Table de matières

Liste des figures

Liste des tableaux

Liste des acronymes

AB: Abstraction Based

ANN: Artificial Neural Network

BH: Berringer-Hullermeier

BWS: Bouguessa-Wang-Sun

CVI: Cluster Validity index (Indice de validité des clusters)

DT: Decision Tree

FCM: Fuzzy C means

FHV: Fuzzy HyperVolume

FS: Fukuyama Sugeno

IB: Instance Based

KNN: K nearest neighbors (K voisins les plus proches)

NNC: Nearest Neighbor Classification

NPC: Normalized Partition Coefficient

PC: Partition Coefficient

PNN: Prototype nearest neighbor

RF: Random forest

SVM: Support Vector machine

TCl: Taux du classement

TRé: Taux de réduction

XB : Xie & Beni

Introduction générale

A l'issue de l'évolution du mode de consommation/production de données, dû principalement aux avancées technologiques réalisées dans la collecte, le stockage et le traitement des données (scanners, internet, base de données, entrepôts de données, XML etc.), *la fouille et la science de données* émergent comme de nouvelles disciplines d'analyse de données. L'objectif de ces disciplines est de fournir les processus, les méthodologies et les techniques adéquates pour gérer, analyser, explorer et rendre compréhensible les grandes masses de données accumulées, dans le but d'une prise de décisions.

Parmi les tâches importantes de la *fouille et la science des données*, on trouve la *prédiction*. Elle est utilisée, principalement, dans l'analyse prédictive et la reconnaissance de formes pour comprendre les données historiques disponibles, prédire les valeurs de certains attributs en vue de prendre des décisions, ou identifier et reconnaître automatiquement des objets implicites dans des environnements complexes. Dans le contexte de ce rapport, nous considérons la prédiction avec l'objectif d'estimer la valeur d'un attribut. Selon la nature de l'attribut à prédire, nous distinguons deux types de prédiction : la ***régression*** et le ***classement***. La régression s'associe aux problèmes où l'attribut cible prend ses valeurs dans un domaine de valeurs réelles, tandis que le classement met l'accent sur les problèmes où l'attribut cible prend des valeurs discrètes dans un ensemble de classes prédéfinies qu'on appelle *étiquettes*.

L'apprentissage automatique, notamment l'apprentissage *supervisé*, se situe au cœur des approches adoptées pour la résolution de la problématique de classement. Plusieurs techniques ont été proposées et qu'on peut classer en deux catégories : les techniques basées sur *l'abstraction* où il s'agit de construire un modèle bien structuré et généralisé à partir de l'ensemble d'apprentissage qui sera utilisé pour classer les nouvelles instances. Les techniques basées sur les *instances* où il s'agit de stocker tout l'ensemble d'apprentissage lors du classement d'un nouvel exemple ; parmi les méthodes de cette approche, on trouve la méthode des K plus proches voisins (K-NN) qui consiste à classer une nouvelle instance à partir des étiquettes des K instances les plus proches.

Malgré la popularité et le succès réalisés par K-NN, en raison de sa simplicité, sa facilité de mise en œuvre et la variété des distances utilisés, plusieurs faiblesses sont toujours présentes et qui sont liées; à l'*incapacité* à fournir des classements corrects dans des environnements complexes où les données d'apprentissage sont *incomplètes*, *imprécises*, *déséquilibrés* et *bruitées*; à la faible tolérance aux *incertitudes* dû au bruit induit par le choix

inapproprié du paramètre K et la technique de vote. Et enfin à *l'inefficacité* dû au fait que la taille du problème croît rapidement en fonction du nombre d'instances et du nombre d'attributs considérés. A cet égard, plusieurs améliorations de la méthode K-NN sont proposées :(1) la proposition de nouvelles mesures de distance pour évaluer la proximité de manière précise et significative. (2) l'apprentissage automatique de la valeur du paramètre K pour sélectionner les meilleurs voisins les plus proches. (3) l'intégration de la théorie des ensembles flous et de la logique floue dans le processus de classement par K-NN pour traiter les incertitudes et les imprécisions. (4) la réduction de l'ensemble d'apprentissage à un sous-ensemble d'instances significatives pour améliorer l'efficacité de K-NN en termes de temps d'exécution et de capacité de stockage.

Dans le travail présenté dans ce rapport, nous nous intéressons, en premier lieu à la réduction de l'ensemble d'apprentissage, et en deuxième lieu, à la gestion des incertitudes. Pour ce faire, nous faisons appel à l'algorithme de regroupement FCM (Fuzzy C-Means) pour générer un ensemble d'apprentissage réduit formés par des prototypes et à une mesure de similarité floue, basée sur une représentation floue des attributs, pour évaluer la proximité entre les instances. Ainsi, les objectifs de ce travail sont (1) la mise en œuvre de cette technique de réduction pour améliorer l'efficacité de K-NN et (2) l'implémentation de la solution proposée sous forme d'un prototype avec une interface graphique simple et conviviale. (3) la réalisation d'une étude expérimentale pour valider la solution proposée.

Le présent rapport est organisé en quatre chapitres :

Le premier chapitre présente le contexte général du projet. Ainsi, nous présentons l'organisme d'accueil, la problématique étudiée et la solution proposée.

Le deuxième chapitre concerne l'état de l'art sur le classement par la méthode des K plus proches voisins qui présente ses inconvénients ainsi que les voix possibles d'amélioration. L'accent sera mis les méthodes de réduction de l'ensemble d'apprentissage pour améliorer l'efficacité.

Le troisième chapitre développe la solution proposée. Il présente, en premier lieu les notions préliminaires utilisés dans notre solution, à savoir les ensembles flous, les variables linguistiques et le regroupement flou. En deuxième lieu, nous décrivons les différentes parties du processus de classement.

Le dernier chapitre, se focalise sur l'implémentation de la solution et l'étude expérimentale.

Le rapport termine par une conclusion générale et les principales perspectives du travail mené dans ce projet.

Chapitre 1
Cadre général du projet

Chapitre 1 :
Cadre général du projet

Introduction

Dans ce chapitre, nous allons donner une description succincte du contexte général de notre projet. Le lieu de stage, le laboratoire Systèmes Intelligents et Applications, est présenté en premier lieu. La problématique traitée dans ce projet est suite discutée. Nous terminons avec une description de la solution proposée et des objectifs du projet.

1. Présentation du lieu de stage

Le travail présenté dans ce rapport a été réalisé au sein du laboratoire Systèmes Intelligents et Applications SIA à la Faculté des Sciences et Techniques de Fès (FST). LSIA, crée en 2011, est une unité de Recherche du Centre d'Etudes Doctorales en Sciences et Techniques de l'Ingénieur domicilié à la Faculté des Sciences et Techniques de Fès et regroupant des laboratoires de recherche tous accrédités par l'Université Sidi Mohamed Ben Abdellah de Fès, et domiciliés à la Faculté des Sciences et Techniques, l'Ecole Supérieure de Technologie, la Faculté Polydisciplinaire de Taza, l'Ecole Nationale des Sciences Appliquées de Fès et l'ENS de Fès.

LSIA est composé de 16 enseignants-chercheurs du département d'Informatique de la FST de Fès et de 23 doctorants. Cette imbrication étroite entre enseignement et recherche, est un élément essentiel de la dynamique du laboratoire.

Les thématiques de recherche se situent au cœur des Sciences et Technologies de l'Information et de la Communication et s'articulent essentiellement autour des thématiques de recherche des enseignants chercheurs du laboratoire et assure une large couverture thématique présentant un atout très important pour le laboratoire.

Le LSIA est organisé en trois équipes :

- Systèmes de Communication et Traitement de Connaissances (SCTC),
- enVironnement Intelligents & Applications (VIA),
- Vision Artificielle & Systèmes Embarqués (VASE).

2. Problématique

La méthode des K plus proches voisins K-NN est une méthode de classement qui fait partie de la famille des méthodes de classement basées sur les instances. Ainsi, On peut caractériser K-NN par les éléments suivants :

- Un ensemble d'apprentissage contenant un ensemble d'instances déjà classées ;
- Une mesure de similarité qui permet mesurer la proximité entre deux instances afin de sélectionner les voisins les plus proches ;
- La valeur K qui représente le nombre de voisins à prendre en considération dans le classement d'une nouvelle instance.
- Une règle de vote qui permet de décider sur l'étiquette à affecter à la nouvelle instance.

Le classement d'une nouvelle instance se déroule selon les étapes suivantes :

- Calculer la distance entre la nouvelle instance et toutes les instances de l'ensemble d'apprentissage.
- Sélectionner les K plus proche voisins ; ceux ayant la plus petite distance avec la nouvelle instance
- Affecter une étiquette de classe à la nouvelle instance ; l'étiquette ayant obtenue la majorité de voix dans le vote par les K plus proches voisins sélectionnés.

K-NN est souvent utilisé comme référence dans les domaines de la fouille de données, l'apprentissage supervisé et la reconnaissance de formes. En effet, il est conté aujourd'hui parmi les dix algorithmes de la fouille de données et ceci pour les raisons suivantes :

- Aucune hypothèse n'est nécessaire sur la structure et la distribution des données d'apprentissage. Ceci rend bénéfique l'utilisation des données non-linéaire sans exigence sur leur régularité.
- Intuitif et simple à comprendre, facile à implémenter et les classements fournis sont facilement explicables aux utilisateurs.
- Ouvert à diverses mesures de distance ; l'algorithme est applicable avec plusieurs distances telles que les distances Euclidienne ; Hamming, Manhattan, etc.
- Evolution continuelle, dans le sens où il s'adapter facilement et rapidement les classements fournis aux changements temps-réel de l'ensemble d'apprentissage.
- Valables pour divers domaines d'application.
- Utile dans certaines tâches de la fouille de données comme le traitement des valeurs manquantes. En fait K-NN permet de prédire la valeur manquante à partir des données historiques.

Malgré les succès et la popularité réalisés par K-NN, il souffre encore d'un certain nombre d'inconvénients :

- K-NN réalise des performances non satisfaisantes, en termes de *précision* et de *tolérance aux imprécisions* et aux *incertitudes*, dans les environnements complexes caractérisés par des données de faible qualité. En effet, les données disponibles, dans les environnements réels, sont souvent :
    - *Incomplètes* dans le sens où la valeur de certains attributs ne sont pas disponibles. Cela requiert un traitement préalable qui pourrait influencer les performances de classements.
    - *Déséquilibrées* où la majorité des instances sont étiquetées dans une classe particulière. Cela pourrait aboutir à un classement erroné des instances des classes les moins courantes, du fait que le modèle de classement donnera plus de préférence à la classe majoritaire.
    - *Imprécises* du fait que, dans certains domaines tels que le diagnostic médical, les attributs sont mesurés par des valeurs linguistiques telles que *bas*, *moyen*, *élevé*, *très élevé*. Et aussi, l'*incertitude* induite par les limites brutales entre les classes, dans le sens où les instances sur la frontière entre deux classes ne sont pas certainement étiquetées.
- K-NN peut aboutir à un classement *erroné* et *incertain* en raison du bruit induit par le choix inapproprié du paramètre K et de la technique de vote permettant de dériver une classe pour une nouvelle instance.
- K-NN stocke toutes les instances de l'ensemble d'apprentissage pendant le classement d'une nouvelle instance. Ainsi, la croissance de la taille de l'ensemble d'apprentissage entraine une dégradation significative de *l'efficacité*, en termes de mémoire de stockage et de temps calcul. En effet, le besoin en quantité de mémoire pour le stockage de l'ensemble d'apprentissage, et du temps de calcul pour évaluer la similarité entre la nouvelle instance et les instances de l'ensemble d'apprentissage, croît avec la taille des données d'apprentissage.

Dans ce rapport, nous sommes concernés par la problématique de l'inefficacité et le traitement des imprécisions et des incertitudes dans le processus de classement par la méthode K-NN.

3. Solution proposée et objectifs du projet

Dans la littérature, on trouve plusieurs voies d'amélioration pour pallier aux faiblesses de K-NN en matière d'efficacité et de gestion d'incertitudes.

- La première voie concerne l'amélioration de l'efficacité en réduisant l'ensemble d'apprentissage à un sous-ensemble d'instances, appelées

prototypes ; ces prototypes serviront par la suite comme ensemble d'apprentissage dans le classement d'une nouvelle instance. Dans ce contexte, nous proposons d'utiliser l'algorithme de regroupement FCM, *Fuzzy C-Means*, pour classifier les instances d'apprentissage dans des clusters ; les centres des clusters obtenus serviront après comme prototypes dans le classement d'une nouvelle instance. Le nombre adéquat de prototypes sera choisi à l'aide d'un *indice de validité de clusters* qui permet d'estimer la qualité des partitions générées.

- La deuxième voie concerne le traitement des incertitudes et des imprécisions à l'aide de la théorie des ensembles flous et la logique floue. En effet, la logique floue a été utilisée de façon satisfaisante dans divers domaines d'application où le bruit, l'incertitude et l'imprécision sont inévitables. En fait, la logique floue se veut une approche parfaitement adéquate à la modélisation des connaissances, *imprécises*, *qualitative et vagues*. Dans notre travail, nous proposons de (1) représenter les attributs de prédiction par des variables linguistiques capables de prendre des valeurs qualitatives telles que *petit, moyen, grand, etc.* (2) d'utiliser une mesure de similarité floue, basée sur les variables linguistiques, pendant le classement d'une nouvelle instance.

Ainsi, nous pouvons regrouper les objectifs de ce travail en trois

- Etudier les méthodes de génération des prototypes, l'algorithme de regroupement FCM et la logique floue
- Implémenter, sous forme d'une application ergonomique et conviviale, la méthode de classement basée sur l'algorithme FCM pour la génération des prototypes et la similarité floue pour le classement.
- Réaliser une expérimentation à l'aide de l'application développée, sur les bases d'apprentissage les plus connues.

Conclusion

Dans ce chapitre, nous avons présenté le contexte général de notre projet de fin d'étude. Nous avons présenté le laboratoire d'accueils. Nous avons, ensuite, discuté les faiblisses de la méthode de classement K-NN, et identifié la problématique à traiter dans ce travail. Enfin, nous avons donné une description succincte de la solution à développer pour répondre à notre problématique. Dans le chapitre suivant, nous allons présenter un état de l'art sur le classement supervisé et ses techniques.

Chapitre 2
Etat de l'art

Chapitre 2 : Etat de l'art

Introduction

Dans ce chapitre, nous présentons un état de l'art sur le classement supéervisé et ses techniques. Nous commençons par le processus de classement supervisé. Nous décrivons ensuite les différentes méthodes de classement ; l'accent est mis sur la méthode K-NN et les voies d'amélioration possibles. Nous terminons par une revue des méthodes de réduction de l'ensemble d'apprentissage utilisées dans l'objetif d'améliorer l'éfficacité de K-NN.

1. Apprentissage supervisé

La *prédiction* est une tâche fondamentale dans le domaine de la fouille et la science des données. Elle est particulièrement utilisée, dans les domaines de l'analyse prédictive et la reconnaissance de formes, pour *comprendre* les données historiques disponibles, *prédire* les valeurs de certains attributs en vue de prendre des décisions, ou *identifier* et reconnaître automatiquement des objets implicites dans des environnements complexes. Dans certaines situations, la prédiction est utilisée dans des tâches comportant un aspect temporel, dans le sens où il s'agit de prédire ce qui se passera dans le futur. Dans le présent travail, la prédiction signifie tout simplement l'attribution d'une valeur à un attribut inconnu dans un problème donné.

La tâche de prédiction, considérée dans l'objectif d'estimer la valeur d'un attribut dans un problème donné, peut être formulée comme un problème de l'apprentissage supervisé [1] dans lequel :

- Les ***variables prédictives*** représentent les attributs *significatifs* et *indépendants* du problème considéré ;
- La ***variable cible*** correspond à l'attribut à prédire ;
- L'ensemble ***d'apprentissage*** décrit les instances résolues du problème avec un ensemble d'échantillons générés à partir de l'ensemble de données d'historique.

Selon la nature de la variable cible (l'attribut à prédire), il est utile de distinguer deux types de prédiction : la ***régression*** et le *classement*. La régression s'associe aux problèmes où la variable cible prend ses valeurs dans un domaine de valeurs réelles, tandis que le classement

met l'accent sur les problèmes où la variable cible prend des valeurs discrètes dans un ensemble d'étiquettes (classes) prédéfinies. Dans ce rapport, nous nous intéressons à la tâche de classement.

1.1. Formulation d'un problème de classement

Formellement, un problème de classement peut être défini par un triplet $D = \{F, C, TR\}$ où :

- $F = \{f_1, f_2, \dots, f_t\}$: un ensemble d'attributs *significatifs* et *indépendants*, qui correspond à l'ensemble des variables prédictives. Ces attributs sont généralement obtenus par un processus automatique, empirique ou manuel de caractérisation du domaine considéré.
- $C = \{C_1, \dots, C_M\}$: un ensemble d'étiquettes qui représentent les classes possibles avec les quelles la variable cible, Y, peut être affectée. Ces classes sont définies, préalablement, par les experts du domaine ou obtenus par des méthodes automatiques de classement.
- $TR = \{e_1, e_2, \dots, e_n\}$: un ensemble d'instances correctement étiquetés, dans le sens où chaque instance est décrite par une paire *attributs-classe*. Cet ensemble est construit à partir de l'historique du domaine dans l'objectif de servir comme ensemble d'apprentissage.

La question principale dans un problème de classement est de découvrir la relation cachée entre les attributs prédictifs de l'ensemble F et la variable cible, Y. La relation exhibée est représentée par une structure référencée par *modèle* de *classement.* C'est une fonction, construite à partir de l'ensemble TR, qui permet de mapper une nouvelle instance à une étiquette de classe prise dans l'ensemble C.

1.2. Processus de classement

La figure 1 illustre le processus générique suivi lors de la résolution d'un problème de classement. Il faut noter que ce processus suppose que l'ensemble d'apprentissage TR est le résultat de la phase de prétraitement. Le processus opère en deux principales étapes : dans la première étape, un algorithme d'apprentissage est appliqué dans l'objectif de construire un modèle de classement en explorant l'ensemble d'apprentissageTR. Dans la deuxième étape, le modèle de classement obtenu est ensuite utilisé pour estimer la classe d'une nouvelle instance sachant les valeurs des attributs prédictifs.

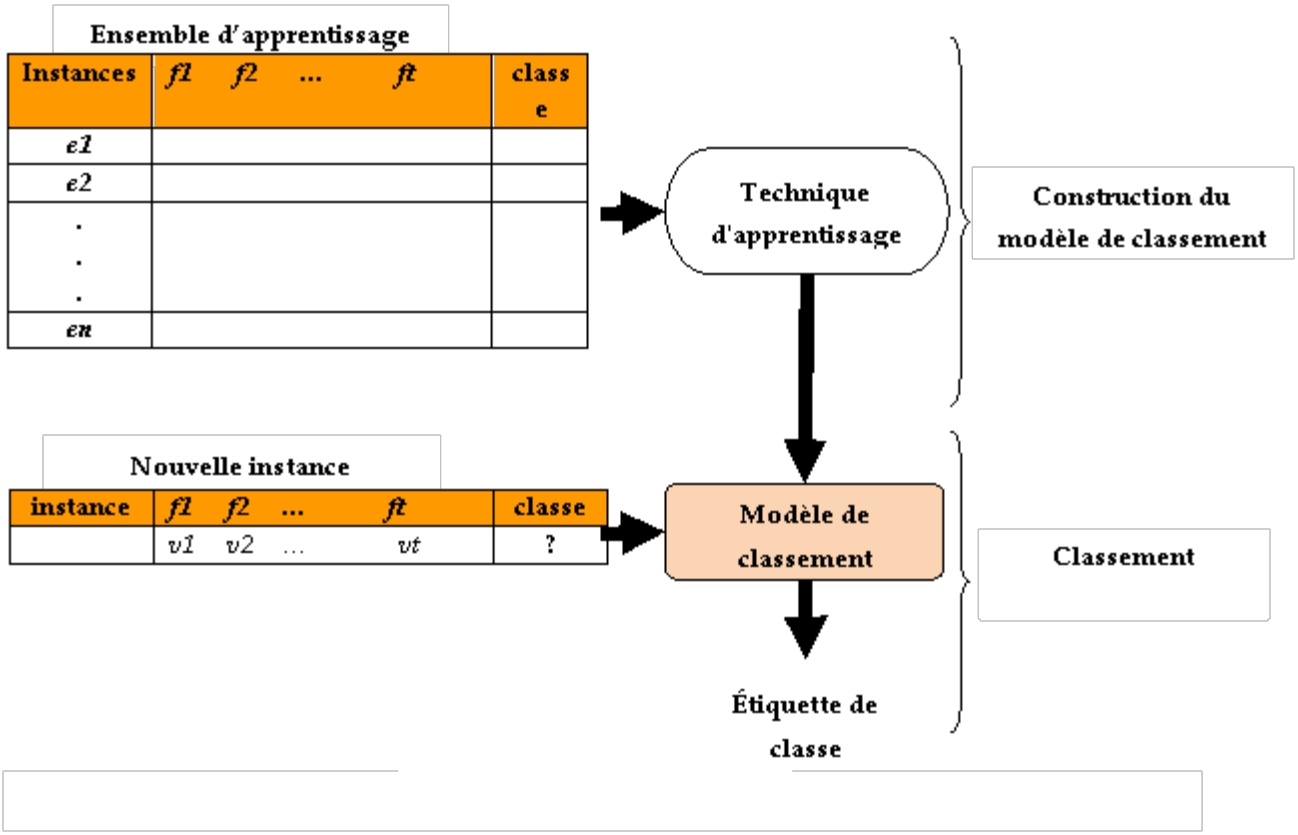

Figure 1 : Processus de prédiction

1.3. Techniques de classement

Plusieurs techniques de classement sont disponibles dans la littérature. Elles peuvent être classées en deux principales approches selon le type de la technique d'apprentissage utilisée : l'approche basée sur l'abstraction et l'approche basée sur les instances (figure 2).

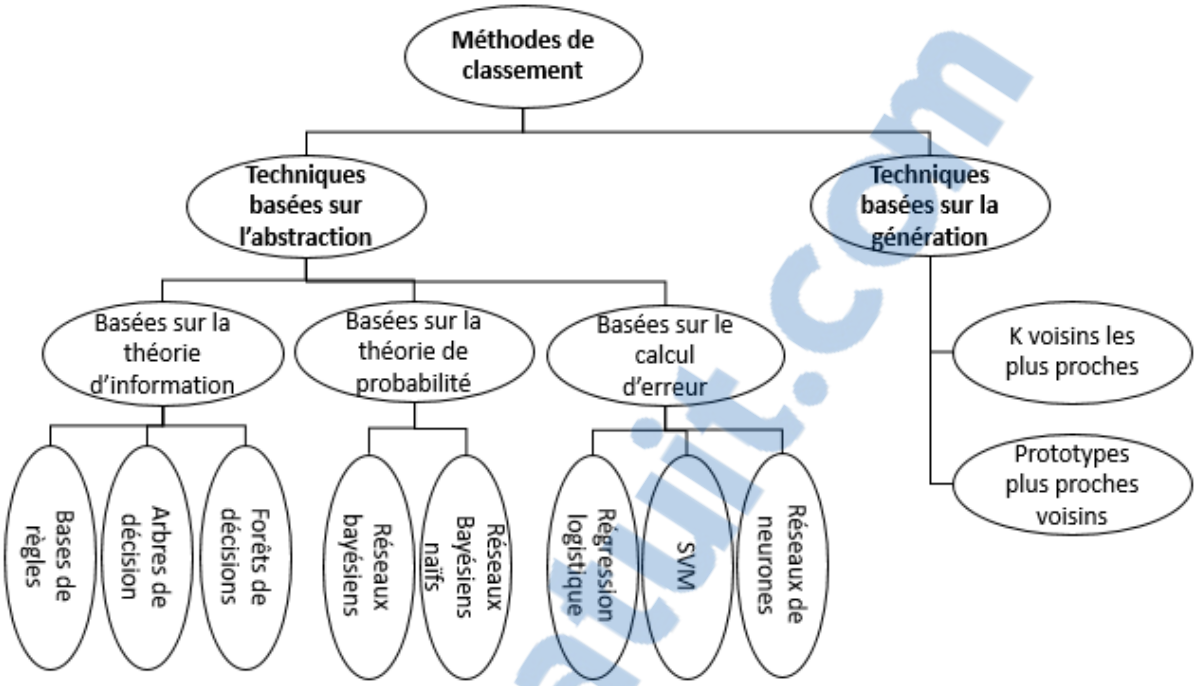

Figure 2 : Techniques de classement

1.3.1. Approche basée sur l'abstraction

Cette approche consiste à extraire les connaissances de classement à partir de l'ensemble d'apprentissage en construire un modèle bien structuré et généralisé. Le modèle résultant est ensuite stocké dans le but de l'utiliser pour classer de nouveaux échantillons ; les classes sont estimées utilisant un mécanisme d'inférence lié au modèle généré. Cette approche apparaît dans de nombreux modèles de classement qu'on peut regrouper en trois catégories, selon la méthode d'apprentissage utilisée :

- Méthodes basées sur la mesure de l'information, tel que les systèmes à base de règles, les arbres de décision [2] et les forêts aléatoires(RF) [3];
- Méthodes basées sur la théorie des probabilités comme les réseaux naïfs bayésiens et bayésiens [4];
- Méthodes basées sur le calcul d'erreurs, tels que la régression linéaire et la régression logistique, les réseaux de neurones artificiels (ANN) [5] et les machines à vecteurs de support (SVM) [6]

1.3.2. Approche basée sur les instances.

Cette approche semble convenir à la tâche de classement dans des domaines complexes où la qualité non satisfaisante des données d'apprentissage rend difficile l'obtention d'un modèle de classement généralisé acceptable. Par exemple, le diagnostic médical [7], l'estimation de l'effort logiciel [8][9], ou la reconnaissance de formes, où les attributs sont

souvent décrits par des valeurs linguistiques telles que *faible*, *moyenne* ou *élevée* et que les ensembles de données sont généralement incomplet et déséquilibré. La principale différence par rapport à l'approche basée sur l'abstraction est le stockage, pendant la phase de classement, d'un ensemble d'instances sélectionnées dans l'ensemble de données d'apprentissage [10]. Le processus de classement dans cette approche repose sur l'hypothèse « les instances similaires possèdent des classes similaire ». Ainsi, les techniques de classement résultantes requièrent les éléments suivants pendant le classement :

- Un ensemble *d'instances* représentant toutes les données d'apprentissage ou un sous ensemble sélectionné automatiquement par des algorithmes spécifiques ou manuellement par des experts du domaine.
- Une mesure de similarité qui calcule la similarité entre la nouvelle instance à classifier et les instances stockées. Le choix de la mesure de similarité est basé sur la nature et la représentation des attributs (numérique, catégorique, linguistique, etc.) et du domaine d'application. En pratique, les mesures de similarité dérivées de mesures de distance telles que les distances euclidiennes, euclidiennes inversées, pondérées ou le principe local-global [11], [12] sont généralement utilisées.
- Une règle de décision qui définit comment affecter une classe à une nouvelle instance à partir des classes des instances les plus similaires. Le choix instances les plus similaires dépend de l'interprétation de la qualification de « plus similaire » ; par exemple, les k premières instances similaires.

Plusieurs techniques ont été développées selon cette approche, telles que la technique des plus proches voisins [13], le classement basé sur la similarité [14], le raisonnement à base de cas utilisant des représentations symboliques plus complexes pour les instances [15], et la technique des plus proches prototypes [16]. Dans la suite, nous nous intéresserons à cette dernière technique.

2. Approche des K plus proches voisins

L'approche des *plus proches voisins* est considérée comme l'une des dix principales méthodes de la fouille de données [17]. C'est une approche d'apprentissage basée sur les instances, dans le sens où une étiquette de classe est affectée à un nouvel exemple en fonction des instances voisines les plus proches, correctement classées. L'algorithme des K plus proche voisin (K-NN) [13]est le plus connu et le plus populaire de cette approche, en raison de sa robustesse et de sa faisabilité [18]. Dans la suite, nous allons présenter un aperçu général sur KNN.

2.1. L'algorithme K-NN

L'algorithme des K plus proche voisin (K-NN) (figure 3) a été proposé par [13]. C'est un algorithme d'apprentissage, basé sur les instances, où l'affectation d'une classe à un nouvel exemple est effectuée en fonction de sa proximité, en termes de distance, aux instances déjà classées. En pratique, l'utilisation de K-NN nécessite le choix de deux paramètres. Le premier c'est la mesure de distance ; le choix dépend de la structure de l'espace des attributs caractérisant les données d'apprentissage. Le deuxième c'est le nombre K de voisins à prendre en compte dans le classement ; le nouvel exemple est affecté à l'étiquette de classe la plus courante parmi celles des K plus proches voisins obtenu par un vote majoritaire. Malgré les avantages présentés ci-dessus, l'algorithme K-NN peut générer des résultats biaisés lors du choix d'une valeur inappropriée de K. Il existe une multitude de façons différentes de choisir la valeur k ; la plus douce et efficace consiste à appliquer plusieurs fois l'algorithme et à modifier la valeur de k dans chaque essai afin d'atteindre la plus grande précision.

1. Calculer la similarité entre la nouvelle instance et les instances d'apprentissage à l'aide la distance sélectionnée.
2. Sélectionner les K voisins les plus proches à la nouvelle instance
3. Affecter l'étiquette à la nouvelle instance à partir des étiquettes des K voisins sélectionnés

Figure 3 : Algorithme KNN

2.2. Les améliorations de K-NN

K-NN a fait ses preuves et a donné de bons résultats au cours des dernières décennies. Sa puissance provient de sa capacité à traiter des problèmes de classement complexes caractérisés par des étiquettes de classe multiples, des données déséquilibrées ou bruitées [19]. Cela est dû au fait que les informations de classement sont contenues dans les voisins les plus proches et ne nécessite pas la connaissance préalable de la structure des données et de leur distribution. Malgré le succès réalisé, KNN souffre toujours de plusieurs faiblesses liées ; à la *précision de classement ;* à *l'inefficacité* qui est dû au fait que la taille du problème croît en fonction du nombre d'instances et du nombre d'attributs considérés ; et à la faible tolérance *des imprécisions* et des *incertitudes* qui apparaissent en raison des limites brutales entre les classes, du bruit induit par le choix inapproprié du paramètre K et de la technique de vote.

En conséquence, plusieurs améliorations du K-NN ont été proposées dans la littérature afin de remédier à ces inconvénients. Les améliorations concernent :

- La conception de nouvelles mesures de distance pour l'évaluation de la proximité de manière précise et significative[20][21]. Dans ce contexte, les techniques de pondération des attributs ont été introduites afin de tenir compte de l'importance des attributs dans le choix des voisins les plus proches.
- Le choix optimal et adéquat du paramètre K pour sélectionner les voisins les plus proches[22]. Dans cette optique, des techniques d'apprentissage automatique de K sont appliquées.
- Le traitement des incertitudes et des imprécisions à travers l'intégration de la théorie des ensembles flous et de la logique floue[23] dans le processus de classement de K-NN.
- L'amélioration de l'efficacité du K-NN en termes de temps d'exécution et de capacité de stockage à travers la réduction de l'ensemble d'apprentissage TR à un sous-ensemble d'instances significatives[24].

3. Amélioration de l'efficacité de K-NN

L'efficacité constitue une préoccupation importante dans l'approche de classement par K-NN, du fait qu'elle exige le stockage et l'exploration de toutes les instances d'apprentissage au cours du processus du classement. En effet, cette exigence conduit, souvent, à un temps d'exécution important et un besoin élevé en stockage, notamment, dans le cas de grands ensembles de données d'apprentissage. Dans la suite, nous allons discuter les voies susceptibles de pallier à l'inefficacité sous-jacente.

3.1. Apprentissage de prototypes

Pour répondre à ce besoin d'efficacité, plusieurs améliorations de K-NN ont été proposées [25]. L'idée commune entre ces améliorations est la réduction de l'ensemble d'apprentissage à un sous-ensemble d'instances appelées *prototypes ;* le classement d'une nouvelle instance est ensuite effectué par K-NN en utilisant les prototypes obtenus comme ensemble d'apprentissage. À cet égard, plusieurs techniques d'apprentissage de prototypes ont été développées pour fournir l'ensemble de prototypes adéquat [24], [26]. Selon le type d'apprentissage, On distingue :

- L'apprentissage par *sélection* [27], où les prototypes sont sélectionnés parmi les instances de l'ensemble d'apprentissage. Les prototypes peuvent être obtenus en condensant les instances de l'ensemble d'apprentissage [28], ou en supprimant les instances bruyantes [29].

- L'apprentissage par *génération* [30], où de nouvelles instances (les prototypes) sont construites à partir de l'ensemble d'apprentissage dans le but de mieux représenter le domaine du problème. Les méthodes de génération de prototypes fonctionnent souvent en modifiant la position ou l'étiquette des instances d'origine ou en générant de nouvelles instances.

3.2. Approches pour l'apprentissage des prototypes

Dans la littérature, plusieurs travaux ont abordé la question de la sélection de prototypes et proposé une multitude de techniques d'apprentissage visant à extraire les prototypes adéquats. Elles peuvent être classées en trois approches (Figure 4) :

Figure 4 : Approches de l'apprentissage de prototypes

- *Condensation* : cette approche fournit les prototypes les plus proches de la limite de décision. Cette méthode garantit l'obtention d'un bon taux de réduction en fonction du nombre d'instances retenues et permet de préserver la précision de classement.
- *Edition* : cette approche tente de supprimer les instances bruyantes ou qui ne correspondent pas à leurs voisins. Cependant, les méthodes d'édition ne suppriment pas les points internes et ne laissent pas une limite de décision plus lisse. Les avantages importants de cette approche résident dans sa capacité à préserver la structure des ensembles d'apprentissage, le taux de réduction est considéré comme faible dans la plupart des cas.
- *Hybride* : cette approche effectue une hybridation des deux approches précédentes. Il essaie d'apprendre les prototypes pouvant maintenir ou augmenter la précision du classement par rapport à l'ensemble d'apprentissage original. À cette fin, l'algorithme d'apprentissage effectue la réduction en fonction de la combinaison des critères suivis par les deux approches précédentes. Les méthodes hybrides sont très utiles et permettent d'obtenir une bonne précision avec de petits ensembles de prototypes.

Conclusion

Dans ce chapitre, nous avons présenté un état de l'art sur les techniques de classement de l'apprentissage supervisé en se focalisant sur la méthode de classement K-NN. Nous avons ensuite identifié et discuté les faiblesses de KNN et aussi les méthodes pour y remédier. Dans le chapitre suivant, nous allons présenter notre solution à la problématique de l'efficacité et à la gestion des imprécisions et incertitudes.

Chapitre 3
Solution Proposée

Chapitre 3 : Solution proposée

Introduction

Dans ce chapitre, avant d'aborder la solution proposée, nous allons commencer par la présentation des notions utilisées dans le développement de notre solution, telles que les ensembles flous, les variables linguistiques et le regroupement flou. Ensuite nous donnons une description de notre solution qui consiste en un modèle de classement par les plus proches prototypes, basé sur les ensembles flous

1. Préliminaires

1.1. Logique floue

La logique floue est une forme de logique à valeurs multiples dans laquelle les valeurs de vérité des variables peuvent être tout nombre réel compris entre 0 et 1 inclus. Il est utilisé pour traiter le concept de vérité partielle, où la valeur de vérité peut être comprise entre complètement vrai et complètement faux. En revanche, dans la logique booléenne, les valeurs de vérité des variables ne peuvent prendre que les valeurs : 0 et 1.

Le terme de logique floue a été introduit avec la proposition de Lotfi Zadeh [31]sur la théorie des ensembles flous. La logique floue avait cependant été étudiée depuis les années 1920, en tant que logique à valeur infinie.

Elle est basée sur l'observation que les personnes prennent des décisions basées sur des informations imprécises et non numériques, les modèles ou ensembles flous étant des moyens mathématiques de représenter des informations vagues et imprécises, d'où le terme flou. Ces modèles ont la capacité de reconnaître, de représenter, de manipuler, d'interpréter et d'utiliser des données et des informations vagues et peu sûres.

La logique floue a été appliquée à de nombreux domaines, de la théorie du contrôle à l'intelligence artificielle.

1.1.1. Ensembles flous

Selon L. Zadeh[31], la logique floue s'appuie sur la théorie des ensembles flous, qui est une extension de la théorie des ensembles classiques. En effet, dans la théorie classique, la notion d'appartenance à un ensemble Aest définie par la fonction caractéristique suivante :

$$\mu_A(x) = \begin{cases} 0 \; si \; x \notin A \\ 1 \; si \; x \in A \end{cases}$$

Qui signifié qu'un élément x est soit dans$A$soit à l'extérieur de$A$.

L'extension à la théorie floue, introduit la notion d'appartenance partielle. En effet, un sous-ensemble flou$A$de l'univers de discours$E$est caractérisé par ce qu'on appelle une fonction d'appartenance. C'est une fonction définie par :

$$\begin{array}{rl} \mu_A: & E \to [0,1] \\ & x \longmapsto \mu_A(x) \end{array}$$

qui associe à chaque élément$x$ de E une valeur,$\mu_A(x)$, dans[0,1] qui représente le degré d'appartenance de x à A.

1.1.2. Fonctions d'appartenances usuelles

Dans la pratique les ensembles flous sont associés à des fonctions d'appartenance simples et usuelles[32]. Trois formes sont fréquemment utilisées, les triangles, les trapézoïdes et les gaussiennes. Dans la suite nous présentons ces formes, pour chaque forme nous donnons l'expression et la représentation graphique.

Forme semi-trapézoïdale gauche (Figure 5) :

$$f(x; a, b) = \max\left(min\left(\frac{x-a}{b-a}, 1\right), 0\right)$$

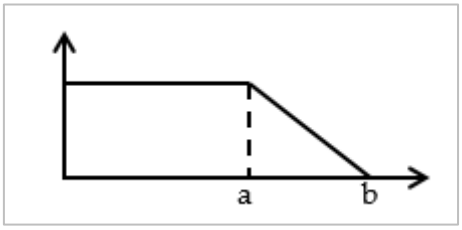

Figure5 : Fonction d'appartenance de type semi-trapézoïde gauche

Forme semi-trapézoïdale droit (Figure 6) :

$$f(x; a, b) = \max\left(\min\left(1, \frac{b-x}{b-a}\right), 0\right)$$

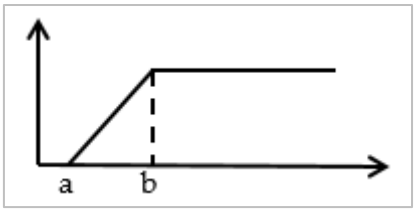

Figure6 : Fonction d'appartenance de type semi-trapézoïde droit

Forme trapézoïdale (Figure 7) :

$$f(x;a,b,c,d) = \max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right)$$

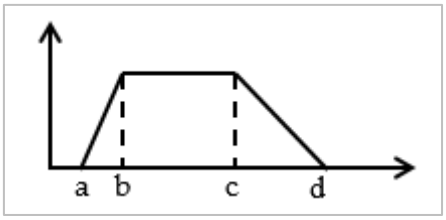

Figure7 : Fonction d'appartenance de type trapézoïde

Forme triangulaire (Figure 8) :

$$f(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$$

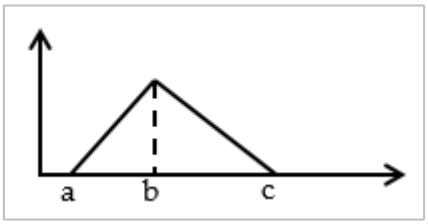

Figure8 : Fonction d'appartenance de type triangle

Forme gaussienne (Figure 9) :

$$f(x\,;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

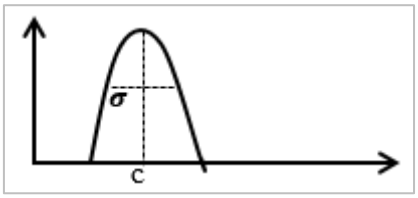

Figure9 :Fonction d'appartenance de type gaussiene

1.1.3. Propriétés des ensembles flous

Un ensemble flou$A$ associé à la fonction d'appartenance désignée parμ_A, est caractérisé par les éléments suivants :

La forme de la fonctionμ_A, qui peut être triangulaire, gaussienne, trapézoïdale, etc.

La hauteur représente la borne supérieure de la fonction d'appartenance. Elle est donnée par :$H_A = sup\mu_A(x)$. L'ensemble flou A est dit normalisé si sa hauteur$H_A = 1$.

Le noyau c'est l'ensemble des éléments$x$ de l'univers de discours$E$ qui appartiennent complètement àA.Le noyau est donné par :

$$N_A = \{x \epsilon A | \mu_A(x) = 1\}$$

Le support c'est l'ensemble des éléments x qui appartiennent, au moins un petit peu, àA. Le support est donné par :$S_A = \{x \epsilon E \mid \mu_A(x) \neq 0\}$

L'α-coupe c'est l'ensemble des éléments qui appartiennent à A avec un degré d'appartenance au moins égal àα. Il est donné par :

$$\alpha - coupe = \{x \epsilon A \mid \mu_A(x) \leq \alpha\}$$

La Figure10 résume ces différentes propriétés :

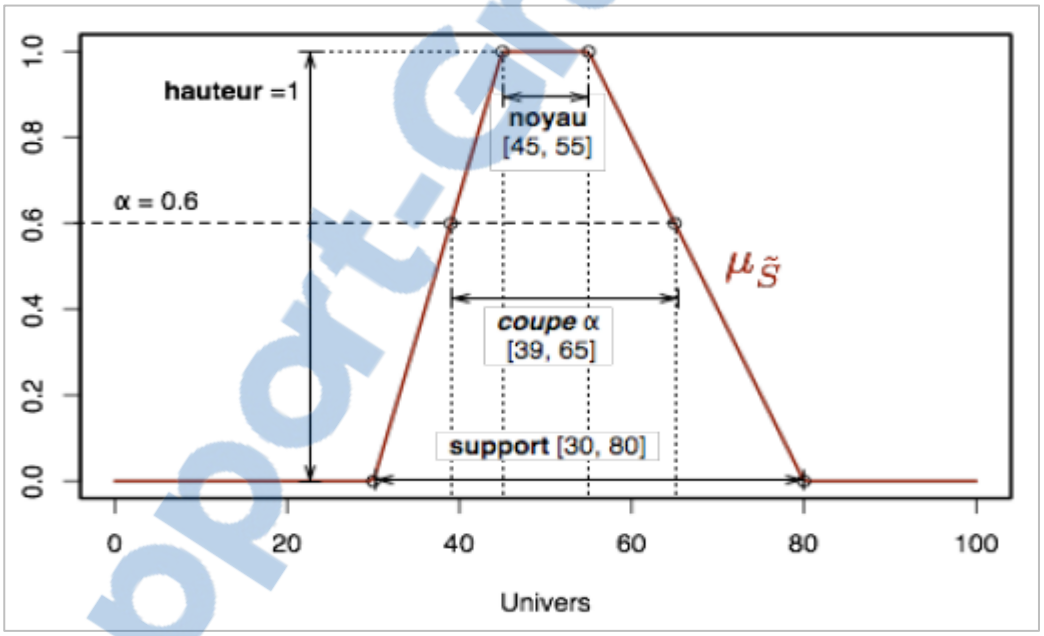

Figure 10 : Propriétés d'un ensemble flou

Un ensemble flou A est dit convexe si :

$$\forall\, x_1, x_2, x_3 \in X, x_1 \leq x_2 \leq x_3 alors \mu_A(x_2) \geq \min\left(\mu_A(x_1), \mu_A(x_3)\right)$$

1.1.4. Opérations sur les ensembles flous

Comme dans le cas des ensembles classiques, les opérations ensemblistes : l'intersection, l'union et le complément sont définies sur les ensembles flous. Dans la littérature (Godjevac, 1999), on trouve une multitude de définitions de ces opérations, ils peuvent être généralisés respectivement grâce à la t-norme, t-conorme (s-norme) et le complément. Le tableau ci-dessous montre les définitions les plus utilisées.

Références	T(a,b)	S(a,b)	C(a)
Zadeh, 1973	min (a,b)	max (a,b)	1-a
Lukasiewicz, 1976	max (a+b-1,0)	min (a+b-1,0)	Sugeno, 1977
Bandler et Kohout, 1980	Ab	a+b-ab	$\frac{1-a}{1+\lambda a}, \quad \lambda > 0$
Hamacher, 1978	ab/(γ+(1-γ); a+b-ab) , γ≥0 .	(a+b −(2-γ)ab)/(1-(1-γ)ab) , γ≥0 .	Yager, 1980
Weber, 1983	a si b=1 b si a=1 0 sinon	a si b=0 b si a=0 1 sinon	$\sqrt[P]{(1-a^P)}, P>0$

Tableau 1 : Opérateurs t-norme, t-conorme et complément

1.2. Variables linguistiques

Une variable linguistique[33] est une variable qui a des valeurs linguistiques au lieu de nombres tel que large, rond ,jeune, haut etc. .Le but essentiel d'utiliser des valeurs linguistiques au lieu des nombres même si les nombres sont plus spécifiques mais sont plus proche de la manière les êtres humains expriment leurs connaissances .Par exemple , la hauteur d'un être humain peut être décrite par des valeurs qualitatives par des valeurs qualitatives modélisables par la variable linguistique Hauteur, qui prend ses valeurs dans l'ensemble de termes linguistiques: court, moyen, grand, très grand.

Une variable linguistique est caractérisée par un quintuple (X, T (X), U, S, M (X)), où X est le nom de la variable, T (X) est l'ensemble de termes contenant les valeurs linguistiques prises par X, U est l'univers du discours (domaine de X), S est la règle syntaxique qui génère les termes linguistiques en T (X), et Mis une partition de sous-ensembles flous de U représentant la règle sémantique qui affecte le sens de chaque terme linguistique en T (X).

Ces sous-ensembles flous sont associés à leurs fonctions d'appartenance qui correspondent chaque entrée de U dans le vecteur MG qui satisfait à la contrainte.

$$\sum_i MG_i = 1$$

Par conséquent, on peut dire qu'une variable linguistique est décrite par une partition floue composée de sous-ensembles flous associés à leurs fonctions d'appartenance. Les ensembles flous adjacent se chevauchent, conséquence naturelle de leurs limites. Chaque ensemble flou

couvre, de manière unique, une partie du domaine variable et partage une autre partie avec les ensembles flous adjacents. Un tel chevauchement met en œuvre une règle sémantique réaliste et pratique, qui capture la nature progressive d'une variable linguistique et fournit donc une transition harmonieuse et cohérente d'un état à l'autre. Pour illustrer cet objectif, **la** Figure 11 illustre une partition floue représentant la taille comme variable linguistique :

(Taille, T(Taille), [0,250], M(Taille))

Avec, T(Taille)= {Très petite, petite, moyenne, grand, très grand}

$$M(Taille) = \{\mu_{\text{Très petite}}, \mu_{\text{petite}}, \mu_{\text{moyenne}}, \mu_{\text{grand}}, \mu_{\text{très grand}}\}$$

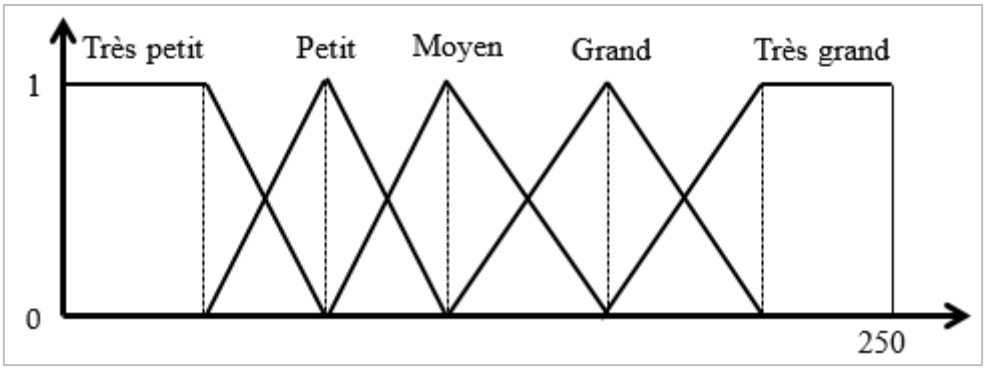

Figure11 : Représentation graphique de la variable linguistique Taille

1.3. Regroupement flou

Le regroupement flou est une méthode de regroupement (clustering) dans laquelle les points de données peuvent appartenir à plusieurs groupes (« clusters »). Le regroupement divise les points de données en groupes basés sur la similarité entre les éléments et cherche à trouver des motifs ou des similitudes entre les éléments d'un ensemble ; Les éléments dans un cluster doivent être aussi proches que possible les uns des autres et aussi différents que possible des éléments d'autres groupes. En calcul, il est beaucoup plus facile de créer des limites floues que de s'installer sur un cluster pour un point.

Dans un regroupement « dur », chaque point de données ne peut être que dans un cluster. Dans un regroupement « doux » ou « flou », les points de données peuvent appartenir à plusieurs groupes. Le regroupement flou utilise des solutions de moindres carrés pour trouver l'emplacement optimal pour n'importe quel point de données. Cet emplacement optimal peut être dans un espace de probabilité entre deux (ou plusieurs) clusters.

1.3.1. Fuzzy C-means

Fuzzy C-Means (FCM)[34] est un algorithme largement utilisé, cet algorithme est pratiquement identique à l'algorithme K-Means. Un point de données peut théoriquement appartenir à tous les groupes, avec une fonction d'appartenance (également appelée note

d'adhésion) comprise entre 0 et 1, où : 0 correspond au point le plus éloigné possible du centre du cluster et 1 à l'emplacement où les données le point est le plus proche du centre.

FCM repose sur une problématique de partitionnement formulée par :

$$MinJ_m(\mathrm{U},\mathrm{V}) = \sum_{j=1}^{c}\sum_{i=1}^{n} \mu_{ij}^{m} \left\|x_i - v_j\right\|^2$$

Avec :

$$\sum_{i=1}^{c} \mu_{ij} = 1, \qquad \forall\, j = 1; \dots, n$$

Avec :

$X = \{x_1, \dots, x_n\}$; le vecteur des données à regrouper,

c : le nombre de clusters désiré,

m : le paramètre de fuzzification,

$\mathrm{V} = (\mathrm{v_i})_{\mathrm{i}\leq \mathrm{c}}$Les centres des clusters,

$\mathrm{U} = (\mu_{\mathrm{ij}})$La matrice de partition qui contient les degrés d'appartenance de toutes les données.

Pour trouver la partition floue optimale, l'algorithme FCM[35] opère itérativement selon les étapes suivantes :

- **Etape 0 :** on choisit le nombre de clusters désiré, le paramètre m et le critère d'arrêt. Ensuite, on initialise arbitrairement la matrice de partition en respectant l'équation

$$\sum_{i=1}^{c} \mu_{ij} = 1, \qquad \forall\, j = 1; \dots, n$$

- **Etape 1 :** on calcule les centres des clusters avec la formule suivante :

$$v_j = \frac{\sum_{k=1}^{n} \mu_{jk}^{m} x_k}{\sum_{k=1}^{n} \mu_{jk}^{m}}$$

- **Etape 2 :** on réajuste la matrice de partition, suivant la position des centres par la formule suivante :

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - v_j\|}{\|x_i - v_k\|}\right)^{1/(m-1)}}$$

- **Etape 3 :** si le critère d'arrêt n'est pas vérifié on retourne à l'étape 1.

1.3.2. Indices de validités

L'inconvénient majeur de l'algorithme FCM est que le nombre de clusters doit être fourni préalablement. Pour choisir le nombre de clusters adéquat, on fait appel à des critères de validité des clusters, qui mesurent la qualité de partitionnement. Les indices de validité internes [36] qui utilisent les informations internes du processus de regroupement pour évaluer la qualité d'une structure des partitions obtenues sans faire référence à des informations externes. Il peut également être utilisé pour estimer le nombre de clusters et l'algorithme de regroupement approprié sans aucune donnée externe.

1.3.3. Indices de validité dans un environnement flou

Un certain nombre d'indices de validité [36]pour le regroupement flou existent dans la littérature. Les premiers indices tels que le coefficient de partage et l'entropie de classement utilisent uniquement les valeurs d'appartenance et ont l'avantage d'être faciles à calculer. Or, il est largement admis qu'une meilleure définition d'un indice de validité considère toujours à la fois la matrice de partition U et l'ensemble de données lui-même. Dans cette section, nous passons en revue certains indices de validité de cluster disponibles dans la littérature.

$X = \{x_1, x_2, \dots, x_n\}$

c : Nombre de clusters

U : Matrice de partition $\left[u_{ij}\right]_{c*n}$ avec $i = \{1, \dots, c\}$ et j= $\{1, \dots, n\}$

V : Les prototypes (Les centres des clusters) $V = \{v_1, \dots, v_c\}$

1.3.3.1. Indices de validité basés sur les degrés d'appartenance

- **Coefficient de Bezdek (PC):**[37]

Bezdek a tenté de définir une mesure de performance basée sur la minimisation du contenu global des intersections floues par paires en U, la matrice de séparation.

L'indice est défini par :

$$V_{PC} = \frac{1}{n} \sum_{i=1}^{c} \sum_{j=1}^{n} {\mu_{ij}}^2$$

En général, on trouve l'optimal nombre de cluster c par la résolution de :

$$max_{2<c<n-1}\, V_{PC}$$

- **Bezdek partition entropy (PE)**[38]**:**

Bezdek a proposé « Partition Entropy (PE) » qui est définie par :

$$V_{PE} = \frac{1}{n}\sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}\log_a \mu_{ij}$$

En général, on trouve l'optimal nombre de cluster c par la résolution de :

$$max_{2<c<n-1}\, V_{PE}$$

1.3.3.2. Indices de validité en utilisant les degrés d'appartenance et l'ensemble de données

- **Indice de Fukuyama and Sugeno (FS):**

Une fonction de validité proposée par Fukuyama et Sugeno (FS) est définie par :

$$V_{Fs} = \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m}\left\|x_j - v_i\right\|^2 - \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m}\left\|v_i - \bar{v}\right\|^2$$

Avec $\bar{v} = \sum_{i=1}^{c}\frac{v_i}{c}$

En général, on trouve l'optimal nombre de cluster c par la résolution de :

$$min_{2<c<n-1}\, V_{FS}$$

- **Indice de Xie and Beni (XB)**[39] **:**

Une fonction de validité proposée par Xie and Beni (XB) avec m = 2 et modifiée par Pal et Bezdek [39] est définie par :

$$V_{XB} = \frac{\sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^{m}\left\|x_j - v_i\right\|^2}{n min_{ij}\left\|v_i - v_j\right\|^2}$$

En général, on trouve l'optimal nombre de cluster c par la résolution de :

$$min_{2<c<n-1}\, V_{XB}$$

- **Gath et Geva Fuzzy Hypervolume(FH)** [40]**:**

Une fonction de validité est proposée par Gath et Geva est définie par :

$$V_{FHV} = \sum_{i=1}^{c}[\det(F_i)]^{\frac{1}{2}}$$

Avec $F_i = \frac{\sum_{i=1}^{j}(u_{ij})^m\,(x_j - v_i)(x_j - v_i)^T}{\sum_{j=1}^{n}(u_{ij})^m}$

La matrice Fi désigne la matrice de covariance floue du cluster i. On peut s'attendre à ce que la valeur de V_{FHV} d'une partition floue soit faible si la partition est étroite. Un extremum pour cet indice indiquerait idéalement une bonne partition.

En général, on trouve l'optimal nombre de cluster c par la résolution de :

$$min_{2<c<n-1}\, V_{F_i}$$

1.4. Mesure de similarité floue

La similarité [41]est un concept clé dans le contexte du classement lors de l'utilisation d'une approche du classement basée sur l'analogie. Le choix d'une mesure de similarité est très important car il influencera le choix des voisins qui contribueront à l'étape du classement.

On considère $x_1 = \left\{x_1^{f_1}, x_1^{f_2}, \dots, x_1^{f_t}\right\}$ et $x_2 = \left\{x_2^{f_1}, x_2^{f_2}, \dots, x_2^{f_t}\right\}$

Cette similarité est évaluée en utilisant les étapes suivantes :

Similarité individuelle :

Évalue la similarité entre a_1 et $a_2$en fonction d'une caractéristique, disonsf_1. De nombreuses mesures de similarité ont été proposées dans [41], telles que somme-produit, max-min, etc. Nous avons utilisé :

$$\text{Simi}_{f_i}(x_1^{f_i}, x_2^{f_i}) = \min(max_k \min\left(\mu_j\left(x_1^{f_i}\right), \mu_j\left(x_2^{f_i}\right)\right), (1 - \min_k \max\left(\mu_j\left(x_1^{f_i}\right), \mu_j\left(x_2^{f_i}\right)\right)))$$

Avec :

$Simi_{f_i}$est la similarité individuelle par rapport à la variable f_i ,

μ_j la fonction d'appartenance associée avec le $j^{ème}$ ensemble flou de l'attribut $f_i; j = 1, \dots, l,$

k est le nombre d'ensembles flous définis pour l'attribut $f_i,$

Si la similarité individuelle est haute on peut dire que l'attribut $f_i$est similaire entre x_1 et x_2 , et si elle basse on dit que l'attribut f_in'est pas similaire entre x_1 et x_2.

Similarité globale :

Calcule le similarité globale entre a_1 et a_2 en utilisant tous les similarités individuelles $\text{Simi}_{f_i}(a_1^{f_i}, a_2^{f_i})$ en les combinant $Sim(x_1, x_2) = \sum_{i=1}^{l} Simi_{f_i}(x_1, x_2)$.

2. Description de la solution

2.1. Formulation du problème

Soit TS l'ensemble de données $TS = \{X_1, X_2, \dots\dots\dots, X_n\}$ Avec n le nombre de données.

Chaque $X_i$contient un nombre d'attributs m $X_i = \{X_{i1}, X_{i2}, \dots, X_{im}\}$.

$C = \{C_1, \dots, C_l\}$ avec l est le nombre de clusters et $P = \{P_1, \dots, P_k\}$ avec k le nombre de prototypes.

2.2. Processus de classement

La méthode proposée permet de classer une nouvelle instance en se basant sur l'approche des plus proches prototypes. Le classement d'une nouvelle instance repose sur un ensemble de prototypes obtenus à l'aide de l'algorithme FCM et des indices de validité et d'une mesure de similarité floue qui permet d'évaluer la similarité entre deux instances. La figure 12 montre le processus de classement par la méthode proposée :

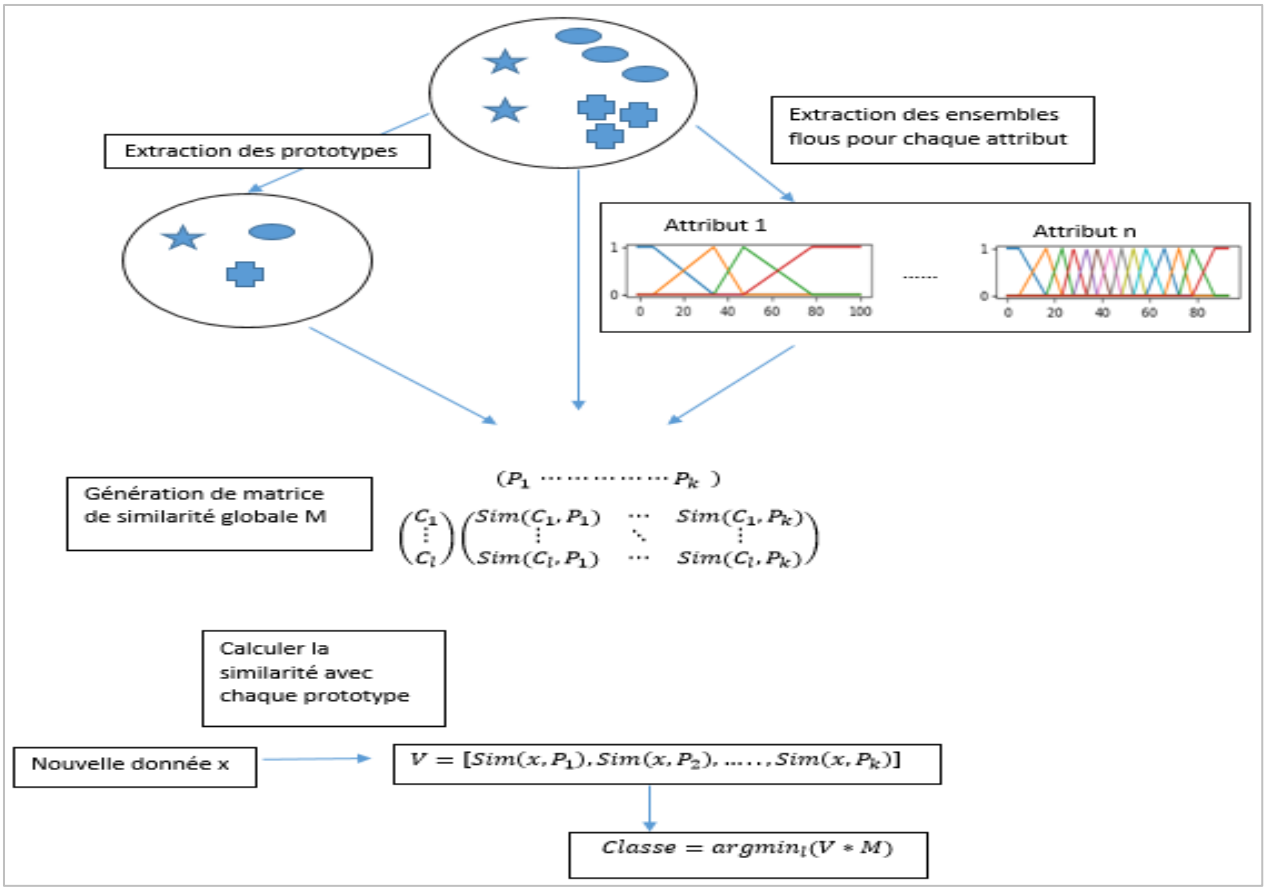

Figure 12 : Processus du classement

2.3. Extraction des prototypes

Extraction des prototypes avec l'aide du Fuzzy C-means et L'indice de validité (Figure13,14)

Algorithme CVI :

1. Pour i=2 jusqu'à n :
   A. Extraction des centres et la matrice d'appartenance avec le nombre de clusters c=i en utilisant le Fuzzy C-means

   B. Appliquer la fonction d'indice de validité
   FinPour

2. Retourner l'indice de validation optimal

Figure 13 : Algorithme d'indice de validité

Algorithme : Extraction des prototypes

1. Appliquer l'algorithme CVI pour sélectionner le nombre de cluster par CVI

c= CVI(data)

2. Appliquer Fuzzy C-means avec le nombre de cluster= c
3. Retourner les centres des clusters pour les utiliser comme des prototypes

Figure 14: Extraction des centres d'attributs

2.4. Extraction des ensembles flous

Extraction des ensembles flous pour chaque attribut à l'aide des CVI (Figure 15)

Algorithme : Extraction des centres des attributs

Pour chaque attribut $data_i i \in [0, m]$:

1. Appliquer l'algorithme CVI pour sélectionner le nombre de cluster par CVI

$$c = CVI(data_i)$$

2. Appliquer Fuzzy C-means avec le nombre de cluster= c
3. Retourner les centres des clusters le maximum et le minimum pour créer l'ensemble flou de cet attribut

FinPour

Figure 15 : Algorithme d'extraction des prototypes

2.5. Calcul des similarités individuelles

- Pour chaque attribut, Calculer la similarité individuelle avec de la donnée avec les prototypes (Figure 16)

Algorithme : Calcul de similarité entre un prototype et une donnée : $SimG(x_i, p_j)$

Pour chaque attribut $k \quad k \in [0, l]$:

0. Calculer les degrés d'appartenance du prototype p_j et de la donnée x_i pour chaque ensemble flou de l'attribut k en utilisant les fonctions d'appartenance.
1. Appliquer la similarité individuelle en utilisant les degrés d'appartenance du prototype et p_j et de la donnée x_i

   Avec la formule :

$$Sim(x_i, p_j, k) = \min(max_k \min\left(\mu_{V_j^k}(x_{ik}), \mu_{V_j^k}(p_{jk})\right), (1 - \min_k \max\left(\mu_{V_j^k}(x_{ik}), \mu_{V_j^k}(p_{jk})\right))$$

   FinPour

   Retourner $moy(Sim(x_i, p_j, k))$

Figure 16:Similarité individuelle

2.6. Génération de la matrice de similarité globale :

- Pour chaque classe, Calculer la similarité globale avec les prototypes (Figure 17) :

Algorithme : Calcul de similarité entre les prototypes et les données $Simc(c_i, pj, data)$

Pour chaque donnée $x_i avec x_i \in data = [x_1, \dots, x_n]$

Si $x_i \in c_i$:

Calculer $SimG(x_i, p_j)$

FinPour

Retourner $\sum_{i=1}^{n} SimG(x_i, p_j)$

Figure 17: Similarité globale des classes et prototypes

- Générer la matrice de similarité :

$$M = \begin{bmatrix} m_{11} & \cdots & m_{1k} \\ \vdots & \ddots & \vdots \\ m_{1l} & \cdots & m_{lk}) \end{bmatrix}$$

$$M = \begin{bmatrix} Simc(c_1, p_1, data) & \cdots & Simc(c_1, p_m, data) \\ \vdots & \ddots & \vdots \\ Simc(c_n, p_1, data) & \cdots & Simc(c_n, p_m, data) \end{bmatrix}$$

Normaliser M pour chaque ligne (Figure 18) :

Pour i allant de 1 jusqu'à k :

$Somme = 0$

Pour j allant de 1 jusqu'à l :

$Somme = Somme + m_{ij}$

FinPour

Pour z allant de 1 jusqu'à l :

$m_{iz} = m_{iz}/somme$

FinPour

FinPour

Retourner M

Figure 18 : Normalisation de la matrice des similarités

2.7. Classement

Classement d'une nouvelle donnée inconnue x (Figure 19) :

Algorithme : Classement d'une nouvelle donnée : $predict(x)$

Pour chaque prototype $p_i avec p_i \in P = [p_1, \dots, p_m]$

Calculer $SimG(x, p_i)$

FinPour

Retourner R= $[SimG(x, p_1), SimG(x, p_2), \dots \dots, SimG(x, p_m)]$

Multiplier le matrice M par R^T

Retourner $Argmax_i = \begin{bmatrix} Simc(c_1, p_1, data) * R_1 & \cdots & Simc(c_1, p_m, data) * R_m \\ \vdots & \ddots & \vdots \\ Simc(c_n, p_1, data) * R_1 & \cdots & Simc(c_n, p_m, data) * R_m \end{bmatrix}$

(La classe de la ligne où la somme de ces éléments est le max).

Figure 19:Classement d'une nouvelle donnée

Conclusion

Dans ce chapitre on a utilisé la logique floue pour essayer de régler le problème d'imprécision et de manques de données, le prototypage par regroupement pour réduire les données, les ensembles flous et les variables linguistiques avec la mesure de similarité pour donnent plus d'interprétabilité. On a utilisé ses notions pour construire la solution proposée qu'on va voir ses résultats dans le chapitre suivant.

Chapitre 4
Application et expérimentations

Chapitre 4 :

Application et expérimentations

Introduction

Dans ce chapitre, nous décrivons le développement de l'application web qui facilite l'utilisation de notre système du classement proposé, de sorte que les développeurs et les chercheurs peuvent utiliser notre service pour former et tester leurs propres ensembles de données afin de comparer leur exactitude et de tester l'un d'entre eux s'il en a besoin dans le processus de prédiction.

1. Outils de développement

1.1. Environnement de développement Pycharm

Pycharm est un IDE, Integrated Développent Environment (EDI environnement de développement intégré en français), spécialisé pour les langages de programmation Python et Django. Il offre de riches et nombreuses fonctionnalités en matière d'édition, de débogage, de développement et de tests.

1.2. Anaconda

Anaconda est une distribution libre et open source3 des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement.

1.3. Flask

Flask est un Framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de Template.

2. Interfaces Graphiques

La conception des interfaces de l'application est une étape très importante puisque toutes les interactions avec le cœur de l'application passent à travers ces interfaces, on doit alors guider l'utilisateur avec les messages d'erreurs et de notification si besoin, ainsi présenter un système complet. Dans cette partie, nous allons présenter quelques interfaces de l'application, répondant aux recommandations ergonomiques de compatibilité, de guidage, de clarté et de souplesse.

2.1. Interface « Classement des bases de données prédéfinis »

Dans cette interface (Figure 20) existe 12 choix chaque choix est une base de données, chacune de ces dernières sont composés des 10 fichiers d'entrainement et de test pour faire 10-fold cross validation.

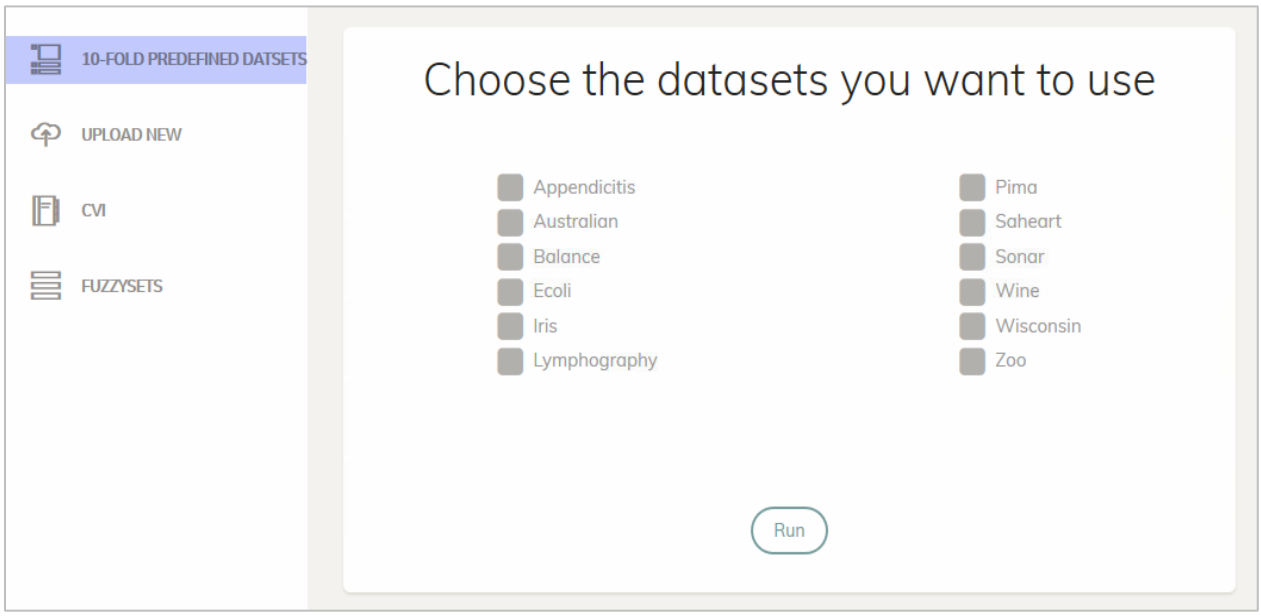

Figure20 : Interface du classement des bases de données prédéfinis

Ici (Figure 21) on a sélectionné IRIS et Zoo par exemple voici les taux du classement correcte trouvé pour chaque indice de validité calculé par la méthode proposée :

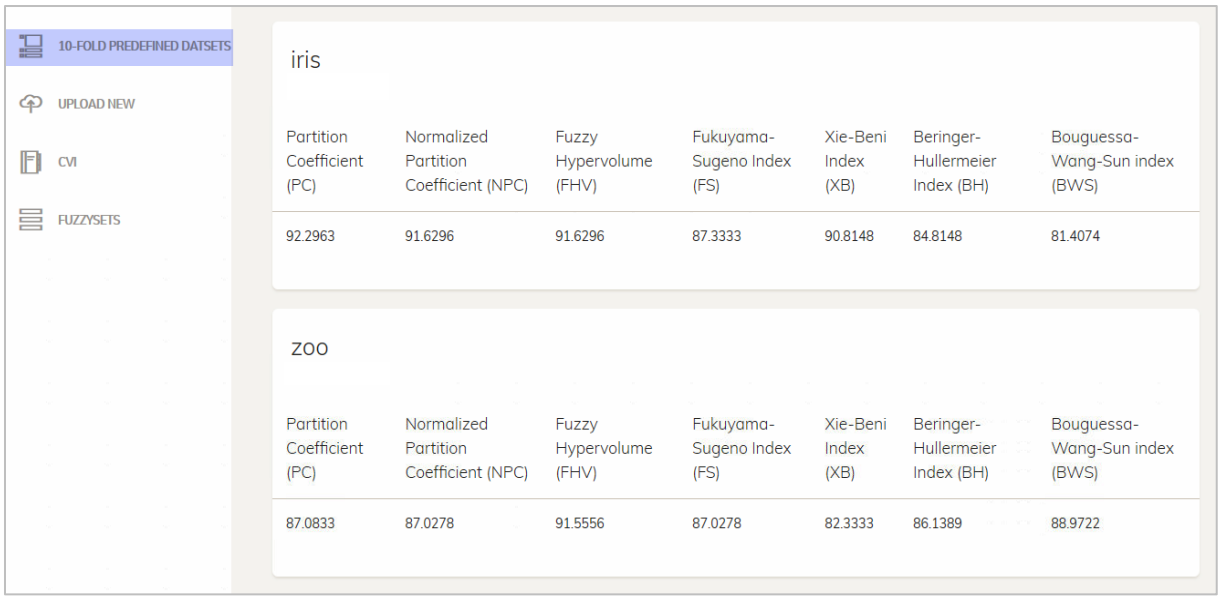

Figure21 : Taux du classement pour les méthodes sélectionnées

2.2. Interface « Nouvelle base de données »

Cette interface (Figure 22) est pour classifier une base de données en la donnant un fichier d'entrainement est un de test.

Au-dessous se trouvé le taux du classement correct pour chaque indice de validité :

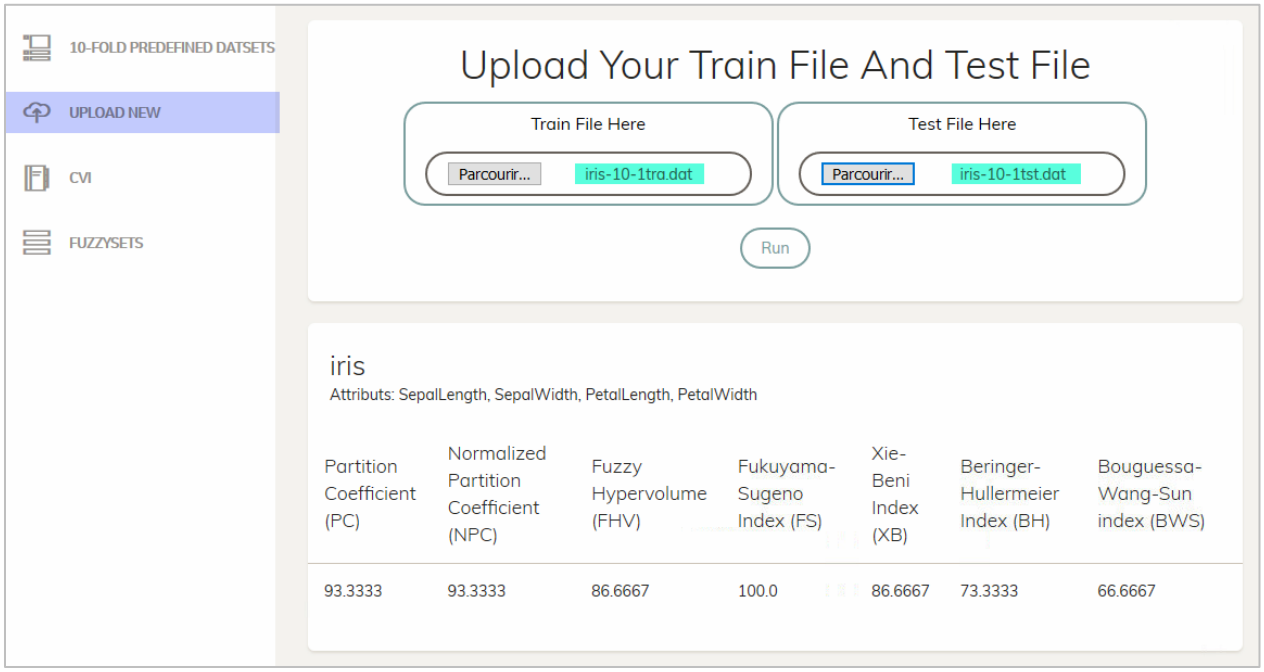

Figure22 : Interface du classement

2.3. Interface « CVI »

Retourne les indices de validité utilisés utilisé dans la méthode du classement (Figure 23) :

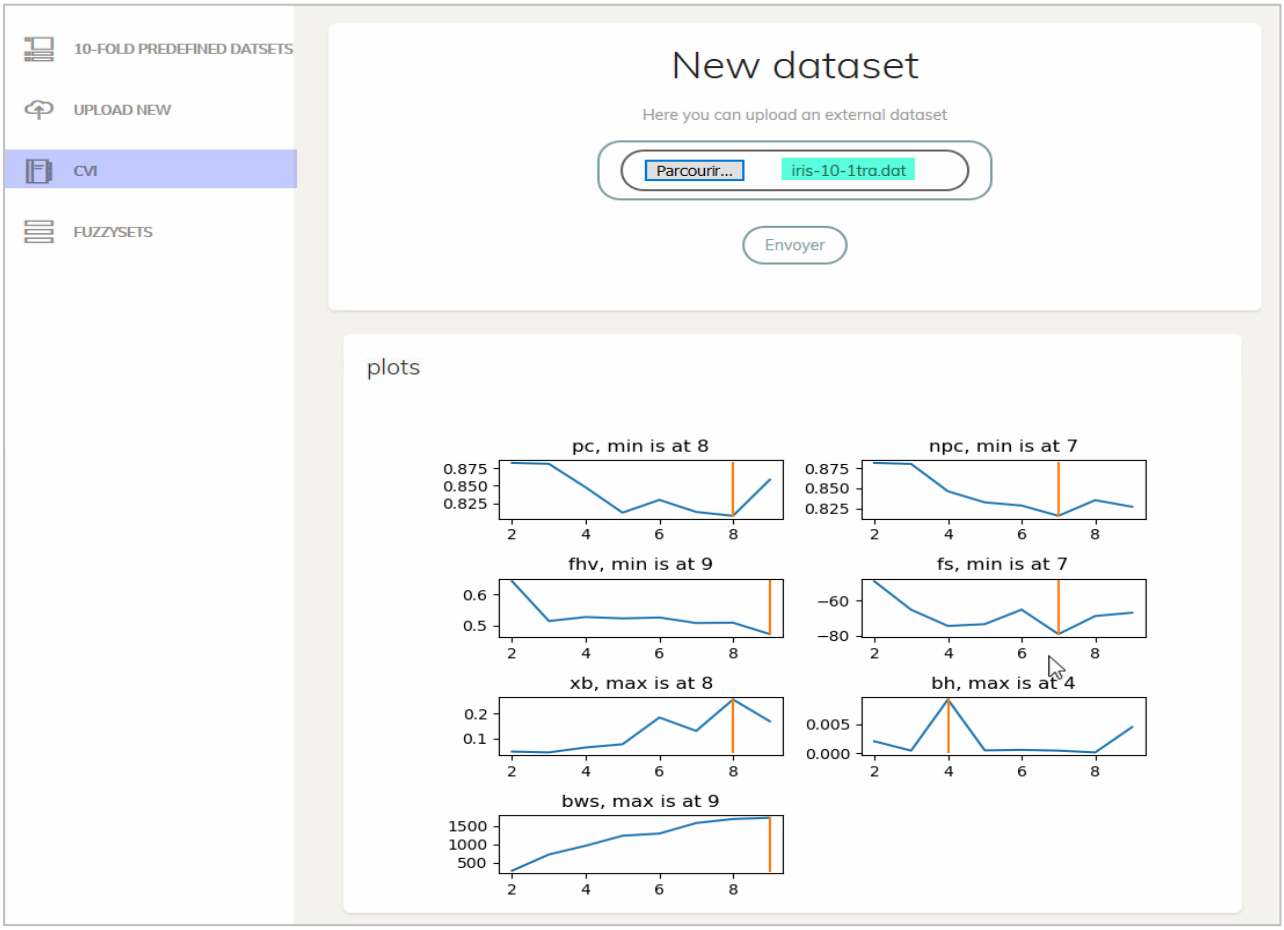

Figure23 : Interface des indices de validité

2.4. Interface « Ensembles flous »

Après avoir sélectionner le fichier le test (Figure 24) :

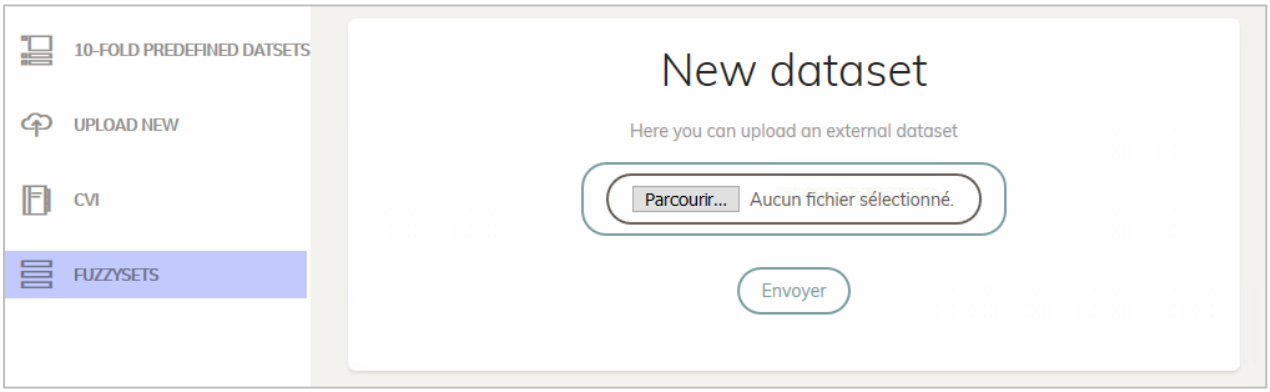

Figure24 : Interface d'affichage des ensembles flous

Le système calcule les ensembles flous pour chaque attribut et les affichent dans un graphe (Figure 25) :

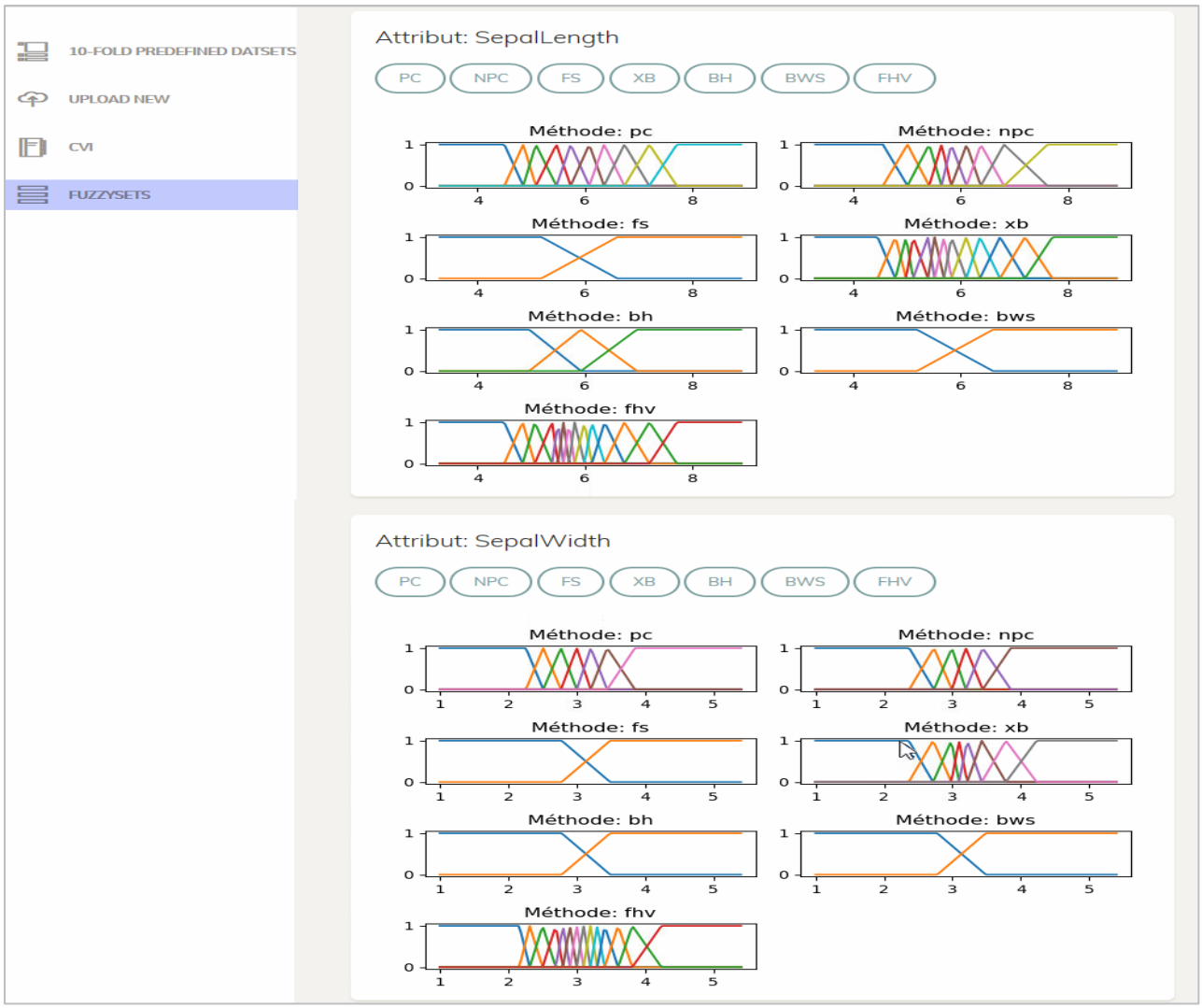

Figure25 : Exemple de résultats pour les ensembles flous

En cliquant sur l'une des buttons 

S'affiche les ensembles flous calculés pour chaque indice de validité (Figure 26) :

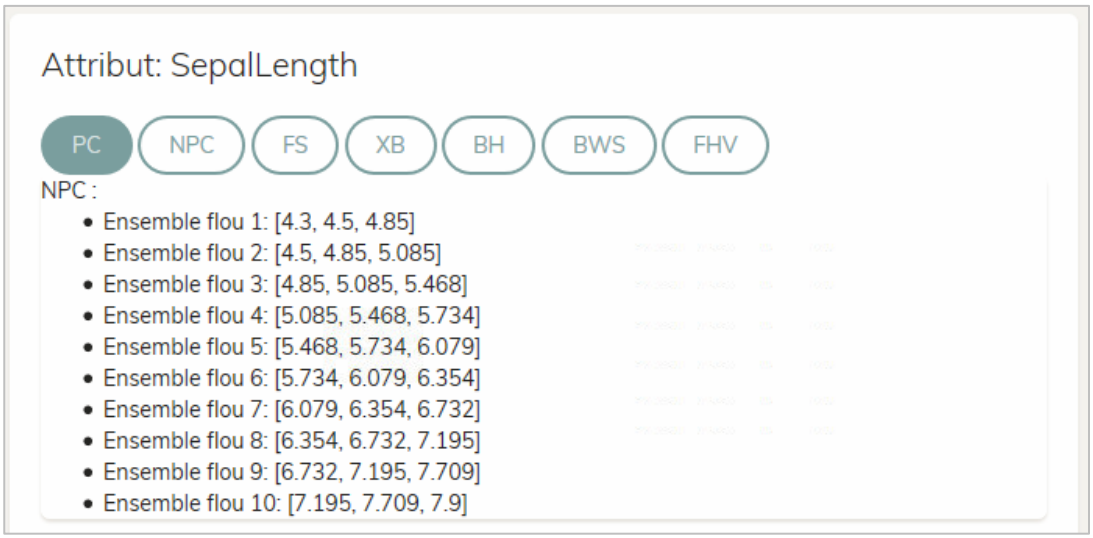

Figure26 : Exemple des valeurs des ensembles flous

3. Expérimentation

3.1. Données utilisées

Un ensemble des bases de données bien connus, depuis KEEL DATASET [42], sont utilisés dans toutes les expériences. Les bases de données sélectionnées représentent diverses difficultés du classement et sont décrits par un ensemble de caractéristiques réels ou nominaux et un certain nombre de classe prédéfinis.

Le tableau 2 résume les principales caractéristiques des bases de données. Pour chaque base de données on a : le nom (Name), le nombre d'exemples (#Patterns), le nombre d'attributs(#Attributes) et le nombre d'étiquettes de classe (#Class) sont indiqués.

Name	#Patterns	#Classes	#Attributes (R,N)
Appendicitis	106	2	(7,0)
Australian	690	2	(3,11)
Balance	625	3	(4,0)
Ecoli	336	8	(7,0)
Iris	150	3	(4,0)
Pima	768	2	(8,0)
Saheart	462	2	(5,4)
Sonar	208	2	(60,0)
Wine	178	3	(13,0)
Wiscounsin	699	2	(9,0)
Zoo	101	7	(0,16)

Tableau 2 : Tableau des bases de données utilisés

Pour mener les expériences, nous avons utilisé la procédure de validation croisée par 10 (10-fold cross validation) [43].Le principe est que les ensembles de données sélectionnés sont partitionnés en dix sous-ensembles de même taille et préservant la même répartition de classe entre les partitions. Le classificateur est ensuite appliqué aux différentes partitions où une partition est sélectionnée en tant qu'ensemble de test et l'ensemble d'apprentissage est composé du reste. Les résultats finaux par jeu de données sont obtenus en faisant la moyenne des résultats obtenus sur les dix partitions.

3.2. Mesures utilisées

3.2.1. Taux de classement TCl

Le taux de classement est calculé par :

$$TCl = \frac{Nombre\ d'instances\ classés\ correctement}{Nombre\ d'instances} * 100$$

Taux du classement de la méthode proposée pour chaque indice de validité (Tableau 3) :

	PC	NPC	FHV	FS	XB	BH	BWS
Appendicitis	**80.3636**	74.6364	77.4545	78.4545	77.4545	77.5455	78.4545
Australian	63.0435	69.8551	75.6522	67.3913	74.7826	**78.9855**	78.5507
Balance	53.7353	51.9946	48.0137	48.7992	52.6195	**79.6765**	48.6611
Ecoli	32.5045	35.9982	34.2692	51.5241	30.1248	**52.4153**	30.7041
Iris	**92.1481**	91.037	92.0741	86.5926	90.2222	85.037	81.8519
Pima	64.3322	64.0655	**74.49**	65.3629	66.2771	71.1081	62.5073
Saheart	60.1573	61.4755	**65.1619**	64.6994	62.1138	61.4709	57.3774
Sonar	**58.2619**	55.8571	53.9286	48.4524	53.4048	53.9048	57.2619
Wine	**76.8954**	75.7843	74.6405	75.1961	72.9085	75.7843	73.5294
Wiscounsin	90.3296	90.9267	80.7958	79.9584	90.641	**90.931**	79.9345
Zoo	86.1667	85.7222	**91.0**	86.7222	84.1944	87.5833	82.6944

Tableau 3 : Tableau de taux du classement des bases de données avec différents indices de validité

Le tableau suivant représente le taux du rapport des différentes méthodes sur les différentes bases de données (Tableau 4) :

	KNN	ALLKNN	CNN	ENN
Appendicitis	90.63	90.72	89.81	**91.63**
Australian	**72.31**	71.44	70.57	65.79
Balance	90.87	91.20	86.07	**91.35**
Ecoli	85.73	**87.21**	64.65	79.76
Iris	**99.33**	**99.33**	90.66	**99.33**
Pima	**78.39**	72	75.51	71.74
Saheart	**71.66**	68.19	70.58	67.33
Sonar	**86.97**	84.57	84.57	82.16
Wine	**83.13**	75.22	81.99	74.67
Wiscounsin	**97.97**	**97.97**	98.11	**97.97**
Zoo	**99.0**	95.66	80.32	85.36

Tableau 4 :Taux de classement des méthodes KNN,ALLKNN,ENN,CNN

Le graphe suivant (Figure 27) montre le taux de classement de la méthode proposée et d'autres méthodes :

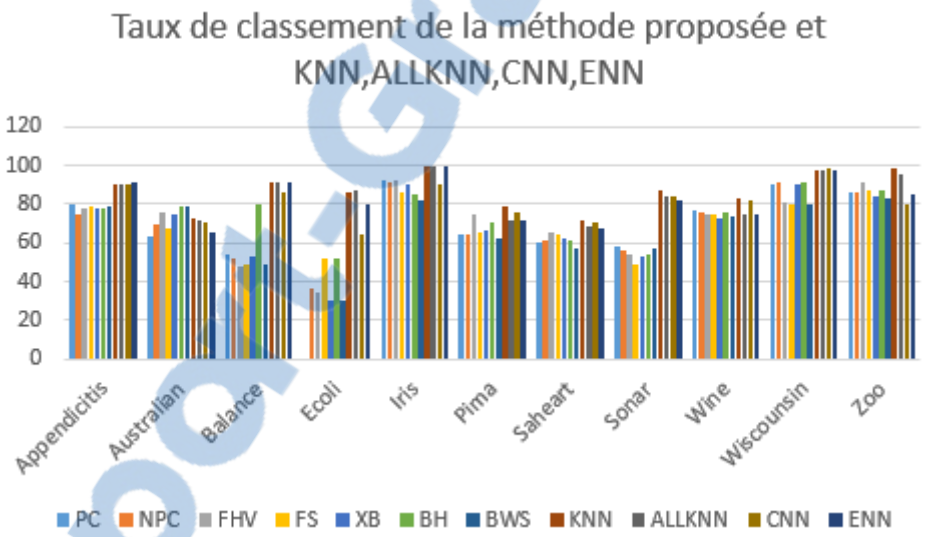

Figure 27 : Graphe de taux de classement

3.2.2. Taux de Réduction TRé

Le taux de réduction est calculé par la formule suivante :

$$TR\acute{e} = 1 - \frac{Nombre\ d'instaces\ généres}{Nombres\ d'instances}$$

Le tableau suivant représente le taux du rapport des différentes méthodes sur les différentes bases de données (Tableau 5) :

	ALKNN	ENN	CNN	Méthode proposée
Appendicitis	0.29	0.22	0.56	0.12
Australian	0.19	0.36	0.30	**0.81**
Balance	0.31	0.28	0.69	**0.96**
Ecoli	0.20	0.39	**0.90**	0.88
Iris	0.04	0.08	0.64	**0.70**
Pima	0.37	0.33	0.40	**0.92**
Saheart	0.33	0.36	0.38	**0.86**
Sonar	0.10	0.14	**0.33**	-0,80
Wine	0.23	0.27	**0.51**	0.28
Wiscounsin	0.02	0.02	0.61	**0.93**
Zoo	0.04	0.13	0.51	-0.20

Tableau 5 : Taux de réduction entre ALLKNN,ENN,CNN et la méthode proposée

Le graphe suivant (Figure 28) montre le taux de réduction de la méthode proposée et d'autres méthodes :

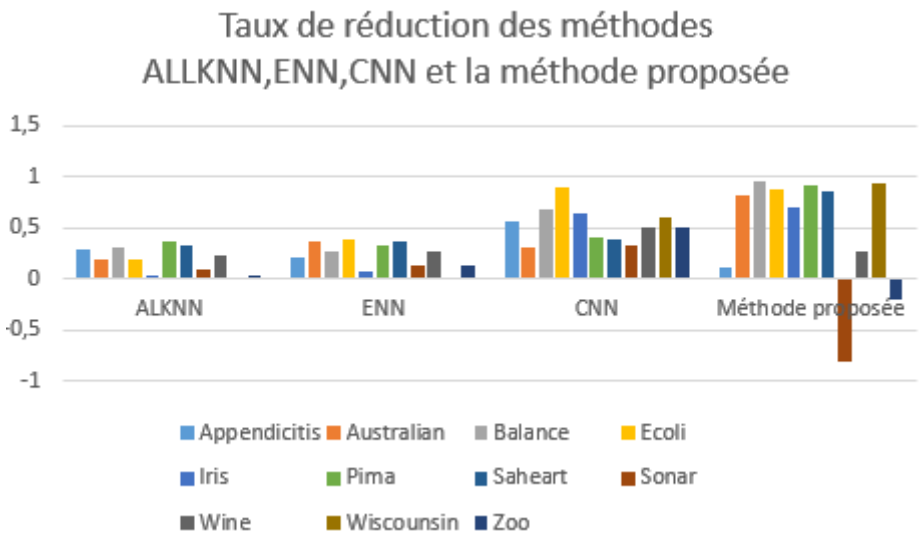

Figure 28 : Graphe de taux de réduction de la méthode proposée et autres méthodes

3.2.3. Rapport de classement et réduction

La formule du rapport est utilisée est :

$$T = TR\acute{e} * \frac{TCl}{100}$$

Le tableau suivant représente le taux du rapport des différentes méthodes sur les différentes bases de données (Tableau 6) :

	ALLKNN	ENN	CNN	Méthode proposée
Appendicitis	0,263088	0,201586	**0,502936**	0,093055
Australian	0,135736	0,236844	0,21171	**0,639783**
Balance	0,28272	0,25578	0,593883	**0,764894**
Ecoli	0,17442	0,311064	0,58185	**0,461255**
Iris	0,039732	0,079464	0,580224	**0,595259**
Pima	0,2664	0,236742	0,30204	**0,654195**
Saheart	0,225027	0,242388	0,268204	**0,52865**
Sonar	0,08457	0,115024	**0,279081**	-0,43124
Wine	0,173006	0,201609	**0,418149**	0,212196
Wiscounsin	0,019594	0,019594	0,598471	**0,845658**
Zoo	0,038264	0,110968	**0,409632**	-0,17517

Tableau 6: Taux de rapport T entre ALLKNN,ENN,CNN et la méthode proposée

Le graphe suivant (Figure 29) montre le taux du rapport T de la méthode proposée et d'autres méthodes :

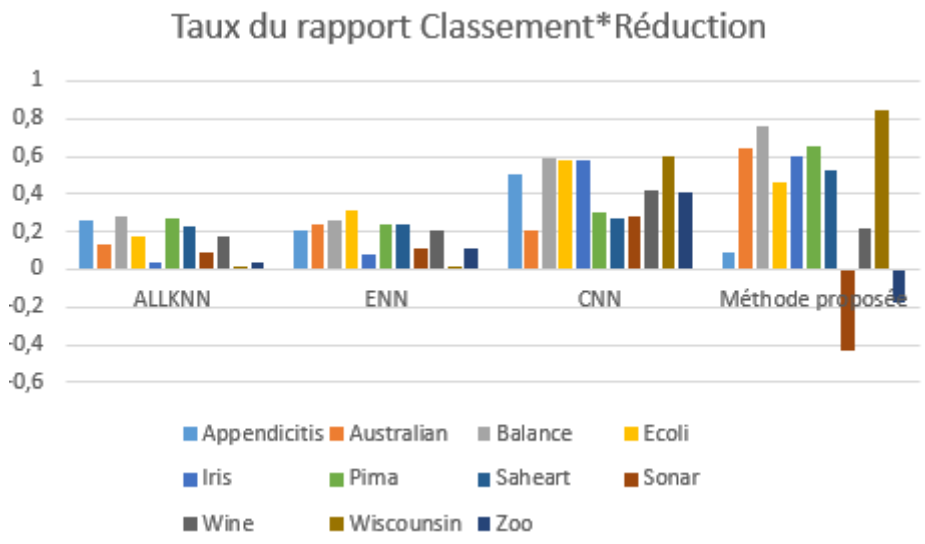

Figure 29 : Graphe du taux de rapport T de la méthode proposée et autres méthodes

Conclusion

La méthode proposée nous a donné de bons résultats par rapport au taux de réduction trouvé, Pour les bases de données Sonar et Zoo on a trouvé des valeurs négatives car le nombre des instances est petit par rapport au nombre d'attributs.

Conclusion générale

Dans ce travail de master, nous avons traité certaines faiblesses de l'approche de classement K-NN. Nous avons ciblé en particulier deux problématiques l'efficacité et la tolérance aux incertitudes. Le problème d'efficacité apparait en raison d'un besoin élevé en matière de stockage et de temps d'exécution dans le cas des données massives. Pour répondre à ce besoin, nous avons utilisé l'approche des plus proches prototypes. Les prototypes sont par de l'algorithme de regroupement Fuzzy C-Means et les indices de validité pour choisir le nombre adéquat de prototypes. D'un autre côté, nous avons utilisé la logique floue pour répondre à la problématique de la gestion des incertitudes dans le processus de classement par KNN. En effet, nous avons utilisé une mesure de similarité floue qui permet d'évaluer la similarité entre les instances pendant le processus de classement.

Nous avons implémenté la méthode proposée dans l'environnement Python et mener des expérimentations sur un ensemble de base de données connues. Les résultats de simulation obtenus ont montré que notre méthode est prometteuse, particulièrement en termes d'interprétabilité.

Cette méthode peut évoluer si on joue sur les fonctions d'appartenance ou l'introduction de nouveau notions tel que l'entropie.

Références

[1] J. E. . Akinsola, F. . Osisanwo, A. O, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised Machine Learning Algorithms: Classification and Comparison," *Int. J. Comput. Trends Technol.*, vol. 48, no. 3, pp. 128–138, 2017.

[2] R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst. Man. Cybern.*, vol. 21, no. 3, pp. 660–674, 1991.

[3] L. BREIMAN, "Random forests," *Ensemble Mach. Learn. Methods Appl.*, pp. 5–32, 2001.

[4] K. P. Murphy, "Naive Bayes classifiers.pdf," pp. 1–8, 2006.

[5] a.T.C. Goh, "Back-propagation neural networks for modeling complex systems," *Artif. Intell. Eng.*, vol. 9, pp. 143–151, 1995.

[6] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[7] S. Chattopadhyay, S. Banerjee, F. A. Rabhi, and U. R. Acharya, "A Case-Based Reasoning system for complex medical diagnosis," *Expert Syst.*, vol. 30, no. 1, pp. 12–20, 2013.

[8] A. Idri and A. Abran, "Fuzzy Analogy : A New Approach for Software Cost Estimation," *11th Int. Work. Softw. Meas.*, pp. 93–101, 2001.

[9] P. Arundacahawat, R. Roy, and A. Al-Ashaab, "An analogy based estimation framework for design rework efforts," *J. Intell. Manuf.*, vol. 24, no. 3, pp. 625–639, 2013.

[10] D. W. AHA, D. KIBLER, and M. K. ALBERT, "Instance-Based Learning Algorithms," *Mach. Learn.*, vol. 6, pp. 37–66, 1991.

[11] B. Bouchon-Meunier, J. Delechamp, C. Marsala, and M. Rifqi, "Analogy as a Basis of Various Forms of Approximate Reasoning," pp. 70–79, 2000.

[12] R. Fullér and P. Majlender, "An analytic approach for obtaining maximal entropy OWA operator weights," *Fuzzy Sets Syst.*, vol. 124, no. 1, pp. 53–57, 2001.

[13] M. Cover T and E. Hart P, "Nearest Neighbor Pattern Classification," *IEEE Trans. Inf. Theory*, pp. 1–12, 1967.

[14] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based Classification: Concepts and Algorithms," *J. Mach. Learn. Res.*, vol. 10, pp. 747--776, 2009.

[15] A. Aamodt and E. Plaza, "Case-Based Reasoning : Foundational Issues , Methodological Variations , and System Approaches , as presented by Praveen Guddeti," *AI Commun.*, vol. 7, no. 1, pp. 39–59, 1994.

[16] L. I. Kuncheva and J. C. Bezdek, "Nearest prototype classification: clustering, genetic algorithms, or random search?," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 28, no. 1, pp. 160–164, 1998.

[17] X. Wu *et al.*, *Top 10 algorithms in data mining*, vol. 14, no. 1. 2008.

[18] J. Gou, T. Xiong, and Y. Kuang, "A novel weighted voting for K-nearest neighbor rule," *J. Comput.*, vol. 6, no. 5, pp. 833–840, 2011.

[19] C. T. Su, L. S. Chen, and Y. Yih, "Knowledge acquisition through information granulation for imbalanced data," *Expert Syst. Appl.*, vol. 31, no. 3, pp. 531–541, 2006.

[20] P. Luukka and O. Kurama, "Similarity classifier with ordered weighted averaging operators," *Expert Syst. Appl.*, vol. 40, no. 4, pp. 995–1002, 2013.

[21] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based Classification: Concepts and Algorithms," *J. Mach. Learn. Res.*, vol. 10, no. March, pp. 747--776, 2009.

[22] A. K. Ghosh, "On optimum choice of k in nearest neighbor classification," *Comput. Stat. Data Anal.*, vol. 50, no. 11, pp. 3113–3123, 2006.

[23] J. Derrac, S. García, and F. Herrera, "Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects," *Inf. Sci. (Ny).*, vol. 260, pp. 98–119, 2014.

[24] S. García, J. Derrac, J. R. Cano, and F. Herrera, "Prototype Selection for Nearest Neighbor Classification: Survey of Methods," *Sci2S.Ugr.Es*, pp. 1–28, 2010.

[25] W. D Randall and M. Tony R, "Reduction Techniques for Instance-Based Learning Algorithms," *Mach. Learn.*, vol. 38, pp. 257–286, 2000.

[26] I. Triguero and F. Herrera, "Prototype Generation for Nearest Neighbor Classification: Survey of Methods," pp. 1–19, 2012.

[27] S. García, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 417–435, 2012.

[28] P. Hart, "The condensed nearest neighbor rule (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 515–516, 2004.

[29] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Trans. Syst. Man Cybern.*, vol. 2, no. 3, pp. 408–421, 1972.

[30] I. Triguero, J. Derrac, S. García, and F. Herrera, "A taxonomy and experimental study on prototype generation for nearest neighbor classification," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 1, pp. 86–100, 2012.

[31] L. A. Zadeh, "Fuzzy sets," *Elsevier Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[32] J. Zhao and B. K. Bose, "Evaluation of membership functions for fuzzy logic controlled induction motor drive," *IEEE 2002 28th Annu. Conf. Ind. Electron. Soc. IECON 02*, pp. 229–234, 2002.

[33] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning-I," *Inf. Sci. (Ny).*, vol. 8, no. 3, pp. 199–249, 1975.

[34] J. C. Bezdek, F. William, and R. EHRLICH, "FCM: THE FUZZY c-MEANS CLUSTERING ALGORITHM," vol. 10, no. 2, pp. 1783–1784, 2005.

[35] R. Suganya and R. Shanthi, "Fuzzy C-Means Algorithm-A Review," *Int. J. Sci. Res.*, vol. 2, no. 11, pp. 1–3, 2012.

[36] W. Wang and Y. Zhang, "On fuzzy cluster validity indices," *Fuzzy Sets Syst.*, vol. 158, no. 19, pp. 2095–2117, 2007.

[37] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, NewYork, 1981.* 1383.

[38] J. C. Bezdek, "Cluster validity with fuzzy sets," *J. Cybern.*, vol. 3, no. 3, pp. 58–73, 1973.

[39] X.L. Xie and G. Beni, "Xie - 1991 - A validity measure for fuzzy clustering.pdf." .

[40] Gath I. and Geva A. B., "Unsupervised optimal fuzzy clustering,IEEE Trans. Pattern Anal. Mach. Intell. 11 (1989) 773–781.," *IEEE Trans. pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 773– 781, 1989.

[41] I. Beg and S. Ashraf, "Similarity measures for fuzzy sets," *Appl. Comput. Math.*, vol. 8, no. 2, pp. 192–202, 2009.

[42] A. F. Ernández, J. L. Uengo, and J. D. Errac, "KEEL: A software tool to assess evolutionary algorithms for Data Mining problems (regression, classification, clustering, pattern mining and so on)," *J. Mult. Log. Soft Comput.*, vol. 17, pp. 255–287, 2011.

[43] M.Stone, "Cross-Validatory Choice and Assessment of Statistical Predictions," *J. R. Stat. Soc. Ser. B (Methodological) Banner*, pp. 111–147, 1974.

Génération des prototypes Basée sur FCM pour l'algorithme KNN

Résumé

Le classement des voisins le plus proche (NNC) est largement utilisé comme méthode du classement, en raison de sa simplicité, de son efficacité et de sa capacité à traiter différents problèmes du classement.

Malgré sa bonne précision du classement, le NNC souffre de nombreux problèmes en termes de temps d'exécution, de sensibilité au bruit, de contraintes de stockage élevées et de manque d'interprétabilité. Dans cet article, nous proposons un algorithme d'apprentissage de prototype. Cet algorithme dérive un petit sous-ensemble de prototypes pertinents de l'apprentissage en utilisant des algorithmes de regroupement. Les prototypes obtenus permettent de traiter les problèmes mentionnés. De plus, l'utilisation de l'algorithme du classement implique un ensemble interprétable de prototypes pouvant décrire le domaine d'application.

Mots clés : Approche des plus proches voisins, Méthodes de Réduction, Logique Floue, Génération des prototypes.

FCM Based prototype generation for KNN

Abstract

The Nearest Neighbor Classification (NNC) has been widely used as a classification method because of its simplicity, efficiency, and ability to handle different classification problems. Despite its good classification accuracy, the NNC suffers from numerous problems in terms of execution time, noise sensitivity, high storage constraints and lack of interpretability. In this article, we propose a prototype-learning algorithm. This algorithm derives a small subset of relevant prototypes of learning using clustering algorithms. The prototypes obtained make it possible to treat the mentioned problems. In addition, the use of the classification algorithm involves an interpretable set of prototypes that can describe the application domain.

Keywords: Nearest Neighbor Approach, Reduction Methods, Fuzzy Logic and Prototype Generation.