.

In addition to this, with the error distributions that have only one associated performance measure, if there is a significant difference involving these error distributions then one learner will have their win count incremented while the other learner will correspondingly have their loss count incremented. With the MRE error distribution, as there are three performance measures, a significant difference can result in both the win and loss counts being incremented for the two learners being compared. For example, if learner i has a smaller MMRE, a larger MdMRE and a larger PRED(25) than learner j, then learner i will have its win count incremented by 2 (for the MMRE and PRED(25)) and its loss count by 1 (for the MdMRE). Conversely, learner j will have its win count incremented by 1 (for the MdMRE) and loss count incremented by 2 (for the MMRE and PRED(25)). In other words the MRE distribution not only has more weight than the other error distributions, but also its effects on the "win–tie–loss" counts are inconsistent in comparison as well.

To address this we decided to focus only on a single error distribution, the absolute residual. As discussed in Section 4.1.2 the AR forms the basis for all five of the error distributions used in the originally study, and therefore their associated performance measures. In Chapter 2, we also saw that the AR is the basis for all numerical performance measures used in evaluating Web effort estimation. In fact the AR is fundamental to any numerical performance measure used to evaluate development effort estimation [38].

Therefore by using the AR we are addressing the issues of weight and inconsistency described in the previous paragraph, focusing on the base error distribution for all those considered in [21], while remaining consistent with the state on the art in the domain of Web effort estimation.

4.4.2 Ranking

As described in Section 4.1.2 the "win–tie–loss" counts recorded during the round-robin evaluation process are used to rank the learners (both solo and ensemble) using three ranking systems: number of losses, number of wins, and number of wins−losses. The number of losses is used for the ranking and selection of solo learners with which to build

Table 4.5: The top 16 solo learners used in ensemble creation, their number of losses, their original ranking based on losses, and their new ranking.

Learners	**Losses**	**Rank$_{losses}$**	**Rank$'_{losses}$**
PCA and CART (yes)	8	1	1
PCA and CART (no)	8	2	1
Stepwise regression and CART (yes)	8	3	1
Stepwise regression and CART (no)	8	4	1
Natural logarithm and Analogy – 5NN	9	5	5
Stepwise regression and Analogy – 5NN	13	6	6
Equal frequency – 5 bins and CART (yes)	14	7	7
Equal frequency – 5 bins and CART (no)	14	8	7
Equal frequency – 5 bins and Analogy – 5NN	15	9	9
Stepwise regression and Analogy – 1NN	18	10	10
CART (yes)	29	11	11
CART (no)	29	12	11
Normalization and CART (yes)	29	13	11
Normalization and CART (no)	29	14	11
Natural logarithm and CART (yes)	29	15	11
Natural logarithm and CART (no)	29	16	11

ensembles; the other two ranking systems are used to calculate ranking instability. When these three ranking systems were used on the Web project data in Tukutuku it was noticed that groups of learners often had the same number of losses, wins or wins−losses.

The first three columns of Table 4.5 show the top 16 solo learners listed in Table 4.3 along with their number of losses and their rankings based on their number of losses. It can be seen, for example, that the first four solo learners are ranked from 1 to 4 despite having the same number of losses (8). While this ranking may not affect building ensembles using the mean and median combination schemes, it does play a significant role when using the IRWM combination scheme as the higher the rank a solo learner has, the more weight its estimate has when used in calculating the ensemble estimate. Therefore the four learners in question would have their estimates weighted differently when calulating an ensemble estimate despite all of them having the same performance in terms of losses.

To rectify this we have decided to assign the same rank to all learners with the same performance. The fourth column of Table 4.5 shows how this new ranking is applied. Naturally the other two ranking systems (wins and wins−losses) are updated in an identical fashion.

Using this new ranking system necessitates changing how the IRWM is calculated. With the original equation described in [21], n learners ranked $r_1, r_2, \dots, r_n$ will be assigned weights from 1 to n. The highest and lowest ranked learners will get weights of n and 1 respectively. These weights $w_1, w_2, \dots, w_n$ are then used along with the learners' corresponding estimates $e_1, e_2, \dots, e_n$ to calculate the ensemble estimate. The equation

Table 4.6: The original (r) and new rankings (r') along with the original (w) and new IRWM weights (w') the top 16 learners would receive, if used to build an ensemble.

Learners	r_i	r'_i	w_i	w'_i
PCA and CART (yes)	1	1	16	14.5
PCA and CART (no)	2	1	15	14.5
Stepwise regression and CART (yes)	3	1	14	14.5
Stepwise regression and CART (no)	4	1	13	14.5
Natural logarithm and Analogy – 5NN	5	5	12	12
Stepwise regression and Analogy – 5NN	6	6	11	11
Equal frequency – 5 bins and CART (yes)	7	7	10	9.5
Equal frequency – 5 bins and CART (no)	8	7	9	9.5
Equal frequency – 5 bins and Analogy – 5NN	9	9	8	8
Stepwise regression and Analogy – 1NN	10	10	7	7
CART (yes)	11	11	6	3.5
CART (no)	12	11	5	3.5
Normalization and CART (yes)	13	11	4	3.5
Normalization and CART (no)	14	11	3	3.5
Natural logarithm and CART (yes)	15	11	2	3.5
Natural logarithm and CART (no)	16	11	1	3.5

can be summarized as:

$$\frac{\sum_{i=1}^{n} w_i e_i}{\sum_{i=1}^{n} i} \tag{4.1}$$

where $w_i = n + 1 - r_i$.

With the new ranking system the only part of the original equation that needs to be changed is the weight learners with the same number of losses (wins or wins$-$losses) get allocated. The weights are first calculated as before using $w_i = n+1-r_i$, and then for all learners with the same number of losses (wins or wins$-$losses), these weights are averaged so that they receive the same weight. An example of the weights if all 16 learners are used to build an ensemble using the original and new methodology are given in Table 4.6. For example it can be seen that the updated weights for the four learners jointly ranked first is calculated as the average of their original weights: $(16 + 15 + 14 + 13)/4 = 14.5$. It can also be seen that the sum of all the weights has not changed; the sum of weights in the w_i and w'_i column is 136. As the denominator of the IRWM equation is the sum of weights, our updated IRWM equation needs no further changes.

4.4.3 Building Ensembles

In [21], 13 solo learners were identified as being "superior", and a selection of the top $M \in \{2, 4, 8, 13\}$ of these solo learners was used to build ensembles. Similarly, in our replication we identified 16 solo learners as being "superior" and built ensembles from the top $M \in \{2, 4, 8, 12, 16\}$ solo learners. Instead of using an arbitrary number of solo learners to build ensembles, the new ranking system described in Subsection 4.4.2 can be used to identify which solo learners to select in the ensemble building process.

As discussed previously, solo learners are ranked according to their number of losses and the top solo learners are used to build ensembles. Solo learners with the same number of losses receive the same rank. Looking at the new rankings of the top 16 solo learners in Table 4.5, it can be seen that there are 7 distinct loss rankings: 1, 5, 6, 7, 9, 10, and 11. We could therefore build ensembles with the top ranked solo learners (i.e. those ranked 1), the top two ranked solo learners (i.e. those ranked 1 and 5), the top three ranked solo learners (i.e. those ranked 1, 5, and 6), the top four ranked solo learners (i.e. those ranked 1, 5, 6, and 7) and so on. The decision as to which top solo learners to build ensembles with is now solely based on experimental results.

With this replication, as with the original study, only the top solo learners are being considered for use in building ensembles; solo learners that outperform all other solo learners when taking into account their maximal change in rank, δr, across the three ranking systems used (wins, losses, and wins−losses). As detailed in Chapter 3 this is only half the equation for building effective ensembles; in addition to being accurate, the learners used to build ensembles need to be diverse too. By only considering solo learners that outperform all other learners, we may be overlooking additional solo learners that whilst not as accurate as the top solo learners should still be considered when building ensembles, due to the diversity they add.

To this end we decided to change the criteria with which solo learners are selected; instead of considering only the solo learners that outperformed all other solo learners, we decided to consider all solo learners that win more comparisons, in the round-robin process, than they lose. In other words solo learners whose wins−losses count is greater than 0. This criterion will be used along with the new ranking system to more thoroughly evaluate ensembles for Web effort estimation by considering more solo learners, and selecting groups of them for ensemble creation on the basis of experimental results instead of an arbitrary criteria.

4.4.4 Bagging

The original study [21] uses leave-one-out cross-validation as the sampling method to train and evaluate the learners investigated: Given project data on n projects, a learner is trained on data from $n-1$ projects (the training set), with the data from the remaining project being used to evaluate the learner (the test case). Leave-one-out cross-validation is a deterministic process with each learner using the exact same set of training data for a particular test case. Whilst this may be useful in terms of experimental reproducibility, it does mean that the different learners used are correlated through the training data. This correlation is compounded by the fact that there is a large amount of overlap between the training sets when the Tukutuku dataset is used.

As discussed in Chapter 3, Tukutuku has Web project data on 195 projects. Each project would thus be used to evaluate a learner trained on data from the remaining 194 projects. The different training sets would vary by data on only 1 project; the project being used as a test case. In other words the training sets would share 193 out of 194 datapoints (i.e. 99.5% of the data).

To address this issue we have decided to use a sampling method known as bootstrap aggregation or bagging [6, 12]. Using a training set of size n, bagging typically generates another training set by drawing n items from the original training set with replacement. Each training set that is generated by bagging is called a bootstrap replicate and contains on average 63.2% of the original training set (the rest of the training set are duplicates due to sampling with replacement). In this way we reduce the amount of overlap and hence correlation between training sets.

The fact that each training set on average only contains 63.2% of unique data could possibly be an issue in that we are effectively reducing the training set size. As the size of a training set decreases, it is less likely to contain training examples that characterize the underlying target effort estimation model, causing estimation accuracy to suffer [20]. As a result we will also consider bagging where the training sets generated contain $2n$ items drawn from the original training set with replacement. In doing so we will include a larger percentage of the original training set hopefully improving estimation accuracy, albeit at the expense of some diversity.

Lastly bagging allows us to generate multiple different training sets from a single dataset. This allows us to perform multiple experimental runs with the solo and ensemble learners using just the Tukutuku dataset, providing us a closer look at their estimation performance.

4.5 Conclusion

This chapter investigated the effectiveness of using ensembles for Web effort estimation. We replicated the methodology described in [21] which successfully used ensembles for effort estimation in the domain of general software development. The results obtained when using this approach on Web project data from the Tukutuku dataset has been promising. We found that the 15 ensembles created using this approach outperformed all but two of the 90 solo learners when taking into account their maximal change in rank, δr, across the three ranking systems used (wins, losses, and wins−losses). In other words they have performed consistently well. We have also shown that these 15 ensembles made smaller errors than their solo counterparts, avoiding the very large errors seen with some of the solo learners.

We have also identified areas where we can build on the approach used in [21] to further evaluate the use of ensembles for Web effort estimation:

- Using the absolute residual (and its associated performance measure MAR) as the only error distribution for evaluating solo learners and ensembles.
- Taking into consideration learners that perform equally well for a particular ranking system.
- Being more inclusive when selecting solo learners with which to build ensembles so as not to ignore the importance of diversity.
- Using bagging as the sampling method to address the correlation between training sets when only leave-one-out cross-validation is used, and to allow multiple experimental runs to be made with a single dataset.

The evaluation of our updated ensemble methodology will be discussed in the next chapter, where we will look at using bagging with Web effort estimation ensembles.

5
Using Bagging With Ensembles For Web Effort Estimation

In the previous chapter we discussed our replication of the work done by Kocaguneli et al. [21], using Web project data in the Tukutuku dataset. We found that ensembles consistently outperformed all but two of the solo learners investigated for Web effort estimation. Following this replication we identified aspects of the methodology that we could expand upon, including the use of bagging when creating Web effort estimation ensembles. The following chapter details this research along with the results, followed by a discussion of our findings.

5.1 Methodology

Section 4.4 describes the areas of the replication that we felt could be expanded on. In summary:

1. Only the absolute residual (AR) error distribution and its associated performance measure the mean AR, will be used for evaluating classifiers obtained from solo learners and ensembles.

2. The ranking system will be updated so that learners with identical performances for a particular ranking system, will receive the same ranking. The IRWM combination

scheme for building ensembles will be adjusted accordingly.

3. The criteria for selecting learners with which to build ensembles will be altered so that more learners are considered. This is done to increase the diversity of learners used to build ensembles.

4. Bagging will be used, alongside leave-one-out cross-validation, when creating training sets to be utilized by learners to produce classifiers. Our bagging methodology will be discussed in Section 5.1.1.

Apart from the above, every other experimental aspect of the replication will remain unchanged. In other words:

- The same set of 90 learners will be evaluated and their classifiers used for building ensembles. These are listed in Table 4.1.

- Ensembles will be built in the same manner using the mean, median and IRWM combination schemes.

- The same Web project data from the Tukutuku dataset will be used. As before, only the numerical Tukutuku variables will be considered and these are listed in Table 3.1.

- We will use the same three ranking schemes for evaluating learners both solo and ensemble: Losses, Wins, and Wins−Losses.

5.1.1 Bagging Methodology

As with the original study [21] our replication utilized leave-one-out cross-validation to train and evaluate the learners investigated: given project data on n projects, a learner is trained on data from $n-1$ projects (the training set), with the data from the remaining project being used to evaluate the resulting classifier (the test case). Each learner will therefore use the exact same set of training data as every other learner, to generate a classifier for a particular test case. Bagging, or bootstrap aggregation, enables creating different training sets by sampling with replacement, data from the original training set [6, 12].

Typically a training set generated by bagging (a "bag") is of the same size as the original training set and contains on average 63.2% of the data (the rest of the training set are duplicates due to sampling with replacement) [6, 12]. As discussed in Section 4.4, this reduces the amount of correlation between training sets. However this also effectively reduces training set size. When the size of a training set decreases, it is less likely to contain training examples that characterize the underlying target effort estimation function, causing estimation accuracy to suffer [20]. In other words bagging could have an adverse impact on the accuracy of our solo classifiers, which could be an issue as effective ensembles require their component classifiers to be accurate.

As such we investigated bagging using different sized bags. Given a training set of size N obtained via leave-one-out cross-validation, we investigated bagging with bags of size N and $2N$. *Bagging N* is the typical form of bagging, generating a bag of the same size as the original training set. *Bagging 2N* on the other hand creates bags that are twice the size of the original training set. Such a bag would include more of the original training data. Two versions of *Bagging 2N* were evaluated:

1. *Bagging* $2N_1$: The entire contents of the $2N$ bag are drawn with replacement from the original training set.

2. *Bagging* $2N_2$: The original training set is used in its entirety along with N samples drawn with replacement from the original training set.

The effect this has on Web effort ensembles was evaluated and the results discussed in the following section. As a control we also evaluated ensembles built without bagging using just leave-one-out cross-validation, but including the improvements we have disccused previously in Section 5.1.

5.2 Results

The following section will detail the results of our experiment using bagging and ensembles for Web effort estimation. We will first start with a look at the results obtained without bagging (i.e. the control), before moving on to the three variants of bagging we are investigating: *Bagging N*, *Bagging* $2N_1$ and *Bagging* $2N_2$.

5.2.1 The Control

For the control we only utilized leave-one-out cross-validation when creating training sets to be used by learners to produce classifiers. As such we used the same set of results (i.e. effort estimates) obtained from our replication discussed in Chapter 4. We however implemented the other improvements listed in the previous section, namely only using the AR when comparing classifiers, the updated ranking system, and the more inclusive criteria for selecting classifiers with which to build ensembles.

Table 5.1 lists the solo learners whose classifers were ranked 1st, 16th, 17th, 19th and 22nd; in other words the top five ranks as based on their number of losses, using the AR for comparison. As with our replication (see Table 4.3), the top ranked solo learners are

Table 5.1: The top five ranked solo learners, based on number of losses, and their related δr values.

Rank	δr	Pre-processing option	Learner
1	0	Stepwise regression	Analogy – 5NN
	1	Natural logarithm	Analogy – 5NN
	2	None	Analogy – 5NN
	2	Normalization	Analogy – 5NN
	4	None	CART (yes)
	4	None	CART (no)
	4	Normalization	CART (yes)
	4	Normalization	CART (no)
	4	Natural logarithm	CART (yes)
	4	Natural logarithm	CART (no)
	4	Sequential forward selection	Analogy – 1NN
	4	Sequential forward selection	Analogy – 5NN
	4	Stepwise regression	Analogy – 1NN
	17	None	Analogy – 1NN
	17	Normalization	Analogy – 1NN
16	7	PCA	Analogy – 5NN
17	7	Stepwise regression	CART (yes)
	7	Stepwise regression	CART (no)
19	14	PCA	CART (yes)
	14	PCA	CART (no)
	15	PCA	Analogy – 1NN
22	12	Sequential forward selection	CART (yes)

variants of either analogy-based learners (with one or five nearest neighbours), or CART (both pruned and unpruned).

There are some differences: A total of 13 of the 16 learners identified by our replication are found in Table 5.1, albeit with different rankings. The remaining three learners, analogy with five nearest neighbours, CART (both pruned and unpruned), all using the equal frequency discretizer (five bins) pre-processor, are ranked 23rd and 24th, falling just outside the top five rankings. These differences are due to the changes we have implemented when running our control.

The classifiers produced by these learners as well as those that had more wins than losses (i.e. a positive wins−losses value) were then used to build ensembles. A total of 49 solo learners fit this criteria, corresponding to the top 21 ranks based on their number of losses. The complete list of solo learners is provided in Appendix I. In all 17 ensembles were built:

- Mean and Median ensembles for the top ranked classifiers[1].
- Mean, Median, and IRWM ensembles for the top 2, top 3, top 4, and top 5 ranked classifiers.

[1]In other words all the solo classifiers ranked first. IRWM is not used as it is identical to the mean when all rankings are the same.

Table 5.2: Ranking based on losses of the top five ranked learners, solo and ensembles, and their related δr values. Ensembles have been shaded grey.

Rank	δ**r**	**Preprocessing option/ combination scheme**	**Learner**
1	0	Mean	Top
	0	Mean	Top 2
	0	IRWM	Top 2
	0	Mean	Top 3
	0	IRWM	Top 3
	0	IRWM	Top 4
	0	IRWM	Top 5
	0	Median	Top Positive
	8	Median	Top
	12	Median	Top 3
	12	Mean	Top 4
	15	Natural Logarithm	Analogy – 5NN
	15	Mean	Top 5
	18	None	Analogy – 5NN
	18	Normalization	Analogy – 5NN
	20	None	CART (yes)
	20	None	CART (no)
	20	Normalization	CART (yes)
	20	Normalization	CART (no)
	20	Natural logarithm	CART (yes)
	20	Natural logarithm	CART (no)
	20	Sequential forward selection	Analogy – 1NN
	20	Sequential forward selection	Analogy – 5NN
	20	Stepwise regression	Analogy – 1NN
	20	IRWM	Top Positive
	34	None	Analogy – 1NN
	34	Normalization	Analogy – 1NN
28	15	Stepwise regression	Analogy – 5NN
	19	Median	Top 2
	19	Median	Top 4
	19	Median	Top 5
32	8	PCA	Analogy – 5NN
33	8	Stepwise regression	CART (yes)
	8	Stepwise regression	CART (no)
35	14	PCA	CART (yes)
	14	PCA	CART (no)

- Mean, Median, and IRWM ensembles for the classifiers with more wins than losses (termed the *top positive* classifiers).

Table 5.2 presents the rankings (in terms of losses) of these ensembles and solo learners, when their resulting classifiers were compared using the AR. Only the top five ranks (i.e. 1, 28, 32, 33 and 35) are shown here, with the complete list of rankings provided in Appendix I.

Looking at Table 5.2 it can be seen that regardless of combination scheme used the ensembles created perform well, with all but one of the 17 ensembles created found in the top two ranks. The only ensemble not present here, the mean top positive ensemble, is ranked 37th, and is therefore found in the top 6 rankings.

We believe that the mean top positive ensemble is the lowest ranked ensemble because out of the three combination schemes used, the mean is the most sensitive measure of central tendency. Since the mean is calculated as the sum of all observations (in this scenario effort estimates) divided by the number of observations, it is susceptible to observations that are outliers. Given that the top positive ensembles are composed of a much larger range of classifiers than their top – top 5 counterparts, the likelihood that one or more of the component classifiers makes an uncharacteristically inaccurate estimate is increased. By giving lower ranked classifiers a smaller weight, the IRWM combination scheme is more robust than the mean in this situation as is the median, which is simply the value that half of the observations fall above or below. This would explain why both the IRWM and median top positive ensembles are still ranked highly.

It can also be seen that of the 27 classifiers found in the top rank, there is an almost even split between ensembles (13) and solo techniques (14). This signifies that there are classifiers produced by solo learners that perform comparably with those produced by the best ensembles. However these top ranked ensembles do tend to be more stable with regard to their ranking, as indicated by their δr value. The first eight top ranked ensembles in Table 5.2 have a δr value of 0. This means that these ensembles were top ranked regardless of ranking system used; losses, wins, or wins−losses. In fact 12 of the 13 top ranked ensembles have a $\delta r \leq 15$, which is the smallest δr value of a top ranked classifier obtained from a solo learner (natural logarithm pre-processor coupled with an analogy-based learner using five nearest neighbours). This is also illustrated in Figure 5.1 where all but one of the learners with a $\delta r \leq 15$ (as marked by the dashed line) are ensembles.

The δr values seen in Figure 5.1 are higher than those seen in our replication despite the same dataset being used. This is due to the new ranking system implemented, where depending on the number of learners with the same ranking, consecutive ranks can be numerically quite different. For example, because there are 27 learners that are top ranked with regards to losses, the top two groups of learners are ranked 1st and 28th respectively.

We can further analyse the performance of the classifiers (from both solo learners and ensembles) by considering their wins, losses, and tie counts separately. Figure 5.2 shows these counts for the control. Note that this time the sum of wins, losses, and ties for every classifier is equal to 106, as each of the 107 classifiers is compared to the other 106 using only the absolute residual and its associated performance measure the mean AR. As with our replication (see Figure 4.3) it can be seen that there is a considerable

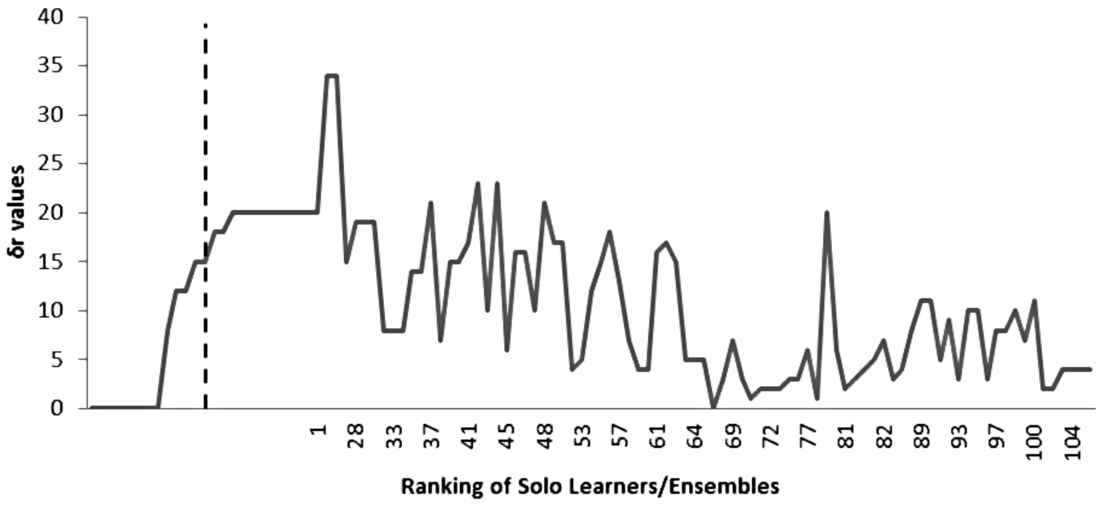


Figure 5.1: A graph showing the δr values of classifiers produced by solo learners and ensembles. The dashed line separates top ranked classifiers with a $\delta\text{r} \leq 15$ from the rest.

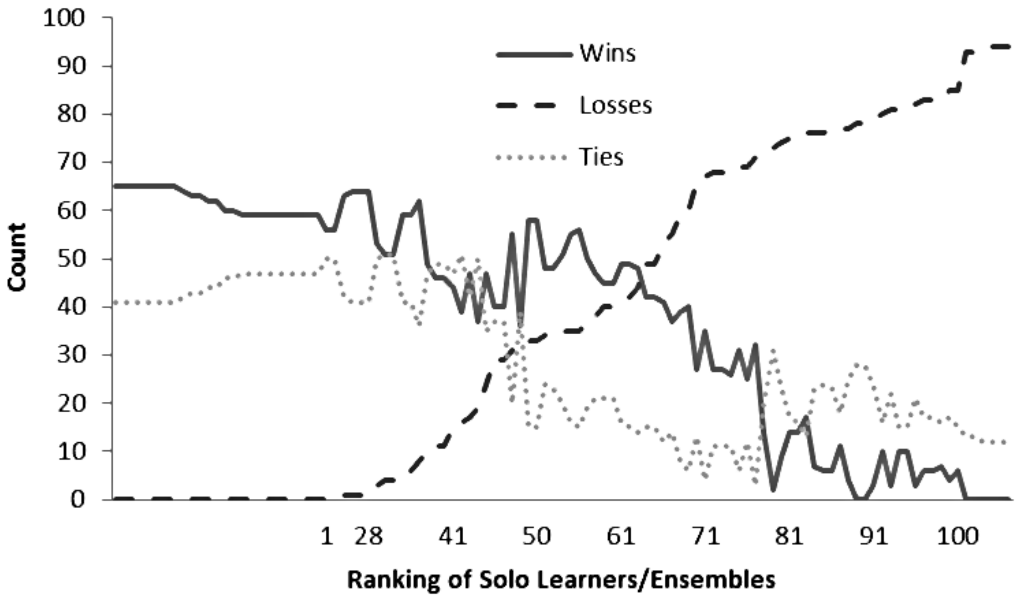


Figure 5.2: A graph of wins, losses, and ties for all estimation techniques as evaluated in the control.

difference in terms of wins and losses, for the classifiers produced by the best (highest ranking) techniques and the worst (lowest ranking) techniques. This provides credence to the rankings seen in Figure 5.1.

In our replication the range of MdMRE values obtained from ensembles and solo learners was compared. As we are now only using the AR for comparison, we will instead compare the mean AR values obtained from ensembles and solo learners, as displayed in Figure 5.3. It can be seen that classifiers from ensembles generate lower mean AR values than classifiers obtained from solo learners. In addition to this, none of the ensembles produce exceedingly large mean AR values as seen with some of the solo techniques. This

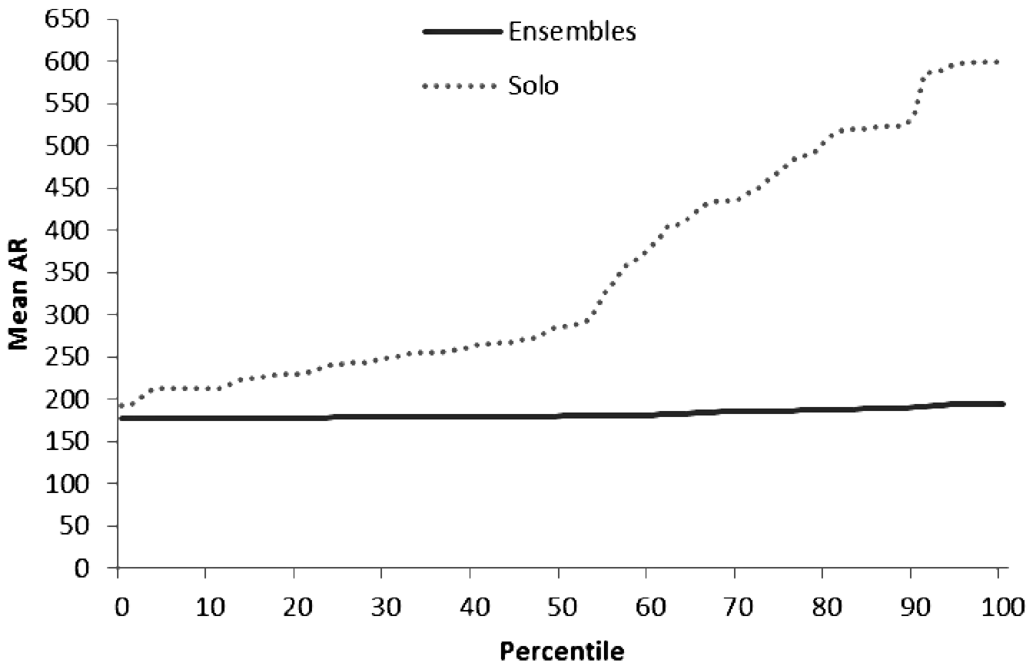


Figure 5.3: The range of mean AR values obtained by ensemble classifiers versus their solo counterparts.

also corresponds to our replication results (see Figure 4.4).

In summary, as with our replication, the ensembles created using our updated methodology produce classifiers that perform favourably in comparison to their solo counterparts. Regardless of the combination scheme used, all but one ensemble is found in the top two ranks; the only exception is the mean top positive ensemble, and this is probably due to the mean combination scheme being more sensitive to outliers. While there is an almost even split of ensembles and solo learners in the top rank, the ensemble learners are more stable since they have smaller δr values. When considering the mean AR performance measure, we can see that ensemble learners generate lower mean AR values than their solo counterparts, whilst avoiding the very large values seen with less accurate solo learners.

The changes in our methodology have resulted in some differences between the control results and those of the replication: Classifiers obtained from solo learners do have slightly different rankings, and the overall δr values recorded for both solo and ensemble learners, are generally higher than they were in the replication. This being the case, it is worth pointing out that whereas the top ranked ensembles in the replication sported some of the largest δr values observed, the top ranked ensembles in the control had some of the lowest δr values; this can be seen when comparing Figures 4.2 and 5.1.

With these results we can verify that the improvements made to our methodology, do not invalidate our previous findings on using ensembles for Web effort estimation. Despite the differences discussed previously, our findings still show that ensembles produce

Table 5.3: Ranking based on losses for solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging N*.

Pre-processing Option	**Learner**	**Average Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stepwise regression	Analogy – 1NN	1.5	1	1	1	1	1	3	4	1	1	1
None	Analogy – 1NN	3.3	4	1	1	1	1	3	10	4	7	1
Normalization	Analogy – 1NN	3.3	4	1	1	1	1	3	10	4	7	1
Stepwise regression	Analogy – 5NN	4.8	8	1	7	5	1	3	1	1	1	20
None	Analogy – 5NN	5.8	18	1	1	13	1	8	1	13	1	1
Normalization	Analogy – 5NN	5.8	18	1	1	13	1	8	1	13	1	1
Sequential forward selection	Analogy – 5NN	7.4	33	1	7	13	1	3	10	4	1	1
Sequential forward selection	Analogy – 1NN	8	1	1	10	5	1	1	21	1	15	24
PCA	Analogy – 1NN	10	1	10	7	1	19	22	17	15	7	1
PCA	Analogy – 5NN	14.8	13	17	18	24	20	11	27	7	10	1
Sequential forward selection	CART (no)	15.8	9	32	1	10	16	30	19	15	10	16
Natural logarithm	Analogy – 5NN	16.4	4	1	10	24	23	11	28	23	6	34
Sequential forward selection	Linear regression	16.6	4	27	10	10	22	8	22	8	21	34
PCA	CART (no)	16.9	13	10	27	13	24	11	24	15	12	20
PCA	CART (yes)	16.9	13	10	27	13	24	11	24	15	12	20
Stepwise regression	CART (no)	17.6	39	10	16	24	1	32	4	8	24	18
Stepwise regression	CART (yes)	17.6	39	10	16	24	1	32	4	8	24	18
Sequential forward selection	CART (yes)	17.9	27	15	14	28	11	29	4	15	21	15
None	CART (no)	18	33	19	19	13	12	22	10	23	28	1
None	CART (yes)	18	33	19	19	13	12	22	10	23	28	1
Normalization	CART (no)	18.7	33	19	21	13	12	20	10	23	35	1
Normalization	CART (yes)	18.7	33	19	21	13	12	20	10	23	35	1
Natural logarithm	CART (no)	18.7	39	19	21	13	17	22	4	23	28	1
Natural logarithm	CART (yes)	18.7	39	19	21	13	17	22	4	23	28	1

classifiers that performed favourably in comparison to their solo counterparts.

In the following sections we will look at the results obtained when bagging was introduced to the ensemble creation process. Unlike the replication and the control, bagging enabled us to perform multiple experimental runs using the Tukutuku dataset; 10 runs for each of the three variants of bagging investigated. This provided us with a more in depth look at using ensembles for Web effort estimation.

5.2.2 Bagging N

We will now look at Web effort estimation ensembles using bagging with leave-one-out cross-validation. *Bagging N* generates a bag of the same size as a regular training set; in this case $N = 194$. Using different bags with the same learner results in different classifiers being produced. As training sets are no longer created deterministically, using bagging enables us to perform multiple runs on the same set of data (i.e. Tukutuku). We performed 10 runs for all of our bagging experiments to obtain a clearer picture of learner performance, both solo and ensemble. To take into account the fact that with bagging a single learner may produce multiple classifiers, when discussing estimation performance we will use the term classifier instead of learner in the sections that follow.

Table 5.3 lists the rankings, for losses, of classifiers obtained from solo learners. Rankings are provided for all 10 runs: Only classifiers with an average ranking ≤ 20 over the 10 runs are shown; a complete list of rankings for losses, wins, and wins−losses is provided in Appendix J.

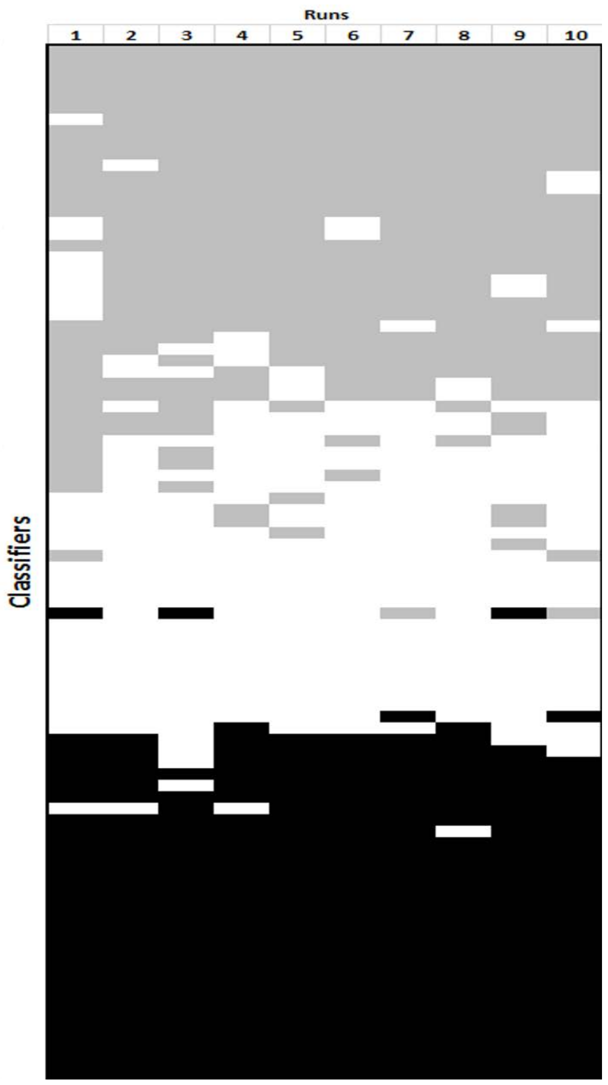


Figure 5.4: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average ranking for losses. Gray indicates a classifier was ranked in the top third for a run, white that it was ranked in the middle third, and black in the bottom third.

Similar to the results obtained for the control (as shown in Table 5.1), variants of analogy-based learners (with one or five nearest neighbours) and CART (pruned and unpruned) produce the top ranking classifiers. The only exception here is the classifier obtained using linear regression as the learner, with sequential forward feature selection as the pre-processer.

Figure 5.4 provides a graphical visualization of how the classifiers change their losses ranking over 10 runs. Classifier performance has been colour coded based on their ranking: Classifiers ranked in the top third (i.e. ranks one to 30) in a run are coloured gray. Classifiers ranked in the middle third (i.e. ranks 31 to 60) are coloured white. Classifiers ranked in the bottom third (i.e. ranks 61 to 90) are coloured black. A summary of changes in rank can then be ascertained quickly without having to parse through 900 numbers

Table 5.4: Learners whose classifiers are ranked in the top third over all 10 runs and all three ranking systems of *Bagging N*.

Pre-processing Option	Learner
Stepwise regression	Analogy – 1NN
None	Analogy – 1NN
Normalization	Analogy – 1NN
Stepwise regression	Analogy – 5NN
None	Analogy – 5NN
Normalization	Analogy – 5NN

(90 classifiers × 10 runs). The classifiers are arranged from top to bottom in order of increasing average rank as seen in Table 5.3. Similar graphical visualizations for the other two ranking systems (wins and wins−losses) are provided in Appendix K.

Looking at Figure 5.4 we can see that there is a definite ordering of classifier performance over the 10 runs. The classifiers that perform the best in the top third remain in the top third for all 10 runs. This holds true at the other end of the spectrum, with the worst performing classifiers found in the bottom third remaining in the bottom third for all 10 runs. It is only at the boundaries between the top third and middle third, and the middle third and bottom third, that we frequently see classifiers whose rankings vary sufficiently to drop their ranking from one third to another (as signified by a change in colour). Even then change in ranking is normally restricted to adjacent thirds; a classifier can drop in ranking from the top third to the middle third or from the middle third to the bottom third and vice-versa.

Only a single classifier has performances that change over all three thirds: partial least squares regression with a sequential forward feature selection pre-processor. We believe that this is due to the implementation of sequential forward feature selection. In order to evaluate potential subsets of features (numerical Tukutuku variables), 10 fold cross-validation is performed on the training set. As the partitioning done during the cross-validation is random, this means that even with identical training sets, different runs of sequential forward feature selection can result in a different subset of features being selected. This adds an additional level of instability on top of that introduced by the change in training sets due to bagging, on a learner whose performances hover near the boundary of the middle third and bottom third.

As we did with the control, we built ensembles using the top, top 2, top 3, top 4, top 5 and top positive groups of solo classifiers for each run. As each run has its own set of top ranking classifiers, the component classifiers of the various ensembles built will be different from run to run. In order to investigate how ensembles built from the same set of classifiers would perform over all 10 runs, we decided to use a set of learners whose classifiers are in the top third of rankings for all 10 runs, and for all three ranking systems.

Table 5.5: Ranking based on losses for ensembles and solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging N*.

Pre-processing Option/ Combination Scheme	**Learner**	**Average Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 4	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 5	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 3	2.1	1	1	1	1	1	1	1	12	1	1
Median	Top Positive	2.3	1	1	1	1	1	1	14	1	1	1
IRWM	Top 2	3.4	1	1	1	1	1	1	14	12	1	1
Mean	Top Overall	4	1	19	1	1	1	13	1	1	1	1
Mean	Top 2	4.6	1	1	1	1	1	13	14	12	1	1
Mean	Top	5.2	1	1	1	19	1	1	14	12	1	1
Median	Top	5.8	1	19	20	1	1	1	1	12	1	1
IRWM	Top Positive	8.1	1	1	1	1	23	24	1	1	27	1
IRWM	Top Overall	8.5	1	19	1	1	1	13	14	12	1	22
Median	Top 5	9.8	1	19	24	1	23	13	14	1	1	1
Median	Top 3	9.8	19	19	20	1	23	1	1	12	1	1
None	Analogy – 1NN	10.2	19	1	1	1	1	13	28	12	25	1
Normalization	Analogy – 1NN	10.2	19	1	1	1	1	13	28	12	25	1
Median	Top 4	10.5	1	19	20	1	23	13	14	12	1	1
Median	Top 2	12.6	19	1	24	19	1	13	14	12	1	22
Stepwise regression	Analogy – 1NN	14.6	1	19	1	19	1	24	28	12	19	22
Median	Top Overall	15.1	19	19	20	19	23	13	14	1	1	22
Sequential forward selection	Analogy – 5NN	16.3	37	1	24	27	1	13	28	12	19	1
None	Analogy – 5NN	16.6	31	1	1	36	1	24	1	30	19	22
Normalization	Analogy – 5NN	16.6	31	1	1	36	1	24	1	30	19	22

There are six learners that fit this criteria, all variants of analogy-based learners, and they are listed in Table 5.4. The ensembles built using their classifiers are referred to as the *top overall* ensembles, since the component classifiers are ranked in the top third **over all** runs and ranking systems. The addition of mean, median and IRWM top overall ensembles means that 20 ensembles were created for each run.

Table 5.5 lists the rankings, for losses, of classifiers obtained from solo learners and ensembles. As before, the rankings are provided for all 10 runs with only classifiers with an average ranking $\leq$ 20 over the 10 runs being shown. A complete list of rankings for losses, wins, and wins$-$losses is provided in Appendix J.

It can be seen that 19 of the 20 ensembles created appear in Table 5.5. In fact the first 16 classifiers with the lowest average losses ranking are from ensembles, and of those four of them have an average rank of one; in other words they were ranked first in all 10 runs. The only ensemble not present is the mean top positive ensemble that has an average losses ranking of 30.6. As discussed in the control, we believe that this is due to the mean being susceptible to effort estimates that are uncharacteristically inaccurate; a situation that is more likely to occur with the wider range of classifier performances incorporated when building the top positive ensembles. The median and IRWM top positive ensembles still perform well as these two combination schemes are less sensitive to outliers.

Unsurprisingly, the top overall ensembles, those that use the same set of consistently top ranking solo classifiers, perform in a similar fashion to the top, top 2, top 3, top 4 and top 5 ensembles. While they are not the best performing ensembles in terms of ranking, all of the top overall ensembles comfortably rank within the top third of classifiers and do so for all three ranking systems for all 10 runs. By utilizing a set of classifiers that rank highly over all 10 runs, we were able to build ensembles that were also consistently top ranked.

Looking at tables J.5 and J.6 in Appendix J, it can be seen that similar results were obtained for the other two ranking systems. The first 11 classifiers with the lowest average wins ranking are from ensembles and all 20 ensemble classifiers are found within the first 23 classifiers with the lowest average wins ranking. In terms of wins−losses, the first 16 classifiers are from ensembles and all 20 ensemble classifiers are found within the first 22 classifiers with the lowest average wins−losses ranking. It is therefore quite clear that ensemble classifiers not only perform well, but they perform consistently well across all runs and ranking systems.

Figure 5.5 provides a graphical visualization of how ensemble and solo classifiers change their ranking, in terms of losses, over 10 runs. As with Figure 5.4, the classifiers are arranged in ascending order of increasing average rank using the same colour coding. Once again we can use this colour coding to stratify classifier performance into three groups, the top third, middle third and bottom third, with all but one ensemble, the mean top positive ensemble, being found in the top third. This ensemble as discussed previously has an average losses ranking of 30.6, and is therefore ranked in both the top third (runs 1, 2, 3, 4, 7, and 8) and the middle third (runs 5, 6, 9, and 10). With the addition of ensembles, classifier rankings now only change between adjacent groups (top third $\longleftrightarrow$ middle third and middle third $\longleftrightarrow$ bottom third) with the boundary between the middle third and bottom third groups of classifiers being much more distinct. Similar graphical visualizations for the other two ranking systems (wins and wins−losses) are provided in Appendix K.

Figure 5.6 displays the δr values of classifiers produced by solo learners and ensembles for all 10 runs. As with the control, the δr values obtained are generally larger than those seen in our replication. In fact, in all runs apart from runs 2, 5 and 10, δr values higher than those in the control are observed. We believe that this is due to bagging reducing the correlation between training sets.

Focusing on the top ranked classifiers, the majority of which are ensembles, we can see that the lowest δr values in the top rank are also obtained from ensemble classifiers. To demonstrate this we have used a dashed line in each of the graphs in Figure 5.6, separating the top ranked ensembles with the lowest δr values from the rest. From this we can conclude that in each run of *Bagging N*, the majority of the top ranked classifiers are obtained from ensembles, and a subset of these are the most consistently performing top ranked classifiers as they have the lowest δr values.

Just as we did in the control, we can also analyse the performance of the classifiers in

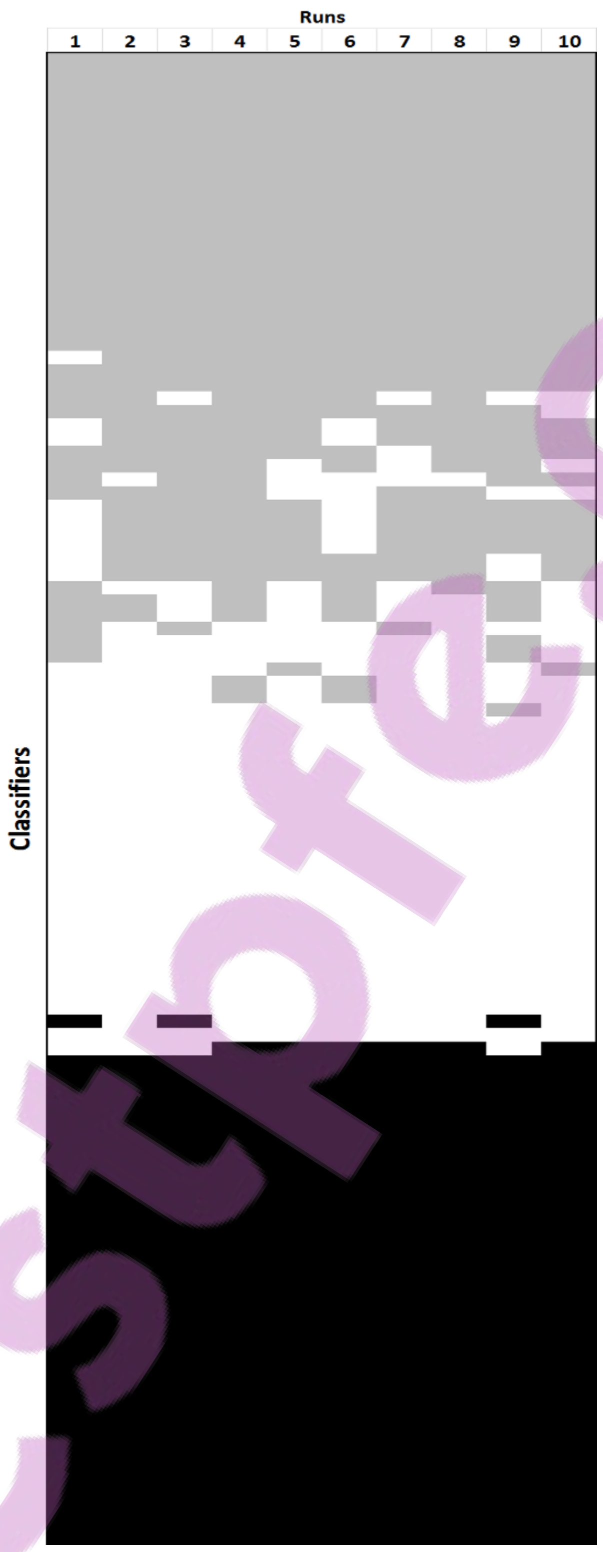


Figure 5.5: A graphical visualization of classifier performance; both solo and ensemble. As before the classifiers are arranged in order of increasing average ranking for losses. The colour system used is identical to that in Figure 5.4.

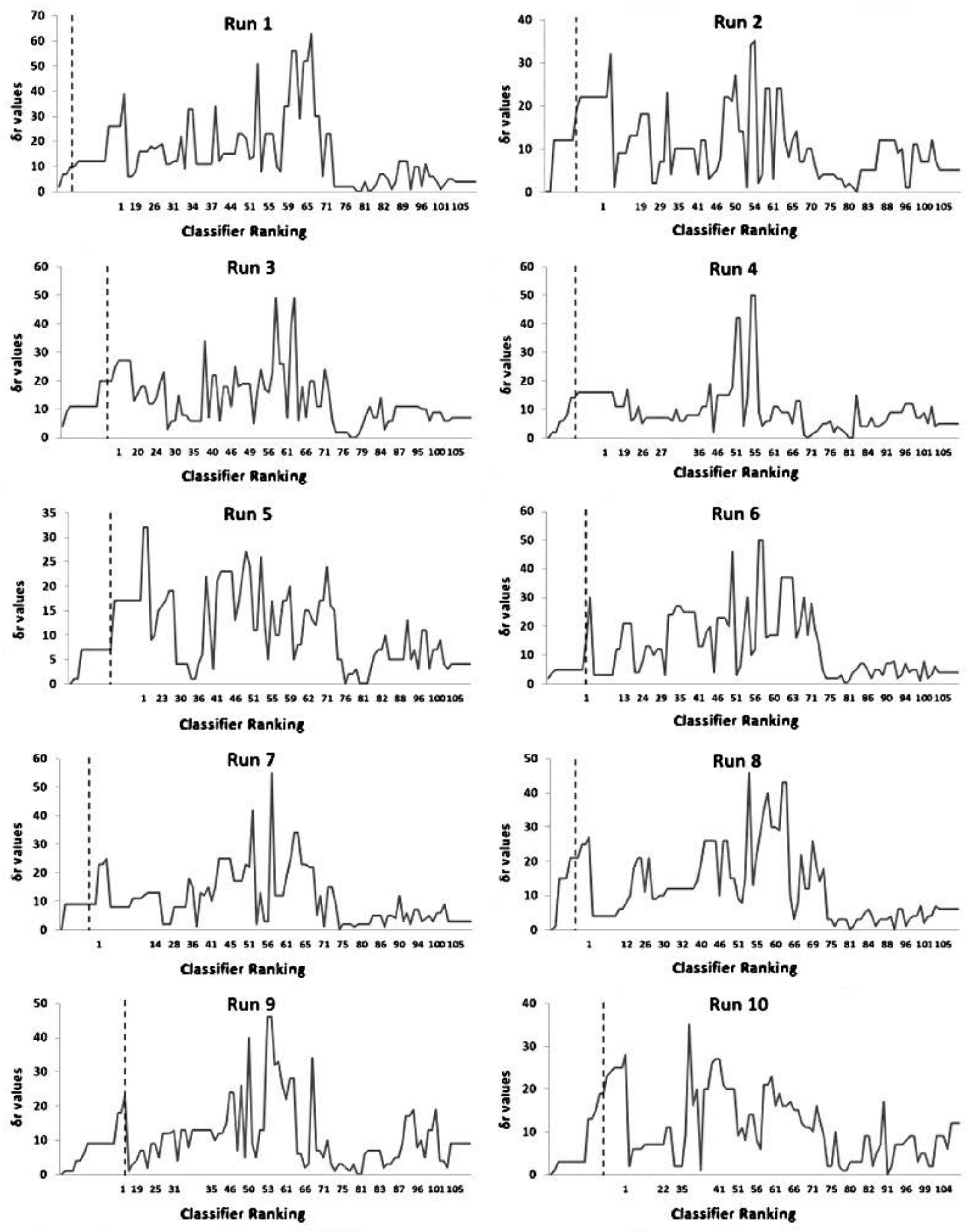


Figure 5.6: Graphs showing the δr values of classifiers produced by solo learners and ensembles for all 10 runs. The dashed line in each graph is used to separate top ranked ensembles which have lower δr values than the rest of the top ranked classifiers.

more detail by comparing their wins, losses, and tie counts separately. Figure 5.7 shows these counts for all 10 runs of *Bagging N*. Here the sum of wins, losses, and ties for every classifier is equal to 109, as each of the 110 classifiers is compared to the other 109 using only the absolute residual and its associated performance measure the mean AR. Once again we can see a decrease in the number of wins and a corresponding increase in the number of losses as classifier ranking worsens. There is thus a considerable difference in

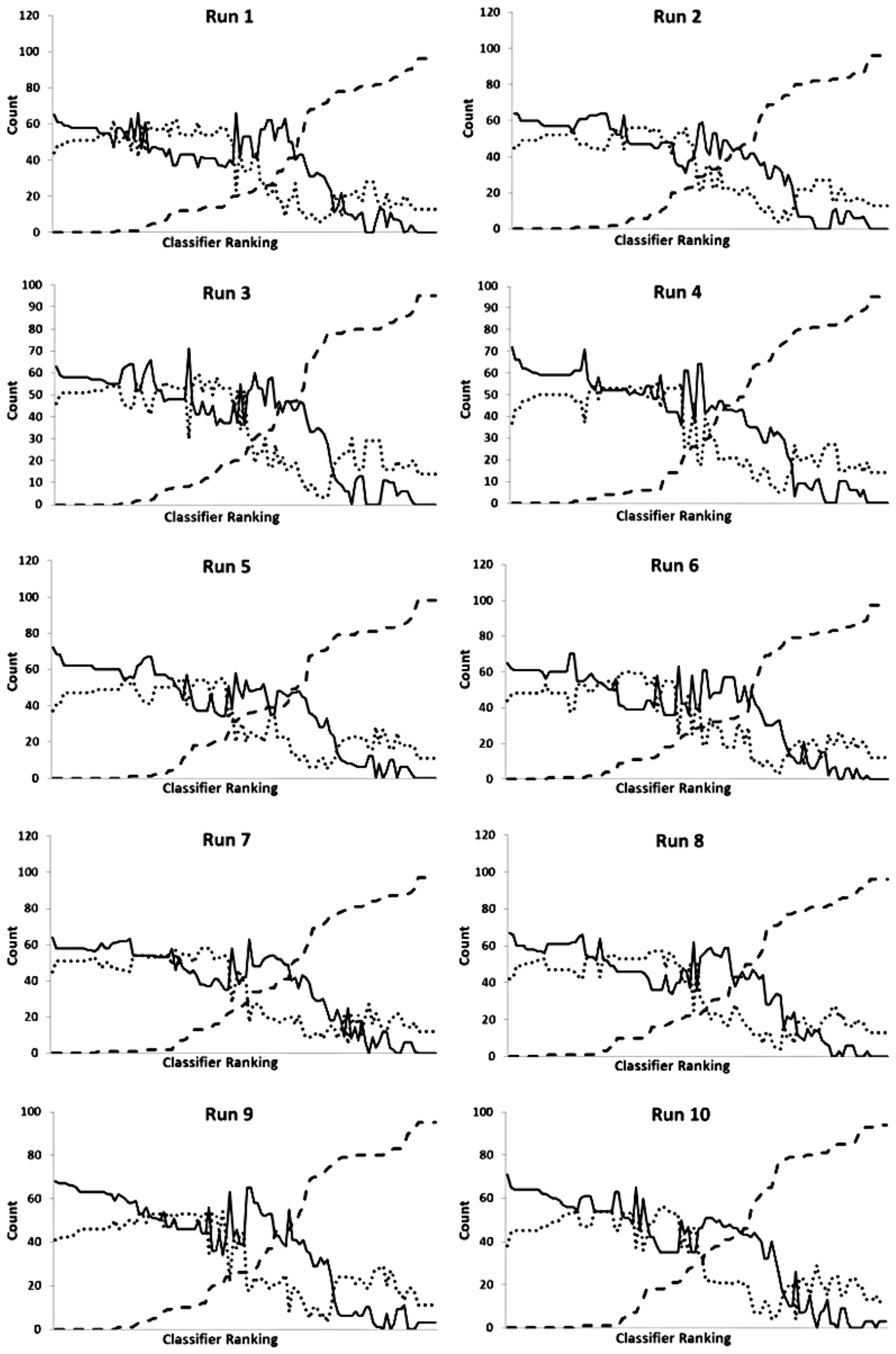


Figure 5.7: Graphs of wins (solid line), losses (dashed line), and ties (dotted line), for all classifiers, over all 10 runs. Note that classifier ranking worsens along the x-axis.

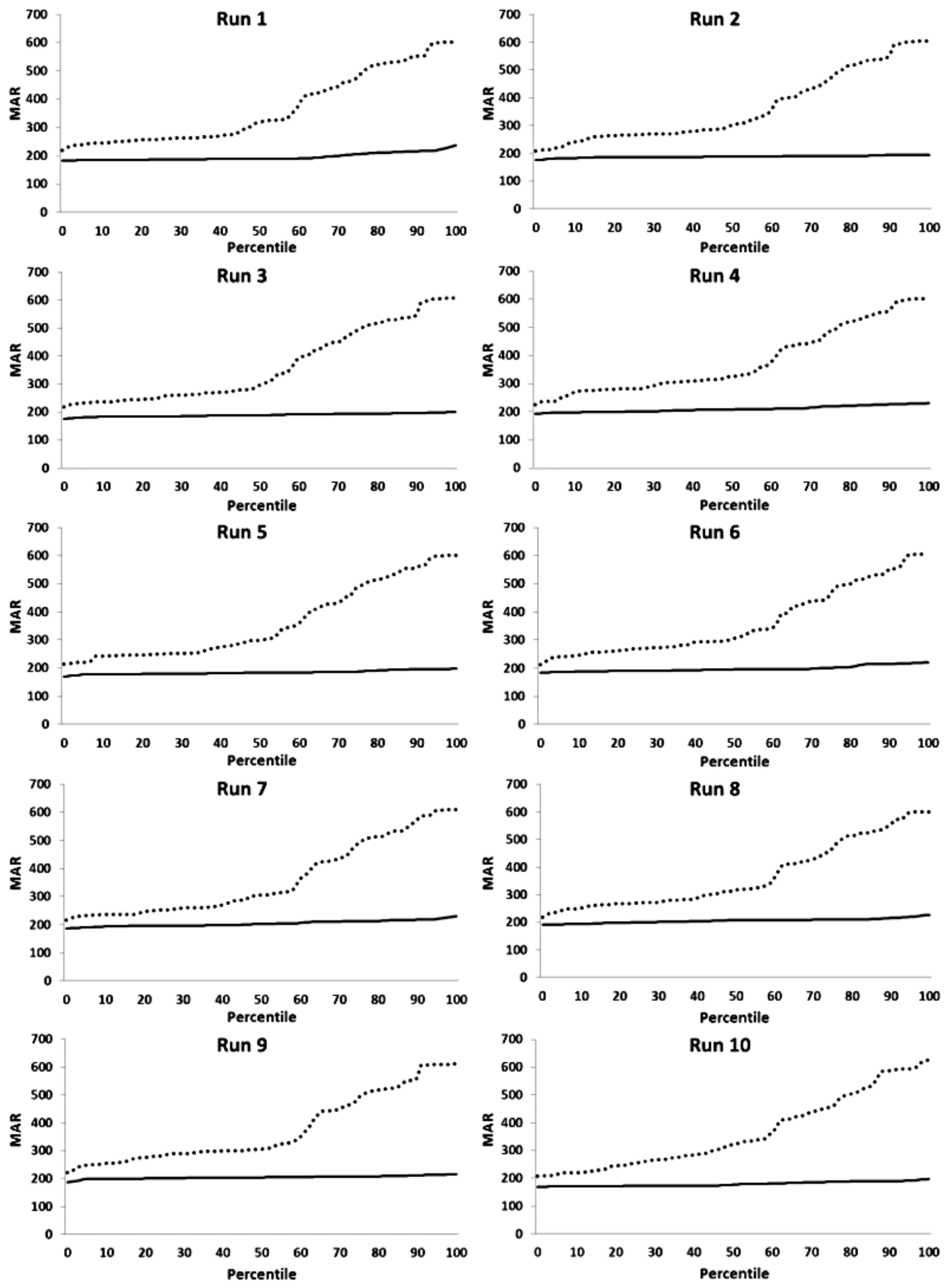


Figure 5.8: The range of mean AR values obtained by ensemble classifiers (solid line) versus their solo counterparts (dotted line), for all 10 runs of *Bagging N*.

terms of wins and losses for the classifiers produced by the highest ranking techniques and the lowest ranking techniques. This supports ranking the classifiers as we have in Figure 5.6.

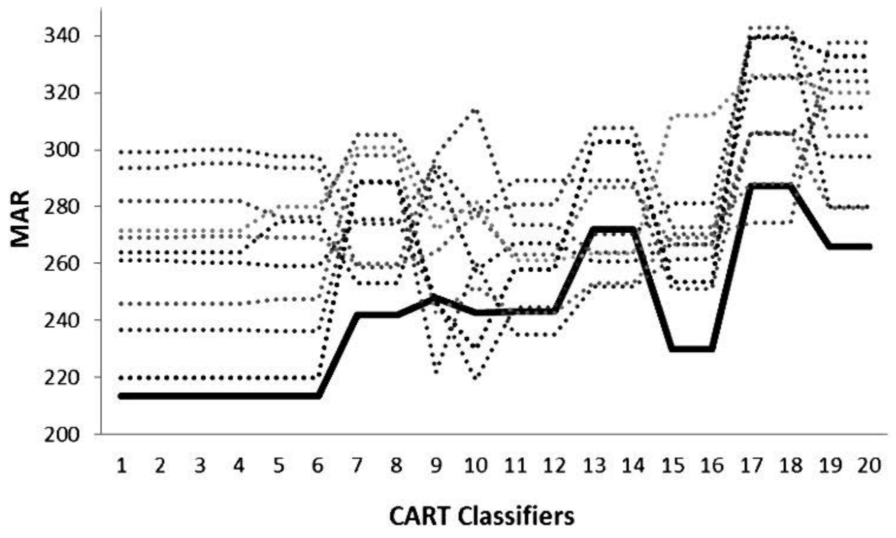
Next, we compared the mean AR values obtained from ensemble and solo classifiers. Looking at Figure 5.8 we can see that each of the 10 runs results in a graph similar to those obtained by our control and replication. Once again ensemble classifiers generate lower mean AR values than solo classifiers, and none of the ensembles produce the exceedingly large mean AR values seen with some of the solo classifiers. Therefore ensemble classifiers still perform well in comparison to their solo counterparts when *Bagging N* is used.

Looking at Figure 5.8 we can also see that the range of mean AR values obtained for solo classifiers with *Bagging N*, is similar to those obtained in the control. This result suggests that despite the reduction in training data due to *Bagging N*, there was still sufficient data for solo classifiers of comparable accuracy to be created. The resulting ensemble classifiers would therefore have a similar range of mean AR values, as is the case here.

Despite the range of mean AR values being similar, not all solo classifiers were uneffected by this reduction in training data. CART-based classifiers have performed less well with *Bagging N* than they did in the control. Looking at Table 5.3, only runs 3, 5 and 10 have CART-based classifiers that are ranked first, and only the 10th run features more than a couple of top ranked CART classifiers. In contrast, six of the 15 classifiers ranked first in the control (40%) were produced by variants of CART. We believe that this drop in rankings is due to CART-based classifiers being unstable learners; learners whose classifiers change with small changes in training set [6]. With *Bagging N*, CART learners have access to, on average, only 63.2% of the original training set when building a classifier [6, 12], resulting in their poorer performance.

Analogy-based learners on the other hand are stable learners. As such, we would expect classifiers obtained from analogy-based learners to be less effected by *Bagging N* than their CART counterparts [6, 12]. This is illustrated in Figure 5.9. The graph at the top of this figure compares the mean AR values obtained by all 20 CART learners (10 pre-processors $\times$ 2 variants of CART – pruned or unpruned) from the control (the solid line), with those obtained from the 10 runs of *Bagging N* (the dotted lines). The graph at the bottom of the figure does the same for all 20 analogy-based learners (10 pre-processors $\times$ 2 variants of analogy – 1 or 5 nearest neighbours). Looking at the top graph we can see that the mean AR values obtained during the 10 runs of *Bagging N* for CART learners, are for the most part, larger than those obtained by the control. In contrast, looking at the bottom graph we can see that there is far more overlap between the mean AR values obtained by the analogy-based learners during the 10 runs of *Bagging N* and the control. The effect *Bagging N* has on the performance of analogy-based learners is visibly smaller.

The change in CART performance may explain why most of the top ranked classifiers were obtained from ensembles when *Bagging N* was used, as illustrated in Table 5.6. Only four out of the 10 runs had more than two top ranked solo learners. In fact in run 9 all the top ranked classifiers were ensembles. This differs from the control where there was an almost even split of top ranked solo and ensemble classifiers.



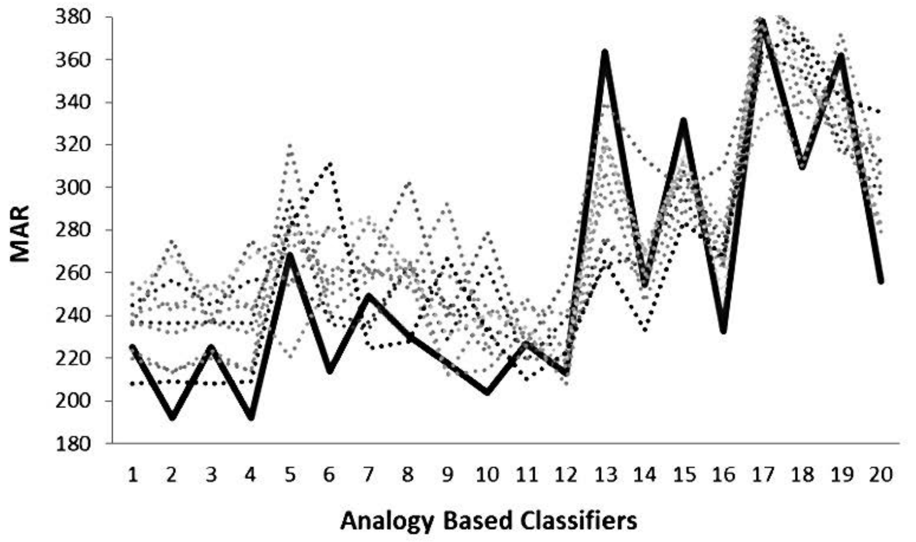


Figure 5.9: Graphs of mean AR values obtained for CART-based classifiers (top) and analogy-based classifiers (bottom). The mean AR values obtained during the 10 runs of *Bagging N* (dotted lines) are compared to the mean AR values obtained during the control (solid line).

In summary, the ensembles created using *Bagging N* produce classifiers that perform favourably in comparison to their solo counterparts. We observed that:

- When ranked in terms of losses all but one ensemble is found in the top third of rankings over all 10 runs; the exception being the mean top positive ensemble which, as in the control, is the lowest ranking ensemble classifier. In fact, when considering the average losses ranking over the 10 runs, the top 16 classifiers are ensemble classifiers. Similar results are obtained with the other two ranking systems: In terms of average wins ranking over the 10 runs, the top 11 classifiers are ensemble classifiers, and in terms of average wins−losses ranking, the top 16 classifiers are ensemble classifiers.

Table 5.6: The number of top ranked solo and ensemble classifiers for each run of *Bagging N*.

Run	Number of Top Ranked Classifiers	
	Solo	Ensemble
1	2	16
2	6	12
3	6	13
4	2	16
5	8	14
6	1	11
7	2	11
8	1	10
9	0	18
10	5	16

- In all 10 runs, most of the top ranked classifiers are obtained from ensembles. This was not the case with the control where there was an almost even split of solo and ensemble classifiers. Whereas unstable solo classifiers like CART have suffered from there being less training data available due to bagging, by combining estimates from a number of classifiers, ensembles have been largely uneffected. In each run a subset of these top ranked ensembles would also have the smallest δr values of all the top ranked classifiers, indicating their stability within that group. This is similar to the findings of the control.

- When looking at classifier performance in terms of mean AR, we found that the range of mean AR values seen with solo and ensemble classifiers created using *Bagging N* were similar to those seen in the control. While unstable solo classifiers saw an increase in mean AR values (due to reason discussed in the previous point), ensemble classifiers still generated lower mean AR values than their solo counterparts whilst avoiding the large mean AR values seen with some solo classifiers.

5.2.3 Bagging $2N_1$

In this section we will look at the results obtained from performing 10 runs of our first variant of *Bagging 2N* for Web effort estimation ensembles, called *Bagging* $2N_1$. *Bagging* $2N_1$ generates a bag that is twice the size of a regular training set, i.e. $2N = 388$. As with *Bagging N*, the entire bag is generated by sampling training set data with replacement. By increasing the number of samples taken from the training set data, the proportion of unique training data contained within each bag (which for *Bagging N* was on average 63.2% [6, 12]), increases.

Table 5.7 lists the rankings, for losses, of classifiers obtained from solo learners. Rankings for solo classifiers with an average ranking ≤ 20 are provided for all 10 runs: A complete list of rankings for losses, wins, and wins$-$losses is provided in Appendix L.

Table 5.7: Ranking based on losses for solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging 2N*$_1$.

Pre-processing Option	**Learner**	**Average Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
None	Analogy – 1NN	2.2	1	1	1	1	1	1	7	1	7	1
Normalization	Analogy – 1NN	2.2	1	1	1	1	1	1	7	1	7	1
Stepwise regression	Analogy – 1NN	3.4	1	1	1	10	1	1	7	1	7	4
None	Analogy – 5NN	4.6	9	1	1	4	1	1	1	12	1	15
Normalization	Analogy – 5NN	4.6	9	1	1	4	1	1	1	12	1	15
Stepwise regression	Analogy – 5NN	4.9	14	7	1	1	1	1	1	1	7	15
Sequential forward selection	Analogy – 1NN	9.4	8	19	9	10	1	1	7	19	16	4
PCA	Analogy – 1NN	9.7	9	7	7	4	9	12	1	16	17	15
Sequential forward selection	Analogy – 5NN	10	1	7	9	10	1	9	1	24	17	21
PCA	Analogy – 5NN	11	9	7	7	21	9	9	7	12	17	12
Natural logarithm	Analogy – 5NN	13.6	1	12	15	4	9	38	18	26	1	12
Equal width – 3 bins	Analogy – 5NN	14.5	15	20	15	30	12	15	13	9	15	1
Sequential forward selection	CART (yes)	15.1	19	7	9	20	24	18	1	12	21	20
Equal width – 5 bins	Analogy – 5NN	16.5	23	20	28	27	13	9	7	16	1	21
Sequential forward selection	CART (no)	17	23	1	31	23	24	1	15	20	17	15
None	CART (no)	17.2	26	12	12	10	37	29	30	1	11	4
None	CART (yes)	17.2	26	12	12	10	37	29	30	1	11	4
Normalization	CART (no)	17.5	26	12	15	10	37	29	30	1	11	4
Normalization	CART (yes)	17.5	26	12	15	10	37	29	30	1	11	4
Natural logarithm	CART (no)	17.8	26	12	15	10	24	27	30	9	21	4
Natural logarithm	CART (yes)	17.8	26	12	15	10	24	27	30	9	21	4
Stepwise regression	CART (no)	18.3	1	20	25	4	32	13	22	31	1	34
Stepwise regression	CART (yes)	18.3	1	20	25	4	32	13	22	31	1	34
Sequential forward selection	Stepwise regression	19.4	17	28	21	22	15	19	13	16	31	12

Once again we can see that all but one of the top ranking classifiers consist of variants of analogy-based learners (with one or five nearest neighbours) and CART (pruned and unpruned). The single exception this time is the classifier obtained using stepwise regression as the learner, with sequential forward feature selection as the pre-processor. As with *Bagging N*, the top ranked analogy-based learners have a better average ranking than the top ranked CART learners.

Figure 5.10 provides the same graphical visualization of classifier performance over 10 runs used with *Bagging N*. As before, the classifiers are arranged in ascending order of increasing average rank, in terms of losses, using the same colour coding: Classifiers ranked in the top third of a run are coloured gray. Those ranked in the middle third are coloured white, and those ranked in the bottom third are coloured black. This graphical visualization is provided for the other two ranking systems in Appendix M.

As was the case with *Bagging N*, we can see that there is a definite ordering of classifier performance, with the best and worst performing classifiers remaining in their respective thirds over all 10 runs. Similarly, at the boundaries between the top third and middle third, and the middle third and bottom third, we can see classifiers changing membership between adjacent thirds over the course of 10 runs. Partial least squares regression with a sequential forward feature selection pre-processor, is once again the only learner that produces a classifier whose performances change over all three thirds, the reason for which we have discussed in the previous section.

To evaluate *Bagging 2N*$_1$ we built the same set of 20 ensembles for each of the 10 runs,

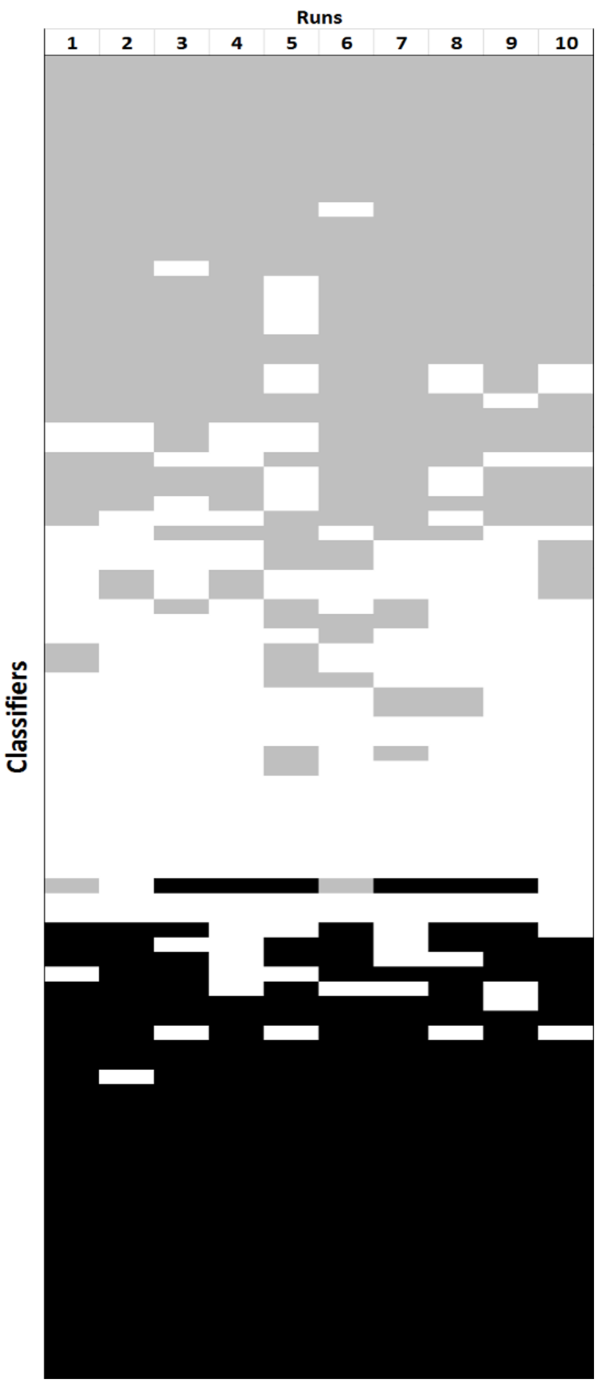


Figure 5.10: A graphical visualization of solo classifier performance for *Bagging 2N*$_1$ with the same ranking and colour system used with *Bagging N*.

using the top, top 2, top 3, top 4, top 5, top positive, and top overall ranked solo classifiers. The seven learners used to build the top overall ensembles are listed in Table 5.8. The only difference between these learners and those seen with *Bagging N* is the inclusion of the linear regression learner with a sequential forward feature selection pre-processor.

Table 5.9 lists the rankings, for losses, of classifiers obtained from solo learners and ensembles. Rankings are once again provided for all classifiers, solo and ensemble, with

Table 5.8: Learners whose classifiers are ranked in the top third over all 10 runs and all three ranking systems for *Bagging 2N*$_1$.

Pre-processing Option	Learner
None	Analogy – 1NN
Normalization	Analogy – 1NN
Stepwise regression	Analogy – 1NN
None	Analogy – 5NN
Normalization	Analogy – 5NN
Stepwise regression	Analogy – 5NN
Sequential forward selection	Linear regression

Table 5.9: Ranking based on losses for ensembles and solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging 2N*$_1$.

Pre-processing Option/ Combination Scheme	Learner	Average Rank	Run Rank									
			1	2	3	4	5	6	7	8	9	10
IRWM	Top 3	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 2	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 2	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 4	1	1	1	1	1	1	1	1	1	1	1
Median	Top	2.1	12	1	1	1	1	1	1	1	1	1
Mean	Top Overall	2.1	12	1	1	1	1	1	1	1	1	1
Mean	Top 5	2.8	1	19	1	1	1	1	1	1	1	1
Median	Top Positive	2.9	1	1	1	1	20	1	1	1	1	1
IRWM	Top Overall	3.5	12	1	1	15	1	1	1	1	1	1
Mean	Top	4.8	1	1	1	15	1	25	1	1	1	1
None	Analogy – 1NN	6	12	1	1	1	1	1	17	1	24	1
Normalization	Analogy – 1NN	6	12	1	1	1	1	1	17	1	24	1
Median	Top Overall	7.4	12	1	1	15	1	1	17	1	24	1
Median	Top 5	8.3	24	1	1	15	20	1	1	1	1	18
Median	Top 3	9.1	12	1	21	15	20	1	1	1	1	18
IRWM	Top Positive	10.3	24	25	21	1	1	1	27	1	1	1
Median	Top 4	11.1	12	1	1	15	20	25	17	1	1	18
Median	Top 2	14.7	1	19	21	22	20	1	17	27	1	18
None	Analogy – 5NN	15	24	19	1	22	1	1	17	31	1	33
Normalization	Analogy – 5NN	15	24	19	1	22	1	1	17	31	1	33

an average ranking ≤ 20 over the 10 runs: A complete list of rankings for losses, wins, and wins−losses is provided in Appendix L.

Focusing on ensemble classifiers we can see that they perform consistently well; this matches our results for *Bagging N* and those of our control. The first 13 classifiers with the lowest average losses ranking are from ensembles, and of those seven of them have an average rank of one having ranked first in all 10 runs. Once again the mean top positive ensemble is the only exception, being the lowest ranked ensemble classifier with an average losses ranking of 31.8. Creating ensembles using the top overall solo classifiers is also a successful strategy with *Bagging 2N*$_1$. These three top overall ensembles perform comparably to the top, top 2, top 3, top 4 and top 5 ensembles, easily ranking within the top third of classifiers for all three ranking systems over all 10 runs.

Looking at Appendix L, it can be seen that similar results were obtained for the other two ranking systems. The first 8 classifiers with the lowest average wins ranking are from ensembles and all 20 ensemble classifiers are found within the first 24 classifiers with the lowest average wins ranking. In terms of wins−losses, the first 18 classifiers are from ensembles and all 20 ensemble classifiers are found within the first 26 classifiers with the lowest average wins−losses ranking. Therefore when using *Bagging* $2N_1$ ensemble classifiers not only perform well, but they perform consistently well across all runs and ranking systems, which was also the case with *Bagging N*.

Our graphical visualization for both ensemble and solo classifier ranking, in terms of losses, over all 10 runs is provided in Figure 5.11. It can be seen that the graph here is very similar to the one obtained with *Bagging N* (Figure 5.5). All ensembles, apart from the mean top positive ensemble, are ranked in the top third for all 10 runs, and in fact, for all three ranking systems (see Appendix M). The mean top positive ensemble is ranked in the top third over half the runs (runs 1, 5, 6, 8, and 10) and in the middle third over the other half (runs 2, 3, 4, 7, and 9). With the addition of ensembles, classifier rankings now only change between adjacent groups (top third $\longleftrightarrow$ middle third and middle third $\longleftrightarrow$ bottom third) with the boundary between the middle third and bottom third groups of classifiers once again being much more distinct.

Figure 5.12 displays the δr values of classifiers produced by solo learners and ensembles for all 10 runs of *Bagging* $2N_1$. The δr values are similar to those seen with *Bagging N* and are therefore higher than those observed in our replication and control (only run 1 has similar δr values). Once again we believe that this is due to bagging reducing the correlation between training sets.

Focusing on the dashed line in each of the graphs in Figure 5.12, we can see that the lowest δr values in the top rank are also obtained from ensemble classifiers. In other words, as with *Bagging N*, the majority of the top ranked classifiers seen with *Bagging* $2N_1$ are obtained from ensembles, and a subset of these are the most consistently performing top ranked classifiers as they have the lowest δr values.

Figure 5.13 shows wins, losses, and tie counts for all 10 runs of *Bagging* $2N_1$. We can see that as classifier ranking worsens, the number of wins decreases along with a corresponding increase in the number of losses. This result has been consistent over all of our experiments using ensembles for Web effort estimation, and supports ranking the classifiers as we have in Figure 5.12.

Figure 5.14 displays the range of mean AR values obtained from ensemble and solo classifiers for each of the 10 runs of *Bagging* $2N_1$. The resulting graphs are similar to those obtained by our control and our 10 runs of *Bagging N*. Once again ensemble classifiers generate lower mean AR values than solo classifiers, and none of the ensembles produce the exceedingly large mean AR values seen with some of the solo classifiers. The fact that ensemble classifiers perform well in comparison to their solo counterparts is not surprising given that this was also the case when *Bagging N* was used.

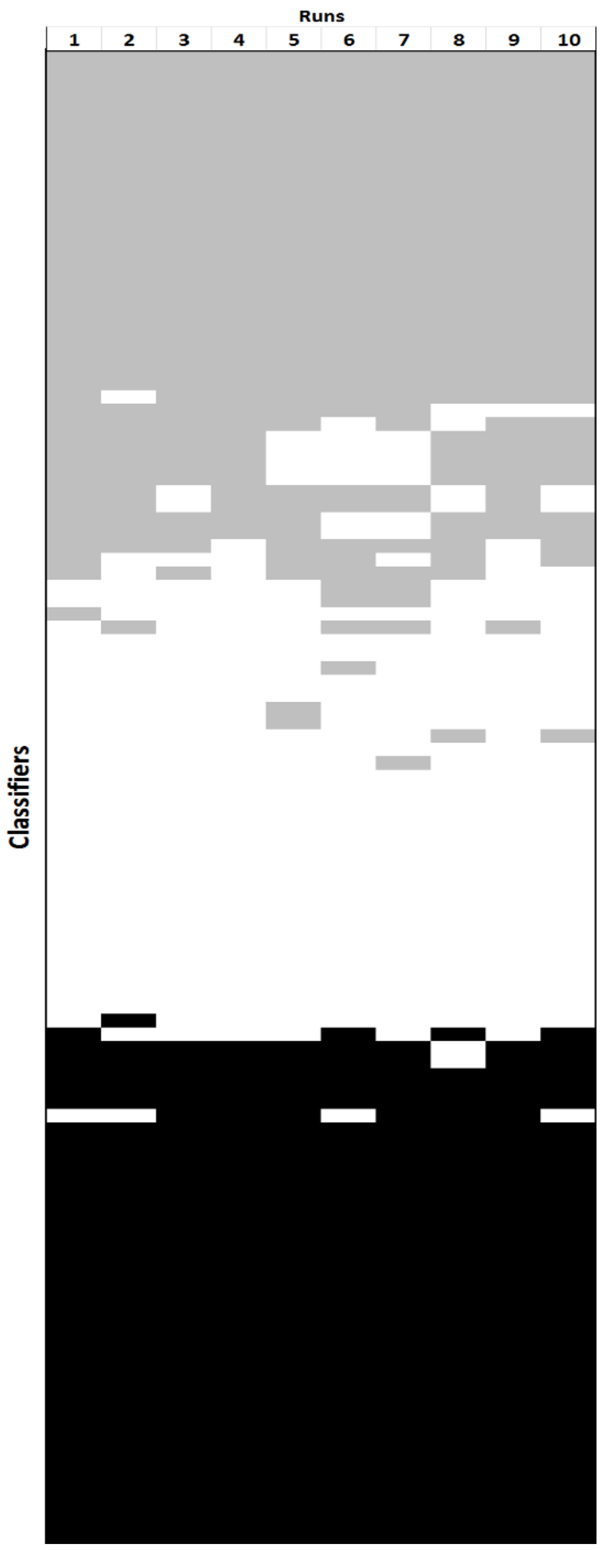


Figure 5.11: A graphical visualization of classifier performance, both solo and ensemble, for *Bagging $2N_1$* with the same ranking and colour system used with *Bagging N*.

In the previous section we saw that the range of mean AR values obtained for solo classifiers with *Bagging N* was similar to those obtained in the control, indicating that despite the reduction in the amount of training data, it was still sufficient for solo classifiers

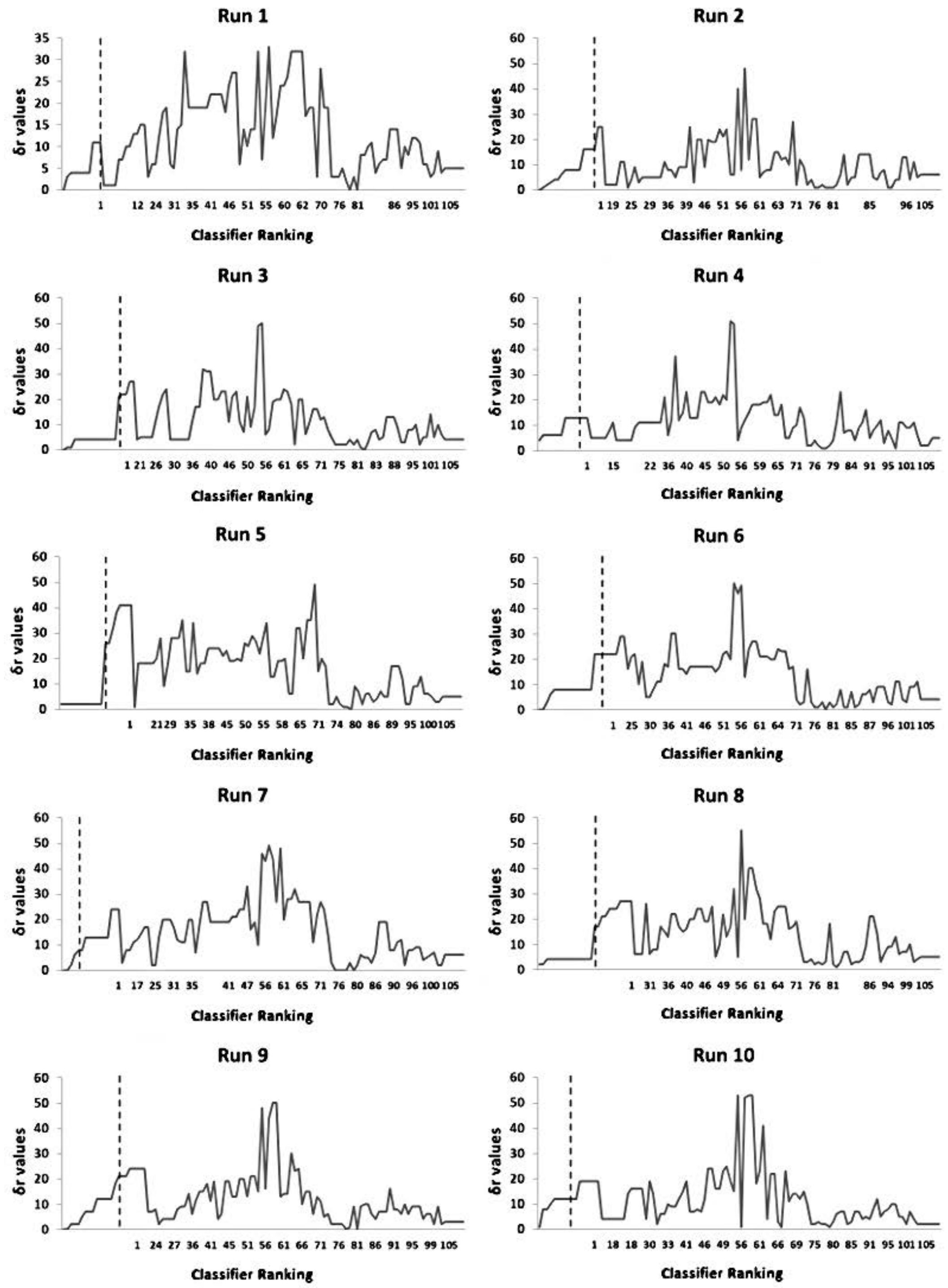


Figure 5.12: Graphs showing the δr values of classifiers produced by solo learners and ensembles for all 10 runs of *Bagging 2*N_1. The dashed line in each graph is used to separate top ranked ensembles which have lower δr values than the rest of the top ranked classifiers.

of comparable accuracy to be created. We would therefore expect that this will hold true with *Bagging 2*N_1 given that more training data is available, and looking at Figure 5.14 we can see that this is indeed the case.

The decrease in amount of training data did however affect the performance of CART-

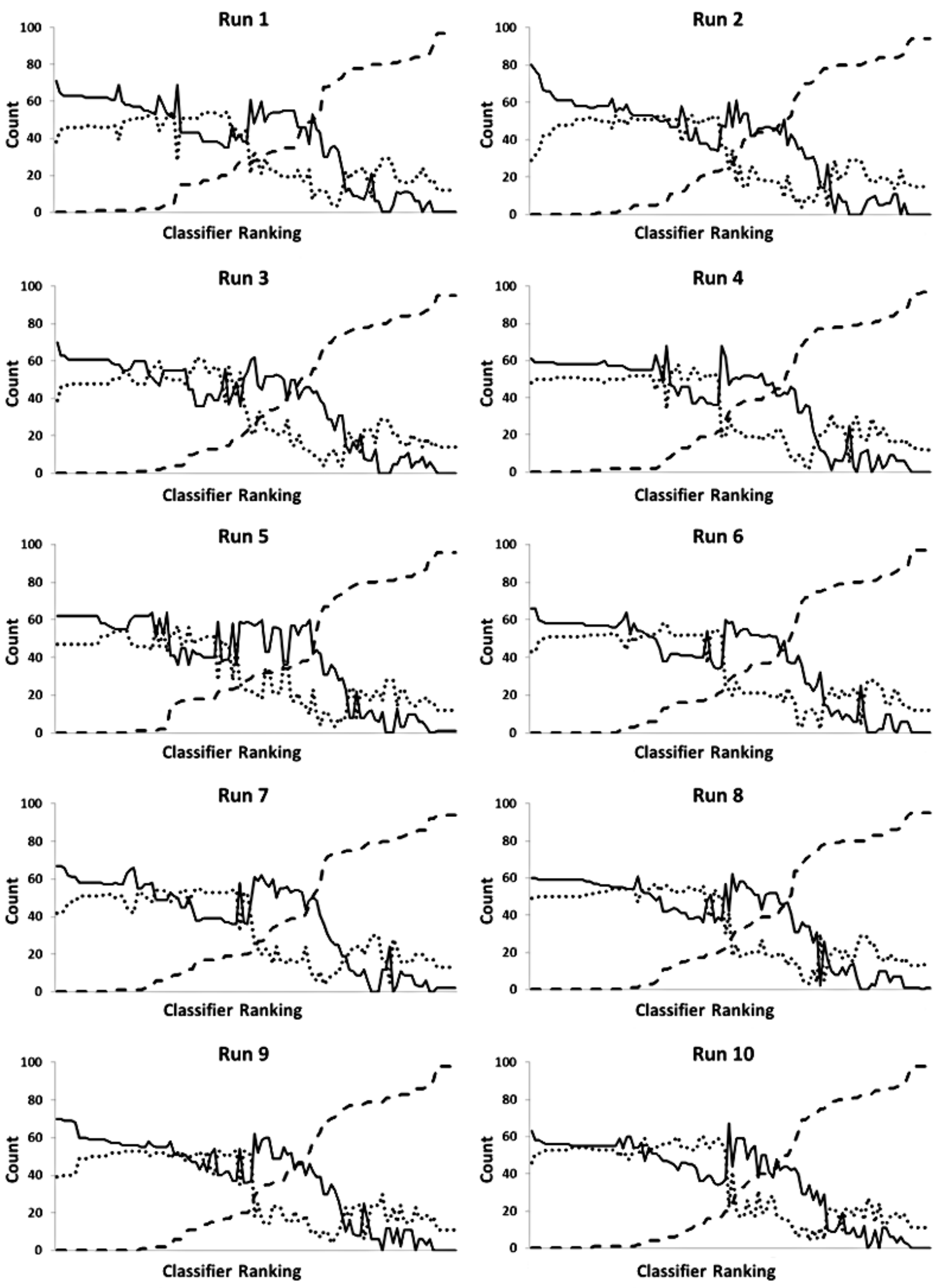


Figure 5.13: Graphs of wins (solid line), losses (dashed line), and ties (dotted line), for all classifiers, over all 10 runs of *Bagging 2*N_1. Note that classifier ranking worsens along the x-axis.

based classifiers with *Bagging N* due to CART being an unstable learner. It would thus be interesting to see how the increase in training data afforded by *Bagging 2*N_1 affects these classifiers.

Considering Table 5.7, we can see that six runs have top ranked CART-based classifiers (runs 1, 2, 6, 7, 8 and 9), with only the 8th run featuring more than a couple of top ranked

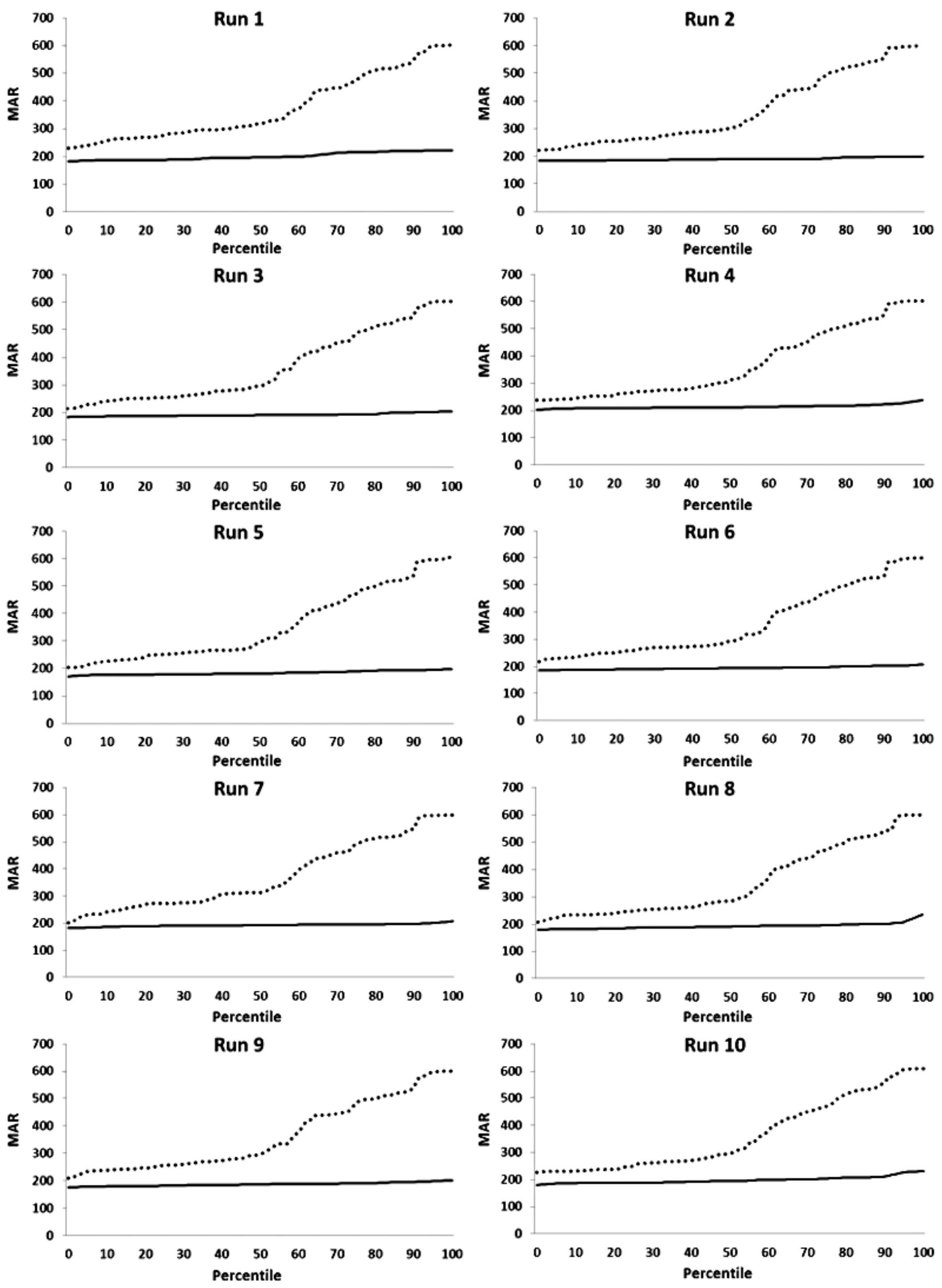
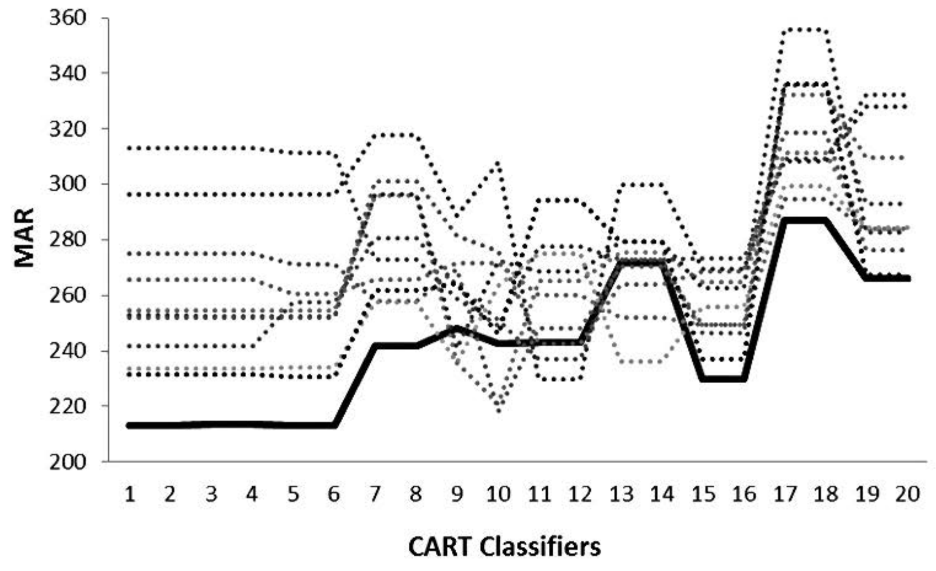


Figure 5.14: The range of mean AR values obtained by ensemble classifiers (solid line) versus their solo counterparts (dotted line), for all 10 runs of *Bagging 2N_1*.

CART classifiers. This is slightly better than the three *Bagging N* runs with top ranked CART-based classifiers, of which only a single one had more than a couple of top ranked CART classifiers. It is still not as good as our control where 40% of the top ranked classifiers were produced by variants of CART.

Figure 5.15 compares the performance of CART and analogy-based classifiers with



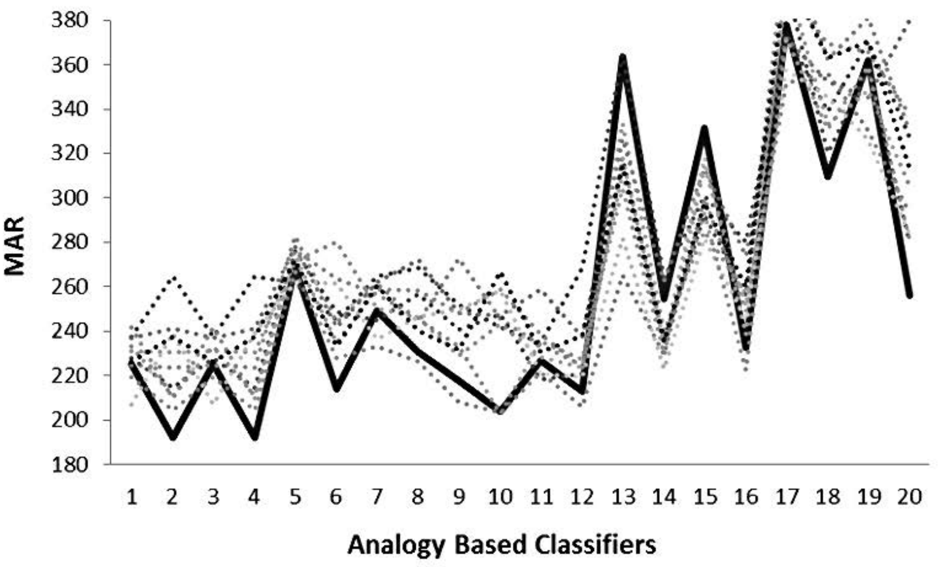


Figure 5.15: Graphs of mean AR values obtained for CART-based classifiers (top) and Analogy-based classifiers (bottom). The mean AR values obtained during the 10 runs of *Bagging 2N*$_1$ (dotted lines) are compared to the mean AR values obtained during the control (solid line).

Bagging 2N$_1$ against their respective performance in the control. The graph at the top of this figure compares the mean AR values obtained by all 20 CART learners from the control (the solid line), with those obtained from the 10 runs of *Bagging 2N*$_1$ (the dotted lines). The graph at the bottom of the figure does the same for all 20 analogy-based learners to provide a comparison of how stable learners are affected by bagging.

Looking at the top graph we can see that the mean AR values obtained during the 10 runs of *Bagging 2N*$_1$ for CART learners, are for the most part, larger than those obtained by the control. This was the case with *Bagging N* where we obtained a very similar result, although we do see some slightly higher mean AR values with *Bagging 2N*$_1$ (see

Table 5.10: The number of top ranked solo and ensemble classifiers for each run of *Bagging 2N_1*.

Run	**Number of Top Ranked Classifiers**	
	Solo	**Ensemble**
1	0	11
2	2	16
3	4	16
4	2	12
5	6	14
6	7	17
7	1	15
8	7	19
9	5	18
10	2	15

Figure 5.9 for comparison). The bottom graph on the other hand shows that there is far more overlap between the mean AR values obtained by the analogy-based learners during the 10 runs of *Bagging 2N_1* and the control. This too was the case with *Bagging N*, although we do see slightly more overlap with *Bagging 2N_1*. While it is unsuprising that stable analogy-based learners perform similarly with *Bagging 2N_1*, it is interesting to observe that despite more training data on average being available, we have not seen an improvement in performance of CART-based classifiers.

There are two possible reasons for this. The first reason would be that the increase in the amount of unique training data made available through *Bagging 2N_1* was not significant. The second reason is related to the duplicate data seen in bags due to sampling with replacement. This may make the bags (i.e. training sets) less representative of the Tukutuku data, making it more difficult for unstable learners like CART, to produce classifiers that accurately approximate the underlying true function. We should obtain a clearer picture of which one of these two reasons is more plausible in the next section where we discuss our second variant of *Bagging 2N*; here every bag contains the full training set of size N (negating the first reason), with the remaining N training cases being drawn with replacement.

Considering Table 5.10 we can see that most of the top ranked classifiers were obtained from ensembles when *Bagging 2N_1* was used. This was the case with *Bagging N*, and as discussed in the previous section, we believe that this is due to abovementioned drop in CART performance.

In summary, our results with *Bagging 2N_1* closely mirror those obtained with *Bagging N*. Ensemble classifiers still perform favourably in comparison to their solo counterparts with one (identical) exception; the mean top positive ensemble. Most of the top ranked classifiers are obtained from ensembles, and in each run a subset of these top ranked ensembles also have the smallest δr values, indicating their stability within this group. As expected, stable analogy-based classifiers are less effected than their unstable CART-

Table 5.11: Ranking based on losses for solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging 2N$_2$*.

Pre-processing Option	Learner	Average Rank	Run Rank 1	2	3	4	5	6	7	8	9	10
Stepwise regression	Analogy – 5NN	1	1	1	1	1	1	1	1	1	1	1
None	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Normalization	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Sequential forward selection	Analogy – 1NN	1.9	10	1	1	1	1	1	1	1	1	1
Stepwise regression	Analogy – 1NN	2	1	1	1	1	1	1	1	1	1	11
None	Analogy – 5NN	4.7	1	1	7	9	17	1	1	8	1	1
Normalization	Analogy – 5NN	4.7	1	1	7	9	17	1	1	8	1	1
Normalization	CART (no)	7.9	13	11	11	9	1	1	1	11	8	13
Normalization	CART (yes)	7.9	13	11	11	9	1	1	1	11	8	13
None	CART (no)	7.9	13	11	11	9	1	1	1	11	8	13
None	CART (yes)	7.9	13	11	11	9	1	1	1	11	8	13
Sequential forward selection	Analogy – 5NN	9	27	23	9	1	11	1	1	1	15	1
Natural logarithm	Analogy – 5NN	9.4	10	8	1	1	21	1	14	22	15	1
PCA	Analogy – 5NN	9.4	1	8	22	9	14	1	1	11	14	13
PCA	Analogy – 1NN	11.6	1	11	9	9	14	19	17	8	15	13
Natural logarithm	CART (no)	11.7	13	11	11	19	11	1	14	16	8	13
Natural logarithm	CART (yes)	11.7	13	11	11	19	11	1	14	16	8	13
Sequential forward selection	CART (yes)	14.2	1	18	21	8	14	21	24	1	21	13
Stepwise regression	CART (no)	15.2	23	19	11	9	26	1	21	16	25	1
Stepwise regression	CART (yes)	15.2	23	19	11	9	26	1	21	16	25	1
Sequential forward selection	CART (no)	16.3	23	8	22	19	1	19	25	16	18	12

based counterparts. Surprisingly the increase in training data afforded by using bags of size $2N$ did not result in an appreciable improvement in performance of CART-based classifiers. We believe that this could be due either to the increase in training data not being significant, or the inclusion of duplicates making bags less representative of the Tukutuku dataset. Our second variant of *Bagging 2N* where bags include all training data should shed light on this matter and is discussed in the next section.

5.2.4 Bagging 2N$_2$

We will now analyze the results obtained from performing 10 runs of our second variant of *Bagging 2N*, called *Bagging 2N$_2$*. As with *Bagging 2N$_1$*, bags are twice the size of a regular training set ($2N = 388$). Each bag consists of:

- $N = 194$ training cases that would make up a training set if we were only using leave-one-out cross-validation. In other words, all projects apart from the test case.
- $N = 194$ training cases drawn from the abovementioned projects via sampling with replacement.

With *Bagging 2N$_2$* we can thus see how bagging performs for effort estimation when there is no loss of training data.

Table 5.11 lists the rankings, for losses, of classifiers obtained from solo learners. Rankings are provided for all 10 runs, for classifiers with an average ranking of ≤ 20: A complete list of rankings for losses, wins, and wins−losses is provided in Appendix N.

We can see that all of the top ranking classifiers consist of variants of analogy-based learners (with one or five nearest neighbours) and CART (pruned and unpruned). As with *Bagging N* and *Bagging 2N*$_1$, the top ranked analogy-based learners have a better average ranking than the top ranked CART-based learners. CART learners have however definitely performed better, in terms of ranking, than they have in our previous two bagging experiments. This indicates that eliminating loss of data from the training set with *Bagging 2N*$_2$ has made a difference in CART performance.

Figure 5.16 provides a graphical visualization of classifier performance over 10 runs of *Bagging 2N*$_2$. As before, the classifiers are arranged in ascending order of increasing average rank, in terms of losses, using the same colour coding: Classifiers ranked in the top third of a run are coloured gray. Those ranked in the middle third are coloured white, and those ranked in the bottom third are coloured black. This graphical visualization is provided for the other two ranking systems in Appendix O.

Once again there is a definite ordering of classifier performance, with the best and worst performing classifiers remaining in their respective thirds over all 10 runs. Similarly, at the boundaries between the top third and middle third, and the middle third and bottom third, we can see classifiers changing membership between adjacent thirds over the course of 10 runs. *Bagging 2N*$_2$ seems to have had a stabilizing effect, reducing the amount of these transitions in comparison to what we saw with *Bagging N* and *Bagging 2N*$_1$ (see Figures 5.4 and 5.10 respectively). In fact, unlike with *Bagging N* and *Bagging 2N*$_1$, performance transitions have now been limited to only the adjacent thirds (top third $\longleftrightarrow$ middle third and middle third $\longleftrightarrow$ bottom third).

The same set of 20 ensembles were created to evaluate *Bagging 2N*$_2$. The solo learners used to build the top overall ensembles are listed in Table 5.12. There are two key differences between this list of top overall solo learners and those seen previously. Firstly, there are considerably more top overall solo learners; 14 of them as opposed to seven and six for *Bagging 2N*$_1$ and *Bagging N* respectively. The other difference is that there are six CART learners in this list, whereas there were none seen with the previous two versions of bagging. Once again it does seem that the inclusion of all training data has improved CART performance.

Table 5.13 lists the rankings, for losses, of classifiers obtained from solo learners and ensembles. As before, rankings are provided for all 10 runs for classifiers with an average losses ranking of ≤ 20: A complete list of rankings for losses, wins, and wins$-$losses is provided in Appendix N.

Focusing on ensemble classifiers we can see that they perform consistently well; this matches our other results with or without bagging. Once again 19 of the 20 ensembles created using *Bagging 2N*$_2$ appear in this table, with the mean top positive ensemble being absent due to its average losses ranking of 41.1. The top overall ensembles are just

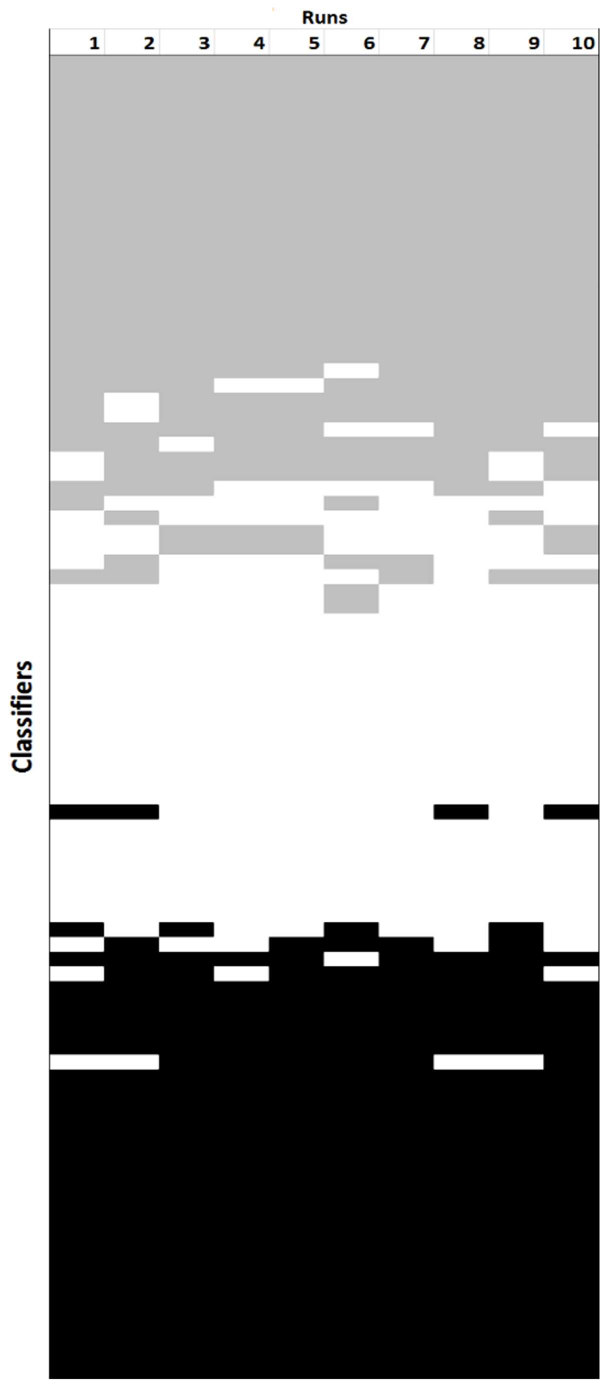


Figure 5.16: A graphical visualization of solo classifier performance for *Bagging 2N*$_2$ with the same ranking and colour system used with *Bagging N*.

as successful as they were previously and rank within the top third of classifiers for all three ranking systems over all 10 runs.

There are some differences though. Unlike with *Bagging N* and *Bagging 2N*$_1$, where only ensemble classifiers were top ranked over all 10 runs (i.e. with an average losses ranking of 1), two of them here were obtained from solo learners using analogy with one nearest neighbour. CART-based classifiers make their appearance for the first time in this

Table 5.12: Learners whose classifiers are ranked in the top third over all 10 runs and all three ranking systems for *Bagging 2N*$_2$.

Pre-processing Option	**Learner**
Stepwise regression	Analogy – 5NN
None	Analogy – 1NN
Normalization	Analogy – 1NN
Sequential forward selection	Analogy – 1NN
Stepwise regression	Analogy – 1NN
None	Analogy – 5NN
Normalization	Analogy – 5NN
Normalization	CART (no)
Normalization	CART (yes)
None	CART (no)
None	CART (yes)
Natural logarithm	Analogy – 5NN
Natural logarithm	CART (no)
Natural logarithm	CART (yes)

list having had average losses rankings of greater than 20 previously. In fact, there are more classifiers with an average losses ranking of ≤ 20 with *Bagging 2N*$_2$; 31 classifiers compared to 25 with *Bagging N* and 23 with *Bagging 2N*$_1$. This again suggests that the inclusion of all training data with *Bagging 2N*$_2$ has had a beneficial effect on solo classifier performance, particular CART-based classifiers.

Looking at Appendix N, it can be seen that similar results were obtained for the other two ranking systems. The first 17 classifiers with the lowest average wins ranking are from ensembles and all 20 ensemble classifiers are found within the first 24 classifiers with the lowest average wins ranking. In terms of wins−losses, the first 18 classifiers are from ensembles and all 20 ensemble classifiers are found within the first 37 classifiers with the lowest average wins−losses ranking. Therefore, when using *Bagging 2N*$_2$ ensemble classifiers not only perform well, but they perform consistently well across all runs and ranking systems, which was also the case with both *Bagging N* and *Bagging 2N*$_1$.

Figure 5.17 provides our graphical visualization for both ensemble and solo classifier losses ranking, over all 10 runs. The findings here are very similar to those seen with *Bagging N* and *Bagging 2N*$_1$: All ensembles, apart from the mean top positive ensemble, are ranked in the top third for all 10 runs for all three ranking systems (graphical visualizations of the other two ranking systems are available in Appendix O). As before, the inclusion of ensembles has had a stabilizing effect on ranking transitions which are now far less frequent than they were when only solo classifiers were considered. Comparing Figure 5.17 to its counterparts for *Bagging N* (Figure 5.5) and *Bagging 2N*$_1$ (Figure 5.11), we can see that *Bagging 2N*$_2$ has provided the clearest stratification of classifier performance out of the three bagging techniques analyzed. There is a difference of note. The

Table 5.13: Ranking based on losses for ensembles and solo classifiers with an average ranking ≤ 20 over 10 runs of *Bagging 2N$_2$*.

Pre-processing Option/ Combination Scheme	**Learner**	**Average Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top Positive	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top Overall	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 2	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
None	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Normalization	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 2	2.4	1	1	15	1	1	1	1	1	1	1
Mean	Top 3	2.4	1	1	15	1	1	1	1	1	1	1
Mean	Top 4	2.4	1	1	15	1	1	1	1	1	1	1
Median	Top 5	2.9	1	1	1	1	1	1	1	1	20	1
Mean	Top 5	3.4	1	1	1	25	1	1	1	1	1	1
Median	Top 4	4.3	1	1	15	1	1	1	1	1	20	1
Mean	Top Overall	4.4	21	1	15	1	1	1	1	1	1	1
Sequential forward selection	Analogy – 1NN	5.9	27	24	1	1	1	1	1	1	1	1
Median	Top 2	6.8	1	1	15	1	26	1	1	1	1	20
Median	Top 3	7.2	1	1	25	1	1	1	1	1	20	20
Median	Top Overall	11.3	24	1	15	20	1	1	29	1	20	1
Median	Top	11.5	21	1	1	20	1	1	29	1	20	20
None	Analogy – 5NN	16.1	1	1	25	26	32	32	1	22	1	20
Normalization	Analogy – 5NN	16.1	1	1	25	26	32	32	1	22	1	20
Stepwise regression	Analogy – 1NN	17.9	21	24	15	20	26	1	1	22	20	29
Stepwise regression	Analogy – 5NN	18.1	1	24	15	20	26	32	1	22	20	20
IRWM	Top Positive	19.1	24	1	15	26	1	32	29	32	1	30
Normalization	CART (no)	19.6	28	27	28	26	1	1	1	27	27	30
Normalization	CART (yes)	19.6	28	27	28	26	1	1	1	27	27	30
None	CART (no)	19.6	28	27	28	26	1	1	1	27	27	30
None	CART (yes)	19.6	28	27	28	26	1	1	1	27	27	30
Sequential forward selection	Analogy – 5NN	19.9	41	37	36	1	26	1	1	22	33	1

mean top positive ensemble, the lowest ranked ensemble per usual, is now ranked in the middle third over all 10 runs. We believe that this is due to the improvement seen with solo classifier performance, particularly those using CART as a learner, displacing the mean top positive ensemble down the rankings.

Figure 5.18 displays the δr values of classifiers produced by solo learners and ensembles for all 10 runs of *Bagging 2N$_2$*. The δr values are higher than those observed in our replication and control with only 4 runs having a similar range of δr values. This was also the case with *Bagging N* and *Bagging 2N$_1$*. However, the range of δr values seen with *Bagging 2N$_2$* are slightly lower than those obtained with the previous two bagging experiments. This is due to there being a higher correlation between training sets, since a *Bagging 2N$_2$* bag contains all training data that would be included in a regular leave-one-out cross-validation training set.

The dashed line in each of the graphs in Figure 5.18 separates top ranked ensemble classifiers with the lowest δr values, from the remaining top ranked classifiers. As was the case with *Bagging N* and *Bagging 2N$_1$*, the majority of top-ranked classifiers are obtained from ensembles, and a subset of these are the most consistently performing top-ranked classifiers, as they have the lowest δr values.

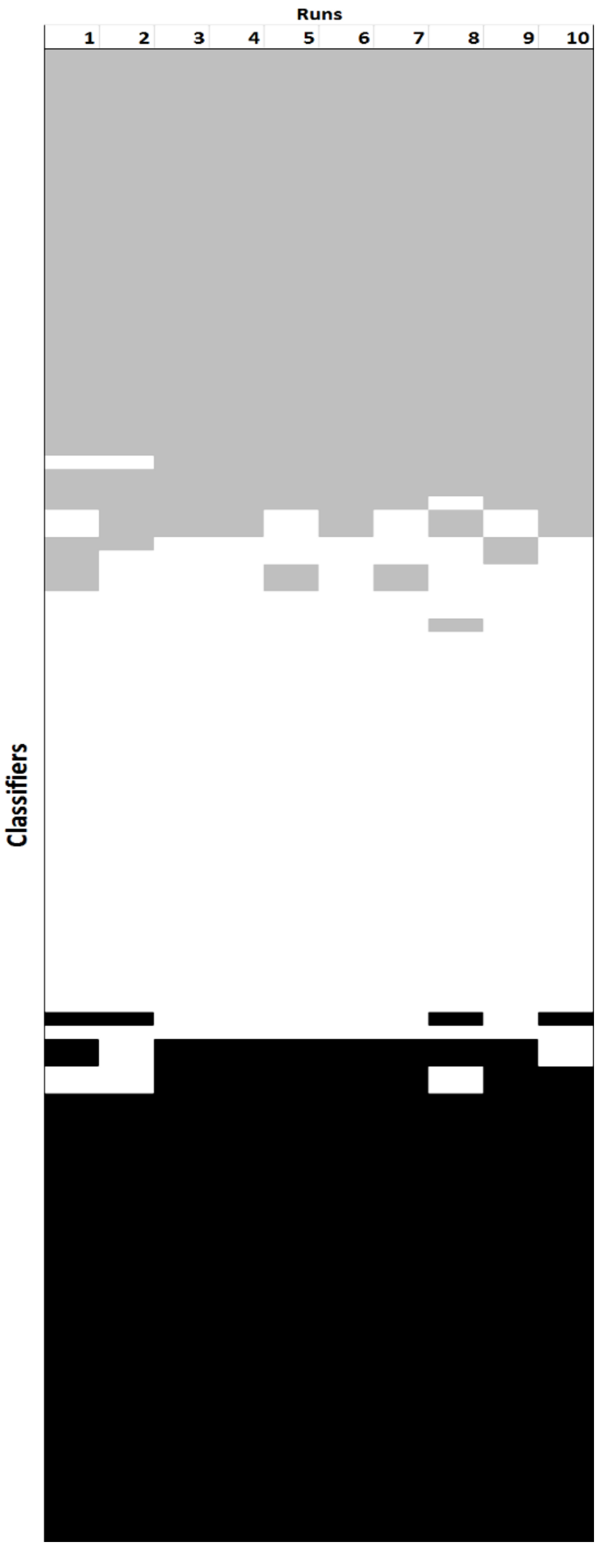


Figure 5.17: A graphical visualization of classifier performance, both solo and ensemble, for *Bagging $2N_2$* with the same ranking and colour system used with *Bagging N*.

Figure 5.19 shows wins, losses, and tie counts for all 10 runs of *Bagging* $2N_2$. We can see that as classifier ranking worsens, the number of wins decreases along with a corresponding increase in the number of losses. This result has been consistent over all

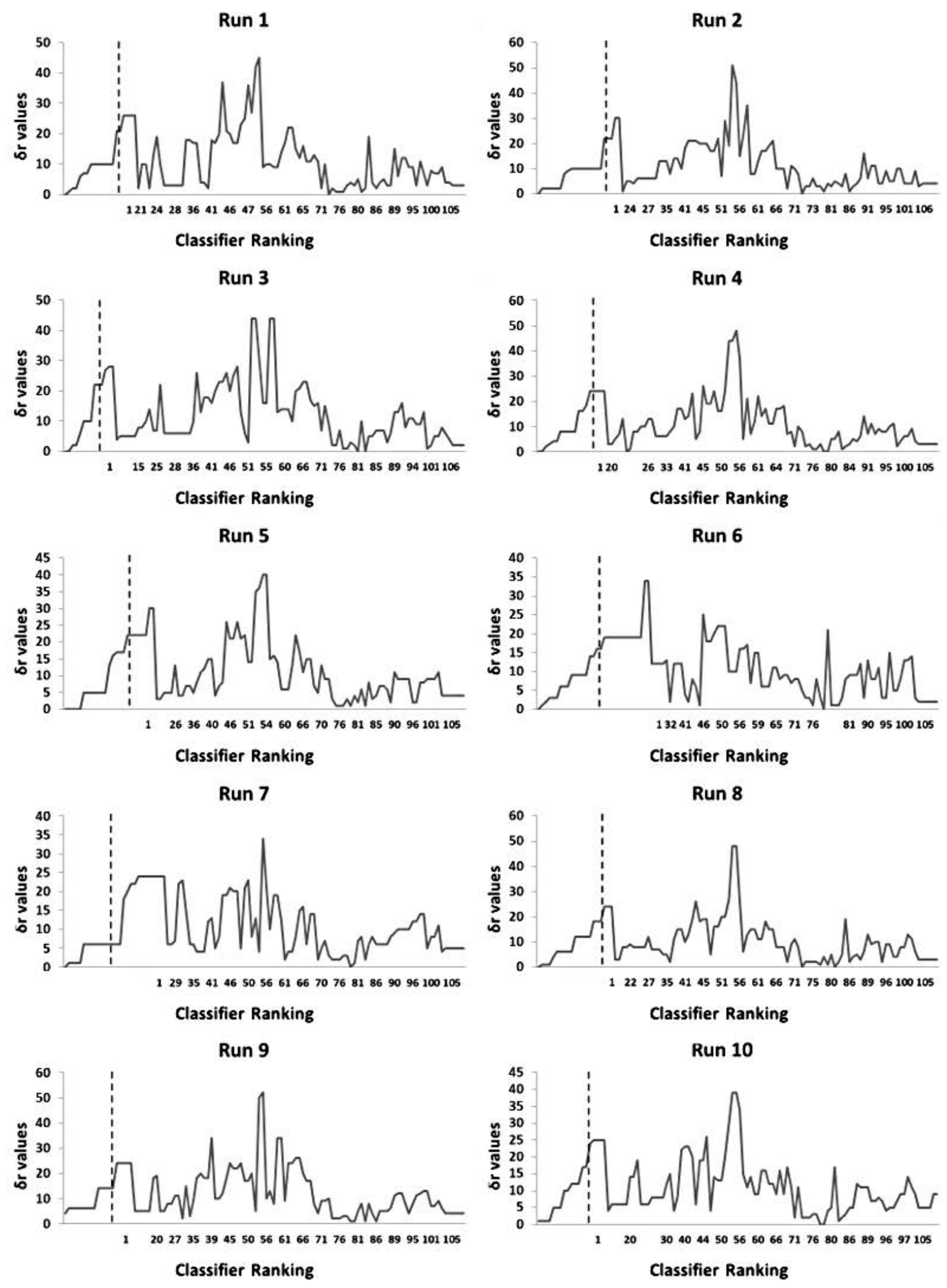


Figure 5.18: Graphs showing the δr values of classifiers produced by solo learners and ensembles for all 10 runs of *Bagging 2N*$_2$. The dashed line in each graph is used to separate top ranked ensembles which have lower δr values than the rest of the top ranked classifiers.

of our experiments using ensembles for Web effort estimation, with or without bagging, and acts as a validation of our ranking system.

Figure 5.20 displays the range of mean AR values obtained from ensemble and solo classifiers for all 10 runs of *Bagging 2N$_2$*. Looking at the graphs we can see that the results

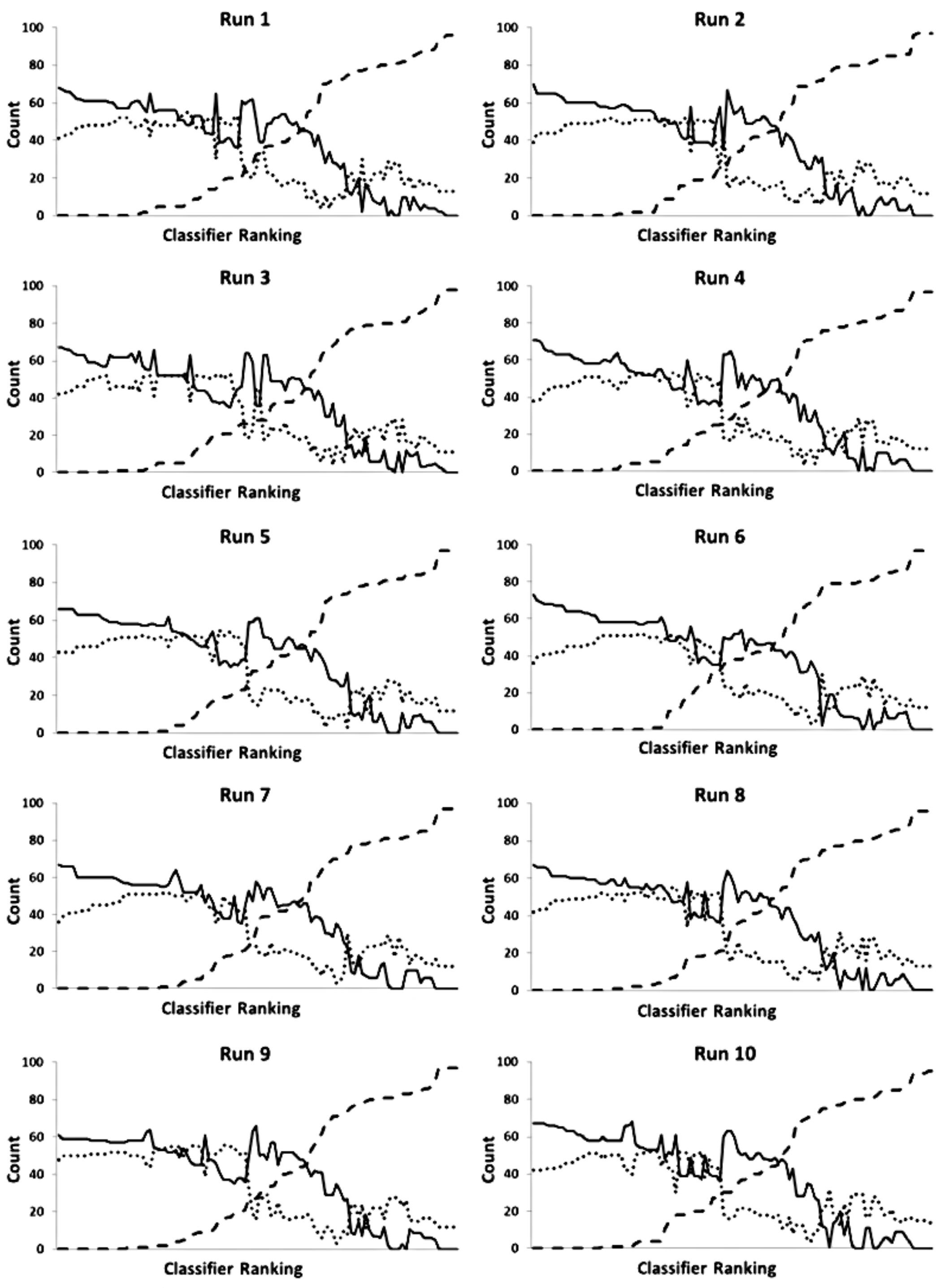


Figure 5.19: Graphs of wins (solid line), losses (dashed line), and ties (dotted line), for all estimation techniques, over all 10 runs of *Bagging 2N*$_2$.

are consistent with what we have obtained from our other experiments using ensembles for Web effort estimation, regardless of whether or not bagging has been used. As before, ensemble classifiers generate lower mean AR values than solo classifiers, and none of the ensembles produce the exceedingly large mean AR values seen with some of the solo classifiers.

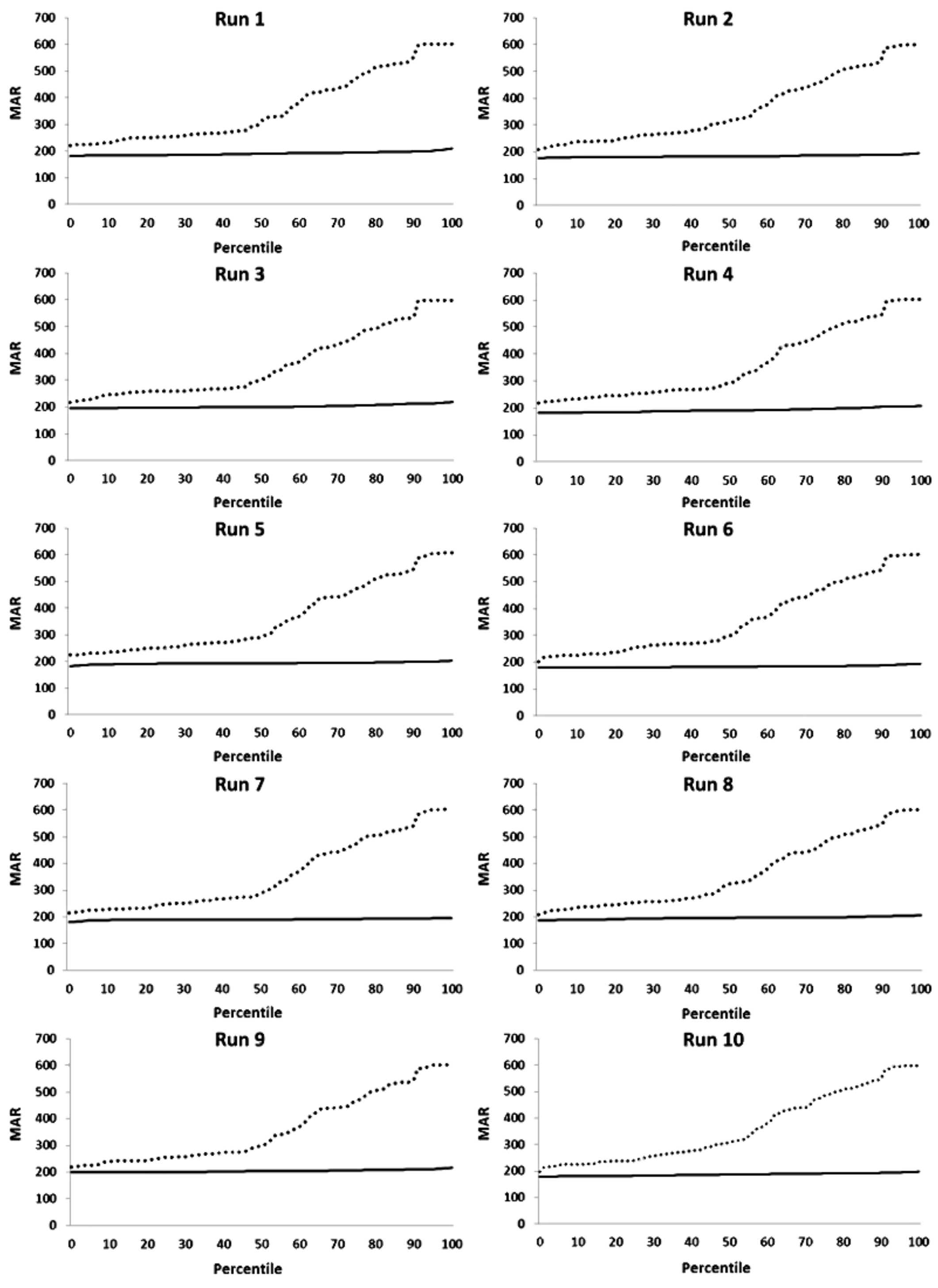


Figure 5.20: The range of mean AR values obtained by ensemble classifiers (solid line) versus their solo counterparts (dotted line), for all 10 runs of *Bagging 2N*$_2$.

With the two previous versions of bagging investigated we saw that the range of mean AR values obtained for solo and ensemble classifiers was similar to those obtained in the control. The fact that solo classifiers performed at a similar level indicated that despite a reduction in training data due to the bagging process, there was enough available for solo classifiers of comparable accuracy to be created. With *Bagging 2N*$_2$, classifiers have access to all the training data that would have been available with leave-one-out cross-validation without bagging, so we would expect the results here to be similar to those of the control as well. Figure 5.20 reveals that this is indeed the case.

Unstable CART-based classifiers did however suffer from a drop in performance, as evidenced by both their ranking and mean AR due to the decrease in training data available. Increasing the amount of training data from *Bagging N* to *Bagging 2N*$_1$ did not make much of a difference, and in the previous section we theorized that this could be due to either the increase in training data not being significant enough, or due to duplicate data making bags less representative of the Tukutuku dataset. *Bagging 2N*$_2$ eliminates the first issue, so analyzing the performance of CART-based classifiers with this variant of bagging should give us a clearer picture of what has been going on.

Figure 5.21 compares the performance of CART and analogy-based classifiers with *Bagging 2N*$_2$ against their respective performance in the control. The graph at the top of this figure compares the mean AR values obtained by all 20 CART learners from the control (the solid line), with those obtained from the 10 runs of *Bagging 2N*$_2$ (the dotted lines). The graph at the bottom of the figure does the same for all 20 analogy-based learners to provide a comparison of how stable learners are effected by bagging.

As with *Bagging N* and *Bagging 2N*$_1$, the graph at the top shows that the mean AR values obtained from CART learners during the 10 runs of *Bagging 2N*$_2$ are, for the most part, larger than those obtained by the control. However, when comparing this graph to its equivalents in Figures 5.9 and 5.15, we can see that *Bagging 2N*$_2$ has resulted in an improvement in performance for most of the 20 CART classifiers. In other words, their mean AR values are closer to those of the control. This corresponds to earlier findings in this section where we noticed an improvement in ranking of CART-based classifiers.

Therefore, including all training data has definitely made a difference in CART performance. The fact that CART performance is still not as good as what it was without bagging (i.e. in the control) indicates that having duplicate data in bags does have a detrimental effect on classifier accuracy. We believe that this is due to the duplicates making bags less representative of the Tukutuku dataset. Thus both of the reasons postulated in the previous section play a role in the changes we have seen in CART performance (and by proxy any unstable learner) with bagging.

Analogy-based classifiers, being stable, have not been affected as much by bagging, and we previously saw considerable overlap between bagging and control results. This trend continues with *Bagging 2N*$_2$ where we have the most overlap seen between bagging performance and the control.

The improvement in performance of CART-based classifiers should result in an increase in the number of top ranked solo classifiers. Considering Table 5.14 we can see that this is indeed the case. While the majority of top ranked classifiers were obtained from

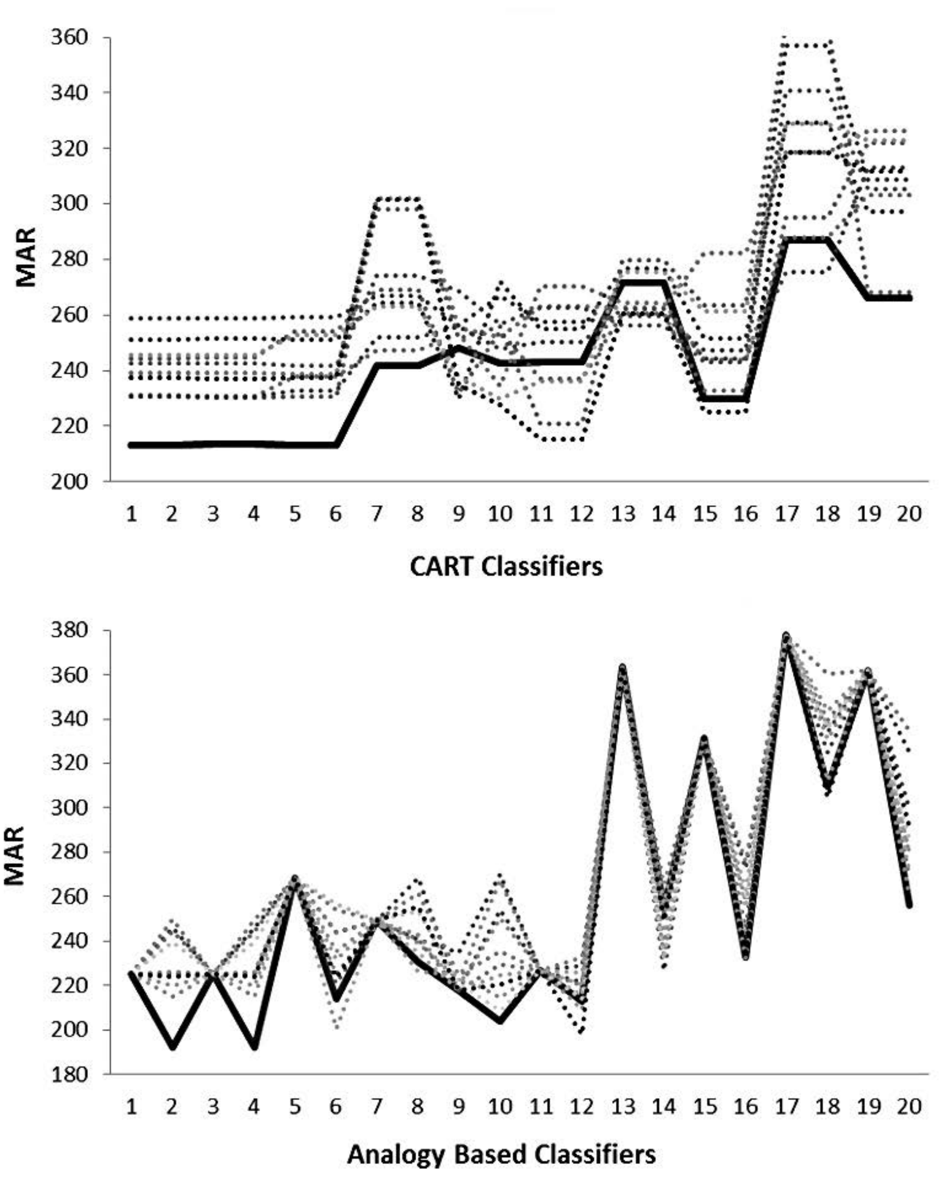


Figure 5.21: Graphs of mean AR values obtained for CART-based classifiers (top) and Analogy-based classifiers (bottom). The mean AR values obtained during the 10 runs of *Bagging 2N*$_2$ (dotted lines) are compared to the mean AR values obtained during the control (solid line).

ensembles, there are clearly more top ranked solo classifiers than there were with *Bagging N* (see Table 5.6) and *Bagging 2N*$_1$ (see Table 5.10). Every run here has more than two top ranked solo classifiers, with runs 6 and 7 having 13 and 12 top ranked solo classifiers respectively.

In summary, *Bagging 2N*$_2$ is just as effective with Web effort estimation ensembles as were *Bagging N* and *Bagging 2N*$_1$. Ensemble classifiers (apart from the mean top positive ensemble yet again) still perform favourably in comparison to their solo counterparts: Most of the top ranked classifiers are obtained from ensembles, and in each run a subset of these top ranked ensembles also have the smallest δr values, indicating their stability

Table 5.14: The number of top ranked solo and ensemble classifiers for each run of *Bagging 2N$_2$*.

Run	Number of Top Ranked Classifiers	
	Solo	Ensemble
1	5	15
2	4	19
3	4	10
4	4	15
5	7	18
6	13	18
7	12	16
8	3	18
9	5	14
10	4	15

within this group. Eliminating the loss of training data due to the bagging process has resulted in an appreciable improvement in performance of CART-based classifiers, in terms of both ranking and mean AR values. Their performance however is still not as good as what was seen in the control, indicating that the inclusion of duplicate data in bags has had a negative effect on their estimation accuracy.

5.3 Discussion

In the previous section we had a look at the results of our three experiments with bagging. We investigated bagging with bags of size N and $2N$. Two variants of the latter were investigated: With *Bagging 2N$_1$*, bags of size $2N$ were created using sampling with replacement. With *Bagging 2N$_2$*, N training cases were selected as they would have with leave-one-out cross-validation (i.e. all data apart from the test case). The remaining N cases were obtained via sampling with replacement to create a $2N$ bag.

Our bagging experiments enabled us to assess ensemble performance over multiple runs; 10 runs for each of the three types of bagging for a total of 30 runs. This was done to provide a more in depth view of ensemble behaviour for Web effort estimation than would otherwise be afforded by leave-one-out cross-validation, which is deterministic in nature. As bagging typically makes less training data available, the three variants of bagging investigated have been chosen to represent different levels of data loss, from *Bagging N* where on average 63.2% of unique training data is present in a bag [6, 12], to no data loss with *Bagging 2N$_2$*.

A total of 20 ensembles were evaluated for each type of bagging. We decided to build ensembles using the best solo classifiers; those ranked in the top, top 2, top 3, top 4 and top 5 rankings. We also decided to evaluate ensembles created with a wider range of solo classifiers; the top positive ensembles were built using all solo classifiers with a positive wins$-$losses count. Lastly as the top solo classifiers vary between runs, we decided to evaluate ensembles consisting of solo classifiers that were ranked in the top third over all 10 runs of a particular type of bagging. In other words, these ensembles would have the same component classifiers over all 10 runs. Mean, median and IRWM combination schemes were utilized in ensemble creation for all of our bagging experiments.

So what do our results say about ensembles for Web effort estimation? We found that over the 30 runs spread across three types of bagging, ensemble classifiers consistently performed well at Web effort estimation using data from the Tukutuku dataset. We believe that this validates our replication results discussed in Chapter 4, and indicates that ensembles are well suited for Web effort estimation, at least when based on the Tukutuku dataset.

What type of ensemble classifiers should practitioners use for Web effort estimation? Looking at the results in the previous section we found that the key to effective Web effort estimation ensembles was for the component classifiers to be accurate. Increasing the diversity of solo classifiers did not seem to improve the performance of the resulting ensembles. The top positive ensembles did not outperform their counterparts, and in fact the mean top positive ensemble was consistently the lowest ranked ensemble over all three types of bagging.

This brings us to the choice of combination scheme. Apart from the mean top positive ensemble, ensemble classifiers performed well regardless of combination scheme used. It is therefore not possible to label any particular combination scheme as being definitive. In terms of losses ranking we can see that the IRWM top 4 and top 5 ensembles are ranked first over all 30 runs of bagging investigated. The mean top 3 and top 4 ensembles are ranked first over 29 of the 30 runs investigated (missing out on top ranking during the third run of *Bagging* $2N_2$). No classifier, solo or ensemble, is top ranked over all 10 runs for the other two ranking systems (wins and wins−losses), although here ensembles using the median combination scheme seem to rank slightly higher.

As all three combination schemes are straight forward to calculate, creating ensembles using one or more of these combination schemes should not prove an issue for practitioners. Care need only be taken when creating an ensemble where there is a wide range of component classifier performance, as is the case with the top positive ensembles. In such a situation we suggest that the mean combination scheme be avoided, for reasons discussed in the previous section.

The top overall ensembles show that a small set of consistently accurate classifiers can be used to create ensembles that perform similarly to those consisting of the top – top 5 groups of classifiers. Therefore if practitioners have an established set of solo classifiers that have proven successful for them in the past, using these classifiers to build an ensemble for Web effort estimation may be an effective strategy.

Lastly, what effect does the bagging process have on estimation performance? We found that unstable solo classifiers like those based on CART perform worse when bagging is used. This is due to two issues: The first issue is related to the fact that the bagging process usually entails a reduction in training data. The second issue deals with duplicate training data in bags, making them less representative of the dataset being used. We found that by negating the first issue with *Bagging 2N*$_2$, we could minimize the effect that bagging has on unstable classifiers. This would be useful for Web effort estimation researchers investigating bagging using unstable learners, and/or a dataset where data loss would be a significant issue overall.

5.4 Conclusion

This chapter investigates the use of bagging when building ensembles for Web effort estimation. With bagging the Tukutuku dataset can be used to investigate Web effort estimation ensembles over multiple runs. 20 ensembles were evaluated over 10 runs for each of the three types of bagging investigated. We found that:

- Web effort estimation ensembles performed consistently well over all runs and all three types of bagging, both in terms of ranking and in terms of performance measured using mean AR. As such we feel that ensembles are a good choice for Web effort estimation.

- Component classifier accuracy seemed to be the key factor for ensemble performance. Ensembles made from the top groups of solo classifiers or from the top overall classifiers performed well regardless of combination scheme used. When there is a wider range of component classifier performance, the mean combination scheme should be avoided. This was evidenced by the mean top positive ensemble being the lowest ranked ensemble over all 30 runs.

- Bagging can adversely effect solo classifier performance, particularly in the case of unstable learners like CART. We attributed this to a decrease in amount of training data as well as duplicate training data in bags. By avoiding data loss with *Bagging 2N*$_2$ we minimized the effect bagging has on unstable learners. Bagging was not an issue for ensemble classifier performance.

In Section 3.2.2 we specified that for an ensemble to be effective, its component classifiers need to be both accurate and diverse [12]. We however found component classifier accuracy to be most important. Our top positive ensembles, those with the widest range, and therefore ostensibly the most diverse set of component classifiers did not perform better than the other ensembles investigated. In fact we found the mean top positive ensemble to be consistently ranked lowest. In the following chapter we will investigate ensemble diversity and its relationship with the results seen.

6 Ensemble Diversity

The general consensus to building an effective ensemble classifier is to choose component classifiers that are both accurate and diverse [12]. An accurate classifier is one whose prediction error on new input data is less than that obtained from random guessing. Diverse classifiers are those that would make different errors on this new data.

In Chapter 3 we gave a simple classification scenario as an example of why accuracy and diversity are important for ensembles. In this scenario we label the input x and the output y, one of two possible classes present in roughly equal amounts. There are n component classifiers with which to build an ensemble using majority voting, where n is odd to avoid ties. These classifiers are labelled $\hat{y}_1 \dots \hat{y}_n$. If there is no diversity among the classifiers then when $\hat{y}_1$ incorrectly classifies x, the remaining classifiers $\hat{y}_2 \dots \hat{y}_n$ will also incorrectly classify x as will the resulting ensemble. If on the other hand the classifiers tend to make different classification decisons, then it is possible for the overall ensemble classification to be correct.

With the above scenario, if the classifiers share the same error rate e, and if they make errors independently from each other, then the probability that a classifier will make an incorrect classification can be modeled with a Binomial distribution, with parameters n and p (being equal to $1 - e$) [12]. The probability that an ensemble from these classifiers makes an incorrect classification would then be equivalent to the probability that more than half of the classifiers make an error. If the individual classifiers are accurate, in this case $e < 0.5$ as there are only two classes present in approximately equal numbers, then this probability would be smaller than the probability that a single classifier makes an error. In other words, the ensemble will be more accurate than its individual classifiers. If, on the other hand, the classifiers are not accurate, the resulting ensemble would be less accurate than its individual classifiers.

This brings us to the accuracy-diversity trade-off [15, 16]. For component classifiers of an ensemble to be diverse, some of these classifiers must make mistakes. As the accuracy of these classifiers increases, the likelihood of mistakes being made decreases, resulting in less diversity. Ideally, there should be a balance between component classifier accuracy and diversity.

In the previous chapter we found that when it came to ensemble performance for Web effort estimation, a regression problem, component classifier accuracy seemed to be more important than diversity. The ensembles with the most diverse set of classifiers, the top positive ensembles, did not perform better than their less diverse counterparts. In fact, the mean top positive ensemble was consistently the lowest ranked ensemble whether or not bagging was used.

In this chapter we will look at analyzing ensemble diversity to try and obtain a better understanding of these results. We will first formalize the accuracy-diversity trade-off just discussed, before using this to calculate and analyze ensemble diversity. We will conclude the chapter with a discussion of our findings.

6.1 The Accuracy-Diversity Trade-Off

In order to analyze the relationship between ensemble accuracy and diversity, we need to be able to formalize and quantify the accuracy-diversity trade-off. Krogh and Vedelsby have done this for neural network ensembles [22] and we will utilize their research here. Before we discuss this formalization, we will revisit the concept of errors with regards to Web effort estimation and in machine learning in general.

As seen in Section 2.2.1, error measures are used as numerical indicators of estimation accuracy. Classifiers which produce smaller errors when estimating development effort are preferred to those with larger errors. Formally we can define error, also referred to as the residual or loss, as a function that calculates the cost of making an estimate of $\hat{y}$ when the actual value is y [15]. The error function we used in our bagging experiments discussed in the previous chapter is the absolute error which can be written as follows:

$$E_{||}(\hat{y}, y) = |\hat{y} - y| \tag{6.1}$$

Krogh and Vedelsby have centred their formalization of the accuracy-diversity trade-off on the square error function [22] which can be defined as follows:

$$E_2(\hat{y}, y) = (\hat{y} - y)^2 \tag{6.2}$$

What this means is that all the diversity analysis that follows in this chapter will use the square error function. Despite the change from using the absolute error function in our bagging experiments, we believe that our findings here using the square error function will still be useful in giving us insight into the results of our bagging experiments. The reasons for this are two-fold: Firstly, the absolute error function and the square error function behave very similarly; both functions always produce a positive output. An increase in absolute error would correspond to an increase in square error, albeit a greater one. The same holds true for a decrease in absolute error. The second reason is that, as discussed previously, the trade-off between component classifier accuracy and diversity is assumed to apply to ensemble performance in general, and is therefore thought to extend to other loss functions as well [15].

Krogh and Vedelsby expressed the square error function of an ensemble (E_2), in terms of the mean error of the component classifiers ($\bar{E_2}$) and the mean diversity of the component classifiers ($\bar{D_2}$) as follows [22, 15]:

$$E_2 = \bar{E_2} - \bar{D_2} \tag{6.3}$$

The mean error of component classifiers is calculated as:

$$\bar{E_2} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y)^2 \tag{6.4}$$

and component classifier diversity can be expressed as:

$$\bar{D_2} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - \hat{y})^2 \tag{6.5}$$

where n is the number of component classifiers in the ensemble, $\hat{y_i}$ is the estimate made by the i^{th} classifier, $\hat{y}$ is the ensemble estimate and y is the actual value.

The mathematical identity describing the accuracy-diversity trade-off in Formula 6.3 only holds true for ensembles created using a weighted mean combination scheme [22], where the weights are positive and sum up to one (i.e. $W_1 + W_2 + \cdots + W_n = 1$). We are therefore not able to use this formalization on ensembles created using the median and IRWM combination schemes, the former for obvious reasons and the latter because the weights used do not sum up to one.

We can however use this formalization on ensembles using the mean combination scheme as this is simply the case of each of the n component classifiers having identical weights of $W = \frac{1}{n}$, the proof of which is shown on the following page.

Proof. Given that $W_1, W_2, \ldots, W_n = \frac{1}{n}$,

$$\begin{aligned} \texttt{Weighted Mean} &= \frac{\sum_{i=1}^{n} W_i X_i}{\sum_{i=1}^{n} W_i} \\ &= \frac{W \sum_{i=1}^{n} X_i}{nW} \\ &= \frac{\sum_{i=1}^{n} X_i}{n} \\ &= \texttt{Mean} \end{aligned}$$

□

Using the results obtained from our bagging experiments described in the previous chapter, we will analyze ensemble error in terms of component classifier accuracy and diversity using the formulae just discussed. The results of this analysis are detailed in the subsequent section.

6.2 Results

In the following section we present the results of our analysis of the accuracy-diversity trade-off. As we are using the formalization specified by Krogh and Vedelsby [22], our analysis will be restricted to ensembles created with the mean combination scheme, and ensemble error will be evaluated using the square error function. As such we are not expecting a one to one correspondence between the diversity analysis here, and the ensemble results discussed in the previous chapter.

The purpose of the following analysis is to provide a general idea of how our ensemble results can be related to component classifier performance and diversity. We believe that this analysis will enable us to better understand why, instead of both component classifier accuracy and diversity playing a role in ensemble performance, our results showed the former to be the most important factor. We believe that this would be useful to inform any future research on using ensembles for Web effort estimation.

Results for the diversity analysis of ensembles will be presented for all three types of bagging investigated. All of the graphs in this section use the same legend, provided in Figure 6.1, to represent the different ensembles.

Looking at Table 3.2, it can be seen that a wide range of development effort values are being estimated. As such, the associated error and diversity values obtained also vary widely, something which is compounded by the square error function. Therefore, the ensemble error, mean component classifier error, and diversity values of all the graphs in this section have undergone a natural logarithm transformation.

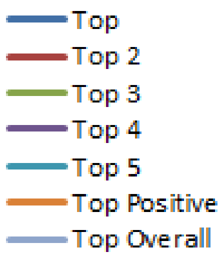


Figure 6.1: Legend representing the seven ensembles used in the analysis.

6.2.1 Bagging N

The accuracy-diversity trade-off results have been summarized in three graphs: Figure 6.2 displays diversity values for all ensembles over all 10 runs of *Bagging N*. Diversity values were calculated using the ensemble results obtained for *Bagging N* and Formula 6.5. Figure 6.3 displays the mean component classifier error for all ensembles over all 10 runs of *Bagging N*. The mean component classifier error was calculated using the same *Bagging N* data and Formula 6.4. Figure 6.4 displays the ensemble error for all ensembles over all 10 runs of *Bagging N*. As discussed in the previous section, ensemble error can be expressed as the difference between mean component classifier error and diversity, as shown in Formula 6.3.

In terms of diversity, from Figure 6.2, we can see that the graphs have a similar general shape: Differences in diversity between the ensembles is greater at the lower percentiles, with the differences gradually converging at the 100th percentile. It therefore appears that the Web effort estimation ensembles have a similar upper limit (maximal diversity), with most of the differences being visible at the opposite end of the spectrum (lower diversity values).

Intuitively we would expect the Top Positive ensemble to have the largest diversity; by including all classifiers with a positive wins$-$losses count, it should have the greatest number of component classifiers and therefore the widest range of classifier performance. Conversely, we would expect the Top and Top Overall ensembles to have the lowest diversity; by only including the top classifiers or those ranked in the top third over all 10 runs respectively, these two ensembles would have a small number of component classifiers of similar performance.

Looking at the graphs in Figure 6.2 and using the legend in Figure 6.1, we can see that this is indeed the case. The Top Positive ensemble (49–54 component classifiers) has the largest diversity in each of the 10 runs. The Top Overall ensemble (6 component classifiers) has the lowest diversity in 5 runs (Runs 1, 2, 3, 5, and 10), while the Top ensemble (2–14 component classifiers) has the lowest diversity in 4 runs (Runs 4, 6, 7, and 8). In Run 9 both of these ensembles share the honors of having the lowest diversity.

However the graphs in Figure 6.2 show that changes in diversity are not as straight-

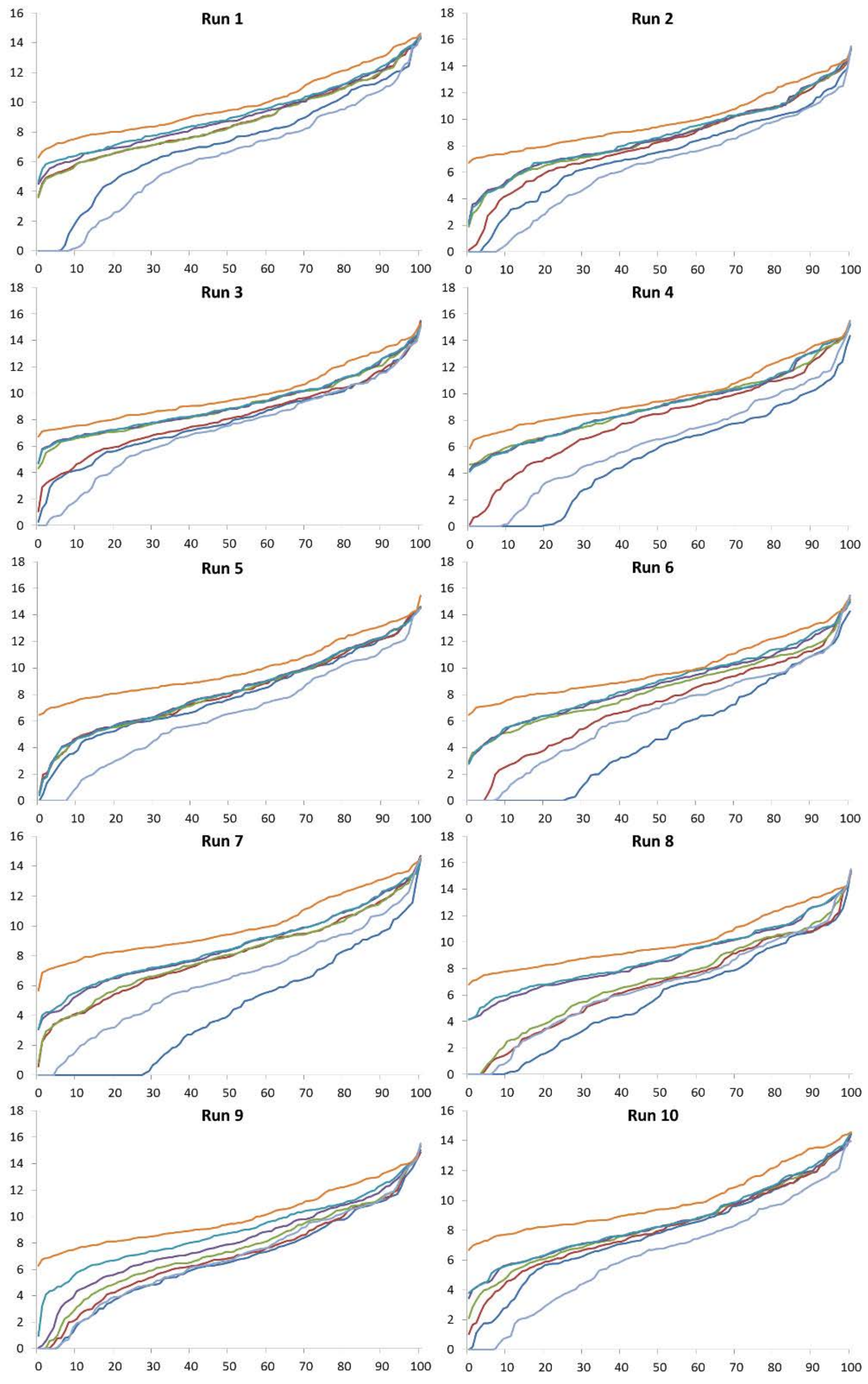


Figure 6.2: Range of diversity values for all ensembles over all 10 runs of *Bagging N*. For each graph the y-axis represents diversity evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

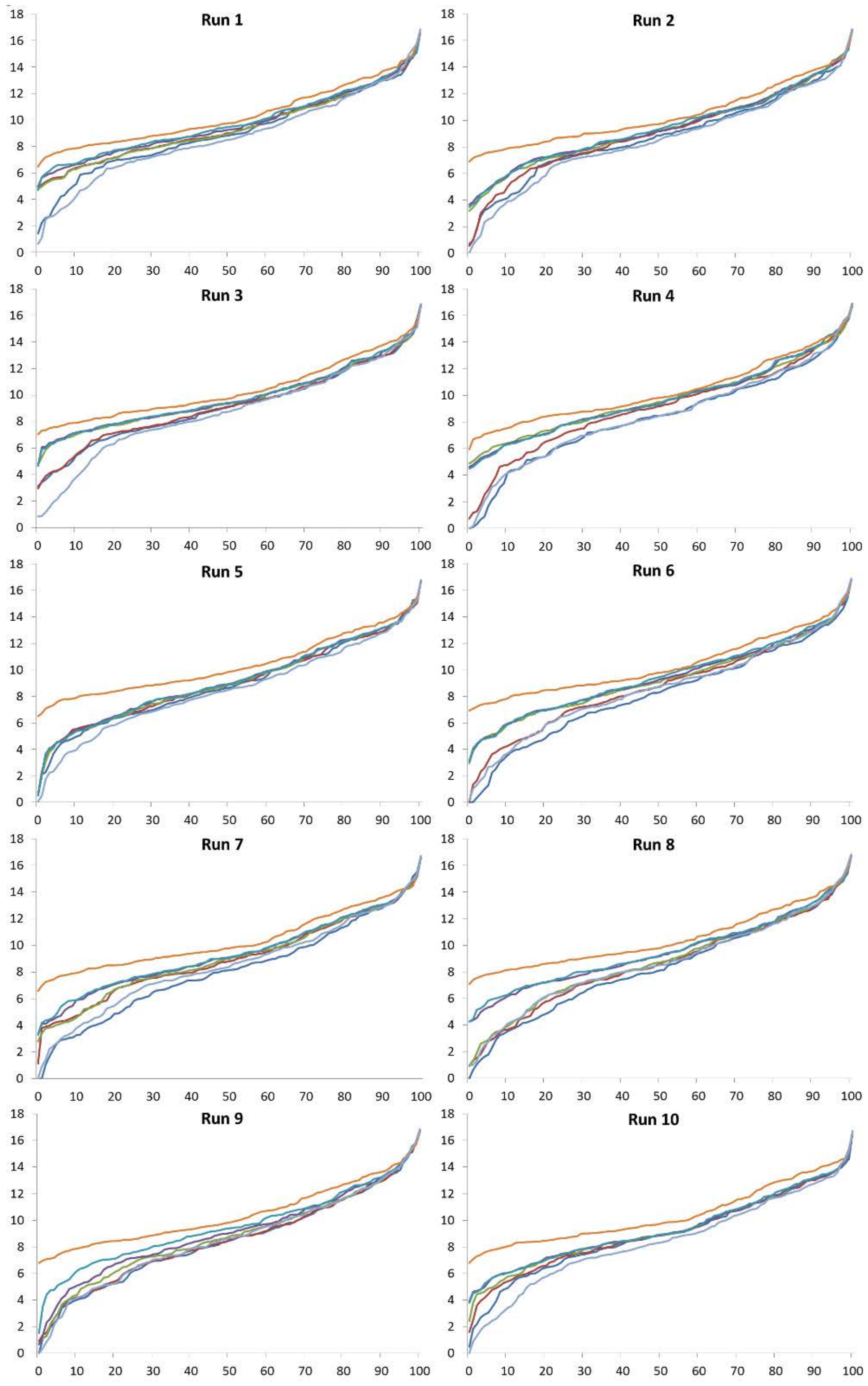


Figure 6.3: Range of mean member error values for all ensembles over all 10 runs of *Bagging N*. For each graph the y-axis represents mean member error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

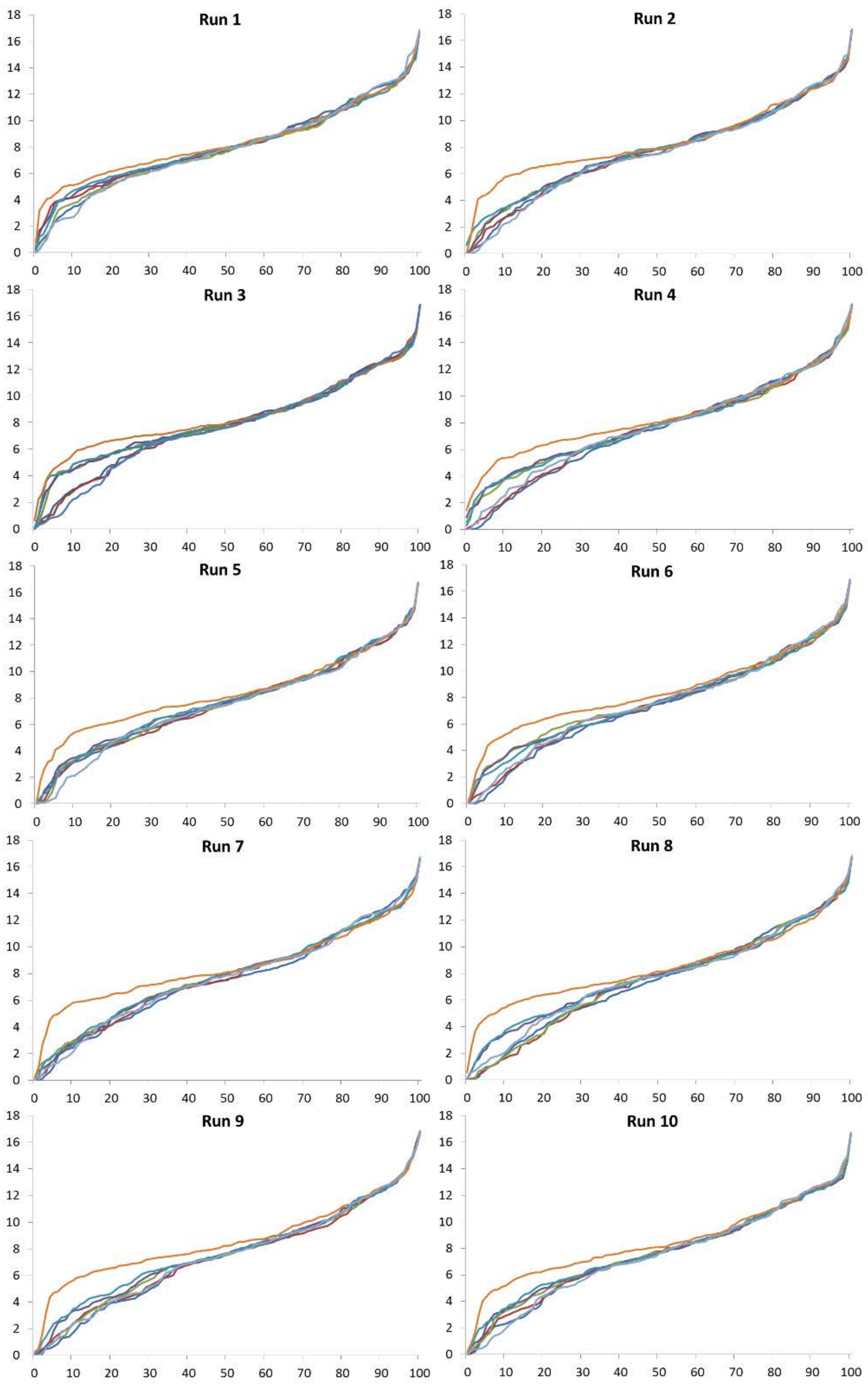


Figure 6.4: Range of ensemble error values for all ensembles over all 10 runs of *Bagging N*. For each graph the y-axis represents ensemble error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

forward as we move from the Top ensemble through to the Top 5 ensemble. While the number of component classifiers increases, the amount by which they do so varies from run to run and can be as small as a single classifier.

For example, in Run 1 the Top 2 and Top 3 ensembles differ by a single classifier, and the resulting two ensembles have a very similar diversity. In Run 6 on the other hand, the Top and Top 2 ensembles differ by five classifiers with the Top 2 ensemble clearly having a larger diversity.

Increasing the number of component classifiers is not the sole factor related to increasing diversity. The characteristics of the component classifiers is important as well. If we add classifiers to an ensemble that are very similar to each other, and/or similar to classifiers already present in the ensemble, the resulting change in diversity may be far less than expected.

For example in Run 2, the Top 3, Top 4 and Top 5 ensembles have a very similar diversity. The component classifier counts are 16, 18, and 24 respectively. The Top 3 and Top 4 ensembles differ by only two classifiers, both variants of analogy based learning with 5 nearest neighbours, so the two ensembles having a similar diversity is not unexpected. The Top 4 and Top 5 ensembles however differ by six classifiers: pruned and unpruned variants of CART with no pre-processor, the normalization pre-processor and the natural logarithm pre-processor. In addition to all six classifiers being variants of CART, in this particular run their effort estimates were virtually identical. This would explain why the Top 5 ensemble has a very similar diversity to the Top 4 ensemble despite what appears to be a sizeable increase in the number of component classifiers.

Moving on to mean component classifier error, we can see that the graphs in Figure 6.3 have a similar shape over the 10 runs of *Bagging N*. As was the case with the diversity graphs, differences in mean member error are largest at the lowest percentiles gradually converging as we approach the 100th percentile. Thus, Web effort estimation ensembles also appear to have a similar upper limit for mean member error, with differences being more noticeable when dealing with smaller values.

Due to the accuracy-diversity trade-off we expect the results for mean component classifier error to be similar to those of diversity when comparing the different ensembles. With the Top and Top Overall ensembles, by choosing a set small set of classifiers, all of which were chosen for their accuracy, we would expect these classifiers to have a low mean member error in addition to a low diversity. With the Top Positive ensemble, by including a large set of classifiers of varying accuracy, we would expect this ensemble to have a larger mean member error in addition to its higher diversity. As we go through the ensembles from the Top ensemble to the Top 5 ensemble, mean member error, like diversity, should change based on the number of new classifiers added as well as their characteristics.

Looking at the graphs in Figure 6.3 and using the legend in Figure 6.1, we can see that mean component classifier error does behave similarly to diversity. This is particularly true of the mean member error for the Top Positive ensemble, which has the highest range of mean member error values over all 10 runs. There are however, a couple of notable differences.

The range of mean member error values for an ensemble increases more rapidly than the corresponding range of diversity values. This is most obvious when looking at the Top and Top Overall ensembles: In all runs, one or both of these ensembles have a diversity of (or close to) 0 for part of their range of diversity values. On the other hand their mean member error graphs start at (or close to) 0, increasing rapidly instead of sustaining these low values. This is to be expected given that the ensembles analyzed have errors greater than 0; using Formula 6.3 for ensemble error, mean member error would have to be larger than diversity for this to occur.

As we go from the ensemble with the lowest range of mean member error values to the ensemble with the highest range of mean member error values, we can see that the differences in these ranges is less pronounced than it was when looking at diversity values. This holds true for all 10 runs. We believe that there are two reasons for this. Firstly, member classifiers for an ensemble were picked solely on the basis of their estimation performance; diversity was not a factor in this selection. Secondly, with *Bagging N*, correlation between training sets is at its lowest with training sets on average sharing around 63.2% of the data. Taking both of these factors into consideration, it makes sense for differences in diversity to be more pronounced than the differences in mean member error. We would also expect this difference to decrease with *Bagging 2*N_1 and *Bagging 2*N_2, as increasing the correlation between training sets should result in ensembles having more similar diversity values.

Lastly, we will look at the range of ensemble errors obtained as displayed by the graphs in Figure 6.4. We can see that these graphs are quite similar over the 10 runs with differences in ensemble error being more pronounced for smaller error values (at the lower percentiles), gradually converging as we approach the 100th percentile. Given that differences in diversity and mean component classifier error are also more pronounced at the lower percentiles, it is no surprise that this is the case for ensemble error as well. When it comes to the largest errors, the Web effort ensembles analyzed perform similarly.

We can make two observations from these graphs that relate to the results obtained in the *Bagging N* experiment discussed in the previous chapter:

- Ensemble errors for the Top–Top 5 ensembles as well as the Top Overall ensemble are similar over most of their range.

- Ensemble error for the Top Positive ensemble is clearly larger than that of the other ensembles up to at least around the 50th percentile.

In Section 5.2.2 we saw that the Top–Top 5 ensembles as well as the Top Overall ensemble performed similarly well, ranking in the top third of classifiers over all 10 runs. This coincides with our first observation where these ensembles have similar ensemble error values over most of their range.

The Top Positive ensemble was found to be have the lowest average ranking over the 10 runs of *Bagging N* and was the only ensemble to be ranked outside of the top third, doing so in runs 5, 6, 9 and 10. This fits with our second observation where the Top Positive ensemble has error values that are larger than those of the other ensembles for at least half of the range of error values obtained. In fact, looking at the graphs for runs 5, 6, 9, and 10 in Figure 6.4, we can see that the range of error values for the Top Positive ensemble is higher than those of the other ensembles for more than half of their range. This is particularly true for runs 6, 9, and 10 where the Top Positive ensemble has the lowest rankings out of the 10 runs.

The graphs in Figure 6.4 also show that the Web effort estimation ensembles analyzed in our *Bagging N* experiment are not "well behaved". With such ensembles, ensemble loss would decrease as the number of component classifiers increases [15]. Despite the number of component classifiers increasing as we go from Top $\rightarrow$ Top 2 $\rightarrow$ Top 3 $\rightarrow$ Top 4 $\rightarrow$ Top 5 $\rightarrow$ Top Positive ensembles, any associated increase in diversity does not appear to be enough to offset the increase in mean member error. This leads to the ensembles performing similarly to one another, or in the case of the Top Positive ensemble, peforming worse than the rest.

6.2.2 Bagging 2N$_1$

Using the ensemble results obtained from our *Bagging 2N*$_1$ experiment, we have calculated the diversity, mean component classifier error and ensemble error for all ensembles over all 10 runs. These accuracy-diversity trade-off results have been summarized in the graphs in Figures 6.5, 6.6, and 6.7 respectively.

In terms of diversity, from Figure 6.5, we can see that the graphs have a similar general shape: Differences in diversity between the ensembles is greater at the lower percentiles, with the differences gradually converging at the 100th percentile. It therefore appears that the Web effort estimation ensembles have a similar upper limit (maximal diversity), with most of the differences being visible at the opposite end of the spectrum (lower diversity values). These diversity results coincide with those obtained for *Bagging N*.

Unlike the results obtained for *Bagging N*, the differences in diversity values between the different ensembles are less pronounced here. As discussed previously with *Bagging N*, the training sets contain on average 63.2% of unique data [6, 12] reducing the correlation between training sets. By increasing the amount of data the training sets share, *Bagging* $\mathit{2}N_1$ increases the correlation between training sets, which we believe results in the different ensembles having more similar diversity values.

The Top Positive ensemble once again has the largest diversity in each of the 10 runs, which is expected given this ensemble has the greatest number of component classifiers (51 – 53 component classifiers), with the widest range of classifier performance.

On the opposite end of the spectrum the Top ensemble (3 – 8 component classifiers) has the lowest diversity in seven out of the 10 runs (Runs 2, 3, 4, 5, 6, 8, and 10). With runs 1, 7, and 9, the Top ensemble shares the lowest diversity with the Top 2 ensemble (Run 1), the Top 2 and Top Overall ensemble (Run 7), and the Top 2 and the Top 3 ensemble (Run 9). Once again we believe that the Top ensemble tends to have the lowest diversity because it contains a small selection of similarly accurate classifiers.

From these results it can be seen that the diversity of the Top Overall ensemble (7 component classifiers) is different here than it was with *Bagging N*, where it was found to have the lowest diversity in 6 runs . The Top Overall ensemble with *Bagging* $\mathit{2}N_1$ is quite similar in terms of component classifiers as its *Bagging N* counterpart. It, in fact, only differs by a single classifier: linear regression with a sequential forward feature selection pre-processor. We believe that there are two reasons why this classifier has caused the Top Overall ensemble to have a larger diversity than what was seen previously with *Bagging N*:

1. Despite being in the top third of classifiers over all 10 runs and 3 ranking schemes, this classifier is ranked significantly lower (average losses ranking of 20.6) than the other six classifiers that consitute the Top Overall ensemble (all analogy-based learners with average losses rankings of between 2.2 and 4.9). In other words, there is now a wider range of performance amongst the constituent classifers.

2. As discussed in Section 5.2.2, the implementation of the pre-processor involves random partitioning of the training set when evaluating potential subsets of features. Therefore even without bagging, identical training sets could result in a different subset of features being selected. This additional layer of instability could also explain the increase in diversity seen with the Top Overall ensemble.

Just as we saw with *Bagging N*, changes in diversity are not as straightforward as we move from the Top ensemble through to the Top 5 ensemble: While the number of component classifiers increases, the amount by which they do so varies from run to run, ranging from a single classifier up to eight classifiers. For example in Run 1, the Top 2 and Top ensembles differ by only a single classifier. As a result, the range of diversity values for these two ensembles is very similar. On the other hand in Run 4, the Top 2 ensemble is composed of six more classifiers than the Top ensemble. As a result, the range

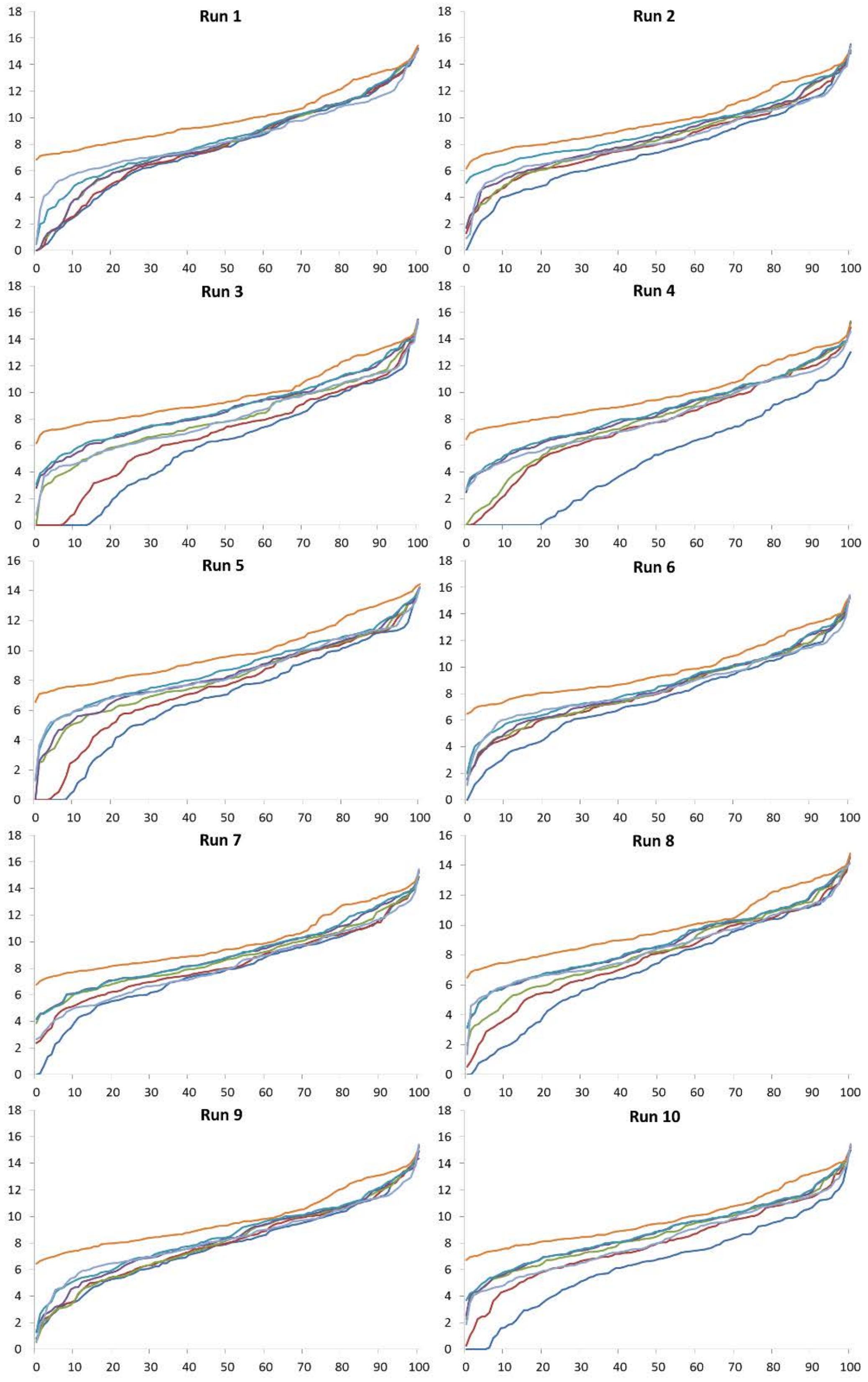


Figure 6.5: Range of diversity values for all ensembles over all 10 runs of *Bagging 2N*$_1$. For each graph the y-axis represents diversity evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

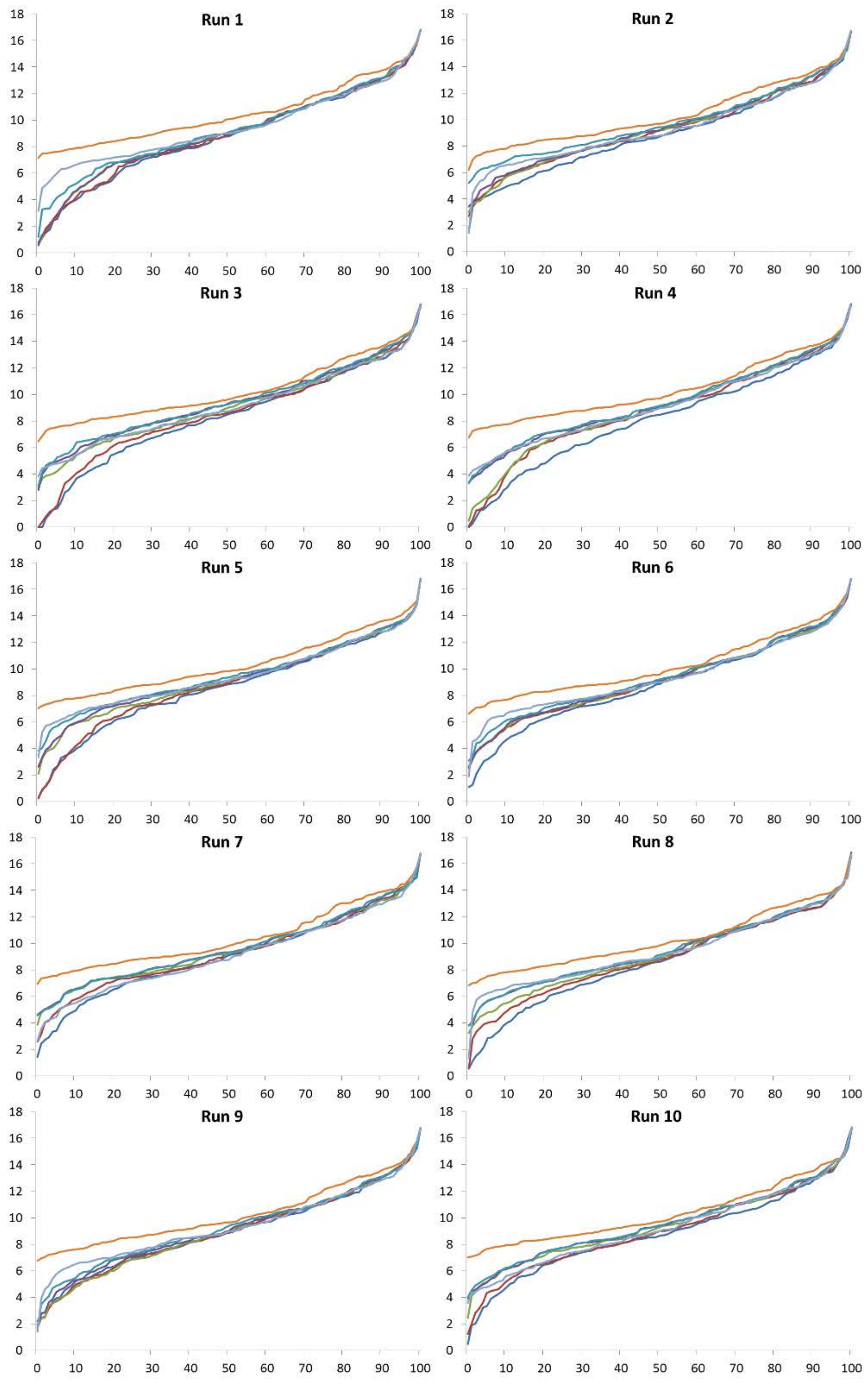


Figure 6.6: Range of mean member error values for all ensembles over all 10 runs of *Bagging 2N*$_1$. For each graph the y-axis represents mean member error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

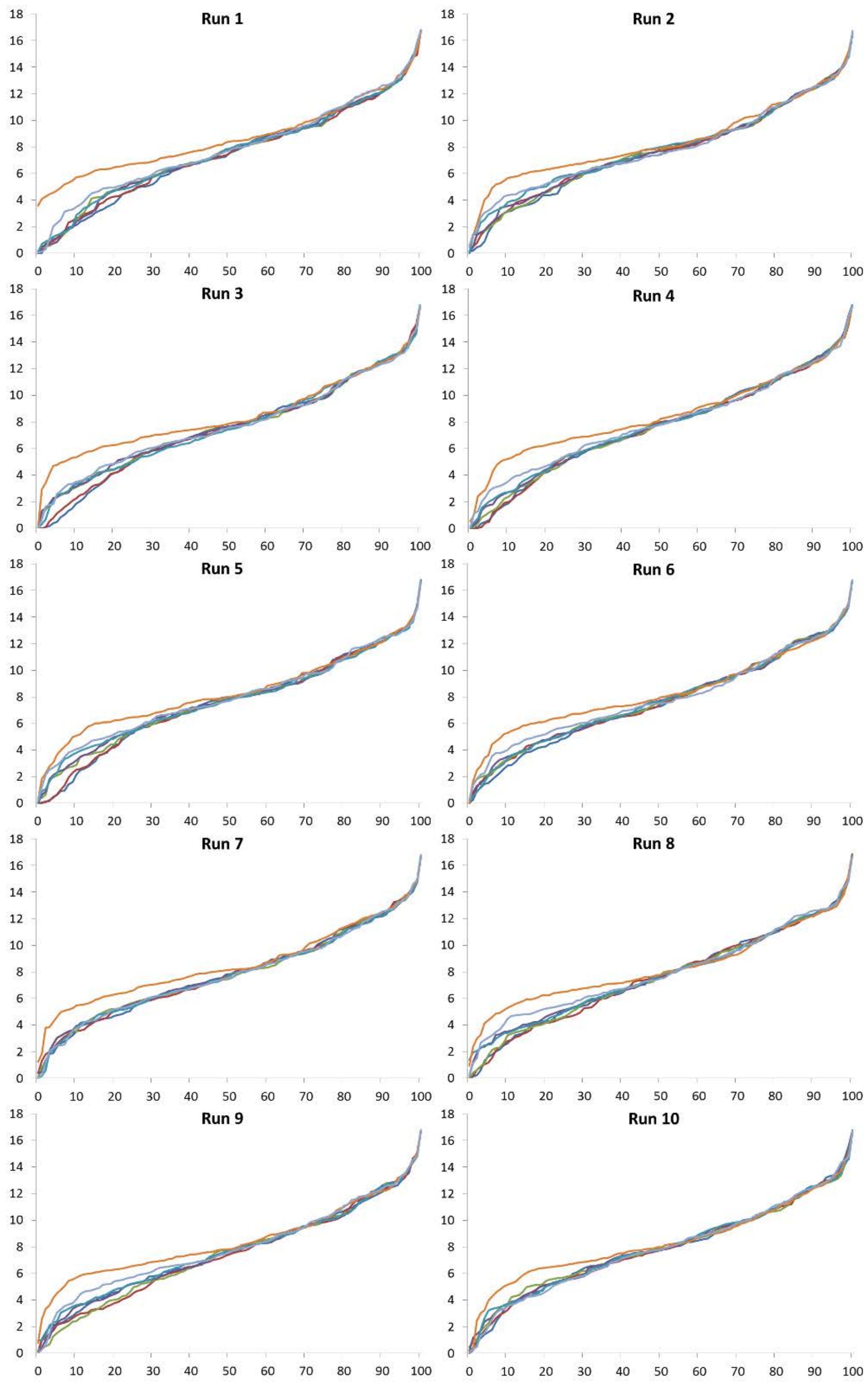


Figure 6.7: Range of ensemble error values for all ensembles over all 10 runs of *Bagging 2*N_1. For each graph the y-axis represents ensemble error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

of diversity values seen for the Top 2 ensemble are clearly higher than those seen for the Top ensemble.

As was the case with *Bagging N*, the number of component classifiers is not the only factor related to ensemble diversity. The characteristics of the component classifiers plays a role as well. For example looking at Run 3, the range of diversity values for the Top 4 and Top 5 ensembles are very similar despite the Top 5 ensemble also having six more member classifiers. This is due to the fact that four of these six classifiers are CART variants that, in this particular run, have very similar estimates to each other, and to the two CART classifiers already present in the Top 4 ensemble.

Looking at mean component classifier error, as shown in the graphs in Figure 6.6, we can see that the results obtained are very similar to those obtained with *Bagging N*:

- The graphs, displaying the range of mean member error values, all have a similar shape over all 10 runs of *Bagging* $2N_1$.
- As dictated by the accuracy-diversity trade-off, ensemble mean member error behaves similarly to that of ensemble diversity. The Top Positive ensemble has the largest range of mean member error values (over all 10 runs), the Top ensemble has the lowest range of mean member error values (over most runs), with the range of mean member error values for the other ensembles lying somewhere in between.
- The differences between the range of mean member error values for the various ensembles during a run, is less pronounced than it is for the range of diversity values. This holds true for all 10 runs. This observation is more subtle in comparison to that seen with *Bagging N*, which is to be expected given the greater correlation between training sets with *Bagging* $2N_1$.
- The differences in mean member error are more obvious at lower percentiles, gradually converging at the 100th percentile. This indicates a possible upper limit for mean member error.
- The range of mean member error values increases more rapidly than the corresponding range of diversity values. Once again this is most obvious when looking at the lower percentiles.

Finally, we will analyze the range of ensemble errors obtained as displayed by the graphs in Figure 6.7. As was the case with the mean member errors discussed previously, the ensemble error results obtained with *Bagging* $2N_1$ are similar to those obtained with *Bagging N*. All ensembles apart from the Top Positive ensemble are ranked in the top third over all 10 runs over all three ranking systems. The latter is the only ensemble to be ranked in the middle third (in half the runs), and clearly has the lowest average losses ranking. Therefore we once again see:

- Ensemble errors for the Top–Top 5 ensembles as well as the Top Overall ensemble are similar over most of their range.
- Ensemble error for the Top Positive ensemble is larger than that of the other ensembles up to at least around the 50th percentile.

In other words, the ensembles created using *Bagging* $2N_1$ are also not well behaved. Any associated increase in diversity seen in the ensembles is not enough to offset the increase in mean member error. This results in the ensembles performing similarly to one another, or in the case of the Top Positive ensemble peforming worse than the rest.

Overall, the diversity analysis of *Bagging* $2N_1$ has given us results similar to those seen with *Bagging N*. This is not surprising considering the fact that the ensembles for these two versions of bagging perform very similarly (see Section 5.2.3). In the next section we will look at the results obtained when performing our diversity analysis on *Bagging* $2N_2$.

6.2.3 Bagging $2N_2$

In this section we will look at the diversity, mean component classifier error, and ensemble error for *Bagging* $2N_2$. A summary of these accuracy-diversity trade-off results for all ensembles over all 10 runs is provided by the graphs in Figures 6.8, 6.9, and 6.10. As with the corresponding graphs for the other versions of bagging investigated, the graphs here use the legend shown in Figure 6.1.

Looking at diversity first, we can see that the graphs in Figure 6.8 have a similar general shape to each other, as well as to those obtained when measuring diversity with *Bagging N* and *Bagging* $2N_1$. Once again we see that differences in diversity between ensembles is greater at lower percentiles with diversity values gradually converging at the 100th percentile, indicating a maximal diversity for all ensembles within a run.

It can be seen from the graphs that the differences in diversity values between the ensembles (with the exception of the Top Positive ensemble), are less pronounced than what was seen with *Bagging* $2N_1$ and *Bagging N*. We believe that this is due to the fact that out of the three versions of bagging investigated, *Bagging* $2N_2$ is the only version where all training data is included, resulting in the largest correlation between training sets.

The Top Positive ensemble predictably has the largest diversity in each of the 10 runs, given that it has the greatest number of component classifiers (49 – 52 component classifiers) with the widest range of classifier performance.

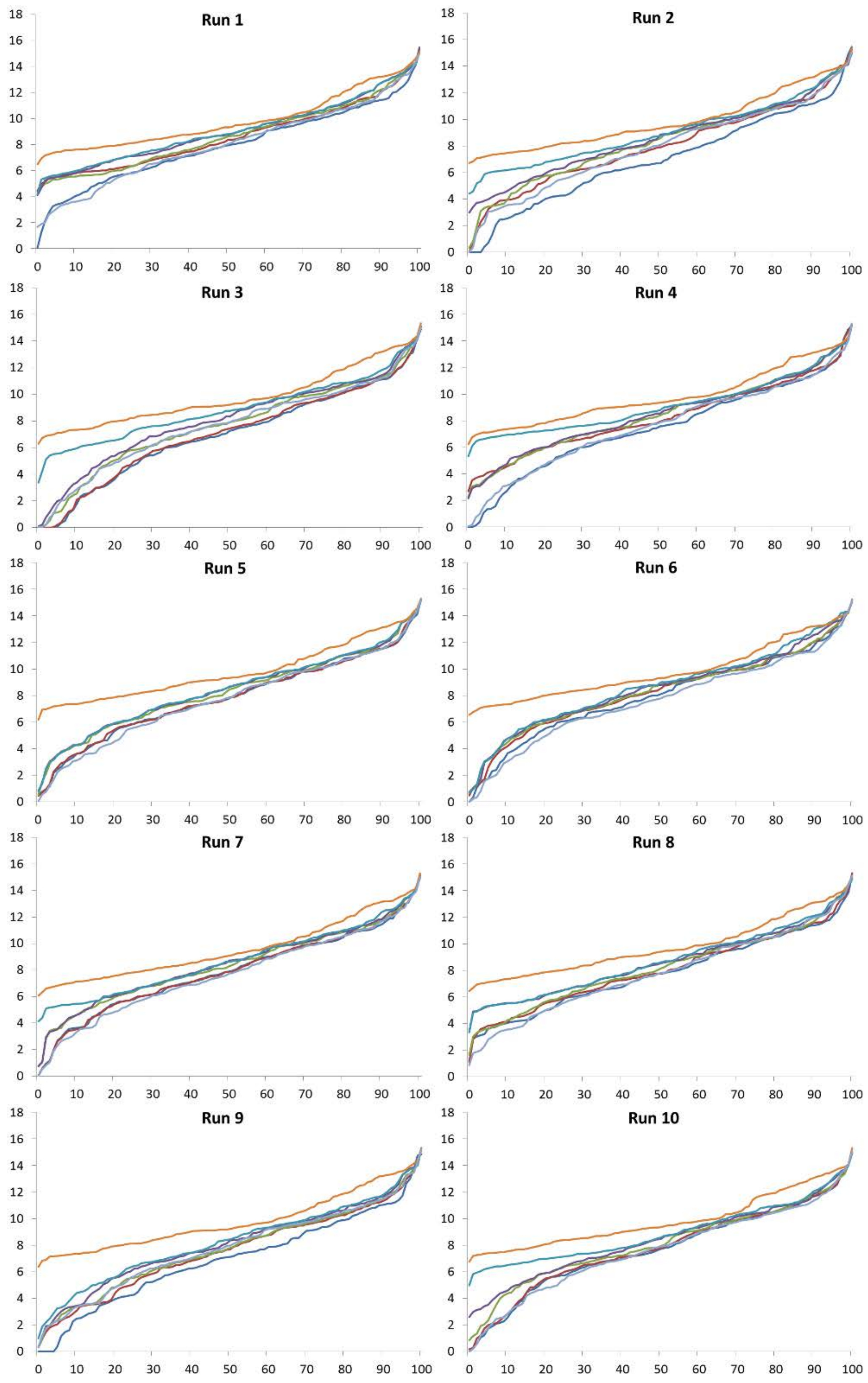


Figure 6.8: Range of diversity values for all ensembles over all 10 runs of *Bagging 2N*$_2$. For each graph the y-axis represents diversity evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

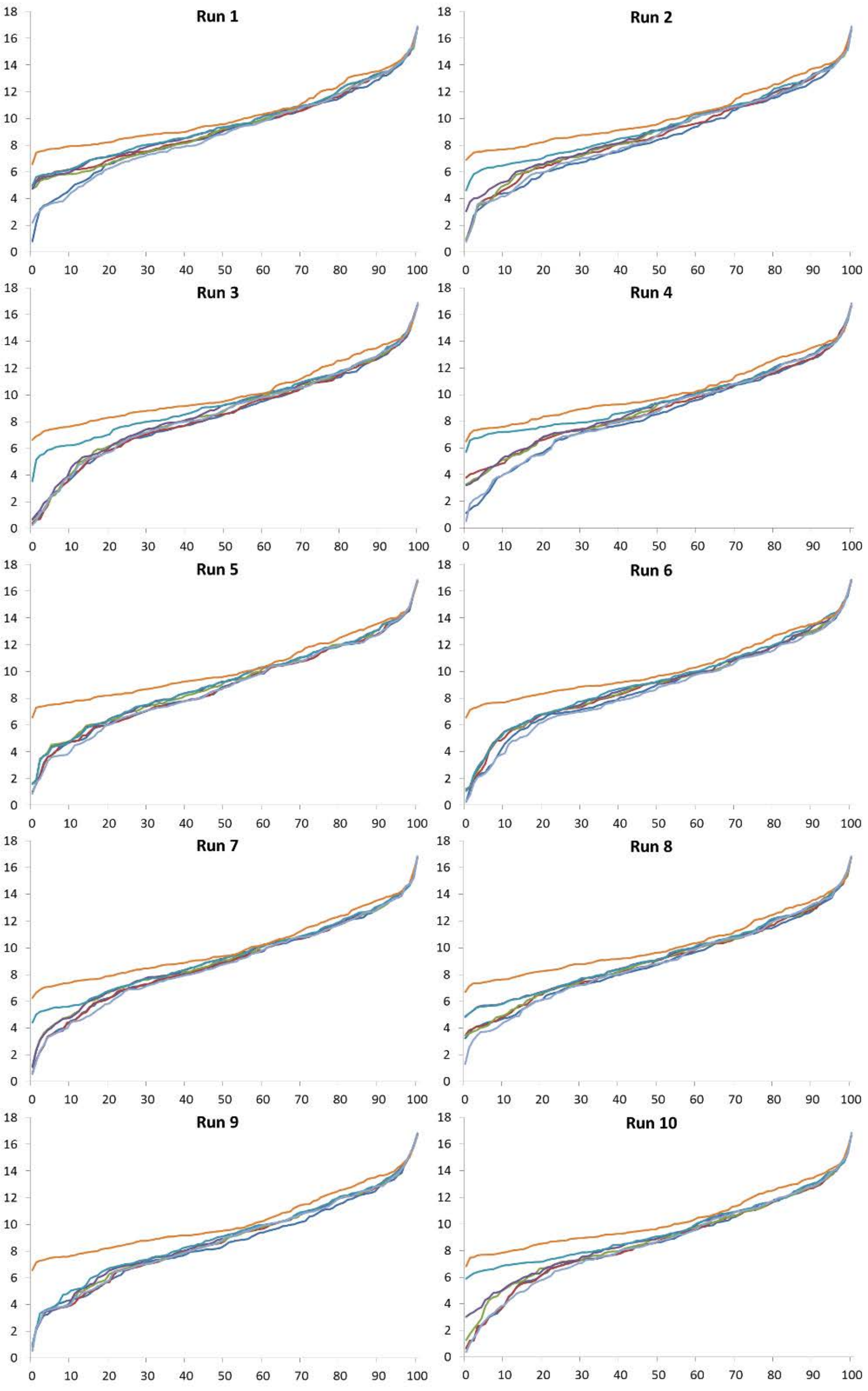


Figure 6.9: Range of mean member error values for all ensembles over all 10 runs of *Bagging 2N*$_2$. For each graph the y-axis represents mean member error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

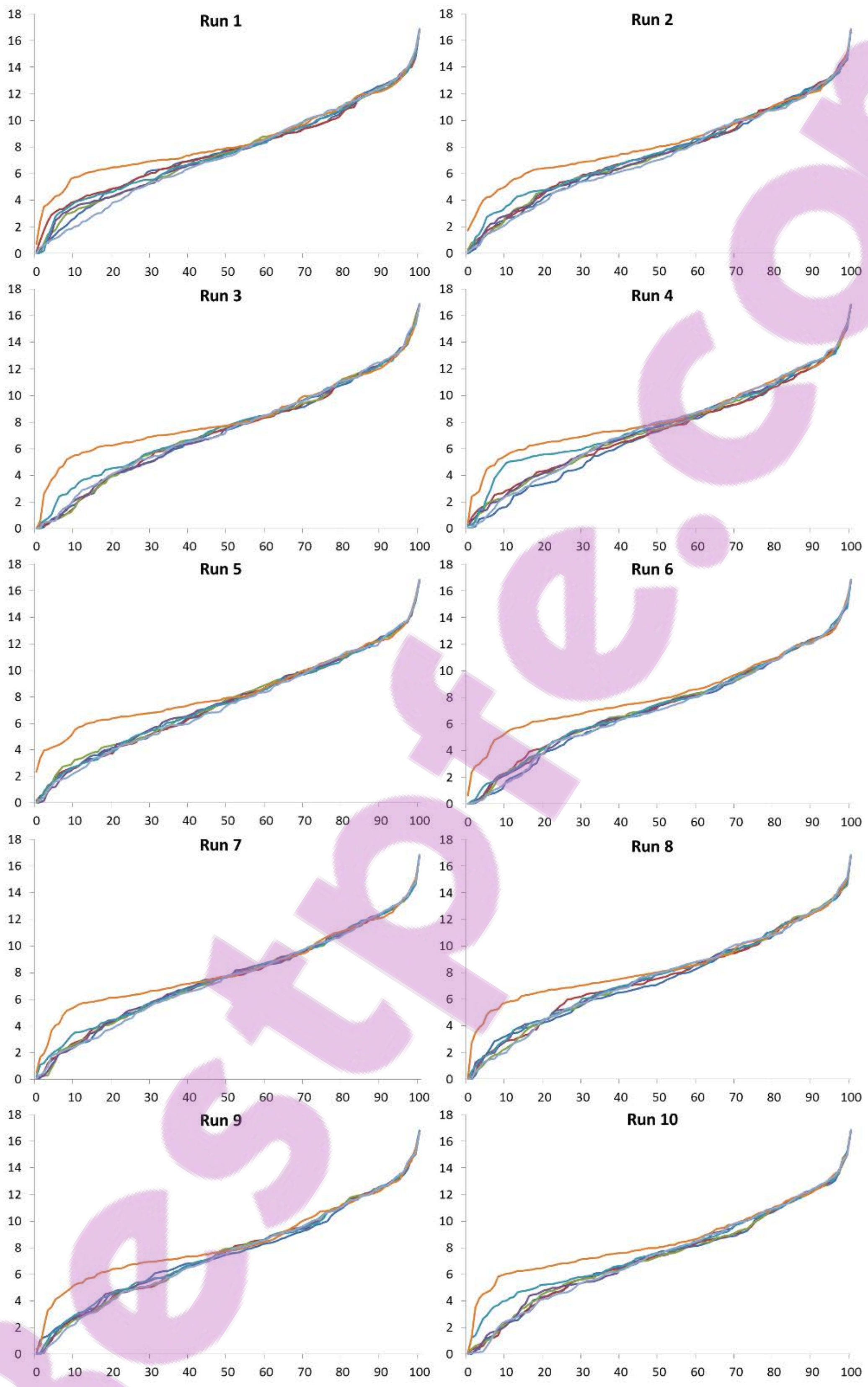


Figure 6.10: Range of ensemble error values for all ensembles over all 10 runs of *Bagging 2N*$_2$. For each graph the y-axis represents ensemble error evaluated using the square error function (transformed using the natural log), and the x-axis the percentile.

On the other hand, three ensembles are associated with having the lowest diversity: the Top ($6-18$ component classifiers), Top 2 ($8-20$ component classifiers) and Top Overall (14 component classifiers) ensembles. The Top ensemble has the lowest diversity in Runs 2 and 9. The Top Overall ensemble has the lowest diversity in Run 6. The Top and Top 2 ensembles share the lowest diversity in Run 3, while the Top and Top Overall ensembles share the lowest diversity in Runs 1 and 4. In the remaining runs (5, 7, 8, and 10) all three ensembles share the lowest diversity.

On closer analysis of these three ensembles we found that they have very similar component classifiers that are:

- Only obtained from variants of analogy-based learners or CART.
- Top ranked, with all of them being in the top third of classifiers when ranked based on losses.

The low diversity of these three ensembles can therefore be attributed to their member classifiers being similar in both characteristics and performance.

As was the case with *Bagging N* and *Bagging* $2N_1$, changes in diversity between the various ensembles is dependent upon both the change in the number of component classifiers and their characteristics. An example of the former can be seen in Run 9, where the Top 2 ensemble has 6 more member classifiers than the Top ensemble. As a result, the range of diversity values seen for the Top 2 ensemble is higher than that observed for the Top ensemble. An example of the latter can be seen in Run 4. Here the Top 2 ensemble only has 1 more member classifier than the Top ensemble and still clearly has a larger range of diversity values. This is due to the fact that the additional classifier (CART with pruning using a sequential forward feature selection pre-processor) made effort estimates that were very different from those made by the existing member classifiers.

In terms of mean member error, the graphs in Figure 6.9 are very similar to their *Bagging N* and *Bagging* $2N_1$ counterparts. A summary of these findings is listed in the previous section. The only differences here are that the Top ensemble shares the lowest range of mean member error values with the Top 2 and Top Overall ensembles, and that the differences seen between ensemble mean member error and diversity ranges is the least pronounced out of all three versions of bagging investigated. This matches the diversity findings just discussed.

The final set of graphs in Figure 6.10 display the range of ensemble errors obtained over all 10 runs of *Bagging* $2N_2$. Once again the results here are similar to those seen with *Bagging N* and *Bagging* $2N_1$: The ensembles analyzed are not well behaved, with the range of ensemble error values indicating that any increase in diversity obtained by including more component classifiers, is not enough to offset the associated increase in mean member error as dictated by the accuracy-diversity trade-off. This coincides with the findings in Section 5.2.4 where the Top Positive ensemble was found to have the lowest losses ranking with the other ensembles having similar performances.

The results of the diversity analysis of *Bagging 2*N_2, fall in line with those obtained with the other two versions of bagging. This makes sense, as ensemble performance with *Bagging 2*N_2 was found to be similar to that of ensemble performance with *Bagging* N and *Bagging 2*N_1 (as seen in Section 5.2.4). We will discuss these results in the following section, looking at their practical implications in particular.

6.3 Discussion

In the previous section we discussed the results obtained when analyzing the accuracy-diversity trade-off, as formalized by Krogh and Vedelsby [22], for ensembles made using the mean combination scheme and bagging. None of these ensembles were found to be well behaved; in other words increasing the number of component classifiers did not result in a decrease in ensemble error. The Top–Top 5 and Top Overall ensembles had a similar range of ensemble error values, while the Top Positive ensemble, in all runs, had a larger range, which was especially prominent up to at least around the 50th percentile.

These results shed light on our findings in Section 5.2 where we saw with ensembles created using the mean combination scheme, increasing ensemble diversity did not result in improved ensemble performance. The ensembles made with the most accurate component classifiers (i.e. the Top–Top 5 and Top Overall ensembles) performed similarly, with the ensemble with the most diversity, the Top Positive ensemble, clearly performing the worst.

By using more member classifiers when creating an ensemble we expect ensemble diversity to increase. Taking into account the accuracy-diversity trade-off, this would also result in mean member error increasing as well. Given Formula 6.3, for ensemble performance to improve, we would need the increase in diversity to be larger than the associated increase in mean member error. Based on the results presented in the previous chapter we can see that this has not been the case.

We believe that this is due to our ensemble building methology, which we based on research done by Kocaguneli et al. [21], being focused primarily on component classifier accuracy. As such, increasing ensemble member count results in including progressively less accurate classifiers without any consideration to what these classifiers bring to ensemble diversity.

So what do these results mean? From a practitioner's perspective we have already shown that Web effort estimation ensembles perform favourably in comparison to their solo counterparts, and more importantly, consistently (as seen by our bagging experiments). With these diversity results we would advise practitioners building such ensembles to simply use the most accurate component classifiers instead of attempting to include classifiers with a wide range of performances. Using a smaller set of member classifiers would also simplify ensemble creation if done manually.

From a research perspective the results here illustrate that the formalization of the accuracy-diversity trade-off, proposed by Krogh and Vedelsby [22], can successfully be used to analyse ensemble performance in the domain of Web effort estimation. We feel that this provides a powerful tool with which researchers can expand on the work done here. For example it opens up an avenue for future research; creating ensembles from component classifiers selected to maximize diversity while at the same time minimizing mean member error. This will be discussed in more detail in the next chapter.

6.4 Conclusion

In this chapter we analyzed the Web effort ensembles created via bagging with regards to the accuracy-diversity trade-off. We used a mathematical formalization specified by Krogh and Vedelsby which expresses ensemble error as the difference between the mean member error and ensemble diversity. In order to use this formalization we restricted the analysis in this chapter to ensembles created using the mean combination scheme.

From this accuracy-diversity trade-off analysis, it can be seen that regardless of the type of bagging used, Web effort estimation ensembles are not well behaved. Increasing the number of component classifiers does not result in a corresponding improvement in ensemble performance. We believe that this is due to our ensemble building methodology being focused on component classifier performance. This corresponds to our findings in the previous chapter where we found component classifier accuracy to be the primary determinant of ensemble performance.

We believe that our research has shown that ensembles are an effective alternative to solo classifiers for Web effort estimation. We have demonstrated that they consistently perform well, doing so over all runs and all three types of bagging, both in terms of ranking and in terms of performance measured using mean AR. As our ensemble building methodology focuses on component classifier accuracy and not diversity, the best strategy when using this methodology would be to build ensembles using a small set of the most accurate solo classifiers. This is particularly important if the mean combination scheme is used.

In the following chapter we will conclude this thesis with a summary of our research achievements as well as a discussion of possible avenues for further research.

7 Conclusions

We conclude this thesis with a summary of our research contributions, a discussion of threats to the validity of our findings, followed by a look at potential avenues for future research.

7.1 Summary

Web development plays an important role in today's industry, and successful Web development requires effective resource management. A resource is any factor that has a bearing on project outcome with related research centering on development effort. A variety of effort estimation techniques have been investigated, but there is no consensus as to which technique is the best; an issue general to software effort estimation. We address this problem by using ensembles of effort estimation techniques, focusing specifically on Web project data. The scientific contributions of our research, as detailed in this thesis, are summarized in the following section.

7.1.1 Systematic Literature Review Of Web Resource Estimation

Despite the research in this thesis being focused on development effort estimation, when we performed our systematic literature review, we wanted to establish the current state of the art with regards to the estimation of any resource relevant to Web development. This would let us document any existing research gaps in this domain, while providing us information on the datasets, predictors, and estimation techniques used in research that is closely related, and therefore relevant to ours.

We found that most work on resource estimation has focused on development effort estimation [3]. Size measures were found to be the most common resource predictor, being used in every single study selected by the review. Empirical research into Web resource estimation has favored the use of industry datasets over academic datasets. A variety of estimation techniques and performance measures have been used with no consensus as to which are the best. Estimation accuracy was found to be very variable depending on the dataset, estimation technique (including its configuration), performance measure, and experimental setup used.

7.1.2 Using Ensembles For Web Effort Estimation

The lack of consensus on an individual estimation technique being the best, motivated us to investigate using ensembles of learners for effort estimation. A previous study [21] had demonstrated that effort estimation ensembles consistently outperformed solo learners, over multiple datasets and performance measures, in the domain of general software estimation. We replicated this study using Web project data from the Tukutuku dataset to verify if this methodology would work with Web effort estimation [4].

We found that the 15 ensembles created performed consistently well. They outperformed all but two of the 90 solo learners investigated, across the three ranking systems and seven performance measures used. The ensembles were also shown to make smaller errors than their solo counterparts, avoiding the very large errors seen with some of the solo learners.

7.1.3 Further Analysis Of Web Effort Estimation Ensembles Using Bagging

We improved on the methodology replicated, focusing on a single performance measure, updating the ranking system so that learners with identical performances for a particular ranking system received the same ranking, and altering the criteria for selecting learners with which to build ensembles so that more learners were considered.

More importantly, we introduced bootstrap aggregation, commonly referred to as bagging, to the process of ensemble creation. With bagging, training sets are generated using sampling with replacement, enabling us to perform multiple test runs using the data from the Tukutuku dataset, while reducing the amount of correlation between training sets within an experimental run. Three versions of bagging were investigated, differing in the amounts of unique data within a training set.

We found that Web estimation ensembles performed consistently well over all runs (10 for each type of bagging) over all three types of bagging, both in terms of ranking and the performance measure used. Bagging was seen to adversely effect solo classifier performance, particularly in the case of unstable learners. This was attributed to a decrease in amount of unique training data, as well as duplicate data in training sets. Bagging was not an issue for ensemble performance. Lastly, we found component classifier accuracy to be the key factor for ensemble performance.

7.1.4 Accuracy-Diversity Trade-Off Analysis Of Ensemble Performance

The general consensus to building an effective ensemble is to choose component classifiers that are both accurate and diverse [12]. In order to obtain a better understanding of our ensemble results, particularly with regards to component classifier accuracy being the primary factor for their performance, we analyzed these results using a mathematical formalization of the accuracy-diversity trade-off. This involves the decomposition of ensemble error into the mean error and diversity of its component classifiers.

We found that regardless of the type of bagging used, our Web effort estimation ensembles were not well behaved. Increasing the number of component classifiers did not result in a respective improvement in ensemble performance. Any gains in ensemble diversity obtained when increasing the number of component classifiers, was not enough to offset the corresponding increase in component classifier error. We believe that this is due to our ensemble building methodology being focused solely on component classifier performance.

7.2 Threats To Validity

Wohlin et al. define the validity of a set of findings as their "trustworthiness" [41]. They distinguish four aspects of validity; conclusion, internal, construct, and external. In the following section we will discuss any factors that threaten these aspects of validity.

7.2.1 Conclusion Validity

Conclusion validity is concerned with whether or not our conclusion about the relationship between the treatment (effort estimation classifiers) and the outcome (their estimates) is correct [41].

The validity of an experiment is dependent on the reliability of its measures. One possible threat to conclusion validity is therefore the quality of the Tukutuku dataset. As discussed in Section 3.1.4, 77.6% of the data in Tukutuku was collected using timesheets [29]. Naturally the use of timesheets is not a guarantee of absolute accuracy with regards to the effort data collected. This remains a weakness of not just the Tukutuku dataset, but other benchmarking datasets like the ISBSG dataset as well [1].

In terms of the effort estimation classifiers used (i.e. our treatment), given that we started by replicated the work described in [21], we used the same code for the nine solo learners and most of the pre-processors. Whatever we had to implement was done with strict adherence to their methodology, with the author of the original study, kindly clearing up any queries we had.

The ensemble classifiers were created using three commonly used combination schemes; mean, median, and IRWM, that are straight forward to calculate. Mean ensemble estimates were further verified during our accuracy-diversity trade-off analysis, as the results obtained fit the mathematical formalization of this trade-off described by [22].

7.2.2 Internal Validity

Internal validity looks at whether there are any confounding factors that affect the independent variables with respect to causality, thereby threatening the conclusion about the relationship between treatment and outcome [41]. We address potential threats to internal validity with our experimental design.

With our replication study, all of our solo classifiers get the same training set for every test case evaluated as we are using leave-one-out cross-validation. Ensemble estimates are obtained by combining the solo classifier estimates for a particular test case.

In terms of our bagging experiments, we first ran a control where bagging was not used (see Section 5.2.1) against which we could compare our bagging results. When bagging was used, each learner utilized the same bag to form a classifier with which to estimate a particular test case.

For all of our experimental runs, control or bagging, we utilized the same set of solo learners, the same ranking system, and the same combination schemes for building ensemble classifiers. As such we only varied whether or not bagging was used, and if bagging was used, the type of bagging utilized.

7.2.3 Construct Validity

Construct validity considers whether our experiments are measuring what they are supposed to measure [41]. As with internal validity, we address potential threats to construct validity with our experimental design.

In terms of our data, the effort predictors in Tukutuku (size measures and cost drivers) were obtained from the results of a survey on forms used by development companies to give quotes on Web development projects [29]. These predictors were further validated with a case study and a second survey. The Tukutuku dataset has also been used extensively in Web effort estimation research (see Section 2.2).

With regards to error measures used, in our replication of the work described in [21] we used seven error measures, all of which had been used in prior effort estimation research. In our updated methodology we settled on the mean absolute residual (MAR), which is based on the underlying distribution of absolute residuals. Absolute residuals are regarded as fundamental to any numerical performance measure used to evaluate development effort estimation [38].

Finally, when we analyzed the accuracy-diversity trade-off, we used a well established mathematical formalization of this trade-off detailed in [22] which our results fit, even though they were obtained without any prior consideration of this formalization.

7.2.4 External Validity

External validity is concerned with whether the results obtained by our research can be generalized to scenarios that do not use our experimental specifications [41]. We have identified two threats to external validity.

The Tukutuku dataset consists of Web project data volunteered by Web development companies from around the globe [29]. The data in Tukutuku is therefore not a random sample of Web project data from a defined population. What this means is that we cannot automatically generalize these results to Web project data from other companies that may be different from those provided for Tukutuku. It is important to note that as far as we know, Tukutuku is the largest repository of Web project data, and is one that has been used extensively in Web effort estimation literature.

The second threat arises from the fact that the selection of learners used is not definitive; there are other learners that have been used in Web effort estimation that we have not investigated. This also applies to pre-processors used, as well as any parameters required by the learners/pre-processors.

7.3 Future Directions

In the following section we will discuss some directions that future research into the use of ensembles for Web effort estimation can take.

7.3.1 Other Solo Learners

The ensembles investigated by our research have been created from 90 different solo learners. These learners were the ones investigated in the study replicated [21], and were selected for having been used for effort estimation research previously. While this is an extensive collection of learners, it is by no means definitive. For example, not all of the learners used for Web effort estimation (see Chapter 2) are in this set of 90 learners. Our ensemble creation methodology could therefore be extended to other solo learners, whether they are already established in the field of Web effort estimation or are being evaluated for the first time in this domain.

7.3.2 Other Ensemble Methods

The methodology we have used to build ensembles in our research is based on work done in [21] which proved successful for general effort estimation, and that we have shown to be successful for Web effort estimation. There are however, other methods that have been used in machine learning literature that can be investigated in conjunction with effort estimation ensembles.

With our methodology, we build ensembles from classifiers obtained from 90 different solo learners. An alternative approach would be to make ensembles using a single base learning algorithm [6, 12, 13]. This would be achieved by running this solo learner multiple times on different subsets of training data. Each run would result in a different classifier, and the estimates made by these different classifiers can then be combined into an ensemble estimate. This method of ensemble creation would be particularly effective with unstable learners; learners whose classifiers change with small changes in training set. These include learners like CART, which we have shown to be some of the more accurate solo learners when used for Web effort estimation.

We have already investigated one technique with which to generate multiple different subsets of training data; bagging. The three versions of bagging described in this thesis could be used to make ensembles from classifiers obtained from a single base learner. Two alternatives to bagging are boosting and cragging. Boosting using the AdaBoost algorithm involves associating each training example with a weight [12, 13]. These weights are adjusted after a learner is run on the training data, such that training examples where the resulting classifier performs poorly have their weights increased, while training examples where the resulting classifier performs well have their weights decreased. Different subsets of training data can then be created by sampling with replacement, where the probability of a training example being sampled is proportional to its weight. Alternatively, the learning algorithm could be modified to directly utilize weighted training data to generate different classifiers. As AdaBoost increases the weights of training examples where a classifier performs poorly, the algorithm is sensitive to noisy data and outliers [13], so care must be taken with the data used.

Cragging (cross-validation aggregating) involves dividing training data into mutually exclusive partitions that are approximately of equal size [15]. All but one of these partitions can then be combined to create a subset of training data. Dividing training data into n partitions enables n different subsets to be created. The number of subsets can be further increased by running the partitioning procedure multiple times: r partitioning runs creating n partitions each would create $r \times n$ different subsets of training data.

It should be noted that both boosting and cragging can also be used in place of bagging with our current methodology. It would be useful to investigate these techniques and see how the resulting ensembles perform in comparison to their bagging counterparts when using Tukutuku data for Web effort estimation.

7.3.3 Focusing On Diversity

Our current ensemble creation methodology focuses on using the most accurate solo classifiers as member classifiers in an ensemble. As a result increasing ensemble diversity simply by including more, progressively less accurate classifiers, did not result in an improvement in ensemble performance. An alternative approach would be to create ensembles using member classifiers that have been selected to optimize ensemble diversity. This is an area of ensemble research [7, 8, 23] albeit not in the domain of Web effort estimation.

A survey of diversity creation methods for ensembles is provided in [7]. Diversity creation techniques are classified as being implicit or explicit. Implicit methods use randomization to create different subsets of training data for use in classifier creation. Bagging is one such technique. Explicit methods on the other hand directly alter the distribution of training data for use in classifier creation. Boosting would be an example of an explicit diversity creation technique.

An in depth look into explicit diversity creation methods and adapting them for Web effort estimation would be a research topic in and of itself. Investigating this area would be worthwhile given the promising results we have already obtained with our Web effort estimation ensembles.

7.3.4 Tool For Ensemble Creation

The ensembles analyzed in our research were built by hand; while the process of running and comparing (via round-robin) solo learners was automated, their ranking and subsequent combination into ensembles was done manually. To simplify the process of ensemble creation, and reduce a potential barrier to practitioners seeking to use ensembles for Web effort estimation, creating a tool to automate the process would be valuable.

The tool would ideally have an intuitive graphical user interface (GUI) that would let a practitioner enter their Web project dataset, and make the necessary choices required to run the effort estimation process (e.g. solo learners to use, type of bagging if any, combination scheme etc). The tool would handle running the solo learners on the dataset, comparing and ranking them, and building the ensembles using the selected combination scheme. The GUI would then display the output desired by the practitioner, which would include ensemble estimates, AR values, ensemble error and diversity, and component classifier error.

Another useful feature would be to make the tool extensible. This would ensure future updates to include new solo learners, combination schemes, or ensemble methods can easily be implemented.

7.4 Conclusion

Accurate effort estimation enables effective managerial decisions to be made by a project manager when embarking on a project. In terms of Web development, research into effort estimation has led to the investigation of a number of techniques, none of which have proven to conclusively be the best.

Our research has demonstrated that while there may not be a single best estimation technique, using ensembles of techniques can result in consistently accurate performances. Practitioners can combine the estimates made by a set of accurate estimation techniques, perhaps a small set of techniques that they have used successfully in the past, to create effective effort estimation ensembles.

All three combination schemes used in our research (mean, median, and IRWM) are straight forward to calculate, and should not prove an issue for practitioners. Care need only be taken when using the mean combination scheme; if practitioners feel that the techniques they are using are not similarly accurate, the median or IRWM combination schemes would be a preferable choice.

For practitioners new to effort estimation or those that have yet to identified a set of learners that work well on their project data, the first step would be to identify a set of accurate solo classifiers. They could achieve this by utilizing our updated ensemble methodology based on the work done by [21].

From a research perspective we have shown that bagging can be used to provide a more in depth look at effort estimation performance. With bagging, multiple estimation runs can be made using a single dataset. We have also demonstrated the effect that the bagging process has on estimation performance; the reduction of training data that bagging usually entails as well as duplicate data in bags, can reduce the estimation accuracy of unstable techniques like CART.

We analyzed three variants of bagging and found that by negating the first issue with *Bagging 2N*$_2$, we could minimize the effect that bagging has on unstable techniques. This would be useful for Web effort estimation researchers investigating bagging using unstable techniques, and/or a dataset where data loss would be a significant issue overall.

Lastly, we have illustrated that the formalization of the accuracy-diversity trade-off, proposed by Krogh and Vedelsby [22], can successfully be used to analyze ensemble performance in the domain of Web effort estimation. As such, this opens up an avenue for future research into using Web effort estimation ensembles, by building ensembles using techniques that have been selected to maximize diversity, while at the same time minimizing their estimation error.

We feel that the results of our research are very promising, and hope that they will encourage practitioners to use ensembles for Web effort estimation, as well as act as a stepping stone towards further research into this domain.

A
SLR Reference Library

S2 Morisio, M., Stamelos, I., Spahos, V., Romano, D., Measuring functionality and productivity in Web-based applications: a case study. In *Proceedings of the 6th International Symposium on Software Metrics* (Boca Raton, FL, USA, November 1999), METRICS '99, IEEE, pp. 111–118.

S3 Mendes, E., Investigating metrics for a development effort prediction model of Web applications. In *Proceedings of the Australian Software Engineering Conference* (Canberra, ACT, Australia, April 2000), ASWEC '00, IEEE, pp. 31–41.

S4 Mendes, E., Counsell, S., Web development effort estimation using analogy. In *Proceedings of the Australian Software Engineering Conference* (Canberra, ACT, Australia, April 2000), ASWEC '00, IEEE, pp. 203–212.

S5 Mendes, E., Hall, W., Towards the prediction of development effort for Web applications. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia* (San Antonio, TX, USA, May 2000), HYPERTEXT '00, ACM, pp. 242–243.

S6 Mendes, E., Mosley, N., Comparing effort prediction models for Web design and authoring using boxplots. In *Proceedings of the 24th Australasian Computer Science Conference* (Gold Coast, QLD, Australia, January 2001), ACSC 2001, IEEE, pp. 125–133.

S7 Mendes, E., Counsell, S., Mosley N., Measurement and Effort Prediction for Web Applications. In *Web Engineering*, vol. 2016 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001, pp. 295–310.

S8 Mendes, E., Counsell, S., Mosley N., Towards the prediction of development effort for hypermedia applications. In *Proceedings of the 12th ACM conference on Hypertext and Hypermedia* (Arhus, Denmark, August 2001), HYPERTEXT '01, ACM, pp. 249–258.

S9 Mendes, E., Mosley, N., Counsell, S., A Comparison of Length, Complexity and Functionality as Size Measures for Predicting Web Design and Authoring Effort. In *Proceedings of the 2001 Conference on Evaluation and Assessment in Software Engineering* (Keele, UK, 2001), EASE '01, pp. 1–14.

S10 Mendes, E., Mosley, N., Counsell, S., Using an engineering approach to understanding and predicting web authoring and design. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences* (Maui, HI, USA, January 2001), HICSS '01, IEEE, pp. 201–210.

S11 Mendes, E., Mosley, N., Counsell, S., Web metrics - estimating design and authoring effort. *IEEE Multimedia 8*, 1 (2001), pp. 50–57.

S14 Mendes, E., Mosley, N., Watson, I., A Comparison of Case-based Reasoning Approaches to Web Hypermedia Project Cost Estimation. In *Proceedings of the 11th International Conference on World Wide Web* (Honolulu, HI, USA, May 2002), WWW '02, ACM, pp. 272–280.

S15 Mendes, E., Mosley, N., Counsell, S., Comparison of Web size measures for predicting Web design and authoring effort. *IEE Proceedings - Software 149*, 3 (2002), pp. 86–92.

S16 Mendes, E., Mosley, N., Counsell, S., The application of case-based reasoning to early Web project cost estimation. In *Proceedings of the 26th Annual International Computer Software and Applications Conference* (Oxford, England, UK, August 2002), COMPSAC '02, IEEE, pp. 393–398.

S17 Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S., A comparison of development effort estimation techniques for Web hypermedia applications. In *Proceedings of the 8th IEEE Symposium on Software Metrics* (Ottawa, Canada, June 2002), METRICS 2002, IEEE, pp. 131–140.

S19 Baresi, L., Morasca, S., Paolini P., Estimating the design effort of Web applications. In *Proceedings of the 9th International Symposium on Software Metrics* (Sydney, Australia, September 2003), METRICS 2003, IEEE, pp. 62–72.

S21 Mendes, E., Counsell, S., Mosley, N., Web hypermedia cost estimation: further assessment and comparison of cost estimation modelling techniques. *The New Review of Hypermedia and Multimedia 8*, 1 (2003).

S22 Mendes, E., Mosley, N., Counsell, S., A replicated assessment of the use of adaptation rules to improve Web cost estimation. In *Proceedings of the International Symposium on Empirical Software Engineering* (Rome, Italy, September 2003), ISESE '03, IEEE, pp. 100–109.

S24 Mendes, E., Mosley, N., Counsell, S., Early Web size measures and effort prediction for Web costimation. In *Proceedings of the 9th International Software Metrics Symposium* (Sydney, Australia, September 2003), METRICS 2003, IEEE, pp. 18–29.

S25 Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S., A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. *Empirical Software Engineering 8*, 2 (2003), pp. 163–196.

S26 Ochoa, S.F., Bastarrica, M.C., Parra, G., Estimating the development effort of Web projects in Chile. In *Proceedings of the First Latin American Web Congress* (Santiago, Chile, November 2003), LA-WEB 2003, IEEE, pp. 114–122.

S27 Ruhe, M., Jeffery, R., Wieczorek, I., Cost estimation for web applications. In *Proceedings on the 25th International Conference on Software Engineering* (Portland, OR, USA, May 2003), ICSE '03, IEEE, pp. 285–294.

S28 Ruhe, M., Jeffery, R., Wieczorek, I., Using Web objects for estimating software development effort for Web applications. In *Proceedings of the 9th International Symposium on Software Metrics* (Sydney, Australia, September 2003), METRICS 2003, IEEE, pp. 30–37.

S30 Candido, E.J.D., Sanches, R., Estimating the size of web applications by using a simplified function point method. In *Proceedings of WebMedia and LA-Web* (Ribeirao Preto-SP, Brazil, October 2004), WebMed/LA-WEB 2004, IEEE, pp. 98–105.

S31 Costagliola, G., Ferrucci, F., Gravino, Tortora, G., Vitiello, G., A COSMIC-FFP Based Method to Estimate Web Application Development Effort. *Web Engineering*, Springer Berlin Heidelberg, 2004, pp. 161–165.

S32 Kitchenham, B.A., Mendes E., A comparison of cross-company and within-company effort estimation models for Web applications. In *Proceedings of the International Conference on Empirical Assessment in Software Engineering* (2004), EASE '04, pp. 47–55.

S34 Kitchenham, B.A., Mendes E., Further comparison of cross-company and within-company effort estimation models for Web applications Software Metrics. In *Proceedings of the 10th International Symposium on Software Metrics* (Chicago, IL, USA, September 2004), METRICS 2004, IEEE, pp. 348–357.

S35 Umbers, P., Miles, G., Resource estimation for Web applications Software Metrics. In *Proceedings of the 10th International Symposium on Software Metrics* (Chicago, IL, USA, September 2004), METRICS 2004, IEEE, pp. 370–381.

S37 Marchetto, A., Trentini, A., Evaluating web applications testability by combining metrics and analogies. In *Proceedings of the ITI 3rd International Conference on Information and Communications Technology* (Cairo, Egypt, December 2005), ICICT '05, IEEE, pp. 751–779.

S39 Mendes, E., Mosley, N., Counsell, S., Exploring case-based reasoning for web hypermedia project cost estimation. *International Journal of Web Engineering and Technology 2*, 1 (2005) pp. 117–143.

S40 Mendes, E., Mosley, N., Counsell, S., Investigating Web size metrics for early Web cost estimation. *Journal of Systems and Software 77*, 2 (2005), pp. 157–172.

S41 Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Vitiello, G., A cosmic-ffp approach to predict web application development effort. *Journal of Web Engineering 5*, 2 (2006), pp. 93–120.

S42 Costagliola, G., Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Vitiello, G., Effort estimation modeling techniques: a case study for web applications. In *Proceedings of the 6th International Conference on Web Engineering* (Palo Alto, CA, USA, July 2006), ICWE '06, ACM, pp. 9–16.

S43 Idri, A., Zahi, A., Abran, A., Software Cost Estimation by Fuzzy Analogy for Web Hypermedia Applications. In *Proceedings of the International Conference on Software Process and Product Measurement* (Cadiz, Spain, 2006), SPPM '06 pp. 53–62.

S44 Mendes, E., Mosley, N., Counsell, S., Web Effort Estimation. *Web Engineering*, Springer Berlin Heidelberg, 2006, pp. 29–73.

S45 Xunmei, G., Guoxin, S., Lizhong, X., Design of a fuzzy decision-making model and its application to software functional size measurement. In *International Conference on Computational Intelligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce* (Sydney, Australia, November 2006), CIMCA '06, IEEE, pp. 988–993.

S46 Abrahão, S., Mendes, E., Gomez, J., Insfran E., A Model-Driven Measurement Procedure for Sizing Web Applications: Design, Automation and Validation. *Model Driven Engineering Languages and Systems*, Springer Berlin Heidelberg, 2007, pp. 467–48.

S47 Baresi, L., Morasca, S., Three empirical studies on estimating the design effort of Web applications. *ACM Transactions on Software Engineering and Methodology 16*, 4 (2007).

S48 Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Comparing Size Measures for Predicting Web Application Development Effort: A Case Study. In *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement* (Madrid, Spain, September 2007), ESEM '07, IEEE, pp. 324–333.

S49 Chae, H.S., Kim, T.Y., Jung, W., Lee, J., Using Metrics for Estimating Maintainability of Web Applications: An Empirical Study. In *6th IEEE/ACIS International Conference on Computer and Information Science* (Melbourne, Australia, July 2007), ICIS 2007, IEEE, pp. 1053–1059.

S50 Idri, A., Zakrani, A., Elkoutbi, M., Abran, A., Fuzzy Radial Basis Function Neural Networks for Web Applications Cost Estimation. In *Proceedings of the 4th International Conference on Innovations in Information Technology* (Dubai, UAE, November 2007), IIT '07, IEEE, pp. 576–580.

S51 Mendes, E., A Comparison of Techniques for Web Effort Estimation. In *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement* (Madrid, Spain, September 2007), ESEM '07, IEEE, pp. 334–343.

S52 Mendes, E., Predicting Web Development Effort Using A Bayesian Network. In *Proceedings of the International Conference on Empirical Assessment in Software Engineering* (April 2007), EASE '07, British Computer Society, pp. 83–93.

S53 Mendes, E., The use of a Bayesian network for web effort estimation. In *Proceedings of the 7th International Conference on Web Engineering* (Como, Italy, July 2007), ICWE '07, Springer Berlin Heidelberg, pp. 90–104.

S54 Mendes, E., Di Martino, S., Ferrucci, F., Gravino, C., A replicated study comparing web effort estimation techniques. In *Proceedings of the 8th International Conference on Web Information Systems Engineering* (Nancy, France, December 2007), WISE '07, Springer Berlin Heidelberg, pp. 423–435.

S55 Mendes, E., Di Martino, S., Ferrucci, F., Gravino, C., Effort estimation: how valuable is it for a web company to use a cross-company data set, compared to using its own single-company data set?. In *Proceedings of the 16th International Conference on the World Wide Web* (Banff, Canada, May 2007), WWW '07, ACM, pp. 963–972.

S56 Reddy, S., Raju, K., Srinivas, T., Devi, G.L., A neural network approach for web cost estimation. In *Proceedings of the 11th IASTED International Conference on Software Engineering and Applications* (Cambridge, MA, USA, November 2007), SEA '07, ACTA Press, pp. 37–41.

S57 Sneed, H.M., Huang, S., Sizing Maintenance Tasks for Web Applications. In *Proceedings of the 11th European Conference on Software Maintenance and Reengineering* (Amsterdam, Netherlands, March 2007), CSMR 2007, IEEE, pp. 171–180.

S58 Aggarwal, N., Prakash, N., Sofat, S., Content management system effort estimation model based on object point analysis. *International Journal of Computer Science and Engineering 2*, 4 (2008), pp. 194–201.

S62 Ferrucci, F., Gravino, C., Di Martino, S., A Case Study Using Web Objects and COSMIC for Effort Estimation of Web Applications. In *Proceedings of the 34th Euromicro Conference on Software Engineering and Advanced Applications* (Parma, Italy, September 2008), SEAA '08, IEEE, pp. 441–448.

S63 Hooi, T.C., Yusoff, Y., Hassan, Z., Comparative Study on Applicability of WEBMO in Web Application Cost Estimation within Klang Valley in Malaysia. In *IEEE 8th International Conference on Computer and Information Technology Workshops* (Sydney, Australia, July 2008), CIT Workshops 2008, pp. 116–121

S64 Idri, A., Zahi, A., Mendes, E., Zakrani, A., Software Cost Estimation Models Using Radial Basis Function Neural Networks. *Software Process and Product Measurement*, Springer Berlin Heidelberg, 2008, pp. 21–31.

S65 Mendes, E., The Use of Bayesian Networks for Web Effort Estimation: Further Investigation, In *Proceedings of the 8th International Conference on Web Engineering* (Yorktown Heights, NY, USA, July 2008), ICWE 2008, IEEE, pp. 203–216.

S66 Mendes, E., Mosley, N., Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE Transactions on Software Engineering 34*, 6 (2008), TSE '08, Page(s): 723–737.

S67 Mendes, E., Di Martino, S., Ferrucci, F., Gravino, C., Cross-company vs. single-company web effort models using the Tukutuku database: An extended study. *Journal of Systems and Software 81*, 5 (2008) pp. 673–690.

S69 Aggarwal, N., Prakash, N., Sofat, S., Web hypermedia content management system effort estimation model. *SIGSOFT Software Engineering Notes 34*, 2 (2009), pp. 1–7.

S70 Barabino, G., Porruvecchio, G., Concas, G., Marchesi, M., De Lorenzi, R., Giaccardi, M., An Empirical Comparison of Function Points and Web Objects. In *International Conference on Computational Intelligence and Software Engineering* (Wuhan, China, December 2009), CiSE 2009, IEEE, pp. 1–4.

S71 Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Using Support Vector Regression for Web Development Effort Estimation. In *Proceedings of the International Conferences on Software Process and Product Measurement* (Kraków, Poland, November 2009), IWSM 2009/Mensura 2009, Springer Berlin Heidelberg, pp. 255–271.

S72 Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Applying support vector regression for web effort estimation using a cross-company dataset. In *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement* (Orlando, FL, USA, October 2009), ESEM '09, IEEE, pp. 191–202.

S74 Di Martino, S., Gravino, C., Estimating web application development effort using COSMIC-FFP method. *International Journal of Computers and Applications 31*, 3 (2009), pp. 153–158.

S75 Di Martino, S., Ferrucci, F., Gravino, C., An empirical study on the use of Web-COBRA and Web Objects to estimate web application development effort. *Web Engineering*, Springer Berlin Heidelberg, 2009, pp. 213–220.

S76 Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Measures and techniques for effort estimation of web applications: An empirical study based on a single-company dataset. *Journal of Web Engineering 8*, 2 (2009), pp. 154–181.

S77 Ferrucci, F., Gravino, C., Di Martino, S., Estimating Web Application Development Effort Using Web-COBRA and COSMIC: An Empirical Study. In *Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications* (Patras, Greece, August 2009), SEAA '09, IEEE, pp. 306–312.

S80 Marchetto, A., OQMw: An OO quality model for web applications. *Tamkang Journal of Science and Engineering 12*, 4 (2009), pp. 459–470.

S81 Mendes, E., Web Cost Estimation and Productivity Benchmarking. *Software Engineering*, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2009, pp. 194–222.

S82 Abrahão, S., Gómez, J., Insfran, E., Validating a size measure for effort estimation in model driven Web development. *Information Sciences: an International Journal 180*, 20 (2010), pp. 3932–3954.

S83 Baker, S., Mendes, E., Aggregating Expert-Driven Causal Maps for Web Effort Estimation. *Advances in Software Engineering*, *Communications in Computer and Information Science*, Springer Berlin Heidelberg, 2010, pp. 264–282.

S84 Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E., How effective is Tabu search to configure support vector regression for effort estimation?. In *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (Timisoara, Romania, September 2010), PROMISE '10, ACM, pp. 1–10.

S85 Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Mendes, E., Investigating Tabu Search for Web Effort Estimation. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications* (Lille, France, September 2010), SEAA '10, IEEE, pp. 350–357.

S87 Lazić, L., Mastorakis, N.E., Two novel effort estimation models based on quality metrics in web projects. *WSEAS Transactions on Information Science and Applications 7*, 7 (2010), pp. 923–934.

S89 Abdelali Z., Idri, A., Applying radial basis function neural networks based on fuzzy clustering to estimate web applications effort. *International Review on Computers and Software 5*, 5 (2010), pp. 516–524.

S90 Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Investigating the use of Support Vector Regression for web effort estimation. *Empirical Software Engineering 16*, 2 (2010), pp. 211–243.

S91 Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E., Using tabu search to configure support vector regression for effort estimation. *Empirical Software Engineering*, Springer Netherlands, 2011, pp. 1–41.

S92 Das, S.S., Devadutta, K., Swain, S.K., Kumar, S., Web Components as a measure for estimating Effort and Size of Web Applications. *International Journal of Computer Science and Information Technologies 2*, 3 (2011), pp. 1137–1143.

S93 Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Using web objects for development effort estimation of web applications: a replicated study. *Product-Focused Software Process Improvement*, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 186–201.

S96 Folgieri, R., Barabino, G., Concas, G., Corona, E., De Lorenzi, R., Marchesi, M., Segni, A., A revised web objects method to estimate web application development effort. In *Proceedings of the 2nd International Workshop on Emerging Trends in Software Metrics* (Waikiki, Honolulu, HI, USA, May 2011), WETSoM '11, ACM, pp. 59–64.

S97 Mendes, E., Knowledge representation using Bayesian networks - A case study in Web effort estimation. *World Congress on Information and Communication Technologies*, 2011, pp. 612–617.

S98 Prakancharoen, S., Web based application maintenance cost estimation modeling using Bayesian SEM. *Advanced Materials Research*, Volumes 403–408, 2011, pp. 3704–3708.

E1 Abran, A., Silva, I., Primera, L., Field Studies using functional size measurement in building estimation models for software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice 14*, 1 (2002), pp. 31–64.

E2 Mendes, E., Mosley, N., Web Metrics and Development Effort Prediction. In *Proceedings of the Australian Conference on Software Measurement* (Sydney, Australia, November 2000), ACOSM '00.

E3 Nageswaran S., Test Effort Estimation Using Use Case Points. In *Proceedings of Quality Week 2001* (San Francisco, CA, USA, May 2001).

E4 Baresi, L., Morasca, S., Paolini, P., An Empirical Study on the Design Effort of Web Applications. In *Proceedings of the 3rd International Conference on Web Information Systems Engineering* (Singapore, December 2002), WISE '02, IEEE, pp. 345–354.

E5 Fewster, R., Mendes, E., Measurement, Prediction and Risk Analysis for Web Applications. In *Proceedings of the 7th International Software Metrics Symposium* (London, England, April 2001), METRICS 2001, IEEE pp. 338–348.

E6 Fewster, R., Mendes, E., Empirical Evaluation and Prediction of Web Applications' Development Effort. In *Proceedings of the International Conference on Empirical Assessment in Software Engineering* (Staffordshire, UK, April 2000), EASE '00.

E7 Mendes, E., Pollino, C., Mosley, N., Building an Expert-based Web Effort Estimation Model using Bayesian Networks. In *Proceedings of the 13th International Conference on Empirical Assessment in Software Engineering* (Durham University, UK, April 2009), EASE '09.

B
Data Extraction Form

Table B.1 displays the form used during the data extraction process of the SLR. Note that the form also incorporates the quality assessment checklist discussed in Section 2.1.5. A separate form was used for each article in the review's final reference library.

Table B.1: SLR data extraction form

Data Item	**Value**	**Supplementary Notes**
Study Information Data		
Study ID		
Title		
Author(s)		
Year of publication		
Reference type		
Publisher		
Data Relevant to Answering Research Questions		
Data characteristics		
What methods/techniques were used for resource estimation?		
Continued on next page		

Data Item	Value	Supplementary Notes
What resource facet is investigated and what resource predictors are used?		
At what stage of the project were the predictors gathered?		
What metrics have been used to measure estimation accuracy?		
What accuracy did these methods/techniques achieve?		
Quality Assessment Checklist		
Are the research aims clearly specified?		Yes/No/Partially
Was the study designed to achieve these aims?		Yes/No/Partially
Are the prediction techniques used clearly described and their selection justified?		Yes/No/Partially
Are the variables considered by the study suitably measured?		Yes/No/Partially
Are the data collection methods adequately detailed?		Yes/No/Partially
Is the data collected adequately described?		Yes/No/Partially
Is the purpose of the data analysis clear?		Yes/No/Partially
Are the statistical techniques used to analyze the data adequately described and their use justified?		Yes/No/Partially
Were potential confounders suitably controlled for in the analysis?		Yes/No/Partially
Are the study findings credible?		Yes/No/Partially
Are negative results (if any) presented?		Yes/No/Partially
Do the researchers discuss any problems with the validity/reliability of their results?		Yes/No/Partially

C Error/Performance Measures

The following appendix provides the mathematical formulae for the error measures and their related performance measures discussed in this thesis. Most of these are listed in Table 2.5; BRE and IBRE are used in our replication study discussed in Chapter 4. For all formulae x represents the actual value and $\hat{x}$ the estimated value.

C.1 Absolute Residual (AR)

$$AR = |x - \hat{x}| \tag{C.1}$$

- MAR — the mean AR.
- MdAR — the median AR.

C.2 Magnitude of Relative Error (MRE)

$$MRE = \frac{|x - \hat{x}|}{x} = \frac{AR}{x} \tag{C.2}$$

The MRE is used for the following performance measures:

- MMRE — the mean MRE.
- MdMRE — the median MRE.
- PRED(25) — the proportion of MRE values less than or equal to 25%.

C.3 Estimation Magnitude of Relative Error (EMRE)

$$EMRE = \frac{|x - \hat{x}|}{\hat{x}} = \frac{AR}{\hat{x}} \tag{C.3}$$

The EMRE is used for the following performance measures:

- MEMRE — the mean EMRE.
- MdEMRE — the median EMRE.

C.4 Balanced Relative Error (BRE)

$$BRE = \frac{|x - \hat{x}|}{Min(x, \hat{x})} = \frac{AR}{Min(x, \hat{x})} \tag{C.4}$$

The BRE is used for the following performance measures:

- MBRE — the mean BRE.

C.5 Inverted Balanced Relative Error

$$IBRE = \frac{|x - \hat{x}|}{Max(x, \hat{x})} = \frac{AR}{Max(x, \hat{x})} \tag{C.5}$$

The IBRE is used for the following performance measures:

- MIBRE — the mean IBRE.

D Usage of Tukutuku Variables For Effort Estimation

Table 2.8 lists the papers identified by our SLR that use Tukutuku variables as predictors for effort estimation. Table D.1 provides a more detailed look at this usage and includes the number of Tukutuku variables used and their type (i.e. whether they are size measures or cost drivers). Note that for certain studies, different prediction techniques or subsets of the Tukutuku dataset used different sets of Tukutuku variables as predictors.

Table D.1: Tukutuku variables used for effort estimation.

Paper ID	Tukutuku Variables
S22	19 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S24	24 variables: size measures like NewWP and TotHigh
S32	13 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S34	11 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S40	10 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S43	9 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S44	10 variables: size measures (e.g. NewWP and TotHigh) and cost drivers (e.g. DocProc and Metrics)
S46	11 variables: size measures like NewWP and TotHigh
S48	11 variables: size measures like NewWP and TotHigh
S50	9 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S51	BN - 6 variables: size measures (e.g. TotWP, NewWP) and cost drivers (e.g. DocProc, Metrics) SWR - 12 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, ProImpr) CBR - 15 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, DocProc) CART - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)

Continued on next page

Paper ID	Tukutuku Variables
S52	BN - 6 variables: size measures (e.g. TotWP, NewWP) and cost drivers (e.g. DocProc, Metrics) SWR - 12 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, ProImpr)
S53	6 variables: size measures (e.g. TotWP, NewWP) and cost drivers (e.g. DocProc, Metrics)
S54	SWR - 12 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, ProImpr) CBR - 15 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, DocProc) CART - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S55	SWR (SCD) - 10 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR (CCD) - 6 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, TeamExp) CBR - 14 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S64	9 variables: size measures (e.g. TotWP, TotHigh) and cost drivers(e.g. DevTeam, TeamExp)
S65	BN - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh), and cost drivers (e.g. DevTeam, Metrics)
S66	BN - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S67	SWR (SCD) - 10 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR (CCD) - 6 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, TeamExp) CBR - 14 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S71	SVR - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) BN - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S72	SVR - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) BN - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S81	3 variables: two size measures (NewWP and TotHigh) and a single cost driver (DevTeam)
S84	SVR - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) BN - 18 variable: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S85	SVR - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) BN - 18 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S89	9 variables: size measures (e.g. TotWP, TotHigh) and cost drivers (e.g. DevTeam, TeamExp)
S90	SVR - 18 variables: size measures (e.g. NewWP,TotHigh) and cost drivers (e.g. DevTeam, TeamExp) SWR - 12 variables: size measures (e.g. TotWP, NewImg) and cost drivers (e.g. DevTeam, Metrics) CBR - 14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
S91	14 variables: size measures (e.g. NewWP, TotHigh) and cost drivers (e.g. DevTeam, Metrics)
BN - Bayesian Network, **SWR** - Stepwise Regression, **CBR** - Case-based Reasoning **CART** - Classification and Regression Trees, **SVR** - Support Vector Regression **SCD** - Single Company Dataset, **CCD** - Cross-Company Dataset	

E
Performance Findings

In this appendix we provide the performance findings of the various estimation techniques identified by our SLR. These performances are expressed as percentages; the only exceptions being the mean and median absolute residual (MAR and MdAR respectively). Appendix C provides the mathematical basis for each of these numerical performance measures.

Effort estimation was most investigated and Table E.1 lists the numerical performances obtained by these effort estimation techniques. Table E.2 provides the performances obtained for estimation techniques used for other facets of resource estimation.

Table E.1: Performance of effort estimation techniques

Estimation Technique	**Accuracy Achieved**
CBR	MMRE: 7.00 – 14430.99 MdMRE: 6.00 – 5146.52 Pred(25): 0.00 – 100.00 Pred(20): 9.43 – 80.02 MEMRE: 2.00 – 3170.00 MdEMRE: 81.00 – 343.00 MAR: 35.40 – 372.01 MdAR: 21.00 – 65.00

Continued on next page

Estimation Technique	Accuracy Achieved
Stepwise regression	MMRE: 1.50 – 2.62E+12 MdMRE: 0.62 – 5668.56 Pred(25): 0.00 – 100.00 MEMRE: 2.00 – 286.00 MdEMRE: 54.00 – 121.00 MAR: 11.20 – 395.10 MdAR: 8.36 – 88.40
Linear regression	MMRE: 1.50 – 110.00 MdMRE: 0.62 – 100.00 Pred(25): 40.00 – 100.00
Bayesian networks	MMRE: 34.26 – 3731.00 MdMRE: 27.42 – 805.00 Pred(25): 0.00 – 33.33 MEMRE: 78.00 – 1306.00 MdEMRE: 35.83 – 238.00
Mean	MMRE: 31.64 – 31208.52 MdMRE: 25.61 – 8781.81 Pred(25): 0.00 – 49.00 MEMRE: 107.00 – 134.00 MdEMRE: 90.00 – 91.00 MdAR: 164.00 – 646.00
Median	MMRE: 32.25 – 32542.41 MdMRE: 23.00 – 9160.36 Pred(25): 0.00 – 66.67 MEMRE: 443.00 – 462.00 MdEMRE: 78.00 – 94.00 MAR: 30.30 – 69.40 MdAR: 15.80 – 164.00
CART	MMRE: 10.00 – 690.40 MdMRE: 7.00 – 83.2 Pred(25): 20.00 – 90.91 MEMRE: 9.00 – 17.00
Support vector regression	MMRE: 59.00 – 4.38E+7 MdMRE: 31.90 – 95.40 Pred(25): 7.70 – 42.00 MEMRE: 49.80 – 1450.00 MdEMRE: 32.60 – 427.00 MdAR: 12.00 – 63.00
	Continued on next page

Estimation Technique	Accuracy Achieved
Expert judgment	MMRE: 10.00 – 68.30 MdMRE: 36.00 Pred(25): 25.00
Web COBRA	MMRE: 11.00 – 29.00 MdMRE: 10.00 – 25.00 Pred(25): 50.00 – 93.00
Radial basis function neural networks	MMRE: 0.00 – 200.00 Pred(25): 25.00 – 100.00
WEBMO	MMRE: 96.17 – 99.88
Tabu search	MMRE: 75.00 – 137.00 MdMRE: 49.00 – 76.00 Pred(25): 14.00 – 31.00 MEMRE: 177.00 – 631.00 MdEMRE: 51.00 – 142.00
CART/CBR Hybrid	MMRE: 11.00 – 31.00 MdMRE: 9.00 – 24.00 Pred(25): 57.00 – 93.00
CART/Linear regression hybrid	MMRE: 17.00 – 19.00 MdMRE: 10.00 – 20.00 Pred(25): 66.00 – 73.00
Content management system effort estimation model	MMRE: 9.00 – 24.00 Pred(25): >80
WEBMO+	MMRE: 5.00 – 10.00
VPM+	MMRE: 7.70 – 16.02
Web component model	MMRE: 7.40 Pred(25): 80
Fuzzy Analogy	MMRE: 27.38 – 737.79 Pred(20): 7.55 – 84.91
Custom Framework (S35)	MRE: 12.20 – 33.81
Custom Framework (S70)	MMRE: 49.00 – 61.00 MdMRE: 19.00 – 36.00 Pred(25): 40.00 – 60.00
Custom Framework (S96)	MMRE: 45.00 – 123.00 MdMRE: 19.00 – 66.00 Pred(25): 40.00 – 62.00

Table E.2: Performance of estimation techniques used for other facets of resource estimation

Estimation technique	**Accuracy achieved (%)**
Maintenance Effort Estimation	
Linear regression	MMRE: 45.00 Pred(25): 53.00
Non-linear regression	MMRE: 26.66 Pred(25): 26.66
Average unit cost model	MMRE: 28.70 – 111.60 Pred(25): 0.00 – 33.30
Maximum likelihood	MMRE: 47.58
Bayesian analysis	MMRE: 44.08
Quality Estimation	
CBR	MMRE: 17.40 – 127.00 Pred(25): 13.60 – 81.80
Linear regression	MMRE: 7.60 – 286.20 Pred(25): 0.00 – 100.00
Size Estimation	
Function point counts	MRE: 0.00 – 73.00 MMRE: 4.00 – 48.00

F
Tukutuku Variables

Table F.1 lists all 25 Tukutuku variables along with their description [27]. These variables can be categorized as company data, project data, or Web application data.

Table F.1: List of all 25 Tukutuku variables.

Variable	**Description**
	Company Data
Country	Country company belongs to.
Established	Year when company was established.
nPeopleWD	Number of people who work on Web design and development.
	Project Data
TypeProj	Type of project (new or enhancement).
nLang	Number of different development languages used.
DocProc	If project followed defined and documented process.
ProImpr	If project team is involved in a process improvement programme.
Metrics	If project team is part of a software metrics programme.
DevTeam	Size of a project's development team.
TeamExp	Average team experience with the development language(s) used.
TotEff	Actual total effort in person hours to develop an application.
	Continued on next page

Variable	**Description**
EstEff	Estimated total effort in person hours to develop an application.
Accuracy	Procedure used to record effort data.
Web Application Data	
TypeApp	Type of Web application developed.
TotWP	Total number of Web pages (new and reused).
NewWP	Total number of new Web pages.
TotImg	Total number of images (new and reused).
NewImg	Total number of new images created.
Fots	Number of features reused without any adaptation.
HFotsA	Number of reused high-effort features/functions adapted.
Hnew	Number of new high-effort features/functions.
TotHigh	Total number of high-effort features/functions.
FotsA	Number of reused low-effort features/functions.
New	Number of new low-effort features/functions.
TotNHigh	Total number of low-effort features/functions.

G
Inverse Rank Weighted Mean

The following Appendix will illustrate how the inverse rank weighted mean or IRWM of n values is calculated. Assuming that X_i is the i^{th} value, then the n values arranged in decreasing order of rank would look like this:

$$X_1, X_2, X_3, \ldots, X_n \tag{G.1}$$

The top-ranked value would receive a weight of n, while the bottom-ranked value would receive a weight of 1. In other words the above values would have the following weights:

$$n, n-1, n-2, \ldots, 1 \tag{G.2}$$

The values and the weights are then combined to calculate the IRWM using the equation below:

$$IRWM = \frac{\sum_{i=1}^{n}(n+1-i)X_i}{\sum_{i=1}^{n} i} \tag{G.3}$$

Both [21] and [31] have used IRWM in their research.

H
Learner Rankings–Replication Study

In the discussion of our results in Chapter 4, we chose to display a subset of the rankings we obtained for the learners investigated in our replication study, focusing on the top ranked learners. This appendix provides the complete list of rankings obtained, in ascending order of losses: Table H.1 lists the rankings of all solo learners, while Table H.2 lists the rankings of all learners, solo and ensemble.

Table H.1: Ranking, in ascending order of losses, of the solo learners and their related δr values, as obtained by our replication study

Rank	δ**r**	**Preprocessing option/ combination scheme**	**Learner**
1	0	Principal component analysis	CART (yes)
2	0	Principal component analysis	CART (no)
3	12	Stepwise regression	CART (yes)
4	12	Stepwise regression	CART (no)
5	0	Natural logarithm	Analogy – 5NN
6	2	Stepwise regression	Analogy – 5NN

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
7	6	Equal frequency – 5 bins	CART (yes)
8	6	Equal frequency – 5 bins	CART (no)
9	6	Equal frequency – 5 bins	Analogy – 5NN
10	4	Stepwise regression	Analogy – 1NN
11	5	None	CART (yes)
12	5	None	CART (no)
13	4	Normalization	CART (yes)
14	4	Normalization	CART (no)
15	7	Natural logarithm	CART (yes)
16	7	Natural logarithm	CART (no)
17	0	Natural logarithm	Analogy – 1NN
18	0	Sequential Forward Selection	Analogy – 1NN
19	0	None	Analogy – 1NN
20	0	Normalization	Analogy – 1NN
21	1	None	Analogy – 5NN
22	1	Normalization	Analogy – 5NN
23	2	Equal frequency – 3 bins	Analogy – 5NN
24	2	Sequential Forward Selection	Analogy – 5NN
25	2	Equal frequency – 5 bins	Analogy – 1NN
26	3	Principal component analysis	Analogy – 5NN
27	3	Equal frequency – 3 bins	CART (yes)
28	3	Equal frequency – 3 bins	CART (no)
29	1	Principal component analysis	Analogy – 1NN
30	2	Sequential Forward Selection	CART (yes)
31	0	Principal component analysis	Stepwise regression
32	8	Equal frequency – 3 bins	Analogy – 1NN
33	1	Stepwise regression	Stepwise regression
34	3	Sequential Forward Selection	CART (no)
35	2	Stepwise regression	Simple linear regression
36	2	None	Stepwise regression
37	2	Normalization	Stepwise regression
38	3	Sequential Forward Selection	Stepwise regression
39	2	Equal width – 5 bins	CART (yes)
40	2	Equal width – 5 bins	CART (no)
41	2	Normalization	Partial least squares regression
42	0	Equal width – 5 bins	Analogy – 5NN
43	3	Equal width – 3 bins	Analogy – 5NN
44	1	Equal width – 3 bins	CART (yes)

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
45	1	Equal width – 3 bins	CART (no)
46	1	None	Simple linear regression
47	0	Sequential Forward Selection	Simple linear regression
48	0	Normalization	Simple linear regression
49	0	Equal width – 3 bins	Stepwise regression
50	0	Equal width – 5 bins	Stepwise regression
51	0	Equal width – 5 bins	Analogy – 1NN
52	2	Equal width – 5 bins	Partial least squares regression
53	1	None	Neural net
54	1	Principal component analysis	Neural net
55	0	Stepwise regression	Neural net
56	4	Natural logarithm	Simple linear regression
57	1	Equal width – 3 bins	Analogy – 1NN
58	1	Sequential Forward Selection	Neural net
59	0	Equal width – 3 bins	Partial least squares regression
60	4	Equal frequency – 3 bins	Principal component regression
61	4	Normalization	Principal component regression
62	1	Natural logarithm	Principal component regression
63	4	Principal component analysis	Simple linear regression
64	2	Equal frequency – 5 bins	Partial least squares regression
65	2	Equal width – 3 bins	Simple linear regression
66	3	Equal width – 5 bins	Simple linear regression
67	4	Equal frequency – 5 bins	Stepwise regression
68	6	Equal width – 5 bins	Principal component regression
69	1	Sequential Forward Selection	Partial least squares regression
70	4	Equal frequency – 3 bins	Partial least squares regression
71	1	Equal frequency – 5 bins	Principal component regression
72	2	Equal frequency – 3 bins	Stepwise regression
73	7	Equal width – 3 bins	Principal component regression
74	2	Equal frequency – 5 bins	Simple linear regression
75	2	Natural logarithm	Stepwise regression
76	1	Natural logarithm	Partial least squares regression
77	5	None	Partial least squares regression
78	5	Principal component analysis	Partial least squares regression
79	3	Stepwise regression	Partial least squares regression
80	10	Equal frequency – 3 bins	Simple linear regression
81	1	Sequential Forward Selection	Principal component regression
82	5	None	Principal component regression

Continued on next page

Rank	**δr**	**Preprocessing option/ combination scheme**	**Learner**
83	5	Principal component analysis	Principal component regression
84	5	Stepwise regression	Principal component regression
85	1	Natural logarithm	Neural net
86	1	Equal frequency – 5 bins	Neural net
87	1	Normalization	Neural net
88	1	Equal width – 3 bins	Neural net
89	1	Equal width – 5 bins	Neural net
90	1	Equal frequency – 3 bins	Neural net

Table H.2: Ranking, in ascending order of losses, of all learners and their related δr values, as obtained by our replication study

Rank	δ**r**	**Preprocessing option/ combination scheme**	**Learner**
1	9	Mean	Top 4
2	9	Median	Top 4
3	6	Mean	Top 8
4	4	IRWM	Top 4
5	1	IRWM	Top 8
6	2	IRWM	Top 12
7	0	Mean	Top 16
8	5	IRWM	Top 16
9	8	Median	Top 8
10	8	Mean	Top 12
11	6	Median	Top 12
12	5	Median	Top 16
13	1	Principal component analysis	CART(yes)
14	1	Principal component analysis	CART(no)
15	1	Mean	Top 2
16	1	Median	Top 2
17	1	IRWM	Top 2
18	0	Equal frequency – 5 bins	Analogy – 5NN
19	0	Stepwise regression	Analogy – 5NN
20	0	Natural logarithm	Analogy – 5NN
21	0	None	CART (yes)
22	0	None	CART (no)
23	3	Normalization	CART (yes)
24	3	Normalization	CART (no)
25	2	Natural logarithm	CART (yes)
26	2	Natural logarithm	CART (no)
27	2	Stepwise regression	Analogy – 1NN
28	0	Equal frequency – 5 bins	CART (yes)
29	0	Equal frequency – 5 bins	CART (no)
30	0	Stepwise regression	CART (yes)
31	0	Stepwise regression	CART (no)
32	0	Natural logarithm	Analogy – 1NN
33	0	Sequential Forward Selection	Analogy – 1NN

Continued on next page

Rank	δr	**Preprocessing option/ combination scheme**	**Learner**
34	0	None	Analogy – 1NN
35	0	Normalization	Analogy – 1NN
36	1	None	Analogy – 5NN
37	1	Normalization	Analogy – 5NN
38	3	Sequential Forward Selection	Analogy – 5NN
39	3	Equal frequency – 3 bins	Analogy – 5NN
40	2	Equal frequency – 5 bins	Analogy – 1NN
41	2	Principal component analysis	Analogy – 5NN
42	3	Equal frequency – 3 bins	CART (yes)
43	3	Equal frequency – 3 bins	CART (no)
44	1	Principal component analysis	Analogy – 1NN
45	1	Sequential Forward Selection	CART (yes)
46	0	Principal component analysis	Stepwise regression
47	8	Equal frequency – 3 bins	Analogy – 1NN
48	1	Stepwise regression	Stepwise regression
49	4	Sequential Forward Selection	CART (no)
50	2	Stepwise regression	Simple linear regression
51	2	None	Stepwise regression
52	2	Normalization	Stepwise regression
53	3	Sequential Forward Selection	Stepwise regression
54	3	Equal width – 5 bins	CART (yes)
55	3	Equal width – 5 bins	CART (no)
56	1	Equal width – 5 bins	Analogy – 5NN
57	3	Normalization	Partial least squares regression
58	3	Equal width – 3 bins	Analogy – 5NN
59	1	Equal width – 3 bins	CART (yes)
60	1	Equal width – 3 bins	CART (no)
61	1	Sequential Forward Selection	Simple linear regression
62	2	None	Simple linear regression
63	0	Normalization	Simple linear regression
64	0	Equal width – 3 bins	Stepwise regression
65	0	Equal width – 5 bins	Stepwise regression
66	0	Equal width – 5 bins	Analogy – 1NN
67	2	Equal width – 5 bins	Partial least squares regression
68	1	None	Neural net
69	1	Principal component analysis	Neural net
70	0	Stepwise regression	Neural net
71	4	Natural logarithm	Simple linear regression

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
72	1	Equal width – 3 bins	Analogy – 1NN
73	1	Sequential Forward Selection	Neural net
74	4	Equal frequency – 3 bins	Principal component regression
75	1	Equal width – 3 bins	Partial least squares regression
76	4	Normalization	Principal component regression
77	1	Natural logarithm	Principal component regression
78	4	Principal component analysis	Simple linear regression
79	2	Equal frequency – 5 bins	Partial least squares regression
80	1	Equal width – 3 bins	Simple linear regression
81	4	Equal width – 5 bins	Simple linear regression
82	4	Equal frequency – 5 bins	Stepwise regression
83	6	Equal width – 5 bins	Principal component regression
84	1	Sequential Forward Selection	Partial least squares regression
85	1	Equal frequency – 3 bins	Stepwise regression
86	3	Equal frequency – 3 bins	Partial least squares regression
87	0	Equal frequency – 5 bins	Principal component regression
88	7	Equal width – 3 bins	Principal component regression
89	2	Equal frequency – 5 bins	Simple linear regression
90	2	Natural logarithm	Stepwise regression
91	1	Natural logarithm	Partial least squares regression
92	5	None	Partial least squares regression
93	5	Principal component analysis	Partial least squares regression
94	3	Stepwise regression	Partial least squares regression
95	10	Equal frequency – 3 bins	Simple linear regression
96	1	Sequential Forward Selection	Principal component regression
97	5	None	Principal component regression
98	5	Principal component analysis	Principal component regression
99	5	Stepwise regression	Principal component regression
100	1	Natural logarithm	Neural net
101	1	Equal frequency – 5 bins	Neural net
102	1	Normalization	Neural net
103	1	Equal width – 3 bins	Neural net
104	1	Equal width – 5 bins	Neural net
105	1	Equal frequency – 3 bins	Neural net

I
Learner Rankings–Control

In the following appendix we provide the rankings, in ascending order of losses, obtained by our bagging experiment control described in Chapter 5. Table I.1 below, lists the rankings of all solo learners. Learners ranked 1 through to 49, corresponding to the top 21 ranks (highlighted in gray), are used in the ensemble creation process. The learners with these rankings all have positive wins−losses values. In others words these learners win more times than they lose during the round-robin evaluation process.

The rankings of all solo learners along with the 17 created ensembles are shown in Table I.2. It can be seen that 13 of these ensembles are ranked first and of these, 11 of them have the lowest δr values seen among the top ranked learners.

Table I.1: Solo learner rankings and δr values, as obtained by our bagging experiment control.

Rank	δ**r**	**Preprocessing option/ combination scheme**	**Learner**
	0	Stepwise regression	Analogy – 5NN
	1	Natural logarithm	Analogy – 5NN
	2	None	Analogy – 5NN
	2	Normalization	Analogy – 5NN
	4	None	CART (yes)

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
1	4	None	CART (no)
	4	Normalization	CART (yes)
	4	Normalization	CART (no)
	4	Natural logarithm	CART (yes)
	4	Natural logarithm	CART (no)
	4	Sequential forward selection	Analogy – 1NN
	4	Sequential forward selection	Analogy – 5NN
	4	Stepwise regression	Analogy – 1NN
	17	None	Analogy – 1NN
	17	Normalization	Analogy – 1NN
16	7	PCA	Analogy – 5NN
17	7	Stepwise regression	CART (yes)
	7	Stepwise regression	CART (no)
19	14	PCA	CART (yes)
	14	PCA	CART (no)
	15	PCA	Analogy – 1NN
22	12	Sequential forward selection	CART (yes)
23	5	Equal frequency – 5 bins	Analogy – 5NN
24	13	Equal frequency – 5 bins	CART (yes)
	13	Equal frequency – 5 bins	CART (no)
26	19	Equal frequency – 3 bins	CART (yes)
	19	Equal frequency – 3 bins	CART (no)
28	7	Sequential forward selection	Stepwise regression
	13	Natural logarithm	Analogy – 1NN
30	14	Equal width – 5 bins	CART (yes)
	14	Equal width – 5 bins	CART (no)
	18	Equal frequency – 3 bins	Analogy – 5NN
33	19	Equal frequency – 3 bins	Analogy – 1NN
34	3	Sequential forward selection	CART (no)
	10	Equal width – 3 bins	Analogy – 5NN
	13	Sequential forward selection	Simple linear regression
	16	Equal width – 5 bins	Analogy – 5NN
38	7	Principal component analysis	Stepwise regression
	12	Equal frequency – 5 bins	Analogy – 1NN
40	13	Stepwise regression	Simple linear regression
41	7	Stepwise regression	Stepwise regression

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
42	3	None	Stepwise regression
	3	Normalization	Stepwise regression
44	16	Normalization	Partial least squares regression
45	17	None	Simple linear regression
46	15	Normalization	Simple linear regression
47	5	Equal width – 3 bins	CART (yes)
	5	Equal width – 3 bins	CART (no)
49	5	Equal width – 3 bins	Stepwise regression
50	0	Equal width – 5 bins	Analogy – 1NN
51	3	Equal width – 5 bins	Stepwise regression
52	7	Equal width – 5 bins	Partial least squares regression
53	3	Equal width – 3 bins	Analogy – 1NN
54	1	Equal width – 3 bins	Partial least squares regression
55	2	None	Neural net
	2	Principal component analysis	Neural net
57	2	Equal width – 5 bins	Principal component regression
	3	Sequential forward selection	Neural net
	3	Normalization	Principal component regression
60	6	Stepwise regression	Neural net
61	1	Natural logarithm	Simple linear regression
62	20	Equal width – 3 bins	Principal component regression
63	6	Natural logarithm	Stepwise regression
64	2	Principal component analysis	Simple linear regression
65	3	Sequential forward selection	Partial least squares regression
	4	Equal width – 3 bins	Simple linear regression
	5	Natural logarithm	Partial least squares regression
	7	Equal frequency – 5 bins	Simple linear regression
69	3	Natural logarithm	Principal component regression
	4	Equal width – 5 bins	Simple linear regression
	8	Equal frequency – 5 bins	Principal component regression
72	11	Equal frequency – 3 bins	Simple linear regression
	11	Equal frequency – 3 bins	Principal component regression
74	5	Equal frequency – 5 bins	Partial least squares regression
75	9	Stepwise regression	Partial least squares regression
76	3	Equal frequency – 3 bins	Partial least squares regression
	10	None	Partial least squares regression
	10	Principal component analysis	Partial least squares regression

Continued on next page

Rank	**δr**	**Preprocessing option/ combination scheme**	**Learner**
79	3	Equal frequency – 5 bins	Stepwise regression
80	8	None	Principal component regression
	8	Principal component analysis	Principal component regression
	10	Sequential forward selection	Principal component regression
83	7	Equal frequency – 3 bins	Stepwise regression
	11	Stepwise regression	Principal component regression
85	2	Natural logarithm	Neural net
	2	Equal frequency – 5	Neural net
87	4	Normalization	Neural net
	4	Equal width – 3 bins	Neural net
	4	Equal width – 5 bins	Neural net
	4	Equal frequency – 3 bins	Neural net

Table I.2: Rankings and δr values for all learners, as obtained by our bagging experiment control.

Rank	δr	Preprocessing option/ combination scheme	Learner
1	0	Mean	Top
	0	Mean	Top 2
	0	IRWM	Top 2
	0	Mean	Top 3
	0	IRWM	Top 3
	0	IRWM	Top 4
	0	IRWM	Top 5
	0	Median	Top Positive
	8	Median	Top
	12	Median	Top 3
	12	Mean	Top 4
	15	Natural Logarithm	Analogy – 5NN
	15	Mean	Top 5
	18	None	Analogy – 5NN
	18	Normalization	Analogy – 5NN
	20	None	CART (yes)
	20	None	CART (no)
	20	Normalization	CART (yes)
	20	Normalization	CART (no)
	20	Natural logarithm	CART (yes)
	20	Natural logarithm	CART (no)
	20	Sequential forward selection	Analogy – 1NN
	20	Sequential forward selection	Analogy – 5NN
	20	Stepwise regression	Analogy – 1NN
	20	IRWM	Top Positive
	34	None	Analogy – 1NN
	34	Normalization	Analogy – 1NN
28	15	Stepwise regression	Analogy – 5NN
	19	Median	Top 2
	19	Median	Top 4
	19	Median	Top 5
32	8	PCA	Analogy – 5NN
33	8	Stepwise regression	CART (yes)
	8	Stepwise regression	CART (no)

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
35	14	PCA	CART (yes)
	14	PCA	CART (no)
37	21	Mean	Top Positive
38	7	Equal frequency – 5 bins	Analogy – 5NN
39	15	Equal frequency – 5 bins	CART (yes)
	15	Equal frequency – 5 bins	CART (no)
41	17	Natural logarithm	Analogy – 1NN
42	23	Equal frequency – 3 bins	Analogy – 5NN
43	10	Principal component analysis	Analogy – 1NN
44	23	Equal frequency – 5 bins	Analogy – 1NN
45	6	Sequential Forward Selection	CART (yes)
46	16	Equal frequency – 3 bins	CART (yes)
	16	Equal frequency – 3 bins	CART (no)
48	10	Sequential Forward Selection	Stepwise regression
	21	Equal frequency – 3 bins	Analogy – 1NN
50	17	Equal width – 5 bins	CART (yes)
	17	Equal width – 5 bins	CART (no)
52	4	Sequential Forward Selection	CART (no)
53	5	Principal component analysis	Stepwise regression
	12	Equal width – 3 bins	Analogy – 5NN
	15	Sequential Forward Selection	Simple linear regression
	18	Equal width – 5 bins	Analogy – 5NN
57	13	Stepwise regression	Simple linear regression
58	7	Stepwise regression	Stepwise regression
59	4	None	Stepwise regression
	4	Normalization	Stepwise regression
61	16	Normalization	Partial least squares regression
62	17	None	Simple linear regression
63	15	Normalization	Simple linear regression
64	5	Equal width – 3 bins	CART (yes)
	5	Equal width – 3 bins	CART (no)
66	5	Equal width – 3 bins	Stepwise regression
67	0	Equal width – 5 bins	Analogy – 1NN
68	3	Equal width – 5 bins	Stepwise regression
69	7	Equal width – 5 bins	Partial least squares regression
70	3	Equal width – 3 bins	Analogy – 1NN
71	1	Equal width – 3 bins	Partial least squares regression

Continued on next page

Rank	δr	Preprocessing option/ combination scheme	Learner
72	2	None	Neural net
	2	Principal component analysis	Neural net
74	2	Equal width – 5 bins	Principal component regression
	3	Sequential Forward Selection	Neural net
	3	Normalization	Principal component regression
77	6	Stepwise regression	Neural net
78	1	Natural logarithm	Simple linear regression
79	20	Equal width – 3 bins	Principal component regression
80	6	Natural logarithm	Stepwise regression
81	2	Principal component analysis	Simple linear regression
82	3	Sequential Forward Selection	Partial least squares regression
	4	Equal width – 3 bins	Simple linear regression
	5	Natural logarithm	Partial least squares regression
	7	Equal frequency – 5 bins	Simple linear regression
86	3	Natural logarithm	Principal component regression
	4	Equal width – 5 bins	Simple linear regression
	8	Equal frequency – 5 bins	Principal component regression
89	11	Equal frequency – 3 bins	Simple linear regression
	11	Equal frequency – 3 bins	Principal component regression
91	5	Equal frequency – 5 bins	Partial least squares regression
92	9	Stepwise regression	Partial least squares regression
93	3	Equal frequency – 3 bins	Partial least squares regression
	10	None	Partial least squares regression
	10	Principal component analysis	Partial least squares regression
96	3	Equal frequency – 5 bins	Stepwise regression
97	8	None	Principal component regression
	8	Principal component analysis	Principal component regression
	10	Sequential Forward Selection	Principal component regression
100	7	Equal frequency – 3 bins	Stepwise regression
	11	Stepwise regression	Principal component regression
102	2	Natural logarithm	Neural net
	2	Equal frequency – 5 bins	Neural net
104	4	Normalization	Neural net
	4	Equal width – 3 bins	Neural net
	4	Equal width – 5 bins	Neural net
	4	Equal frequency – 3 bins	Neural net

J
Learner Rankings–Bagging N

Classifier rankings for all *Bagging N* runs are provided in this Appendix. Tables J.1, J.2, and J.3 list the rankings of solo classifiers, in ascending order of average rank, for losses, wins and wins−losses respectively. Tables J.4, J.5, and J.6 list the rankings of solo and ensemble classifiers, in ascending order of average rank, for losses, wins, and wins−losses respectively. Note that ensemble classifiers are highlighted in gray.

Due to space constraints acronyms had to be used for certain pre-processors and learners in these tables. The acronyms used are explained in the very last row of each table.

Table J.1: **Losses** ranking for solo learners, over 10 runs of *Bagging N*

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stepwise regression	Analogy – 1NN	1.5	1	1	1	1	1	3	4	1	1	1
None	Analogy – 1NN	3.3	4	1	1	1	1	3	10	4	7	1
Normalization	Analogy – 1NN	3.3	4	1	1	1	1	3	10	4	7	1
Stepwise regression	Analogy – 5NN	4.8	8	1	7	5	1	3	1	1	1	20
None	Analogy – 5NN	5.8	18	1	1	13	1	8	1	13	1	1
Normalization	Analogy – 5NN	5.8	18	1	1	13	1	8	1	13	1	1
SFS	Analogy – 5NN	7.4	33	1	7	13	1	3	10	4	1	1
SFS	Analogy – 1NN	8	1	1	10	5	1	1	21	1	15	24
PCA	Analogy – 1NN	10	1	10	7	1	19	22	17	15	7	1
PCA	Analogy – 5NN	14.8	13	17	18	24	20	11	27	7	10	1
SFS	CART (no)	15.8	9	32	1	10	16	30	19	15	10	16
Natural logarithm	Analogy – 5NN	16.4	4	1	10	24	23	11	28	23	6	34
SFS	Linear regression	16.6	4	27	10	10	22	8	22	8	21	34
Continued on next page												

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PCA	CART (yes)	16.9	13	10	27	13	24	11	24	15	12	20
PCA	CART (no)	16.9	13	10	27	13	24	11	24	15	12	20
Stepwise regression	CART (yes)	17.6	39	10	16	24	1	32	4	8	24	18
Stepwise regression	CART (no)	17.6	39	10	16	24	1	32	4	8	24	18
SFS	CART (yes)	17.9	27	15	14	28	11	29	4	15	21	15
None	CART (yes)	18	33	19	19	13	12	22	10	23	28	1
None	CART (no)	18	33	19	19	13	12	22	10	23	28	1
Normalization	CART (yes)	18.7	33	19	21	13	12	20	10	23	35	1
Normalization	CART (no)	18.7	33	19	21	13	12	20	10	23	35	1
Natural logarithm	CART (yes)	18.7	39	19	21	13	17	22	4	23	28	1
Natural logarithm	CART (no)	18.7	39	19	21	13	17	22	4	23	28	1
SFS	Stepwise regression	20.3	13	27	10	10	29	17	37	15	12	33
Equal width – 5 bins	Analogy – 5NN	20.3	24	17	14	34	24	17	24	8	21	20
Equal width – 3 bins	Analogy – 5NN	21.7	18	15	39	31	30	17	20	15	15	17
Stepwise regression	Simple linear regression	24.2	13	31	21	38	27	30	17	12	28	25
Natural logarithm	Analogy – 1NN	25	27	41	41	5	39	1	29	23	15	29
Equal frequency – 3 bins	CART (yes)	25.9	25	25	30	5	46	15	29	34	24	26
Equal frequency – 3 bins	CART (no)	25.9	25	25	30	5	46	15	29	34	24	26
Stepwise regression	Stepwise regression	29.8	9	41	21	41	21	42	37	15	37	34
Equal frequency – 5 bins	CART (yes)	32.5	27	27	30	31	32	45	32	39	28	34
Equal frequency – 5 bins	CART (no)	32.5	27	27	30	31	32	45	32	39	28	34
None	Simple linear regression	34.5	18	48	36	43	32	22	34	23	49	40
Normalization	Stepwise regression	34.8	9	33	30	43	32	34	45	32	44	46
None	Stepwise regression	34.8	9	33	30	43	32	34	45	32	44	46
Normalization	Simple linear regression	36.1	18	48	36	46	32	22	34	36	49	40
PCA	Stepwise regression	37.4	27	45	29	49	32	34	34	46	38	40
Equal frequency – 3 bins	Analogy – 5NN	38.4	33	38	39	37	27	45	41	42	42	40
Equal width – 5 bins	CART (Yes)	38.6	48	38	49	28	49	37	42	49	15	31
Equal width – 5 bins	CART (no)	38.6	48	38	49	28	49	37	42	49	15	31
Equal frequency – 3 bins	Analogy – 1NN	38.8	32	43	36	38	30	49	42	41	38	39
Equal frequency – 5 bins	Analogy – 5NN	39	46	33	42	35	42	41	48	43	20	40
Equal width – 3 bins	Analogy – 1NN	40.3	18	37	42	50	42	42	51	48	47	26
Equal width – 3 bins	CART (yes)	42.2	39	50	47	47	40	37	37	36	40	49
Equal width – 3 bins	CART (no)	42.2	39	50	47	47	40	37	37	36	40	49
Equal frequency – 5 bins	Analogy – 1NN	43.5	47	43	44	35	42	50	47	44	43	40
Equal width – 5 bins	Analogy – 1NN	45.8	50	36	46	40	48	48	49	46	47	48
SFS	PLSR	48.2	69	45	67	41	42	42	22	44	81	29
Equal width – 3 bins	Stepwise regression	50.4	45	53	52	51	51	51	53	49	46	53
Normalization	PLSR	50.7	51	47	45	52	53	53	50	52	52	52
Equal width – 5 bins	Stepwise regression	51.6	51	52	51	52	52	52	51	53	51	51
Equal width – 5 bins	PLSR	53.8	53	53	53	56	54	54	54	54	53	54
None	Neural net	54.8	54	55	54	54	56	55	55	55	55	55
PCA	Neural net	54.8	54	55	54	54	56	55	55	55	55	55
Equal width – 3 bins	PLSR	56.2	56	55	56	58	55	58	55	57	57	55
Stepwise regression	Neural net	57.6	57	58	57	57	59	57	58	57	58	58
Equal width – 3 bins	Simple linear regression	58.6	57	59	58	59	56	58	64	59	54	62
Normalization	PCR	60	59	60	59	62	60	60	59	61	60	60
Equal width – 5 bins	Simple linear regression	61.7	62	62	59	61	61	66	67	61	59	59
Natural logarithm	Simple linear regression	61.8	62	63	59	64	62	61	60	64	63	60
Natural logarithm	Stepwise regression	62.4	62	63	59	64	62	61	60	68	63	62
Equal width – 3 bins	PCR	62.6	61	63	64	63	62	61	64	64	62	62
Equal width – 5 bins	PCR	62.9	62	63	59	64	62	61	63	67	61	67
Natural logarithm	PLSR	64	67	63	64	68	62	61	66	64	63	62
SFS	Neural net	64.4	60	60	67	60	68	70	62	63	67	67
Natural logarithm	PCR	66.2	62	68	64	64	68	67	67	73	67	62

Continued on next page

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PCA	Simple linear regression	67.6	68	74	67	68	62	67	72	60	67	71
Equal frequency – 5 bins	PCR	68.7	69	68	67	71	68	67	70	73	67	67
Equal frequency – 3 bins	PCR	70	69	68	67	71	74	74	70	73	67	67
SFS	PCR	70.5	73	74	75	68	68	74	67	68	67	71
Equal frequency – 5 bins	Simple linear regression	71.6	69	68	67	71	68	77	80	77	63	76
Equal frequency – 5 bins	PLSR	71.7	76	68	67	71	76	70	72	76	67	74
PCA	PLSR	74.4	73	76	77	76	76	70	74	68	78	76
None	PLSR	74.4	73	76	77	76	76	70	74	68	78	76
Stepwise regression	PLSR	74.4	76	80	75	76	74	74	74	68	76	71
Equal frequency – 3 bins	Simple linear regression	76.3	81	68	67	84	68	82	77	81	76	79
Equal frequency – 5 bins	Stepwise regression	77.6	82	76	83	71	79	80	77	82	67	79
Equal frequency – 3 bins	PLSR	78.5	82	76	79	83	83	82	77	82	67	74
PCA	PCR	79.5	78	80	80	79	80	77	80	77	82	82
None	PCR	79.5	78	80	80	79	80	77	80	77	82	82
Stepwise regression	PCR	81.7	80	80	80	81	82	80	83	80	84	87
Equal frequency – 3 bins	Stepwise regression	82.7	84	84	84	82	84	84	84	84	78	79
Equal frequency – 5 bins	Neural net	84.9	85	85	85	85	85	85	85	85	85	84
Natural logarithm	Neural net	84.9	85	85	85	85	85	85	85	85	85	84
Equal width – 5 bins	Neural net	84.9	85	85	85	85	85	85	85	85	85	84
Equal frequency – 3 bins	Neural net	85.2	85	85	85	85	85	85	85	85	85	87
Normalization	Neural net	85.2	85	85	85	85	85	85	85	85	85	87
Equal width – 3 bins	Neural net	85.2	85	85	85	85	85	85	85	85	85	87

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table J.2: **Wins** ranking for solo learners, over 10 runs of *Bagging N*.

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stepwise regression	Analogy – 5NN	3	5	1	1	8	1	5	2	2	4	1
Stepwise regression	Analogy – 1NN	4.6	6	4	5	3	2	11	3	2	6	4
SFS	Simple linear regression	5.6	1	3	8	3	8	1	1	1	8	22
None	Analogy – 1NN	10.3	15	6	8	6	13	14	7	13	14	7
Normalization	Analogy – 1NN	10.3	15	6	8	6	13	14	7	13	14	7
SFS	Analogy – 5NN	10.7	28	6	16	14	2	14	7	10	5	5
None	Analogy – 5NN	11.1	23	6	8	22	2	17	3	20	8	2
Normalization	Analogy – 5NN	11.1	23	6	8	22	2	17	3	20	8	2
SFS	Stepwise regression	11.9	10	4	27	3	18	6	22	9	3	17
Natural logarithm	Analogy – 5NN	16.7	7	2	3	13	22	11	31	24	7	47
Stepwise regression	CART (no)	17.3	39	11	16	10	2	28	7	17	26	17
Stepwise regression	CART (yes)	17.3	39	11	16	10	2	28	7	17	26	17
Stepwise regression	Simple linear regression	19	10	33	13	33	22	27	3	13	8	28
SFS	Analogy – 1NN	19.3	20	13	27	10	13	11	31	7	35	26
Equal width – 5 bins	CART (no)	19.4	18	18	39	1	32	7	22	39	1	17
Equal width – 5 bins	CART (yes)	19.4	18	18	39	1	32	7	22	39	1	17
Stepwise regression	Stepwise regression	19.5	12	32	20	30	20	19	16	7	14	25
Equal width – 3 bins	CART (no)	20.5	7	40	27	34	27	7	7	4	14	38
Equal width – 3 bins	CART (yes)	20.5	7	40	27	34	27	7	7	4	14	38
None	Simple linear regression	20.7	3	43	5	34	19	2	27	4	42	28
None	CART (no)	21.6	28	25	21	14	9	42	16	24	28	9
None	CART (yes)	21.6	28	25	21	14	9	42	16	24	28	9
PCA	Analogy – 5NN	22.2	35	18	27	24	22	19	35	17	19	6
Natural logarithm	CART (no)	22.2	39	25	21	14	13	42	7	24	28	9
Natural logarithm	CART (yes)	22.2	39	25	21	14	13	42	7	24	28	9
Normalization	CART (no)	22.3	28	25	21	14	9	42	16	24	35	9
Normalization	CART (yes)	22.3	28	25	21	14	9	42	16	24	35	9
Normalization	Simple linear regression	22.5	3	43	4	41	22	2	27	13	42	28
Normalization	Stepwise regression	22.8	12	14	13	34	27	23	25	10	39	31
None	Stepwise regression	22.8	12	14	13	34	27	23	25	10	39	31
PCA	Stepwise regression	23.9	15	33	2	48	22	23	27	24	19	26
PCA	Analogy – 1NN	25.6	23	25	16	14	40	39	30	39	21	9
SFS	CART (yes)	28.5	39	21	27	30	20	39	16	36	41	16
SFS	CART (no)	29.2	27	38	7	27	34	48	31	34	24	22
Natural logarithm	Analogy – 1NN	29.2	23	49	46	9	42	4	36	24	24	35
Equal width – 3 bins	Stepwise regression	30.2	2	51	35	50	27	23	38	23	12	41
PCA	CART (no)	31.8	20	16	42	27	44	21	42	44	21	41
PCA	CART (yes)	31.8	20	16	42	27	44	21	42	44	21	41
Equal width – 3 bins	Analogy – 5NN	32.1	35	21	36	46	38	33	31	24	35	22
Equal width – 5 bins	Analogy – 5NN	32.7	35	21	36	34	42	33	38	22	28	38
Normalization	PLSR	36.5	28	33	27	50	38	39	38	34	42	36
Equal frequency – 5 bins	Analogy – 5NN	38	44	21	48	41	44	28	46	47	13	48
Equal width – 5 bins	Stepwise regression	38.7	28	46	27	55	36	33	36	39	46	41
Equal width – 5 bins	Analogy – 1NN	39.4	44	42	41	34	36	33	44	39	45	36
Equal width – 3 bins	Analogy – 1NN	40.6	35	43	38	50	34	28	52	46	49	31
Equal frequency – 5 bins	CART (no)	43	47	33	42	41	44	49	50	48	28	48
Equal frequency – 5 bins	CART (yes)	43	47	33	42	41	44	49	50	48	28	48
SFS	PLSR	43.4	65	39	61	30	40	32	38	36	62	31
Equal frequency – 3 bins	CART (no)	44	47	46	48	24	50	37	46	48	46	48
Equal frequency – 3 bins	CART (yes)	44	47	46	48	24	50	37	46	48	46	48
Equal width – 5 bins	PLSR	47.1	51	49	47	50	49	49	45	36	50	45
Equal frequency – 5 bins	Analogy – 1NN	48.4	46	51	48	41	50	49	53	48	50	48
Equal frequency – 3 bins	Analogy – 5NN	50.1	51	51	52	47	54	49	46	53	50	48

Continued on next page

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Equal frequency – 3 bins	Analogy – 1NN	52	51	57	52	48	54	49	53	54	54	48
Equal width – 3 bins	PLSR	52.8	54	51	54	54	50	56	55	54	55	45
Stepwise regression	Neural net	56	55	55	55	56	56	57	56	57	57	56
Equal width – 3 bins	Simple linear regression	56.4	56	56	58	56	56	55	59	56	53	59
None	Neural net	57.5	56	59	56	59	58	57	57	58	58	57
PCA	Neural net	57.5	56	59	56	59	58	57	57	58	58	57
Equal width – 5 bins	Simple linear regression	60.3	61	61	59	61	61	61	62	61	56	60
Normalization	PCR	60.5	59	61	60	62	60	60	60	62	60	61
SFS	Neural net	61.9	60	58	61	58	63	67	60	60	63	69
Equal width – 5 bins	PCR	62.8	62	63	63	68	62	62	62	63	61	62
PCA	PLSR	65.1	63	65	67	64	65	63	67	65	65	67
None	PLSR	65.1	63	65	67	64	65	63	67	65	65	67
PCA	Simple linear regression	65.8	69	65	64	63	65	68	69	63	63	69
Natural logarithm	Simple linear regression	66.6	65	68	70	68	68	65	62	67	70	63
SFS	PCR	66.6	72	64	64	64	63	72	66	70	67	64
Natural logarithm	Stepwise regression	67.8	69	68	67	68	69	68	62	72	70	65
Stepwise regression	PLSR	68	65	68	64	64	69	71	72	68	68	71
Equal width – 3 bins	PCR	69.5	65	68	78	78	69	66	69	69	68	65
Natural logarithm	PLSR	70.5	72	68	71	72	72	68	71	70	70	71
Natural logarithm	PCR	71.5	69	73	71	71	73	72	72	73	70	71
PCA	PCR	75	72	73	71	72	73	72	74	73	86	84
None	PCR	75	72	73	71	72	73	72	74	73	86	84
Stepwise regression	PCR	75.4	76	73	71	72	73	72	74	73	86	84
Equal frequency – 5 bins	PCR	76.9	81	80	78	80	81	72	74	79	70	74
Equal frequency – 3 bins	PLSR	77.7	77	77	76	72	79	80	80	79	76	81
Equal frequency – 5 bins	Simple linear regression	78.8	81	80	78	80	73	81	82	79	70	84
Equal frequency – 5 bins	PLSR	79	78	80	78	77	79	78	78	77	84	81
Equal frequency – 3 bins	Stepwise regression	79.3	79	77	78	78	81	78	81	77	83	81
Equal frequency – 5 bins	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Natural logarithm	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Equal width – 5 bins	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Equal frequency – 3 bins	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Normalization	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Equal width – 3 bins	Neural net	79.3	81	80	78	80	81	81	82	79	76	75
Equal frequency – 5 bins	Stepwise regression	80	79	77	77	80	81	81	78	79	84	84
Equal frequency – 3 bins	Simple linear regression	80.4	81	80	78	80	73	81	82	79	86	84
Equal frequency – 3 bins	PCR	81.2	81	80	78	80	81	81	82	79	86	84

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table J.3: **Wins − Losses** ranking for solo learners, over 10 runs of *Bagging N*.

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stepwise regression	Analogy – 5NN	2.3	2	1	1	8	1	3	1	1	1	4
Stepwise regression	Analogy – 1NN	2.6	2	3	2	1	2	5	4	1	3	3
None	Analogy – 1NN	6.8	9	4	5	2	8	6	11	6	10	7
Normalization	Analogy – 1NN	6.8	9	4	5	2	8	6	11	6	10	7
SFS	Analogy – 5NN	8.8	30	4	10	14	2	6	11	5	2	4
None	Analogy – 5NN	9.4	23	4	5	24	2	9	2	18	6	1
Normalization	Analogy – 5NN	9.4	23	4	5	24	2	9	2	18	6	1
SFS	Simple linear regression	9.5	1	12	9	4	18	1	5	3	10	32
SFS	Analogy – 1NN	12.9	14	9	15	10	8	4	22	4	22	21
SFS	Stepwise regression	14.8	7	15	15	4	22	14	26	9	6	30
Natural logarithm	Analogy – 5NN	15.6	4	2	2	14	24	9	27	26	6	42
Stepwise regression	CART (no)	16.7	40	10	13	12	2	31	7	12	22	18
Stepwise regression	CART (yes)	16.7	40	10	13	12	2	31	7	12	22	18
PCA	Analogy – 1NN	17.4	15	16	10	10	23	33	19	26	13	9
PCA	Analogy – 5NN	18.1	25	16	23	26	20	15	28	9	13	6
Stepwise regression	Simple linear regression	19.2	7	31	15	35	25	30	5	9	13	22
None	CART (no)	21	30	21	23	14	8	36	15	26	28	9
None	CART (yes)	21	30	21	23	14	8	36	15	26	28	9
Normalization	CART (no)	22.1	30	21	28	14	8	33	15	26	37	9
Normalization	CART (yes)	22.1	30	21	28	14	8	33	15	26	37	9
Natural logarithm	CART (no)	22.4	40	21	28	14	15	36	7	26	28	9
Natural logarithm	CART (yes)	22.4	40	21	28	14	15	36	7	26	28	9
SFS	CART (no)	22.4	21	33	4	26	20	42	20	24	17	17
Stepwise regression	Stepwise regression	23.4	9	39	20	38	19	22	25	8	21	33
SFS	CART (yes)	23.5	30	16	18	30	17	36	11	25	36	16
Equal width – 3 bins	Analogy – 5NN	24.9	26	16	34	34	33	22	21	23	22	18
Equal width – 5 bins	Analogy – 5NN	25.3	29	20	22	31	34	22	33	17	22	23
PCA	CART (no)	25.5	19	12	32	28	36	16	36	35	17	24
PCA	CART (yes)	25.5	19	12	32	28	36	16	36	35	17	24
None	Simple linear regression	27.4	5	49	20	43	25	9	28	12	48	35
Natural logarithm	Analogy – 1NN	27.6	26	44	41	9	40	2	38	26	20	30
Normalization	Simple linear regression	28.7	5	49	18	46	27	9	28	22	48	35
Normalization	Stepwise regression	29.2	9	27	23	43	29	22	39	18	43	39
None	Stepwise regression	29.2	9	27	23	43	29	22	39	18	43	39
PCA	Stepwise regression	30	21	41	12	49	27	22	28	39	27	34
Equal width – 5 bins	CART (no)	30.3	36	34	50	6	44	18	33	51	4	27
Equal width – 5 bins	CART (yes)	30.3	36	34	50	6	44	18	33	51	4	27
Equal width – 3 bins	CART (no)	31	17	46	43	47	31	18	22	15	28	43
Equal width – 3 bins	CART (yes)	31	17	46	43	47	31	18	22	15	28	43
Equal frequency – 3 bins	CART (no)	35.1	38	36	37	14	50	22	41	37	39	37
Equal frequency – 3 bins	CART (yes)	35.1	38	36	37	14	50	22	41	37	39	37
Equal frequency – 5 bins	CART (no)	37.6	40	29	34	31	41	46	43	41	28	43
Equal frequency – 5 bins	CART (yes)	37.6	40	29	34	31	41	46	43	41	28	43
Equal width – 3 bins	Analogy – 1NN	40.3	26	40	37	50	34	42	52	48	50	24
Equal frequency – 5 bins	Analogy – 5NN	40.7	48	31	46	36	47	41	47	47	16	48
Equal width – 5 bins	Analogy – 1NN	43.5	49	38	47	38	38	45	49	43	47	41
Equal frequency – 3 bins	Analogy – 5NN	44.6	47	44	45	41	41	46	45	44	45	48
Equal frequency – 3 bins	Analogy – 1NN	45.3	46	51	41	42	44	50	46	44	42	47
Equal width – 3 bins	Stepwise regression	45.8	15	54	52	51	47	49	52	44	41	53
SFS	PLSR	45.9	69	43	62	38	38	44	28	40	70	27
Equal frequency – 5 bins	Analogy – 1NN	46.8	49	46	47	36	49	51	48	48	46	48
Normalization	PLSR	49.3	51	42	40	52	53	53	49	50	51	52

Continued on next page

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Equal width – 5 bins	Stepwise regression	51.2	51	52	47	54	52	52	49	53	51	51
Equal width – 5 bins	PLSR	53.5	53	53	53	53	54	54	54	54	53	54
Equal width – 3 bins	PLSR	55.5	54	55	56	57	55	58	55	55	55	55
None	Neural net	56	54	58	54	55	58	55	56	58	56	56
PCA	Neural net	56	54	58	54	55	58	55	56	58	56	56
Stepwise regression	Neural net	57.4	57	56	57	58	57	59	58	56	58	58
Equal width – 3 bins	Simple linear regression	57.6	58	57	58	59	56	57	61	56	54	60
Normalization	PCR	60.4	59	61	60	62	60	60	59	62	60	61
Equal width – 5 bins	Simple linear regression	60.8	61	62	59	61	61	61	65	61	58	59
SFS	Neural net	62.3	60	60	62	60	63	69	60	60	62	67
Equal width – 5 bins	PCR	62.6	62	63	61	64	62	62	64	64	61	63
Natural logarithm	Simple linear regression	64.2	64	65	65	64	66	63	62	65	66	62
Natural logarithm	Stepwise regression	65.7	65	65	64	64	67	67	62	72	66	65
PCA	Simple linear regression	66.1	69	65	65	63	63	69	72	63	62	70
Equal width – 3 bins	PCR	66.9	63	65	73	73	67	64	67	68	64	65
SFS	PCR	67.1	73	64	67	64	63	74	66	71	65	64
Natural logarithm	PLSR	68.3	71	65	69	72	69	67	68	69	66	67
PCA	PLSR	68.6	67	70	69	69	70	64	69	65	70	73
None	PLSR	68.6	67	70	69	69	70	64	69	65	70	73
Natural logarithm	PCR	69.9	65	72	69	68	73	71	71	73	70	67
Stepwise regression	PLSR	71.5	72	73	67	69	72	73	74	69	75	71
Equal frequency – 5 bins	PCR	74.9	76	79	74	78	80	71	73	77	70	71
PCA	PCR	76.2	74	74	74	75	76	75	76	74	82	82
None	PCR	76.2	74	74	74	75	76	75	76	74	82	82
Equal frequency – 5 bins	PLSR	76.3	76	79	74	74	78	77	75	77	77	76
Equal frequency – 5 bins	Simple linear regression	76.8	76	79	74	78	73	80	83	80	66	79
Equal frequency – 3 bins	PCR	77.4	76	79	74	78	81	79	76	77	79	75
Stepwise regression	PCR	78.9	80	74	74	77	78	77	79	76	84	90
Equal frequency – 3 bins	Simple linear regression	80	82	79	74	84	73	84	82	81	81	80
Equal frequency – 3 bins	PLSR	80.3	81	77	82	82	83	82	81	83	76	76
Equal frequency – 5 bins	Stepwise regression	80.3	82	77	83	78	82	81	80	83	77	80
Equal frequency – 3 bins	Stepwise regression	82.4	84	84	84	83	84	82	84	81	80	78
Equal frequency – 5 bins	Neural net	84.6	85	85	85	85	85	85	85	85	84	82
Natural logarithm	Neural net	84.6	85	85	85	85	85	85	85	85	84	82
Equal width – 5 bins	Neural net	84.6	85	85	85	85	85	85	85	85	84	82
Equal frequency – 3 bins	Neural net	85.1	85	85	85	85	85	85	85	85	84	87
Normalization	Neural net	85.1	85	85	85	85	85	85	85	85	84	87
Equal width – 3 bins	Neural net	85.1	85	85	85	85	85	85	85	85	84	87

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table J.4: **Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging N*. Ensembles are highlighted gray.

Pre-processing	Learner	Av.	Run Rank									
Option		Rank	1	2	3	4	5	6	7	8	9	10
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 4	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 5	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 3	2.1	1	1	1	1	1	1	1	12	1	1
Median	Top Positive	2.3	1	1	1	1	1	1	14	1	1	1
IRWM	Top 2	3.4	1	1	1	1	1	1	14	12	1	1
Mean	Top Overall	4	1	19	1	1	1	13	1	1	1	1
Mean	Top 2	4.6	1	1	1	1	1	13	14	12	1	1
Mean	Top	5.2	1	1	1	19	1	1	14	12	1	1
Median	Top	5.8	1	19	20	1	1	1	1	12	1	1
IRWM	Top Positive	8.1	1	1	1	1	23	24	1	1	27	1
IRWM	Top Overall	8.5	1	19	1	1	1	13	14	12	1	22
Median	Top 5	9.8	1	19	24	1	23	13	14	1	1	1
Median	Top 3	9.8	19	19	20	1	23	1	1	12	1	1
None	Analogy – 1NN	10.2	19	1	1	1	1	13	28	12	25	1
Normalization	Analogy – 1NN	10.2	19	1	1	1	1	13	28	12	25	1
Median	Top 4	10.5	1	19	20	1	23	13	14	12	1	1
Median	Top 2	12.6	19	1	24	19	1	13	14	12	1	22
Stepwise regression	Analogy – 1NN	14.6	1	19	1	19	1	24	28	12	19	22
Median	Top Overall	15.1	19	19	20	19	23	13	14	1	1	22
SFS	Analogy – 5NN	16.3	37	1	24	27	1	13	28	12	19	1
None	Analogy – 5NN	16.6	31	1	1	36	1	24	1	30	19	22
Normalization	Analogy – 5NN	16.6	31	1	1	36	1	24	1	30	19	22
SFS	Analogy – 1NN	21.4	1	1	40	23	23	1	39	1	48	37
Stepwise regression	Analogy – 5NN	23.5	26	19	35	26	23	24	14	12	19	37
Stepwise regression	CART (yes)	29.1	44	29	30	36	1	41	14	28	33	35
Stepwise regression	CART (no)	29.1	44	29	30	36	1	41	14	28	33	35
PCA	Analogy – 5NN	29.2	33	34	32	34	36	29	40	26	27	1
Natural logarithm	Analogy – 5NN	30.6	19	19	24	34	40	32	41	32	24	41
PCA	Analogy – 1NN	30.6	26	41	24	23	38	41	37	40	35	1
Mean	Top Positive	30.8	19	29	29	36	39	44	1	26	45	40
Natural logarithm	CART (yes)	31.5	44	35	35	27	34	37	14	32	35	22
Natural logarithm	CART (no)	31.5	44	35	35	27	34	37	14	32	35	22
None	CART (yes)	31.6	37	35	33	27	30	37	28	32	35	22
None	CART (no)	31.6	37	35	33	27	30	37	28	32	35	22
Normalization	CART (yes)	32.4	37	35	35	27	30	35	28	32	43	22
Normalization	CART (no)	32.4	37	35	35	27	30	35	28	32	43	22
Natural logarithm	Analogy – 1NN	34.9	34	47	48	23	41	13	41	32	31	39
PCA	CART (yes)	37	29	29	40	36	41	29	47	49	29	41
PCA	CART (no)	37	29	29	40	36	41	29	47	49	29	41
SFS	CART (no)	39.2	28	57	1	45	46	56	36	40	42	41
EF – 5 bins	CART (yes)	41.5	34	41	44	46	41	46	45	42	35	41
EF – 5 bins	CART (no)	41.5	34	41	44	46	41	46	45	42	35	41
SFS	CART (yes)	46	59	45	43	53	36	55	37	46	52	34
EF – 3 bins	CART (yes)	46.6	44	52	49	36	62	33	41	42	53	54
EF – 3 bins	CART (no)	46.6	44	52	49	36	62	33	41	42	53	54
EF – 5 bins	Analogy – 5NN	46.9	51	46	49	46	47	45	53	52	32	48
EF – 3 bins	Analogy – 1NN	47.3	37	50	47	50	47	49	50	48	48	47
EF – 3 bins	Analogy – 5NN	49.3	50	49	55	53	50	46	49	47	46	48
EF – 5 bins	Analogy – 1NN	49.9	52	50	52	46	47	52	51	55	46	48

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EW – 3 bins	Analogy – 5NN	51.7	43	44	61	57	55	51	53	52	50	51
EW – 5 bins	Analogy – 5NN	52.4	58	47	46	58	52	52	55	51	53	52
SFS	SLR	53.9	52	54	52	51	52	50	57	52	58	61
SFS	SWR	55.3	59	54	52	51	55	54	61	57	50	60
Stepwise regression	SLR	56	59	54	56	59	54	59	52	55	59	53
Stepwise regression	SWR	59.5	55	64	56	61	51	69	61	57	60	61
Normalization	SWR	60.9	55	58	59	63	57	60	67	60	64	66
None	SWR	60.9	55	58	59	63	57	60	67	60	64	66
None	SLR	61.8	62	68	61	63	59	56	58	59	69	63
PCA	SWR	62.3	64	65	58	69	59	60	58	66	61	63
Normalization	SLR	62.4	62	68	61	66	59	56	58	62	69	63
EW – 5 bins	CART (yes)	63.1	68	62	69	55	69	63	65	69	53	58
EW – 5 bins	CART (no)	63.1	68	62	69	55	69	63	65	69	53	58
EW – 3 bins	Analogy – 1NN	64.3	54	61	64	70	67	68	71	66	66	56
EW – 3 bins	CART (yes)	64.8	65	70	67	67	62	63	61	62	62	69
EW – 3 bins	CART (no)	64.8	65	70	67	67	62	63	61	62	62	69
EW – 5 bins	Analogy – 1NN	66.1	70	58	66	60	68	70	69	66	66	68
EW – 3 bins	SWR	70.6	67	73	72	71	71	71	73	69	66	73
Normalization	PLSR	70.7	71	67	65	72	73	73	70	72	72	72
SFS	PLSR	71	89	65	87	61	62	67	56	65	101	57
EW – 5 bins	SWR	71.6	71	72	71	72	72	72	71	73	71	71
EW – 5 bins	PLSR	73.8	73	73	73	76	74	74	74	74	73	74
None	Neural net	74.8	74	75	74	74	76	75	75	75	75	75
PCA	Neural net	74.8	74	75	74	74	76	75	75	75	75	75
EW – 3 bins	PLSR	76.2	76	75	76	78	75	78	75	77	77	75
Stepwise regression	Neural net	77.6	77	78	77	77	79	77	78	77	78	78
EF – 3 bins	SLR	78.6	77	79	78	79	76	78	84	79	74	82
Normalization	PCR	80	79	80	79	82	80	80	79	81	80	80
EW – 5 bins	SLR	81.7	82	82	79	81	81	86	87	81	79	79
Natural logarithm	SLR	81.8	82	83	79	84	82	81	80	84	83	80
Natural logarithm	SWR	82.4	82	83	79	84	82	81	80	88	83	82
EW – 3 bins	PCR	82.6	81	83	84	83	82	81	84	84	82	82
EW – 5 bins	PCR	82.9	82	83	79	84	82	81	83	87	81	87
Natural logarithm	PLSR	84	87	83	84	88	82	81	86	84	83	82
SFS	Neural net	84.4	80	80	87	80	88	90	82	83	87	87
Natural logarithm	PCR	86.2	82	88	84	84	88	87	87	93	87	82
PCA	SLR	87.6	88	94	87	88	82	87	92	80	87	91
EF – 5 bins	PCR	88.7	89	88	87	91	88	87	90	93	87	87
EF – 3 bins	PCR	90	89	88	87	91	94	94	90	93	87	87
SFS	PCR	90.5	93	94	95	88	88	94	87	88	87	91
EF – 5 bins	SLR	91.6	89	88	87	91	88	97	100	97	83	96
EF – 5 bins	PLSR	91.7	96	88	87	91	96	90	92	96	87	94
PCA	PLSR	94.4	93	96	97	96	96	90	94	88	98	96
None	PLSR	94.4	93	96	97	96	96	90	94	88	98	96
Stepwise regression	PLSR	94.4	96	100	95	96	94	94	94	88	96	91
EF – 3 bins	SLR	96.3	101	88	87	104	88	102	97	101	96	99
EF – 5 bins	SWR	97.6	102	96	103	91	99	100	97	102	87	99
EF – 3 bins	PLSR	98.5	102	96	99	103	103	102	97	102	87	94
PCA	PCR	99.5	98	100	100	99	100	97	100	97	102	102
None	PCR	99.5	98	100	100	99	100	97	100	97	102	102
Stepwise regression	PCR	101.7	100	100	100	101	102	100	103	100	104	107
EF – 3 bins	SWR	102.7	104	104	104	102	104	104	104	104	98	99
EF – 5 bins	Neural net	104.9	105	105	105	105	105	105	105	105	105	104
Natural logarithm	Neural net	104.9	105	105	105	105	105	105	105	105	105	104
EW – 5 bins	Neural net	104.9	105	105	105	105	105	105	105	105	105	104

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EF – 3 bins	Neural net	105.2	105	105	105	105	105	105	105	105	105	107
Normalization	Neural net	105.2	105	105	105	105	105	105	105	105	105	107
EW – 3 bins	Neural net	105.2	105	105	105	105	105	105	105	105	105	107

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression **SLR**–Simple linear regression **SWR**–Stepwise regression
EF–Equal frequency **EW**–Equal width

Table J.5: **Wins** ranking for ensemble and solo classifiers, over 10 runs of *Bagging N*. Ensembles are highlighted gray.

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top 5	3.2	8	1	2	1	6	1	7	1	1	4
Median	Top positive	4.3	3	1	5	7	2	5	7	2	10	1
Median	Top 4	6.4	8	1	3	3	4	1	10	4	10	20
Median	Top 2	6.6	4	1	5	2	1	16	2	9	10	16
Median	Top 3	6.7	1	6	3	3	8	3	10	6	7	20
Median	Top	7.4	11	6	8	15	2	6	10	2	10	4
IRWM	Top 2	8.4	13	13	12	7	8	6	7	9	5	4
Median	Top overall	9.1	13	1	5	9	7	16	4	22	2	12
IRWM	Top 3	9.8	13	13	21	9	8	6	10	9	5	4
Mean	Top	10	27	13	21	9	8	6	4	6	2	4
Mean	Top 2	10.2	13	13	12	15	8	16	10	9	2	4
Stepwise regression	Analogy – 5NN	10.4	8	6	1	28	4	23	4	9	19	2
IRWM	Top 4	10.9	13	13	12	17	8	6	10	16	10	4
IRWM	Top overall	11.7	13	10	12	17	8	16	10	9	10	12
IRWM	Top 5	11.8	13	13	12	17	8	6	10	16	19	4
Mean	Top overall	11.8	13	10	10	17	8	16	10	22	10	2
Mean	Top 3	15	13	20	12	17	8	16	24	16	10	14
SFS	SLR	16.3	1	19	28	9	26	4	2	6	26	42
Mean	Top 4	17.2	27	23	12	17	18	6	10	26	19	14
Stepwise regression	Analogy – 1NN	17.2	11	20	21	9	18	31	10	9	23	20
Mean	Top positive	18.3	27	6	26	17	27	24	1	5	30	20
Mean	Top 5	19.3	27	23	12	17	18	6	24	26	24	16
IRWM	Top positive	25.6	27	23	21	17	32	26	26	28	30	26
SFS	SWR	28	25	20	47	9	38	24	42	28	10	37
None	Analogy – 5NN	28.3	43	23	28	42	18	37	10	40	26	16
Normalization	Analogy – 5NN	28.3	43	23	28	42	18	37	10	40	26	16
None	Analogy – 1NN	29	35	23	28	17	33	34	27	33	34	26
Normalization	Analogy – 1NN	29	35	23	28	17	33	34	27	33	34	26
SFS	Analogy – 5NN	29.6	48	23	36	34	18	34	27	30	22	24
Natural logarithm	Analogy – 5NN	32.5	13	10	10	33	42	31	51	44	24	67
EW – 5 bins	CART (no)	36.3	38	38	59	5	52	26	42	59	7	37
EW – 5 bins	CART (yes)	36.3	38	38	59	5	52	26	42	59	7	37
None	SLR	36.5	6	63	21	54	39	6	47	19	62	48
Stepwise regression	CART (no)	36.9	59	31	36	30	18	48	27	37	46	37
Stepwise regression	CART (yes)	36.9	59	31	36	30	18	48	27	37	46	37
Stepwise regression	SLR	37	25	53	33	53	42	47	10	33	26	48
Normalization	SLR	38	6	63	12	61	42	6	47	33	62	48
EW – 3 bins	CART (no)	38.5	13	60	47	54	47	26	27	19	34	58
EW – 3 bins	CART (yes)	38.5	13	60	47	54	47	26	27	19	34	58
SFS	Analogy – 1NN	38.8	40	33	47	30	33	31	51	22	55	46
Stepwise regression	SWR	39	32	52	40	50	40	39	36	22	34	45
None	CART (no)	41.4	48	45	41	34	27	62	36	44	48	29
None	CART (yes)	41.4	48	45	41	34	27	62	36	44	48	29
PCA	Analogy – 5NN	42.1	55	38	47	44	42	39	55	37	39	25
Normalization	CART (no)	42.1	48	45	41	34	27	62	36	44	55	29
Normalization	CART (yes)	42.1	48	45	41	34	27	62	36	44	55	29
Natural logarithm	CART (no)	42.2	59	45	41	34	33	62	27	44	48	29
Natural logarithm	CART (yes)	42.2	59	45	41	34	33	62	27	44	48	29
PCA	SWR	42.6	35	53	9	68	42	43	47	44	39	46
Normalization	SWR	42.8	32	34	33	54	47	43	45	30	59	51
None	SWR	42.8	32	34	33	54	47	43	45	30	59	51
PCA	Analogy – 1NN	45.6	43	45	36	34	60	59	50	59	41	29

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Natural logarithm	Analogy – 1NN	48.4	43	69	66	29	62	16	56	44	44	55
EW – 3 bins	SWR	48.4	4	71	55	70	47	43	58	43	32	61
SFS	CART (yes)	48.5	59	41	47	50	40	59	36	56	61	36
SFS	CART (no)	49.1	47	58	26	47	54	68	51	54	44	42
PCA	CART (no)	51.8	40	36	62	47	64	41	62	64	41	61
PCA	CART (yes)	51.8	40	36	62	47	64	41	62	64	41	61
EW – 3 bins	Analogy – 5NN	52.1	55	41	56	66	58	53	51	44	55	42
EW – 5 bins	Analogy – 5NN	52.7	55	41	56	54	62	53	58	42	48	58
Normalization	PLSR	56.5	48	53	47	70	58	59	58	54	62	56
EF – 5 bins	Analogy – 5NN	58	64	41	68	61	64	48	66	67	33	68
EW – 5 bins	SWR	58.7	48	66	47	75	56	53	56	59	66	61
EW – 5 bins	Analogy – 1NN	59.4	64	62	61	54	56	53	64	59	65	56
EW – 3 bins	Analogy – 1NN	60.6	55	63	58	70	54	48	72	66	69	51
EF – 5 bins	CART (no)	63	67	53	62	61	64	69	70	68	48	68
EF – 5 bins	CART (yes)	63	67	53	62	61	64	69	70	68	48	68
SFS	PLSR	63.4	85	59	81	50	60	52	58	56	82	51
EF – 3 bins	CART (no)	64	67	66	68	44	70	57	66	68	66	68
EF – 3 bins	CART (yes)	64	67	66	68	44	70	57	66	68	66	68
EW – 5 bins	PLSR	67.1	71	69	67	70	69	69	65	56	70	65
EF – 5 bins	Analogy – 1NN	68.4	66	71	68	61	70	69	73	68	70	68
EF – 3 bins	Analogy – 5NN	70.1	71	71	72	67	74	69	66	73	70	68
EF – 3 bins	Analogy – 1NN	72	71	77	72	68	74	69	73	74	74	68
EW – 3 bins	PLSR	72.8	74	71	74	74	70	76	75	74	75	65
Stepwise regression	Neural net	76	75	75	75	76	76	77	76	77	77	76
EW – 3 bins	SLR	76.4	76	76	78	76	76	75	79	76	73	79
None	Neural net	77.5	76	79	76	79	78	77	77	78	78	77
PCA	Neural net	77.5	76	79	76	79	78	77	77	78	78	77
EW – 5 bins	SLR	80.3	81	81	79	81	81	81	82	81	76	80
Normalization	PCR	80.5	79	81	80	82	80	80	80	82	80	81
SFS	Neural net	81.9	80	78	81	78	83	87	80	80	83	89
EW – 5 bins	PCR	82.8	82	83	83	88	82	82	82	83	81	82
PCA	PLSR	85.1	83	85	87	84	85	83	87	85	85	87
None	PLSR	85.1	83	85	87	84	85	83	87	85	85	87
PCA	SLR	85.8	89	85	84	83	85	88	89	83	83	89
Natural logarithm	SLR	86.6	85	88	90	88	88	85	82	87	90	83
SFS	PCR	86.6	92	84	84	84	83	92	86	90	87	84
Natural logarithm	SWR	87.8	89	88	87	88	89	88	82	92	90	85
Stepwise regression	PLSR	88	85	88	84	84	89	91	92	88	88	91
EW – 3 bins	PCR	89.5	85	88	98	98	89	86	89	89	88	85
Natural logarithm	PLSR	90.5	92	88	91	92	92	88	91	90	90	91
Natural logarithm	PCR	91.5	89	93	91	91	93	92	92	93	90	91
PCA	PCR	95	92	93	91	92	93	92	94	93	106	104
None	PCR	95	92	93	91	92	93	92	94	93	106	104
Stepwise regression	PCR	95.4	96	93	91	92	93	92	94	93	106	104
EF – 5 bins	PCR	96.9	101	100	98	100	101	92	94	99	90	94
EF – 3 bins	PLSR	97.7	97	97	96	92	99	100	100	99	96	101
EF – 5 bins	SLR	98.8	101	100	98	100	93	101	102	99	90	104
EF – 5 bins	PLSR	99	98	100	98	97	99	98	98	97	104	101
EF – 3 bins	SWR	99.3	99	97	98	98	101	98	101	97	103	101
EF – 5 bins	Neural net	99.3	101	100	98	100	101	101	102	99	96	95
Natural logarithm	Neural net	99.3	101	100	98	100	101	101	102	99	96	95
EW – 5 bins	Neural net	99.3	101	100	98	100	101	101	102	99	96	95
EF – 3 bins	Neural net	99.3	101	100	98	100	101	101	102	99	96	95
Normalization	Neural net	99.3	101	100	98	100	101	101	102	99	96	95
EW – 3 bins	Neural net	99.3	101	100	98	100	101	101	102	99	96	95

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EF – 5 bins	SWR	100	99	97	97	100	101	101	98	99	104	104
EF – 3 bins	SLR	100.4	101	100	98	100	93	101	102	99	106	104
EF – 3 bins	PCR	101.2	101	100	98	100	101	101	102	99	106	104

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression
EF–Equal frequency **EW**–Equal width

Table J.6: **Wins − Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging N*. Ensembles are highlighted gray.

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top 5	2.7	4	3	1	1	6	1	6	1	1	3
Median	Top positive	3.2	1	1	2	5	2	4	6	2	8	1
Median	Top	5.7	6	6	7	8	2	5	9	3	8	3
Median	Top 2	6.2	3	1	7	2	1	14	2	8	8	16
Median	Top 4	6.3	4	3	2	3	4	1	18	4	8	16
IRWM	Top 2	6.7	8	10	10	5	7	5	6	8	5	3
Median	Top 3	7	1	6	2	3	17	3	9	6	7	16
IRWM	Top 3	8	8	10	18	7	7	5	9	8	5	3
IRWM	Top 4	8.1	8	10	10	13	7	5	9	8	8	3
Mean	Top	8.1	19	10	18	8	7	5	3	6	2	3
Mean	Top 2	8.8	8	10	10	8	7	14	18	8	2	3
IRWM	Top 5	8.9	8	10	10	13	7	5	9	8	16	3
Median	Top overall	8.9	17	3	6	8	7	14	3	18	2	11
Mean	Top overall	9.8	8	10	9	13	7	14	9	18	8	2
IRWM	Top overall	10.7	8	10	10	13	7	14	18	8	8	11
Mean	Top 3	11.5	8	19	10	13	7	13	18	8	8	11
Stepwise regression	Analogy – 5NN	11.5	8	6	2	23	4	20	3	8	18	23
Mean	Top 4	14.2	19	20	10	13	18	5	9	21	16	11
Mean	Top 5	15.9	19	20	10	13	18	5	18	21	20	15
Stepwise regression	Analogy – 1NN	16.5	6	20	18	8	18	23	23	8	20	21
IRWM	Top positive	21.4	19	20	18	13	26	22	23	23	25	25
Mean	Top positive	23.4	23	9	28	25	36	33	1	5	38	36
None	Analogy – 1NN	23.5	24	20	24	13	26	23	29	25	26	25
Normalization	Analogy – 1NN	23.5	24	20	24	13	26	23	29	25	26	25
None	Analogy – 5NN	23.5	31	20	24	38	18	27	9	30	22	16
Normalization	Analogy – 5NN	23.5	31	20	24	38	18	27	9	30	22	16
SFS	Analogy – 5NN	25.1	40	20	29	28	18	23	29	24	19	21
Natural logarithm	Analogy – 5NN	28.3	17	10	18	28	39	29	42	33	22	45
Stepwise regression	CART (yes)	31.9	49	30	31	28	18	35	25	28	38	37
Stepwise regression	CART (no)	31.9	49	30	31	28	18	35	25	28	38	37
SFS	Analogy – 1NN	32.2	26	29	40	26	33	21	39	18	51	39
PCA	Analogy – 5NN	33.1	37	34	33	38	38	30	42	27	28	24
None	CART (yes)	34.1	40	35	33	28	26	44	32	33	41	29
None	CART (no)	34.1	40	35	33	28	26	44	32	33	41	29
Normalization	CART (yes)	34.8	40	35	36	28	26	42	32	33	47	29
Normalization	CART (no)	34.8	40	35	36	28	26	42	32	33	47	29
PCA	Analogy – 1NN	35	28	41	29	27	40	44	37	43	33	28
Natural logarithm	CART (yes)	35.4	49	35	36	28	34	44	25	33	41	29
Natural logarithm	CART (no)	35.4	49	35	36	28	34	44	25	33	41	29
Natural logarithm	Analogy – 1NN	37	34	52	54	23	44	14	44	33	32	40
PCA	CART (yes)	39.5	29	32	43	43	45	31	45	55	29	43
PCA	CART (no)	39.5	29	32	43	43	45	31	45	55	29	43
SFS	SLR	41.7	27	47	42	45	41	34	41	32	49	59
SFS	CART (no)	41.8	31	56	23	47	41	67	37	41	34	41
SFS	SWR	44.8	38	48	49	45	45	35	56	43	34	55
SFS	CART (yes)	48	67	43	41	54	36	65	36	47	56	35
EF – 5 bins	CART (yes)	48.1	49	43	47	50	45	54	49	55	41	48
EF – 5 bins	CART (no)	48.1	49	43	47	50	45	54	49	55	41	48
Stepwise regression	SLR	49.1	38	53	49	59	51	63	40	43	50	45
EW – 3 bins	Analogy – 5NN	51.9	48	41	63	57	60	54	51	51	52	42
EF – 5 bins	Analogy – 5NN	52.3	61	46	58	50	50	49	63	62	29	55
Stepwise regression	SWR	53	40	63	51	61	43	63	55	42	52	60

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		Rank	1	2	3	4	5	6	7	8	9	10
EW – 5 bins	Analogy – 5NN	53.8	65	49	45	56	63	58	56	47	52	47
EF – 3 bins	CART (yes)	53.9	59	54	58	41	70	35	45	55	61	61
EF – 3 bins	CART (no)	53.9	59	54	58	41	70	35	45	55	61	61
None	SLR	54.4	34	70	53	63	52	35	59	46	68	64
Normalization	SLR	55.8	34	70	52	66	56	35	59	54	68	64
EF – 3 bins	Analogy – 5NN	57.1	62	56	62	58	55	54	51	61	57	55
EF – 3 bins	Analogy – 1NN	57.2	56	61	55	55	54	59	58	62	61	51
Normalization	SWR	57.2	40	50	55	63	56	59	67	51	65	66
None	SWR	57.2	40	50	55	63	56	59	67	51	65	66
EW – 5 bins	CART (yes)	58.4	68	59	70	47	67	50	65	71	34	53
EW – 5 bins	CART (no)	58.4	68	59	70	47	67	50	65	71	34	53
EF – 5 bins	Analogy – 1NN	59.2	66	58	61	50	52	66	63	64	57	55
EW – 3 bins	CART (yes)	59.5	56	68	66	67	60	50	53	49	57	69
EW – 3 bins	CART (no)	59.5	56	68	66	67	60	50	53	49	57	69
PCA	SWR	59.7	62	65	46	69	56	59	59	65	55	61
EW – 3 bins	Analogy – 1NN	64.9	62	64	64	70	64	68	72	69	68	48
EW – 5 bins	Analogy – 1NN	66.7	70	62	68	60	66	70	69	67	67	68
EW – 3 bins	SWR	68.9	55	74	72	71	69	71	72	68	64	73
SFS	PLSR	69.8	89	67	82	61	65	68	59	66	90	51
Normalization	PLSR	70.2	71	66	65	72	73	73	69	70	71	72
EW – 5 bins	SWR	71.3	71	72	68	74	72	72	69	73	71	71
EW – 5 bins	PLSR	73.5	73	73	73	73	74	74	74	74	73	74
EW – 3 bins	PLSR	75.5	74	75	76	77	75	78	75	75	75	75
None	Neural net	76	74	78	74	75	78	75	76	78	76	76
PCA	Neural net	76	74	78	74	75	78	75	76	78	76	76
Stepwise regression	Neural net	77.4	77	76	77	78	77	79	78	76	78	78
EW – 3 bins	SLR	77.6	78	77	78	79	76	77	81	76	74	80
Normalization	PCR	80.4	79	81	80	82	80	80	79	82	80	81
EW – 5 bins	SLR	80.8	81	82	79	81	81	81	85	81	78	79
SFS	Neural net	82.3	80	80	82	80	83	89	80	80	82	87
EW – 5 bins	PCR	82.6	82	83	81	84	82	82	84	84	81	83
Natural logarithm	SLR	84.2	84	85	85	84	86	83	82	85	86	82
Natural logarithm	SWR	85.7	85	85	84	84	87	87	82	92	86	85
PCA	SLR	86.1	89	85	85	83	83	89	92	83	82	90
EW – 3 bins	PCR	86.9	83	85	93	93	87	84	87	88	84	85
SFS	PCR	87.1	93	84	87	84	83	94	86	91	85	84
Natural logarithm	PLSR	88.3	91	85	89	92	89	87	88	89	86	87
PCA	PLSR	88.6	87	90	89	89	90	84	89	85	90	93
None	PLSR	88.6	87	90	89	89	90	84	89	85	90	93
Natural logarithm	PCR	89.9	85	92	89	88	93	91	91	93	90	87
Stepwise regression	PLSR	91.5	92	93	87	89	92	93	94	89	95	91
EF – 5 bins	PCR	94.9	96	99	94	98	100	91	93	97	90	91
PCA	PCR	96.2	94	94	94	95	96	95	96	94	102	102
None	PCR	96.2	94	94	94	95	96	95	96	94	102	102
EF – 5 bins	PLSR	96.3	96	99	94	94	98	97	95	97	97	96
EF – 5 bins	SLR	96.8	96	99	94	98	93	100	103	100	86	99
EF – 3 bins	PCR	97.4	96	99	94	98	101	99	96	97	99	95
Stepwise regression	PCR	98.9	100	94	94	97	98	97	99	96	104	110
EF – 3 bins	SLR	100	102	99	94	104	93	104	102	101	101	100
EF – 5 bins	SWR	100.3	102	97	103	98	102	101	100	103	97	100
EF – 3 bins	PLSR	100.3	101	97	102	102	103	102	101	103	96	96
EF – 3 bins	SWR	102.4	104	104	104	103	104	102	104	101	100	98
EF – 5 bins	Neural net	104.6	105	105	105	105	105	105	105	105	104	102
Natural logarithm	Neural net	104.6	105	105	105	105	105	105	105	105	104	102
EW – 5 bins	Neural net	104.6	105	105	105	105	105	105	105	105	104	102

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EF – 3 bins	Neural net	105.1	105	105	105	105	105	105	105	105	104	107
Normalization	Neural net	105.1	105	105	105	105	105	105	105	105	104	107
EW – 3 bins	Neural net	105.1	105	105	105	105	105	105	105	105	104	107

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

K Graphical Visualization Of Ranking–Bagging N

A graphical visualization of classifier ranking for the **Wins** and **Wins−Losses** ranking systems for *Bagging N*, is provided in this appendix. Figures K.1 and K.3 display solo classifier performance over all 10 runs for **Wins** and **Wins−Losses** respectively. Figures K.2 and K.4 display classifier performance, solo and ensemble, over all 10 runs for **Wins** and **Wins−Losses** respectively.

All four figures use the same color coding to represent classifier ranking: Classifiers ranked in the top third in a run are coloured gray, classifiers ranked in the middle third are coloured white, and classifiers ranked in the bottom third are coloured black.

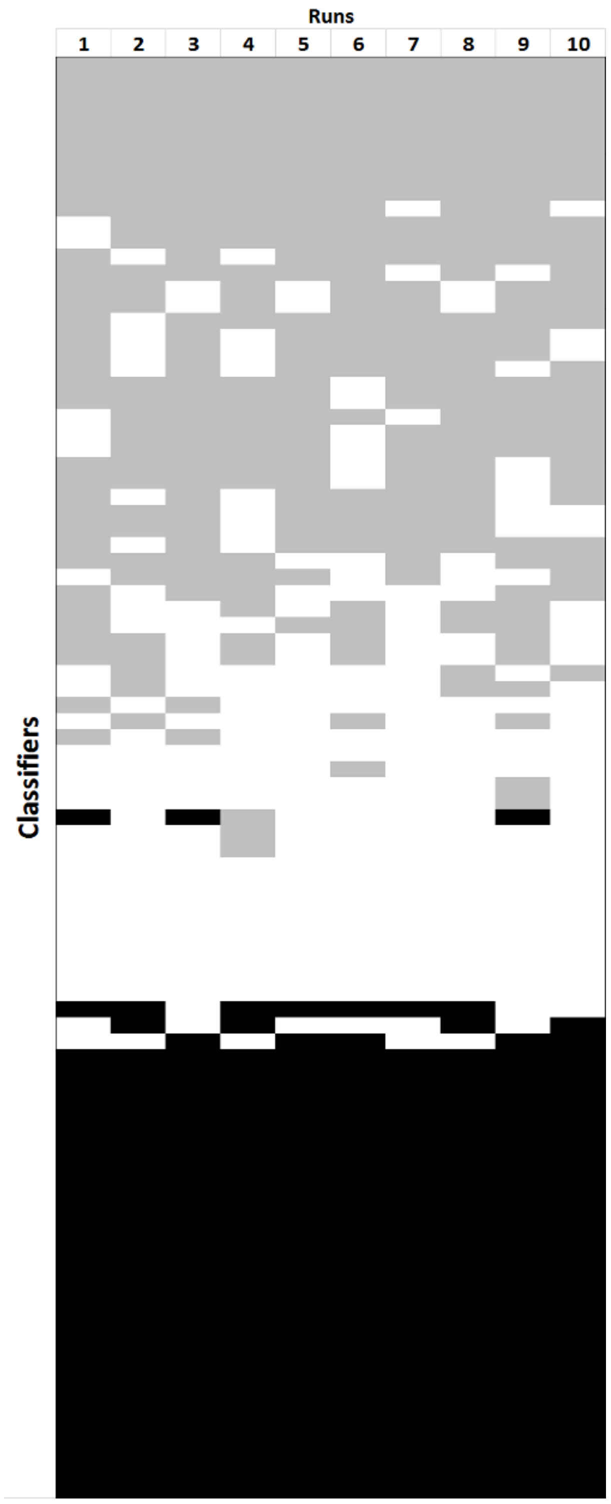


Figure K.1: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins** ranking.

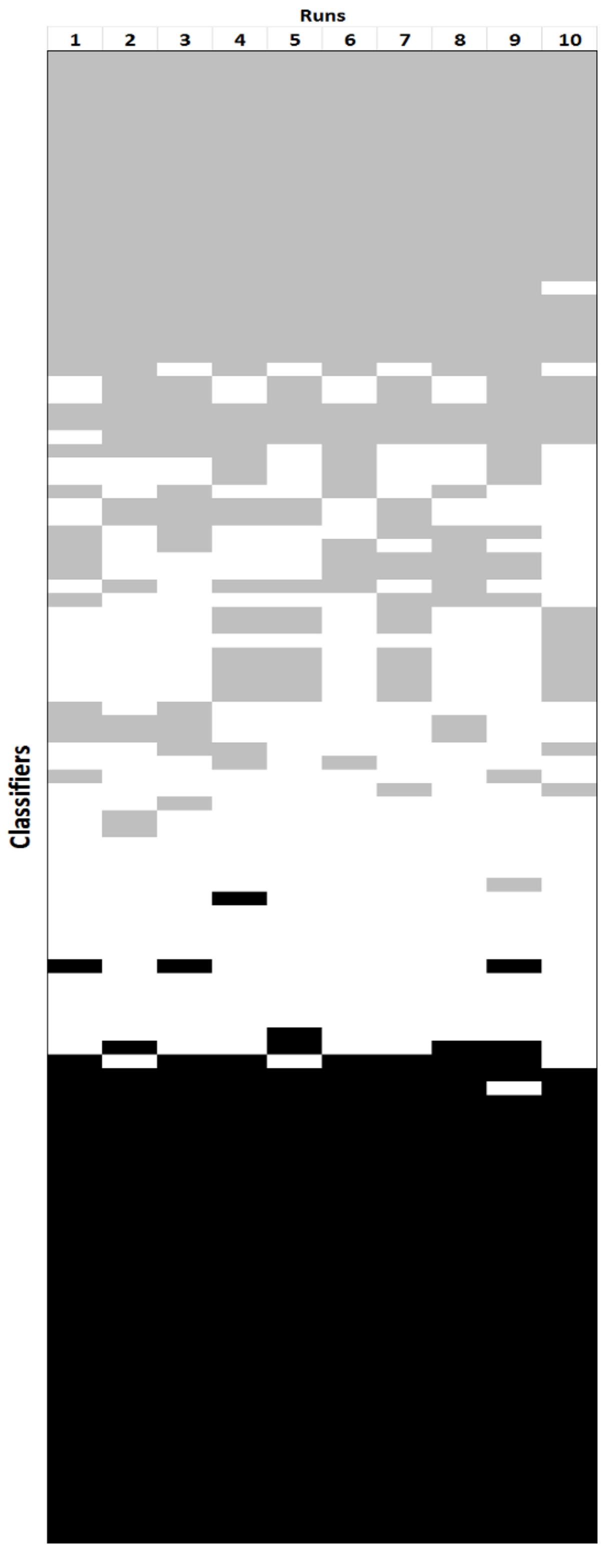


Figure K.2: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins** ranking.

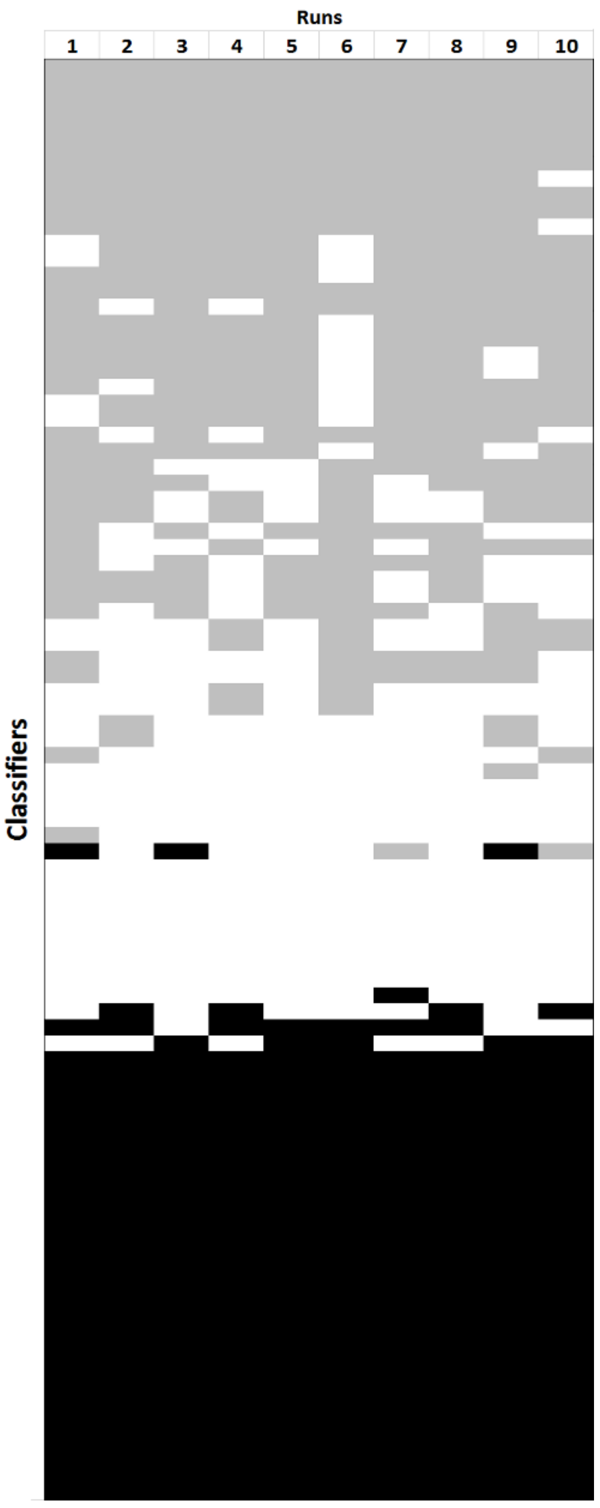


Figure K.3: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins−Losses** ranking.

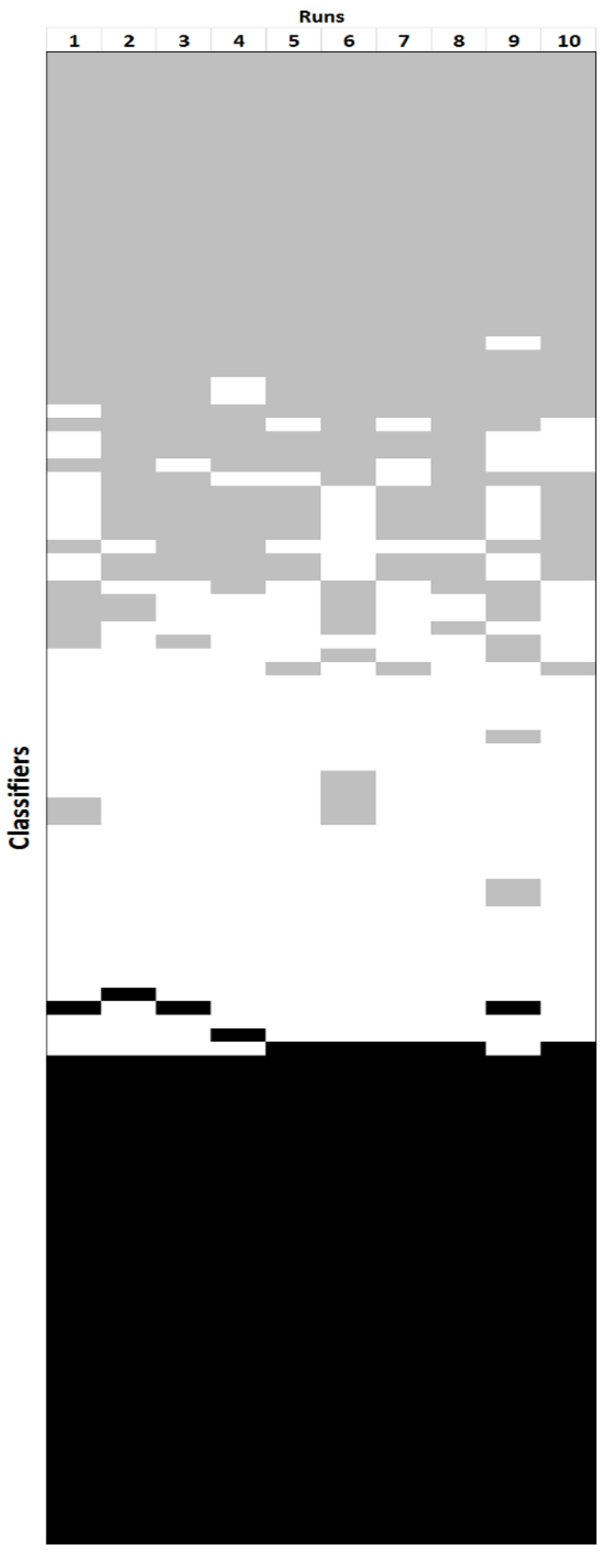


Figure K.4: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins−Losses** ranking.

L Learner Rankings–Bagging $2N_1$

Classifier rankings for all *Bagging $2N_1$* runs are provided in this Appendix. Tables L.1, L.2, and L.3 list the rankings of solo classifiers, in ascending order of average rank, for losses, wins and wins−losses respectively. Tables L.4, L.5, and L.6 list the rankings of solo and ensemble classifiers, in ascending order of average rank, for losses, wins, and wins−losses respectively. Note that ensemble classifiers are highlighted in gray.

Due to space constraints acronyms had to be used for certain pre-processors and learners in these tables. The acronyms used are explained in the very last row of each table.

Table L.1: **Losses** ranking for solo learners, over 10 runs of *Bagging $2N_1$*

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
None	Analogy – 1NN	2.2	1	1	1	1	1	1	7	1	7	1
Normalization	Analogy – 1NN	2.2	1	1	1	1	1	1	7	1	7	1
Stepwise regression	Analogy – 1NN	3.4	1	1	1	10	1	1	7	1	7	4
None	Analogy – 5NN	4.6	9	1	1	4	1	1	1	12	1	15
Normalization	Analogy – 5NN	4.6	9	1	1	4	1	1	1	12	1	15
Stepwise regression	Analogy – 5NN	4.9	14	7	1	1	1	1	1	1	7	15
SFS	Analogy – 1NN	9.4	8	19	9	10	1	1	7	19	16	4
PCA	Analogy – 1NN	9.7	9	7	7	4	9	12	1	16	17	15
SFS	Analogy – 5NN	10	1	7	9	10	1	9	1	24	17	21
PCA	Analogy – 5NN	11	9	7	7	21	9	9	7	12	17	12
Natural logarithm	Analogy – 5NN	13.6	1	12	15	4	9	38	18	26	1	12
Equal width – 3 bins	Analogy – 5NN	14.5	15	20	15	30	12	15	13	9	15	1
SFS	CART (yes)	15.1	19	7	9	20	24	18	1	12	21	20

Continued on next page

Pre-processing Option	Learner	Av. Rank	Run Rank 1	2	3	4	5	6	7	8	9	10
Equal width – 5 bins	Analogy – 5NN	16.5	23	20	28	27	13	9	7	16	1	21
SFS	CART (no)	17	23	1	31	23	24	1	15	20	17	15
None	CART (yes)	17.2	26	12	12	10	37	29	30	1	11	4
None	CART (no)	17.2	26	12	12	10	37	29	30	1	11	4
Normalization	CART (yes)	17.5	26	12	15	10	37	29	30	1	11	4
Normalization	CART (no)	17.5	26	12	15	10	37	29	30	1	11	4
Natural logarithm	CART (yes)	17.8	26	12	15	10	24	27	30	9	21	4
Natural logarithm	CART (no)	17.8	26	12	15	10	24	27	30	9	21	4
Stepwise regression	CART (yes)	18.3	1	20	25	4	32	13	22	31	1	34
Stepwise regression	CART (no)	18.3	1	20	25	4	32	13	22	31	1	34
SFS	Stepwise regression	19.4	17	28	21	22	15	19	13	16	31	12
SFS	Simple linear regression	20.6	16	20	21	19	18	22	19	20	24	27
PCA	CART (yes)	26.2	34	32	29	32	37	15	15	22	25	21
PCA	CART (no)	26.2	34	32	29	32	37	15	15	22	25	21
Stepwise regression	Simple linear regression	26.3	18	25	34	32	16	23	19	24	38	34
Equal frequency – 3 bins	CART (yes)	26.7	19	30	23	28	32	20	24	34	27	30
Equal frequency – 3 bins	CART (no)	26.7	19	30	23	28	32	20	24	34	27	30
Natural logarithm	Analogy – 1NN	26.8	9	28	36	24	37	29	24	26	27	28
Stepwise regression	Stepwise regression	28	23	32	41	32	16	29	21	31	27	28
PCA	Stepwise regression	32.2	34	41	12	30	14	49	27	30	38	47
Equal width – 5 bins	CART (yes)	34.2	34	37	45	32	24	25	40	41	34	30
Equal width – 5 bins	CART (no)	34.2	34	37	45	32	24	25	40	41	34	30
Equal frequency – 5 bins	CART (yes)	34.5	42	25	31	24	48	38	40	36	36	25
Equal frequency – 5 bins	CART (no)	34.5	42	25	31	24	48	38	40	36	36	25
Equal frequency – 3 bins	Analogy – 1NN	35.5	45	35	25	42	30	47	30	36	31	34
None	Simple linear regression	35.8	34	48	39	38	18	29	28	41	44	39
Equal frequency – 3 bins	Analogy – 5NN	36.6	41	35	34	38	32	29	40	41	42	34
Normalization	Stepwise regression	37	26	43	36	45	21	38	40	39	40	42
None	Stepwise regression	37	26	43	36	45	21	38	40	39	40	42
Normalization	Simple linear regression	37.5	34	47	39	45	18	29	39	41	44	39
Equal width – 3 bins	CART (yes)	41	48	43	48	38	37	45	30	26	46	49
Equal width – 3 bins	CART (no)	41	48	43	48	38	37	45	30	26	46	49
Equal frequency – 5 bins	Analogy – 5NN	41.7	44	37	41	44	47	44	40	47	31	42
Equal frequency – 5 bins	Analogy – 1NN	43.8	45	40	43	42	50	48	40	48	43	39
Equal width – 3 bins	Analogy – 1NN	44.6	51	51	47	48	30	52	29	41	49	48
Normalization	PLSR	46.2	52	52	48	50	23	38	50	50	50	49
Equal width – 5 bins	Analogy – 1NN	47.1	47	42	44	48	50	51	49	49	46	45
Equal width – 3 bins	Stepwise regression	50.3	50	49	51	52	46	49	51	51	51	53
Equal width – 5 bins	Stepwise regression	52.3	53	54	51	51	52	53	53	52	52	52
Equal width – 5 bins	PLSR	53.6	56	53	53	53	53	54	52	55	53	54
None	Neural net	54.8	54	56	55	54	54	56	57	53	54	55
PCA	Neural net	54.8	54	56	55	54	54	56	57	53	54	55
Equal width – 3 bins	PLSR	55.4	57	55	54	56	54	55	54	56	56	57
SFS	PLSR	55.6	19	49	68	71	69	23	74	66	71	46
Stepwise regression	Neural net	57.5	58	58	59	57	57	59	55	57	57	58
Normalization	PCR	58	59	59	57	58	58	59	56	58	58	58
Equal width – 5 bins	PCR	61.1	61	61	62	59	60	65	60	62	61	60
Natural logarithm	Simple linear regression	61.1	61	61	60	59	63	61	60	62	61	63
Equal width – 3 bins	PCR	61.7	61	64	63	59	62	65	60	60	61	62
Natural logarithm	Stepwise regression	62.2	60	65	61	59	60	61	63	62	66	65
Equal width – 5 bins	Simple linear regression	62.7	61	73	63	59	66	58	57	66	59	65
Natural logarithm	PLSR	62.7	61	65	63	64	63	61	63	62	60	65
Equal width – 3 bins	Simple linear regression	64.8	66	61	63	67	63	67	70	61	65	65
SFS	Neural net	65.4	73	65	57	71	59	67	70	59	73	60
Natural logarithm	PCR	65.5	66	65	68	64	69	67	66	66	61	63

Continued on next page

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Equal frequency – 5 bins	Simple linear regression	66.2	66	65	67	64	66	67	66	66	66	69
PCA	Simple linear regression	66.7	66	60	73	67	66	64	65	66	71	69
Equal frequency – 5 bins	PCR	67.6	66	65	68	67	69	72	66	66	66	71
Equal frequency – 3 bins	PCR	68.7	66	65	72	67	69	72	66	73	66	71
SFS	PCR	72	75	73	75	71	74	67	70	78	66	71
Equal frequency – 5 bins	PLSR	72.8	66	72	73	75	75	72	74	74	73	74
Equal frequency – 3 bins	Simple linear regression	74.5	81	82	68	71	69	81	73	66	79	75
PCA	PLSR	75.7	75	76	75	78	75	77	77	74	75	75
None	PLSR	75.7	75	76	75	78	75	77	77	74	75	75
Stepwise regression	PLSR	76.6	78	73	81	75	79	72	76	82	75	75
Equal frequency – 5 bins	Stepwise regression	76.8	73	76	78	77	75	77	80	74	75	83
PCA	PCR	79.6	79	76	78	81	80	82	83	79	79	79
None	PCR	79.6	79	76	78	81	80	82	83	79	79	79
Equal frequency – 3 bins	PLSR	79.7	82	76	82	78	82	76	79	79	82	81
Stepwise regression	PCR	82.6	82	83	83	83	83	84	80	85	82	81
Equal frequency – 3 bins	Stepwise regression	83	84	84	84	84	83	80	80	83	84	84
Natural logarithm	Neural net	84.9	85	85	85	85	85	85	85	84	85	85
Equal frequency – 5 bins	Neural net	85	85	85	85	85	85	85	85	85	85	85
Equal width – 5 bins	Neural net	85	85	85	85	85	85	85	85	85	85	85
Equal frequency – 3 bins	Neural net	85.3	85	85	85	88	85	85	85	85	85	85
Normalization	Neural net	85.3	85	85	85	88	85	85	85	85	85	85
Equal width – 3 bins	Neural net	85.3	85	85	85	88	85	85	85	85	85	85

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

Table L.2: **Wins** ranking for solo learners, over 10 runs of *Bagging $2N_1$*.

Pre-processing Option	Learner	Av. Rank	Run Rank 1	2	3	4	5	6	7	8	9	10
SFS	Simple linear regression	1.9	3	2	1	1	4	3	1	1	1	2
Stepwise regression	Analogy – 1NN	4.9	4	4	3	11	2	1	8	3	8	5
SFS	Stepwise regression	5.7	4	1	2	2	9	4	24	6	4	1
Stepwise regression	Analogy – 5NN	5.8	16	3	3	3	1	5	6	1	5	15
None	Analogy – 1NN	8.7	7	7	5	3	22	5	14	6	12	6
Normalization	Analogy – 1NN	8.7	7	7	5	3	22	5	14	6	12	6
None	Analogy – 5NN	11.2	10	4	8	6	22	10	8	19	6	19
Normalization	Analogy – 5NN	11.2	10	4	8	6	22	10	8	19	6	19
Equal width – 5 bins	CART (yes)	13.4	10	11	25	21	14	12	18	19	2	2
Equal width – 5 bins	CART (no)	13.4	10	11	25	21	14	12	18	19	2	2
Natural logarithm	Analogy – 5NN	15.6	9	9	7	6	3	43	20	36	8	15
None	Simple linear regression	16.1	10	35	17	20	9	20	3	19	20	8
Stepwise regression	Simple linear regression	16.2	16	28	19	21	6	14	5	17	12	24
Equal width – 3 bins	Analogy – 5NN	16.8	29	28	16	27	6	1	2	17	19	23
SFS	Analogy – 5NN	18.1	16	10	17	11	9	18	3	36	32	29
SFS	Analogy – 1NN	19	22	35	23	13	19	5	8	28	22	15
Stepwise regression	Stepwise regression	20.4	22	26	25	27	9	20	6	14	20	35
Normalization	Simple linear regression	20.9	16	35	19	29	14	20	12	24	22	18
PCA	Stepwise regression	21.2	10	42	23	24	6	34	12	9	28	24
None	CART (yes)	21.8	34	14	8	13	42	38	40	9	12	8
None	CART (no)	21.8	34	14	8	13	42	38	40	9	12	8
Normalization	CART (yes)	21.8	34	14	8	13	42	38	40	9	12	8
Normalization	CART (no)	21.8	34	14	8	13	42	38	40	9	12	8
Natural logarithm	CART (yes)	22.8	34	14	8	13	40	35	40	14	22	8
Natural logarithm	CART (no)	22.8	34	14	8	13	40	35	40	14	22	8
Normalization	Stepwise regression	23	16	28	19	31	19	24	20	24	28	21
None	Stepwise regression	23	16	28	19	31	19	24	20	24	28	21
Stepwise regression	CART (yes)	23.9	1	24	34	6	30	14	35	42	8	45
Stepwise regression	CART (no)	23.9	1	24	34	6	30	14	35	42	8	45
Equal width – 3 bins	CART (yes)	24.3	30	35	32	24	14	24	14	3	32	35
Equal width – 3 bins	CART (no)	24.3	30	35	32	24	14	24	14	3	32	35
Equal width – 5 bins	Analogy – 5NN	25.2	27	27	30	31	9	14	24	30	22	38
PCA	Analogy – 5NN	26	25	20	30	31	27	18	23	28	32	26
Equal width – 3 bins	Stepwise regression	28.7	22	22	38	35	4	32	24	32	40	38
PCA	Analogy – 1NN	29.1	26	22	28	29	29	31	28	30	28	40
SFS	CART (yes)	30.1	41	11	28	43	34	33	28	24	27	32
SFS	CART (no)	30.7	42	20	34	35	34	5	34	39	32	32
Natural logarithm	Analogy – 1NN	31	6	28	34	35	32	35	35	35	38	32
Normalization	PLSR	33.2	33	40	38	41	22	20	28	33	37	40
PCA	CART (yes)	37.1	43	44	45	46	36	24	28	39	40	26
PCA	CART (no)	37.1	43	44	45	46	36	24	28	39	40	26
Equal width – 5 bins	Analogy – 1NN	39.9	30	42	45	43	36	47	38	33	38	47
Equal frequency – 5 bins	CART (yes)	40.3	43	28	42	35	46	43	48	45	44	29
Equal frequency – 5 bins	CART (no)	40.3	43	28	42	35	46	43	48	45	44	29
Equal width – 3 bins	Analogy – 1NN	41.5	50	50	44	43	27	50	24	36	44	47
Equal width – 5 bins	Stepwise regression	41.8	34	52	40	42	32	50	39	42	47	40
Equal width – 5 bins	PLSR	42.6	53	44	45	40	36	38	28	51	47	44
Equal frequency – 5 bins	Analogy – 5NN	45.8	43	47	40	52	48	46	48	45	40	49
Equal frequency – 3 bins	CART (yes)	47.8	43	47	51	48	49	47	46	48	49	50
Equal frequency – 3 bins	CART (no)	47.8	43	47	51	48	49	47	46	48	49	50
Equal frequency – 3 bins	Analogy – 5NN	50.4	50	52	49	48	49	53	51	51	49	52
Equal frequency – 5 bins	Analogy – 1NN	51.1	53	52	51	48	49	53	51	50	52	52
SFS	PLSR	51.4	28	40	63	61	64	24	66	62	63	43

Continued on next page

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Equal width – 3 bins	PLSR	51.8	52	51	49	52	49	52	51	54	54	54
Equal frequency – 3 bins	Analogy – 1NN	52.3	53	55	51	52	49	55	51	51	52	54
Stepwise regression	Neural net	55.3	56	56	55	55	55	56	55	54	55	56
None	Neural net	56.6	57	57	57	56	56	57	57	56	56	57
PCA	Neural net	56.6	57	57	57	56	56	57	57	56	56	57
Normalization	PCR	58.9	59	60	59	59	59	59	56	60	58	60
Equal width – 3 bins	Simple linear regression	59.6	60	59	60	58	61	59	59	59	59	62
SFS	Neural net	60.6	63	63	55	63	58	66	62	56	63	57
Equal width – 5 bins	PCR	62	69	62	61	61	60	64	60	61	61	61
Equal width – 5 bins	Simple linear regression	62.2	61	67	63	60	64	61	60	63	60	63
PCA	Simple linear regression	64.4	62	61	76	63	63	65	62	64	63	65
SFS	PCR	65	67	69	65	66	62	69	62	65	62	63
PCA	PLSR	65.6	63	63	67	67	66	66	68	65	66	65
None	PLSR	65.6	63	63	67	67	66	66	68	65	66	65
Natural logarithm	Simple linear regression	66.7	71	67	62	63	69	63	66	65	71	70
Stepwise regression	PLSR	67.6	67	66	67	67	66	69	68	72	66	68
Natural logarithm	Stepwise regression	68.1	63	70	65	70	69	62	68	69	73	72
Natural logarithm	PLSR	70.1	72	70	71	72	69	69	68	69	69	72
Equal width – 3 bins	PCR	71.8	69	78	70	82	69	72	62	78	70	68
Natural logarithm	PCR	74.1	73	79	81	72	74	73	73	75	71	70
Equal frequency – 5 bins	Simple linear regression	74.1	73	79	71	71	73	73	85	69	73	74
PCA	PCR	74.1	73	72	73	72	74	73	85	72	73	74
None	PCR	74.1	73	72	73	72	74	73	85	72	73	74
Stepwise regression	PCR	75.6	73	72	73	72	86	73	73	87	73	74
Equal frequency – 3 bins	PLSR	77.3	78	75	78	77	77	78	75	76	80	79
Equal frequency – 5 bins	PLSR	78	80	77	76	78	77	81	76	77	79	79
Equal frequency – 5 bins	Stepwise regression	78.1	78	75	78	81	77	78	76	78	81	79
Equal frequency – 3 bins	Stepwise regression	79.9	80	79	78	78	86	78	76	80	82	82
Equal frequency – 5 bins	PCR	80.4	80	79	81	78	86	81	85	87	73	74
Natural logarithm	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Equal frequency – 5 bins	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Equal width – 5 bins	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Equal frequency – 3 bins	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Normalization	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Equal width – 3 bins	Neural net	80.8	80	79	81	83	80	81	79	80	82	83
Equal frequency – 3 bins	PCR	82.7	80	79	81	83	86	81	85	87	82	83
Equal frequency – 3 bins	Simple linear regression	82.7	80	79	81	83	86	81	85	87	82	83

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table L.3: **Wins−Losses** ranking for solo learners, over 10 runs of *Bagging $2N_1$*.

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Stepwise regression	Analogy – 1NN	3.5	3	1	1	10	2	1	6	2	7	2
Stepwise regression	Analogy – 5NN	3.8	14	1	1	2	1	2	2	1	1	13
None	Analogy – 1NN	4.3	4	5	3	2	6	2	8	3	8	2
Normalization	Analogy – 1NN	4.3	4	5	3	2	6	2	8	3	8	2
None	Analogy – 5NN	6.9	10	1	6	5	6	7	3	14	1	16
Normalization	Analogy – 5NN	6.9	10	1	6	5	6	7	3	14	1	16
SFS	Simple linear regression	9.9	8	7	5	1	17	16	8	13	10	14
SFS	Stepwise regression	11.9	12	19	6	10	14	13	17	9	18	1
SFS	Analogy – 1NN	12.7	12	23	18	13	5	2	6	20	17	11
SFS	Analogy – 5NN	13.5	8	7	12	10	4	11	1	33	25	24
Natural logarithm	Analogy – 5NN	14.3	6	7	9	5	3	43	17	37	4	12
Equal width – 3 bins	Analogy – 5NN	15.1	18	25	17	32	6	9	3	12	15	14
PCA	Analogy – 5NN	16.8	15	12	22	21	13	11	12	17	25	20
PCA	Analogy – 1NN	18.7	16	19	19	20	14	17	13	20	23	26
None	CART (yes)	19.9	30	12	9	13	39	38	38	5	10	5
None	CART (no)	19.9	30	12	9	13	39	38	38	5	10	5
Normalization	CART (yes)	20.2	30	12	12	13	39	38	38	5	10	5
Normalization	CART (no)	20.2	30	12	12	13	39	38	38	5	10	5
Natural logarithm	CART (yes)	20.6	30	12	12	13	34	35	38	9	18	5
Natural logarithm	CART (no)	20.6	30	12	12	13	34	35	38	9	18	5
Stepwise regression	CART (yes)	20.6	1	21	29	5	30	13	27	38	4	38
Stepwise regression	CART (no)	20.6	1	21	29	5	30	13	27	38	4	38
Equal width – 5 bins	Analogy – 5NN	20.7	28	23	27	32	11	10	13	20	15	28
SFS	CART (yes)	21.3	29	10	21	23	30	24	13	16	23	24
Stepwise regression	Simple linear regression	22.1	17	26	23	24	14	20	11	23	29	34
SFS	CART (no)	23.5	38	11	32	22	30	2	23	31	25	21
Equal width – 5 bins	CART (yes)	24.5	19	30	40	24	23	20	27	26	18	18
Equal width – 5 bins	CART (no)	24.5	19	30	40	24	23	20	27	26	18	18
Stepwise regression	Stepwise regression	25.5	19	30	31	34	17	25	13	25	28	33
None	Simple linear regression	25.6	19	43	24	24	19	25	17	26	33	26
PCA	Stepwise regression	27.3	19	43	19	24	11	47	20	23	30	37
Normalization	Stepwise regression	29.1	19	37	24	38	21	29	31	26	31	35
None	Stepwise regression	29.1	19	37	24	38	21	29	31	26	31	35
Natural logarithm	Analogy – 1NN	29.2	6	29	33	24	34	37	31	33	33	32
Normalization	Simple linear regression	30	26	42	27	37	20	25	24	31	37	31
PCA	CART (yes)	31.3	41	33	36	40	37	18	21	33	33	21
PCA	CART (no)	31.3	41	33	36	40	37	18	21	33	33	21
Equal width – 3 bins	CART (yes)	35.8	46	39	47	34	27	31	24	18	44	48
Equal width – 3 bins	CART (no)	35.8	46	39	47	34	27	31	24	18	44	48
Equal frequency – 5 bins	CART (yes)	37.4	43	26	34	24	48	43	47	42	39	28
Equal frequency – 5 bins	CART (no)	37.4	43	26	34	24	48	43	47	42	39	28
Equal frequency – 3 bins	CART (yes)	37.6	36	33	36	40	44	33	35	42	39	38
Equal frequency – 3 bins	CART (no)	37.6	36	33	36	40	44	33	35	42	39	38
Normalization	PLSR	43.2	51	51	49	47	23	28	37	46	49	51
Equal width – 3 bins	Stepwise regression	44.3	40	39	51	51	26	38	45	49	51	53
Equal width – 5 bins	Analogy – 1NN	44.6	39	45	45	47	47	51	38	41	47	46
Equal width – 3 bins	Analogy – 1NN	45.2	53	52	50	47	29	52	31	40	50	48
Equal frequency – 3 bins	Analogy – 1NN	45.4	49	48	40	46	43	49	45	47	43	44
Equal frequency – 5 bins	Analogy – 5NN	45.4	46	46	40	50	48	46	47	47	38	46
Equal frequency – 3 bins	Analogy – 5NN	45.7	43	46	44	44	44	48	50	50	46	42
Equal frequency – 5 bins	Analogy – 1NN	48.1	49	49	45	45	51	49	50	51	48	44
Equal width – 5 bins	Stepwise regression	52.4	52	54	52	52	52	53	53	52	52	52

Continued on next page

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
SFS	PLSR	53.1	27	49	65	63	66	23	69	62	65	42
Equal width – 5 bins	PLSR	53.2	54	53	53	53	53	54	52	53	53	54
Equal width – 3 bins	PLSR	54.5	54	55	54	54	54	55	54	56	54	55
None	Neural net	55.6	56	56	55	55	55	57	57	54	55	56
PCA	Neural net	55.6	56	56	55	55	55	57	57	54	55	56
Stepwise regression	Neural net	56.7	58	56	58	57	57	56	55	57	55	58
Normalization	PCR	58.5	59	59	59	58	58	59	56	59	58	60
Equal width – 3 bins	Simple linear regression	60.2	60	60	60	59	61	61	60	60	59	62
Equal width – 5 bins	PCR	61.5	64	62	61	61	60	64	60	61	61	61
Equal width – 5 bins	Simple linear regression	61.8	61	66	63	60	63	60	59	63	60	63
SFS	Neural net	62.9	66	63	57	65	58	67	67	58	69	59
Natural logarithm	Simple linear regression	64.1	66	63	61	62	68	63	63	65	65	65
PCA	Simple linear regression	64.4	63	61	73	63	62	64	64	64	65	65
Natural logarithm	Stepwise regression	65.3	61	66	63	65	63	62	65	66	73	69
Natural logarithm	PLSR	66.5	68	66	67	69	68	64	65	66	63	69
Equal width – 3 bins	PCR	67.4	64	73	66	74	66	69	62	73	64	63
SFS	PCR	67.4	71	71	69	67	63	68	67	71	62	65
PCA	PLSR	69.9	68	66	70	72	70	71	72	69	70	71
None	PLSR	69.9	68	66	70	72	70	71	72	69	70	71
Stepwise regression	PLSR	71	72	65	72	69	73	69	71	76	70	73
Natural logarithm	PCR	71.2	73	78	77	69	73	71	70	71	65	65
Equal frequency – 5 bins	Simple linear regression	71.6	73	78	68	68	70	71	74	68	73	73
Equal frequency – 5 bins	PLSR	75.2	78	72	73	76	75	76	74	76	76	76
Equal frequency – 5 bins	PCR	76	78	78	77	74	79	76	74	76	73	75
PCA	PCR	76.7	76	73	75	78	75	79	83	74	78	76
None	PCR	76.7	76	73	75	78	75	79	83	74	78	76
Equal frequency – 3 bins	PCR	77.9	78	78	80	77	79	76	74	80	77	80
Equal frequency – 5 bins	Stepwise regression	79	75	76	80	82	75	79	81	80	80	82
Equal frequency – 3 bins	PLSR	80	82	76	83	78	82	75	79	82	82	81
Equal frequency – 3 bins	Simple linear regression	80.4	83	83	77	78	79	84	79	76	83	82
Stepwise regression	PCR	81.1	78	78	80	83	83	82	78	90	80	79
Equal frequency – 3 bins	Stepwise regression	83.3	84	84	84	84	83	82	81	83	84	84
Natural logarithm	Neural net	84.7	85	85	85	85	85	85	83	84	85	85
Equal frequency – 5 bins	Neural net	84.8	85	85	85	85	85	85	83	85	85	85
Equal width – 5 bins	Neural net	84.8	85	85	85	85	85	85	83	85	85	85
Equal frequency – 3 bins	Neural net	85.1	85	85	85	88	85	85	83	85	85	85
Normalization	Neural net	85.1	85	85	85	88	85	85	83	85	85	85
Equal width – 3 bins	Neural net	85.1	85	85	85	88	85	85	83	85	85	85

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table L.4: **Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging $2N_1$*. Ensembles are highlighted gray.

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
IRWM	Top 3	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 2	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 2	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 4	1	1	1	1	1	1	1	1	1	1	1
Median	Top	2.1	12	1	1	1	1	1	1	1	1	1
Mean	Top overall	2.1	12	1	1	1	1	1	1	1	1	1
Mean	Top 5	2.8	1	19	1	1	1	1	1	1	1	1
Median	Top positive	3	1	1	1	1	21	1	1	1	1	1
IRWM	Top overall	3.5	12	1	1	15	1	1	1	1	1	1
Mean	Top	4.8	1	1	1	15	1	25	1	1	1	1
None	Analogy – 1NN	6	12	1	1	1	1	1	17	1	24	1
Normalization	Analogy – 1NN	6	12	1	1	1	1	1	17	1	24	1
Median	Top overall	7.4	12	1	1	15	1	1	17	1	24	1
Median	Top 5	8.4	24	1	1	15	21	1	1	1	1	18
Median	Top 3	9.2	12	1	21	15	21	1	1	1	1	18
IRWM	Top positive	10.3	24	25	21	1	1	1	27	1	1	1
Median	Top 4	11.2	12	1	1	15	21	25	17	1	1	18
Median	Top 2	14.8	1	19	21	22	21	1	17	27	1	18
None	Analogy – 5NN	15	24	19	1	22	1	1	17	31	1	33
Normalization	Analogy – 5NN	15	24	19	1	22	1	1	17	31	1	33
Stepwise regression	Analogy – 1NN	21.7	12	19	21	36	21	25	25	1	27	30
Stepwise regression	Analogy – 5NN	22.6	31	25	21	15	29	1	17	27	27	33
SFS	Analogy – 1NN	23.5	24	46	26	22	1	1	25	36	36	18
SFS	Analogy – 5NN	23.6	24	19	26	22	1	30	1	39	37	37
Natural logarithm	Analogy – 5NN	27.3	12	25	36	22	29	46	32	40	1	30
None	CART (yes)	28.2	35	29	30	22	38	41	41	1	27	18
None	CART (no)	28.2	35	29	30	22	38	41	41	1	27	18
Normalization	CART (yes)	28.4	35	29	32	22	38	41	41	1	27	18
Normalization	CART (no)	28.4	35	29	32	22	38	41	41	1	27	18
Stepwise regression	CART (yes)	29.9	12	36	37	22	35	30	35	42	1	49
Stepwise regression	CART (no)	29.9	12	36	37	22	35	30	35	42	1	49
Natural logarithm	CART (yes)	30.8	35	29	32	22	32	39	41	27	33	18
Natural logarithm	CART (no)	30.8	35	29	32	22	32	39	41	27	33	18
PCA	Analogy – 5NN	31.5	31	29	26	39	29	28	33	31	37	32
Mean	Top positive	31.8	34	42	46	38	21	28	47	1	43	18
PCA	Analogy – 1NN	32.7	31	38	26	37	21	33	28	35	37	41
PCA	CART (yes)	39.4	41	44	42	47	38	33	28	37	47	37
PCA	CART (no)	39.4	41	44	42	47	38	33	28	37	47	37
Natural logarithm	Analogy – 1NN	39.8	30	39	46	42	38	41	37	40	40	45
SFS	CART (yes)	41.3	51	28	50	40	58	36	31	49	33	37
EF – 5 bins	CART (yes)	43.7	41	39	42	42	47	46	47	46	45	42
EF – 5 bins	CART (no)	43.7	41	39	42	42	47	46	47	46	45	42
SFS	CART (no)	44.3	50	43	57	40	58	1	54	55	41	44
EF – 3 bins	CART (yes)	46.2	51	48	40	45	63	37	39	44	49	46
EF – 3 bins	CART (no)	46.2	51	48	40	45	63	37	39	44	49	46
EF – 3 bins	Analogy – 1NN	47	46	51	39	50	34	52	52	46	51	49
EF – 3 bins	Analogy – 5NN	47.5	46	51	46	47	35	51	47	49	54	49
EW – 3 bins	Analogy – 5NN	48.1	49	54	53	56	45	54	55	31	51	33
EF – 5 bins	Analogy – 5NN	48.6	45	47	49	52	45	49	52	52	41	54

Continued on next page

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EW – 5 bins	Analogy – 5NN	48.9	58	53	56	55	47	49	33	49	43	46
EF – 5 bins	Analogy – 1NN	50.1	46	48	50	50	50	53	47	53	51	53
SFS	SWR	52.9	55	57	54	54	52	55	37	54	57	54
SFS	SLR	54.7	51	55	54	52	55	56	56	56	55	57
Stepwise regression	SLR	57.3	55	56	58	59	53	58	56	57	61	60
Stepwise regression	SWR	58.4	59	58	63	58	54	61	58	61	56	56
PCA	SWR	59.6	62	61	52	56	50	69	59	60	60	67
EW – 5 bins	CART (yes)	61.4	62	59	65	59	65	59	65	64	58	58
EW – 5 bins	CART (no)	61.4	62	59	65	59	65	59	65	64	58	58
None	SLR	61.8	62	68	61	62	55	61	60	64	64	61
Normalization	SWR	62	60	63	59	65	58	64	65	62	61	63
None	SWR	62	60	63	59	65	58	64	65	62	61	63
Normalization	SLR	62.7	62	67	62	67	55	61	64	64	64	61
EW – 3 bins	CART (yes)	65.2	68	63	68	62	68	67	62	58	67	69
EW – 3 bins	CART (no)	65.2	68	63	68	62	68	67	62	58	67	69
EW – 5 bins	Analogy – 1NN	67	67	62	64	68	71	71	69	69	66	63
EW – 3 bins	Analogy – 1NN	67.6	70	71	67	68	67	72	61	64	69	67
Normalization	PLSR	68.7	72	72	68	70	62	64	70	70	70	69
EW – 3 bins	SWR	70.7	70	69	71	72	70	69	71	71	71	73
EW – 5 bins	SWR	72.3	73	74	71	71	72	73	73	72	72	72
EW – 5 bins	PLSR	73.6	76	73	73	73	73	74	72	75	73	74
None	Neural net	74.8	74	76	75	74	74	76	77	73	74	75
PCA	Neural net	74.8	74	76	75	74	74	76	77	73	74	75
EW – 3 bins	PLSR	75.4	77	75	74	76	74	75	74	76	76	77
Stepwise regression	Neural net	77.5	78	78	79	77	77	79	75	77	77	78
Normalization	PCR	78	79	79	77	78	78	79	76	78	78	78
SFS	PLSR	78.6	55	69	88	91	89	57	94	86	91	66
EW – 5 bins	PCR	81.1	81	81	82	79	80	85	80	82	81	80
Natural logarithm	SLR	81.1	81	81	80	79	83	81	80	82	81	83
EW – 3 bins	PCR	81.7	81	84	83	79	82	85	80	80	81	82
Natural logarithm	SWR	82.2	80	85	81	79	80	81	83	82	86	85
EW – 5 bins	SLR	82.7	81	93	83	79	86	78	77	86	79	85
Natural logarithm	PLSR	82.7	81	85	83	84	83	81	83	82	80	85
EW – 3 bins	SLR	84.8	86	81	83	87	83	87	90	81	85	85
SFS	Neural net	85.4	93	85	77	91	79	87	90	79	93	80
Natural logarithm	PCR	85.5	86	85	88	84	89	87	86	86	81	83
EF – 5 bins	SLR	86.2	86	85	87	84	86	87	86	86	86	89
PCA	SLR	86.7	86	80	93	87	86	84	85	86	91	89
EF – 5 bins	PCR	87.6	86	85	88	87	89	92	86	86	86	91
EF – 3 bins	PCR	88.7	86	85	92	87	89	92	86	93	86	91
SFS	PCR	92	95	93	95	91	94	87	90	98	86	91
EF – 5 bins	PLSR	92.8	86	92	93	95	95	92	94	94	93	94
EF – 3 bins	SLR	94.5	101	102	88	91	89	101	93	86	99	95
PCA	PLSR	95.7	95	96	95	98	95	97	97	94	95	95
None	PLSR	95.7	95	96	95	98	95	97	97	94	95	95
Stepwise regression	PLSR	96.6	98	93	101	95	99	92	96	102	95	95
EF – 5 bins	SWR	96.8	93	96	98	97	95	97	100	94	95	103
PCA	PCR	99.6	99	96	98	101	100	102	103	99	99	99
None	PCR	99.6	99	96	98	101	100	102	103	99	99	99
EF – 3 bins	PLSR	99.7	102	96	102	98	102	96	99	99	102	101
Stepwise regression	PCR	102.6	102	103	103	103	103	104	100	105	102	101
EF – 3 bins	SWR	103	104	104	104	104	103	100	100	103	104	104
Natural logarithm	Neural net	104.9	105	105	105	105	105	105	105	104	105	105
EF – 5 bins	Neural net	105	105	105	105	105	105	105	105	105	105	105
EW – 5 bins	Neural net	105	105	105	105	105	105	105	105	105	105	105

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EF – 3 bins	Neural net	105.3	105	105	105	108	105	105	105	105	105	105
Normalization	Neural net	105.3	105	105	105	108	105	105	105	105	105	105
EW – 3 bins	Neural net	105.3	105	105	105	108	105	105	105	105	105	105

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression **SLR**–Simple linear regression **SWR**–Stepwise regression
EF–Equal frequency **EW**–Equal width

Table L.5: **Wins** ranking for ensemble and solo classifiers, over 10 runs of *Bagging 2*N_1. Ensembles are highlighted gray.

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top positive	3.1	4	4	5	5	1	1	3	3	3	2
Median	Top 5	3.5	5	1	1	14	3	1	1	5	1	3
Median	Top 4	5.3	12	3	2	14	3	3	5	5	3	3
Median	Top 3	6.3	12	2	18	7	3	9	1	5	1	5
IRWM	Top 3	7.8	5	9	5	7	3	9	14	5	8	13
IRWM	Top 4	7.8	5	9	2	7	3	9	14	5	13	11
Mean	Top 2	8.4	5	17	5	7	3	7	14	5	8	13
Median	Top	8.7	2	7	22	14	3	9	9	5	3	13
SFS	SLR	8.7	19	15	4	1	21	7	7	1	7	5
IRWM	Top 5	9	5	9	5	7	3	9	14	5	13	20
IRWM	Top 2	9.6	5	9	5	7	3	9	25	5	8	20
IRWM	Top overall	9.6	19	5	5	14	3	9	7	3	22	9
Median	Top overall	9.9	19	5	5	6	3	9	6	18	19	9
Median	Top 2	9.9	1	8	18	3	3	4	3	33	6	20
Mean	Top	10.1	5	9	5	14	3	9	25	5	13	13
Mean	Top 3	10.5	12	17	5	7	3	9	14	5	13	20
Mean	Top 5	11.8	12	17	5	14	3	9	14	5	19	20
Mean	Top 4	12.3	12	17	5	14	3	9	25	5	13	20
SFS	SWR	15.6	22	9	5	4	27	9	44	22	13	1
Stepwise regression	Analogy – 1NN	17.4	22	17	18	31	3	4	25	18	25	11
Stepwise regression	Analogy – 5NN	17.7	36	16	18	14	1	23	14	2	19	34
Mean	Top positive	18.6	2	17	25	1	21	9	14	25	38	34
Mean	Top overall	19	27	17	5	14	27	23	14	25	25	13
IRWM	Top positive	23.2	27	26	25	14	33	23	14	25	25	20
None	Analogy – 1NN	25.3	25	26	23	14	42	23	34	22	31	13
Normalization	Analogy – 1NN	25.3	25	26	23	14	42	23	34	22	31	13
None	Analogy – 5NN	29.8	30	17	28	26	42	30	25	39	22	39
Normalization	Analogy – 5NN	29.8	30	17	28	26	42	30	25	39	22	39
EW – 5 bins	CART (yes)	30.2	30	31	45	41	33	32	38	39	8	5
EW – 5 bins	CART (no)	30.2	30	31	45	41	33	32	38	39	8	5
EW – 3 bins	Analogy – 5NN	33.5	49	48	36	47	24	4	9	37	38	43
None	SLR	34	30	55	37	40	27	40	12	39	40	20
Natural logarithm	Analogy – 5NN	34.5	27	29	25	26	20	63	40	56	25	34
Stepwise regression	SLR	34.7	36	48	39	41	24	34	13	37	31	44
SFS	Analogy – 5NN	36.5	36	30	37	31	27	38	9	56	52	49
SFS	Analogy – 1NN	38.4	42	55	43	33	39	23	25	48	42	34
Stepwise regression	SWR	38.9	42	46	45	47	27	40	14	33	40	55
None	CART (yes)	40.8	54	34	28	33	62	58	60	28	31	20
None	CART (no)	40.8	54	34	28	33	62	58	60	28	31	20
Normalization	CART (yes)	40.8	54	34	28	33	62	58	60	28	31	20
Normalization	CART (no)	40.8	54	34	28	33	62	58	60	28	31	20
Normalization	SLR	40.8	36	55	39	49	33	40	32	44	42	38
PCA	SWR	40.9	30	62	43	44	24	54	32	28	48	44
Natural logarithm	CART (yes)	41.9	54	34	28	33	60	55	60	33	42	20
Natural logarithm	CART (no)	41.9	54	34	28	33	60	55	60	33	42	20
Stepwise regression	CART (yes)	42.7	12	44	54	26	50	34	55	62	25	65
Stepwise regression	CART (no)	42.7	12	44	54	26	50	34	55	62	25	65
Normalization	SWR	43	36	48	39	51	39	44	40	44	48	41
None	SWR	43	36	48	39	51	39	44	40	44	48	41
EW – 3 bins	CART (yes)	43.7	50	55	52	44	33	44	34	18	52	55
EW – 3 bins	CART (no)	43.7	50	55	52	44	33	44	34	18	52	55
EW – 5 bins	Analogy – 5NN	45	47	47	50	51	27	34	44	50	42	58

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PCA	Analogy – 5NN	46	45	40	50	51	47	38	43	48	52	46
EW – 3 bins	SWR	48.4	42	42	58	55	21	52	44	52	60	58
PCA	Analogy – 1NN	49.1	46	42	48	49	49	51	48	50	48	60
SFS	CART (yes)	50.1	61	31	48	63	54	53	48	44	47	52
SFS	CART (no)	50.5	62	40	54	55	54	23	54	59	52	52
Natural logarithm	Analogy – 1NN	50.8	24	48	54	55	52	55	55	55	58	52
Normalization	PLSR	53.2	53	60	58	61	42	40	48	53	57	60
PCA	CART (yes)	57.1	63	64	65	66	56	44	48	59	60	46
PCA	CART (no)	57.1	63	64	65	66	56	44	48	59	60	46
EW – 5 bins	Analogy – 1NN	59.9	50	62	65	63	56	67	58	53	58	67
EF – 5 bins	CART (yes)	60.3	63	48	62	55	66	63	68	65	64	49
EF – 5 bins	CART (no)	60.3	63	48	62	55	66	63	68	65	64	49
EW – 3 bins	Analogy – 1NN	61.5	70	70	64	63	47	70	44	56	64	67
EW – 5 bins	SWR	61.8	54	72	60	62	52	70	59	62	67	60
EW – 5 bins	PLSR	62.6	73	64	65	60	56	58	48	71	67	64
EF – 5 bins	Analogy – 5NN	65.8	63	67	60	72	68	66	68	65	60	69
EF – 3 bins	CART (yes)	67.8	63	67	71	68	69	67	66	68	69	70
EF – 3 bins	CART (no)	67.8	63	67	71	68	69	67	66	68	69	70
EF – 3 bins	Analogy – 5NN	70.4	70	72	69	68	69	73	71	71	69	72
EF – 5 bins	Analogy – 1NN	71.1	73	72	71	68	69	73	71	70	72	72
SFS	PLSR	71.4	48	60	83	81	84	44	86	82	83	63
EW – 3 bins	PLSR	71.8	72	71	69	72	69	72	71	74	74	74
EF – 3 bins	Analogy – 1NN	72.3	73	75	71	72	69	75	71	71	72	74
Stepwise regression	Neural net	75.3	76	76	75	75	75	76	75	74	75	76
None	Neural net	76.6	77	77	77	76	76	77	77	76	76	77
PCA	Neural net	76.6	77	77	77	76	76	77	77	76	76	77
Normalization	PCR	78.9	79	80	79	79	79	79	76	80	78	80
EW – 3 bins	SLR	79.6	80	79	80	78	81	79	79	79	79	82
SFS	Neural net	80.6	83	83	75	83	78	86	82	76	83	77
EW – 5 bins	PCR	82	89	82	81	81	80	84	80	81	81	81
EW – 5 bins	SLR	82.2	81	87	83	80	84	81	80	83	80	83
PCA	SLR	84.4	82	81	96	83	83	85	82	84	83	85
SFS	PCR	85	87	89	85	86	82	89	82	85	82	83
PCA	PLSR	85.6	83	83	87	87	86	86	88	85	86	85
None	PLSR	85.6	83	83	87	87	86	86	88	85	86	85
Natural logarithm	SLR	86.7	91	87	82	83	89	83	86	85	91	90
Stepwise regression	PLSR	87.6	87	86	87	87	86	89	88	92	86	88
Natural logarithm	SWR	88.1	83	90	85	90	89	82	88	89	93	92
Natural logarithm	PLSR	90.1	92	90	91	92	89	89	88	89	89	92
EW – 3 bins	PCR	91.8	89	98	90	102	89	92	82	98	90	88
Natural logarithm	PCR	94.1	93	99	101	92	94	93	93	95	91	90
EF – 5 bins	SLR	94.1	93	99	91	91	93	93	105	89	93	94
PCA	PCR	94.1	93	92	93	92	94	93	105	92	93	94
None	PCR	94.1	93	92	93	92	94	93	105	92	93	94
Stepwise regression	PCR	95.6	93	92	93	92	106	93	93	107	93	94
EF – 3 bins	PLSR	97.3	98	95	98	97	97	98	95	96	100	99
EF – 5 bins	PLSR	98	100	97	96	98	97	101	96	97	99	99
EF – 5 bins	SWR	98.1	98	95	98	101	97	98	96	98	101	99
EF – 3 bins	SWR	99.9	100	99	98	98	106	98	96	100	102	102
EF – 5 bins	PCR	100.4	100	99	101	98	106	101	105	107	93	94
Natural logarithm	Neural net	100.8	100	99	101	103	100	101	99	100	102	103
EF – 5 bins	Neural net	100.8	100	99	101	103	100	101	99	100	102	103
EW – 5 bins	Neural net	100.8	100	99	101	103	100	101	99	100	102	103
EF – 3 bins	Neural net	100.8	100	99	101	103	100	101	99	100	102	103
Normalization	Neural net	100.8	100	99	101	103	100	101	99	100	102	103

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EW – 3 bins	Neural net	100.8	100	99	101	103	100	101	99	100	102	103
EF – 3 bins	PCR	102.7	100	99	101	103	106	101	105	107	102	103
EF – 3 bins	SLR	102.7	100	99	101	103	106	101	105	107	102	103

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression
EF–Equal frequency **EW**–Equal width

Table L.6: **Wins−Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging* $2N_1$. Ensembles are highlighted gray.

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top positive	2.2	3	4	4	1	1	1	3	1	3	1
IRWM	Top 3	5.8	4	8	4	4	2	7	10	4	7	8
IRWM	Top 4	5.8	4	8	2	4	2	7	10	4	10	7
Median	Top 5	5.8	13	1	1	19	15	1	1	4	1	2
Mean	Top 2	6.2	4	14	4	4	2	5	10	4	7	8
IRWM	Top 5	6.8	4	8	4	4	2	7	10	4	10	15
Median	Top	6.8	2	7	16	11	2	7	8	4	3	8
Median	Top 4	6.9	13	3	2	19	15	3	5	4	3	2
IRWM	Top 2	7.2	4	8	4	4	2	7	17	4	7	15
Median	Top 3	7.4	13	2	16	11	15	7	1	4	1	4
Mean	Top 3	8	10	14	4	4	2	7	10	4	10	15
IRWM	Top overall	8.2	18	5	4	19	2	7	6	1	16	4
Median	Top overall	8.3	18	5	4	4	2	7	6	17	16	4
Mean	Top 4	9.4	10	14	4	11	2	7	17	4	10	15
Mean	Top	9.4	4	8	4	19	2	18	17	4	10	8
Mean	Top 5	9.6	10	18	4	11	2	7	10	4	15	15
Median	Top 2	10.5	1	8	16	1	15	4	4	28	6	22
Mean	Top overall	15.2	24	14	4	11	22	18	10	21	20	8
Stepwise regression	Analogy – 5NN	17.5	32	18	16	19	2	18	17	1	20	32
Stepwise regression	Analogy – 1NN	18.5	20	18	16	30	15	5	23	17	26	15
None	Analogy – 1NN	19.4	21	18	21	11	26	18	26	19	26	8
Normalization	Analogy – 1NN	19.4	21	18	21	11	26	18	26	19	26	8
IRWM	Top positive	20.7	27	25	23	11	24	18	23	21	20	15
None	Analogy – 5NN	24.7	28	18	23	24	26	26	21	32	16	33
Normalization	Analogy – 5NN	24.7	28	18	23	24	26	26	21	32	16	33
Mean	Top positive	26.5	21	36	37	1	20	28	35	21	37	29
SFS	Analogy – 5NN	29.4	30	26	28	29	22	32	8	40	39	40
Natural logarithm	Analogy – 5NN	29.5	24	26	28	24	20	50	28	44	20	31
SFS	Analogy – 1NN	30.7	31	46	34	30	25	18	23	35	36	29
Stepwise regression	CART (no)	33.6	13	37	39	24	39	29	38	46	20	51
Stepwise regression	CART (yes)	33.6	13	37	39	24	39	29	38	46	20	51
Natural logarithm	CART (yes)	33.7	37	29	28	30	42	41	47	28	33	22
Natural logarithm	CART (no)	33.7	37	29	28	30	42	41	47	28	33	22
None	CART (yes)	33.8	37	29	26	30	50	44	47	24	29	22
None	CART (no)	33.8	37	29	26	30	50	44	47	24	29	22
Normalization	CART (yes)	34	37	29	28	30	50	44	47	24	29	22
Normalization	CART (no)	34	37	29	28	30	50	44	47	24	29	22
PCA	Analogy – 5NN	34.5	33	35	36	40	30	29	32	34	39	37
PCA	Analogy – 1NN	35.6	34	37	35	39	30	35	29	35	38	44
SFS	SWR	39	36	45	39	42	35	39	37	37	47	33
SFS	SLR	39.9	35	44	38	38	37	40	38	38	43	48
EW – 3 bins	Analogy – 5NN	40.4	43	50	42	56	32	37	35	31	42	36
Natural logarithm	Analogy – 1NN	40.8	24	41	43	43	42	43	41	40	45	46
PCA	CART (yes)	42	45	47	48	47	46	33	29	40	47	38
PCA	CART (no)	42	45	47	48	47	46	33	29	40	47	38
SFS	CART (yes)	42.9	61	28	43	46	67	37	32	39	35	41
EW – 5 bins	Analogy – 5NN	45	59	47	56	54	33	36	34	45	39	47
SFS	CART (no)	48.3	64	40	62	41	67	18	44	60	43	44
EF – 5 bins	CART (yes)	48.6	45	41	46	43	62	50	55	50	52	42
EF – 5 bins	CART (no)	48.6	45	41	46	43	62	50	55	50	52	42
Stepwise regression	SLR	50.6	43	51	55	56	35	53	42	54	55	62
Stepwise regression	SWR	52.8	45	55	63	62	37	58	43	56	54	55

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		Rank	1	2	3	4	5	6	7	8	9	10
EF – 3 bins	CART (yes)	53.3	65	53	52	47	69	44	47	50	55	51
EF – 3 bins	CART (no)	53.3	65	53	52	47	69	44	47	50	55	51
PCA	SWR	54.4	52	66	43	54	33	70	46	54	61	65
EF – 5 bins	Analogy – 5NN	55	52	52	50	56	62	56	60	56	46	60
EW – 5 bins	CART (yes)	55.3	52	56	64	56	50	53	64	62	47	49
EW – 5 bins	CART (no)	55.3	52	56	64	56	50	53	64	62	47	49
None	SLR	55.8	52	68	56	56	41	58	44	62	64	57
EF – 3 bins	Analogy – 5NN	57.6	60	59	54	51	59	58	60	59	60	56
EF – 3 bins	Analogy – 1NN	58.1	61	60	50	53	58	65	63	56	58	57
Normalization	SWR	59	45	61	56	65	46	63	67	62	62	63
None	SWR	59	45	61	56	65	46	63	67	62	62	63
EF – 5 bins	Analogy – 1NN	59.5	61	56	60	52	66	65	60	60	58	57
Normalization	SLR	59.8	58	66	61	65	42	58	55	67	65	61
EW – 3 bins	CART (yes)	62.6	69	63	67	63	59	67	55	48	66	69
EW – 3 bins	CART (no)	62.6	69	63	67	63	59	67	55	48	66	69
Normalization	PLSR	67.2	71	71	69	70	50	62	69	70	69	71
EW – 5 bins	Analogy – 1NN	68.5	67	69	66	68	71	71	69	69	68	67
EW – 3 bins	SWR	68.5	68	63	71	71	57	69	71	71	71	73
EW – 3 bins	Analogy – 1NN	68.7	73	72	70	68	62	72	64	68	70	68
EW – 5 bins	SWR	72.4	72	74	72	72	72	73	73	72	72	72
EW – 5 bins	PLSR	73.2	74	73	73	73	73	74	72	73	73	74
EW – 3 bins	PLSR	74.5	74	75	74	74	74	75	74	76	74	75
SFS	PLSR	75.5	52	70	85	83	86	57	89	82	85	66
None	Neural net	75.6	76	76	75	75	75	77	77	74	75	76
PCA	Neural net	75.6	76	76	75	75	75	77	77	74	75	76
Stepwise regression	Neural net	76.7	78	76	78	77	77	76	75	77	75	78
Normalization	PCR	78.5	79	79	79	78	78	79	76	79	78	80
EW – 3 bins	SLR	80.2	80	80	80	79	81	81	80	80	79	82
EW – 5 bins	PCR	81.5	84	82	81	81	80	84	80	81	81	81
EW – 5 bins	SLR	81.8	81	86	83	80	83	80	79	83	80	83
SFS	Neural net	82.9	86	83	77	85	78	87	87	78	89	79
Natural logarithm	SLR	84.1	86	83	81	82	88	83	83	85	85	85
PCA	SLR	84.4	83	81	93	83	82	84	84	84	85	85
Natural logarithm	SWR	85.3	81	86	83	85	83	82	85	86	93	89
Natural logarithm	PLSR	86.5	88	86	87	89	88	84	85	86	83	89
EW – 3 bins	PCR	87.4	84	93	86	94	86	89	82	93	84	83
SFS	PCR	87.4	91	91	89	87	83	88	87	91	82	85
PCA	PLSR	89.9	88	86	90	92	90	91	92	89	90	91
None	PLSR	89.9	88	86	90	92	90	91	92	89	90	91
Stepwise regression	PLSR	91	92	85	92	89	93	89	91	96	90	93
Natural logarithm	PCR	91.2	93	98	97	89	93	91	90	91	85	85
EF – 5 bins	SLR	91.6	93	98	88	88	90	91	94	88	93	93
EF – 5 bins	PLSR	95.2	98	92	93	96	95	96	94	96	96	96
EF – 5 bins	PCR	96	98	98	97	94	99	96	94	96	93	95
PCA	PCR	96.7	96	93	95	98	95	99	103	94	98	96
None	PCR	96.7	96	93	95	98	95	99	103	94	98	96
EF – 3 bins	PCR	97.9	98	98	100	97	99	96	94	100	97	100
EF – 5 bins	SWR	99.1	95	96	100	102	95	99	102	100	100	102
EF – 3 bins	PLSR	100	102	96	103	98	102	95	99	102	102	101
EF – 3 bins	SLR	100.4	103	103	97	98	99	104	99	96	103	102
Stepwise regression	PCR	101.1	98	98	100	103	103	102	98	110	100	99
EF – 3 bins	SWR	103.3	104	104	104	104	103	102	101	103	104	104
Natural logarithm	Neural net	104.7	105	105	105	105	105	105	103	104	105	105
EF – 5 bins	Neural net	104.8	105	105	105	105	105	105	103	105	105	105
EW – 5 bins	Neural net	104.8	105	105	105	105	105	105	103	105	105	105

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EF – 3 bins	Neural net	105.1	105	105	105	108	105	105	103	105	105	105
Normalization	Neural net	105.1	105	105	105	108	105	105	103	105	105	105
EW – 3 bins	Neural net	105.1	105	105	105	108	105	105	103	105	105	105

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

M

Graphical Visualization Of Ranking–Bagging $2N_1$

A graphical visualization of classifier ranking for the **Wins** and **Wins−Losses** ranking systems for *Bagging* $2N_1$, is provided in this appendix. Figures M.1 and M.3 display solo classifier performance over all 10 runs for **Wins** and **Wins−Losses** respectively. Figures M.2 and M.4 display classifier performance, solo and ensemble, over all 10 runs for **Wins** and **Wins−Losses** respectively.

All four figures use the same color coding to represent classifier ranking: Classifiers ranked in the top third in a run are coloured gray, classifiers ranked in the middle third are coloured white, and classifiers ranked in the bottom third are coloured black.

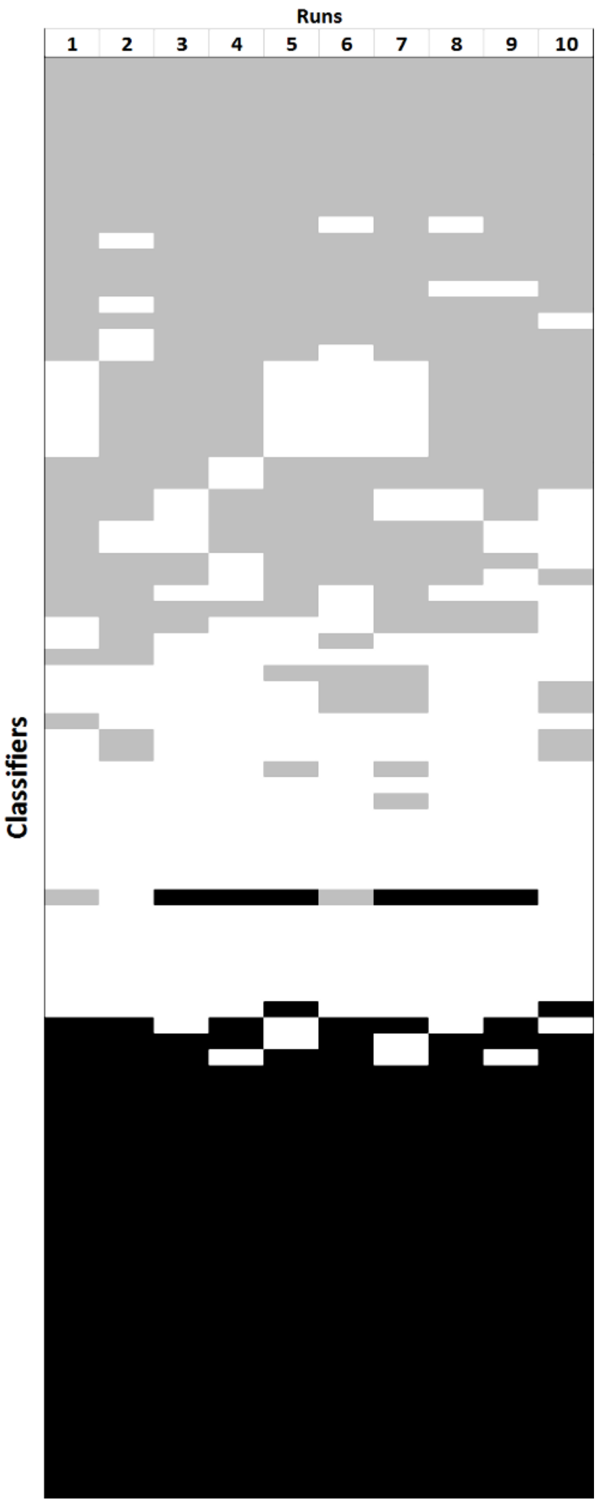


Figure M.1: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins** ranking.

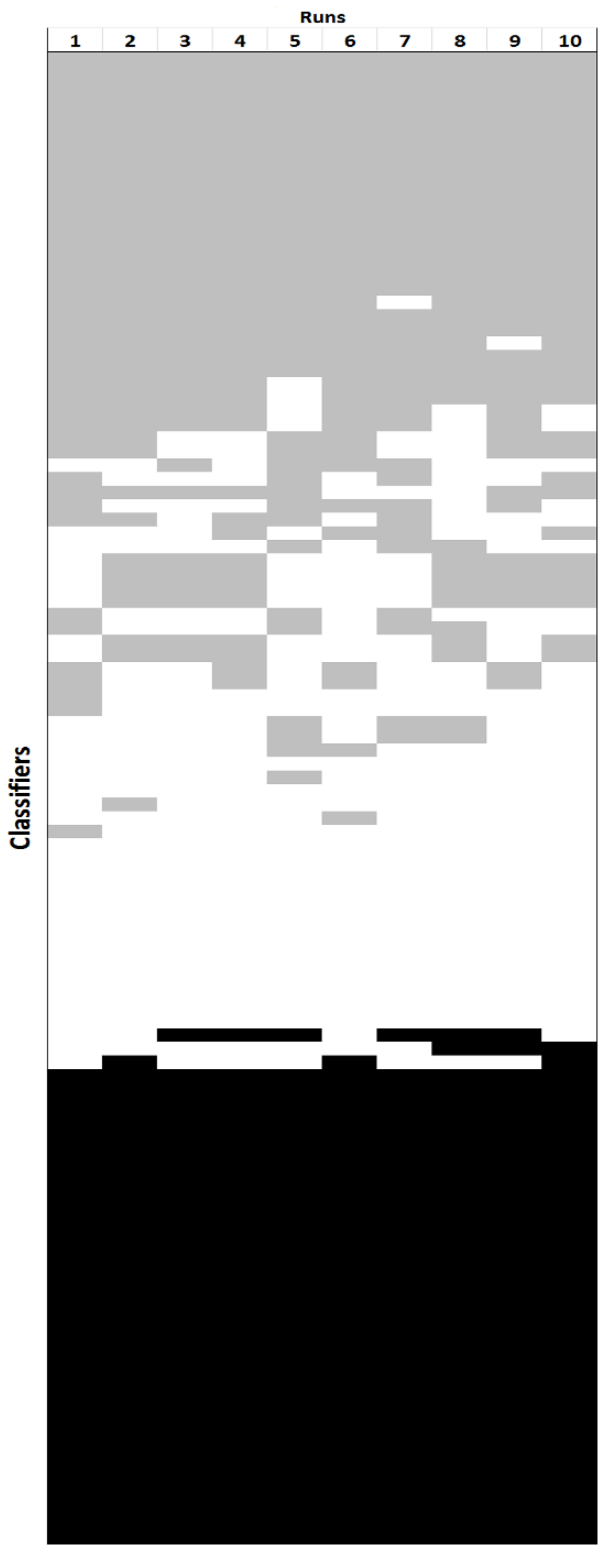


Figure M.2: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins** ranking.

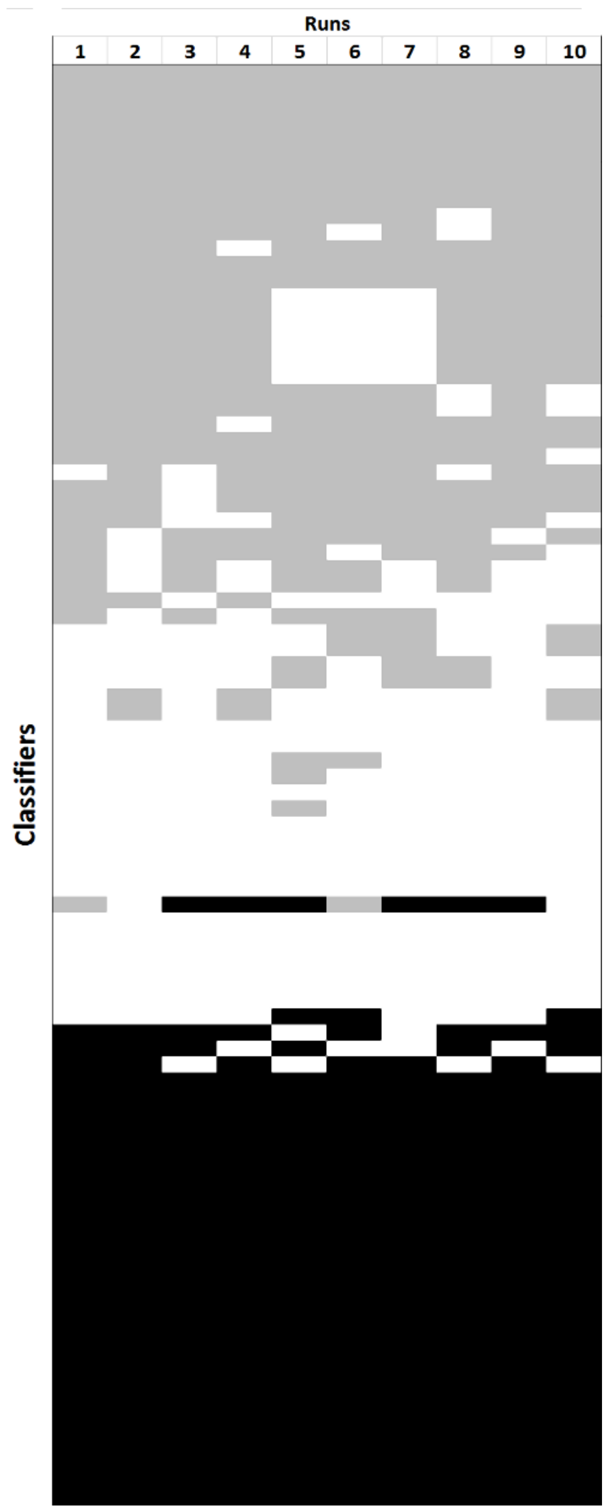


Figure M.3: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins−Losses** ranking.

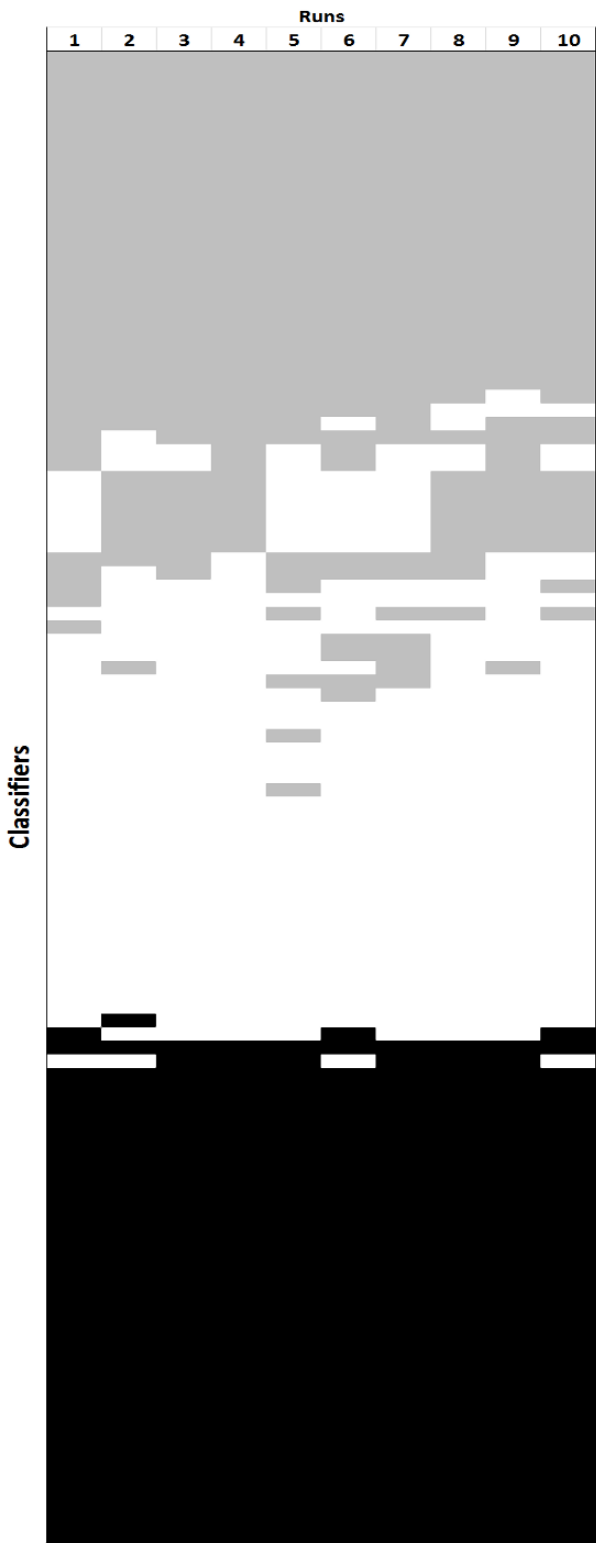


Figure M.4: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins**−**Losses** ranking.

N

Learner Rankings–Bagging $2N_2$

Classifier rankings for all *Bagging* $2N_2$ runs are provided in this Appendix. Tables N.1, N.2, and N.3 list the rankings of solo classifiers, in ascending order of average rank, for losses, wins and wins−losses respectively. Tables N.4, N.5, and N.6 list the rankings of solo and ensemble classifiers, in ascending order of average rank, for losses, wins, and wins−losses respectively. Note that ensemble classifiers are highlighted in gray.

Due to space constraints acronyms had to be used for certain pre-processors and learners in these tables. The acronyms used are explained in the very last row of each table.

Table N.1: **Losses** ranking for solo learners, over 10 runs of *Bagging* $2N_2$

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Stepwise regression	Analogy – 5NN	1	1	1	1	1	1	1	1	1	1	1
None	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Normalization	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
SFS	Analogy – 1NN	1.9	10	1	1	1	1	1	1	1	1	1
Stepwise regression	Analogy – 1NN	2	1	1	1	1	1	1	1	1	1	11
None	Analogy – 5NN	4.7	1	1	7	9	17	1	1	8	1	1
Normalization	Analogy – 5NN	4.7	1	1	7	9	17	1	1	8	1	1
Normalization	CART (yes)	7.9	13	11	11	9	1	1	1	11	8	13
Normalization	CART (no)	7.9	13	11	11	9	1	1	1	11	8	13
None	CART (yes)	7.9	13	11	11	9	1	1	1	11	8	13
None	CART (no)	7.9	13	11	11	9	1	1	1	11	8	13
SFS	Analogy – 5NN	9	27	23	9	1	11	1	1	1	15	1
Natural logarithm	Analogy – 5NN	9.4	10	8	1	1	21	1	14	22	15	1
Continued on next page												

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank 1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PCA	Analogy – 5NN	9.4	1	8	22	9	14	1	1	11	14	13
PCA	Analogy – 1NN	11.6	1	11	9	9	14	19	17	8	15	13
Natural logarithm	CART (yes)	11.7	13	11	11	19	11	1	14	16	8	13
Natural logarithm	CART (no)	11.7	13	11	11	19	11	1	14	16	8	13
SFS	CART (yes)	14.2	1	18	21	8	14	21	24	1	21	13
Stepwise regression	CART (yes)	15.2	23	19	11	9	26	1	21	16	25	1
Stepwise regression	CART (no)	15.2	23	19	11	9	26	1	21	16	25	1
SFS	CART (no)	16.3	23	8	22	19	1	19	25	16	18	12
SFS	Stepwise regression	22.9	19	28	19	25	22	32	17	23	19	25
Equal width – 3 bins	Analogy – 5NN	23.1	10	19	22	32	31	30	21	16	21	29
PCA	CART (yes)	24.7	19	32	30	26	17	27	19	24	23	30
PCA	CART (no)	24.7	19	32	30	26	17	27	19	24	23	30
SFS	Simple linear regression	25.9	22	19	19	22	28	32	33	27	19	38
Natural logarithm	Analogy – 1NN	26.5	28	23	32	30	25	22	29	26	28	22
Equal frequency – 3 bins	CART (yes)	27	31	25	28	26	23	25	26	27	33	26
Equal frequency – 3 bins	CART (no)	27	31	25	28	26	23	25	26	27	33	26
Stepwise regression	Simple linear regression	30.4	28	28	22	32	33	32	40	30	27	32
Equal frequency – 3 bins	Analogy – 1NN	32.5	28	37	33	35	31	30	31	31	37	32
Stepwise regression	Stepwise regression	32.6	31	28	34	35	34	32	38	32	30	32
Equal width – 5 bins	CART (yes)	33.3	37	46	22	22	28	45	36	44	31	22
Equal width – 5 bins	CART (no)	33.3	37	46	22	22	28	45	36	44	31	22
Equal frequency – 5 bins	Analogy – 5NN	33.6	37	28	44	31	38	27	28	32	33	38
Equal width – 5 bins	Analogy – 5NN	34.4	23	27	48	40	49	36	30	35	28	28
Equal frequency – 5 bins	CART (yes)	34.7	37	34	40	35	36	22	33	37	38	35
Equal frequency – 5 bins	CART (no)	34.7	37	34	40	35	36	22	33	37	38	35
Equal frequency – 3 bins	Analogy – 5NN	34.8	31	34	40	39	35	37	32	32	33	35
PCA	Stepwise regression	39.8	37	41	35	40	38	41	44	35	46	41
Equal frequency – 5 bins	Analogy – 1NN	39.9	43	38	44	40	38	38	38	39	43	38
Normalization	Stepwise regression	40	31	38	35	44	41	41	41	41	47	41
None	Stepwise regression	40	31	38	35	44	41	41	41	41	47	41
None	Simple linear regression	41.9	45	41	38	43	41	38	45	39	43	46
Normalization	Simple linear regression	43.1	45	45	38	44	45	38	46	41	43	46
Equal width – 3 bins	CART (yes)	44.6	47	41	44	44	46	48	48	46	38	44
Equal width – 3 bins	CART (no)	44.6	47	41	44	44	46	48	48	46	38	44
Normalization	PLSR	47.4	44	48	48	49	46	47	47	48	49	48
Equal width – 5 bins	Analogy – 1NN	50.2	51	49	51	51	50	50	50	49	50	51
Equal width – 3 bins	Stepwise regression	50.6	49	50	50	52	50	51	52	51	52	49
Equal width – 5 bins	Stepwise regression	50.6	50	51	52	50	52	51	50	49	51	50
SFS	PLSR	50.7	66	66	43	34	41	41	43	69	38	66
Equal width – 5 bins	PLSR	52.6	52	52	53	53	53	53	53	52	53	52
None	Neural net	54.5	56	53	54	56	56	54	54	55	54	53
PCA	Neural net	54.5	56	53	54	56	56	54	54	55	54	53
Equal width – 3 bins	PLSR	54.8	53	53	56	54	54	57	56	53	56	56
Equal width – 3 bins	Analogy – 1NN	54.8	53	53	57	54	55	56	56	53	56	55
Stepwise regression	Neural net	57.6	55	57	58	58	59	58	58	58	58	57
Normalization	PCR	57.9	56	57	58	59	56	59	59	58	59	58
Equal width – 5 bins	PCR	60.6	65	59	61	60	59	61	59	60	63	59
Natural logarithm	Simple linear regression	60.7	60	61	60	60	61	61	63	60	61	60
Equal width – 3 bins	PCR	61.8	62	62	61	63	63	60	61	62	62	62
Natural logarithm	Stepwise regression	62.3	60	63	63	60	62	61	66	62	66	60
Equal width – 3 bins	Simple linear regression	63.4	62	64	65	64	65	61	63	62	65	63
Equal width – 5 bins	Simple linear regression	64.2	69	66	63	64	63	61	61	66	66	63
Natural logarithm	PLSR	64.7	66	64	65	67	65	61	63	67	63	66
PCA	Simple linear regression	65.4	62	66	69	64	69	61	69	62	69	63
Equal frequency – 5 bins	Simple linear regression	65.9	66	66	65	68	65	61	66	67	69	66

Continued on next page

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
SFS	Neural net	66.5	59	59	74	71	65	72	76	57	60	72
Natural logarithm	PCR	66.9	69	66	65	68	70	61	66	69	66	69
Equal frequency – 5 bins	PCR	69.6	69	71	69	68	72	70	70	69	69	69
Equal frequency – 3 bins	PCR	70.3	72	71	69	71	70	70	70	72	69	69
Equal frequency – 5 bins	PLSR	72.1	72	74	69	73	72	72	70	75	69	75
Equal frequency – 3 bins	Simple linear regression	72.3	72	71	74	73	72	72	70	72	75	72
SFS	PCR	74.6	79	75	74	77	72	77	70	72	78	72
Equal frequency – 5 bins	Stepwise regression	75.9	80	81	69	80	76	75	70	78	74	76
PCA	PLSR	77	75	77	77	75	76	80	78	76	79	77
None	PLSR	77	75	77	77	75	76	80	78	76	79	77
Stepwise regression	PLSR	77.8	77	75	79	78	76	80	76	80	75	82
Equal frequency – 3 bins	PLSR	77.8	77	81	80	79	76	75	80	78	75	77
PCA	PCR	79.8	82	77	82	80	81	78	80	80	81	77
None	PCR	79.8	82	77	82	80	81	78	80	80	81	77
Stepwise regression	PCR	82.3	80	81	84	83	83	80	83	83	83	83
Equal frequency – 3 bins	Stepwise regression	83.4	84	81	81	84	84	84	84	84	84	84
Natural logarithm	Neural net	85	85	85	85	85	85	85	85	85	85	85
Equal frequency – 5 bins	Neural net	85.2	85	86	86	85	85	85	85	85	85	85
Equal frequency – 3 bins	Neural net	85.5	87	86	87	85	85	85	85	85	85	85
Equal width – 5 bins	Neural net	85.5	87	86	87	85	85	85	85	85	85	85
Normalization	Neural net	85.9	87	86	87	85	85	85	85	85	85	89
Equal width – 3 bins	Neural net	85.9	87	86	87	85	85	85	85	85	85	89

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

Table N.2: **Wins** ranking for solo learners, over 10 runs of *Bagging $2N_2$*.

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Stepwise regression	Analogy – 5NN	3.5	4	3	5	4	5	2	1	3	2	6
Stepwise regression	Analogy – 1NN	5.1	4	3	6	7	5	2	1	4	4	15
SFS	Analogy – 1NN	6.9	17	5	9	8	5	2	6	6	5	6
SFS	Simple linear regression	7	2	1	1	1	3	20	15	2	1	24
SFS	Stepwise regression	7.3	6	5	1	4	3	20	27	1	3	3
None	Analogy – 5NN	8.5	7	5	12	14	17	2	7	10	5	6
Normalization	Analogy – 5NN	8.5	7	5	12	14	17	2	7	10	5	6
None	Analogy – 1NN	8.8	7	11	10	8	12	16	7	6	5	6
Normalization	Analogy – 1NN	8.8	7	11	10	8	12	16	7	6	5	6
Natural logarithm	Analogy – 5NN	9.2	17	5	6	4	21	1	4	19	12	3
Normalization	CART (yes)	11.4	11	13	14	19	5	2	7	15	12	16
Normalization	CART (no)	11.4	11	13	14	19	5	2	7	15	12	16
None	CART (yes)	11.9	11	13	14	16	5	2	7	15	18	18
None	CART (no)	11.9	11	13	14	16	5	2	7	15	18	18
Natural logarithm	CART (yes)	13.8	11	13	14	19	12	2	15	19	15	18
Natural logarithm	CART (no)	13.8	11	13	14	19	12	2	15	19	15	18
SFS	Analogy – 5NN	14.4	39	31	26	8	12	2	1	4	15	6
Equal width – 5 bins	CART (yes)	14.8	19	36	3	2	1	34	18	29	5	1
Equal width – 5 bins	CART (no)	14.8	19	36	3	2	1	34	18	29	5	1
Equal width – 5 bins	Analogy – 5NN	17.1	1	2	31	19	42	19	5	25	24	3
Stepwise regression	CART (yes)	17.9	21	28	14	12	35	2	21	10	30	6
Stepwise regression	CART (no)	17.9	21	28	14	12	35	2	21	10	30	6
Equal width – 3 bins	Analogy – 5NN	19.3	2	5	31	34	22	23	20	6	28	22
Stepwise regression	Simple linear regression	19.5	26	13	6	16	22	24	28	10	26	24
None	Simple linear regression	24.9	29	22	23	26	22	24	30	25	18	30
PCA	Analogy – 5NN	25.5	26	20	42	25	28	16	21	31	24	22
Stepwise regression	Stepwise regression	26.8	24	21	26	28	25	24	29	34	30	27
Normalization	Simple linear regression	26.8	33	24	22	26	28	24	30	25	18	38
SFS	CART (yes)	27.3	21	24	34	19	31	24	37	19	37	27
PCA	Analogy – 1NN	27.4	26	23	31	28	25	24	34	28	28	27
PCA	Stepwise regression	28.3	24	31	26	28	28	36	37	23	26	24
Equal width – 3 bins	CART (yes)	30.3	36	24	23	28	31	39	34	38	18	32
Equal width – 3 bins	CART (no)	30.3	36	24	23	28	31	39	34	38	18	32
SFS	CART (no)	31	42	28	42	33	25	22	30	23	35	30
Normalization	Stepwise regression	32.7	31	31	26	36	37	36	37	31	30	32
None	Stepwise regression	32.7	31	31	26	36	37	36	37	31	30	32
Natural logarithm	Analogy – 1NN	33.4	39	31	37	34	31	30	30	34	36	32
PCA	CART (yes)	34.7	33	42	37	36	19	45	21	34	37	43
PCA	CART (no)	34.7	33	42	37	36	19	45	21	34	37	43
Normalization	PLSR	38	29	38	34	42	37	39	41	40	41	39
Equal width – 3 bins	Stepwise regression	39.1	36	39	34	42	37	43	45	40	43	32
Equal frequency – 5 bins	Analogy – 5NN	40.1	46	39	46	36	50	30	26	43	43	42
Equal width – 5 bins	Stepwise regression	41.1	39	41	37	42	43	43	45	40	42	39
Equal frequency – 5 bins	CART (yes)	43.9	44	45	46	46	47	30	47	44	47	43
Equal frequency – 5 bins	CART (no)	43.9	44	45	46	46	47	30	47	44	47	43
Equal width – 5 bins	PLSR	44.5	42	44	44	45	44	48	50	44	43	41
Equal frequency – 3 bins	CART (yes)	46.4	46	45	51	46	45	52	43	44	49	43
Equal frequency – 3 bins	CART (no)	46.4	46	45	51	46	45	52	43	44	49	43
SFS	PLSR	47.7	64	62	37	36	37	39	41	63	37	61
Equal frequency – 3 bins	Analogy – 5NN	47.9	46	45	49	51	51	49	47	49	49	43
Equal width – 5 bins	Analogy – 1NN	48.3	50	50	45	50	47	45	51	49	46	50
Equal frequency – 5 bins	Analogy – 1NN	50.8	50	50	51	51	51	51	51	51	52	50
Equal width – 3 bins	PLSR	51.8	53	53	49	51	51	49	53	53	53	53

Continued on next page

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Equal frequency – 3 bins	Analogy – 1NN	52.8	52	52	54	54	54	52	53	52	53	52
Stepwise regression	Neural net	54.6	54	54	55	55	55	55	55	54	55	54
None	Neural net	56.1	55	56	56	56	57	57	56	57	56	55
PCA	Neural net	56.1	55	56	56	56	57	57	56	57	56	55
Equal width – 3 bins	Analogy – 1NN	56.4	55	59	58	56	56	56	58	55	56	55
Normalization	PCR	59	59	59	58	59	59	59	60	59	60	58
Equal width – 5 bins	PCR	60	61	58	61	60	59	60	59	61	62	59
Equal width – 3 bins	Simple linear regression	60.5	60	61	60	61	61	60	61	60	61	60
SFS	Neural net	60.6	55	55	63	64	62	64	64	55	59	65
PCA	Simple linear regression	62.6	62	65	63	62	63	62	63	62	63	61
Equal width – 5 bins	Simple linear regression	64.3	66	63	63	63	63	64	68	65	65	63
Natural logarithm	Simple linear regression	65	63	64	62	65	63	69	69	65	66	64
SFS	PCR	65.7	72	71	66	67	63	62	62	63	66	65
Stepwise regression	PLSR	66.1	66	66	66	67	67	66	64	67	64	68
PCA	PLSR	66.8	66	67	68	67	68	67	64	67	66	68
None	PLSR	66.8	66	67	68	67	68	67	64	67	66	68
Natural logarithm	Stepwise regression	67.8	65	67	68	65	68	70	72	67	71	65
Natural logarithm	PLSR	71.2	70	72	72	72	72	70	71	71	71	71
Natural logarithm	PCR	72.5	72	72	72	72	72	70	72	72	71	80
Stepwise regression	PCR	72.9	72	72	76	74	72	73	72	72	74	72
PCA	PCR	73.3	75	72	77	74	72	73	72	72	74	72
None	PCR	73.3	75	72	77	74	72	73	72	72	74	72
Equal width – 3 bins	PCR	74.1	81	67	71	71	71	81	69	81	70	79
Equal frequency – 5 bins	Simple linear regression	74.7	71	82	72	74	72	73	72	72	81	78
Equal frequency – 3 bins	PLSR	77.9	78	78	79	78	78	78	78	78	79	75
Equal frequency – 3 bins	Stepwise regression	78.4	75	78	79	80	80	81	80	78	78	75
Equal frequency – 5 bins	PCR	78.5	84	77	72	82	81	73	80	82	74	80
Equal frequency – 5 bins	Stepwise regression	78.6	78	78	82	78	78	78	80	78	81	75
Equal frequency – 5 bins	PLSR	79.5	78	78	82	80	81	80	79	77	80	80
Natural logarithm	Neural net	81.1	81	82	79	82	81	83	80	82	81	80
Equal frequency – 5 bins	Neural net	81.4	81	82	82	82	81	83	80	82	81	80
Equal frequency – 3 bins	PCR	82	84	82	85	82	81	83	80	82	81	80
Equal frequency – 3 bins	Simple linear regression	82	84	82	85	82	81	83	80	82	81	80
Equal frequency – 3 bins	Neural net	82	84	82	85	82	81	83	80	82	81	80
Equal width – 5 bins	Neural net	82	84	82	85	82	81	83	80	82	81	80
Normalization	Neural net	82	84	82	85	82	81	83	80	82	81	80
Equal width – 3 bins	Neural net	82	84	82	85	82	81	83	80	82	81	80

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection **PLSR**–Partial least squares regression

Table N.3: **Wins−Losses** ranking for solo learners, over 10 runs of *Bagging* $2N_2$.

Pre-processing Option	Learner	Av.	Run Rank									
		Rank	1	2	3	4	5	6	7	8	9	10
Stepwise regression	Analogy – 5NN	1.3	1	2	1	1	1	2	1	1	1	2
Stepwise regression	Analogy – 1NN	2.8	1	2	2	3	1	2	1	2	3	11
SFS	Analogy – 1NN	4.1	10	4	4	4	1	2	5	5	4	2
None	Analogy – 1NN	6.3	4	7	7	4	8	16	6	5	4	2
Normalization	Analogy – 1NN	6.3	4	7	7	4	8	16	6	5	4	2
None	Analogy – 5NN	6.7	4	4	11	13	13	2	6	8	4	2
Normalization	Analogy – 5NN	6.7	4	4	11	13	13	2	6	8	4	2
Natural logarithm	Analogy – 5NN	8.4	10	7	2	1	22	1	4	22	14	1
Normalization	CART (yes)	10.2	14	10	14	19	1	2	6	12	10	14
Normalization	CART (no)	10.2	14	10	14	19	1	2	6	12	10	14
None	CART (yes)	10.5	14	10	14	16	1	2	6	12	14	16
None	CART (no)	10.5	14	10	14	16	1	2	6	12	14	16
SFS	Analogy – 5NN	11.8	32	26	22	4	10	2	1	2	17	2
Natural logarithm	CART (yes)	13.1	14	10	14	21	10	2	14	18	12	16
Natural logarithm	CART (no)	13.1	14	10	14	21	10	2	14	18	12	16
SFS	Stepwise regression	13.7	10	22	4	13	13	26	20	4	9	16
SFS	Simple linear regression	15.2	8	1	4	8	22	26	28	23	2	30
Stepwise regression	CART (yes)	16	22	22	14	11	29	2	20	12	26	2
Stepwise regression	CART (no)	16	22	22	14	11	29	2	20	12	26	2
PCA	Analogy – 5NN	19.4	20	10	26	21	22	16	16	24	18	21
SFS	CART (yes)	20.4	10	20	24	16	25	21	27	10	28	23
Equal width – 3 bins	Analogy – 5NN	20.5	3	17	24	33	26	26	17	10	22	27
PCA	Analogy – 1NN	21.2	20	18	23	24	21	20	24	20	19	23
SFS	CART (no)	22	27	19	26	25	13	19	26	20	22	23
Equal width – 5 bins	CART (yes)	23.5	27	41	9	9	19	34	30	34	20	12
Equal width – 5 bins	CART (no)	23.5	27	41	9	9	19	34	30	34	20	12
Stepwise regression	Simple linear regression	25.9	26	25	13	26	28	30	34	25	24	28
Equal width – 5 bins	Analogy – 5NN	26.8	8	20	42	27	47	25	23	30	24	22
Natural logarithm	Analogy – 1NN	28.6	35	26	36	27	26	22	28	28	32	26
PCA	CART (yes)	28.7	24	39	34	27	17	34	17	26	29	40
PCA	CART (no)	28.7	24	39	34	27	17	34	17	26	29	40
Stepwise regression	Stepwise regression	30.4	30	28	28	31	31	30	34	32	31	29
None	Simple linear regression	33.3	37	29	30	33	32	32	38	31	35	36
PCA	Stepwise regression	33.6	31	37	30	33	35	34	39	29	37	31
Normalization	Stepwise regression	34.7	32	30	30	42	37	34	36	34	40	32
None	Stepwise regression	34.7	32	30	30	42	37	34	36	34	40	32
Normalization	Simple linear regression	35.2	38	37	28	36	36	32	39	32	35	39
Equal frequency – 3 bins	CART (yes)	36.7	42	35	39	38	32	41	32	34	42	32
Equal frequency – 3 bins	CART (no)	36.7	42	35	39	38	32	41	32	34	42	32
Equal frequency – 5 bins	Analogy – 5NN	37.7	45	30	48	32	47	29	25	40	38	43
Equal width – 3 bins	CART (yes)	40	46	30	37	38	40	48	46	46	33	36
Equal width – 3 bins	CART (no)	40	46	30	37	38	40	48	46	46	33	36
Equal frequency – 5 bins	CART (yes)	41.1	39	43	45	44	44	22	44	42	45	43
Equal frequency – 5 bins	CART (no)	41.1	39	43	45	44	44	22	44	42	45	43
Equal frequency – 3 bins	Analogy – 5NN	43.4	42	43	47	46	46	45	39	41	42	43
Equal frequency – 3 bins	Analogy – 1NN	43.8	39	48	43	46	42	44	39	42	48	47
Normalization	PLSR	44.3	36	46	43	49	42	46	46	48	45	42
Equal frequency – 5 bins	Analogy – 1NN	48	48	47	49	48	49	47	49	45	49	49
SFS	PLSR	48.5	65	63	39	36	37	41	39	65	38	62
Equal width – 3 bins	Stepwise regression	49.9	49	49	50	51	50	50	51	50	52	47
Equal width – 5 bins	Stepwise regression	50.1	50	50	51	50	51	50	50	49	50	50
Equal width – 5 bins	Analogy – 1NN	51.6	52	51	52	52	52	52	52	51	50	52

Continued on next page

Pre-processing Option	**Learner**	**Av.**	**Run Rank**									
		Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Equal width – 5 bins	PLSR	52.4	51	52	53	53	53	53	53	52	53	51
Equal width – 3 bins	PLSR	53.6	53	53	54	54	54	54	54	53	54	53
Stepwise regression	Neural net	55.6	54	54	57	55	55	58	57	55	57	54
None	Neural net	55.7	56	55	55	57	57	55	55	57	55	55
PCA	Neural net	55.7	56	55	55	57	57	55	55	57	55	55
Equal width – 3 bins	Analogy – 1NN	56.6	55	58	58	56	55	57	58	54	58	57
Normalization	PCR	59.1	59	60	59	59	59	59	60	59	59	58
Equal width – 5 bins	PCR	60	61	58	61	60	60	60	59	60	62	59
Equal width – 3 bins	Simple linear regression	60.6	60	61	60	61	61	60	61	61	61	60
SFS	Neural net	62	58	57	65	66	62	64	68	55	59	66
Natural logarithm	Simple linear regression	62.9	61	62	61	63	63	65	66	63	63	62
PCA	Simple linear regression	63.3	63	66	64	62	66	62	63	62	64	61
Equal width – 5 bins	Simple linear regression	63.8	66	64	63	63	64	63	63	64	66	62
Natural logarithm	Stepwise regression	65.8	64	66	65	63	65	66	70	65	69	65
Equal width – 3 bins	PCR	67.7	72	64	65	66	67	72	65	73	64	69
Natural logarithm	PLSR	67.7	67	68	69	68	69	66	67	68	67	68
SFS	PCR	68.5	75	71	68	72	67	66	62	67	71	66
Natural logarithm	PCR	70.2	71	70	69	69	72	66	70	70	69	76
Stepwise regression	PLSR	71.3	72	68	72	74	71	73	68	74	68	73
Equal frequency – 5 bins	Simple linear regression	71.4	68	78	69	72	69	70	70	69	79	70
PCA	PLSR	71.5	68	72	74	69	72	74	73	70	73	70
None	PLSR	71.5	68	72	74	69	72	74	73	70	73	70
Equal frequency – 5 bins	PCR	74.8	75	72	72	75	81	71	79	76	71	76
Equal frequency – 5 bins	PLSR	76.4	74	75	76	75	81	76	75	75	75	82
PCA	PCR	76.8	81	75	82	77	75	77	75	76	76	74
None	PCR	76.8	81	75	82	77	75	77	75	76	76	74
Equal frequency – 3 bins	PCR	78.3	77	80	78	77	77	81	79	79	79	76
Stepwise regression	PCR	79.2	77	78	82	82	77	81	78	79	79	79
Equal frequency – 5 bins	Stepwise regression	79.7	81	82	76	82	77	77	79	82	82	79
Equal frequency – 3 bins	PLSR	79.9	80	82	80	80	77	77	83	82	76	82
Equal frequency – 3 bins	Simple linear regression	80	77	80	79	80	81	83	79	79	83	79
Equal frequency – 3 bins	Stepwise regression	83.5	84	82	81	84	84	84	84	84	84	84
Natural logarithm	Neural net	85	85	85	85	85	85	85	85	85	85	85
Equal frequency – 5 bins	Neural net	85.2	85	86	86	85	85	85	85	85	85	85
Equal frequency – 3 bins	Neural net	85.5	87	86	87	85	85	85	85	85	85	85
Equal width – 5 bins	Neural net	85.5	87	86	87	85	85	85	85	85	85	85
Normalization	Neural net	85.9	87	86	87	85	85	85	85	85	85	89
Equal width – 3 bins	Neural net	85.9	87	86	87	85	85	85	85	85	85	89

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

Table N.4: **Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging $2N_2$*. Ensembles are highlighted gray.

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top positive	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top overall	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 2	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 3	1	1	1	1	1	1	1	1	1	1	1
Mean	Top	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 4	1	1	1	1	1	1	1	1	1	1	1
IRWM	Top 5	1	1	1	1	1	1	1	1	1	1	1
None	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Normalization	Analogy – 1NN	1	1	1	1	1	1	1	1	1	1	1
Mean	Top 2	2.4	1	1	15	1	1	1	1	1	1	1
Mean	Top 3	2.4	1	1	15	1	1	1	1	1	1	1
Mean	Top 4	2.4	1	1	15	1	1	1	1	1	1	1
Median	Top 5	2.9	1	1	1	1	1	1	1	1	20	1
Mean	Top 5	3.4	1	1	1	25	1	1	1	1	1	1
Median	Top 4	4.3	1	1	15	1	1	1	1	1	20	1
Mean	Top overall	4.4	21	1	15	1	1	1	1	1	1	1
SFS	Analogy – 1NN	5.9	27	24	1	1	1	1	1	1	1	1
Median	Top 2	6.8	1	1	15	1	26	1	1	1	1	20
Median	Top 3	7.2	1	1	25	1	1	1	1	1	20	20
Median	Top overall	11.3	24	1	15	20	1	1	29	1	20	1
Median	Top	11.5	21	1	1	20	1	1	29	1	20	20
None	Analogy – 5NN	16.1	1	1	25	26	32	32	1	22	1	20
Normalization	Analogy – 5NN	16.1	1	1	25	26	32	32	1	22	1	20
Stepwise regression	Analogy – 1NN	17.9	21	24	15	20	26	1	1	22	20	29
Stepwise regression	Analogy – 5NN	18.1	1	24	15	20	26	32	1	22	20	20
IRWM	Top positive	19.1	24	1	15	26	1	32	29	32	1	30
Normalization	CART (yes)	19.6	28	27	28	26	1	1	1	27	27	30
Normalization	CART (no)	19.6	28	27	28	26	1	1	1	27	27	30
None	CART (yes)	19.6	28	27	28	26	1	1	1	27	27	30
None	CART (no)	19.6	28	27	28	26	1	1	1	27	27	30
SFS	Analogy – 5NN	19.9	41	37	36	1	26	1	1	22	33	1
Natural logarithm	CART (yes)	26.1	28	27	28	33	26	1	29	32	27	30
Natural logarithm	CART (no)	26.1	28	27	28	33	26	1	29	32	27	30
Natural logarithm	Analogy – 5NN	26.6	24	27	1	20	36	32	34	37	35	20
Stepwise regression	CART (yes)	30.9	38	35	28	33	40	1	37	35	42	20
Stepwise regression	CART (no)	30.9	38	35	28	33	40	1	37	35	42	20
PCA	Analogy – 5NN	35.9	28	27	42	37	37	37	37	41	35	38
PCA	Analogy – 1NN	37	28	37	38	38	37	42	41	37	33	39
PCA	CART (yes)	38.7	36	41	39	39	32	47	35	39	39	40
PCA	CART (no)	38.7	36	41	39	39	32	47	35	39	39	40
Natural logarithm	Analogy – 1NN	40.2	42	37	41	41	39	38	42	41	44	37
Mean	Top positive	41.1	42	44	37	42	43	44	37	43	39	40
SFS	CART (yes)	43.7	40	51	50	44	43	45	52	27	37	48
SFS	CART (no)	44.2	42	40	49	45	42	41	54	48	37	44
EF – 5 bins	Analogy – 5NN	44.6	46	41	46	42	49	42	42	45	45	48
EF – 3 bins	Analogy – 5NN	45.1	45	45	43	47	45	49	46	43	45	43
EF – 5 bins	CART (yes)	45.5	47	45	43	47	46	38	47	48	50	44
EF – 5 bins	CART (no)	45.5	47	45	43	47	46	38	47	48	50	44
EF – 5 bins	Analogy – 1NN	47.8	47	48	46	45	49	46	50	51	52	44
EF – 3 bins	CART (yes)	49.6	56	48	55	50	51	50	44	45	47	50
EF – 3 bins	CART (no)	49.6	56	48	55	50	51	50	44	45	47	50
EF – 3 bins	Analogy – 1NN	49.7	47	53	46	50	48	50	50	52	49	52

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EW – 3 bins	Analogy – 5NN	53.2	47	52	51	57	57	53	53	52	53	57
SFS	SWR	53.4	52	58	51	56	53	56	49	54	53	52
SFS	SLR	54.4	53	53	51	53	54	56	56	55	53	60
Stepwise regression	SLR	56.4	55	56	54	57	58	54	60	56	56	58
Stepwise regression	SWR	57.6	58	56	59	60	59	54	57	57	58	58
EW – 5 bins	CART (yes)	58.6	61	66	55	53	54	65	57	64	59	52
EW – 5 bins	CART (no)	58.6	61	66	55	53	54	65	57	64	59	52
EW – 5 bins	Analogy – 5NN	58.8	53	55	68	61	69	56	55	58	57	56
Normalization	SWR	61.2	58	59	60	64	60	62	61	61	67	60
None	SWR	61.2	58	59	60	64	60	62	61	61	67	60
PCA	SWR	61.4	61	61	60	62	60	62	64	58	66	60
None	SLR	63	65	61	63	63	64	59	65	60	64	66
Normalization	SLR	64	65	65	63	64	65	59	66	63	64	66
EW – 3 bins	CART (yes)	65.2	67	61	66	64	66	68	68	66	62	64
EW – 3 bins	CART (no)	65.2	67	61	66	64	66	68	68	66	62	64
Normalization	PLSR	67.5	64	68	69	69	66	67	67	68	69	68
EW – 5 bins	Analogy – 1NN	70.2	71	69	71	71	70	70	70	69	70	71
EW – 3 bins	SWR	70.6	69	70	70	72	70	71	72	71	72	69
EW – 5 bins	SWR	70.6	70	71	72	70	72	71	70	69	71	70
SFS	PLSR	71	86	86	63	59	60	59	61	89	61	86
EW – 5 bins	PLSR	72.6	72	72	73	73	73	73	73	72	73	72
None	Neural net	74.5	76	73	74	76	76	74	74	75	74	73
PCA	Neural net	74.5	76	73	74	76	76	74	74	75	74	73
EW – 3 bins	PLSR	74.8	73	73	76	74	74	77	76	73	76	76
EW – 3 bins	Analogy – 1NN	74.8	73	73	77	74	75	76	76	73	76	75
Stepwise regression	Neural net	77.6	75	77	78	78	79	78	78	78	78	77
Normalization	PCR	77.9	76	77	78	79	76	79	79	78	79	78
EW – 5 bins	PCR	80.6	85	79	81	80	79	81	79	80	83	79
Natural logarithm	SLR	80.7	80	81	80	80	81	81	83	80	81	80
EW – 3 bins	PCR	81.8	82	82	81	83	83	80	81	82	82	82
Natural logarithm	SWR	82.3	80	83	83	80	82	81	86	82	86	80
EW – 3 bins	SLR	83.4	82	84	85	84	85	81	83	82	85	83
EW – 5 bins	SLR	84.2	89	86	83	84	83	81	81	86	86	83
Natural logarithm	PLSR	84.7	86	84	85	87	85	81	83	87	83	86
PCA	SLR	85.4	82	86	89	84	89	81	89	82	89	83
EF – 5 bins	SLR	85.9	86	86	85	88	85	81	86	87	89	86
SFS	Neural net	86.5	79	79	94	91	85	92	96	77	80	92
Natural logarithm	PCR	86.9	89	86	85	88	90	81	86	89	86	89
EF – 5 bins	PCR	89.6	89	91	89	88	92	90	90	89	89	89
EF – 3 bins	PCR	90.3	92	91	89	91	90	90	90	92	89	89
EF – 5 bins	PLSR	92.1	92	94	89	93	92	92	90	95	89	95
EF – 3 bins	SLR	92.3	92	91	94	93	92	92	90	92	95	92
SFS	PCR	94.6	99	95	94	97	92	97	90	92	98	92
EF – 5 bins	SWR	95.9	100	101	89	100	96	95	90	98	94	96
PCA	PLSR	97	95	97	97	95	96	100	98	96	99	97
None	PLSR	97	95	97	97	95	96	100	98	96	99	97
Stepwise regression	PLSR	97.8	97	95	99	98	96	100	96	100	95	102
EF – 3 bins	PLSR	97.8	97	101	100	99	96	95	100	98	95	97
PCA	PCR	99.8	102	97	102	100	101	98	100	100	101	97
None	PCR	99.8	102	97	102	100	101	98	100	100	101	97
Stepwise regression	PCR	102.3	100	101	104	103	103	100	103	103	103	103
EF – 3 bins	SWR	103.4	104	101	101	104	104	104	104	104	104	104
Natural logarithm	Neural net	105	105	105	105	105	105	105	105	105	105	105
EF – 5 bins	Neural net	105.2	105	106	106	105	105	105	105	105	105	105
EF – 3 bins	Neural net	105.5	107	106	107	105	105	105	105	105	105	105

Continued on next page

Pre-processing	Learner	Av.	Run Rank									
Option		Rank	1	2	3	4	5	6	7	8	9	10
EW – 5 bins	Neural net	105.5	107	106	107	105	105	105	105	105	105	105
Normalization	Neural net	105.9	107	106	107	105	105	105	105	105	105	109
EW – 3 bins	Neural net	105.9	107	106	107	105	105	105	105	105	105	109

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression **SLR**–Simple linear regression **SWR**–Stepwise regression
EF–Equal frequency **EW**–Equal width

Table N.5: **Wins** ranking for ensemble and solo classifiers, over 10 runs of *Bagging 2N*$_2$. Ensembles are highlighted gray.

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top	4	11	3	1	8	1	4	7	1	3	1
Median	Top 3	4.1	3	3	3	4	1	2	2	2	15	6
Median	Top 2	5.1	7	3	6	1	13	4	2	2	7	6
Median	Top 4	5.3	2	3	18	5	1	1	1	5	15	2
Median	Top 5	5.5	1	1	23	5	1	3	2	2	15	2
Median	Top positive	6.9	3	3	11	9	6	10	2	7	5	13
IRWM	Top overall	7	8	10	7	9	6	7	7	7	7	2
IRWM	Top 2	7.1	11	11	3	3	6	10	7	7	7	6
Median	Top overall	7.7	5	3	18	16	1	4	6	7	15	2
IRWM	Top 3	8.3	11	11	3	9	6	10	7	13	7	6
Mean	Top	9.5	22	11	1	1	14	7	19	7	7	6
Mean	Top 2	10.5	11	11	7	9	6	15	7	13	15	11
IRWM	Top 4	10.9	11	11	11	9	17	10	7	13	7	13
Mean	Top overall	12.2	11	9	18	25	6	7	7	19	7	13
IRWM	Top 5	12.3	11	11	11	17	18	10	7	13	7	18
Mean	Top 3	13.8	11	11	18	17	18	17	7	13	15	11
Mean	Top 4	15.7	8	11	11	19	18	15	23	19	15	18
SFS	SLR	17	11	2	7	5	18	40	35	7	1	44
Stepwise regression	Analogy – 5NN	17.1	22	21	18	19	23	20	7	13	2	26
Mean	Top 5	17.3	11	11	23	25	6	17	21	19	15	25
Mean	Top positive	20.4	5	23	11	19	36	38	25	24	5	18
Stepwise regression	Analogy – 1NN	20.9	22	21	23	24	23	20	7	19	15	35
SFS	SWR	21	25	23	7	19	18	40	47	6	3	22
IRWM	Top positive	22.9	26	23	29	25	23	20	25	25	15	18
SFS	Analogy – 1NN	25.5	37	23	28	25	23	20	23	25	25	26
Natural logarithm	Analogy – 5NN	27.4	37	23	23	19	41	19	19	39	32	22
None	Analogy – 5NN	27.8	27	23	32	34	36	20	25	30	25	26
Normalization	Analogy – 5NN	27.8	27	23	32	34	36	20	25	30	25	26
None	Analogy – 1NN	27.9	27	31	29	25	31	35	25	25	25	26
Normalization	Analogy – 1NN	27.9	27	31	29	25	31	35	25	25	25	26
Normalization	CART (yes)	30.8	31	33	34	39	23	20	25	35	32	36
Normalization	CART (no)	30.8	31	33	34	39	23	20	25	35	32	36
EW – 5 bins	CART (yes)	30.8	39	56	11	9	14	54	38	49	25	13
EW – 5 bins	CART (no)	30.8	39	56	11	9	14	54	38	49	25	13
None	CART (yes)	31.3	31	33	34	36	23	20	25	35	38	38
None	CART (no)	31.3	31	33	34	36	23	20	25	35	38	38
SFS	Analogy – 5NN	31.9	59	51	46	25	31	20	7	19	35	26
Natural logarithm	CART (yes)	33.5	31	33	34	39	31	20	35	39	35	38
Natural logarithm	CART (no)	33.5	31	33	34	39	31	20	35	39	35	38
EW – 5 bins	Analogy – 5NN	34.2	8	11	51	39	62	39	21	45	44	22
Stepwise regression	CART (yes)	37.7	41	48	34	32	55	20	41	30	50	26
Stepwise regression	CART (no)	37.7	41	48	34	32	55	20	41	30	50	26
EW – 3 bins	Analogy – 5NN	37.9	11	23	51	54	42	43	40	25	48	42
Stepwise regression	SLR	39.2	46	33	23	36	42	44	48	30	46	44
None	SLR	44.9	49	42	43	46	42	44	50	45	38	50
PCA	Analogy – 5NN	45.4	46	40	62	45	48	35	41	51	44	42
Stepwise regression	SWR	46.8	44	41	46	48	45	44	49	54	50	47
Normalization	SLR	46.8	53	44	42	46	48	44	50	45	38	58
SFS	CART (yes)	47.3	41	44	54	39	51	44	57	39	57	47
PCA	Analogy – 1NN	47.4	46	43	51	48	45	44	54	48	48	47
PCA	SWR	48.3	44	51	46	48	48	56	57	43	46	44
EW – 3 bins	CART (yes)	50.3	56	44	43	48	51	59	54	58	38	52

Continued on next page

Pre-processing Option	Learner	Av. Rank	Run Rank 1	2	3	4	5	6	7	8	9	10
EW – 3 bins	CART (no)	50.3	56	44	43	48	51	59	54	58	38	52
SFS	CART (no)	51	62	48	62	53	45	42	50	43	55	50
Normalization	SWR	52.7	51	51	46	56	57	56	57	51	50	52
None	SWR	52.7	51	51	46	56	57	56	57	51	50	52
Natural logarithm	Analogy – 1NN	53.4	59	51	57	54	51	50	50	54	56	52
PCA	CART (yes)	54.7	53	62	57	56	39	65	41	54	57	63
PCA	CART (no)	54.7	53	62	57	56	39	65	41	54	57	63
Normalization	PLSR	58	49	58	54	62	57	59	61	60	61	59
EW – 3 bins	SWR	59.1	56	59	54	62	57	63	65	60	63	52
EF – 5 bins	Analogy – 5NN	60.1	66	59	66	56	70	50	46	63	63	62
EW – 5 bins	SWR	61.1	59	61	57	62	63	63	65	60	62	59
EF – 5 bins	CART (yes)	63.9	64	65	66	66	67	50	67	64	67	63
EF – 5 bins	CART (no)	63.9	64	65	66	66	67	50	67	64	67	63
EW – 5 bins	PLSR	64.5	62	64	64	65	64	68	70	64	63	61
EF – 3 bins	CART (yes)	66.4	66	65	71	66	65	72	63	64	69	63
EF – 3 bins	CART (no)	66.4	66	65	71	66	65	72	63	64	69	63
SFS	PLSR	67.7	84	82	57	56	57	59	61	83	57	81
EF – 3 bins	Analogy – 5NN	67.9	66	65	69	71	71	69	67	69	69	63
EW – 5 bins	Analogy – 1NN	68.3	70	70	65	70	67	65	71	69	66	70
EF – 5 bins	Analogy – 1NN	70.8	70	70	71	71	71	71	71	71	72	70
EW – 3 bins	PLSR	71.8	73	73	69	71	71	69	73	73	73	73
EF – 3 bins	Analogy – 1NN	72.8	72	72	74	74	74	72	73	72	73	72
Stepwise regression	Neural net	74.6	74	74	75	75	75	75	75	74	75	74
None	Neural net	76.1	75	76	76	76	77	77	76	77	76	75
PCA	Neural net	76.1	75	76	76	76	77	77	76	77	76	75
EW – 3 bins	Analogy – 1NN	76.4	75	79	78	76	76	76	78	75	76	75
Normalization	PCR	79	79	79	78	79	79	79	80	79	80	78
EW – 5 bins	PCR	80	81	78	81	80	79	80	79	81	82	79
EW – 3 bins	SLR	80.5	80	81	80	81	81	80	81	80	81	80
SFS	Neural net	80.6	75	75	83	84	82	84	84	75	79	85
PCA	SLR	82.6	82	85	83	82	83	82	83	82	83	81
EW – 5 bins	SLR	84.3	86	83	83	83	83	84	88	85	85	83
Natural logarithm	SLR	85	83	84	82	85	83	89	89	85	86	84
SFS	PCR	85.7	92	91	86	87	83	82	82	83	86	85
Stepwise regression	PLSR	86.1	86	86	86	87	87	86	84	87	84	88
PCA	PLSR	86.8	86	87	88	87	88	87	84	87	86	88
None	PLSR	86.8	86	87	88	87	88	87	84	87	86	88
Natural logarithm	SWR	87.8	85	87	88	85	88	90	92	87	91	85
Natural logarithm	PLSR	91.2	90	92	92	92	92	90	91	91	91	91
Natural logarithm	PCR	92.5	92	92	92	92	92	90	92	92	91	100
Stepwise regression	PCR	92.9	92	92	96	94	92	93	92	92	94	92
PCA	PCR	93.3	95	92	97	94	92	93	92	92	94	92
None	PCR	93.3	95	92	97	94	92	93	92	92	94	92
EW – 3 bins	PCR	94.1	101	87	91	91	91	101	89	101	90	99
EF – 5 bins	SLR	94.7	91	102	92	94	92	93	92	92	101	98
EF – 3 bins	PLSR	97.9	98	98	99	98	98	98	98	98	99	95
EF – 3 bins	SWR	98.4	95	98	99	100	100	101	100	98	98	95
EF – 5 bins	PCR	98.5	104	97	92	102	101	93	100	102	94	100
EF – 5 bins	SWR	98.6	98	98	102	98	98	98	100	98	101	95
EF – 5 bins	PLSR	99.5	98	98	102	100	101	100	99	97	100	100
Natural logarithm	Neural net	101.1	101	102	99	102	101	103	100	102	101	100
EF – 5 bins	Neural net	101.4	101	102	102	102	101	103	100	102	101	100
EF – 3 bins	PCR	102	104	102	105	102	101	103	100	102	101	100
EF – 3 bins	SLR	102	104	102	105	102	101	103	100	102	101	100
EF – 3 bins	Neural net	102	104	102	105	102	101	103	100	102	101	100

Continued on next page

Pre-processing Option	Learner	Av. Rank	Run Rank 1	2	3	4	5	6	7	8	9	10
EW – 5 bins	Neural net	102	104	102	105	102	101	103	100	102	101	100
Normalization	Neural net	102	104	102	105	102	101	103	100	102	101	100
EW – 3 bins	Neural net	102	104	102	105	102	101	103	100	102	101	100

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression
EF–Equal frequency **EW**–Equal width

Table N.6: **Wins−Losses** ranking for ensemble and solo classifiers, over 10 runs of *Bagging 2N*$_2$. Ensembles are highlighted gray.

Pre-processing Option	**Learner**	**Av. Rank**	**Run Rank**									
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Median	Top 2	4.7	5	2	5	1	13	4	2	2	4	9
Median	Top 3	4.7	3	2	5	4	1	2	2	2	17	9
Median	Top 4	4.8	2	2	13	5	1	1	1	5	17	1
Median	Top 5	5.1	1	1	18	5	1	3	2	2	17	1
Median	Top	5.3	16	2	1	7	1	4	18	1	2	1
IRWM	Top overall	5.9	7	9	5	7	6	7	7	6	4	1
Median	Top positive	6	3	2	8	7	6	10	2	6	3	13
IRWM	Top 2	6.4	9	10	3	3	6	10	7	6	4	6
Median	Top overall	6.9	6	2	13	13	1	4	6	6	17	1
IRWM	Top 3	7.3	9	10	3	7	6	10	7	11	4	6
Mean	Top	8.2	16	10	1	1	13	7	18	6	4	6
IRWM	Top 4	9.4	9	10	8	7	15	10	7	11	4	13
Mean	Top 2	9.4	9	10	8	7	6	15	7	11	12	9
IRWM	Top 5	10.4	9	10	8	13	16	10	7	11	4	16
Mean	Top overall	10.9	16	8	13	19	6	7	7	16	4	13
Mean	Top 3	11.7	9	10	13	13	16	17	7	11	12	9
Mean	Top 4	14.1	7	10	12	16	16	15	21	16	12	16
Mean	Top 5	15.1	9	10	18	25	6	17	20	16	12	18
Stepwise regression	Analogy – 5NN	16.9	16	19	13	17	25	31	7	16	1	24
Stepwise regression	Analogy – 1NN	19.8	20	19	21	19	25	20	7	20	17	30
SFS	Analogy – 1NN	21	27	24	21	19	19	20	21	22	17	20
None	Analogy – 1NN	22.6	21	24	23	19	25	31	24	22	17	20
Normalization	Analogy – 1NN	22.6	21	24	23	19	25	31	24	22	17	20
IRWM	Top positive	24	25	19	25	26	19	31	32	27	12	24
None	Analogy – 5NN	24.7	21	19	27	27	32	31	24	25	17	24
Normalization	Analogy – 5NN	24.7	21	19	27	27	32	31	24	25	17	24
Natural logarithm	Analogy – 5NN	25.3	26	27	18	17	36	19	21	37	34	18
Normalization	CART (yes)	26.7	28	28	29	33	19	20	24	28	27	31
Normalization	CART (no)	26.7	28	28	29	33	19	20	24	28	27	31
SFS	Analogy – 5NN	27	46	38	37	19	29	20	7	20	34	20
None	CART (yes)	27.2	28	28	29	31	19	20	24	28	32	33
None	CART (no)	27.2	28	28	29	31	19	20	24	28	32	33
Natural logarithm	CART (yes)	30	28	28	29	35	29	20	33	35	30	33
Natural logarithm	CART (no)	30	28	28	29	35	29	20	33	35	30	33
Stepwise regression	CART (yes)	32.4	37	35	29	27	42	20	38	32	40	24
Stepwise regression	CART (no)	32.4	37	35	29	27	42	20	38	32	40	24
Mean	Top positive	34.5	28	38	26	37	39	39	35	39	27	37
PCA	Analogy – 5NN	38.4	35	34	47	38	38	37	38	43	36	38
PCA	Analogy – 1NN	38.6	35	35	40	39	37	43	42	38	37	40
PCA	CART (yes)	42.2	40	47	44	44	34	47	36	40	43	47
PCA	CART (no)	42.2	40	47	44	44	34	47	36	40	43	47
Natural logarithm	Analogy – 1NN	42.6	47	38	46	44	40	39	42	45	46	39
SFS	CART (yes)	43	39	44	48	43	44	45	49	32	42	44
SFS	CART (no)	44.4	48	38	52	49	41	38	50	46	38	44
SFS	SWR	44.4	44	49	38	47	44	50	44	42	43	43
SFS	SLR	45.6	44	38	38	40	48	50	54	47	38	59
EF – 5 bins	Analogy – 5NN	48	52	44	54	47	53	44	41	48	47	50
EW – 3 bins	Analogy – 5NN	48.2	42	43	48	59	52	47	45	43	50	53
EF – 5 bins	CART (yes)	50	49	51	50	51	49	39	52	53	56	50
EF – 5 bins	CART (no)	50	49	51	50	51	49	39	52	53	56	50
EF – 3 bins	Analogy – 5NN	51.3	49	51	52	54	51	55	51	50	51	49
EW – 5 bins	CART (yes)	51.6	56	66	40	41	46	58	56	64	48	41

Continued on next page

Pre-processing Option	Learner	Av. Rank	Run Rank									
			1	2	3	4	5	6	7	8	9	10
EW – 5 bins	CART (no)	51.6	56	66	40	41	46	58	56	64	48	41
Stepwise regression	SLR	51.7	53	50	43	50	53	53	59	49	52	55
EW – 5 bins	Analogy – 5NN	53	43	46	68	55	69	46	48	57	52	46
EF – 5 bins	Analogy – 1NN	55.1	54	57	55	53	57	52	55	55	60	53
EF – 3 bins	CART (yes)	56.1	65	55	65	55	53	63	46	50	52	57
EF – 3 bins	CART (no)	56.1	65	55	65	55	53	63	46	50	52	57
Stepwise regression	SWR	56.6	56	51	56	58	59	53	59	60	58	56
EF – 3 bins	Analogy – 1NN	58.7	55	63	56	59	57	63	56	57	58	63
None	SLR	60.6	63	58	59	61	60	56	63	59	63	64
PCA	SWR	60.7	59	64	59	61	61	58	65	56	65	59
Normalization	SWR	61.7	60	59	59	67	63	58	61	62	67	61
None	SWR	61.7	60	59	59	67	63	58	61	62	67	61
Normalization	SLR	62.4	64	64	58	64	62	56	65	61	63	67
EW – 3 bins	CART (yes)	64.6	67	59	63	65	66	68	67	66	61	64
EW – 3 bins	CART (no)	64.6	67	59	63	65	66	68	67	66	61	64
Normalization	PLSR	67.5	62	68	69	69	68	67	67	68	69	68
EW – 3 bins	SWR	70.1	69	69	70	71	70	70	71	70	72	69
EW – 5 bins	SWR	70.1	70	70	71	70	71	70	70	69	70	70
EW – 5 bins	Analogy – 1NN	71.6	72	71	72	72	72	72	72	71	70	72
SFS	PLSR	72.3	85	83	67	63	63	66	63	85	66	82
EW – 5 bins	PLSR	72.4	71	72	73	73	73	73	73	72	73	71
EW – 3 bins	PLSR	73.6	73	73	74	74	74	74	74	73	74	73
Stepwise regression	Neural net	75.6	74	74	77	75	75	78	77	75	77	74
None	Neural net	75.7	76	75	75	77	77	75	75	77	75	75
PCA	Neural net	75.7	76	75	75	77	77	75	75	77	75	75
EW – 3 bins	Analogy – 1NN	76.6	75	78	78	76	75	77	78	74	78	77
Normalization	PCR	79.1	79	80	79	79	79	79	80	79	79	78
EW – 5 bins	PCR	80	81	78	81	80	80	80	79	80	82	79
EW – 3 bins	SLR	80.6	80	81	80	81	81	80	81	81	81	80
SFS	Neural net	82	78	77	85	86	82	84	88	75	79	86
Natural logarithm	SLR	82.9	81	82	81	83	83	85	86	83	83	82
PCA	SLR	83.3	83	86	84	82	86	82	83	82	84	81
EW – 5 bins	SLR	83.8	86	84	83	83	84	83	83	84	86	82
Natural logarithm	SWR	85.8	84	86	85	83	85	86	90	85	89	85
EW – 3 bins	PCR	87.7	92	84	85	86	87	92	85	93	84	89
Natural logarithm	PLSR	87.7	87	88	89	88	89	86	87	88	87	88
SFS	PCR	88.5	95	91	88	92	87	86	82	87	91	86
Natural logarithm	PCR	90.2	91	90	89	89	92	86	90	90	89	96
Stepwise regression	PLSR	91.3	92	88	92	94	91	93	88	94	88	93
EF – 5 bins	SLR	91.4	88	98	89	92	89	90	90	89	99	90
PCA	PLSR	91.5	88	92	94	89	92	94	93	90	93	90
None	PLSR	91.5	88	92	94	89	92	94	93	90	93	90
EF – 5 bins	PCR	94.8	95	92	92	95	101	91	99	96	91	96
EF – 5 bins	PLSR	96.4	94	95	96	95	101	96	95	95	95	102
PCA	PCR	96.8	101	95	102	97	95	97	95	96	96	94
None	PCR	96.8	101	95	102	97	95	97	95	96	96	94
EF – 3 bins	PCR	98.3	97	100	98	97	97	101	99	99	99	96
Stepwise regression	PCR	99.2	97	98	102	102	97	101	98	99	99	99
EF – 5 bins	SWR	99.7	101	102	96	102	97	97	99	102	102	99
EF – 3 bins	PLSR	99.9	100	102	100	100	97	97	103	102	96	102
EF – 3 bins	SLR	100	97	100	99	100	101	103	99	99	103	99
EF – 3 bins	SWR	103.5	104	102	101	104	104	104	104	104	104	104
Natural logarithm	Neural net	105	105	105	105	105	105	105	105	105	105	105
EF – 5 bins	Neural net	105.2	105	106	106	105	105	105	105	105	105	105
EF – 3 bins	Neural net	105.5	107	106	107	105	105	105	105	105	105	105

Continued on next page

Pre-processing	**Learner**	**Av.**	**Run Rank**									
Option		**Rank**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EW – 5 bins	Neural net	105.5	107	106	107	105	105	105	105	105	105	105
Normalization	Neural net	105.9	107	106	107	105	105	105	105	105	105	109
EW – 3 bins	Neural net	105.9	107	106	107	105	105	105	105	105	105	109

PCA–Principal component analysis **PCR**–Principal component regression **SFS**–Sequential forward selection
PLSR–Partial least squares regression

O
Graphical Visualization Of Ranking–Bagging $2N_2$

A graphical visualization of classifier ranking for the **Wins** and **Wins−Losses** ranking systems for *Bagging* $2N_2$, is provided in this appendix. Figures O.1 and O.3 display solo classifier performance over all 10 runs for **Wins** and **Wins−Losses** respectively. Figures O.2 and O.4 display classifier performance, solo and ensemble, over all 10 runs for **Wins** and **Wins−Losses** respectively.

All four figures use the same color coding to represent classifier ranking: Classifiers ranked in the top third in a run are coloured gray, classifiers ranked in the middle third are coloured white, and classifiers ranked in the bottom third are coloured black.

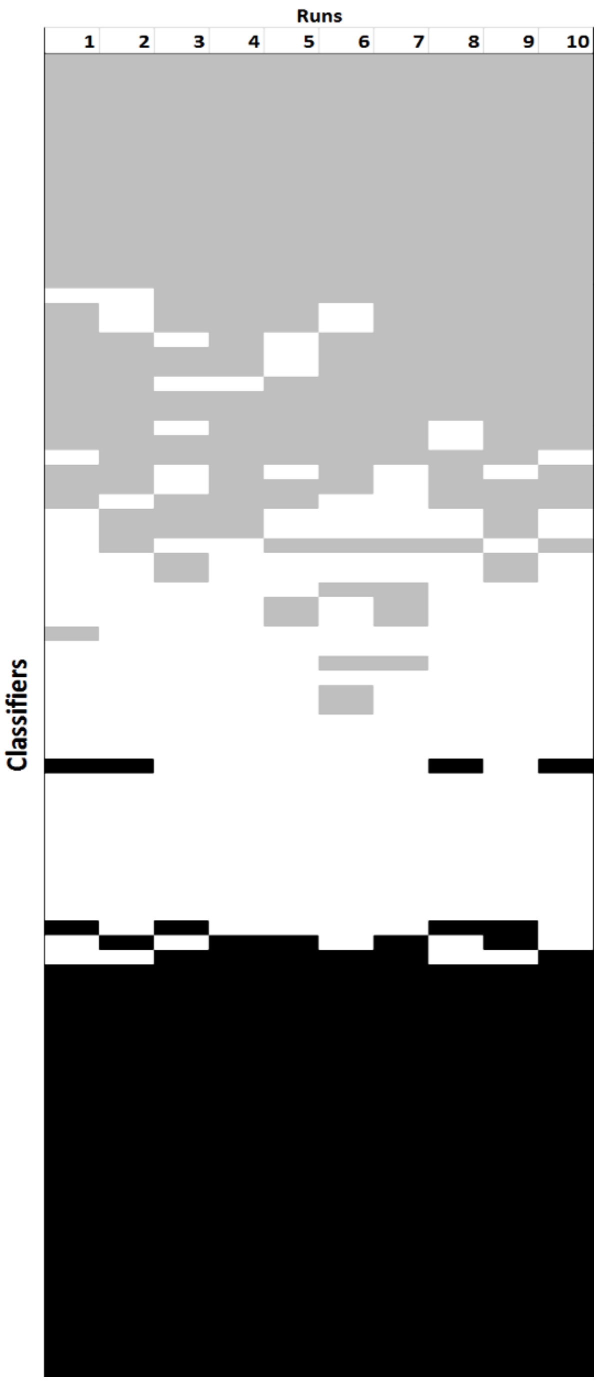


Figure O.1: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins** ranking.

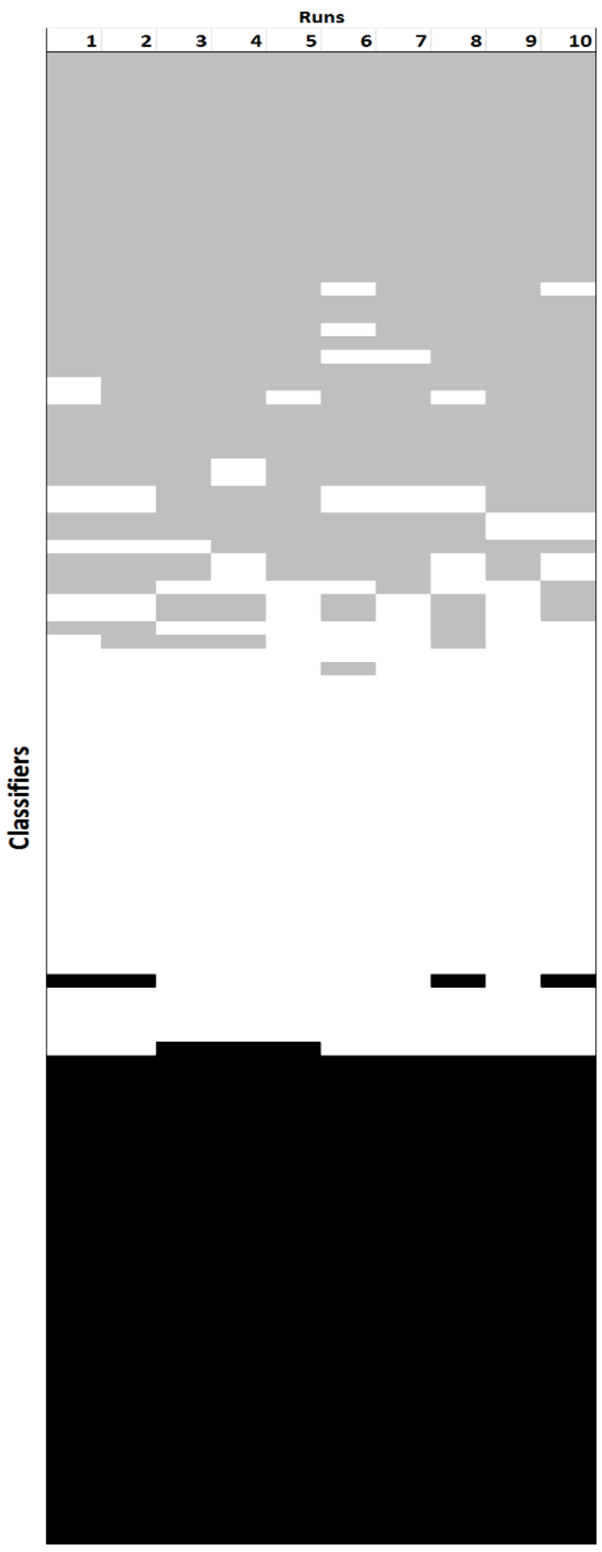


Figure O.2: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins** ranking.

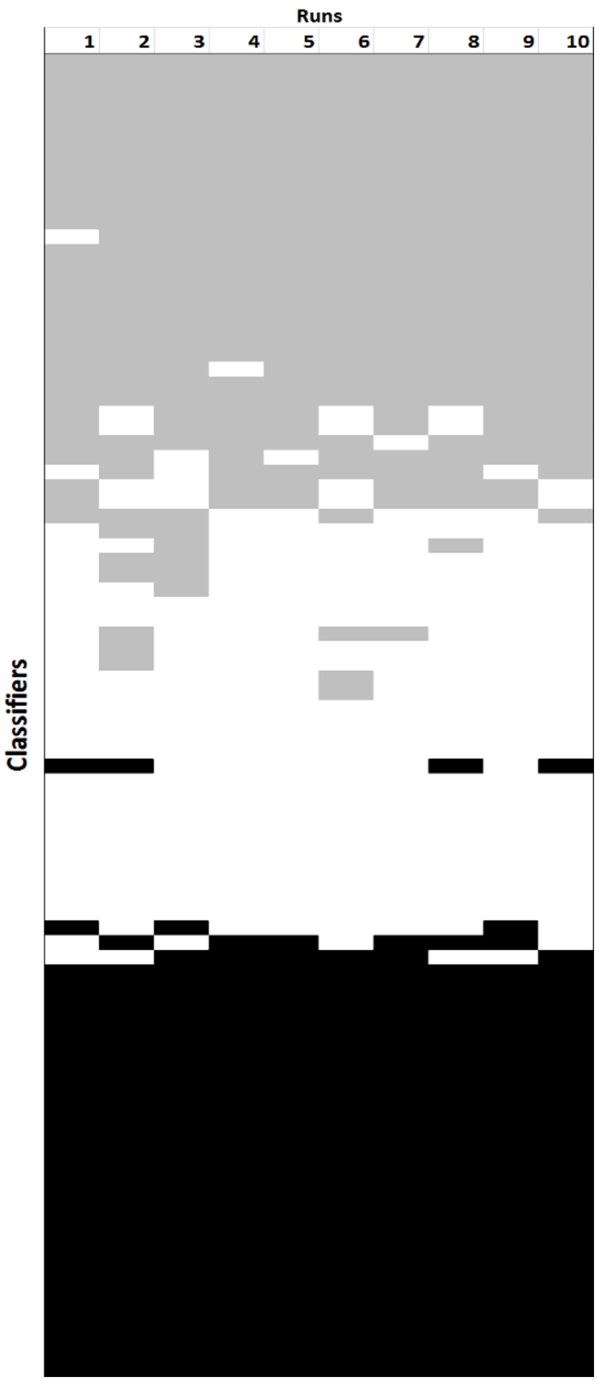


Figure O.3: A graphical visualization of solo classifier performance. The classifiers are arranged in order of increasing average **Wins−Losses** ranking.

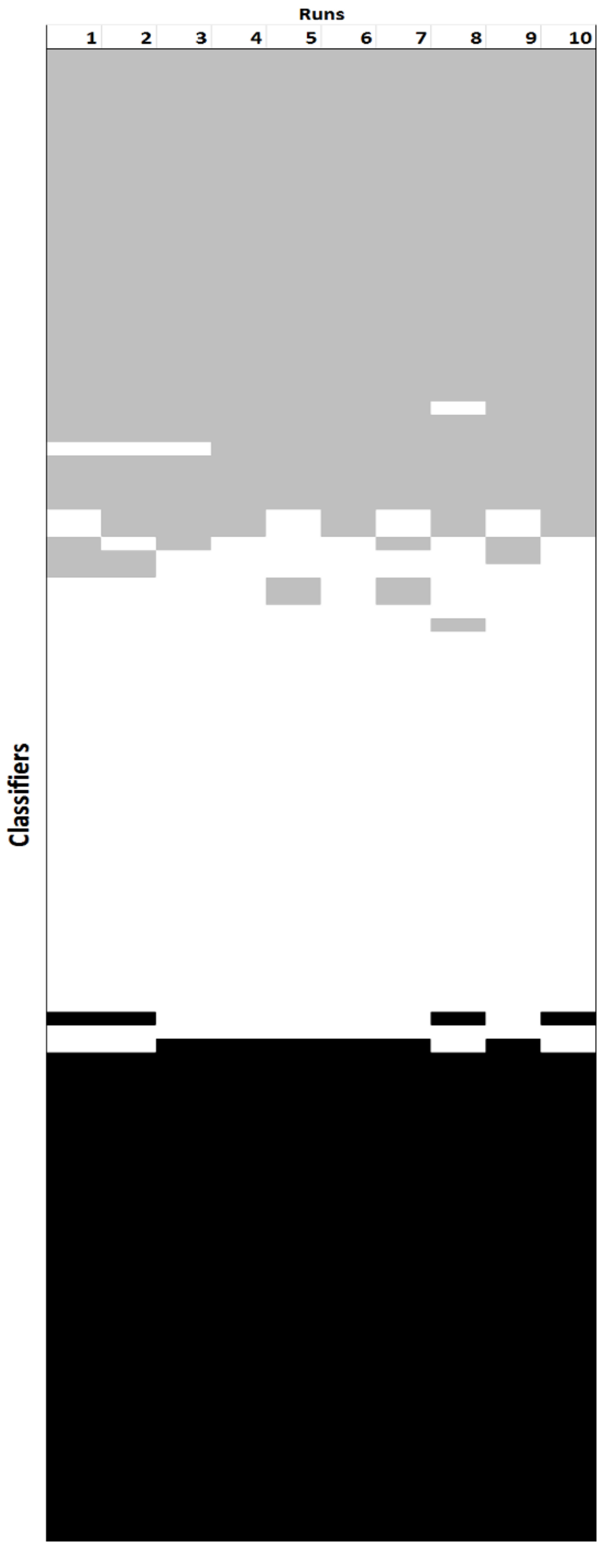


Figure O.4: A graphical visualization of classifier performance, solo and ensemble. The classifiers are arranged in order of increasing average **Wins**−**Losses** ranking.

Bibliography

[1] Abran, A. *Data Collection and Industry Standards: The ISBSG Repository.* John Wiley & Sons, Inc, 2015, pp. 161–184.

[2] Agusa, K. Software engineering evolution. In *Software Evolution, 2004. Proceedings. 7th International Workshop on Principles of* (Kyoto, Japan, September 2004), IWPSE '04, pp. 3–8.

[3] Azhar, D., Mendes, E., and Riddle, P. A systematic review of web resource estimation. In *Proceedings of the 8th International Conference on Predictive Models in Software Engineering* (Lund, Sweden, 2012), PROMISE '12, ACM, pp. 49–58.

[4] Azhar, D., Riddle, P., Mendes, E., Mittas, N., and Angelis, L. Using ensembles for web effort estimation. In *Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium on* (Baltimore, MD, USA, Oct 2013), ESEM 2013, IEEE, pp. 173–182.

[5] Boehm, B., and Valerdi, R. Achievements and challenges in cocomo-based software resource estimation. *Software, IEEE 25*, 5 (September 2008), 74–83.

[6] Breiman, L. Bagging predictors. *Machine Learning 24*, 2 (1996), 123–140.

[7] Brown, G., Wyatt, J., Harris, R., and Yao, X. Diversity creation methods: a survey and categorisation. *Information Fusion 6*, 1 (2005), 5–20.

[8] Brown, G., Wyatt, J. L., and Tiňo, P. Managing diversity in regression ensembles. *J. Mach. Learn. Res. 6* (Dec. 2005), 1621–1650.

[9] Carver, J. C. Towards reporting guidelines for experimental replications: A proposal. In *Proceedings of the 1st International Workshop on Replication in Empirical Software Engineering Research.* (Cape Town, South Africa, May 2010), RESER 2010, ACM.

[10] Conte, S. D., Dunsmore, H. E., and Shen, V. Y. *Software engineering metrics and models.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.

[11] Corazza, A., Martino, S., Ferrucci, F., Gravino, C., Sarro, F., and Mendes, E. Using tabu search to configure support vector regression for effort estimation. *Empirical Software Engineering 18*, 3 (2013), 506–546.

[12] Dietterich, T. Ensemble methods in machine learning. In *Multiple Classifier Systems*, vol. 1857 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 2000, pp. 1–15.

[13] Dietterich, T. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning 40*, 2 (2000), 139–157.

[14] Fenton, N., Marsh, W., Neil, M., Cates, P., Forey, S., and Tailor, M. Making resource decisions for software projects. In *Software Engineering, 2004. Proceedings. 26th International Conference on* (Edinburgh, Scotland, May 2004), ICSE 2004, IEEE, pp. 397–406.

[15] Goebel, M. *Ensemble Learning By Data Resampling.* The University of Auckland, 2004.

[16] Goebel, M., Riddle, P. J., and Barley, M. A unified decomposition of ensemble loss for predicting ensemble performance. In *Proceedings of the Nineteenth International Conference on Machine Learning* (Sydney, NSW, Australia, 2002), ICML '02, Morgan Kaufmann Publishers Inc., pp. 211–218.

[17] Kitchenham, B., and Charters, S. Guidelines for performing systematic literature reviews in software engineering. Tech. rep., Software Engineering Group, School of Computer Science and Mathematics, Keele University, July 2007. Version 2.3.

[18] Kitchenham, B., Mendes, E., and Travassos, G. H. A systematic review of cross-vs. within-company cost estimation studies. In *Evaluation and Assessment in Software, 2006. Proceedings. 10th International Conference on* (UK, 2006), EASE '06, British Computer Society, pp. 81–90.

[19] Kitchenham, B., Pickard, L., MacDonell, S., and Shepperd, M. What accuracy statistics really measure [software estimation]. *Software, IEE Proceedings - 148*, 3 (June 2001), 81–85.

[20] Kocaguneli, E., and Menzies, T. Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software 86*, 7 (2013), 1879 – 1890.

[21] Kocaguneli, E., Menzies, T., and Keung, J. On the value of ensemble effort estimation. *Software Engineering, IEEE Transactions on 38*, 6 (2012), 1403–1416.

[22] Krogh, A., and Vedelsby, J. Neural network ensembles, cross validation and active learning. In *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. MIT Press, Cambridge, CA, 1995, pp. 231–238.

[23] Melville, P. Creating diverse ensemble classifiers. Tech. rep., The University of Texas at Austin, Nov 2003.

[24] Mendes, E. A comparison of techniques for web effort estimation. In *Proceedings of the First International Symposium on Empircal Software Engineering and Measurement* (Madrid, Spain, September 2007), ESEM 2007, IEEE, pp. 334–343.

[25] Mendes, E. Predicting web development effort using a bayesian network. In *Evaluation and Assessment in Software, 2007. Proceedings. 11th International Conference on* (UK, April 2007), EASE '07, British Computer Society, pp. 83–93.

[26] Mendes, E. Web cost estimation and productivity benchmarking. In *Software Engineering*, A. Lucia and F. Ferrucci, Eds., vol. 5413 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, pp. 194–222.

[27] Mendes, E., Di Martino, S., Ferrucci, F., and Gravino, C. Effort estimation: How valuable is it for a web company to use a cross-company data set, compared to using its own single-company data set? In *Proceedings of the 16th International Conference on World Wide Web* (Banff, Alberta, Canada, 2007), WWW '07, ACM, pp. 963–972.

[28] Mendes, E., and Kitchenham, B. Further comparison of cross-company and within-company effort estimation models for web applications. In *Software Metrics, 2004. Proceedings. 10th International Symposium on* (Chicago, IL, USA, September 2004), METRICS 2004, IEEE, pp. 348–357.

[29] Mendes, E., Mosley, N., and Counsell, S. Investigating web size metrics for early web cost estimation. *J. Syst. Softw. 77*, 2 (August 2005), 157–172.

[30] Mendes, E., Mosley, N., and Counsell, S. The need for web engineering: An introduction. In *Web Engineering*, E. Mendes and N. Mosley, Eds. Springer Berlin Heidelberg, 2006, pp. 1–27.

[31] Mendes, E., Watson, I., Triggs, C., Mosley, N., and Counsell, S. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering 8*, 2 (2003), 163–196.

[32] Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., Peters, F., and Turhan, B. The promise repository of empiral software engineering data, 2012.

[33] Petticrew, M., and Roberts, H. *Systematic Reviews in the Social Sciences: A Practical Guide*. Blackwell Pub., 2006.

[34] Reifer, D. Web development: estimating quick-to-market software. *Software, IEEE 17*, 6 (Nov 2000), 57–64.

[35] Reifer, D. Estimating web development costs: There are differences. *Crosstalk, The Journal of Defense Software Engineering* (June 2013).

[36] Ruhe, M., Jeffery, R., and Wieczorek, I. Using web objects for estimating software development effort for web applications. In *Software Metrics Symposium, 2003. Proceedings. Ninth International* (Sydney, Australia, September 2003), METRICS 2003, IEEE, pp. 30–37.

[37] Russell, S. J., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 2 ed. Pearson Education, 2003.

[38] Shepperd, M., and MacDonell, S. Evaluating prediction systems in software project estimation. *Information and Software Technology 54*, 8 (2012), 820 – 827. Special Issue: Voice of the Editorial BoardSpecial Issue: Voice of the Editorial Board.

[39] Shull, F. J., Carver, J. C., Vegas, S., and Juristo, N. The role of replications in empirical software engineering. *Empirical Software Engineering. 13*, 2 (April 2008), 211–218.

[40] Umbers, P., and Miles, G. Resource estimation for web applications. In *Software Metrics, 2004. Proceedings. 10th International Symposium on* (Chicago, IL, USA, September 2004), METRICS 2004, IEEE, pp. 370–381.

[41] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. *Experimentation in Software Engineering*, 1 ed. Springer-Verlag Berlin Heidelberg, 2012.