Table des matières

Liste des tableaux

Liste des figures

1. Introduction

En 2019, près de 97,2 % des dirigeants de 65 firmes les plus importantes de notre époque confirment investir massivement dans les technologies dites Big Data. Parmi ces 65 firmes, 73,8 % d'entre elles sont des services financiers. À la question est-ce que ces nouvelles technologies vont impacter le monde de la finance ? Nous pouvons d'ores et déjà répondre par l'affirmative, mais quels vont être concrètement les impacts sur ces métiers ? Nous en venons naturellement à poser la problématique, ligne conductrice de notre travail : Comment les technologies Big Data et la Data Science vont-elles façonner le futur de la finance moderne ?

Afin de répondre à cette question, il nous semblait pertinent de nous intéresser aux deux écosystèmes que sont la finance et la Data Science mais surtout à la pertinence de leur mélange. Ces deux mondes sont désormais liés et nous allons tenter de démontrer comment. Dans ce travail, la combinaison des deux s'effectuera à travers un développement informatique où les données financières rencontreront les technologies Big Data afin d'effectuer une analyse financière classique sur le marché actions américain.

"Big Data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it…" (Ariely, 2013). Les motivations de ce travail proviennent d'une part du constat très bien résumé par la citation précédente. Qu'est-ce que le Big Data ? Y a-t-il une manière de définir ce terme facilement ? Où se trouve la véritable révolution ? Qui comprend réellement ce concept ? Quelle est la réelle plus-value de ces nouvelles technologies ? D'autre part, je suis étudiant en finance, mais profondément passionné par l'informatique et la combinaison des deux est parfaitement alignée avec mes objectifs professionnels. L'informatique fait partie de la Finance d'aujourd'hui et de demain. Rechercher et apporter du nouveau contenu sur ces thématiques ne pouvait que me motiver.

L'objectif de ce travail sera donc de démontrer la complémentarité de ces deux mondes. Afin de répondre à cet objectif et à notre problématique, nous avons tout d'abord privilégié une approche théorique. Ainsi, nous avons effectué une recherche approfondie dans la littérature disponible sur le sujet. Plus de 120 sources jugées pertinentes ont été analysées afin d'apporter des clarifications sur les termes et la portée de cette révolution. Puis nous avons complété ce travail théorique avec une approche plus empirique par la création d'un environnement de développement informatique Big Data et l'écriture d'algorithmes typiquement utilisés en finance.

Le travail comporte quatre parties avec chacune son axe de développement. Dans un premier temps, nous souhaitions commencer par une vue générale sur le sujet. En effet, nous commençons par définir les termes Big Data et Data Science, mais nous définissons également le contexte dans lequel ces technologies apparaissent. Qu'est-il important de garder en tête quand nous évoluons dans ces nouveaux domaines ? Le but de cette première partie est également d'apporter des clés de compréhension aux lecteurs afin qu'ils puissent mieux appréhender ce qu'est un département de Data Science. Ces unités commencent à voir le jours dans de nombreuses entreprises et les services financiers ne font pas exception. Il semblait important de définir le background et les rôles de ce type d'équipes car elles sont au cœur de cette évolution. Dans un second temps, notre travail revêt un aspect plus technique et nous rentrons dans la définition des technologies utilisées lors de notre développement informatique. Nous justifions leurs apparitions et énonçons pourquoi elles se trouvent être les plus adaptées à l'exploitation du Big Data. Dans un troisième temps, le travail bifurque vers le second pan de ce travail : la finance. Quel est son actuel environnement technologique ? Pourquoi le Big Data s'impose-t-il dans ce milieu ? Nous décrivons également les types de données exploitables au sein des services financiers et les analysons à travers les jalons posés lors de notre première partie. La dernière sous-partie consacrée à la finance décrit une dizaine de « uses cases » de la Data Science appliquée à la finance. Cette partie est primordiale, car elle apporte beaucoup d'éléments de réponse à notre problématique. Quelles sont les opportunités qu'offrent le Big Data à la finance ? Même si la définition de ces cas d'utilisations reste succincte, nous avons tenté d'englober les principes les plus importants à travers un framework proposé par Oracle. Finalement, l'ultime partie de ce travail concerne le développement informatique. Quel est le but du développement ? Quel est l'environnement mis en place ? Quelles sources de données choisies ? En conclusion, nous présenterons les résultats. Tout d'abord, nous aborderons notre développement d'un point de vue purement technique afin de déterminer la pertinence de nos choix technologique. Puis nous nous mettrons dans la peau d'un analyste financier pour démontrer que grâce à notre développement nous avons effectivement générer des informations exploitables stratégiquement.

Il est désormais temps de commencer notre grand plongeon dans le lac de données qui s'offre à nous.

2. Définition des termes & Contexte

2.1 Le Big Data

Le Big Data est un terme flou pour une grande partie des professionnels et particuliers. Que regroupe ce terme ? Les données que j'exploite peuvent-elles être légitimement classées dans cette catégorie ? Quelles sont les caractéristiques intrinsèques du Big Data ?

Toutes ces questions trouvent souvent différentes réponses dans la littérature. Pour y répondre et définir le terme, Big Data, nous allons tout d'abord nous attarder sur 3 des nombreuses lois technologiques (Delort, 2018) de notre ère. Ces dernières régissent le rythme et l'envergure des évolutions informatiques, il est donc pertinent de les étudier. Puis nous analyserons brièvement les différentes sources de données et leurs classifications pour finir sur la décomposition d'un « mythe », celui que nous appellerons le mythe des 3 V.

2.1.1 Les lois technologiques

La première serait la loi de Moore (Moore, 1965) du nom de son inventeur, Gordon Moore fondateur de Fairchild Semiconductor. Cette dernière implique un doublement de la densité des transistors dans les semi-conducteurs tous les deux ans. Une plus grande densité de ces composants induit une plus grande puissance de calcul. La seconde loi est similaire, mais concerne la densité de l'espace de stockage sur les disques magnétiques (Walter, 2005). Selon Mark Kryder, ingénieur fondateur de la Carnegie Mellon University's Data Storage Sytems Center et CTO de Seagate Technology, la densité de stockage sur les disques durs devrait doubler tous les 18 mois environ. La dernière de ces lois concerne les bandes passantes des réseaux professionnels (Nielsen, 1998). Tous les ans, la bande passante de ces réseaux augmenterait de 50 % selon Jakob Nielsen Ph. D en Interaction Homme machine et détenteur 79 brevets sur le sujet.

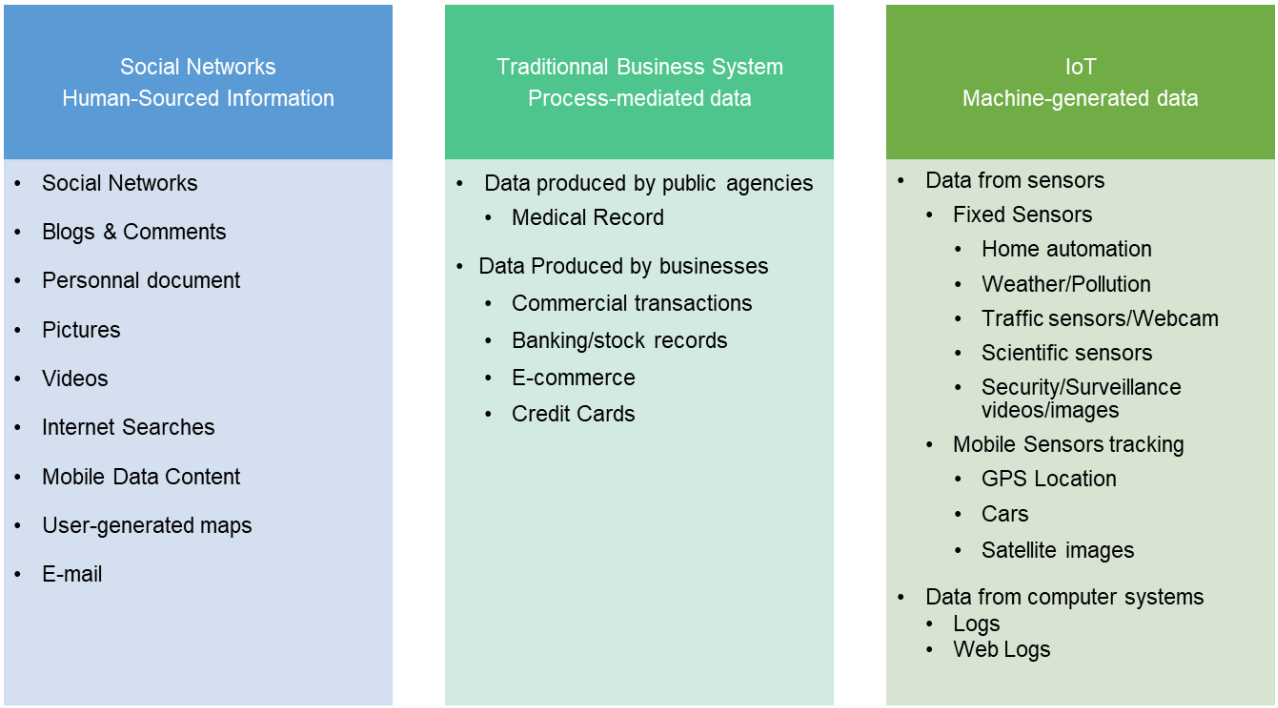
La loi de Moore, de Kryder, de Nielsen démontre le rythme effréné de nos évolutions technologiques. Pour beaucoup, ces évolutions sont la genèse du Big Data. Sans une puissance de calcul considérable, un espace de stockage suffisant et une bande passante performante, l'apparition du Big Data et son exploitation n'aurait pu être possible.

2.1.2 Les sources de données

Ces avancées technologiques ont permis l'apparition de nombreuses sources de données aussi diverses que variées. La classification de ces différentes sources par l'UNECE (Vale, 2013), la commission économique pour l'Europe des Nations Unies, met en exergue 3 grandes classes de sources de données :

(Vale, 2013)

Figure 1: UNECE Classification of Big Data Types

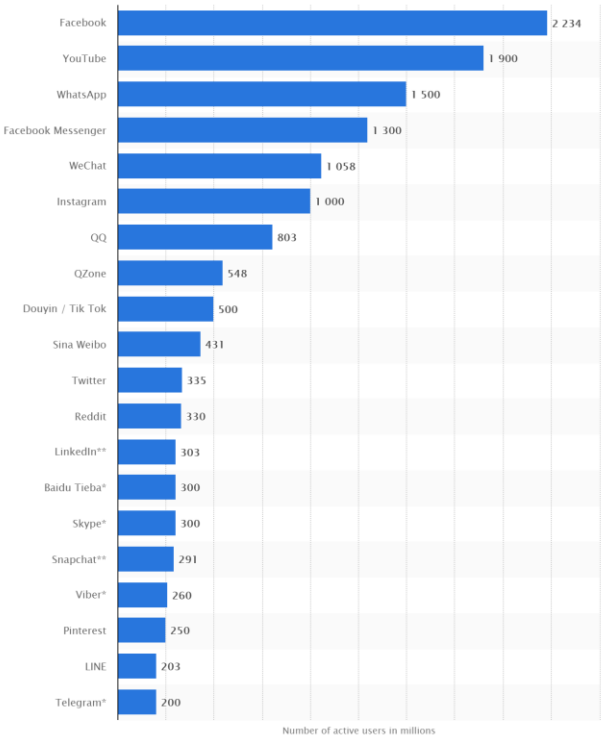


"There are 2.5 quintillion bytes of data created each day at our current pace" […] *"Over the last two years alone 90 percent of the data in the world was generated"* (Marr, 2018). Ces simples phrases résument parfaitement l'explosion de données à laquelle nous faisons face. Cette abondance de donnée est dû à la démocratisation massive des 3 types de sources de données citées précédemment et à l'évolution technologique dont les 3 lois technologiques précitées sont également le symbole. Selon différents rapports de Statista (Statista, 2019), Facebook compte désormais plus de 2 milliards d'utilisateurs et le cabinet de consulting McKinsey (Manyika et al., 2011) reporte 30 milliards de pièces de contenu publiées sur ce même réseau social tous les mois. Ces chiffres démontrent la démocratisation massive des réseaux sociaux. Leurs expansions mondiales soutenues par l'explosion du marché des smartphones génèrent des quantités de données structurées et non structurées inégalées jusqu'alors. En effet, en 2017 le nombre d'abonnés mobile a dépassé la barre des 5 milliards (GSMA, 2018) soit près de deux tiers de la population mondiale et ce chiffre tendrait à atteindre près de 6 milliards dans les prochaines années. En outre, un rapport de l'entreprise Cisco fait l'état du nombre d'objet connecté par personne. Ce ratio passerait de 0,04 par personne en 2013 à près de 7 en 2020 (Evans, 2011). Cisco explique également que l'IoT est utilisé dans tous les secteurs : la production, les transports, la distribution... L'IoT permet de gérer diverses tâches tel que l'inventaire des stocks, l'« asset tracking », le « fleet maangement » ou encore la

facturation en temps réel (Biren Gandhi, 2015). La 4e révolution industrielle ou l'industrie 4.0 n'est pas simplement un concept. Ces transformations sont des réalités.

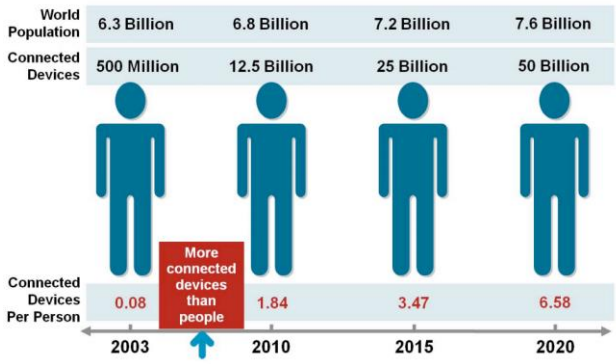
(Statista, 2019)

Figure 2: Most popular social networks as of October 2018



(Evans, 2011)

Figure 3: Number of connected devices per Person



La grande différence apportée par ces nouvelles sources est la multiplication des données publiques de types non structurées et semi-structurées (Biernat, Lutz, 2015). Auparavant les données étaient le plus souvent des données privées indexées de manière « classique » dans des bases de données ou générées par des ERP. Désormais les réseaux sociaux, les moteurs de recherches fournissent une abondance de données publiques non structurées à travers des API, du texte, des vidéos, etc. Ces nouvelles

données sont plus complexes à exploiter et ont amené au développement des nouvelles technologies dont nous reparlerons dans ce travail.

Tableau 1 : Les différents niveaux de structuration des données

Niveau de structuration	Modèle de données	Exemples	Facilités de traitement
Structurée	Système de données relationnelles objet/colonne	BdD d'entreprise	Facile (indexé)
Semi-structurée	XML, JSON, CSV, Logs	API Google, API Twitter, web, logs…	Facile (non indexé)
Non structurée	Texte, Images, Vidéos	Post, e-mails, documents	Complexe

(Biernat, Lutz, 2015)

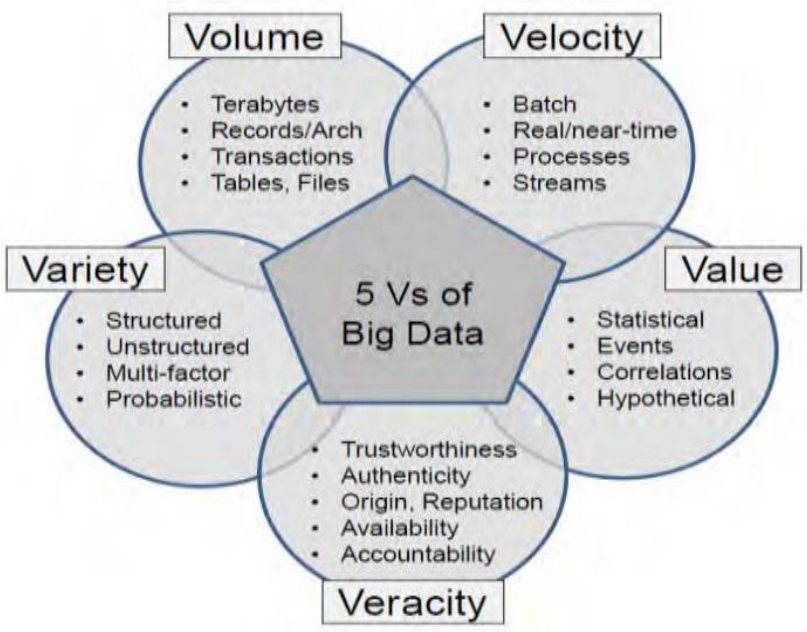
En résumé, les sources de données se multiplient à une telle vitesse que la quantité d'informations générées dépassent les mesures traditionnelles pour atteindre des valeurs difficilement appréhendables pour l'esprit humain. De plus, la structure de ces nouvelles données impose une gestion plus gourmande en ressources et a poussé à l'apparition de nouvelles méthodes. Grâce à ces phénomènes, le terme Big Data à commencer à émerger et différents auteurs se sont confrontés à le définir.

2.1.3 Le mythe des 3 V

C'est en 2001 que le « mythe » des 3 V fait son apparition dans le monde professionnel (Laney, 2001). En effet, Doug Laney, actuellement VP Analyst chez Gartner, publie un article décrivant le changement de paradigme qui s'opèrerait dans le « Data Management » des nouveaux E-Business. En deux pages, Doug Laney décompose la révolution des données en 3 V : Volume, Velocity, Variety. Cependant, le terme Big Data n'est pas utilisé une seule fois par l'auteur primé de nombreuse fois dans le milieu de la Data Science. Pourtant, ces 3 V sont désormais constamment assimilés au Big Data. Plus tard IBM ajoutera le V de Veracity (Kobielus, 2013), puis Oracle celui de Value (Cackett et al., 2013).

(Ishwarappa, Anuradha, 2015)

Figure 4 : 5 vs of Big Data



Dans une publication de l'International Research Journal of Engineering and Technology (IRJET), les auteurs vont jusqu'à décomposer le Big Data en 17 caractéristiques commençantes par V (Arockia Panimalar et al., 2017). Parmi elles, ils s'ajoutent Volatility, Virality, et Variability. Ainsi à travers ces différentes lectures nous pouvons admettre que la définition dépend beaucoup des volontés des auteurs et de leurs propres expériences comme utilisateurs de ce genre de données.

En 2016 Rob Kitchin et Gavin McArdle proposent une analyse de 26 datasets considérées comme Big Data par la majorité des professionnels. Ces données vont des données mobiles, aux recherches internet, aux données de la CCTV, etc. Ils ont passées ces 26 datasets à travers 7 traits qu'ils ont déterminés à travers leurs recherches dans lequel nous retrouvons les 3 V de Doug Laney. Le but de cette analyse était de trouver les caractéristiques discriminantes de ces datasets. Qu'ont-elles en communs ?

Tableau 2: Kitchin Traits to evaluate datasets

	Small Data	**Big Data**
Volume	Limited to large	Very Large
Velocity	Slow, Freeze-framed, bundled	Fast, Continuous
Variety	Limited to wide	Wide
Exhaustivity	Samples	Entire population
Resolution & Indexcality	Course and weak to tight and strong	Tight and strong
Relationality	Weak to strong	Strong
Extensionality & Scalability	Low to Middling	High

(Kitchin, McArdle, 2016)

En conclusion de leurs analyses les auteurs définissent l'« exhaustivity » et la « velocity » comme les traits déterminants de la classification des datasets. L'exhaustivité se réfère la taille de l'échantillon de donnée capturée versus la population entière alors que la velocity se réfère à la vitesse de génération des données pouvant aller de ponctuel à l'enregistrement en temps réel. *« Small data are slow and sampled. Big Data are quick and n=all »* (Kitchin, McArdle, 2016). Les autres traits ne sont pour eux qu'optionnels et par exemple le volume n'est pas une caractéristique nécessaire. Les relevés de pollutions sont enregistrés via des capteurs de manières constantes et en temps réel, mais ne nécessitent pas un stockage démesuré. Grâce à cette classification, les auteurs déterminent également qu'il n'y aurait donc pas qu'un seul type de Big Data, mais plusieurs. Ainsi définir le terme avec des caractéristiques générales ne ferait que réduire l'étendue incommensurable de types de données Big Data que l'humanité génère désormais.

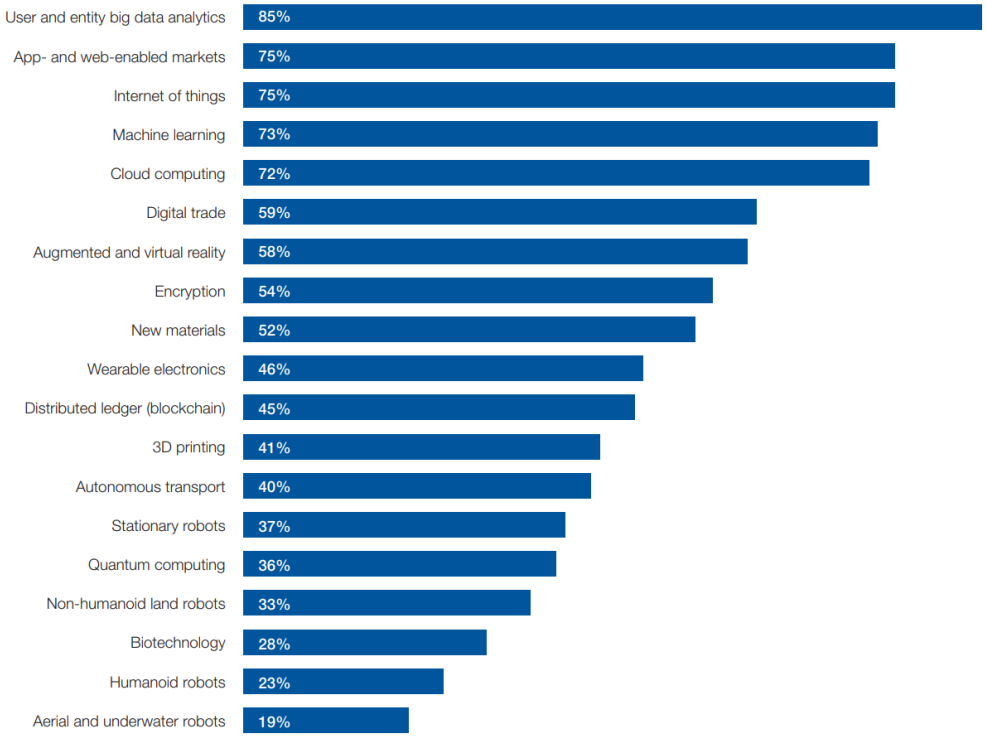
Le Big Data est donc le résultat de nombreuses évolutions technologiques. Ces dernières ont permis la multiplication des sources de données et de leurs structures favorisant ainsi l'expansion du volume total généré par l'humanité. Grâce à cela, de nouvelles technologies et de nouveaux métiers ont vu le jour afin de permettre aux entreprises d'exploiter ces nouvelles sources d'information pour répondre aux nouveaux défis auquel elles font face.

2.2 La Data Science

En 2012, Harvard Business Review qualifiait le Data Scientist comme: *« The Sexiest Job of the 21st century »* (Davenport, Patil, 2012). Un sondage réalisé auprès des professionnels par le World Economic Forum place en première position des technologies à adopter d'ici à 2022, l'*« User and Big Data Analytics »* avec près de 85 % des répondants la plaçant comme une priorité. Le machine-learning est 4e avec 73 % des réponses. Pour les professionnels de la Finance, le classement est similaire avec pour unique différence une troisième place attribuée au machine-learning au coude à coude avec la Blockchain. Le secteur de l'informatique classe le Big Data et le « machine-learning » à la première et deuxième place au coude à coude avec le cloud computing et l'app and web development.

(Till Alexander et al., 2018)

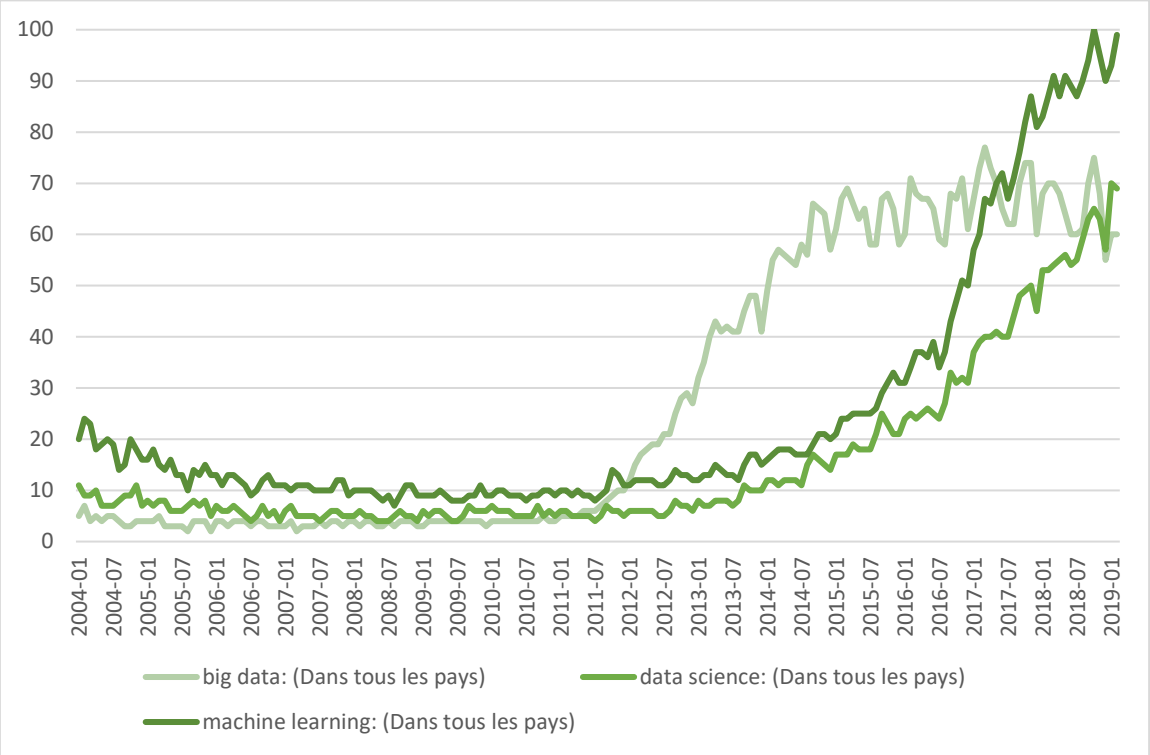
Figure 5: Technologies by proportion of companies likely to adopt them by 2022 (projected)



Plus particulièrement, en Suisse le Swiss Job Index de Michael Page rapporte une hausse de 54,3 % entre juin 2017 et juin 2018 pour la demande de gestionnaires des informations/données. Google Trend permet également de voir l'explosion de popularité de ces nouveaux termes depuis 7 ans dans les recherches des internautes (100 démontre un intérêt maximal).

(Google Trends, 2019)

Figure 6: Data Science Keywords in Google Trends



Tous ces chiffres montrent l'intérêt grandissant des entreprises et des particuliers pour les rois de la Data Science et le nouveau pétrole de notre ère. Qu'est-ce que la Data Science exactement ?

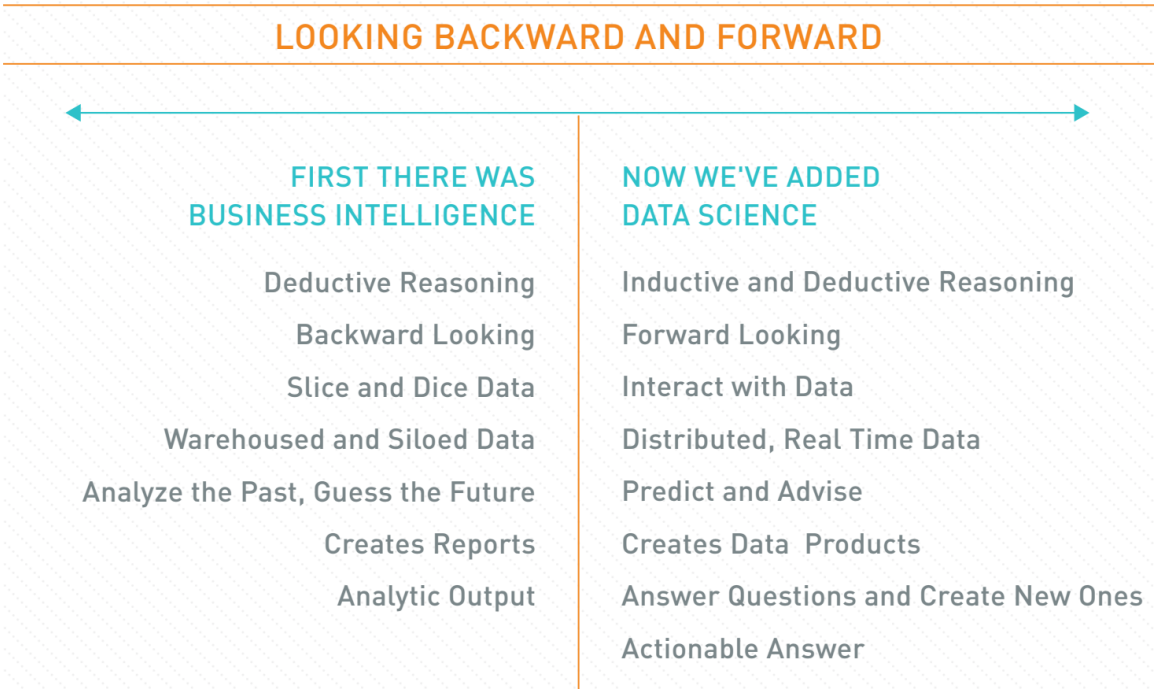
2.2.1 Business Intelligence VS Data Science, une définition de la science des données

> *"Data Science is the extraction of useful knowledge directly from data through a process of discovery, or of hypothesis formulation and hypothesis testing."*(NIST Big Data Public Working Group Definitions and Taxonomies Subgroup, 2015)

Quels éléments différencient la Business Intelligence et la Data Science ? Tout d'abord, la façon de raisonner est diamétralement opposée. Le premier domaine utilise ce que l'on appelle un raisonnement déductif. *« Les Hommes sont mortels, Socrate est un Homme, donc Socrate est mortel »* (Delort, 2018). Dans ce genre de raisonnement, nous partons d'une idée générale, d'une loi pour en tirer des conséquences. Le second domaine, la Data Science, procède avec un raisonnement dit inductif. Ce type de raisonnement permet de partir de faits particuliers, de données, pour en tirer des lois et des principes généraux. De plus, dans le raisonnement inductif, les mathématiques et les statistiques sont utilisées par la suite pour calculer l'incertitude liée aux principes déduits. En bref, *« Partir d'un modèle correspond à adopter un raisonnement déductif, partir des données est adopter un raisonnement inductif, avec une incertitude à apprécier »* (Delort, 2018).

(Booz Allen Hamilton, 2015)

Figure 7: Business Intelligence and Data Science

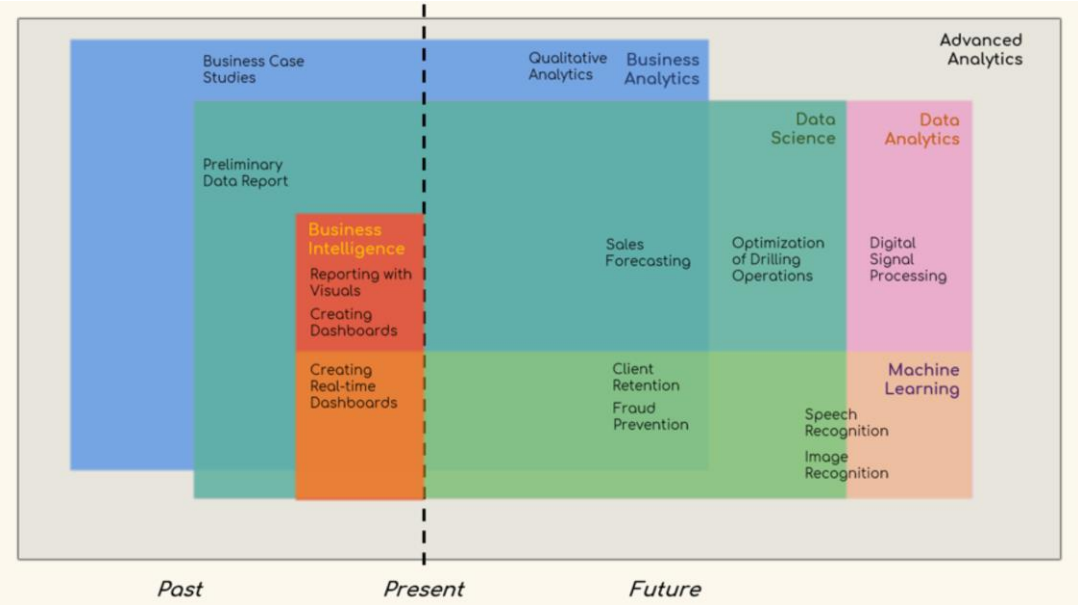


La seconde différence avec l'analyse de donnée orientée BI ou Data Science c'est le résultat final. Alors qu'en Business Intelligence, le produit final serait les rapports et tableaux de bord descriptifs, en Data Science, le « Field Guide Data Science » nous parle de « Data Products » (Booz Allen Hamilton, 2015). Ces produits de données offrent des informations directement exploitables aux utilisateurs/clients sans les exposer aux modèles construits et données utilisées. Les recommandations de films, d'amis, des prédictions de mouvements sur les marchés financiers, ou encore un diagnostic de santé peuvent être assimilé à des « Data Products ». Ainsi la Data Science permet d'être dans l'action et la prédiction là où la Business Intelligence est dans la réaction et dans l'analyse des faits passés pour envisager le futur.

La business intelligence et la Data Science sont complémentaires et la première n'est pas devenu obsolète après la démocratisation de la seconde dans le monde professionnel. La BI fait valoir la *« descriptive analytics »* (Ghosh, 2018) et doit se concentrer sur l'analyse opérationnelle d'une entreprise pour favoriser les décisions stratégiques. La Data Science équivaut, elle, à la *« Predictive »* ou *« Prescriptive Analytics »* (Ghosh, 2018). Elle met en évidence des perspectives et des relations entre les données que l'humain ne peut déceler seules dans un simple tableau de bord et suivi de KPI. Les deux domaines font parties d'un seul et même ensemble ce qu'Oracle appelle l'« Advanced Analytics ».

(Valchanov, 2018)

Figure 8: Advanced Analytics Ecosystem



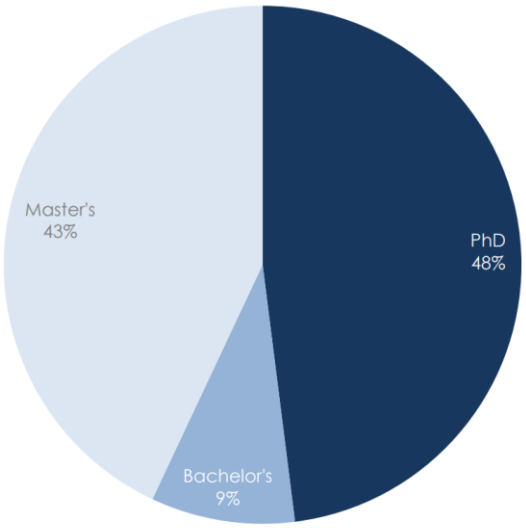
Cependant, pour occuper une place dans un service de Business Intelligence ou de Data Science, les compétences demandées ne sont pas les mêmes. Sans rentrer dans les détails, la BI requiert bien souvent moins de « coding skills » et l'approche avec les statistiques descriptives est la plus souvent utilisée. Les connaissances d'outils standards de visualisations tels que Tableau, QlickView et PowerBI est clairement souhaité, de même que les compétences métiers et proche de l'opérationnel. De plus, les données utilisées en BI sont souvent des données stockées de manières classiques et structurées dans des « Data Warehouses » prévu à cet effet. En Data Science, c'est une tout autre histoire.

2.2.2 Les compétences clés de la Data Science :

Dans un sondage réalisé par BurtchWorks en mai 2018, la répartition en termes de diplôme parmi les professionnels de la Data Science interrogés est la suivante :

(Burtch, 2018)

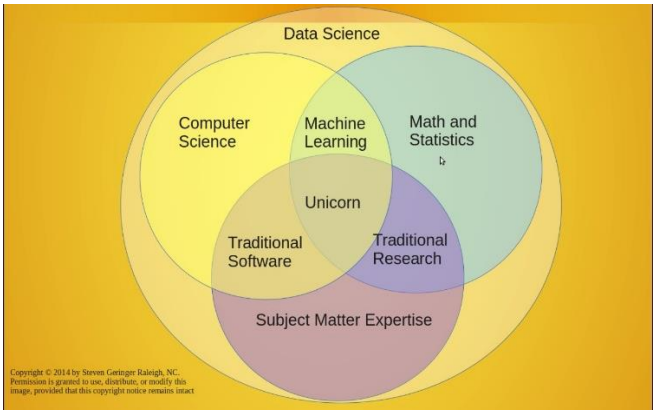
Figure 9: Distribution of Data Scientists by Education



Parmi ces personnes, 25 % ont un diplôme en matière quantitative (mathématiques, statistiques), 20 % en computer science. 20 % en sciences naturelles (life sciences, physique), 18 % sont des ingénieurs et 12 % possèdent un diplôme en Business Administration ou en Économie. Le spectre de profil se révèle donc large et même si les matières quantitatives ont la part belle, il semble tout de même qu'il n'y ait pas qu'un seul type de chemin possible pour travailler dans une équipe de Data Science. Pourquoi des parcours si différents peuvent-ils se retrouver à travailler au sein du même domaine ?

(Geringer, 2014)

Figure 10: Venn Diagram of Data Science



La réponse est simple : la Data Science est une histoire d'équipe et chacun dans son domaine d'expertise apporte de nouvelles perspectives aux différents projets analytiques. Nous pouvons définir l'ensemble des compétences clés nécessaires pour devenir indispensables dans une équipe d'analystes de données. Le diagramme de Venn ci-dessus décrit parfaitement les compétences parfois transversales qu'un professionnel de la Data Science peut travailler en fonction de son propre parcours, mais également de son rôle dans l'équipe. Nous voyons déjà apparaître 3 grands domaines : l'informatique, les mathématiques et statistiques ainsi que les compétences métiers. Au centre de ce diagramme, l'« Unicorn » serait un expert dans chacun des domaines, ce qui est évidemment rare, voire impossible, tant chacun des domaines seuls est déjà très vaste.

Le modèle de classification proposée par la doctorante Dana M. Dedge Parks apporte une certaine granularité au modèle ci-dessus. Le résultat de ces interviews et différents sondages lui ont permis de classifier les compétences des data scientists en 5 grandes catégories regroupées dans le tableau suivant.

Tableau 3: Key Skills for the Data Science

Key Skills	**Descriptions**
Data Analysis Functions	These capabilities are the types of math and analytical methods and modeling skills needed to solve data problems.
Programming & Tools	These are the technical abilities that are required for data scientists to take data and either use a tool or write code to customize a tool.
Machine Learning Techniques and Solutions	These are methods used to review data and build supervised and unsupervised machine learning proficiency.
Interdisciplinary Knowledge	These represent the functions that must exist across academic disciplines such as accounting, economics and information systems.
Critical Abilities	The capabilities listed here are the softer skills that are required for data results to be summarized and communicated.

(Parks, 2017)

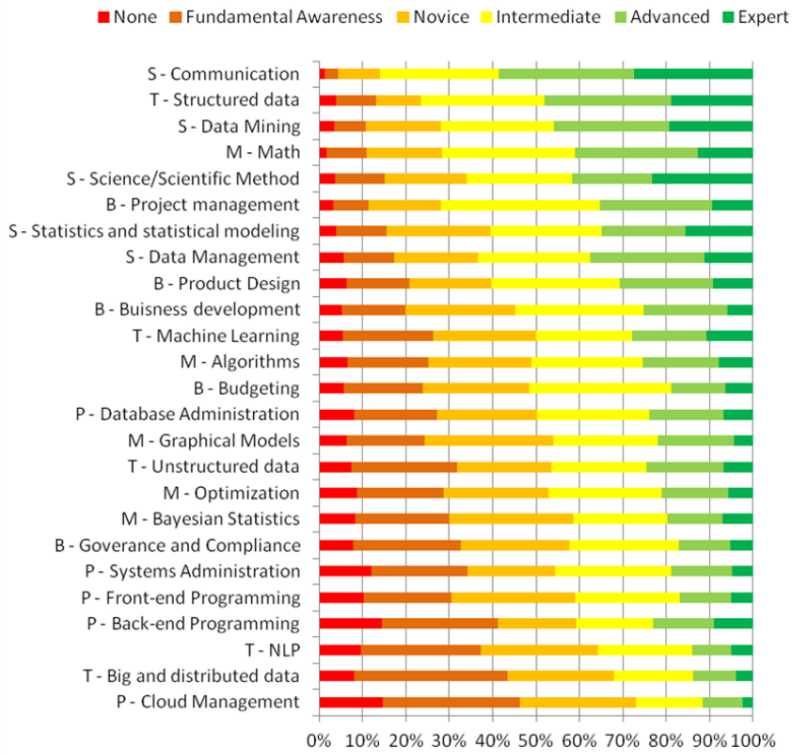
Pour aller encore plus loin, un sondage réalisé auprès de 410 professionnels révèle le type de compétences le plus répandues et dans lesquelles ces personnes ont le plus de maîtrise. On retrouve dans le top 10 :

- 60 % de compétences propres à l'analyse de données ou « Data Analysis Functions » au sens de Dana Parks
- 30 % d'« Interdisciplinary Knowledge »
- 10 % de « Critical Abilities » avec la Communication qui se place en première position des compétences à posséder.

Le Machine-learning et les Programming skills n'apparaissent que plus tard dans le classement. Ce sondage nous révèle donc que les compétences business transversales, mais également certains soft skills ne sont pas à sous-estimé. Il ne suffit donc pas d'être un programmeur expérimenté ou un doctorant en mathématiques pour être considéré comme efficace dans un travail lié à la Data Science. Savoir communiquer ses idées, avoir une pensée critique et la curiosité sont des « softs skills » nécessaires dans ce milieu. De même le Project Management et la capacité à comprendre les besoins des opérations et du business sans difficulté grâce à des compétences interdisciplinaires permettent à un Data Scientist d'être une redoutable arme face à de nouvelles problématiques.

(Hayes, 2015)

Figure 11: Proficiency Levels of 25 Data skills



La Data Science n'a donc pas qu'une seule et même facette au même titre que le Big Data lui-même n'est pas une seule et même entité. La Data Science est une histoire d'équipe et de compétences complémentaires. Les rôles d'une telle équipe doivent

comprendre des experts dans tous ces domaines et par conséquent le terme général « Data Scientist » n'est plus suffisant pour décrire la multitude de rôles nécessaire au bon fonctionnement d'une équipe d'analyste.

2.2.3 Les rôles :

L'analyse de différent workforce framework (EDISON, SAIC, Gartner, Springboard)) développé par différent organisme à propos de la Data Science met en lumière différents rôles bien distincts.

Tableau 4: Summary of roles used across multiple case studies

	EDISON	SAIC	Gartner	Sprinboard
Data Science researcher	✓			
Data Scientist	✓	✓	✓	✓
Data Architect	✓			✓
Data Analyst	✓			✓
Data Science programmer	✓			
Data Engineer		✓	✓	✓

(Saltz, Grady, 2017)

Data Scientist et Data Engineer sont donc au centre de la majeure partie de ces frameworks suivis des Data Architects et Data Analyst. En interrogeant le moteur de recherche de Linkedin avec ces intitulés de postes dans l'UE, le nombre de postes ouverts pour chacun des intitulés parle de lui-même.

Tableau 5 : Résultats de recherche de poste dans la Data Science dans l'UE sur Linkedin (15.02.2019)

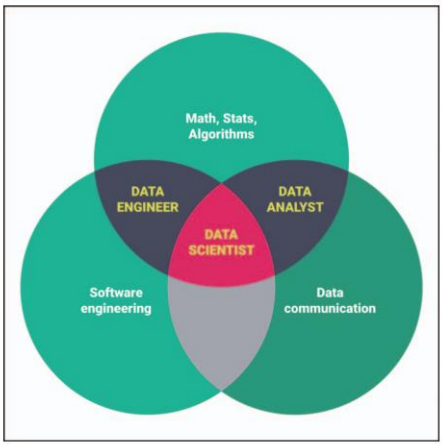
	Nombre de postes ouverts
Data Scientist	6 812
Data Analyst	6 101
Data Engineer	4 446
Data Architect	1 083
Data Science Researcher	2
Data Science Programmer	2

(Linkedin, 2019)

Les recherches ont été faites en mettant l'entièreté des termes entre crochets afin d'obtenir le nombre de postes contenant la combinaison des deux mots et non uniquement le mot data ou science. Le trio de tête est donc celui que Sprinboard décrit dans son guide de carrière.

(Saltz, Grady, 2017)

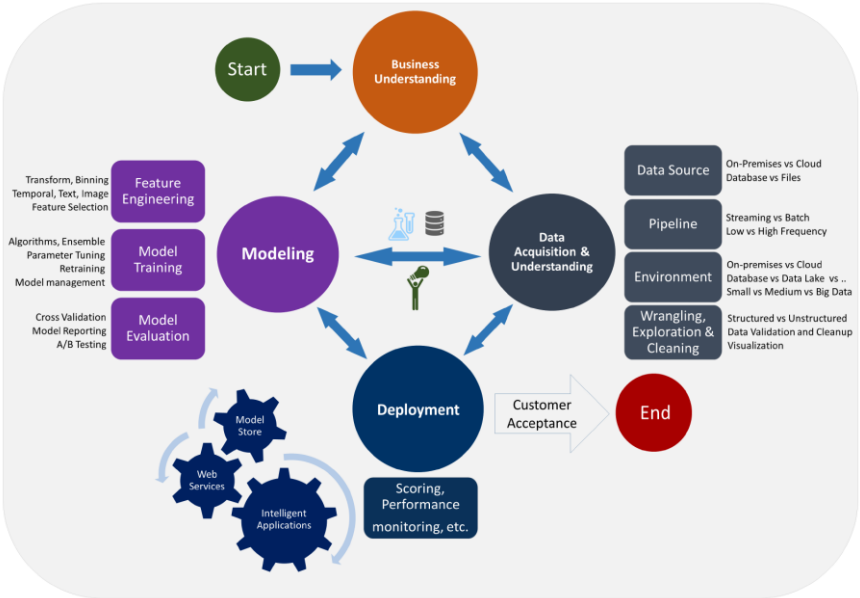
Figure 12: Springboard Careers Path



Le Data Architect pourrait venir combler le vide de ce diagramme et l'image serait complète.Afin de définir chacun des rôles avec précision, nous allons nous intéresser au processus complet de la Data Science puis incorporer ces rôles un à un afin d'identifier leur utilité.

(Kline, 2003)

Figure 13: Data Science as a process



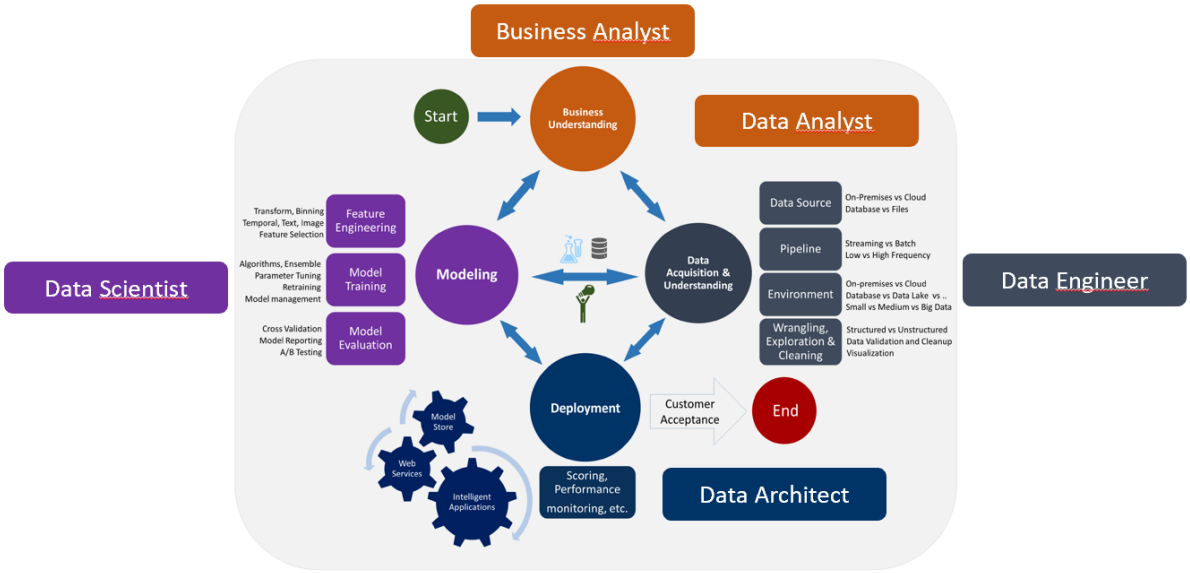
Un projet de Data Science commence le plus souvent par un « Business Requirements ». Dans le schéma ci-dessus ce serait l'étape du « Business Understanding. C'est à ce moment-là que les Business Analyst interviennent. Ayant des compétences métiers développées, c'est à eux que reviennent la lourde tâche de définir l'horizon d'un projet de Data Science, mais également choisir quel projet aura le plus d'impact stratégique pour l'entreprise. Les Data Analysts peuvent être considérés comme des Business Analyst avec des compétences supplémentaires en visualisation des données. Ils ont capables déjà d'offrir des informations exploitables avec les données existantes et à partir des modèles crées par leurs collègues en Data Science (Springboard, 2016). Selon le framework EDISON, les data analysts peuvent être décrit comme capable d': *« analyzes large variety of data to extract information about system, service, or organization performance and present them in usable/actionable form »* (Saltz, Grady, 2017).

Le processus de « Data Acquisition & Understanding » est le royaume du « Data Engineer ». Ce dernier est plus qualifié que son collègue Analyst en termes de programmation. Il prépare les données à leurs utilisations, retranscrit les requêtes en codes et implémente les modèles créés par les data scientists au cœur de l'architecture informatiques d'une entreprise. Selon Springboard ils: « *relies mostly on his or her software engineering experience to handle large amounts of data at scale. Typically focuses on coding, cleaning up data sets, and implementing requests that come from data scientists. […] When somebody takes the predictive model from the data scientist and implements it in code, they are typically playing the role of a data engineer »* (Springboard, 2016).

Le Data Architect lui serait à cheval entre « Data Acquisition & Understanding » et le « Deployment ». En effet aussi qualifié qu'un « Data Engineer » en termes de compétences informatiques il est en charge du design de toute l'infrastructure hardware/software nécessaire au stockage et à l'exploitation des données. Sachant le rôle d'un Data Engineer qui se doit de connaître l'architecture mise en place parfaitement afin d'en exploiter tout le potentiel, nous pouvons émettre l'hypothèse que ces deux rôles finissent par se confondre. Le Data Architect devient Engineer une fois le déploiement effectué. Les futurs changements d'infrastructure seront ainsi opérés par l'Engineer. Cela rejoindrait la définition donnée par le framework EDISON qui donne au data engineer la capacité de : *« Designs/develops/codes large data (science) analytics applications to support scientific or enterprise/business processes »* (Demchenko et al., 2017). Le mot « design » provoquerait la fusion des deux rôles en un.

Le data scientist lui existe bel et bien et serait en charge de la partie « modelling ». Entre les matières quantitatives et l'informatique, son rôle est de construire les modèles prédictifs basés sur l'exploitation des données afin de produire des « data products ». « *A Data Scientist is a practitioner who has sufficient knowledge in the overlapping regimes of expertise in business needs, domain knowledge, analytical skills, and programming and systems engineering expertise to manage the end-to-end scientific method process through each stage in the Big Data lifecycle, till the delivery of an expected scientific and business value to science or industry. »(Demchenko et al., 2017).*

Figure 14: Data Roles Through the Data Science Process

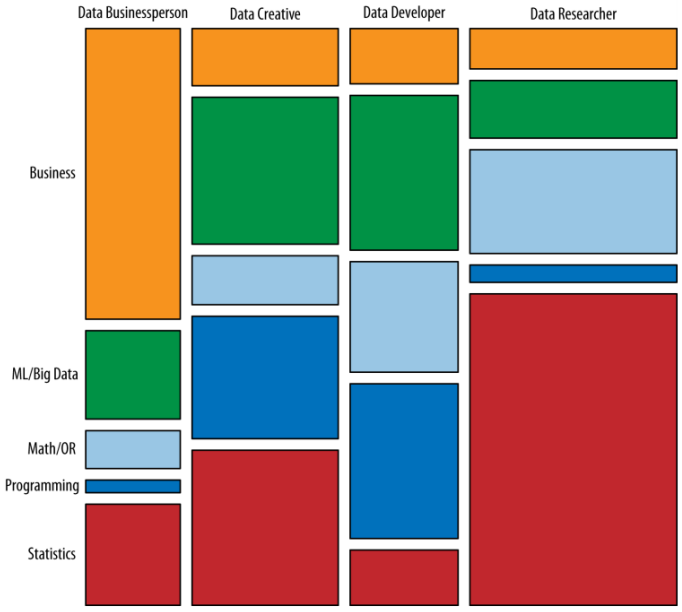


En termes de compétence clé, il est facilement repérable où chacun trouve ces points forts. Si nous comparons ces différents rôles aux groupes définis dans le sondage mené par O'Reily :

- Le Data/Business Analyst serait le Data Businessperson
- Le Data Engineer et l'Architect seraient certainement le Data Developper
- Le Data Scientist seraient un mélange entre le Data Creative et le Data Reseacher en fonction de son utilisation de l'intelligence artificielle

(Harris et al., 2013)

Figure 15: Clustering of Data Scientist



En résumé, la Data Science est un mélange de compétences complémentaires. Ce mélange permet aux équipes dans la Data Science d'analyser les nouvelles problématiques business de plusieurs points de vue différents pour trouver les solutions les plus adaptées. Ces professionnels de la donnée sont des ingénieurs informatiques, des mathématiciens/mathématiciens et des experts business travaillant ensemble pour offrir non seulement aux managements de nouvelles informations exploitables pour diriger la stratégie de leurs entreprises, mais également de nouveaux outils et expériences adaptés à leurs clients. Cette nouvelle science et ces nouveaux rôles ont été accompagnés par des changements technologiques afin de permettre la mise en place des solutions évolutives innovantes. Quelles sont-elles ? Quelles sont leurs utilités ? L'objectif de notre 3e chapitre est de s'attarder sur ces questions.

3. Les technologies Big Data

3.1 Les grandes tendances

Les technologies informatiques font indéniablement partie des compétences des personnes employées dans la Data Science. Avec plus ou moins d'expertise, chacun à son niveau dans une équipe analytique doit connaître certains outils afin de tirer parti des données à sa disposition. Les données dont nous parlons sont générées et stockées par des processus informatiques, il est donc intuitif que leurs utilisations se fassent à travers le même biais.

Le tableau ci-dessous divise les compétences technologiques en 3 grandes divisions.

Tableau 6: Classifications of technology skills in Data Science

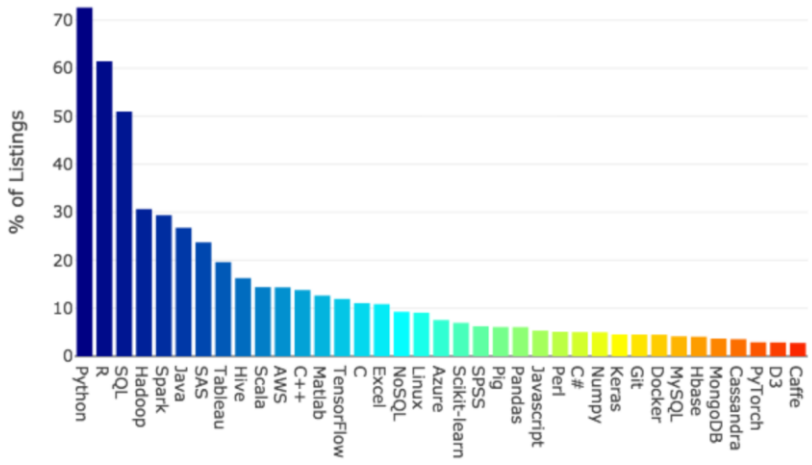
Classifications	**Exemples**
Les applications	Tableau, PowerBI, Excel, SAS…
Les langages de programmation	R, Python, SQL, Java, NoSQL..
Les infrastructures	Hadoop, Cassandra, Pig, Hive…

(Radovilsky et al., 2018) :

Cette classification est étroitement liée avec les différents rôles décrits précédemment dans le milieu de l'analyse de données. Les applications sont les principaux outils des Data/Business Analyst alors les langages de programmation sont ceux des data scientists. Les différentes infrastructures sont par contre le domaine d'expertise des Data Engineers et Architects. Bien évidemment ces classes ne sont pas hermétiques pour chacun des rôles. Chacun peut choisir ces armes parmi ces différents domaines pour amener son équipe à trouver les meilleures solutions. Voici un classement des compétences les plus demandées sur le marché de l'emploi actuellement.

(Hale, 2018)

Figure 16: Technology Skills in Data Science Job Listing



La première place revient à Python. 70 % des descriptions de postes citent ce langage comme un « must-have ». De plus beaucoup des autres compétences de ce graphique sont des librairies codées pour ce langage telles que Scikit-Learn, Pandas, Numpy et PyTorch. Grâce entre autres à cela, 52 % de ces compétences sont des langages de programmation, 1 tiers concernent les infrastructures et seulement 14 % des applications.

Tableau 7: Classification of the most in demand skills

Classes	Compétences	Part
Langages de programmation	Python, R, SQL, Java, Scala, C++, Tensorflow, C, NoSQL, Scikit Learn, Pandas, Javascript, Perl, C#, Numpy, Keras, MySQL, PyTorch, D3, Caffe	52.78%
Infrastructure	Hadoop, Spark, Hive, AWS, Linux, Azure, Pig, Git, Docker, Hbase, MongoDB, Cassandra	33.33%
Applications	SAS, Tableau, Excel, SPSS, Matlab	13,89 %

Nous n'allons pas nous attarder sur les applications qui sont des outils la plupart « All-in One ». Le but de ce travail est de mettre en place un environnement de développement Big Data et développer un PoC (« Proof of Concept ») appliqué à la finance. Ainsi nous allons nous concentrer sur les langages de programmation les plus couramment demandées et utilisées, mais également sur les architectures Hadoop et Spark, incontournables dans ce milieu. Ces quatre éléments nous permettront de nous plonger au cœur des concepts de Data Science appliquée à la finance. Ainsi dans ce chapitre, nous ferons un rapide aperçu du langage Python qui sera le langage principal de notre développement. Ensuite nous décrirons l'architecture d'Apache Hadoop et le framework Spark qui seront le cœur de la structure de notre projet et ce sont ces deux éléments qui qualifieront notre projet de projet Big Data.

3.2 Le langage Python

Le 20 février 1991, Guido van Rossum publie la première version du langage Python (Rossum, 2009). Les principales caractéristiques de ce langage sont :

- Sa philosophie open source : cela lui a permis d'obtenir l'extensibilité qui le caractérise grâce aux nombreuses librairies annexes et cela en restant gratuit.
- Son caractère portable : il est utilisable facilement à travers l'ensemble des OS/plateformes.
- Son process d'interprétation/compilation : Permets le test rapide de n'importe quel code
- Sa facilité d'écriture : la déclaration dynamique des variables et l'indentation du code en simple tabulation sont quelques-unes de ces possibilités qui rendent python simple d'écriture et lisible.
- Orienté objet : Ce paradigme de programmation permet la construction de code simple et de manière efficiente

(Swinnen, 2008)

Figure 17 : Python Compilation/Interpretation

Python permet ainsi d'écrire rapidement, facilement, et de tester des algorithmes « on-the-spot » indépendamment de leur complexité. Ceci en fait un outil parfait pour l'exploration et l'analyse de données. En outre, depuis l'apparition de notebook tels que Jupyter, le partage de script entre les équipes et la visualisation de données ont été grandement facilités et de fait la collaboration également. Désormais Python peut être déployé à moindre coût pour effectuer des calculs scientifiques, du machine-learning, de la manipulation de données ou même de la programmation web. Cet aspect très versatile la rendu populaire à travers des petites, mais également de grandes entreprises comme Google, YouTube ou encore Facebook (Python, 2018). Le Tiobe Index le place désormais à la 3e place des langages les plus couramment utilisés et le « Developper Survey » de StackOverflow place Python en première position des langages les plus souhaités avec 25 % des voix (StackOverflow, 2018).

(Tiobe, 2019)

Figure 18 : Tiobe Index

Mar 2019	Mar 2018	Change	Programming Language	Ratings	Change
1	1		Java	14.880%	-0.06%
2	2		C	13.305%	+0.55%
3	4	^	Python	8.262%	+2.39%
4	3	v	C++	8.126%	+1.67%
5	6	^	Visual Basic .NET	6.429%	+2.34%
6	5	v	C#	3.267%	-1.80%
7	8	^	JavaScript	2.426%	-1.49%
8	7	v	PHP	2.420%	-1.59%
9	10	^	SQL	1.926%	-0.76%

Le site StackShares recense quant à lui les avis des développeurs et voici les avantages les plus cités :

- Great Libraries
- Readable Code
- Beautiful Code
- Rapid Developpment
- Large Community
- Open-Source
- Elegant
- Great Community
- Object Oriented
- Dynamics typing

Parmi les librairies de python cité précédemment dans l'introduction de ce chapitre, Scikit-learn, TensorFlow ou Pandas sont des librairies spécialement adaptées à la Data Science. Que ce soit pour de la data manipulation, visualisation ou l'écriture d'algorithmes de Machine Learning, le développeur python possède des librairies très puissantes et souvent mises à jour par la communauté.

Le meilleur des langages sans données n'est pas très utile en Data Science et des données sans architecture de stockage ne sont pas exploitables. C'est pourquoi nous allons nous attarder sur les technologies développées sous les licences Apache qui sont désormais les plus populaires dans le monde de la Data Science.

3.3 Apache Hadoop

Qu'est-ce que Hadoop ? Si vous travaillez de loin ou de près avec des équipes dans l'analyse de données, ce terme est inévitable. Afin de définir cette nouvelle philosophie d'infrastructure de données, nous sommes obligés d'en explorer l'historique.

3.3.1 La genèse d'Hadoop

Avec la multiplication des sources et l'élargissement des volumes des données, les entreprises comme Google et Facebook ne pouvaient plus satisfaire leurs besoins avec les infrastructures classiques de type RDBMS (Relationnal Database Management System). Avant Hadoop, augmenter sa capacité de stockage et sa puissance de calcul signifiait payer plus cher les fournisseurs d'espaces de stockage tels que Microsoft, IBM, Oracle afin d'utiliser du hardware plus puissant pour constituer les serveurs où étaient conservées les bases de données (Intricity101, 2012 a). À l'échelle de Google ou Facebook, cette philosophie résultait à créer des immenses bases de données reliées

aux systèmes de l'entreprise par des goulets d'étranglements par lequel passaient les requêtes de la compagnie. Plus les volumes grandissaient, plus les canaux de communications devenaient petits et inefficients.

3.3.1.1 Google File System

En 2003, Sanjay Ghemawat, Howard Gobioff, et Shun-Tak Leung, ingénieurs chez Google, publient le « Google File System » whitepaper. Le but de cette architecture éponyme était d'adresser les problèmes de « scaling-up » (achat de hardware plus puissant), mais également de performance de calcul que leur entreprise rencontrait déjà en 2003. *"We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients"* (Ghemawat et al., 2003). Les grandes lignes de ce rapport présentent les principaux avantages de cette architecture :

- **Distributed File System :** Au lieu de stocker l'ensemble des données sur un serveur sous forme de tables en lignes/colonnes, les données sont réparties sous forme de fichier compressé à travers un cluster de serveur qui peut contenir des milliers de machines appelées « nodes ».
- **Scalable :** Le cluster est extensible à volonté, la capacité de stockage s'en retrouve multipliée, mais également la puissance de calcul.
- **Running on inexpensive commodity hardware :** Le hardware nécessaire pour constituer le cluster n'est pas nécessairement du high-end. Ceci permet un meilleur contrôle des coûts et transforme le « scaling-up » par le « scaling-out ». Cette transformation se résume à la multiplication des machines plutôt que l'amélioration de ces dernières.
- **Data Replication :** les données sont répliquées à travers différents nœuds du cluster. Le facteur de réplication est de 3 par défaut.
- **Fault Tolerance :** Les architectures RDMS demandent une intégrité des données parfaites avant le retour d'un résultat aux clients. C'est le A d'Atomicité et le C de Consistence du concept ACID (Serra, PassTv, 2016). En opposition le GFS permet l'« Eventual Consistency ». Ainsi si l'un des nœuds du cluster génère une quelconque erreur ou tombe en panne, la requête du client continuera à s'exécuter à travers le cluster. L'intégrité des données viendra donc avec le temps et n'est pas immédiate. De plus grâce à la duplication des données, les données restent accessibles à travers d'autres nodes.

3.3.1.2 MapReduce

À la suite de ce whitepaper, d'autres employés de Google publient « Map Reduce : Simplified Data Processing on Large Clusters » (Dean, Ghemawat, 2008). Ce rapport pose les bases du modèle algorithmique MapReduce. Ce modèle de programmation a été développé pour répondre à la nouvelle structure de stockage en clusters de Google. « *Programs written in this functional style are automatically parallelized and executed on*

a large cluster of commodity machines" (Dean, Ghemawat, 2008). Ainsi ce modèle permet de :

- **Map :** séparer les tâches d'un algorithme à travers l'entièreté du cluster et ainsi permettre à chaque nœud de travailler sur ces propres données. En langage technique:

  "Map: takes as input an object with a key (k,v) and returns a bunch of key-value pairs: (k1, v1), (k2, v2), …, (kn, vn). The framework collects all the pairs with the same key k and associates with k all the values for k: (k, [v1,.. ,vn]) » (Serra, PassTv, 2016)

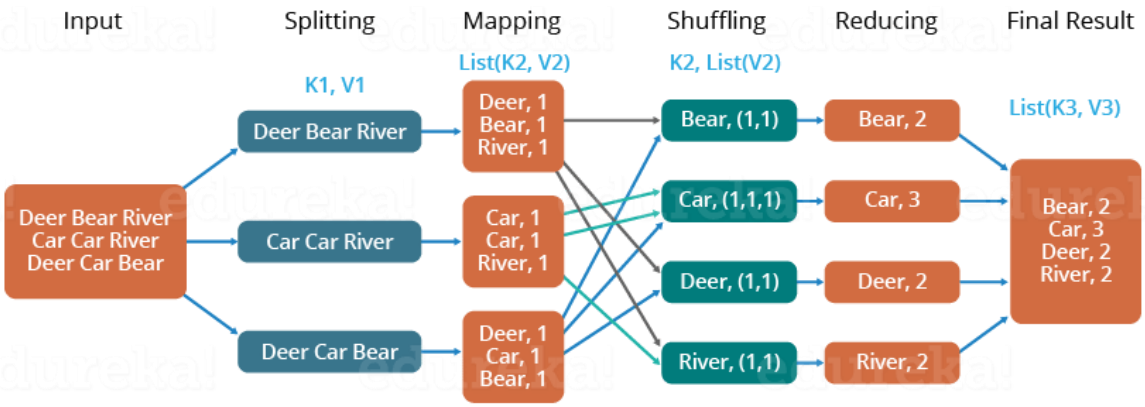
- **Reduce :** chaque nœud renvoie son résultat au master node. Ce dernier se chargera d'agréger l'ensemble des résultats pour offrir aux clients un seul et même résultat.

  "Reduce: takes as input a key and a list of values (k, [v1,.. , vn]) and combine them somehow." (Serra, PassTv, 2016)

Un exemple courant pour expliquer le concept de MapReduce est celui du Wordcount Algorithm. Le but ici est de définir dans un texte combien de fois chaque mot apparaît à travers l'ouvrage.

(Bakshi, 2016)

Figure 19: WordCount Process with MapReduce



Ce modèle permet de décomposer des tâches complexes en plusieurs petites tâches puis de les distribuer à travers tout un cluster afin que chacun des nodes travaille sur les données qu'il contient. C'est l'avènement du « parallelized computing ».

Avec la combinaison du GFS et de MapReduce, Google était désormais capable d'interroger ces immenses bases de données avec rapidité et à moindre coût. D'autres entreprises telles que Facebook ont suivi ces concepts et développé leurs solutions. Une architecture a cependant su s'imposer dans le monde professionnel par sa philosophie

open source et sa reprise des concepts de base adressant les problématiques majeures du « Big Data Computing ».

3.3.2 Hadoop en détails

En 2005, Doug Cutting et Mike Caferalla travaillant à l'époque chez Yahoo! s'inspirent des concepts de Google pour donner naissance à une nouvelle architecture de données. Ainsi naquis Hadoop projet devenu par la suite open source au sein l'Apache Foundation en 2006. Désormais c'est l'infrastructure Big Data la plus répandue dans le monde et sa maîtrise de ses concepts est très souvent demandés dans la Data Science (cf. Figure 15). Alors que GFS & MapReduce sont des modèles propriétaires de Google écrits en C++, Hadoop était à la base une simple réplication de ces concepts dans un environnement Open-Source écrit en Java (Torlone, 2016). Ainsi Hadoop est décomposé en deux éléments majeurs le HDFS ou Hadoop Distributed File System et MapReduce, le modèle de Google. L'Hadoop Distributed File System modifie la philosophie de stockage des données pour se calquer sur la philosophie du GFS. Afin de détailler un peu plus son fonctionnement, nous allons l'opposer à un RDBMS classique.

Dans un RDBMS il existe un schéma classique de table en lignes et colonne dont les croisements ou cellules sont les données. Dans ces systèmes, les relations entre les données sont prédéfinies logiquement par un jeu de clé primaire/étrangère et autres mécanismes assurant l'intégrité de la base de données. Ainsi, si des données doivent être transférées d'une base de données à l'autre, les types de données et les relations entre les tables se doivent d'être identiques à la base de données d'origine. En jargon, cela est le « Schema On Write », la logique doit être implémentée avant l'utilisation. En outre, les données ne sont pas répliquées de manière systématique et la consistance des résultats fournis par les requêtes se doit d'être absolument parfaite. En cas de panne de l'un ou l'autre des serveurs, le résultat final est compromis. Le processus assurant une complète intégrité des données se nomme le « Two Phase commit » (Intricity101, 2012 b).

(EDUCBA, 2018)

Figure 20 : RDBMS vs Hadoop

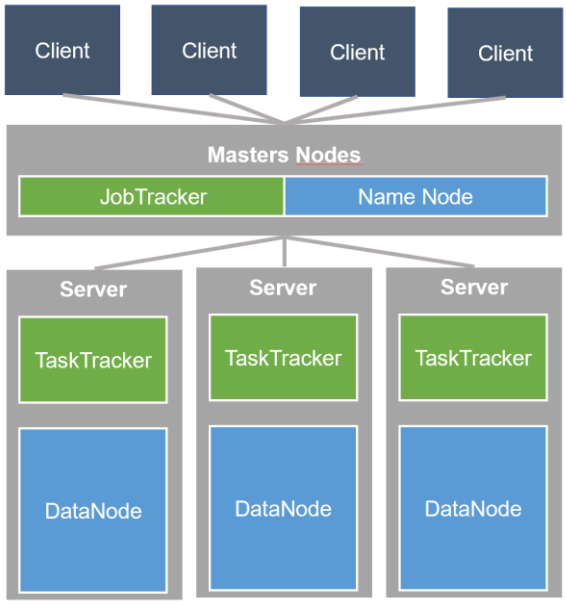
Features	RDBMS	Hadoop
Data Variety	Mainly for Structured data	Used for Structured, Semi-Structured and Unstructured data
Data Storage	Average size data (GBS)	Use for large data set (TBS and PBS)
Querying	SQL Language	HQL (Hive Query Language)
Schema	Required on write (static schema)	Required on read (dynamic schema)
Speed	Reads are fast	Both Reads/Writes are fast
Cost	Mostly Licensed	Free
Use Case	OLTP (Online Transaction Processing)	Analytics (Audio, Video, Logs etc.). Data discovery
Data Objects	Works on Relational Tables	Works on Key/Value Pair
Throughput	Low	High
Scalability	Vertical (Scaling-up)	Horizontal (Scaling-out)
Hardware	High-End Servers	Commodity/Utility Hardware
Integrity	High (ACID)	Low
Consistency	Two Phases Commit	Eventual Consistency

En revanche, Hadoop stocke les données sous forme de fichier compressé, répliqué à travers un ensemble de serveur esclave, les DataNodes. Ainsi la logique des relations entre les données et les types de données sont définis uniquement dans le code du client lorsqu'il s'adresse aux serveurs et non au sein même du cluster. C'est la philosophie « Schema on Read », la logique est implémentée dès que nous souhaitons accéder aux données pas à l'écriture des données. Cela augmente grandement la flexibilité de l'architecture ce qui lui confère la possibilité de travailler avec des données non structurées telles que des vidéos, images, web logs, etc. Lorsque nous souhaitons tirer des informations stockées dans Hadoop, notre requête est adressée à deux masters nodes : le Name Node et le JobTracker. Le Name Node possède la localisation de toutes les données conservées dans les DataNodes du cluster. Alors que le JobTracker lui répartit la charge de travail à travers tous les nœuds du réseau via les TaskTrackers. Ainsi, chaque nœud esclave du réseau effectue les opérations concernant ses propres données pour ensuite restituer son résultat qui sera consolidé avec celui des autres pour fournir un résultat global à l'utilisateur. Ce processus n'est qu'une simple application du concept de MapReduce au sein du GFS de Google. Par ailleurs les deux master nodes sont aussi essentiels dans la gestion de la « Fault Tolerance ». En effet, chacun contacte

ces « slaves » à intervalle régulier afin qu'ils leurs retournent leurs statuts. Si un slave ne répond plus, le master se charge de redistribuer ses tâches ailleurs dans le cluster et de les réinitialiser afin qu'ils soient disponibles pour de futures tâches.

(Hortonworks, 2013)

Figure 21 : Hadoop Architecture



Les professionnels détenaient enfin une solution libre de droits et performante pour les accompagner dans la transformation digitale de leurs entreprises. Hadoop n'a pas remplacé les RDBMS qui continuent à être utilisés massivement dans le business, 89 % de part de marché selon Gartner (Serra, PassTv, 2016). Cependant l'explosion de l'intérêt à son égard en fait le symbole des technologies Big Data et cela justifie qu'Hadoop fasse partie intégrante de l'environnement développé dans ce travail. En complément d'Hadoop, d'autres technologies seront utilisées, c'est pourquoi il est nécessaire de prendre un peu de recul sur l'écosystème dans son ensemble.

3.3.3 L'écosystème Hadoop de nos jours

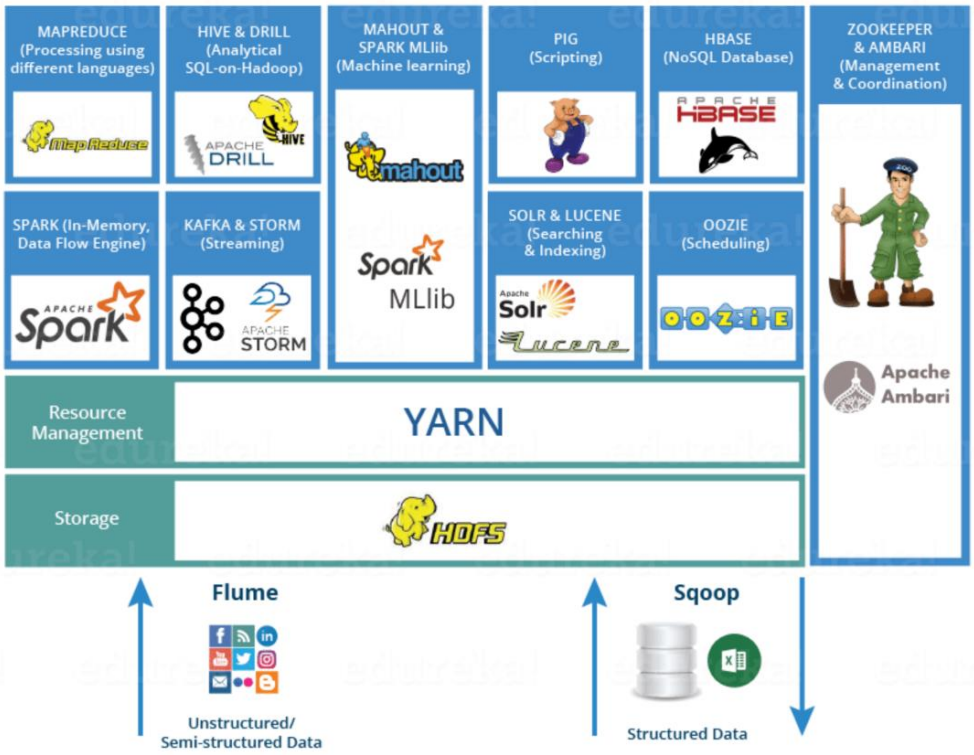
De nos jours l'écosystème s'est étoffé afin d'apporter un ensemble de solutions cohérentes aux professionnels. Si HDFS et MapReduce restent la base de l'architecture Hadoop, il existe désormais de nouveaux outils spécifiques qui améliorent et étendent les capacités en termes de gestion de données d'Hadoop. Sans rentrer dans les détails, voici les principaux (Cloudera, Inc., 2015) :

- **YARN :** Yet Another Ressource Negotiator introduit lors de la seconde release d'Hadoop c'est une couche supplémentaire qui sépare le Data Processing via MapReduce et le ressource management soit la gestion

faite par le JobTracker et le NameNode. Plusieurs tâches peuvent être gérées plus efficacement à travers le cluster.

- **SQOOP :** Combinaison entre SQL et Hadoop, SQOOP est un connecteur qui permet d'intégrer des données provenant de RDBMS classique à HDFS
- **Flume :** Alors que SQOOP permet l'intégration des données structurées de manière classique au cœur d'HDFS, Flume permet de connecter des données live telles celles de capteurs ou des web logs à Hadoop. Ce type de données sont appelées Streaming Event Data et elles font partie intégrante de notre vie digitale.
- **Hive :** développé par Facebook cet outil permet à tous les développeurs de s'adresser à Hadoop de la même manière qu'à un RDBMS classique c'est-à-dire à travers le langage SQL. HiveQL ressemble sensiblement à SQL, mais ce n'est pas une copie. Avant cela, des lignes de codes en Java étaient nécessaires et la courbe d'apprentissage de Java est bien plus longue que celle de SQL : cela a contribué à la démocratisation d'Hadoop.
- **Spark :** Le nouveau « data processing engine » de l'écosystème qui commence petit à petit à faire sa place grâce à ces performances plus efficaces que la combinaison HDFS & MapReduce. De plus Spark permet l'utilisation de plusieurs langages largement répandue. L'intérêt pour Apache Spark grandit d'année en année et il fait partie du top 5 des compétences les plus demandées (cf. Figure 15) dans la Data Science. Pourquoi un tel engouement ? Qu'ajoute Spark à la révolution Hadoop ? C'est le sujet de notre prochaine partie, car nous allons également l'intégrer dans notre environnement de travail.

Figure 22 : Hadoop Ecosystem



(Cloudera, Inc., 2015)

3.4 Le framework Apache Spark

Dans la partie précédente, nous avons mis en exergue le fait qu'Apache Hadoop n'était pas un unique projet, mais un ensemble de solutions en constante évolution. Ainsi l'architecture est toujours à même de répondre aux besoins des développeurs et à faire face à la mutation de l'environnement Data de notre ère. L'écosystème Apache Hadoop s'étoffe année après année et ses problèmes trouvent souvent leurs solutions dans de nouveaux développements. À l'instar d'Apache Spark développé dans le but d'améliorer la rapidité d'exécution et le temps de réponse des outils de base que sont HDFS et MapReduce. En 2010, des doctorants de l'université de Californie, Berkeley publie le : « Spark: Cluster Computing with Working Sets ». Dans ce rapport, les étudiants exposent les qualités de leur nouveau framework : *« Spark can outperform Hadoop by 10x in iterative machine learning jobs, and can be used to interactively query a 39 GB dataset with sub-second response time »* (Zaharia et al., 2010). Comment est-ce possible ? Quelles sont les méthodes utilisées ?

3.4.1 Les motivations du développement d'Apache Spark

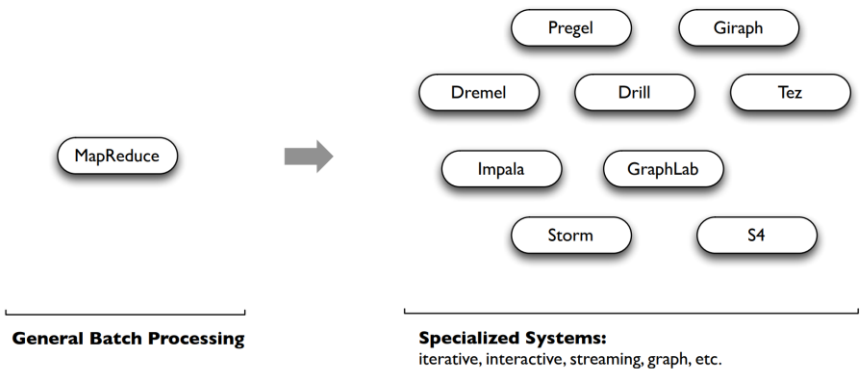
Avant de répondre à ces questions, nous devons comprendre pourquoi Spark a vu le jour. Après sa démocratisation, MapReduce a montré deux limitations majeures :

- La difficulté de programmer directement les workflows
- Les goulets d'étranglement de communication due aux nombreuses étapes read/write nécessaires pour accomplir les tâches itératives

Par conséquent, la communauté de développeurs s'est affairée à résoudre ces problèmes et une quantité de projets ont vu le jour chacun avec son but précis.

(Nathan, 2017)

Figure 23: Batch Processing VS Specialized Systems



Le problème de tous ces systèmes réside dans le fait que pour chaque nouvelle tâche ou application un nouveau système est nécessaire. Spark a tenté d'intégrer l'ensemble des fonctionnalités dans un système facile d'accès pour les développeurs tout en gardant les avantages tels que la « fault tolerance » du MapReduce. Pour cela les étudiants ont dû mettre en place deux nouveaux concepts :

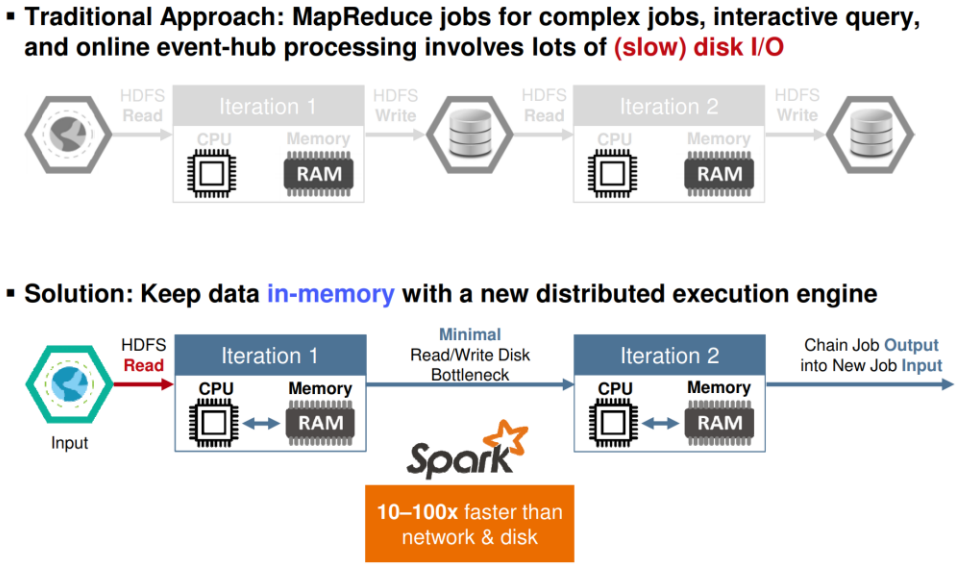
- Les RDD : Resilient Distributed Datasets
- Les Shared Variables: Broadcast variables & Accumulators

3.4.2 Les développements clés d'Apache Spark

Les RDD sont la base du framework Spark: *"A resilient distributed dataset (RDD) is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. The elements of an RDD need not exist in physical storage; instead, a handle to an RDD contains enough information to compute the RDD starting from data in reliable storage. This means that RDDs can always be reconstructed if nodes fail"* (Zaharia et al., 2010). Grâce à ce nouveau concept les développeurs de Spark ont limité les opérations de Read/Write sur les disques en permettant au cluster de travailler sur des données dites « in-memory » soit en cache. L'accès aux données est ainsi accéléré. De plus, grâce à la réplication des données à travers le cluster, le risque de perdre des informations est grandement diminué.

(Beekman, 2017)

Figure 24: Spark VS Hadoop Workflows



Les Shared Variables sont également nouvelles et permettent de répondre à certains des uses cases dans la computation parallèle. Les variables dites broadcasted permettent à l'utilisateur de partager des données dites de dimension ou lookup table à travers

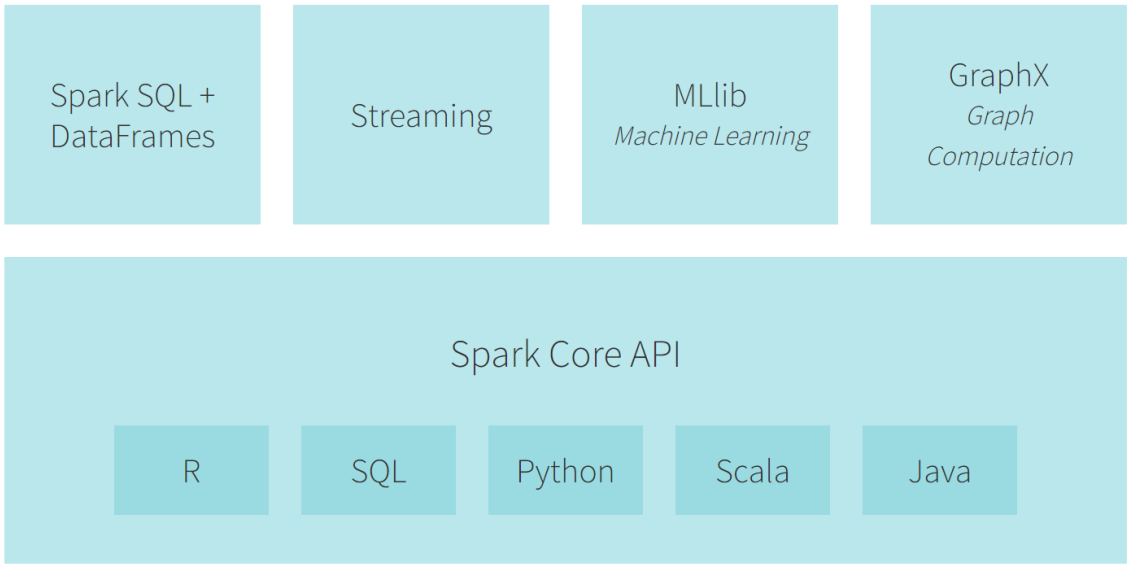
l'ensemble du cluster afin que chaque node puissent accéder à ses données sans créer de goulets. Les Accumulators sont des variables compteurs partagées et synchronisées à travers tout le cluster. Ce type de variable est très utile dans le cadre de processus itératifs afin de savoir où en sont les tâches de chacun des nodes et ainsi coordonner l'ensemble du processus (Kane, 2017).

Ces développements qui sont la base d'Apache Spark ne sont que le début de la vision de ses auteurs. Comme nous l'avons mentionné, l'idée derrière Spark était également d'offrir la possibilité de rassembler un ensemble d'outils afin de performer des autant du batch processing que du streaming au sein d'un seul et même environnement. Spark permet cela grâce à son environnement flexible.

3.4.3 L'architecture Spark

(Databricks, 2019)

Figure 25 : Spark Ecosystem



Comme vous pouvez le constater sur ce schéma, Spark, malgré le fait qu'il ait été développé en Scala, permet l'utilisation de nombreux autres langages de programmation via différentes API comme R, Python, Scala et Java. Les principes de RDD et de Shared Variables sont bien évidemment accessibles via ces langages. Puis en un coup d'œil, nous pouvons également observer la complétude du framework avec :

- **Spark SQL :** fourni une abstraction de programmation supplémentaire à Spark : Les DataFrames. Ces dataframes rendent Spark complètement compatible avec le langage SQL classique tout en augmentant 100x la vitesse d'exécution des Hive Query sur HDFS.

- **Spark Streaming :** Comme son nom l'indique ce composant est dédié à l'event base data et est déjà compatible avec les connecteurs d'Hadoop tels que Flume, Kafka, etc.
- **MLlib :** C'est la librairie de Spark dédié aux machine learning dans un environnement de computation parallèle.
- **GraphX :** Le moteur de computation graphique de Spark qui peut aisément remplacer les outils existants de Hadoop comme Giraph.

Les développeurs de Spark ont ainsi couvert l'ensemble des besoins majeurs tout en offrant la possibilité de programmer dans plusieurs langages différents en conservant des performances nettement supérieures. Quelles sont les performances des deux écosystèmes ?

3.4.4 Hadoop VS Spark :

Les chiffres parlent d'eux même. Dans un test qui se résumait à trier 1 TB de donnée, Spark effectue l'opération en 4x moins de temps avec 10x moins de machines.

Figure 26: Sorting operation—Comparison of performance

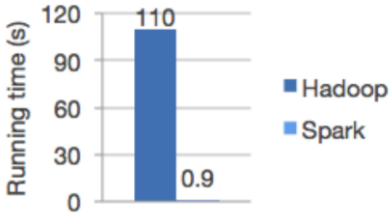
	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

(Nan, Su, 2017)

Même constat lors d'opération statistique telle que la régression linéaire. Spark met 100x moins de temps qu'Hadoop pour effectuer l'opération.

(Feng, 2019)

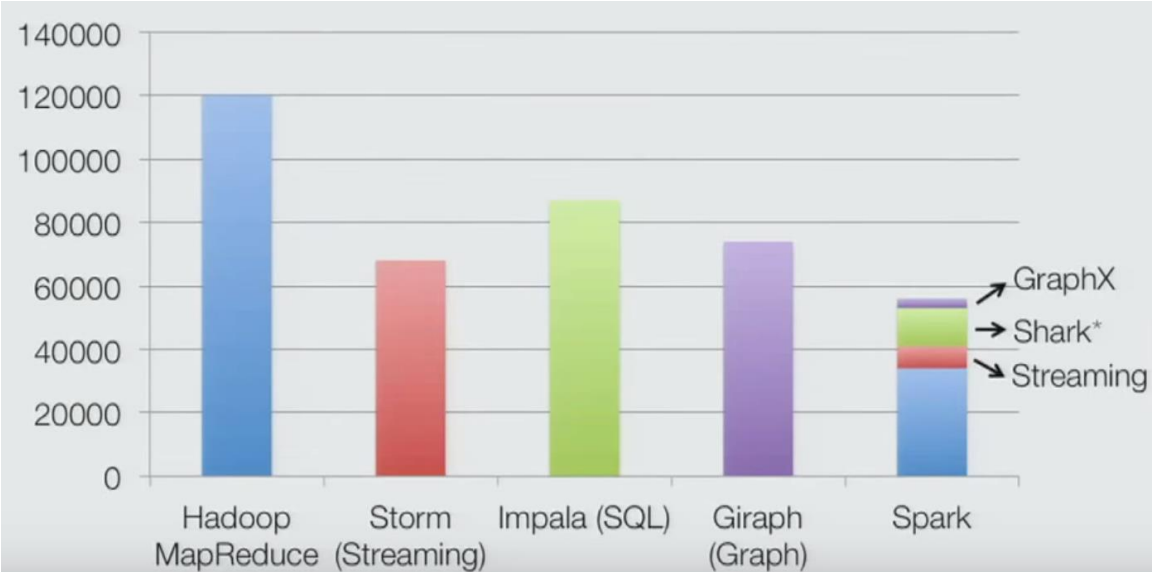
Figure 27: Logistic Regression—Comparison of performances



Spark a tellement été optimisé au niveau de ses performances et de son concept que même une comparaison entre la taille de son code et celui d'Hadoop et des composants nécessaires pour remplir la même fonctionnalité donne Spark gagnant.

(Spark Summit, 2013)

Figure 28: Code Size in lines



Ce qui reste impressionnant c'est que les pendants Streaming, Graph, SQL ajouté au core de Spark ne sont même pas plus lourds qu'un seul des composants d'Hadoop, Storm par exemple. Cela provient non seulement du fait que Spark est développé en Scala, plus compacte, mais également de la philosophie de l'in-memory processing.

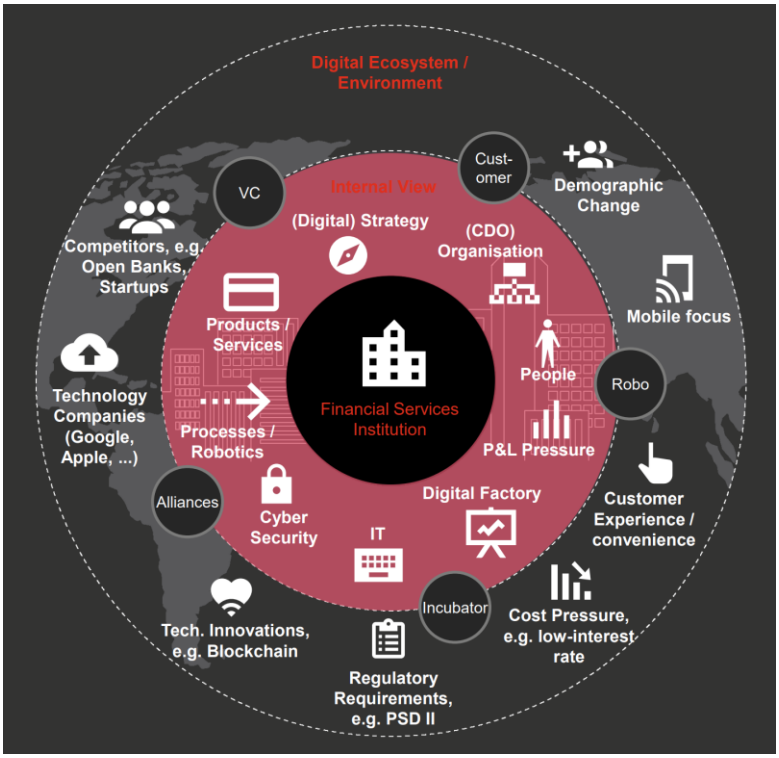
En conclusion de cette comparaison, il est clair que Spark superforme Hadoop et donc tout à fait normal que sa maîtrise deviennent un must-have en Data Science peu importe son domaine d'application. La recherche sur ces différentes technologies nous aura permis de définir le cadre de notre développement informatique. Il désormais temps de

nous attarder sur notre domaine d'expertise : la finance. Dans le prochain chapitre, nous allons nous plonger au cœur de la Data Science appliquée en finance afin de définir les principaux use cases et les pratiques les plus communes dans ce milieu.

4. La Data Science appliquée à la Finance

4.1 L'environnement technologique

Figure 29 : Financial Services Digital Environnement



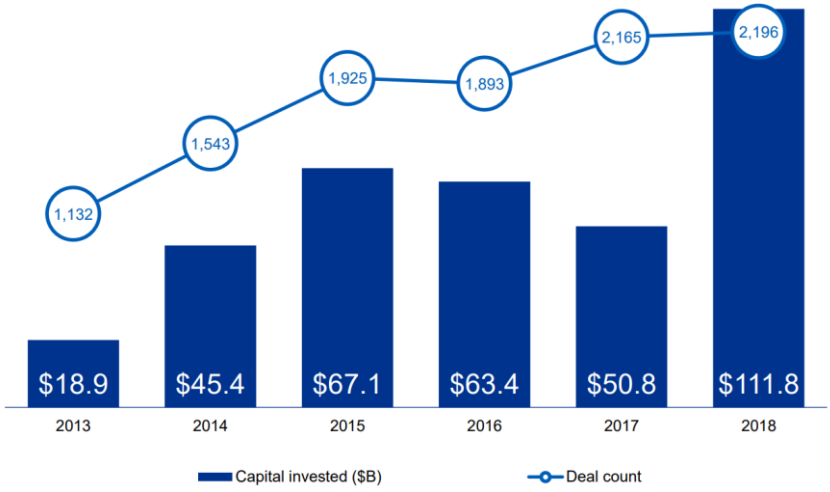
(Adewolu, 2018)

La figure ci-dessus résume parfaitement la situation actuelle du secteur financier et de son environnement technologique. Nous allons simplement souligner quelques aspects externes qui poussent les banques et assureurs à se restructurer technologiquement afin de rester compétitifs.

4.1.1 Les nouveaux entrants

De nos jours, l'industrie financière subit les répercussions de la 4e Révolution industrielle. De nouveaux concurrents plus agiles, plus flexibles apparaissent chaque jour avec de nouveaux produits et de nouveaux business models qui rivalisent d'efficacité avec les leaders du secteur. En 2018, le total des investissements en fintech se montait à 111,8 milliards de dollars plus du double des montants de l'année précédente (Pollari, Ruddenklau, 2019).

Figure 30: Total Investment Activities (VC, PE, M&A) in fintech



(Pollari, Ruddenklau, 2019)

En Suisse, Swisscom recense, en 2018, 357 start-up fintech encore actives et créées il y a moins de 10 ans dans le pays. Ainsi nous ne pouvons plus considérer ces nouveaux acteurs comme des signaux faibles dans l'industrie financière, mais plutôt comme de réelles menaces. Les domaines d'activités de ces start-ups sont divers et variés :

- Alternative form of financing (Crowdfunding) & virtual exchange
- Data-driven insights: Data Management, Data Analysis
- Cryptomonnaies,
- Investing & Asset Management: social trading, robots-advisory
- Market Info & Advisory portal : Portail de comparaison basé sur l'affiliation
- Payment: Mobile payment, Alternative transactions Rails
- Insurance

Un exemple américain, Wealthfront, crée en 2008 en réponse au manque de confiance provoquée par la crise financière, propose des services de conseil financier complètement dématérialisé au travers de chat-bot et de robot-advisors. En à peine 10 ans, cette start-up a réussi à rassembler sous sa gestion plus 12 milliards d'actifs. (Wealthfront, 2019) C'est une belle preuve de la croissance rapide de ces fintechs et des segments de marchés qu'elles peuvent attaquer.

En plus de ces nouvelles start-ups, les bigtechs telles que les GAFAM ou BATIX en Chine se lancent également dans les services financiers. Leurs offres très souvent concurrentes des banques et des assurances commencent à faire de l'ombre à certaines sociétés. Par exemple, Ant Financial la branche fintech d'Alibaba en Chine compte 451 millions

d'utilisateurs de ses services de paiements pour 153 millions de transactions par jours en 2016 (Alibaba, 2016). À titre de comparaison la même année, MasterCard possédait 1,4 milliard de comptes pour 180 millions de transactions par jours. La fréquence d'utilisation est significativement très proche, le terme de substitut peut aisément être utilisé. Ant Financial propose également des services de wealth management, d'assurances ou encore des crédits. L'ensemble des activités bancaires du back au front office sont représentés et peuvent aisément remplacés les services d'une banque traditionnelle. Les services de paiements, d'investissements, de financements et d'assurances vont considérablement être bouleversés par ce genre d'initiative démarré par les géants de la technologie. Samsung Pay, Apple Pay et récemment l'Apple Card, ou le Google Wallet sont autant de produits montrant l'appétit des GAFAM pour le secteur financier. L'appendice 1 vous présentera l'ensemble des initiatives des Big Tech en finance. Pas moins de 34 solutions y sont inscrites. En outre, 26 % des institutions financières ont déclaré en 2018 avoir déjà lancé des partenariats avec ces géants et 27 % le prévoyaient dans les 12 prochains mois (Ruddenklau, 2018). Les banques et assureurs sont bel et bien à la recherche des compétences techniques qui leur font défaut pour profiter des nombreuses opportunités qu'offrent les bouleversements technologiques de notre ère.

Les fintechs développent des technologies efficientes et possèdent le savoir-faire pour leur exploitation. Cependant, ces start-ups ont certains désavantages face aux institutions financières : aucune base de clients et donc d'information pour la prospection, un manque de réputation, et un capital limité (de la Mano, Padilla, 2018). C'est pourquoi ces entreprises sont souvent destinées à être rachetées plutôt qu'à devenir des géants de la finance. La véritable menace provient bel et bien des BigTechs. En effet, elles possèdent déjà une large clientèle à travers leurs plateformes en plus de l'ensemble de leurs données personnelles, une réputation déjà rayonnante, un accès aux marchés des capitaux et déjà des revenues très élevés (de la Mano, Padilla, 2018). En outre, ces grandes firmes sont les principaux acteurs du Big Data, de l'intelligence artificielle, du cloud computing, de la cryptographie, et génère la plus grande part du trafic sur internet.

La maîtrise de ces technologies est la clé de la finance de demain, car l'ensemble des innovations financières sont de près ou de loin liées à ces technologies. La figure 31 ci-dessous vous donnera un aperçu des révolutions en marche. Sans technologies et infrastructures, les services financiers sont bel et bien battus. L'arrivée de nouveaux entrants et l'augmentation de la pression concurrentielle ne sont pas les seules raisons qui justifient l'intérêt de la finance envers ces nouvelles technologies. Nous allons

maintenant brièvement présenter quelques-unes des autres motivations à l'origine de ces modifications de comportements.

Figure 31: Disruptive's innovations impacting financial services

(McWaters, 2015)

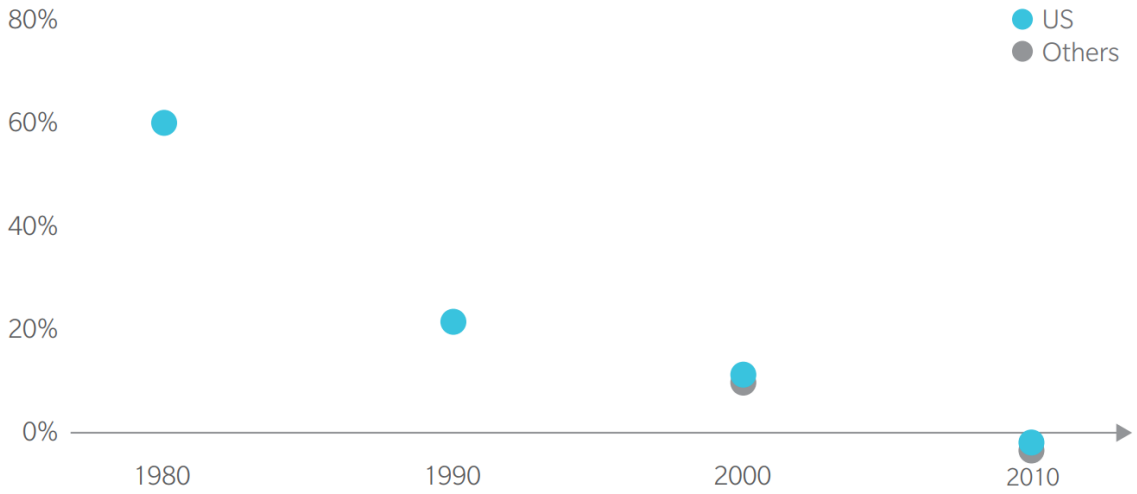
4.1.2 Les autres moteurs de cette révolution

Dans un premier temps, les services financiers ne sont pas épargnés par les changements de paradigme impliqués par la démocratisation des smartphones, de l'IoT et des réseaux sociaux. Trois vecteurs de données dont nous avons parlé dans notre premier chapitre. Les nouveaux et futurs clients des banques, assimilés aux « digital natives », poussent toutes les industries à la mise en place de plateforme électronique. Il en résulte la baisse des interactions face à face. Au sein des banques, la diminution de 9 % des postes dans les fonctions de conseils, entre 2008 et 2012, annonce bel et bien l'avènement des contacts dématérialisés. En effet, les chat-bots ou les robots-advisors semblent en phase avec les souhaits des générations Y et Z (Gasser et al., 2018).

Dans un second temps, nous traversons une période où les taux d'intérêt n'ont jamais été aussi bas. Le loyer de l'argent en Europe est à 0 %, en Suisse à -0,750 %, au Japon -

0,1 % seul celui des États-Unis repart légèrement à la hausse avec 2,5 % (Global Rates, 2019). Il est évident qu'en ces temps, les marges des services financiers sont fortement impactées.

Figure 32: "Risk-free" deposit revenue as a % of total bank revenue



(Oliver Wyman, 2018)

Dans un dernier temps, en plus des pressions sur les marges exercées par la baisse des taux, les régulations au lendemain de la crise financière se sont multipliées et elles sont accompagnées de coûts importants. Depuis 2008, les coûts estimés pour l'industrie bancaire s'élèveraient à $780 Mia uniquement pour la mise aux normes. À cela, il faut rajouter les amendes qui entre 2008 et 2018 se seraient montées à 321 $ Mia (Roy et al., 2018). Beaucoup de banques indiquent que parfois 50 % des coûts de leurs projets sont induits par la mise aux normes de leurs structures (Gasser et al., 2018).

Dans un environnement, où les grandes tendances technologiques portées par les nouvelles générations impliquent la venue de nouveaux entrants, où les taux d'intérêt historiquement bas et la hausse des coûts liés aux nouvelles régulations font pression sur les marges du secteur ; les institutions financières se doivent d'agir. En effet elles doivent saisir les nouvelles opportunités technologiques afin améliorer leurs services ou créer de nouveaux produits pour répondre au mieux aux besoins de leurs clients. De cette manière elles continueront à rester des intermédiaires crédibles face à de nouveaux concurrents sans leurs expériences et savoir-faire. Parmi toutes les innovations qui touche le secteur, la Data Science est une priorité.

4.1.3 La place de la Data Science en Finance

Selon un rapport d'Accenture, en 2016, près de 70 % des entreprises de services financiers exploraient déjà des solutions liées aux Big Data et à l'analyse prédictive. Environ la même part pensait déjà à cette époque que la maîtrise du Big Data était d'une importance capitale pour l'avenir de leurs métiers. À tel point qu'à cette même époque, près de 54 % des firmes du secteur avaient créé des postes de CDO ou Chief Data Officer (Sarrocco et al., 2016). Le marché du Big Data en 2018 représentait 166 milliards de dollars en hausse de 11,7 % par rapport à 2017. Les services financiers seraient les premiers investisseurs dans ces nouvelles technologies en étant responsable de 13,6 % des dépenses dans ce domaine, soit à eux seuls plus de 20 milliards de dollars (IDC, 2018).

Ces chiffres nous montrent l'importance de la Data Science au sein du secteur financier et la détermination de ces acteurs à ne pas se laisser dépasser par leurs concurrents. Le besoin pour des « data-driven businesses » comme les banques ou les assureurs d'investir dans le Big Data est clair. En outre, IBM estime qu'au sein des banques à travers le monde encore plus de 220 milliards de lignes de codes écrites dans le langage COBOL (langage de programmation datant de 1959) régissent encore plus de 70 % des transactions (IBM, 2017). Si nous ajoutons à cela le fait que 80 % des dépenses en informatiques de ces mêmes banques seraient dédiés à la simple maintenance des applications existantes, il est facile de comprendre le besoin de renouveau et d'efficience.

En quoi les architectures Big Data et la Data Science peuvent apporter une partie des réponses à ces problématiques ? Nous allons tout d'abord faire un aperçu de l'environnement de données dans lequel évolue le secteur financier actuellement pour finaliser ce chapitre avec la description d'un ensemble de uses cases pertinentes de la Data Science appliquée à la finance.

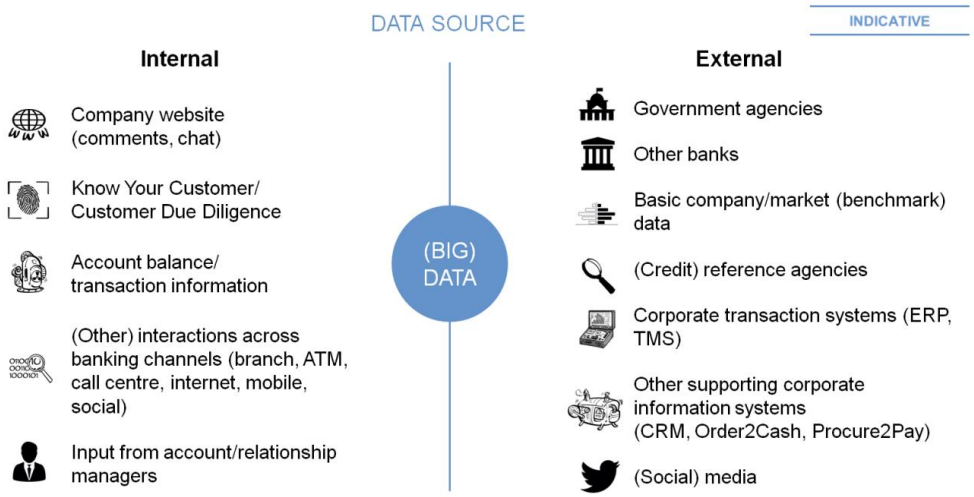
4.2 Un aperçu des données financières

Entre 80 % et 90 % des données que les secteurs financiers possèdent sont de types semi ou non structurés c'est-à-dire sous forme de texte, documents et autres. Nous comprenons ainsi l'intérêt grandissant pour les technologies Big Data des compagnies financières (Oracle Corporations, 2012). L'exploitation des de ces données à travers la Data Science est la clé de voute des nouveaux modèles d'affaires possibles pour les institutions financières. Ces dernières regorgent de données en tout genre et il est difficile de toutes les référencées et quantifiées. Cependant notre travail nous amènera à manipuler des données similaires à celles stockées dans les systèmes informatiques d'une banque, il nous semble donc important de tenter de définir un certain cadre.

4.2.1 Classifications des sources de données du secteur financier

L'Euro Banking Association propose une classification des données en termes d'accessibilité. L'EBA différencie les sources de données internes des externes. Inutile de préciser que la figure 33 ci-dessous n'est pas exhaustive.

Figure 33: High-level view of available data sources from a bank point of view



(Szmukler, 2017)

Les données internes sont le fruit des processus propres à l'entreprise financière, ici une banque, alors que les données externes sont générées par des tiers tels que des fournisseurs de données financières, des entreprises, les médiaux sociaux ou encore des agences gouvernementales. La digitalisation des différents processus interne a permis l'apparition de nouveaux canaux de communication et donc la création de nouvelles données en grandes parties non structurées à l'instar du E-Banking des banques ou les Customer Portal des assurances.

Parallèlement, l'entreprise Oracle définit elle 3 classes de données pour le secteur financier (Oracle Corporations, 2015). Les deux grandes catégories de sources de données sont claires, les données de marchés et entreprises et celles des clients.

Tableau 8: Data Type in the Financial Services Industry

Types	Exampes
Customers Data	ATMs Call Centers Web-based and mobile sources Branches/Brokerage units Mortgage units Credit cards Debt including student and auto loans Volatility measures that impact the clients' portfolios
Financial Business Forecasts	News Industry data Trading data Regulatory data Analyst reports (internal and competing banks) Alerts about events (news, blogs, Twitter and other messaging feed)
Other	Advertising response data Social media data

(Oracle Corporations, 2015)

Classifier les données des entreprises financières par leurs niveaux d'accessibilités ou par leur objet (clients, marchés, entreprises) permet d'appréhender un peu plus les opportunités qu'elles recèlent. Avant de nous attarder sur les nombreux usages faits de ces données, nous allons les décrire un peu plus en détails

4.2.2 Zoom sur les données de marchés et d'entreprises.

Les données financières telles que celles des exchanges à travers le monde possèdent assurément les caractéristiques d'exhaustivité et de vélocité au sens de Kitchin et McArdle évoqué dans notre première partie. Il suffit de scruter un terminal de fournisseurs de données financières lors des heures de trading et cela frappera n'importe quel observateur. Chaque transaction est enregistrée dans son ensemble et en temps réel. D'un point de vue quantitatif, les volumes de transactions pour les deux exchanges les plus importants, sont d'environ 1,2 milliard pour le NASDAQ (Nasdaq, 2019) et prêt de 1 milliard sur le NYSE (Nasdaq, 2019), et ce en une seule journée. Le marché des actions seul compte des échanges évalués à 77 566 trillions de dollars en 2017 (World Federation of Exchanges, 2018b) soit 118 % du PIB mondial (World Federation of Exchanges,

2018a). Rien que le New York Stock Exchange générait à lui seul 2 térabytes de données de transactions dans les tradings hours en 2012. Emile Werr, directeur de l'architecture de donnée du NYSE, projetait à cette même époque une hausse jusqu'à 10 petabytes en 2015 (Turner et al., 2013).

Le marché des fournisseurs de données financières fleuri avec une hausse de revenus des 9 leaders (dont Bloomberg, Factsets, Thomson Reuters…) de 33,8 % entre 2010 et 2017 pour atteindre environ 40 milliards de dollars (Harris, 2018). Si nous regardons la factsheet de Worldscope, la base de données de Thomson Reuters, le nombre de données disponibles est impressionnant et ce « data model » est similaire pour tous les concurrents du secteur.

Figure 34: Worldscope Database Key Figures

Nombre compagnies	51 100
Nombre de pays	70
Indices ciblés	FTSE All worlds, Dow Jones, MSCI World, MSCI EMF, S&P Global, S&P/Citigroup
Base Year	1980

(Thomson Reuters Financial, 2007)

L'étendue des données par son ancienneté et sa géographie est déjà immense, mais c'est en se penchant sur les différents data points ou mesures disponibles que nous découvrons l'ampleur de ce type de bases de données.

Figure 35: Worldscope—Measure Types

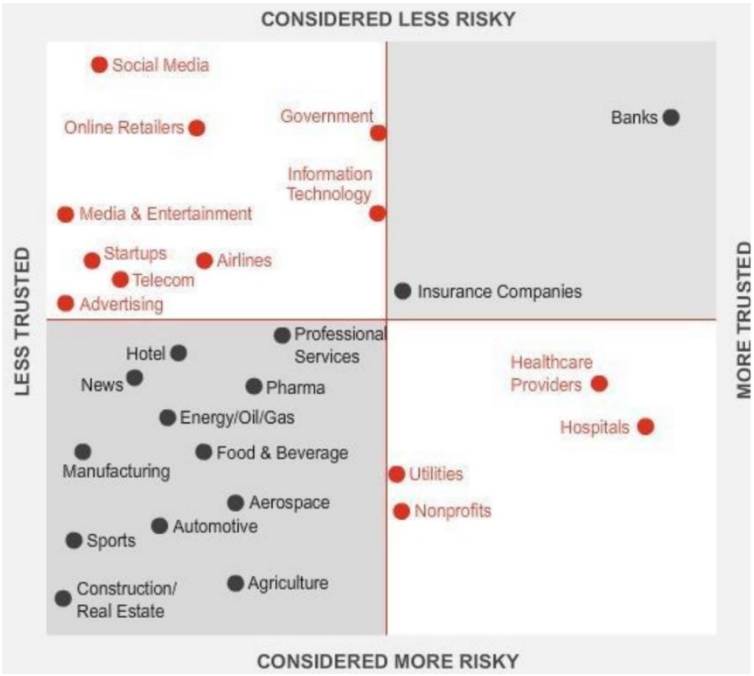
Type	Mesures
Company Header Information and CV Profiles	Address, SIC Codes, Auditors Company Status, Officers, Product segment, geographic segments, business descriptions, investor relations
Financial statements	Balance sheet, Income statement, Cash Flows statement
Historical Growth Rates & Valuation Ratios	Growth, Profitability, Asset Utilization, Leverage, Liquidity, Foreign Business
Security and Market Data	Multiple Share Info, Stock Prices, Stock Performance, SEDOL/ISIN, CUSIP/Ticker, Exchange listing(s)
Others	Exchange Rates, Global Industry Groups, Key Index Membership

(Thomson Reuters Financial, 2007)

Pour les entreprises de services financiers, Thomson Reuters décrit environ 1300 points de mesures différentes et d'indicateurs. Les sources de ces données sont principalement : les compagnies elles-mêmes, les exchanges, les régulateurs et la presse financière. L'exhaustivité de ces données ne fait aucun doute. Certaines de ces données sont générées en temps réels comme tous les indicateurs de performances et d'autres sont en différé comme les données comptables. L'apparition du High Trading Frequency quant à lui est également un bon indicateur de la vélocité des données financières. *« High-frequency trading means to rapidly trade large volumes of security by using automated financial tools »* (Tian et al., 2015). En 2012, 85 % du volume total aurait été généré par ce genre de pratique contre seulement 15 % en 2003 (Atkins et al., 2018). La course à la « low latency » et aux modèles prédictifs fait rage en finance et démontre l'importance de la vélocité dans l'exploitation des données de marchés. Il est évident que les données de marchés représentent une immense part de la stratégie des banques. Cependant, les banques comme les autres services financiers n'existeraient pas sans leurs clients et c'est pourquoi leurs données ont une importance capitale dans les processus de data management du secteur financier.

4.2.3 Les données clientes

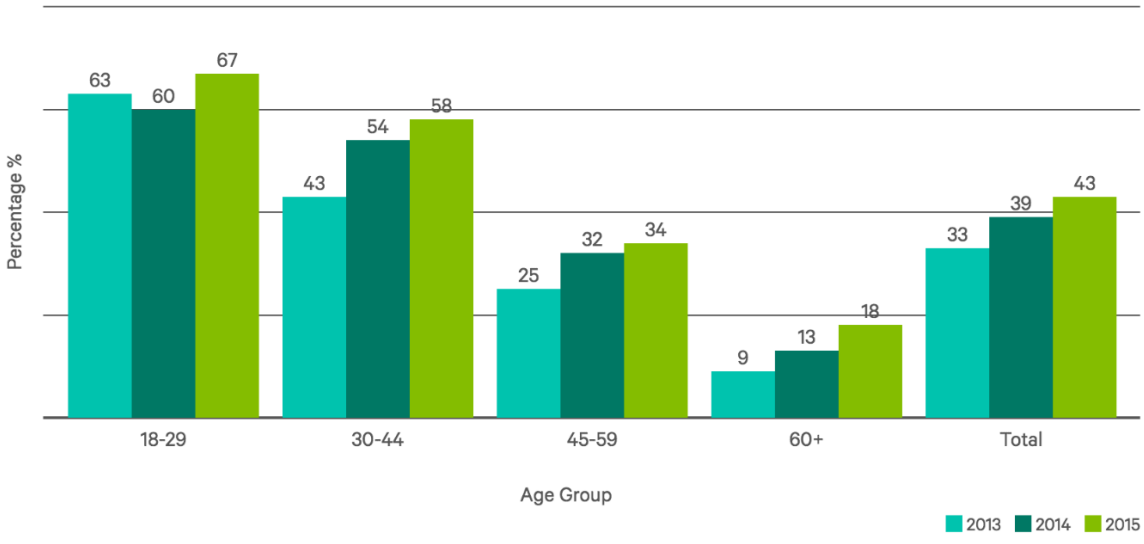
Figure 36 Consumer ratings of trustworthiness versus risk of cyber-attack across various industries:



(WEF, 2019)

À l'heure actuelle, les consommateurs pensent en majorité que les banques et les assureurs sont les industries les plus à même de gérer leurs données personnelles de la manière la plus sûre possible face aux risques encourus. Cela joue grandement en leurs faveurs, ces entreprises doivent capitaliser sur cette confiance.

Figure 37: Mobile Banking penetration rate by Age



(ValuePenguin, 2019)

Si nous ajoutons à cette confiance, la sensible augmentation du digital Banking à travers les smartphones, nous ne pouvons que supposer l'augmentation des données clientes reçues et générer par le secteur financier. De plus, les interactions clients via les canaux digitaux comptaient déjà en 2015 pour 30 % du total et McKinsey projetait 50 % d'ici à 2020 (McKinsey & Company, 2018). Cela ne peut être que corrélé avec une augmentation des données disponibles pour les banques et assurances.

Pour commencer, lors de toutes ouvertures de relations bancaires par exemple, les clients sont invités à partager des justificatifs de leurs identités, des possibles autres ayants droits économiques, des valeurs patrimoniales, de l'origine des fonds ou des dernières transactions récurrentes faites. D'un point de vue basique, c'est ce que possède une banque à l'ouverture d'une relation. Ces informations sont bien évidemment enrichies après une enquête interne sur le background du client afin de vérifier son statut d'un point de vue compliance. Puis dès l'ouverture, la banque commence à recevoir d'autres types de données propres à la banque : Types de produits utilisés, s'il investit quel type de profil de risques, ses revenus moyens, etc. Un profil financier se construit.

Puis, à notre ère digitale, ce profil est encore enrichi par l'arrivée de méta données via l'utilisation de smartphone ou du web et médias sociaux. Les données de localisation, de timings d'utilisation de telles ou telles applications, sur quel site notre client était avant d'aller sur son eBanking, etc. Toutes ces données permettent aux banques de construire un profil encore plus précis de leurs clients. Dans un de leur rapport destiné aux bonnes pratiques à adopter par les institutions financières lorsqu'elles exploitent les données clients, le WEF nous propose une comparaison non exhaustive des anciennes et nouvelles données disponibles.

Tableau 9 : Traditionnal & Emergin types of customer data

	Traditionnal	**Emerging**
Identity	Public Records, Tax Filling	Fingerprints, photograph, iris scans, digital IDs
Health	Medical Records, Insurance claims	Fitness tracking, sleep/eating habits
Financial	Bank Statement, credit scores	Peer-to-peer payment, online budgeting
Social	Organization registry	Social media connections and activities
Location	Telephone books	Geolocalisation tracking
Media Behavior	Library check out histories	Web browsing activities, content streaming

(World Economic Forum, 2018)

Il est clair que désormais les clients seront de plus en plus scrutés pour des raisons de sécurités, mais bien évidemment de marketing également.

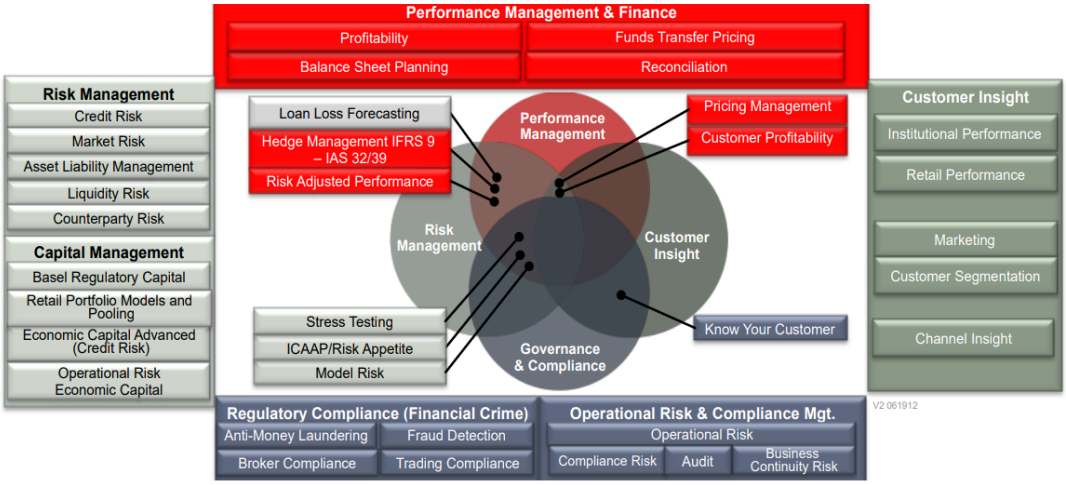
Afin de nous plonger plus profondément dans ce que font exactement les services financiers avec l'ensemble de ces données, nous allons poursuivre avuec une brève présentation des principaux uses cases de la Data Science en finance.

4.3 Les différents uses cases

Oracle définit 4 grands arcs où la Data Science et l'analytique peuvent avoir un impact considérable sur les activités des services financiers :

- Le management de la performance
- La connaissance du client
- La Gouvernance et Compliance
- Le management du risque

Figure 38: Analytics in Financial Services



(Oracle Corporations, 2015)

Dans le cadre de ce canvas, nous allons décrire quelques-uns des nombreux uses cases de la Data Science appliquée à la finance.

4.3.1 Le management de la performance

4.3.1.1 Le trading algorithmique :

4.3.1.1.1 Le trading haute fréquence/high frequency trading (HFT)

Parmi les pratiques en termes de trading algorithmique, le trading haute fréquence est l'une des pratiques les plus répandues. Elle est définie par (Agarwal, 2012) :

- Utilisations d'algorithme ultra-performant et rapides pour générer et effectuer des ordres boursiers
- Utilisations de canaux de communications souvent privés avec les exchanges afin de minimiser les latences
- Taux de turnovers des positions très élevés
- Peu de position overnights, tout est liquidé.

Le trading haute fréquence se nourrit des données des exchanges. Environ 50 Milliards de data points sont générés par jours par les marchés américains (Gutierrez, 2017). Autant de points de mesures que les stratégies implémentées dans les algorithmes par les quantitatives analysts peuvent exploiter à leurs avantages. Les différentes stratégies utilisées par ces algorithmes sont les suivantes :

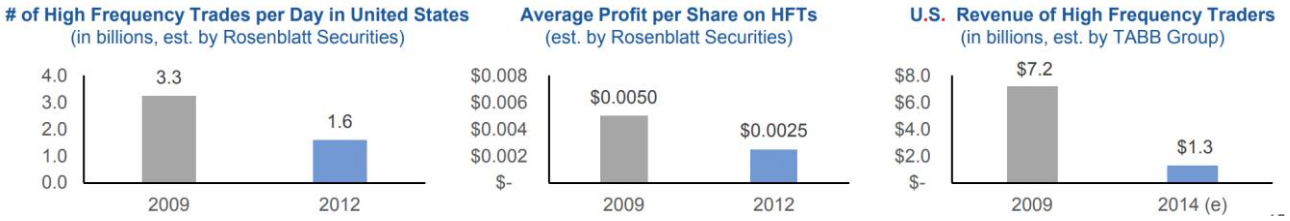
- **Market Making :** Placements d'ordre limite de vente au-dessus du prix courant ou placement d'ordres d'achat en dessous du prix courant afin que lorsque le prix atteint cette limite l'algorithme effectue l'action et profite donc du bid/ask spread.

- **Liquidity Rebate Tarding :** Achat et vente au prix courant de quantité importante d'actifs afin d'obtenir des rabais de la part des exchanges en qualité de liquidity providers.
- **Statistical Arbitrage :** La recherche d'arbitrage entre les marchés et les différents actifs est également une stratégie importante dans le trading haute fréquence. Ils sont entre autres la raison de leurs absences à l'œil des investisseurs traditionnels. En effet, lorsque par exemple un arbitrage entre un produit dérivé et son sous-jacent apparaît, il n'est pas rare qu'à la seconde ou celui-ci survient déjà plusieurs algorithmes l'aient repéré et effectué les trades nécessaires. Ainsi les prix convergent et l'arbitrage disparaît et ceux en une fraction de seconde.
- **Momentum Ignition :** Placement d'ordre de taille importante puis annulation. Avec de telles actions, les algorithmes de Momentum Ignition peuvent déclencher les algorithmes des autres acteurs du marché et ainsi créer un mouvement de marchés. Il leur suffit ensuite de prendre les positions en face des stratégies des algorithmes concurrents.
- **Leveraging Structural Differences :** Ici nous parlons essentiellement de la façon d'obtenir les données. L'emplacement des serveurs, la connectique, la privatisation de la bande passante ou des infrastructures. Plus les installations sont proches des exchanges et moins il y a de personnes les utilisant plus les données sont reçues tôt par les algorithmes. Certains voient ces pratiques comme du délit d'initiés, mais à la fraction de seconde près. Imaginez qu'avec ce genre d'avantages, les algorithmes de trading perçoivent les ordres passés par leurs concurrents une fraction de seconde avant que ceux-ci soient exécutés. Avec ce genre d'informations, il leur suffit par la suite de prendre la position inverse afin d'être sûrs de réaliser l'opération à un certain prix.

La plupart des stratégies relèvent de concepts de finance retranscrit en algorithmie par des développeurs. Pour le HFT, les banques et gestionnaires doivent non seulement obtenir les données d'une manière efficace mais également traduire leurs points de vue sur le marché afin d'obtenir une martingale pendant un temps. La vitesse d'exécution de ces algorithmes est de l'ordre de la milliseconde, l'efficacité prime. Ainsi non seulement les stratégies financières se doivent d'être parfaites mais les modèles informatiques également pour espérer devancer ces concurrents. C'est un bel exemple de la data science au service de la finance.

Les grandes années du HFT étaient de 2009-2010, ou plus de 60 % des trades étaient réalisés par des machines. Depuis le trading haute fréquence a été quelque peu délaissé à cause de la volatilité faible des marchés (manques d'opportunités), une liquidité améliorée, les coûts d'infrastructures élevés, mais surtout l'intérêt grandissant des régulations pour ces pratiques et leurs encadrements.

Figure 39: High Frequency Trading Evolution



(McWaters, 2015)

4.3.1.1.2 Robo-advisory :

Avec l'avènement de l'intelligence artificielle, du Big Data et la perte de confiance envers les institutions financières en 2008, le concept de Robo-advisory et d'automated trading sont apparu. Il est désormais possible d'automatiser la gestion de fortune. Ces robots advisors définissent en premier lieu un profil de risque et des objectifs d'investissement. Puis ils définissent l'horizon de temps et autres éléments nécessaires via un questionnaire comme la performance souhaitée tout comme le ferait un conseiller humain. Cette interprétation des réponses humaines et la formulation de recommandations serait impossible sans IA ni data science.

Après stockage de ces éléments, les robots advisors construisent une stratégie d'investissement basée sur les stratégies fondamentales tels que celle de Markovitz concernant l'allocation d'actifs. Ces robots rééquilibrent les portefeuilles automatiquement et proposent également parfois des solutions d'optimisations fiscales. Pour rester low costs, les stratégies offertes sont souvent composées d'ETF et autres instruments de gestions passives. L'accès à un conseiller financier n'a jamais été aussi simple.

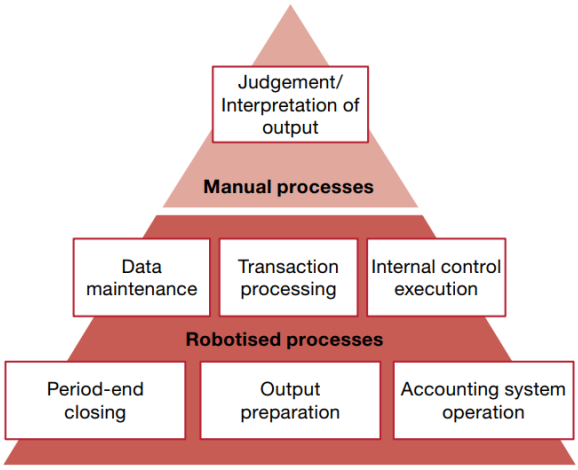
Les USA comptent parmi les 1ers investisseurs dans les robots Advisors, captant 57 % des investissements totaux et décomptant plus de 200 robots onlines. La valeur des actifs sous gestions par ces robots américains serait évaluée à 400 milliards en 2018 avec un taux de croissance de 31 % annuel. Ainsi la masse sous gestion en 2023 pourrait atteindre près 1,5 Trillons de dollars. En Europe, nous comptons plus de 70 de ces machines et le rythme de croissance est le même (Abraham et al., 2019).

4.3.1.2 Robotic Process Automation:

L'automatisation des processus est également l'une des applications de la Data Science dans l'amélioration des performances des services financiers. Le lien entre l'automation et la Data Science peut paraître de prime à bord faible, mais en réalité ces deux éléments sont totalement complémentaires.

Grâce à l'automatisation de certains processus, les données ne sont plus capturées par des humains et donc la consistance des données (sauf erreur de programmation et/ou d'architecture) est elle-même renforcée. Ainsi, l'entreprise peut par la suite surveiller ces processus avec une meilleure vue d'ensemble (process mining) ou encore simuler un changement de processus de manière très simple (process simulation). De plus le machine learning et l'IA peuvent ensuite être utilisées pour optimiser ces mêmes processus ou proposer de nouvelles solutions (Tian, 2018). Il est vrai que tous les processus ne sont pas automatisables et lorsque le jugement ou l'interprétation est de mise, l'humain reste indispensable.

Figure 40: Tasks which can be automated



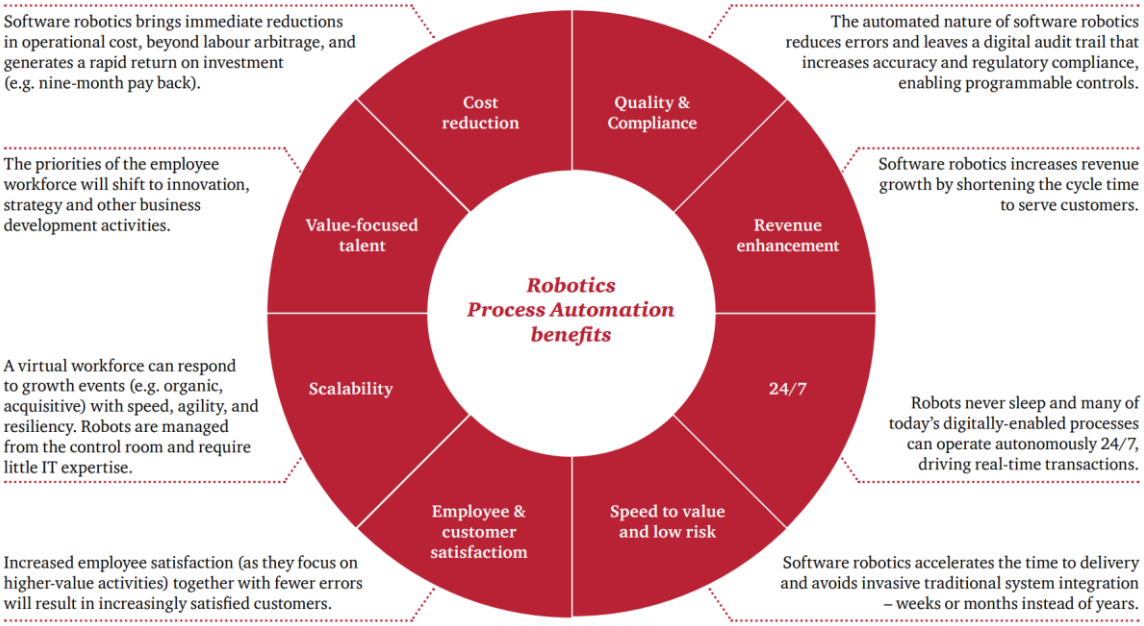
(Mäder, Akiki, 2017)

Avec la génération et le stockage des données de l'ensemble des processus automatisés, les services financiers peuvent augmenter leurs efficiences et leurs compétitivités. Par ailleurs ce use cases ne s'appliquent pas essentiellement à la finance, mais à n'importe quelle industrie. À l'heure actuelle, selon un sondage d'IBM, 91 % des entreprises auraient automatisé certains de leurs processus (Chao et al., 2018). Sachant que l'automatisation a plusieurs niveaux de complexité, une part aussi importante reste tout de même un excellent indicateur de la tendance. Ces niveaux de complexités sont décrits par IBM :

- **Intelligent :** Le robot en charge du process possède une certaine intelligence pour prendre un certain nombre de décisions vis-à-vis des entrées de données
- **Avancée :** le robot poursuit des chemins prédéfinis par les algorithmes en effectuant des analyses tout de même complexes souvent basés sur des techniques avancées d'intelligence artificielle.
- **Basique :** Le robot poursuit uniquement des chemins prédéterminés pour des tâches simples.

Les avantages retirés par un tel changement sont clairs et la figure 39 résume parfaitement ces derniers.

Figure 41: Benefit from the RPA



(Mäder, Akiki, 2017)

4.3.1.3 Back testing investment strategy

Le Back testing de stratégies correspond à l'action d'utiliser des données de marchées passées pour simuler un comportement des cours dans le futur. Grâce à cela les analystes peuvent tester leurs hypothèses et stratégies sur une simulation du futur. Après simulation des indicateurs tels que le Sharpe Ratio, le maximum drawdawn, Total P/L, VaR etc. sont calculés pour valider ou non la stratégie.

Selon l'article 12.4 3 ii des accords de Bâle *"A trading desk must produce, at least weekly, appropriate risk management reports. This would include (…) internal and regulatory risk measure reports, including trading desk value-at-risk (VaR)/expected shortfall (ES), trading desk VaR/ES sensitivities to risk factors, back testing and p-value."* (Bank for international settlements, 2019). Ainsi, la pratique du backtesting en risque management est une obligation même si la liberté du modèle utilisé est laissée au choix des banques.

D'un point de vue technique, les données utilisées sont des données de marchés et donc structurées le plus souvent stockées dans des RDBMS classiques. Pour une simulation de Monte-Carlo pour 5 000 scénarios à 500 000 trades chacun, la quantité de données moyennes générées peut atteindre facilement 5-6 térabytes (Oracle Corporations, 2012). Ces simulations impliquent énormément d'écriture et de lecture, mais l'intégrité complète

des transactions comme pour un service de paiement n'est pas nécessaire. Ainsi des bases de données de types NoSQL voir des architectures Big Data permettrait d'accélérer ce type de processus.

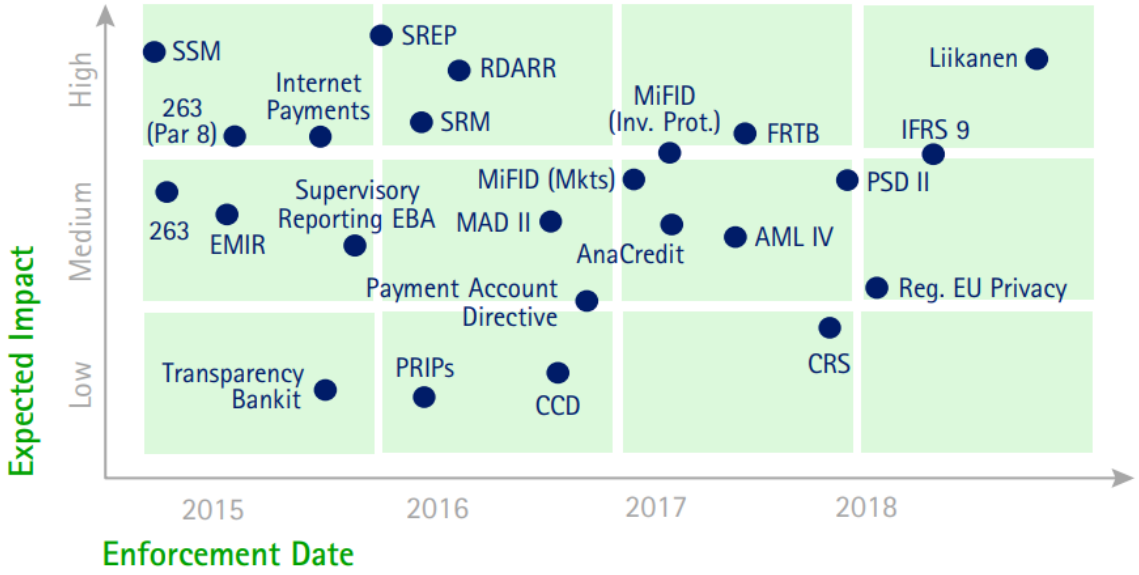
Par exemple, une simulation de la VaR avec l'aide de l'algorithme MapReduce lancé en parallèle sur un cluster a permis de faire baisser son temps d'exécution de 15 h heures sur 200 nœuds à 20 minutes sur uniquement 60 nœuds. Au-delà de la vitesse d'exécution, le nombre de simulations réalisées avait également augmenté (Oracle Corporations, 2012).

L'augmentation du nombre de simulations et de leur vitesse permet aux banques de valider des modèles plus complexes à plus grandes échelles. Entre conformité aux régulations et hausse de la performance, la Data Science ici est un véritable atout. Dans notre prochaine partie nous allons nous attarder sur les autres risques et le domaine de la compliance où la Data Science peut encore jouer un rôle.

4.3.2 Le risk management et la régulation

Comme nous l'avons cité précédemment, l'environnement légal du secteur financier est très strict et son évolution est constante. En 2018, deux tiers des entreprises du secteur s'attendaient à une hausse des coûts liés à la compliance et près de 43 % prévoyaient une hausse de leurs équipes de compliance. Autre fait intéressant 41 % des compagnies déclaraient la même année évaluer les solutions proposées par des fintechs en termes de regtech, comprenez regulations technology (Thomson Reuters Financial, 2018).

Figure 42: Regulation enforcement date versus impact



(Sarrocco et al., 2016)

La figure 42 ci-dessus vous donne un aperçu du nombre de règles créées liées à l'activité des services financiers. Maîtriser ces risques, ces coûts et implémenter les nouveaux processus de régulations de manières efficientes sont des points essentiels. Comment la Data Science peut-elle aider ?

4.3.2.1 Le credit scoring

« summary of a person's apparent creditworthiness that is used to make underwriting decisions » (Hurley, Adebayo, 2017a). Le plus vieux système de ce type est le FICO score aux USA. Ce score compile des informations telles que l'historique de paiements, l'historique de crédit ou l'utilisation des crédits à disposition. Aux USA les 3 grandes agences de références en matière de crédit reçoivent environ 1,3 milliard de mise à jour pour 200 millions d'utilisateurs par jours, et ce uniquement sur le type de données précitées (Hurley, Adebayo, 2017 b).

De nouveaux arrivants sur le marché du scoring de crédit propose des solutions 100 % compliance, augmentant le taux d'approbation et diminuant le taux de défaut. Comment font ces nouvelles structures ? Bien qu'elles utilisent les données conventionnelles de crédit tel que FICO le fait, ces sociétés enrichissent les profils des clients avec des données bien souvent dites comportementales.

Tableau 10: Credit scoring data examples

Entreprises et produits	Exemple de données
LexisNexis—RiskView	Residential stability,asset ownership, life-stage analysis, property deeds and mortgages, tax records, criminal history, employment and address history, liens and judgments, ID verification, and professional licensure
FICO — Expansion Score	Purchase payment plans, checking accounts, property data, public records, demand deposit account records, cells and landline utility bill information, bankruptcy, liens, judgments, membership club records, debit data, and property asset information.
Experian—Income Insight	Rental payment data, public record data
Equifax—Decision 360	Telco utility payments, verified employment, modeled income, verified income, spending capacity, property/asset information, scheduled monthly payments, current debt payments, debt-to-income ratio, bankruptcy scores
TransUnion—CreditVIsion	address history, balances on trade lines, credit limit, amounts past due, actual payment amount
ZestFinance	Major bureau credit reports and thousands of other variables" including financial information, technology usage, and how quickly a user scrolls through terms of service
LendUp	Major bureau credit reports, social network data, how quickly a user scrolls through its site
Kreditech	Location data (e.g., GPS), social graphing (likes, friends, locations, posts), behavioral analytics (movement and duration on a webpage), e-commerce shopping behavior, device data (app installed, operating systems).
Earnest	Current job, salary, education history, balances in savings or retirement accounts, online profile data (e.g., Linkedln), and credit card information.
Demyst Data	Credit scores, occupation verification, fraud checks, employment stability, work history, and online social footprint

(Hurley, Adebayo, 2017 a)

Les tableaux 9 vous donneront un aperçu des types de données usitées par ces nouvelles méthodes. La confidentialité et le respect de la sphère privée peuvent évidemment faire débat. Cependant ce n'est pas le sujet de notre travail donc nous le mettrons de côté pour la suite de notre analyse. Ainsi, ce type de pratique nous montre comment le machine learning et la Data Science peuvent permettre l'amélioration des prévisions de défauts et diminuer le risque des entreprises émettrices en utilisant de nouveaux types de données. Beaucoup de ces nouvelles données sont non structurées et nécessitent donc une architecture adaptée pour être convenablement exploitée. Encore une fois Hadoop & Spark sont pleinement capables de gérer ces nouveaux insights pour aider les banques à faire les meilleurs choix stratégiques.

4.3.2.2 Détection de fraudes

53 % des institutions financières ont augmenté leurs coûts liés aux crimes économiques entre 2014 et 2016. Entre 2012 et 2017, le secteur financier a vu une hausse de 246 % des pertes dues aux fraudes de cartes de crédit. 342 milliards de dollars c'est le montant total des amendes infligées pour non-compliances des structures financières, et 89 % des directeurs s'attendent à voir ces coûts se multiplier (McWaters, 2015). Ces quelques chiffres nous montrent l'impact grandissant des fraudes en finance. La compliance et le besoin de monitoring des clients n'ont jamais été aussi importants. Malheureusement à l'heure actuelle, seuls 16 % des institutions sont capables de détecter les fraudes avant qu'elles ne soient réalisées (IBM, 2017).

Les technologies Big Data permettraient de mettre en relation des données et découvrir des relations jusque-là impossibles à déceler. Des exemples donnés par Oracle seraient, la mise en relation des appels et/ou e-mails envoyés par un trader juste avant l'exécution d'un ordre boursier important. Les rogues traders pourraient être détectés bien plus facilement. Un autre exemple serait la géolocalisation du client via son smartphone lorsque ces cartes de crédit ou de débit sont utilisées dans le monde (Oracle Corporations, 2015). Les comportements onlines des clients ou les adresses IP corrélées avec les informations traditionnelles nécessaires pour poursuivre les recommandations compliance KYC (Know your customer) dépasseraient facilement les attentes des régulateurs. De plus l'automatisation des processus et la suppression des interactions et contrôles humains permettront également d'améliorer la qualité des données et leurs contrôles en cas de problèmes. Enfin le machine learning couplé avec les infrastructures Big Data permettrait de scruter toutes ces données en « streaming », soit en direct. Ceci permettrait aux institutions financières de mettre en lumière des fraudes bien plus vite

qu'actuellement et ainsi réagir plus rapidement également en bloquant des transactions ou en demandant des identifications supplémentaires.

4.3.2.3 Analyses des cyberattaques

En septembre 2017, l'agence de crédit américaine Equifax fait face à une brèche informatique au travers de laquelle les données de 143 millions d'Américains sont piratées (WEF, 2019). En outre, une analyse d'IBM explique que les acteurs financiers reçoivent 65 % de cyberattaques en plus que les autres secteurs. Entre 2013 et 2016, plus de 3000 brèches ont été reportées dont 40 % auraient mené à des pertes de données (IBM, 2017). Dans un environnement où la RGPD est désormais instituée en Europe et où la sphère privée sur internet fait débat, les acteurs financiers qui possèdent des données dites sensibles sur leurs clients se doivent de les protéger comme il se doit.

Les digitalisations des banques apportent non seulement son lot de nouvelles données, mais également de nombreux points d'entrées pour les cybercriminels. Par exemple, la loi PSD II (Payment Service Directive II) oblige les banques de l'UE à fournir l'accès aux comptes bancaires de ses clients à certains tiers agrégés via des open API. Cette loi a été mise en place pour favoriser l'arrivée des nouveaux acteurs tels que les fintechs ou les bigtechs. Cela peut être vu d'un bon œil concernant l'innovation, mais la sécurité des données s'en retrouve menacée. L'ouverture des données à des tiers entraîne forcément de nouvelles menaces. Quelles sont les infrastructures de ce tiers ? Sont-elles aussi lourdement sécurisées que celle de la banque ?

Une société américaine comme Guidewire, développeur de software, spécialisée dans les logiciels d'analyses de risques pour les assureurs propose désormais des solutions comme Cyence Risk Analytics qui a déjà convaincu des organisations financières comme le S&P500 (Piecuch, Stott, 2018). Pourquoi ? Guidewire pratique le « data listening » qui consiste à collecter et surveiller des données techniques (web logs, network…) et comportementales des utilisateurs puis d'appliquer des analyses à l'aide du machine learning pour déceler des cyber attaques et ainsi les contrecarrés (Guidewire Software, 2018). Ce genre de pratique permettrait aux institutions financières de diminuer leur exposition au risque grandissant des cyberattaques.

Dans la dernière partie de notre chapitre, nous allons maintenant nous concentrer sur les clients et ce que la Data Science peut leur apporter en termes de plus-value.

4.3.3 La connaissance du client

Auparavant, les relations qui liaient une banque et ses clients étaient établies sur des années et étaient souvent transgénérationnelles. Désormais avec l'explosion

technologique et l'arrivée des nouveaux concurrents, le marché voit s'opérer une baisse des coûts de transfert pour les clients. En 2012, près de 50 % des clients de banques avaient soient changé soit planifié de changer de banque (Gutierrez, 2017). Plus récemment il a été estimé qu'environ 41 % de la génération dite Z seraient prêts à acheter des services financiers à des fournisseurs tels que Google ou Amazon (Levi et al., 2018). Il n'est pas rare que ces derniers possèdent plusieurs types de comptes auprès de différents fournisseurs, voir change de banque pour leurs crédits et dépôts. Cette versatilité empêche les services financiers de conserver une vue d'ensemble de leurs clients et de leurs comportements. Ainsi, le besoin d'enrichir les données déjà disponibles dans les CRM internes avec des données externes s'est peu à peu développer. Les médias sociaux, les navigateurs web, la géolocalisation, tout est bon pour obtenir une vue à 360 ° des comportements de leurs clients. Près de 71 % des banques estiment que la hausse de leur revenu proviendra d'une meilleure compréhension de leurs clients et que le meilleur moyen d'y parvenir sera la maîtrise des technologies Big Data (Gutierrez, 2017.)

La notion de temps réel avec le streaming et le volume de données non structurées poussent les banques et assurances vers les technologies Big Data et la Data Science pour tirer un maximum de profit de ces nouveaux éléments et visiblement cela fonctionne. Pour exemple, les fournisseurs de paiement ayant mis en place des technologies d'analytiques avancées ont augmenté leurs satisfactions clients de 5 à 10 % tout en baissant leurs coûts opérationnels de 15 à 20 % (McKinsey & Company, 2018). De plus, les banques qui analysent précisément leurs données clientes possèdent des parts de marchés légèrement plus élevé que leurs concurrents (~4 %) (Coumaros et al., 2014).

4.3.3.1 Acquisition de clients & Limiter les départs

Une banque américaine aurait maximisé son taux de conversion de prospect de 100 % en décidant d'insuffler dans son CRM le résultat d'une solution d'analyses de données clientes on et offline. Ces données auraient également été utilisées pour fournir des « insights » ciblés à leur call center, mais également des recommandations à leurs web developer afin d'améliorer leur design et le rendre plus agréable aux utilisateurs. Pour une grande banque européenne, l'analyse prédictive des données clientes internes et externes aurait augmenté son taux de conversion de 7 fois son niveau avant la mise en place de ces techniques (Coumaros et al., 2014).

Une banque européenne de plus de 2 millions de clients avec environ 200 data points pour chacun à réussi à créer un modèle prédictif pour identifier les potentiels prochain départ. Ce modèle génère une « score card » à travers des techniques d'intelligence

artificielle puis alerte les relations managers. Cela a permis à la banque d'éviter des pertes d'environ 30 € Mio par année (Coumaros et al., 2014).

4.3.3.2 Marketing ciblé :

Imaginez une cliente, Jane Doe. Cette dernière est célibataire et reçoit sa paye sur son compte en banques tous les mois. Si sa banque surveillait ses réseaux sociaux, elle saurait que Jane venait de se fiancer et pourrait lui proposer un « wedding loan ». Puis lorsque, Jane rechercherait une maison activement à travers différents sites, sa banque pourrait commencer à lui proposer des offres d'hypothèques. Ce concept s'appliquerait à la venue du premier enfant, de sa scolarité jusqu'à la retraite de Jane (Oracle Corporations, 2015). Cet exemple tiré d'un rapport d'Oracle et enrichi par nos soins peut rappeler des œuvres littéraires dystopiques. D'un point de vue purement business, ce type de marketing est d'une efficacité redoutable et possible qu'avec l'œuvre de la Data Science. Une des plus grandes banques américaines a vu ses revenus augmenter de $1,2 Mia (1,5 % des revenus annuels) à la suite de la mise en place d'un partenariat avec une entreprise spécialisée dans le marketing basé sur les données. Cette même banque aurait également augmenté son taux de conversion à 3,5 % pour une moyenne dans l'industrie à 0,4 % (Levi et al., 2018). À travers ces techniques ces banques ont donc été capables de créer des profiles sociaux précis de leurs clients, de découvrir des patterns dans leurs dépenses, d'identifier leurs canaux de communications et de transactions préférés. Grâce à cela elles ont ensuite été capable de leurs concevoir une offre adaptée aux moments où ils en avaient besoin.

4.3.3.3 Customer feedback analysis, Sentiment analysis et Brand reputation

Tout d'abord, attardons-nous sur quelques statistiques qui justifient l'engagement des marques sur les réseaux sociaux. 80 % des consommateurs sont engagés auprès de leurs marques préférées sur les réseaux sociaux. Qu'ils commentent, aiment les postes ou s'adressent directement aux services clients des entreprises qu'ils aiment ou chez qui ils sont clients, cette statistique ne peut être ignorée même pour les services financiers. De plus les clients sont le plus souvent connectés à leurs marques via une moyenne de 7 réseaux sociaux différents. La nécessité d'avoir une présence sur l'ensemble du paysage social d'internet semble donc évidente. Un client est prêt à dépenser 20 % de plus et est prêt à recommandé la marque 30 % fois plus lorsqu'il reçoit une réponse sur Twitter de sa marque (Walgrove, 2018). En 2012 seulement, 5 % des consommateurs utilisaient tweeter pour relater une bonne ou une mauvaise expérience avec leurs marques, en 2015 cette part des consommateurs explose à 17 %, et ce uniquement sur

Twitter. Enfin, 88 % des consommateurs font autant confiance aux recommandations online qu'offline (Cox, 2016).

Cet ensemble de statistique démontre l'importance de l'usage des médias sociaux par les consommateurs de nos jours. Les avis, les commentaires, les recommandations, tout se passent majoritairement sur ces plateformes désormais. C'est aux marques, les services financiers inclus, d'utiliser ces ressources à bon escient pour analyser et prédire les mouvements et réactions de leurs clients. À travers des techniques d'intelligence artificielle, comme le Nature Langage Processing, il est désormais possible pour les ordinateurs de donner une appréciation sur des commentaires humains. Si trop de commentaires négatifs sur une nouvelle offre de cartes de crédit apparaissent, peut-être est-il temps de modifier ou supprimer l'offre. Si plusieurs clients se plaignent des lenteurs du service client, peut-être est-il temps de modifier notre processus. Ces procédés permettraient aux banques de s'adapter en temps réel aux évolutions des avis. Cela leurs permettrait de répondre de manière adaptée aux attentes de leurs clients. C'est la route vers la mass customisation des services. L'exploitation de ces données non structurées requiert encore une fois des technologies Big Data et une équipe de data scientist pour tirer parti au mieux des possibles insights dégagés par ces données.

En conclusion, nous avons pu mettre en évidence les différents uses cases de la Data Science appliquée aux secteurs financiers. Que ce soit en termes de performance, de risk management ou de connaissance des clients, les services financiers ont désormais de nouvelles possibilités offertes par les nouvelles technologies et l'abondance de données disponibles. Bien que les quelques exemples fournis ne soient pas une liste exhaustive, nous apercevons l'importance cruciale de la Data Science dans l'avenir des services financiers en termes de rentabilités et de risques. C'est avec la ferme intention, de nous exposer concrètement à ces nombreuses opportunités que la dernière partie de notre travail résumera le développement informatique réalisé en parallèle de ce travail. L'intérêt de cette dernière partie n'est pas de mettre en place le prochain algorithme de trading d'un grand hedge funds ou le processus de crédit scoring d'une banque internationale. En effet, l'objectif poursuivi ici serait plutôt d'appréhender les technologies citées dans ce travail et de manipuler un jeu de données typiquement disponibles dans le monde de la finance.

5. Le développement informatique

Lors du choix d'un développement pertinent pour ce travail, plusieurs problématiques se sont posées :

- Quel serait le problème à résoudre ? Quel serait le but de ce développement ?
- Quelles données devront être utilisées ? Où les trouver ?
- Quelles technologies ? Comment les exploiter au mieux ?

Dans cette partie, nous allons répondre à ces questions puis nous consacrerons la fin du travail à l'analyse des résultats observés. Nous adopterons non seulement un point de vue financier, mais également d'un point de vue technique afin de savoir si les choix de technologies ont été pertinents.

5.1 Les objectifs

5.1.1 L'objectif technique

Dans la partie précédente, nous avons explorés les différents use cases de la Data Science appliquée à la finance. Une multitude de possibilités s'offraient donc à nous. Dans un premier temps, nous souhaitions déterminer quelle serait la plus-value du développement d'un point de vue technique. Une contrainte a très vite réduit les possibilités : l'utilisation d'intelligence artificielle. Afin de construire des modèles de prévisions et exploiter pleinement le potentiel du Big Data, l'utilisation de l'IA est désormais une pratique plus que répandue dans les équipes de Data Science. Sans connaissances poussées non seulement en informatique, mais surtout en matières quantitatives telles que les mathématiques les, statistiques, ou encore la physique la maîtrise des algorithmes de machine learning semblait compliqué en si peu de temps. Dès lors du choix du sujet, ce travail était déjà orienté vers l'architecture Big Data plutôt que sur les modèles multi facteurs concevables avec l'IA. Nous démontrerons donc l'avantage compétitif et technique des architectures Big Data par rapport aux technologies classiques de stockage et de calcul. Puis le développement devra également fournir un panel des possibilités qu'offre le langage le plus répandu dans la Data Science : Python.

5.1.2 L'objectif métier

D'autre part, le développement devait également avoir un intérêt d'un point de vue purement financier. La phrase suivante a régi notre choix : « Pourrions-nous utiliser ce développement le dans monde professionnel ? » La gestion de portefeuille et de leurs risques ouvre un champ conséquent de possibilités en lien avec le monde professionnel et la filière de ce bachelor. Ainsi, notre choix se tournait peu à peu vers les marchés financiers. L'exploitation des données clientes pour le marketing ou la détection de fraude

semblait plus difficile à mettre en œuvre. Tout en restant loin des chats-bots des robots-advisors, des systèmes de recommandations ou d'analyses de sentiments, l'envie de développer un outil d'analyses de portefeuille devenait une évidence.

5.1.3 Le choix final

Après avoir appréhendé les technologies et manipulé les langages, nous devions arrêter concrètement ce que ferait notre programme. Voici le « workflow » final sélectionné.

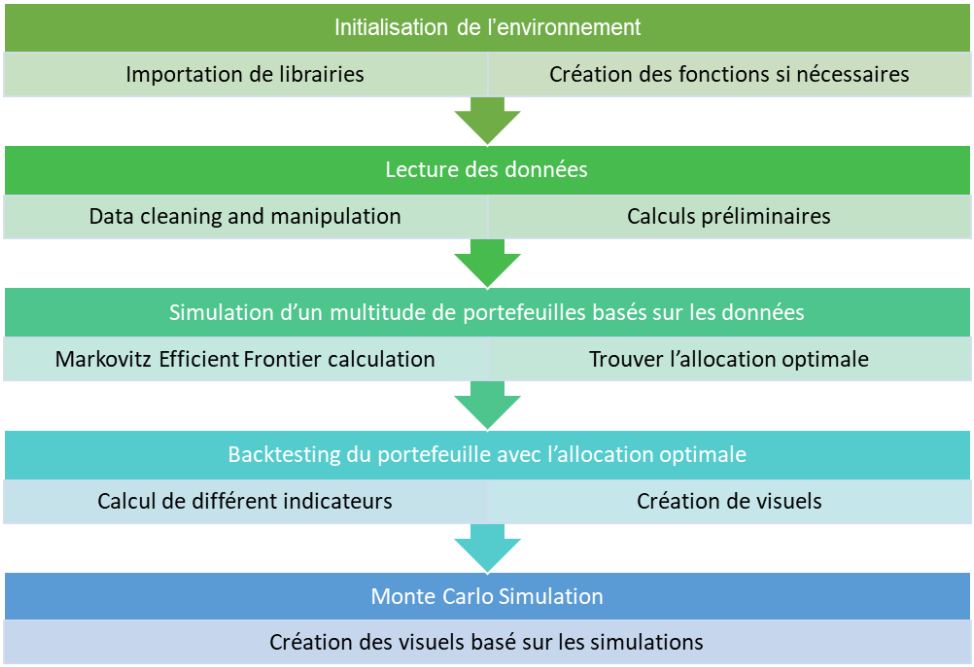


Figure 43 : Development Workflow

5.2 Le choix des données

5.2.1 Les contraintes rencontrées

Vous l'aurez compris avec la figure 43, le développement se concentrera sur l'analyse de portefeuille. Pour ce faire notre programme avait besoin de données et dans le cadre d'un projet Big Data, l'exhaustivité et la vitesse sont prépondérantes dans notre choix. L'utilisation d'un streaming de données live au travers d'un API a tout d'abord été évaluée, mais étant donné que notre analyse possède plus un caractère historique que de gestion en direct, l'idée a été écartée. Ensuite, pour simplifier le nombre d'indicateurs à calculer et offrir une certaine standardisation et reproductibilité à notre algorithme nous avons décidé de nous concentrer sur un portefeuille 100 % long actions. L'addition d'autres classes d'actifs tels que les obligations, dérivés ou même alternatives multiplierait les données de types dimension (taux d'intérêt, sous-jacent, inflation, etc.). Par mesure de précautions au niveau du temps nous avons décidé de nous concentrer sur une seule

classe d'actifs. Il avait également été décidé de reconstruire un indice tel que les S&P500 et simplement éviter l'étape de construction de portefeuille. Cependant les données historiques de pondération ou d'outstanding shares n'ont pas été trouvées en libre-service. Après révision de nos attentes, la création d'un portefeuille dit optimal au sens de Markovitz semblait donc être une bonne alternative. Il ne manquait plus qu'à trouver les données actions. Quels marchés actions ? Quels marchés géographiques ? La prochaine partie est consacrée au cœur du travail : les données.

5.2.2 Les données sélectionnées

Il est de notoriété publique que des sites destinés à la Data Science telle que Quandl ou Kaggle offre des datasets en libres services à des fins d'expérimentations. C'est donc sur ces sites que nous avons recherché notre dataset initial. Dans nos recherches nous avons découvert le dataset suivant : « AMEX, NYSE, and NASDAQ stocks histories » proposés sur Kaggle par Jiun Yen (Yen, 2019). Plus de 8000 actions chacune stockée dans des CSV séparés. La période proposée de ces données historiques est de l'IPO de chaque société au 18.04.2019. Chacune des lignes des csv est un business day, l'exhaustivité est donc respectée, la vélocité ne l'est pas, car les données sont historiques. Cependant le caractère véloce des données en terme général est respecté. En effet, les données actions, sont générés en temps normal à haute vélocité comme précisé dans notre 3e partie. Ce type de datasets pourrait donc être considéré comme suffisant pour réaliser notre développement. Pour ce qui est du volume, l'ensemble des csv pèsent seulement 2,38 Go. Les technologies utilisées peuvent gérer jusqu'à des téra voir des petta octets de données, l'hypothèse de non-concordance des technologies avec notre but est donc posée. Comment va réagir notre environnement face à des donnés pouvant être géré sans forcément devoir faire appel à des technologies Big Data ?

5.2.3 Le Data Flow

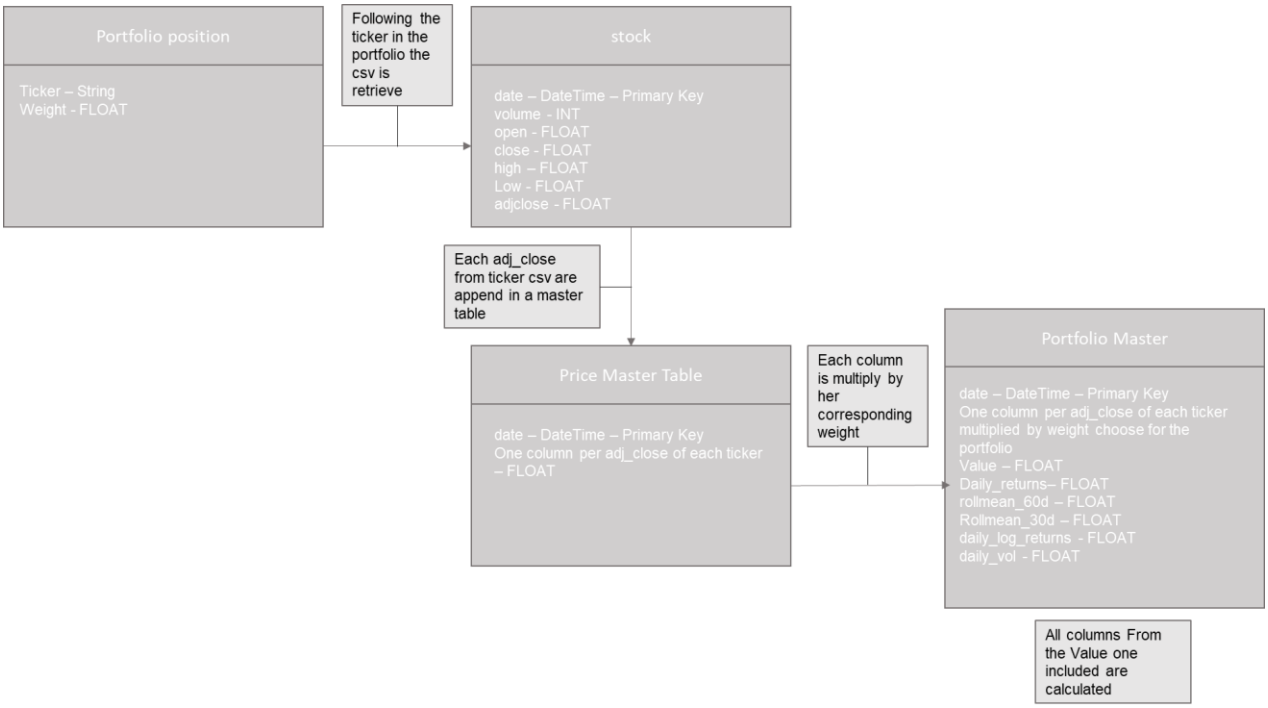


Figure 44 : Data Flow

Tout commence par un fichier CSV comportant les tickers souhaités dans le portefeuille. Celui utilisé ici a été créé à partir des données provenant du site Motley Fool (Staff, 2019). Notre portefeuille témoin contiendra donc une sélection de 20 titres faite par les analystes de ce site de nouvelles financières. Bien que les pondérations que nous allons utiliser seront celles obtenues lors du calcul du portefeuille optimal, il nous semblait important de laisser ouverte la possibilité que les poids puissent être saisis manuellement par l'utilisateur. Bien que l'algorithme ne prenne pas cette possibilité en compte, de légères modifications le permettraient aisément.

Grâce à ce fichier CSV, il nous est désormais possible d'aller chercher l'ensemble des « adj close » pour chaque ticker contenu dans le CSV qui leur est propre. Toutes ces colonnes de prix sont fusionnées dans une seule et même dataframe : « price_master ». Puis chacune des colonnes est multipliée par la pondération du ticker dans le portefeuille pour ainsi obtenir l'évolution de la valeur de chaque position dans le portefeuille. La dataframe « portfolio_master » est ensuite notre base pour l'ensemble des calculs d'indicateurs et la création des visuels.

Nous avons désormais les données et leurs flows dans notre programme. Il ne nous manque plus que l'environnement mis en place. Même si les technologies utilisées ont été présentées dans notre seconde partie nous allons brièvement faire un rappel de ce qui a été mis en place.

5.3 L'environnement de développement :

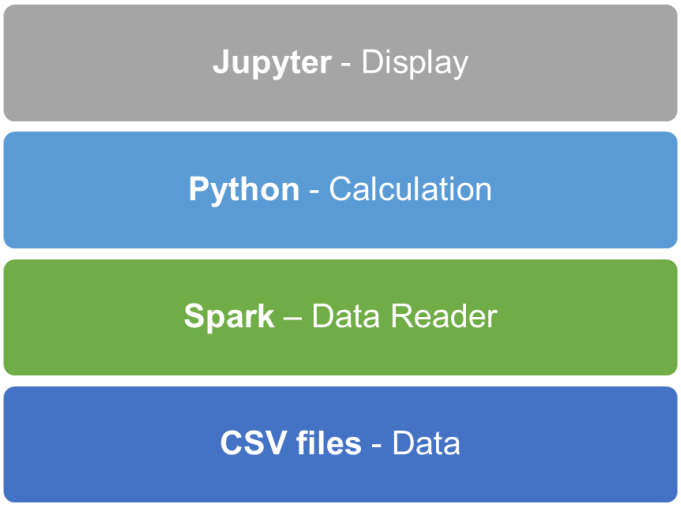


Figure 45 : Development Stack

Le notebook Jupyter, installé nativement avec le gestionnaire de package python Anaconda, nous a permis d'écrire et tester mon code directement via un navigateur internet. L'ensemble du programme est codé en python avec l'usage des librairies classiques telles que pandas ou matplotlib.

Après avoir configuré Apache Spark, avec la simulation d'une instance HDFS et la création d'un JVM avec Java, nous avons utilisé le package findspark de python qui permet à jupyter de retrouver les fichiers nécessaires à lancer une instance de spark depuis le notebook. Puis la librairie pyspark et sa collection de fonctions dédiée à la manipulation de données avec Spark nous a permis de nous rapprocher d'un développement Big Data et de ses problématiques. Enfin comme cité dans la partie précédente, les données sont stockées sous format csv dans un dossier dédié.

La difficulté rencontrée fut de choisir quand utiliser Apache Spark plutôt que les librairies de Python. La librairie pandas sur le volume de données choisi est amplement suffisante donc les problèmes qui normalement surviennent et imposent Spark comme la solution ne sont pas apparus pendant ce développement. Nous étudierons cela dans notre prochaine partie qui sera consacrée à la présentation des résultats informatique.

5.4 Les résultats :

5.4.1 Les résultats techniques

Le premier constat que nous pouvons faire techniquement est le suivant : la mise en parallèle des opérations effectuées par Spark sur ses dataframes n'est pas une solution qui convient à tous les problèmes. Les uses cases avant le développement d'une telle solution se doivent d'être étudié de près. L'hypothèse faite au début du développement

était que la puissance d'Apache Spark surclasserait des librairies telles que Pandas pour la manipulation de données. Nous allons étudier les résultats observés afin de valider ou non cette affirmation.

```
#=================================SPARK SESSION BEGIN============================================

#Create a SparkSession (the config bit is only for Windows!)
spark = SparkSession.builder.config("spark.sql.warehouse.dir", "file:///C:/temp").appName("portfolio_simu").getOrCreate()

#=================================SPARK - IMPORT PORTFOLIO TICKER DIM TABLES=====================================
#Weight could be define in this table but as we are looking for the optimal allocation here after we drop the column

path = "C:/Users/huniv/TB/datasets/test_port.csv"
#Get the raw data in DataFrame from the csv
portfolio_position = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path)
#rename columns
portfolio_position = portfolio_position.withColumnRenamed('Symbol', 'ticker')
#cast the ticker values into string
portfolio_position = portfolio_position.withColumn('ticker', portfolio_position["ticker"].cast("string"))
#drop the weight column (not required here)
portfolio_position = portfolio_position.drop('Weight')
#convert the dim table into pandas df to be able to iterate each ticker in a loop.
pd_portfolio_position = portfolio_position.select("*").toPandas()
```

Figure 46 : Portfolio table retrieving code

La première expérience effectuée était dédiée à la récupération des données. La problématique était de récupérer les fichiers csv de chaque ticker nécessaire à la construction de mon portefeuille. Ainsi nous avons créé une table ne contenant que les tickers du portefeuille et leurs pondérations. Pour rappel, dans notre programme la colonne des pondérations est supprimée, car nous allons rechercher la pondération optimale avec le modèle de frontière efficiente de Markovitz. Le stock picking ne se fait pas en quelques heures et nous avons préféré privilégier le but du programme plutôt que d'ajouter un élément à notre travail. C'est pourquoi la liste des 20 actions de Motley Fool est utilisé, mais rien n'empêche de modifier notre protefeuille comme bon nous semble. Ainsi le fichier « test_port.csv » est notre table de référence.

Tableau 11 : Dim Table for portfolio position

Symbol	**Weight**
String	Float

Une fois cette table crée, nous devions donc récupérer l'évolution des prix de chaque ticker dans une seule et même dataframe notre : « price_master ». Pour ce faire nous n'avons pas trouvé d'autre solution que d'itérer les tickers dans une boucle afin de modifier le chemin des fichiers récupérés à chaque itération. Pour itérer à travers notre table de référence de portfolio, cette dernière devait être transformé en dataframe pandas. Pourquoi ? Spark ne permet pas l'itération à travers les lignes de ses dataframes. Cela va à l'encontre de son existence même. Spark est capable d'appliquer une opération à toutes les lignes d'une dataframe de manière concurrente. Empêcher Spark de le faire en

appliquant une itération serait donc contre-productif. Voilà un premier exemple des limites de Spark par rapport à notre use case.

```
#==================================SPARK - CREATE THE ADJ CLOSE FACTS TABLES================================
#create a master spark dataframe which will receive all adj_close of all ticker. One col = One Price Evolution of a ticker
schema = StructType([])
price_master = spark.createDataFrame([], schema)

path = "C:/Users/huniv/TB/datasets/full_history/"
columns_to_drop = ['date', 'volume', 'open', 'close', 'high', 'low']

#for each ticker in the dim table, get the correct csv files in a spark dataframe and transform it
for idx, row in pd_portfolio_position.iterrows():

    #get the current ticker
    curr_ticker = row['ticker']

    ##Get the raw data in DataFrame from the csv
    stock = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path + curr_ticker + ".csv")
    #Ex: adj_close => adj_close_MSFT . To differentiate each column in the master dataframe
    stock = stock.withColumnRenamed('adjclose', 'adjclose_' + curr_ticker)
    #Cast the date
    stock = stock.withColumn('myDate', F.to_date(stock.date, 'yyyy-MM-dd'))
    #Sort
    stock = stock.orderBy('myDate')
    #Drop unrequired column
    stock = stock.drop(*columns_to_drop)

    #if it's the first tiker duplicate in the master, otherwise make a left join on the date
    #NULL Values stored if the next ticker doesn't exist the day pf the row
    if idx == 0:
        price_master = stock
    else:
        price_master = price_master.join(stock, on="myDate", how="left")

#Select the begining of your investment. Preferably a date where all ticker was public yet
price_master = price_master.filter(price_master["myDate"] >= F.lit('2018-01-01'))
#convert the master spark dataframe into a pandas dataframe
pd_price_master = price_master.select("*").toPandas()

#Cast all adj_close columns into numeric values
cols=[i for i in pd_price_master.columns if i not in ["myDate"]]

for col in cols:
    pd_price_master[col] = pd.to_numeric(pd_price_master[col])

#Cast the date
pd_price_master["myDate"]= pd.to_datetime(pd_price_master["myDate"], infer_datetime_format=True)
#Set the date as the row index (Primary Key)
pd_price_master = pd_price_master.set_index('myDate')
```

Figure 47 : Spark – Retrieving several csv and join them in a master table

```
#==================================PANDAS - CREATE THE ADJ CLOSE FACTS TABLES================================
columns_to_drop = ['volume', 'open', 'close', 'high', 'low']

#for each ticker in the dim table, get the correct csv files in a spark dataframe and transform it
for idx, row in pd_portfolio_position.iterrows():

    #get the current ticker
    curr_ticker = row['ticker']

    data = pd.read_csv(path + curr_ticker +'.csv')
    data = data.rename(columns={'adjclose':'adjclose_' + curr_ticker})
    data.date = pd.to_datetime(data['date'])
    data = data.drop(columns_to_drop, axis=1)
    data = data.set_index("date")

    #if it's the first tiker duplicate in the master, otherwise make a left join on the date
    #NULL Values stored if the next ticker doesn't exist the day pf the row
    if idx == 0:
        master = data
    else:
        master = pd.merge(master,
                  data,
                  on='date')
```

Figure 48 - Pandas - retrieving several csv and join them in a master table

La suite du code fut l'occasion de comparer nos deux méthodes au niveau de leurs temps d'exécutions. Notre postulat de départ devrait placer Spark en tête. Or lorsque l'on compare le temps de lecture des deux méthodes voilà ce que nous obtenons :

- Lecture avec Spark : 50,1 s
- Lecture avec Pandas : 2,67 s

Le constat est amer pour Spark, Pandas est près de 19 fois plus rapide sur notre machine. Le problème provient peut-être de cette itération qui force spark à créer une multitude de dataframe à la suite. À des fins uniquement d'expérimentation, nous avons testé une autre méthode afin de décortiquer cette différence de timing très importante. Cette expérience n'a pas de sens pour notre programme, car les données modelées ainsi ne seront pas exploitable. Cependant il est tout même intéressant de regarder le résultat. Pour effectuer cette expérimentation nous avons créé un nouveau notebook : « Batch CSV Reading comparison ». Dans ce notebook, nous concaténons l'ensemble des fichiers csv de tous les tickers disponible dans le dossier soit 2,38 Go.

```
#=================================SPARK SESSION BEGIN=================================================
path = "C:/Users/huniv/TB/datasets/full_history/"

#Create a SparkSession (the config bit is only for Windows!)
spark = SparkSession.builder.config("spark.sql.warehouse.dir", "file:///C:/temp").appName("batch_simu").getOrCreate()

#=================================SPARK - IMPORT PORTFOLIO TICKER DIM TABLES=================================
#Weight could be define in this table but as we are looking for the optimal allocation here after we drop the column
##Get the raw data in DataFrame from the csv
stock = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path + "*.csv")
```

Figure 49 : Spark - Batch Reading

```
path = "C:/Users/huniv/TB/datasets/full_history/"

files = os.listdir(path)

# glob.glob('data*.csv') - returns List[str]
# pd.read_csv(f) - returns pd.DataFrame()
# for f in glob.glob() - returns a List[DataFrames]
# pd.concat() - returns one pd.DataFrame()
df = pd.concat([pd.read_csv(path + f) for f in os.listdir(path)], ignore_index = True)
```

Figure 50 : Pandas - Batch Reading

La victoire ici revient à Spark. Les temps de lecture et de concaténation de l'ensemble des csv sont plus de 2 fois moins élevés pour Spark que pour pandas :

- Batch Reading avec Spark : 1 min 51 s
- Batch Reading avec Pandas : 3 min 52 s

Dans la syntaxe du code, nous remarquons que Spark n'a pas besoin de loop. Il comprend ce qu'il doit faire de tous ces fichiers car Spark infère le schéma lorsqu'il lit les fichiers. Pandas doit être préparé à passer à travers l'ensemble des fichiers via la boucle for initiée. Visiblement l'itération et la pose de condition stricte ne conviennent pas à Apache Spark. Comme précisé, les données conditionnées ainsi par Spark ne sont pas exploitables pour notre travail. Pourquoi ? Il nous manquerait un champ, celui indiquant à quel ticker appartient la ligne, car actuellement tous les tickers sont mélangés et non reconnaissables. Le data model utilisé ne convient donc pas au potentiel d'Apache Spark. Cependant nous avons souhaité poursuivre nos tests en poussant l'expérience jusqu'à l'agrégation de ces 26 168 366 lignes. Ce chiffre a été obtenu par les simples fonctions count() disponibles dans Spark et Pandas. Cette fois-ci Pandas reprend le dessus :

- Comptage avec Spark : 50 s
- Comptage avec Pandas : 14,1 s

Pour ce qui est de l'agrégation, nous avons simplement appliqué aux deux méthodes les fonctions sum() à toutes les colonnes puis regroupé toutes les données par années puis par mois.

```
stock.createOrReplaceTempView("stock")
stock_grouped = spark.sql("SELECT YEAR(date) as year, \
                                  MONTH(date) as month, \
                                  SUM(volume) as volume, \
                                  SUM(open) as open, \
                                  SUM(close) as close, \
                                  SUM(high) as high, \
                                  SUM(low) as low, \
                                  SUM(adjclose) as adjclose \
                           FROM stock \
                           GROUP BY year, month")
```

Figure 51 : Aggregate with Spark

```
 #Cast the date
df["date"]= pd.to_datetime(df["date"], infer_datetime_format=True)
#Set the date as the row index (Primary Key)
df = df.set_index('date')
 #Cast the date
df["volume"] = pd.to_numeric(df["volume"])
df["year"]=df.index.year
df["month"]=df.index.month
test = df.groupby(['year', 'month']).sum()
```

Figure 52 : Aggregate with Pandas

Ici, Spark possède une importante avance surclassant de très loin les performances de Pandas.

- Agrégation avec Spark : 758 ms
- Agrégation avec Pandas : 27,6 s

Ainsi nous pouvons en conclure que Spark reprend largement l'avantage sur un datasets plus que conséquent et effectue les mêmes opérations bien plus rapidement que Pandas. Le datasets de 26 168 366 lignes est géré de la lecture à l'agrégation en 2 minutes 42 secondes alors que la librairie Pandas effectue les mêmes types d'opérations en 4 minutes 34. Ce que nous pouvons également observer, c'est la syntaxe du code. Spark transmet simplement une requête SQL alors que Pandas doit s'assurer de « caster » les types de données, définir ses index pour ensuite commencer son opération. La facilité éprouvée lorsque nous nous adressons aux données avec Spark est étonnante de simplicité. Bien que les timings ne soient pas si espacés, il est envisageable que sur des téraoctets Pandas ne puissent tout simplement pas avoir assez de mémoires pour effectuer l'ensemble des opérations ; Spark, lui, est conçu pour. L'inverse reste également vrai. Les petits datasets ne conviennent pas à Spark. Pour exemple, dans notre développement, nous calculons la matrice de corrélation des tickers afin de visualiser la diversification du portefeuille. Ici le datasets est nettement plus petit que 26 millions de lignes (577 lignes).

- Calcul de corrélation avec Spark 3 min 44 s
- Calcul de corrélation avec Pandas : 4,13 s

La « Scalability » de Spark lui permettant de travailler sur un cluster entier n'est absolument pas reproductible par Pandas et le fossé se créer proportionnellement au volume de données. Et ce fossé se creuse également en faveur de Pandas lorsque les données traitées sont peu conséquentes. Notre analyse ici ne se porte que sur les temps d'exécution, déjà d'excellents indicateurs de performance. Une piste de recherche supplémentaire serait d'aller étudier la consommation de mémoire des scripts à l'aide des Listeners de Spark et d'autres librairies Python.

Tâches	Spark	Pandas	Données
Lecture avec itération	50,1 s	2,67 s	>8000 csv
Batch Reading	1 min 51 s	3 min 52 s	>8000 csv
Comptage	50 s	14,1 s	>26 Mio de lignes
Agrégation	758 ms	27,6 s	>26 Mio de lignes
Corrélation	3 min 44 s	4,13 s	577

Figure 53 : Récapitulatifs des temps d'exécutions.

En conclusion après l'analyse de ces résultats, il est nécessaire de rappeler qu'Apache Spark est un Framework construit pour une approche distribuée où les calculs s'effectuent sur des centaines de nœuds, assurant la conformité des données, mais acceptant également une certaine fault tolerance. Ces différentes propriétés ont un coût que Pandas ignore, car cette librairie est destinée à être utilisée sur une seule machine et ses cœurs disponibles. Spark travaille parfois sur des centaines de disques et mémoires différentes et qui plus est à travers un réseau (même sur une machine locale). Pandas se défait de ces contraintes et il est normal d'observer une vitesse plus élevée lors de certains de nos tests. Le concept de mise en parallèle des calculs et de processus distribué lié à Spark implique une toute nouvelle façon de développer imposant certaines règles et manières parfois trop contraignantes et bien éloignées des concepts basiques de l'algorithmique. La « learning curve » peut s'avérer plus importante et il est donc nécessaire de définir précisément ces besoins avant d'employer le mauvais outil. Enfin le dernier mot sur nos tests techniques concernera la mise en place d'un environnement local de Spark. Ce mode d'utilisation trouve son intérêt dans les tests et la préparation non dans la performance. Il n'est donc pas surprenant de voir des algorithmes de librairies dédiés à

être utilisés en local battre Spark sur différents sujets. Sur notre environnement Spark disposait de 4 cœurs, 384,1 Mb de mémoire cache et une JVM Java Virtual Machine pour déployer son efficacité. Dans un cluster de serveur, c'est des centaines de machines avec chacun 8 à 16 cœurs, des gigas de mémoires disponibles et leurs propres JVM pour effectuer toutes les opérations. Il est donc important de prendre du recul et savoir reconnaitre que les faiblesses trouvées dans le Framework sont également liées à notre environnement de test.

Après cet aperçu de notre travail d'un point de vue purement technique, nous plongeons dans l'ultime partie de ce travail, les résultats métiers, ceux du point de vue d'un gestionnaire de portefeuille.

5.4.2 Le Résultat métiers

Comme nous l'avons précédemment précisé, les tickers choisis n'ont pas été le fruit d'une recherche approfondie pour la construction d'un portefeuille. La liste utilisée provient du site Motley Fool relatant les 20 tickers à obtenir pour l'année 2019. Ainsi un fichier csv a été créé avec la liste suivantes :

Tableau 12 : Portfolio's Tickers

Company	Ticker	Secteurs
The Vanguard Total Stock Market ETF	NYSE : VTI	ETF
The Vanguard Total International Stock ETF	NASDAQ:VXUS	ETF
Amazon.com	NASDAQ:AMZ	Consumer Cyclical
Alphabet	NASDAQ:GOOG NASDAQ:GOOGL	Technology
Facebook	NASDAQ:FB	Technology
Intuitive Surgical	NASDAQ:ISRG	Healthcare
Axon Enterprises	NASDAQ:AAXN	Basic Materials
AT&T	NYSE:T	Communications
Verizon Communications	NYSE:VZ	Communications
Ford Motor Company	NYSE:F	Consumer Cyclical
General Motors Company	NYSE:GM	Consumer Cyclical
ONEOK	NYSE:OKE	Energy
TerraForm Power	NASDAQ:TERP	Utilities
Brookfield Infrastructure Partners L.P.	NYSE:BIP	Utilities
CareTrust REIT	NASDAQ:CTRE	Real Estate

iRobot	NASDAQ:IRBT	Technology
Lululemon Athletica	NASDAQ:LULU	Consumer Cyclical
Wayfair	NYSE:W	Consumer Cyclical
Netflix	NASDAQ:NFLX	Consumer Cyclical
Constellation Brands	NYSE:STZ	Technology

Ce choix est totalement arbitraire et peut être revu par la suite par l'utilisateur une fois qu'il aura analysé les entreprises et fais ces choix. Il lui suffira d'ajouter ou supprimer une ligne dans le csv de base. Ne pas avoir un représentant des Financials Services par exemple peut s'avérer surprenant. Par contre l'addition des deux ETF est un bon moyen de diversifier le portefeuille facilement. Cependant l'est-il assez ?

5.4.2.1 Matrice de corrélation

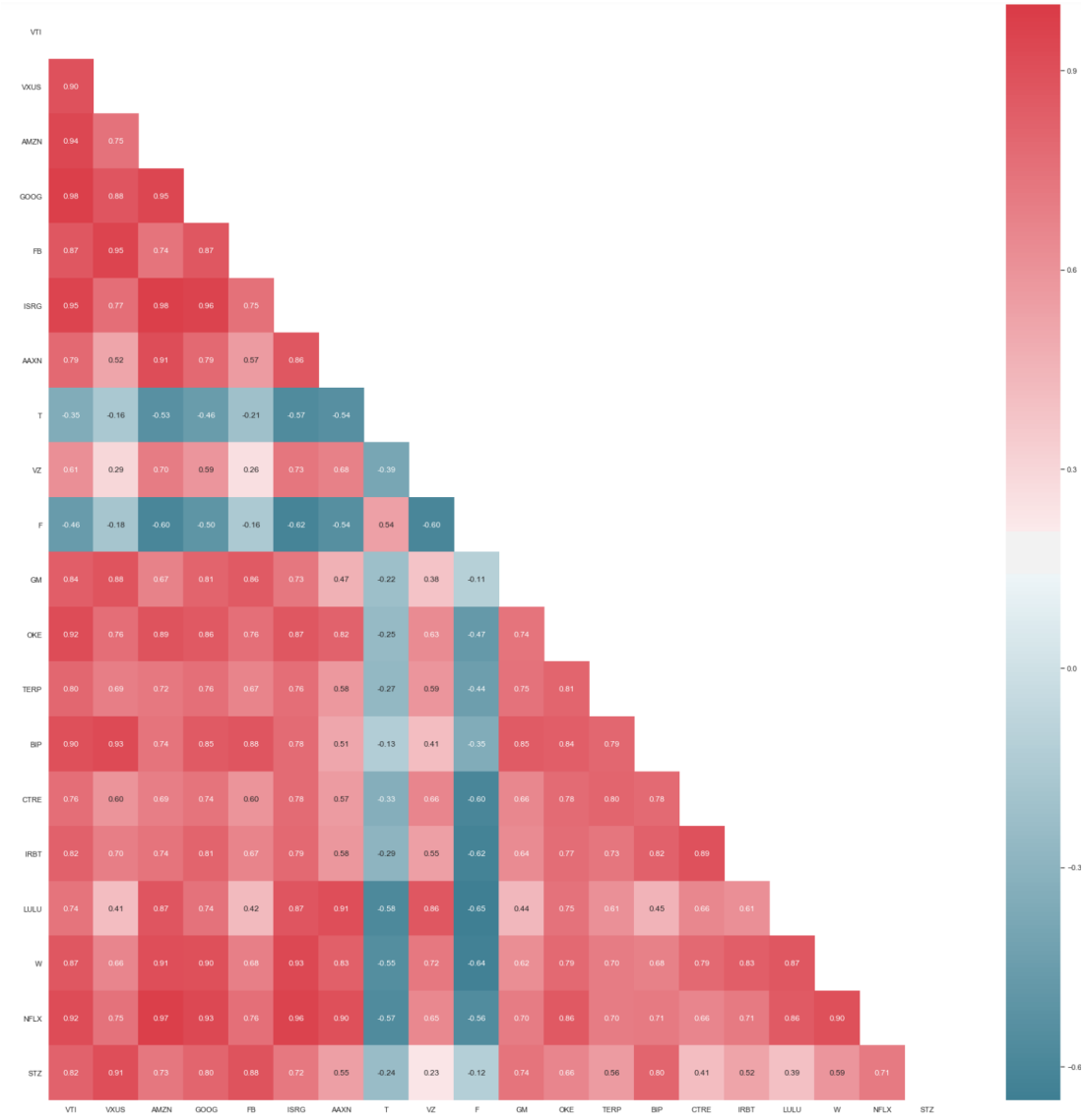


Figure 54 : Matrice de corrélation (Spark)

En un coup d'œil sur la matrice, nous voyons immédiatement qu'AT&T & Ford sont les compagnies, les plus décorrélés des autres avec des ratios négatifs, mais pas extrêmes. Toutes les actions sont fortement corrélées avec les deux ETF, ce qui n'est pas une surprise, car ils reflètent la santé du marché actions en général. Ce graphique nous permet d'apprécier donc toutes ces relations et d'affirmer ainsi que ce portefeuille dans l'état actuel n'est largement pas assez diversifié. Une bonne matrice de corrélation devrait contenir plus de valeurs négatives signe de mouvement de compensation et/ou des valeurs égal à 0 signe d'une décorrélation maximale entre les titres. Avec de telles valeurs, le portefeuille pourrait être considéré comme diversifié.

5.4.2.2 Le modèle de markovitz

Tout d'abord, faisons un petit rappel sur le modèle appliqué. La frontière d'efficience de Markovitz délimite l'ensemble des rendements possibles pour des portefeuilles avec une volatilité donnée. Les rendements sont totalement dépendants des actions choisies et de leurs performances sur les marchés. La volatilité du portefeuille est déterminée avec la matrice de covariance qui lie l'ensemble des titres ensemble. Plus la covariance est basse entre les titres plus le portfolio est sûr. La mesure de rendement utilisé dans cet algorithme sont les log returns du portfolio :

$$Position\ Daily\ Log\ Return = \ln(\frac{S_n}{S_{n-1}})$$

Ce calcul est appliqué à chaque position, ce qui nous donne les rendements journaliers de chaque titre. Puis la moyenne est calculée pour chaque titre, pondéré par le poids dans le portefeuille et enfin annualisé sur 252 jours. Ceci nous donne donc le rendement annuel du portefeuille.

$$Portfolio\ Annual\ Log\ Return = \sum_{i=0}^{n} Daily\ Log\ Return_i * w_i * 252$$

Pour ce qui est de la volatilité, la formule de base pour calculée celle d'un portefeuille possède comme paramètre les variances et covariance des titres pondérés avec les poids correspondants.

$$\sigma_{portfolio} = \sqrt{w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1^2w_2^2cov_{1,2}}$$

L'intérieur de la racine peut être calculé avec à l'aide d'un calcul matriciel et c'est cette métdhode que nous avons répliquées algorithmiquement.

$$\sigma^2_{portfolio} = [w_1\ w_2]\begin{bmatrix}\sigma_1^2 & cov_{1,2}\\ cov_{2,1} & \sigma_2^2\end{bmatrix}\begin{bmatrix}w_1\\ w_2\end{bmatrix}$$

Enfin, pour terminer ce rappel théorique, concentrons-nous sur le Sharpe Ratio et son objectif.

$$Sharpe\ ratio = \frac{Portfolio\ return - Risk\ Free\ Rate}{Portfolio\ Volatility}$$

Le sharpe ratio est un calcul de rentabilité obtenu par unité de risque prise. La réduction avec le risk free rate nous permet de savoir la part des rendements qui est propre au portefeuille. Dans notre algorithme, le risk free rate est = 0 dû à l'environnement de taux bas dans lequel nous évoluons ces derniers temps. Cependant la variable à quand même été créée et peut être configurer facilement. Maintenant que les concepts de base ont été définis, plongeons-nous dans nos résultats.

```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 24.66
Annualised Volatility: 14.43
Annualised Sharpe Ratio: 1.71


            VTI  VXUS  AMZN  GOOG    FB  ISRG  AAXN     T    VZ     F    GM  \
allocation  6.0  3.44  4.56  2.48  1.49  9.21  6.39  0.16  3.17  0.08  6.45

             OKE  TERP    BIP   CTRE  IRBT   LULU     W  NFLX   STZ
allocation  2.62  0.05  15.48  10.95   4.0  11.05  0.05  3.02  9.35
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 17.78
Annualised Volatility: 13.11
Annualised Sharpe Ratio: 1.36


             VTI  VXUS  AMZN  GOOG    FB  ISRG  AAXN      T    VZ     F    GM  \
allocation  4.79  9.81  0.36  6.57  1.35   0.1  7.33  10.34  9.63  4.09  4.41

             OKE  TERP   BIP   CTRE  IRBT  LULU    W  NFLX   STZ
allocation  4.99  2.75  6.21  10.32  3.75  2.02  0.6  1.45  9.14
--------------------------------------------------------------------------------
```

Figure 55 : Optimal Portfolio Allocation

Dans cette partie, nous sommes parties du postulat que l'utilisateur savait quelles actions il souhaitait, mais pas l'allocation qui lui permettrait de profiter pleinement de sa stratégie. À travers une simulation de 50 000 portefeuilles différents générés de manière aléatoire, notre programme calcul la volatilité, le sharpe ratio et le rendement pour chacun des portefeuilles. Grâce à cette simulation les deux portefeuilles avec les allocations optimales au sens de markovitz sont trouvés : Celui avec le sharpe ratio le plus élevé et celui avec la volatilité la plus basse (voir figure 55). Dans la suite de notre développement nous avons fait le choix d'utiliser l'allocation proposé pour le Sharpe ratio le plus élevé afin de maximiser nos rendements. Peut-on mieux faire ? Peut -on trouver mieux qu'un Sharpe Ratio de 1,71 ? Le deuxième algorithme dédié à cette partie utilise une fonction

d'optimisation de la librairie Scipy : minimize. Son utilisation s'effectue à la manière du solver Excel avec la pose de contraintes, de bornes et bien sûr de la fonction, ici, à minimiser. Voici les résultats que propose cet algorithme :

```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 29.05
Annualised Volatility: 14.63
Annualised Sharpe Ratio: 1.98


            VTI  VXUS  AMZN  GOOG   FB   ISRG  AAXN    T    VZ    F   GM  \
allocation  0.0   0.0  5.39   0.0  0.0  15.02  7.13  0.0  9.02  0.0  0.0

              OKE  TERP    BIP   CTRE  IRBT  LULU    W  NFLX  STZ
allocation  10.23   0.0  15.33  25.38  0.24  9.08  0.0  3.18  0.0
--------------------------------------------------------------------------------
```

Figure 56 : Allocation proposal via Scipy

Le premier constat se fait directement sur le Sharpe Ratio de 0,3 point de plus que celui calculé via notre simulation. L'optimisation via Scipy obtient donc des résultats plus précis. Le deuxième constat peut être fait sur les pondérations. Sur ce screenshot, beaucoup sont à zéros et la somme de l'ensemble est égale à environ 91,20 %. La fonction minimize de Scipy est capable de faire descendre les pondérations à 10^{e}-16, car elle n'a pas été bornée. Une solution pourrait être d'ajouter une contrainte sur une allocation minimale. Voilà les résultats obtenus avec une obligation d'allouer 1 % de notre budget à chacun des titres.

```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 27.92
Annualised Volatility: 14.63
Annualised Sharpe Ratio: 1.91


            VTI  VXUS  AMZN  GOOG   FB   ISRG  AAXN    T   VZ    F   GM   OKE  \
allocation  1.0   1.0  3.73   1.0  1.0  13.92  6.73  1.0  5.4  1.0  1.0  9.72

            TERP    BIP   CTRE  IRBT  LULU    W  NFLX  STZ
allocation   1.0  13.65  25.05   1.0  8.34  1.0  2.47  1.0
--------------------------------------------------------------------------------
```

Figure 57 : Allocation proposal via Scipy with strict constraints

Le portfolio garde la même volatilité perd du rendement et il en résulte un Sharpe Ratio légèrement plus faible. Cependant l'allocation proposée ici semble plus réalisable qu'avec des pondérations à 10^{e}-16.

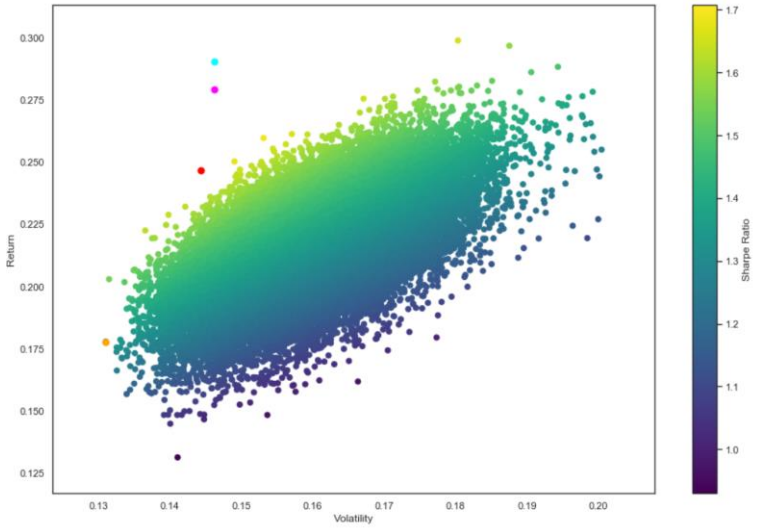


Figure 58 : Simulation VS Optimisation

Sur la figure 58, nous avons tracé l'ensemble des 50 000 portfolios simulés : en rouge, le portfolio avec le maximum Sharpe Ratio et en Orange celui avec la volatilité minimum. Puis nous avons ajouté les deux portefeuilles calculés avec l'optimisation : en cyan celui sans contrainte d'allocation minimum et en magenta celui avec les contraintes. Visiblement les portefeuilles simulés sont loin des résultats de l'algorithme d'optimisation. Selon le modèle de Markovitz, cela ne signifierait qu'aucun des portfolios simulés ne serait sur la frontière d'efficience, car la création du portfolio cyan est encore possible et il semble bien trop éloigné de la masse des simulations. Vérifions cela en traçant la frontière.

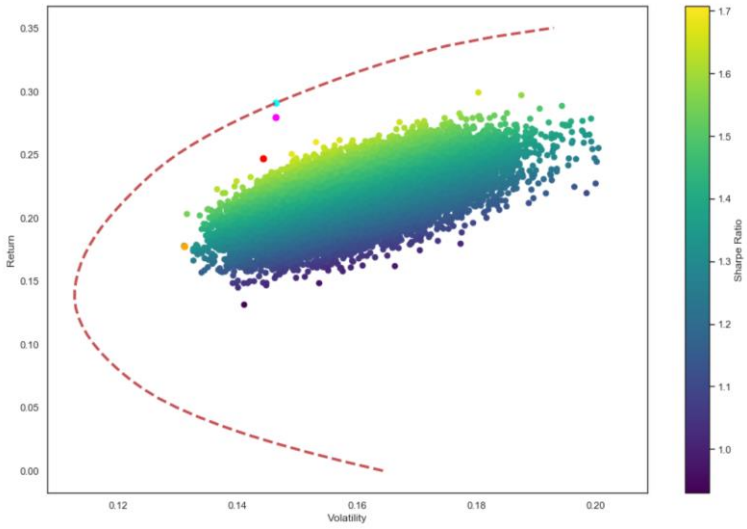


Figure 59 : Simulation VS Optimization with Efficient frontier

Le résultat est sans appel, les allocations calculées avec la fonction d'optimisation de Scipy sont nettement plus intéressantes que celles calculées via simulations. Nous

pouvons voir que le portfolio cyan se trouve sur la frontière et le magenta juste en dessous à cause de la contrainte d'allocation minimale. Les portfolios orange et rouge sont donc les portefeuilles optimaux, oui, mais sur les 50 000 simulations et non sur l'ensemble des possibles. Pour la suite de nos calculs, nous avons décidé de garder les pondérations proposées par le portfolio magenta.

5.4.2.3 Les indicateurs de performance

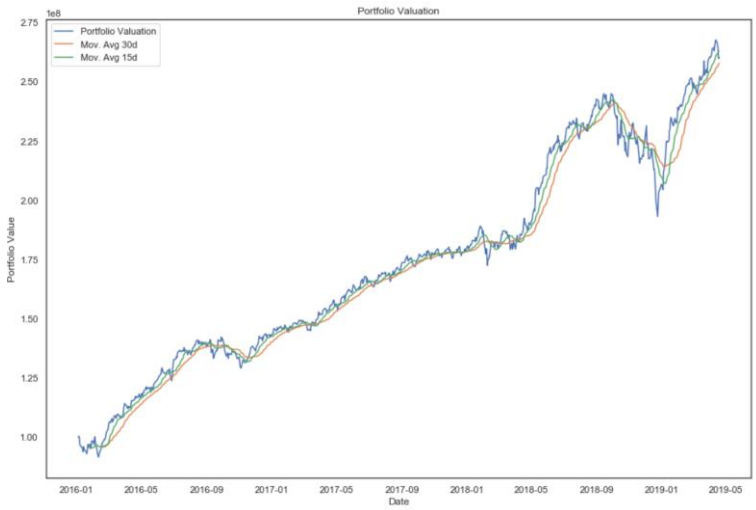


Figure 60 : Portfolio Valuation & Moving Average

Dans la suite de notre développement, nous avons décidé de calculer un panel d'indicateurs de performances et d'analyses de portefeuille. Dans un premier temps la figure 60 nous montre l'évolution de la valeur du portefeuille commençant à $ 1 Mio le 01.01.2016 et terminant à près de 2,6 Mio en avril 2019 avec un CAGR de 22,4 % et une volatilité annuel de 15,36 %. Ces valeurs diffèrent légèrement de celle obtenue avec la fonction d'optimisation, car les méthodes de calcul sont légèrement différentes. Nous pouvons observer quelques chutes durant la période. Les deux majeurs se trouvent fin 2016 et fin de2018. Là où moving average sont intéressantes, c'est que peu de temps après le début de la chute, les deux moyennes se croisent. La moyenne à 15 jours devient inférieure à la moyenne de 30 jours et inversement lorsque le portefeuille observe une hausse. Cette méthode d'analyse peut facilement être implémentée dans un algorithme de trading afin de déclencher les ordres correspondants à ses tendances et à la stratégie du gestionnaire.

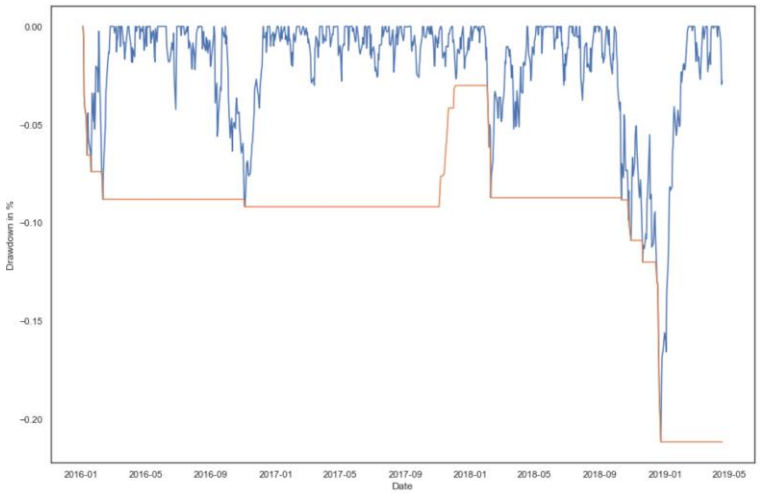


Figure 61 : Max Drawdown

Le second indicateur est le Max Drawdown qui correspond à la plus forte baisse sur une année de 252 jours. Comme cité précédemment les baisses de 2016 et 2018 sont les plus importantes. Sur toute la période, le maximum drawdown a été observé le 28.12.2019 et se montait à -21,18 %. Pour rappel ce dernier trimestre 2018, le S&P500 plongeait de près de 14 % et le NASDAQ de 18 % (Imbert, 2018). L'effet est donc normalement répercuté sur nos performances la même période.

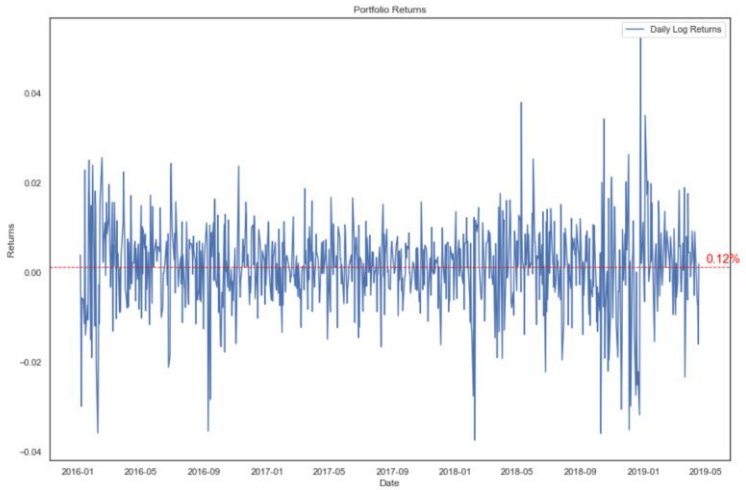


Figure 62 : Daily Log Returns

Les troisièmes indicateurs calculés sont les log returns. Le 24.12.2018, le portefeuille enregistre sa plus grande baisse, mais c'est 2 jours après qu'il remonte au plus haut avec un rendement calculé à 5,23 %. En revanche la plus grosse perte a été enregistré non pas en décembre, mais le 08.02.2019 avec des rendements calculés à-3.74%. La moyenne journalière est de 0,12 % et un écart type de 0,97 %.

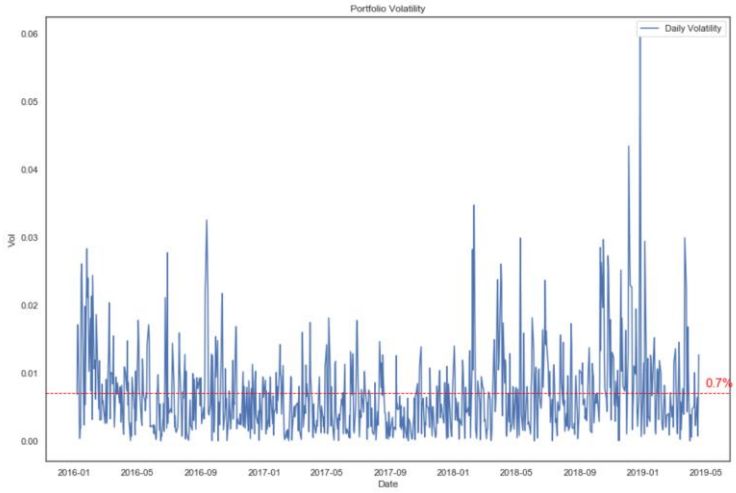


Figure 63 : Daily Volatility

Sans surprise, le jour le plus volatile fut celui succédant à la plus forte baisse du marché. En effet le 26.12.2018, nous observons un pic à près de 6 % de volatilité sur le rendement du portefeuille. Avec une moyenne de 0,7 %, et quelques pics au-dessus des 3 % lors des fortes baisses de 2016 et 2018, le portefeuille semble réagir à certains stimuli. Lesquels sont-ils ? À quel point notre portefeuille est-il influencé par le marché ?

5.4.2.4 Capital Asset Pricing Model (CAPM)

Le CAPM est un modèle utilisé pour décrire la relation étroite entre le risque et la rentabilité d'un actif financier. Un placement se doit d'offrir une rémunération en fonction de la « time value of money », mais également du niveau risque pris lors de l'investissement. Dans l'équation du CAPM, l'expected return de l'investissement est fonction du risk free rate, du Beta de l'actif et de l'excess market returns.

$$ER_i = R_f + \beta_i(ER_m - R_f)$$

Qu'est-ce que le Bêta ? C'est la mesure de sensibilité d'un actif par rapport à son benchmark et répond à la question si le marché monte comment réagis l'actif ? Il évolue dans le même sens, dans le sens inverse ? À quelle intensité ?

$$\beta_i = \frac{\sigma_{er_i er_m}}{\sigma^2_{er_m}}$$

Dans notre cas, nous allons calculer le Bêta de notre portefeuille afin de comprendre dans quelle mesure le marché influence notre investissement. Le benchmark choisi est le S&P500 au travers d'un ETF le SPY.

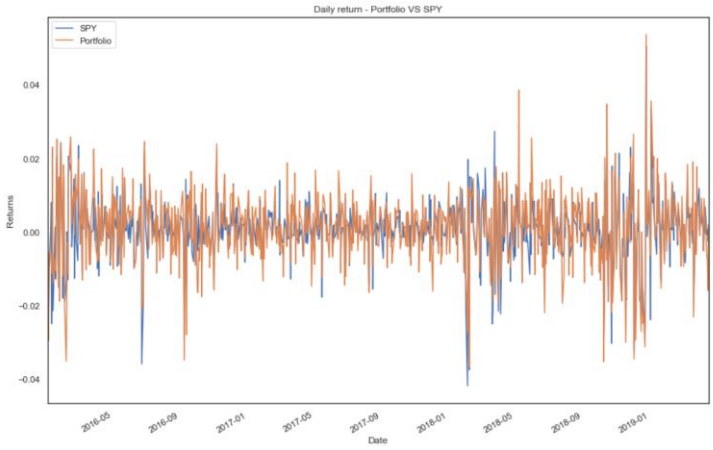


Figure 64 : SPY Returns VS Portfolio Returns

Au préalable, nous dessinons les rendements de notre portfolio et de l'ETF afin déjà d'obtenir une idée générale de ce que pourra être le Bêta. À première vue les deux rendements semblent de même ampleur et de même volatilité. Cela présage un Bêta proche de 1. Lorsque le marché monte de 100 bps, notre portfolio prendra également 100 bps.

Grâce à la régression linéaire de la librairie Scipy, nous avons pu calculer non seulement le Bêta, mais également l'Alpha du portefeuille par rapport au SPY. La figure 65 montre un nuage de points des rendements du SPY en fonction des rendements de notre portfolio. Nous voyons clairement une relation positive entre ces deux index.

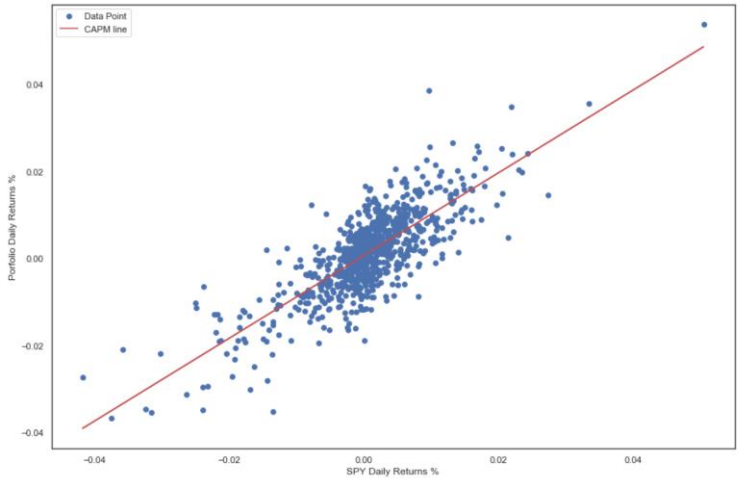


Figure 65 : CAPM Line

La régression linéaire nous permet d'exprimer cette relation sous la forme :

$$y = ax + b + e$$

Ici a est le Bêta du portfolio, ou la pente de la droite de régression, b est l'alpha du portefeuille ou l'ordonnée à l'origine et e le terme d'erreur. Ainsi exprimer la droite de régression devient :

$$ER_i = \beta ER_m + \alpha + e$$

L'alpha mesure l'excess returns généré par le portefeuille par rapport au benchmark. Cet indicateur permet également de savoir le style d'investissement du gérant. Un alpha proche de 0 indiquerait une gestion passive comme la réplication d'un indice alors qu'un alpha plus grand permettrait d'affirmer que le gérant est capable de générer plus de rendements que le benchmark et donc sa gestion serait plus active. La pente de la droite de régression est calculée avec la formule du Bêta, voilà pourquoi nous nous dirigeons vers cette solution.

En conclusion le Bêta de notre portefeuille est égal à 0,949 et l'alpha à 0,000 6. Ces valeurs nous permettent d'affirmer que notre portefeuille est positivement corrélé avec le marché américain. Quoiqu'il se passe sur les marchés, notre portefeuille le reflètera. C'est pourquoi les plus grosses chutes de valeur de notre portefeuille sont liées à de mauvaises conditions de marché. Notre choix d'actions ne s'est pas fait après le résutlat d'une recherche poussée, nous sommes exposés à un unique marché géographique et nous ne possédons qu'une classe d'actifs. Voilà les raisons de telles valeurs.

5.4.2.5 Simulation de Monte-Carlo

Pour terminer notre analyse financière, nous avons décidé de mettre en place une Simulation de Monte-Carlo afin de déterminer les possibles évolutions de la valeur de notre portefeuille sur la prochaine année soit 252 jours. Une simulation de Monte-Carlo revient à générer pour chaque trajectoire une série aléatoire de rendements suivant une loi normale de moyenne CAGR divisée par 252 et d'écart type égal à volatilité annuel divisé par racine de 252. Le but est d'ensuite générer une liste de 10 000 trajectoires possible suivant ces rendements qui sont régénérés à chaque nouvelle simulation.

$$Forecasted\ Returns\ for\ one\ day \sim N(\frac{CAGR}{252}, \frac{Annualized\ Volatility}{\sqrt{252}})$$

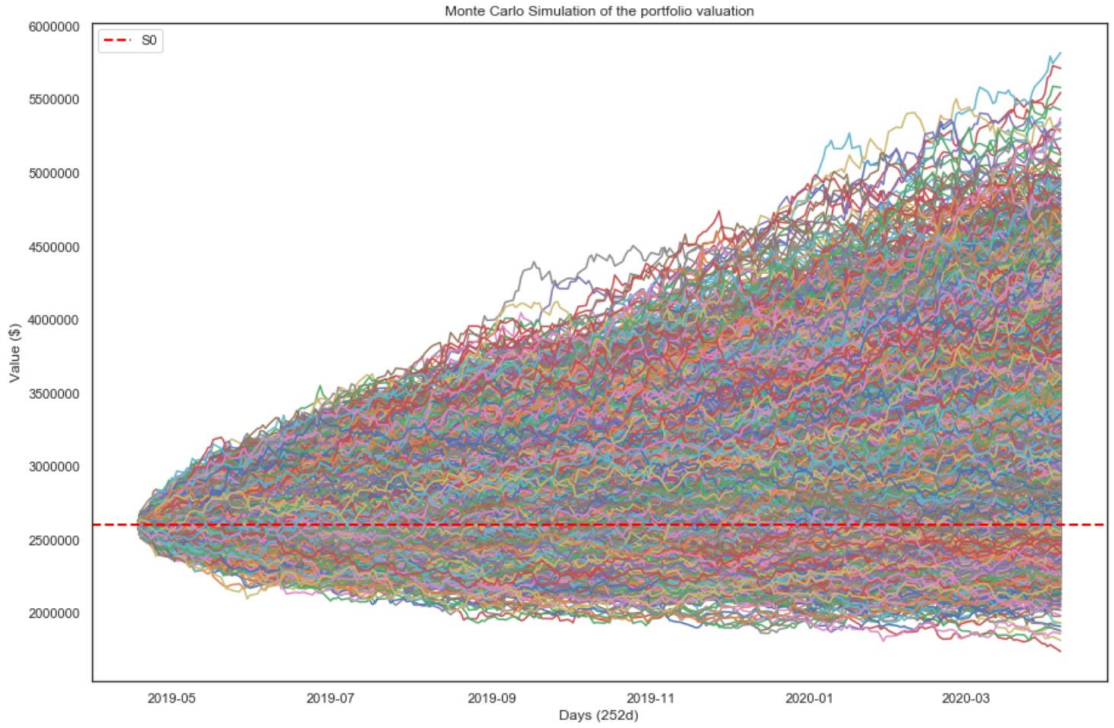


Figure 66 : Monte Carlo Simulation

Rien qu'avec ce graphique représentant l'ensemble des simulations, nous pouvons d'ores et déjà dire que la probabilité que le portefeuille perde de la valeur au bout de la prochaine année est nettement plus faible que celle où il est réévalué à la hausse. Pour en avoir le cœur net, nous dessinons le graphique de distribution des simulations.

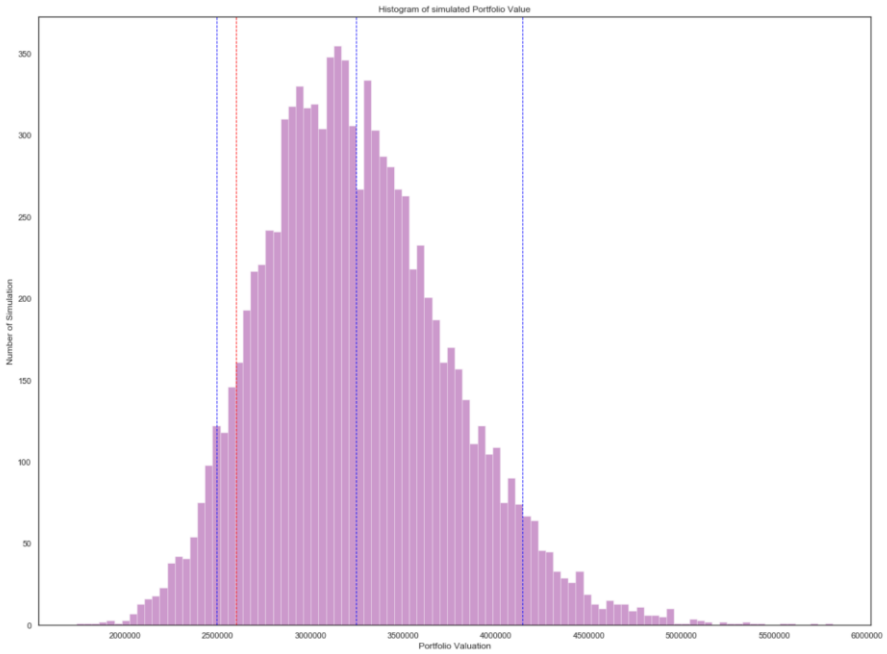


Figure 67 : Simulated Portfolio Valuation distribution

Sans surprise, la distribution des résultats est proche d'une loi normale. Le calcul des percentiles 5 et 95 nous indique que dans 90 % des cas la valeur du portefeuille sera comprise entre 2 493 729,98 $ et 4 141 557,41 $. La moyenne des simulations avoisine les $3 247 917,71, ce qui représente déjà une hausse de près de 25 % par rapport au prix initial (2 599 219,56 $). Une autre façon de voir ces simulations serait en termes de gains/pertes.

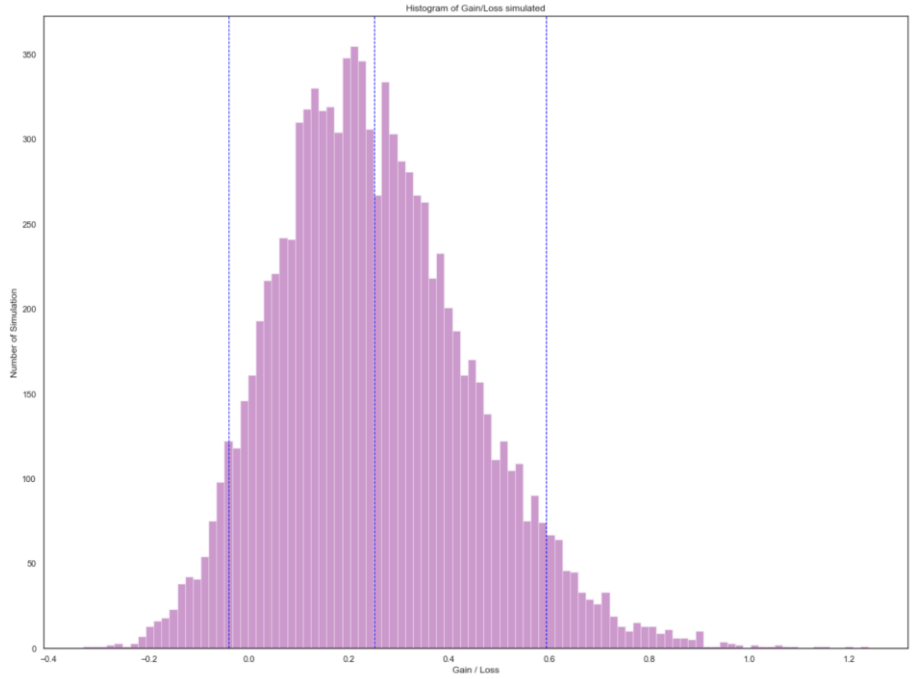


Figure 68 : Simulated Gains/Loss distribution

Ici, le percentile de 5 % nous indique donc que dans 5 % des cas notre portefeuille pourrait perdre 100 455,35 $ en une année soit une perte de 3,86 % de notre valeur initiale. Les gains moyens se monteraient à 640 981,14 $ soit la hausse de 25 % déjà calculée précédemment. Sur la base de ces rendements finaux, nous pouvons calculer la VaR annuel du portefeuille et en déduire la journalière.

```
--------------------------------------------------------------------------------
Annual VaR
VaR at 95% Confidence: -13.82%
VaR at 99% Confidence: -13.64%
VaR at 99.99% Confidence: -13.59%
--------------------------------------------------------------------------------
Monthly VaR
VaR at 95% Confidence: -3.09%
VaR at 99% Confidence: -3.05%
VaR at 99.99% Confidence: -3.04%
--------------------------------------------------------------------------------
Daily VaR
VaR at 95% Confidence: -0.87%
VaR at 99% Confidence: -0.86%
VaR at 99.99% Confidence: -0.86%
```

Figure 69 : Value At Risk

Que signifie la Value At Risk ? Simplement que sur une année complète, nous sommes sûr à 99,99 % que notre portefeuille ne perdra pas plus 13,59 % de sa valeur. Cependant, il reste 0,01 % chance pour nous perdions plus. Le même constat peut être fait de manière journalière avec une perte possible maximum de 0,86 % ou mensuel avec une perte maximum de 3.04%.

Voilà qui conclut notre ultime partie sur les résultats de notre développement et par la même occasion notre travail. Nous avons pu observer que Python et Spark nous offrent tout de même un bel éventail de possibilité de calcul et d'analyses. Même si le portefeuille témoin ici n'est pas des plus intéressants dans ses choix de titres, nos algorithmes peuvent désormais être appliqué à n'importe quel autre type de portefeuilles, il suffit de modifier le csv de départ. Le calcul de la matrice de corrélation, le calcul de l'allocation optimale et de la frontière efficiente, les différents indicateurs, ainsi que la simulation de Monte-Carlo et le calcul de la VaR se réaliseront de la même manière. Les pistes d'évolutions seraient le traitement des pondérations prédéfinies dans le csv de départ et bien évidemment le traitement d'autres classes d'actifs. Puis après avoir ajouté ces modifications, pourquoi ne pas construire un réel modèle prédictif basé sur de l'intelligence artificielle ?

Conclusion

L'ambition de ce travail était d'apporter un ensemble d'éléments de réponse à notre problématique : Comment les technologies Big Data et la Data Science vont-elles façonner le futur de la finance moderne ?

Dans un premier temps, il a fallu poser le contexte et définir les termes de cette recherche. Le Big Data est le résultat d'une course effrénée à l'évolution technologique dont les lois de Moore, Kryder & Nielsen sont le symbole. De plus, ces dernières années le nombre de sources de données n'a cessé d'exploser. L'IoT, l'utilisation des smartphones, les réseaux sociaux sont les principaux moteurs de cette explosion. De fait, le volume de données non structurées s'est décuplé et avec lui est apparu le besoin de l'exploiter. Après avoir analysé sa source, nous avons étudié les caractéristiques intrinsèques du Big Data. De nombreuses définitions prêtent au Big Data les traits de Vélocité, Volume et Variété, ce que nous avons appelé le mythe des 3 V's. Dans notre recherche, nous avons préféré retenir les caractéristiques d'Exhaustivité et de Vélocité qui permettent amplement de distinguer le Big Data du Small Data et qui reste les points communs les plus discriminants des données dites Big Data. En étant moins stricte dans notre définition, le Big Data obtient une certaine granularité. Cela nous permet d'affirmer qu'il n'y a pas qu'un seul type de big data même une multitude suivant l'usage et le type des données. Définir strictement un terme, c'est déjà réduire son ampleur. Il était important de rester objectif et prendre le plus de recul sur cette définition. Notre source pour cette définition a, selon nous, cerné les enjeux du choix des mots au termes d'une analyse exhaustive.

Dans un second temps, nous sommes concentrés plus particulièrement sur la Data Science. La définition à retenir c'est celle du raisonnement inductif. Ce type de raisonnement est tourné vers le futur. C'est en partant des données que nous définissons des modèles qui nous permettront d'anticiper certains comportements. C'est le but de la Data Science. Pour comprendre l'ampleur de la tâche des équipes analytiques, nous avons décrit leurs tâches et leurs compétences. Avec des horizons différents, des diplômes différents et donc de compétences différentes, il est clair que la data science est une histoire d'équipe. Chacun à son importance et son rôle mais ajouter une dimension analytique à une entreprise ne se fait certainement pas avec une seule et même personne. Bien que les matières informatiques et quantitatives soient primordiales, nous avons soulignés également l'importance des compétences business et des softs skills. En Data Science, il est nécessaire de réunir les bonnes informations, créer des modèles robustes et des architectures adaptées, mais surtout pouvoir communiquer ces

résultats de manières fluides à son management ou à ses clients. Sans quoi le travail effectué n'a plus de valeurs stratégiques.

Au préalable de notre développement informatique, nous avons, ensuite, décidé de nous plonger dans le domaine technique. Nous ne voulions pas utiliser des technologies sans les comprendre. De plus, l'apparition des technologies citées nous apporte quelques éléments de réponses sur l'impact du Big Data sur le monde professionnel. L'architecture reine du Big Data est bien évidemment Hadoop qui reprend les principes de bases explorés par Google (GFS & Mapreduce). Son apparition a été motivée par l'explosion des volumes de données, la nécessité d'aller encore plus vite et la maîtrise des coûts par le remplacement du scaling-up par le scaling-out. Grâce à Hadoop, nous assistons désormais à l'avènement de la computation parallèle. Cette dernière offre des possibilités quasiment infinies en termes de puissance et de réalisation pour les équipes de Data Science. Sans elle, le big data n'a pas de raison d'être. Puis nous avons scruté l'apparition du framework Spark qui a encore rebattu les cartes. Spark fait mieux, plus vite, et tout simplement plus qu'Hadoop. C'est la réponse des développeurs à une architecture en kit qui commençait à devenir lourde à gérer. Avec Spark, les entreprises telles que les Banques peuvent tester plus de modèles plus vite et de manière presque continue sans avoir de risques de pertes de données. Le Big Data peut enfin être exploité à sa juste valeur.

Ces deux premières parties nous ont permis de démystifier certaines idées sur le Big Data et la Data Science. Si nous ne comprenions pas les notions de base de notre travail et son contexte, il nous semblait compliquer de pouvoir comprendre l'ensemble des enjeux impactant notre réponse à notre problématique. Poser ces définitions nous ont permis de poser un cadre et commencer ainsi à répondre à notre problématique.

D'un point de vue purement RH, nous avons pu voir que les équipes dédiées à la Data Science regorgent de nouveau style de talents. Ces derniers sont parfois hors des murs des institutions financières et cela force au recrutement. D'un point de vue technique, les technologies étudiées sont des leviers qui permettent au service financier d'exploiter pleinement le potentiel des données qu'ils collectent. Sans ces technologies, le traitement et les simulations de modèles à grandes échelles deviennent impossibles. La mise à jour des systèmes informatiques des banques est donc également l'un des impacts de Data Science sur la finance. Cela se reflètent dans leurs investissements. Des partenariats des services financiers avec les BigTechs à des acquisitions de start-up fintech tout est bon pour ne pas se laisser distancer par la concurrence et emmagasiner un savoir-faire parfois hors des compétences internes.

Sans les talents de la data science et ces technologies, la finance comme les autres industries dans le monde ne pourrait imaginer extraire des informations désormais vitales pour leur futur. La suite de notre travail s'est attelée ensuite sur comment le big data s'intègrait au milieu de la finance. Tout d'abord, pourquoi la data science a pu s'imposé dans les banques ? La menace de nouveaux entrants à l'image des BigTechs ou Fintech sur les marchés, l'environnement de taux très bas et la hausse des coûts de régulations font pression sur les marges de l'ensemble du secteur financier. Quelles solutions ? Développé de nouvelles idées, services, exploités de nouvelles ressources afin de rester compétitifs. Ainsi la Data Science est devenue l'une des premières préoccupations des banques. Mais pourquoi le big data ? Les banques c'est entre 80 % et 90 % de données non structurées : le royaume de la data Science. Lorsque les données sont structurées par exemple celle des marchés, la vitesse de traitement reste un avantage. De plus, croiser et enrichir les données structurées avec d'autres non structurées permet de découvrir de nouvelles opportunités jusque-là indécelables. C'est par ce jeu de corrélation que la science des données ouvre de nouveaux horizons à la finance. C'est ainsi que notre travail nous a amené à décrire différents use cases de la Data Science appliquée à la finance, éléments de réponse clés à notre problématique. Le management de la performance est influencé par le Big Data. En effet, l'apparition de divers méthodes de trading algorithmique ont changés la physionomie des marchés. Le Back Testing de stratégies, requis légalement, a été non seulement accéléré mais aussi multiplié.. Puis avec une approche plus orientée processus, la data science peut consolider et améliorer les procédures actuelles en diminuant le risque d'erreur ou accélérant le temps de traitement via la RPA. Le marketing est également impacté avec un enrichissement des données traditionnelles avec un nouveau type d'information. Un profilage pertinent peut se mettre en place et augmenter le nombre d'adhérents, diminuer le nombre de départ ou simplement atteindre plus facilement une cible précise. Enfin le management du risque opérationnel tel que les fraudes ou les cyberattaques peut aussi profiter du pouvoir du Big Data. Les banques font face à une augmentation des réglementations et la technologie peut les aider à mettre en place les nouveaux processus qui en découlent afin de rester compliant et adopter une attitude prescriptive. Il n'y a pas un secteur de la finance ou le Big Data ne peut pas apporter une plus-value. Même l'advisory, symbole du contact humain, peut être assisté par ordinateur. La compétitivité s'en trouve amélioré et les modèles d'affaires commencent donc à transformer.

C'est dans ce cadre théorique dense que nous avons souhaité par la suite développer une solution informatique dédiée à la gestion de portefeuille. Le but était de démontrer l'efficacité des technologies Big Data dans un simple Proof of Concept. Techniquement,

les limites de notre environnement de test et du choix de nos données ne nous ont pas permis d'exploiter Apache Spark à son plein potentiel. Il est cependant clair que dans un environnement de la taille d'une multinationale son utilisation offre de nouvelles perspectives en termes de rapidités, volume et d'efficacité. Nos tests le prouvent, sans Spark les temps d'exécutions s'allongent proportionnellement au volume traité. Cependant, notre développement nous a tout de même permis de simuler la création de 50 000 portefeuilles en 5 min pour décider de notre allocation puis de simuler 10 000 trajectoires futures en moins de 2 minutes. Ces performances ont été réalisées avec un petit jeu de données dans un environnement local. Imaginez ce qu'une centaine de machines pourrait réaliser sur des données en streaming. Ce développement nous a permis de nous exposer à ces nouvelles technologies afin de nous rendre compte de leurs potentiels. La data science a déjà commencé à apporter énormément à la finance moderne. La recherche et simulation de ces concepts nous ont permis d'appréhender encore mieux le véritable aspect de la data science dans un domaine tel que la finance. En quelques mois, à l'aide nos connaissances financières et de l'apprentissage des bases des technologies Big Data, il nous a été possible de rendre un rapport détaillé avec des « insights » actionnables. Le résultat du travail de toute une équipe dédiée ne peut être qu'impressionnant dans de telles conditions.

Ce travail voulait étudier notre problématique de deux points de vue différents celui de la technique et celui du métier. C'est un parti pris qui a pu nous coûter du temps et limiter nos tests. Notre développement pourrait largement être amélioré avec une meilleure exploitation de Spark et le choix d'un meilleur data models. Du point de vue financier, il est certain que notre analyse n'est pas complète et s'apparente plus à un raisonnement déductif qu'inductif. Une piste de recherche supplémentaire qui ajouterait beaucoup à notre travail serait l'étude et la maîtrise de l'intelligence artificielle. D'autres part, le travail préliminaire de stock picking pourrait être implémentés être algorithmiquement avec un accès aux données financières propres aux entreprises. Pourquoi pas envisager un outil d'analyses financières recommandant ensuite les actions et autres titres à prendre ? La Data Science n'a de limite que l'imagination de ses développeurs.

En conclusion, nous pouvons donc affirmer que la technologie et la Data Science sont déjà au service de la finance moderne. C'est à elle de savoir l'exploiter pour améliorer son service et sa compétitivité. Au travers de notre travail, il est déjà clair qu'elle possède déjà les clés de recherche pour le faire.

Bibliographie

ABRAHAM, Facundo, SCHMUKLER, Sergio et TESSADA, José, 2019. *Robo-Advisors: Investing through Machines* [en ligne]. 21 février 2019. S.l. : World Bank Group. [Consulté le 1 avril 2019]. Disponible à l'adresse : http://documents.worldbank.org/curated/en/275041551196836758/pdf/Robo-Advisors-Investing-through-Machines.pdf.

ADEWOLU, Bayo, 2018. A perspective on FinTech's growing influence on Financial Services. In : . juillet 2018. p. 25.

AGARWAL, Anuj, 2012. *High Frequency Trading: Evolution and the Future* [en ligne]. S.l. Capgemini Consulting. [Consulté le 1 avril 2019]. Disponible à l'adresse : https://www.capgemini.com/wp-content/uploads/2017/07/High_Frequency_Trading__Evolution_and_the_Future.pdf.

ALIBABA, 2016. *Ant Financial* [en ligne]. juin 2016. S.l. : s.n. [Consulté le 1 avril 2019]. Disponible à l'adresse : https://www.alibabagroup.com/en/ir/pdf/160614/12.pdf.

ARIELY, Dan, 2013. Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone... http://fb.me/1r54f62oU. In : *@danariely* [en ligne]. 6 janvier 2013. [Consulté le 6 février 2019]. Disponible à l'adresse : https://twitter.com/danariely/status/287952257926971392?lang=fr.

AROCKIA PANIMALAR, S., VARNEKHA SHREE, S. et VENESHIA KATHRINE, A., 2017. The 17 V's Of Big Data. In : [en ligne]. septembre 2017. Vol. 4, n° 9. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www.irjet.net/archives/V4/i9/IRJET-V4I957.pdf.

ATKINS, Adam, NIRANJAN, Mahesan et GERDING, Enrico, 2018. Financial news predicts stock market volatility better than close price. In : *The Journal of Finance and Data Science*. 1 juin 2018. Vol. 4, n° 2, p. 120-137. DOI 10.1016/j.jfds.2018.02.002.

BAKSHI, Ashish, 2016. MapReduce Tutorial | Mapreduce Example in Apache Hadoop. In : *Edureka* [en ligne]. 15 novembre 2016. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.edureka.co/blog/mapreduce-tutorial/.

BANK FOR INTERNATIONAL SETTLEMENTS, 2019. Minimum capital requirements for market risk. In : . janvier 2019. p. 136.

BEEKMAN, Bernie, 2017. IBM Data Science Experience Overview. In : [en ligne]. S.l. 2017. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www-01.ibm.com/events/wwe/grp/grp304.nsf/vLookupPDFs/Bernie%20Beekman%20Presentation/$file/Bernie%20Beekman%20Presentation.pdf.

BIERNAT, Érice et LUTZ, Michel, 2015. *Data Science : Fondamentaux et études de cas*. S.l. : Eyrolles. ISBN 978-2-212-14243-3.

BIREN GANDHI, 2015. Get Cloud Resources to the IoT Edge with Fog Computing. In : [en ligne]. Internet. S.l. 25 août 2015. [Consulté le 5 février 2019]. Disponible à l'adresse : https://www.slideshare.net/biren_gandhi/get-cloud-resources-to-the-iot-edge-with-fog-computing.

BOOZ ALLEN HAMILTON, 2015. The Field Guide to Data Science. In : . 2015. p. 126.

BURTCH, Linda, 2018. *Salaries of Data Scientist* [en ligne]. USA. Burtch Works - Executive recruiting. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www.burtchworks.com/wp-content/uploads/2018/05/Burtch-Works-Study_DS-2018.pdf.

CACKETT, Doug, BOND, Andrew et GOUK, John, 2013. *Information Management and Big Data. A Reference Architecture* [en ligne]. Redwood Shores, CA. Oracle. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www.oracle.com/technetwork/topics/entarch/articles/info-mgmt-big-data-ref-arch-1902853.pdf.

CHAO, Gene, HURST, Elli et SCHOCKLEY, Rebecca, 2018. The evolution of process automation. In : . 2018. p. 13.

CLOUDERA, INC., 2015. *Apache Hadoop & Big Data 101: The Basics* [en ligne]. 2015. [Consulté le 3 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=AZovvBgRLIY&ab_channel=Cloudera%2CInc.

COUMAROS, Jean, DE ROAYS, Stanislas, CHRETIEN, Laurence, BUVAT, Jerome, KVJ, Subrahmanyam, CLERK, Vishal et AULIARD, Olivier, 2014. *Big Data Alchemy: How can Banks Maximize the Value of their Customer Data?* [en ligne]. S.l. Capgemini Consulting. [Consulté le 18 mars 2019]. Disponible à l'adresse : https://www.capgemini.com/wp-content/uploads/2017/07/bigdatainbanking_2705_v5_1.pdf.

COX, Jeremy, 2016. Customer-Centricity in Financial Services: The Rules of Engagement. In : . 2016. p. 19.

DATABRICKS, 2019. Apache Spark Tutorial: Getting Started with Apache Spark Tutorial. In : *Databricks* [en ligne]. 2019. [Consulté le 3 mars 2019]. Disponible à l'adresse : https://databricks.com/spark/getting-started-with-apache-spark.

DAVENPORT, Thomas H. et PATIL, D. J., 2012. Data Scientist: The Sexiest Job of the 21st Century. In : *Harvard Business Review* [en ligne]. 1 octobre 2012. n° October 2012. [Consulté le 8 février 2019]. Disponible à l'adresse : https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century.

DE LA MANO, Miguel et PADILLA, Jorge, 2018. *Big Tech Banking* [en ligne]. S.l. [Consulté le 31 mars 2019]. Disponible à l'adresse : https://s1.aebanca.es/wp-content/uploads/2018/12/de-la-mano-padilla-2018-big-tech-banking-15.0.pdf.

DEAN, Jeffrey et GHEMAWAT, Sanjay, 2008. MapReduce: simplified data processing on large clusters. In : *Communications of the ACM*. 1 janvier 2008. Vol. 51, n° 1, p. 107. DOI 10.1145/1327452.1327492.

DELORT, Pierre, 2018. Le Big Data. In : *Le Big Data*. 2ème. Paris : Paul Angoulvent. Que sais-je ? p. 6. ISBN 978-2-13-079221-5.

DEMCHENKO, Yuri, BELLOUM, Adam et WIKTORSKI, Tomasz, 2017. 675419 : *EDISON Data Science Framework: Part 4. Data Science Professional Profiles (DSPP) Release 2* [en ligne]. Amsterdam. [Consulté le 1 février 2019]. Disponible à l'adresse : http://edison-project.eu/sites/edison-project.eu/files/attached_files/node-486/edison-dspp-release2-v04.pdf.

EDUCBA, 2018. HADOOP vs RDBMS |Know The 12 Useful Differences. In : *EDUCBA* [en ligne]. 16 février 2018. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.educba.com/hadoop-vs-rdbms/.

EVANS, Dave, 2011. *How the Next Evolution of the Internet Is Changing Everything*. S.l.

FENG, Wenqiang, 2019. Learning Apache Spark with Python. In : . 2019. p. 231.

FINANCIAL STABILITY BOARD, 2018. *FinTech and market structure in financial services:Market developments and potential financial stability implications* [en ligne]. S.l. [Consulté le 31 mars 2019]. Disponible à l'adresse : http://www.fsb.org/wp-content/uploads/P140219.pdf.

GASSER, Urs, GASSMANN, Oliver, HENS, Thorsten, LEIFER, Larry, PUSCHMANN, Thomas et ZHAO, Leon, 2018. *Digital Banking 2025* [en ligne]. S.l. [Consulté le 31 mars 2019]. Disponible à l'adresse : http://www.fintech.uzh.ch/dam/jcr:2ef31105-1a60-43da-a657-8b140e47b3b4/Digital%20Banking%202025%20FINAL.pdf.

GERINGER, Steve, 2014. Data Science Venn Diagram v2.0. In : *Steve's Machine Learning Blog* [en ligne]. 2014. [Consulté le 9 février 2019]. Disponible à l'adresse : http://www.anlytcs.com/2014/01/data-science-venn-diagram-v20.html.

GHEMAWAT, Sanjay, GOBIOFF, Howard et LEUNG, Shun-Tak, 2003. *The Google File System* [en ligne]. New York, USA. Google. [Consulté le 1 février 2019]. Disponible à l'adresse : http://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf.

GHOSH, Paramita, 2018. Data Science vs. Business Intelligence. In : *DATAVERSITY* [en ligne]. 13 mars 2018. [Consulté le 9 février 2019]. Disponible à l'adresse : https://www.dataversity.net/data-science-vs-business-intelligence/.

GLOBAL RATES, 2019. Central banks, summary of current interest rates. In : *Global Rates* [en ligne]. 2019. [Consulté le 1 avril 2019]. Disponible à l'adresse : https://www.global-rates.com/interest-rates/central-banks/central-banks.aspx.

GOOGLE TRENDS, 2019. Google Trends. In : *Google Trends* [en ligne]. 8 février 2019. [Consulté le 8 février 2019]. Disponible à l'adresse : https://trends.google.fr/trends/explore?date=all&q=big%20data,data%20science,machine%20learning.

GSMA, 2018. *The Mobile Economy Global 2018* [en ligne]. London. GSM Association. [Consulté le 1 février 2018]. Disponible à l'adresse : https://www.gsma.com/mobileeconomy/wp-content/uploads/2018/02/The-Mobile-Economy-Global-2018.pdf.

GUIDEWIRE SOFTWARE, 2018. Guidewire Cyence Risk Analytics. In : *Guidewire* [en ligne]. 12 juin 2018. [Consulté le 29 avril 2019]. Disponible à l'adresse : https://www.guidewire.com/products/guidewire-cyence-risk-analytics.

GUTIERREZ, Daniel, 2017. *Big Data for finance* [en ligne]. S.l. Dell & Intel. [Consulté le 1 avril 2019]. Disponible à l'adresse : https://www.em360tech.com/wp-content/uploads/2017/10/Whitepaper-Big-Data-in-Finance.pdf.

HALE, Jeff, 2018. The Most in Demand Skills for Data Scientists. In : *Towards Data Science* [en ligne]. 12 octobre 2018. [Consulté le 17 février 2019]. Disponible à l'adresse : https://towardsdatascience.com/the-most-in-demand-skills-for-data-scientists-4a4a8db896db.

HARRIS, Harlan D., MURPHY, Sean Patrick et VAISMAN, Marck, 2013. *Analyzing the analyzers: an introspective survey of data scientists and their work*. First edition. Beijing : O'Reilly. ISBN 978-1-4493-7176-0. HD8039.I37 H37 2013

HARRIS, Keiren, 2018. Top Global Financial Information Providers, Analysis of Who, Why & Future Trends. In : *The Evolving World of Market Data* [en ligne]. 16 mai 2018. [Consulté le 6 février 2019]. Disponible à l'adresse : https://www.marketdata.guru/2018/05/16/top-global-financial-information-providers-analysis-of-who-why-future-trends/.

HAYES, Bob E, 2015. Optimizing your Data Science Team. In : . 2015. p. 4.

HORTONWORKS, 2013. *Basic Introduction to Apache Hadoop* [en ligne]. 2013. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=OoEpfb6yga8&ab_channel=Hortonworks.

HURLEY, Mikella et ADEBAYO, Julius, 2017a. CREDIT SCORING IN THE ERA OF BIG DATA. In : *Big Data*. 2017. Vol. 18, p. 70.

HURLEY, Mikella et ADEBAYO, Julius, 2017b. CREDIT SCORING IN THE ERA OF BIG DATA. In : *Big Data*. 2017. Vol. 18, p. 70.

IBM, 2017. IBM Z - The Banking Platform for the Future. In : [en ligne]. 17 octobre 2017. [Consulté le 18 mars 2019]. Disponible à l'adresse : https://www.ibm.com/industries/banking-financial-markets/resources/core-banking/ibm-z-banking-platform/.

IDC, 2018. Revenues for Big Data and Business Analytics Solutions Forecast to Reach $260 Billion in 2022, Led by the Banking and Manufacturing Industries, According to IDC. In : *IDC: The premier global market intelligence company* [en ligne]. août 2018. [Consulté le 8 avril 2019]. Disponible à l'adresse : https://www.idc.com/getdoc.jsp?containerId=prUS44215218.

IMBERT, Fred, 2018. US stocks post worst year in a decade as the S&P 500 falls more than 6% in 2018. In : *CNBC* [en ligne]. 31 décembre 2018. [Consulté le 9 juillet 2019]. Disponible à l'adresse : https://www.cnbc.com/2018/12/31/stock-market-wall-street-stocks-eye-us-china-trade-talks.html.

INTRICITY101, 2012a. *What is Hadoop?* [en ligne]. 2012. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=9s-vSeWej1U&ab_channel=Intricity101.

INTRICITY101, 2012b. *What is Hadoop?: SQL Comparison* [en ligne]. 2012. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=MfF750YVDxM&ab_channel=Intricity101.

ISHWARAPPA et ANURADHA, J., 2015. A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. In : *Procedia Computer Science*. 2015. Vol. 48, p. 319-324. DOI 10.1016/j.procs.2015.04.188.

KANE, Frank, 2017. *Frank Kane's Taming Big Data with Apache Spark and Python*. Packt Publishing. Birmingham : Packt. Packt Publishin. ISBN 978-1-78728-794-5.

KITCHIN, Rob et MCARDLE, Gavin, 2016. What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets. In : *Big Data & Society* [en ligne]. août 2016. Vol. 1, n° 10. [Consulté le 1 février 2019]. DOI 10.1177/2053951716631130. Disponible à l'adresse : http://eprints.maynoothuniversity.ie/7278/1/RK-Big-Data.pdf.

KLINE, Kevin, 2003. The Database Pro's Guide to the Team Data Science Process. In : . 2003. p. 35.

KOBIELUS, James, 2013. Measuring the Business Value of Big Data. In : *IBM Big Data Hub* [en ligne]. 9 mai 2013. [Consulté le 5 février 2019]. Disponible à l'adresse : /blog/measuring-business-value-big-data.

LANEY, Doug, 2001. 3D Data Management: Controlling Data Volume, Velocity and Variety. In : [en ligne]. 6 février 2001. n° 949. [Consulté le 1 février 2019]. Disponible à l'adresse : https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf.

LEVI, David, SOLOMOU, Luke et MCDONALD, Craig, 2018. *Bank Data, Untapped high return asset or a junk bond ?* [en ligne]. S.l. Accenture Strategy. [Consulté le 1 juin 2019]. Disponible à l'adresse : https://www.accenture.com/t00010101T000000Z__w__/au-en/_acnmedia/PDF-74/Accenture-Bank-Data-Asset-Value-or-Junk-Bond.pdf.

MÄDER, Patrick et AKIKI, Patrick, 2017. Reimagine and transform your finance function in the digital age. In : *Big Data*. avril 2017. p. 44.

MARR, Bernard, 2018. How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read. In : *Forbes* [en ligne]. 21 mai 2018. [Consulté le 4 février 2019]. Disponible à l'adresse : https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/.

MCKINSEY & COMPANY, 2018. 11 : *McKinsey on Payments Special Edition on Advanced Analytics in Banking* [en ligne]. USA. McKinsey & Company. [Consulté le 1 juin 2019]. Disponible à l'adresse : https://www.mckinsey.com/~/media/McKinsey/Industries/Financial%20Services/Our%20Insights/McKinsey%20on%20Payments%2028%20Special%20edition%20on%20advanced%20analytics%20in%20banking/McK-on-Payments-28-Special-Edition-Advanced-Analytics-in-Banking.ashx.

MCWATERS, R. Jesse, 2015. *The Future of Financial Services How disruptive innovations are reshaping the way financial services are structured, provisioned and consumed* [en ligne]. S.l. World Economic Forum & Deloitte. [Consulté le 16 mars 2019]. Disponible à l'adresse : http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf.

MOORE, E. Moore, 1965. Cramming more components onto integrated circuits. In : *Electronics Magazine* [en ligne]. 19 avril 1965. [Consulté le 4 février 2018]. Disponible à l'adresse : https://hasler.ece.gatech.edu/Published_papers/Technology_overview/gordon_moore_1965_article.pdf.

NAN, Siyu et SU, Zhongda, 2017. Spark vs. Hadoop MapReduce. In : . 2017. p. 4.

NASDAQ, 2019. Full Market Share Summary. In : *NasdaqTrader.com* [en ligne]. février 2019. [Consulté le 6 février 2019]. Disponible à l'adresse : http://www.nasdaqtrader.com/trader.aspx?id=FullVolumeSummary#.

NATHAN, Paco, 2017. Intro to Apache Spark. In : [en ligne]. Workshop. S.l. 2017. [Consulté le 1 février 2019]. Disponible à l'adresse : https://stanford.edu/~rezab/sparkclass/slides/itas_workshop.pdf.

NIELSEN, Jakob, 1998. Nielsen's Law of Internet Bandwidth. In : *Nielsen Norman Group* [en ligne]. 6 janvier 1998. [Consulté le 4 février 2019]. Disponible à l'adresse : https://www.nngroup.com/articles/law-of-bandwidth/.

NIST BIG DATA PUBLIC WORKING GROUP DEFINITIONS AND TAXONOMIES SUBGROUP, 2015. NIST SP 1500-1 : *NIST Big Data Interoperability Framework: Volume 1, Definitions* [en ligne]. S.l. National Institute of Standards and Technology. [Consulté le 8 février 2019]. Disponible à l'adresse : https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1500-1.pdf.

OLIVER WYMAN, 2018. The state of the financial services industry. In : . 2018. p. 41.

ORACLE CORPORATIONS, 2012. Financial Services Data Management. In : . juin 2012. p. 23.

ORACLE CORPORATIONS, 2015. *Big Data in Financial Services and Banking Architect's Guide and Reference Architecture Introduction* [en ligne]. USA. Oracle Corporations. [Consulté le 16 mars 2019]. Disponible à l'adresse : http://www.oracle.com/us/technologies/big-data/big-data-in-financial-services-wp-2415760.pdf.

PARKS, Dana M Dedge, 2017. Defining Data Science and Data Scientist. In : . 10 décembre 2017. p. 78.

PIECUCH, John et STOTT, Diana, 2018. S&P GLOBAL RATINGS360™ TO INCLUDE CYBER RISK INSIGHTS FROM GUIDEWIRE SOFTWARE'S CYENCE RISK ANALYTICS - Media Releases - S&P Global Ratings. In : [en ligne]. 16 février 2018. [Consulté le 29 avril 2019]. Disponible à l'adresse : https://www.spratings.com/en_US/media-releases/-/asset_publisher/cebizYBoiIER/content/s-p-global-ratings360-to-include-cyber-risk-insights-from-guidewire-software-s-cyence-risk-analytics?inheritRedirect=false.

POLLARI, Ian et RUDDENKLAU, Anton, 2019. *The Pulse of Fintech 2018* [en ligne]. S.l. KPMG. [Consulté le 31 mars 2019]. Disponible à l'adresse : https://assets.kpmg/content/dam/kpmg/xx/pdf/2019/02/the-pulse-of-fintech-2018.pdf.

PYTHON, Real, 2018. 8 World-Class Software Companies That Use Python – Real Python. In : [en ligne]. 2018. [Consulté le 10 mars 2019]. Disponible à l'adresse : https://realpython.com/world-class-companies-using-python/.

RADOVILSKY, Zinovy, HEGDE, Vishwanath et ACHARYA, Anuja, 2018. Skills Requirements of Business Data Analytics and Data Science Jobs: A Comparative Analysis. In : . 2018. Vol. 16, n° 1, p. 20.

ROSSUM, Guido Van, 2009. The History of Python: A Brief Timeline of Python. In : *The History of Python* [en ligne]. 20 janvier 2009. [Consulté le 10 mars 2019]. Disponible à l'adresse : https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html.

ROY, Subas, HEANEY, Michael et SEIBERT, Hanjo, 2018. TRANSFORMING COMPLIANCE INTO COMPETITIVE ADVANTAGE. In : . 2018. p. 10.

RUDDENKLAU, Anton, 2018. Tech giants in financial services. In : . janvier 2018. p. 2.

SALTZ, J. S. et GRADY, N. W., 2017. The ambiguity of data science team roles and the need for a data science workforce framework. In : *2017 IEEE International Conference on Big Data (Big Data)*. S.l. : s.n. décembre 2017. p. 2355-2361.

SARROCCO, Fabrizio, MORABITO, Vincenzo et MEYER, Gregor, 2016. Exploring Next Generation Financial Services - The Big Data Revolution. In : . 2016. p. 12.

SERRA, James et PASSTV, 2016. *24 Hours of PASS: EDP 2016 - Relational Databases vs Non Relational Databases vs Hadoop* [en ligne]. 2016. [Consulté le 2 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=_joSrDXAVh0&ab_channel=PASStv.

SPARK SUMMIT, 2013. *Spark Summit 2013 - The State of Spark, and Where We're Going Next - Matei Zaharia* [en ligne]. 2013. [Consulté le 3 mars 2019]. Disponible à l'adresse : https://www.youtube.com/watch?v=nU6vO2EJAb4&feature=youtu.be&ab_channel=SparkSummit.

SPRINGBOARD, 2016. *A beginner's guide to Getting your first data science job* [en ligne]. S.l. Springboard. [Consulté le 1 février 2019]. Disponible à l'adresse : https://d1v7bd3b0sjvlo.cloudfront.net/uploads/resources/1475623482_Data_Science_Careers_Guide_Springboard_Final-1-1.pdf.

STACKOVERFLOW, 2018. Stack Overflow Developer Survey 2018. In : *Stack Overflow* [en ligne]. 2018. [Consulté le 10 mars 2019]. Disponible à l'adresse : https://stackoverflow.com/insights/survey/2018/?utm_source=so-owned&utm_medium=social&utm_campaign=dev-survey-2018&utm_content=social-share.

STAFF, Motley Fool, 2019. 20 of the Top Stocks to Buy in 2019 (Including the 2 Every Investor Should Own) -. In : *The Motley Fool* [en ligne]. 3 avril 2019. [Consulté le 9 juillet 2019]. Disponible à l'adresse : https://www.fool.com/investing/top-stocks-to-buy.aspx.

STATISTA, 2019. Global social media ranking 2018 | Statistic. In : *Statista* [en ligne]. janvier 2019. [Consulté le 4 février 2019]. Disponible à l'adresse : https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/.

SWINNEN, Gérard, 2008. Apprendre à programmer avec Python. In : *Developpez.com* [en ligne]. mai 2008. [Consulté le 10 mars 2019]. Disponible à l'adresse : http://python.developpez.com/cours/TutoSwinnen/.

SZMUKLER, Daniel, 2017. *Data Exploration Opportunities in Corporate Banking Key concepts and applications* [en ligne]. Paris. Euro Banking Association. [Consulté le 14 mars 2019]. Disponible à l'adresse : https://www.abe-

eba.eu/epaper/epaper-data_exploration_opportunities_in_corporate_banking/epaper/ausgabe.pdf.

THOMSON REUTERS FINANCIAL, 2007. *Worldscope Database Datatype Definitions Guide* [en ligne]. 2007. S.l. : Thomson Reuters. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www.unisg.ch/-/media/dateien/unisg/bibliothek/recherche/datenbanken/thomsononebanker/t1b_datatypedefinitionsguide.pdf?la=en&hash=2CDADC3A77771CA0A7765EC9248FDF1178D130E2.

THOMSON REUTERS FINANCIAL, 2018. Cost of Compliance 2018 Report: Your biggest challenges revealed. In : [en ligne]. 2018. [Consulté le 21 avril 2019]. Disponible à l'adresse : https://legal.thomsonreuters.com/en/insights/articles/cost-of-compliance-2018-report-your-biggest-challenges-revealed.

TIAN, Xinhui, HAN, Rui, WANG, Lei, LU, Gang et ZHAN, Jianfeng, 2015. Latency critical big data computing in finance. In : *The Journal of Finance and Data Science*. 1 décembre 2015. Vol. 1, n° 1, p. 33-41. DOI 10.1016/j.jfds.2015.07.002.

TIAN, Zack, 2018. RPA + data analytics = an unprecedented force for digital transformation. In : [en ligne]. 7 août 2018. [Consulté le 23 avril 2019]. Disponible à l'adresse : https://www.pwc.ch/en/insights/digital/rpa-data-analytics-an-unprecedented-force-for-digital-transformation.html.

TILL ALEXANDER, Leopold, SAADIA, Zahidi et VESSELINA, Ratcheva, 2018. *The Future of Jobs Report 2018* [en ligne]. Insight Report. Genève. World Economic Forum. Disponible à l'adresse : http://www3.weforum.org/docs/WEF_Future_of_Jobs_2018.pdf.

TIOBE, 2019. TIOBE Index | TIOBE - The Software Quality Company. In : [en ligne]. mars 2019. [Consulté le 10 mars 2019]. Disponible à l'adresse : https://www.tiobe.com/tiobe-index/.

TORLONE, Ricardo, 2016. Hadoop & MapReduce. In : [en ligne]. Rome. 2016. [Consulté le 1 février 2019]. Disponible à l'adresse : http://torlone.dia.uniroma3.it/bigdata/L2-Hadoop.pdf.

TURNER, David, SCHROECK, Michael et SHOCKLEY, Rebecca, 2013. *Analytics: The real-world use of big data in financial services* [en ligne]. USA. IBM® Institute for Business Value, Saïd Business School at the University of Oxford. [Consulté le 1 février 2019]. Disponible à l'adresse : https://www-935.ibm.com/services/multimedia/Analytics_The_real_world_use_of_big_data_in_Financial_services_Mai_2013.pdf.

VALCHANOV, Iliya, 2018. A Visual Guide to Analytics, Data Science, Business Intelligence, Machine Learning, and AI. In : [en ligne]. 3 mai 2018. [Consulté le 9 février 2019]. Disponible à l'adresse : https://www.datascience.com/blog/data-science-vs-business-intelligence-machine-learning-ai.

VALE, Steven, 2013. Classification of Types of Big Data - Big Data - UNECE Statswiki. In : *UNECE Statswiki* [en ligne]. 27 juin 2013. [Consulté le 4 février 2019]. Disponible à l'adresse : https://statswiki.unece.org/display/bigdata/Classification+of+Types+of+Big+Data.

VALUEPENGUIN, 2019. Consumer Banking: Statistics and Trends in 2019. In : *ValuePenguin* [en ligne]. 2019. [Consulté le 20 avril 2019]. Disponible à l'adresse : https://www.valuepenguin.com/banking/statistics-and-trends.

WALGROVE, Amanda, 2018. 5 Stats That Prove Social is the New Frontier of Customer Care. In : *Sprinklr* [en ligne]. 22 août 2018. [Consulté le 13 juin 2019]. Disponible à l'adresse : https://blog.sprinklr.com/5-stats-social-media-customer-care/.

WEALTHFRONT, 2019. Wealthfront: Build a free financial plan and automate your investments for a low cost. In : [en ligne]. 2019. [Consulté le 1 avril 2019]. Disponible à l'adresse : https://www.wealthfront.com/.

WEF, 2019. Data and the future of financial services – The European Sting - Critical News & Insights on European Politics, Economy, Foreign Affairs, Business & Technology - europeansting.com. In : [en ligne]. 11 avril 2019. [Consulté le 20 avril 2019]. Disponible à l'adresse : https://europeansting.com/2019/04/11/data-and-the-future-of-financial-services/.

WORLD ECONOMIC FORUM, 2018. *The Appropriate Use of Customer Data in Financial Services* [en ligne]. Whitepaper. Switzerland. World Economic Forum. [Consulté le 1 juin 2019]. Disponible à l'adresse : http://www3.weforum.org/docs/WP_Roadmap_Appropriate_Use_Customer_Data.pdf.

WORLD FEDERATION OF EXCHANGES, 2018a. Stocks traded, total value (% of GDP) | Data. In : *The World Bank* [en ligne]. 2018. [Consulté le 6 février 2019]. Disponible à l'adresse : https://data.worldbank.org/indicator/CM.MKT.TRAD.GD.ZS?end=2017&start=1975&view=chart.

WORLD FEDERATION OF EXCHANGES, 2018b. Stocks traded, total value (current US$) | Data. In : *The World Bank* [en ligne]. 2018. [Consulté le 6 février 2019]. Disponible à l'adresse : https://data.worldbank.org/indicator/CM.MKT.TRAD.CD?end=2017&start=1975&view=chart.

YEN, Jiun, 2019. AMEX, NYSE, NASDAQ stock histories. In : [en ligne]. avril 2019. [Consulté le 1 juin 2019]. Disponible à l'adresse : https://kaggle.com/qks1lver/amex-nyse-nasdaq-stock-histories.

ZAHARIA, Matei, CHOWDHURY, Mosharaf, FRANKLIN, Michael J, SHENKER, Scott et STOICA, Ion, 2010. Spark: Cluster Computing with Working Sets. In : . 2010. p. 7.

Annexe 1 : Financials activities of Big Tech Companies

	Alibaba	Tencent	Baidu	Google	Amazon	Facebook	Apple	Samsung	Microsoft	Vodafone	Mercado Libre
Payments	AliPay (largest mobile paymensts platform in China)	Tenpay (#2 mobile payments platform in China)	Baidu Wallet – cooperation with PayPal	Google Pay – layers over existing card network	Amazon Pay – layers over existing card network	Messenger Pay – layers over existing card network	Apple Pay – layers over existing card network	Samsung Pay – layers over existing card network	Microsoft Pay – layers over existing card network	M-Pesa (32 million active users in East Africa and India)	Mercado Pago (offered in 8 markets in Latin America)
Lending and short-term credit	MYBank (SME lending for rural areas and online merchants)	WeBank (Personal micro-loans)	Baixin Bank (financial products and small loans)	Collaboration with Lending Club	Temporary financing in Amazon Lending; direct lending to merchants	Pilot in collaboration with Clearbanc	n/a	n/a	n/a	Offered through M-Shwari mobile banking service	Mercado Crédito (small loans to retail and SME clients)
Current accounts	Offered through MYBank	Offered through WeBank	Offered through Baixin Bank	n/a	Reports of talks with banks	n/a	n/a	n/a	n/a	Offered through M-Shwari	n/a
Asset management	Yu'e Bao (world's largest MMF)	License to offer mutual funds	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	Pilots ongoing in 2018
Insurance	60% stake in Cathay Insurance China, founding stake in Zhong An Insurance	Online insurance service in life and property insurance	Joint venture with Allianz, and Hillhouse Capital announced	Insurance on Google Compare (discontinued)	Partnership with JPMorgan Chase and Berkshire Hathaway on health insurance	n/a	Cooperation with Allianz oı cyber insurance discounts	n/a	n/a	n/a	Pilots ongoing in 2018

(Financial Stability Board, 2018)

Annexe 2 : Porfolio Creation & Analysis Code (full)

```
#=================================IMPORT LIBRAIRIES=================================================
import os
import math

import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import Row
from pyspark.sql import functions as F
from pyspark.sql.types import StructType
from pyspark.sql.window import Window
from pyspark.mllib.stat import Statistics

import pandas as pd
from pandas.plotting import register_matplotlib_converters

import numpy as np

from scipy.stats import norm
from scipy.optimize import minimize
from scipy import stats

import statsmodels.api as sm

import seaborn as sns

import matplotlib.pyplot as plt
```

```
#=================================SPARK SESSION BEGIN================================================

#Create a SparkSession (the config bit is only for Windows!)
spark = SparkSession.builder.config("spark.sql.warehouse.dir", "file:///C:/temp").appName("portfolio_simu").getOrCreate()

#=================================SPARK - IMPORT PORTFOLIO TICKER DIM TABLES=========================
#Weight could be define in this table but as we are looking for the optimal allocation here after we drop the column

path = "C:/Users/huniv/TB/datasets/test_port I.csv"  # CHANGE TO YOUR DIRECTORY WHERE YOUR PORTFOLIO CSV IS STORED
#Get the raw data in DataFrame from the csv
portfolio_position = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path)
#rename columns
portfolio_position = portfolio_position.withColumnRenamed('Symbol', 'ticker')
#cast the ticker values into string
portfolio_position = portfolio_position.withColumn('ticker', portfolio_position["ticker"].cast("string"))
#drop the weight column (not required here)
portfolio_position = portfolio_position.drop('Weight')
#convert the dim table into pandas df to be able to iterate each ticker in a loop.
pd_portfolio_position = portfolio_position.select("*").toPandas()
```

```
#Check if a csv files is available for the choosed ticker
path = "C:/Users/huniv/TB/datasets/full_history/" # CHANGE TO YOUR DIRECTORY WHERE TICKER'S CSV ARE STORED
for idx, row in pd_portfolio_position.iterrows():

    curr_ticker = row['ticker']

    if os.path.exists(path + curr_ticker + '.csv'):
        continue
    else:
        print(curr_ticker)
```

```
#==================================SPARK - CREATE THE ADJ CLOSE FACTS TABLES==================================
#create a master spark dataframe which will receive all adj_close of all ticker. One col = One Price Evolution of a ticker
schema = StructType([])
price_master = spark.createDataFrame([], schema)

width = 30
height = 30

path = "C:/Users/huniv/TB/datasets/full_history/" # CHANGE TO YOUR DIRECTORY WHERE YOUR PORTFOLIO CSV IS STORED
columns_to_drop = ['date', 'volume', 'open', 'close', 'high', 'low']

#for each ticker in the dim table, get the correct csv files in a spark dataframe and transform it
for idx, row in pd_portfolio_position.iterrows():

    #get the current ticker
    curr_ticker = row['ticker']

    ##Get the raw data in DataFrame from the csv
    stock = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path + curr_ticker + ".csv")
    #Ex: adj_close => adj_close_MSFT . To differentiate each column in the master dataframe
    stock = stock.withColumnRenamed('adjclose', 'adjclose_' + curr_ticker)
    #Cast the date
    stock = stock.withColumn('myDate', F.to_date(stock.date, 'yyyy-MM-dd'))
    #Sort
    stock = stock.orderBy('myDate')
    #Drop unrequired column
    stock = stock.drop(*columns_to_drop)

    #if it's the first tiker duplicate in the master, otherwise make a left join on the date
    #NULL Values stored if the next ticker doesn't exist the day pf the row
    if idx == 0:
        price_master = stock
    else:
        price_master = price_master.join(stock, on="myDate", how="left")

#Select the begining of your investment. Preferably a date where all ticker was public yet
price_master = price_master.filter(price_master["myDate"] >= F.lit('2016-01-01'))
#convert the master spark dataframe into a pandas dataframe
pd_price_master = price_master.select("*").toPandas()

#Cast all adj_close columns into numeric values
cols=[i for i in pd_price_master.columns if i not in ["myDate"]]

for col in cols:
    pd_price_master[col] = pd.to_numeric(pd_price_master[col])

#Cast the date
pd_price_master["myDate"]= pd.to_datetime(pd_price_master["myDate"], infer_datetime_format=True)
#Set the date as the row index (Primary Key)
pd_price_master = pd_price_master.set_index('myDate')
```

```
#==================================PANDAS - CREATE THE ADJ CLOSE FACTS TABLES==================================
columns_to_drop = ['volume', 'open', 'close', 'high', 'low']

#for each ticker in the dim table, get the correct csv files in a spark dataframe and transform it
for idx, row in pd_portfolio_position.iterrows():

    #get the current ticker
    curr_ticker = row['ticker']

    data = pd.read_csv(path + curr_ticker +'.csv')
    data = data.rename(columns={'adjclose':'adjclose_' + curr_ticker})
    data.date = pd.to_datetime(data['date'])
    data = data.drop(columns_to_drop, axis=1)
    data = data.set_index("date")

    #if it's the first tiker duplicate in the master, otherwise make a left join on the date
    #NULL Values stored if the next ticker doesn't exist the day pf the row
    if idx == 0:
        master = data
    else:
        master = pd.merge(master,
                  data,
                  on='date')
```

```
#==================================SPARK - CALCULATE THE CORRELATION ==================================

df = price_master.select([c for c in price_master.columns if c != 'myDate'])
col_names = df.columns
col_names = [s.strip('adjclose_') for s in col_names]
features = df.rdd.map(lambda row: row[0:])
corr_mat=Statistics.corr(features, method="pearson")
corr_df = pd.DataFrame(corr_mat)
corr_df.index, corr_df.columns = col_names, col_names

#-------------------------VISUALIZE THE CORRELATION----------------------------------
sns.set(style="white")

#Plot figsize
fig, ax = plt.subplots(figsize=(width,height))
#Generate Color Map
colormap = sns.diverging_palette(220, 10, as_cmap=True)
# Generate a mask for the upper triangle
mask = np.zeros_like(corr_df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
#Generate Heat Map, allow annotations and place floats in map
hm = sns.heatmap(corr_df, mask=mask, cmap=colormap, annot=True, fmt=".2f")
hm.set_yticklabels(hm.get_yticklabels(), rotation=360)
#Apply xticks
plt.xticks(np.arange(len(corr_df.columns))+0.5, corr_df.columns);
#Apply yticks
plt.yticks(np.arange(len(corr_df.columns))+0.5, corr_df.columns)
#show plot
plt.show()
#==================================PANDAS - CALCULATE THE CORRELATION ==================================
corr_data = master.corr()
col_names = master.columns
col_names = [s.strip('adjclose_') for s in col_names]
corr_data.index, corr_data.columns = col_names, col_names

#-------------------------VISUALIZE THE CORRELATION----------------------------------
sns.set(style="white")

#Plot figsize
fig, ax = plt.subplots(figsize=(width,height))
#Generate Color Map
colormap = sns.diverging_palette(220, 10, as_cmap=True)
# Generate a mask for the upper triangle
mask = np.zeros_like(corr_data, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
#Generate Heat Map, allow annotations and place floats in map
hm = sns.heatmap(corr_data, mask=mask, cmap=colormap, annot=True, fmt=".2f")
hm.set_yticklabels(hm.get_yticklabels(), rotation=360)
#Apply xticks
plt.xticks(np.arange(len(corr_data.columns))+0.5, corr_data.columns);
#Apply yticks
plt.yticks(np.arange(len(corr_data.columns))+0.5, corr_data.columns)
#show plot
plt.show()
#==================================SIMULATION - FIND THE BEST ALLOCATION==================================
#Source : Fábio Neves - https://towardsdatascience.com/python-markowitz-optimization-b5e1623060f5

nb_position = len(pd_price_master.columns)

#https://ycharts.com/indicators/3_month_t_bill
risk_free_rate = 0.0218

#Create data frames withthe daily log returns for each ticker
log_ret = np.log(pd_price_master/pd_price_master.shift(1))

#---------------------- SIMULATION - INI---------------------------------------

#initialize the random fuunctions with a seed, to garanty reproductibility
np.random.seed(42)
#intialize the number of portfolio to create
num_ports = 50000
#initialise an array which will contains all ticker ponderation for each simulate portfolio
#Row = 1 portfolio
#columns = ticker ponderation
all_weights = np.zeros((num_ports, len(pd_price_master.columns)))
#initialise a 1D array which will contains the returns of each portfolio
ret_arr = np.zeros(num_ports)
#initialise a 1D array which will contains the volatility of each portfolio
vol_arr = np.zeros(num_ports)
#initialise a 1D array which will contains the sharpe ratio of each portfolio
sharpe_arr = np.zeros(num_ports)
```

```
#---------------------SIMULATION - CREATE X PORTFOLIO WITH A RANDOM ALLOCATION--------------------------------
#For each simulated portfolio
for x in range(num_ports):

    #-----------Generate random weights for each ticker-----------
    weights = np.array(np.random.random(nb_position))
    #Rebase all random weight on 100
    weights = weights/np.sum(weights)
    # Save weights in one row for one portfolio
    all_weights[x,:] = weights

    #-----------Calculate the Annual Expected return--------------
    ret_arr[x] = np.sum( (log_ret.mean() * weights * 252))

    #-----------Calculate Annual Expected volatility--------------
    #Covariance calculate the correlation between each ticker
    #This impact is ponderate with the weigth one time on first dim
    #Then on the second axis of the correlation matrix due to the double entries aspect of the cov matrix
    #One matrix times a column vector = one vectors
    #one vector times a vector = scalar
    #Square to correctly annualized the volatitly obtain -> *Square(252)
    vol_arr[x] = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov()*252, weights)))

    #-----------Calculate Sharpe Ratio----------------------------
    #return / Vol
    #no risk free rate here due to low interest rate but could be added easily
    sharpe_arr[x] = (ret_arr[x]-risk_free_rate)/vol_arr[x] #Element wise division
#---------------- SIMULATION - DISPLAY THE BEST PORTFOLIO DATA-------------------------------------------------

max_sr_ret = ret_arr[sharpe_arr.argmax()]
max_sr_vol = vol_arr[sharpe_arr.argmax()]
max_sr_sr = sharpe_arr[sharpe_arr.argmax()]
max_sharpe_allocation = pd.DataFrame(all_weights[sharpe_arr.argmax(), :],index=col_names,columns=['allocation'])
max_sharpe_allocation.allocation = [round(i*100,2)for i in max_sharpe_allocation.allocation]

min_vol_ret = ret_arr[vol_arr.argmin()]
min_vol_vol = vol_arr[vol_arr.argmin()]
min_vol_sr = sharpe_arr[vol_arr.argmin()]
min_vol_allocation = pd.DataFrame(all_weights[vol_arr.argmin(), :],index=col_names,columns=['allocation'])
min_vol_allocation.allocation = [round(i*100,2)for i in min_vol_allocation.allocation]

print("-"*80)
print("Maximum Sharpe Ratio Portfolio Allocation\n")
print("Annualised Return:", round(max_sr_ret*100,2))
print("Annualised Volatility:", round(max_sr_vol*100,2))
print("Annualised Sharpe Ratio:",round(max_sr_sr,2))
print("\n")
print(max_sharpe_allocation.T)
print("-"*80)

print("-"*80)
print("Minimum Volatility Portfolio Allocation\n")
print("Annualised Return:", round(min_vol_ret*100,2))
print("Annualised Volatility:", round(min_vol_vol*100,2))
print("Annualised Sharpe Ratio:",round(min_vol_sr,2))
print("\n")
print(min_vol_allocation.T)
print("-"*80)
```

```
#==================================OPTIMISATION - FIND THE BEST ALLOCATION=============================================
#Source : Fábio Neves - https://towardsdatascience.com/python-markowitz-optimization-b5e1623060f5

#-----------OPTIMISATION - INI---------------------------
def get_ret_vol_sr(weights):
    #Get the return, the vol, and the sharpe ration for a given set of weights
    weights = np.array(weights)
    ret = np.sum(log_ret.mean() * weights) * 252
    vol = np.sqrt(np.dot(weights.T, np.dot(log_ret.cov()*252, weights)))
    sr = (ret-risk_free_rate)/vol
    return np.array([ret, vol, sr])

def neg_sharpe(weights):
    # the number 2 is the sharpe ratio index from the get_ret_vol_sr
    return get_ret_vol_sr(weights)[2] * -1

def check_sum(weights):
    #return 0 if sum of the weights is 1
    return np.sum(weights)-1

def minimize_volatility(weights):
    return get_ret_vol_sr(weights)[1]

#-----------OPTIMISATION - APPLY MINIMIZE ALOGIRTHM----------------
cons_light = ({'type':'eq', 'fun':check_sum}) #The sum of each weight should be 1

#bounds = ((0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1),(0,1)) #Each weight could be bounds between 0 or 100%
bounds = []
for i in range(nb_position):
    bounds.append((0,1))
tuple(bounds)

#The best allocation first guess is equiponderate
first_guess = 1/nb_position
init_guess = []
for i in range(nb_position):
    init_guess.append(first_guess)

#find the portfolio with the maximum sharpe ratio
#Negative Sharpe Ratio has to be minimize to find the portfolio weights with the sharpe ratio max
#The neg_sharpe with his -1 combined with the minimize optimizazion algorithm finally find the greatest Sharpe Ratio
#because it will find the most far from zeroes as it's negative and without the -1, it will be the greatest
opt_results = minimize (neg_sharpe, init_guess, method="SLSQP", bounds=bounds, constraints=cons_light)
```

```
#-----------OPTIMISATION - DISPLAY THE BEST PORTFOLIO DATA--------
print(opt_results)

allocation = pd.DataFrame(opt_results.x,index=col_names,columns=['allocation'])
allocation.allocation = [round(i*100,2)for i in allocation.allocation]

ret_opt = round(get_ret_vol_sr(opt_results.x)[0],2)
vol_opt = round(get_ret_vol_sr(opt_results.x)[1],2)
sr_opt = round(get_ret_vol_sr(opt_results.x)[2],2)

print("-"*80)
print("Maximum Sharpe Ratio Portfolio Allocation\n")
print("Annualised Return:", ret_opt)
print("Annualised Volatility:", vol_opt)
print("Annualised Sharpe Ratio:",sr_opt)
print("\n")
print(allocation.T)
print("-"*80)
```

```
#-----------OPTIMISATION - APPLY MINIMIZE ALOGIRTHM----------------
cons_strict = ({'type':'eq', 'fun':check_sum}, {'type':'ineq', 'fun':lambda w: w-0.01}) #The sum of each weight should be 1

#find the portfolio with the maximum sharpe ratio
#Negative Sharpe Ratio has to be minimize to find the portfolio weights with the sharpe ratio max
#The neg_sharpe with his -1 combined with the minimize optimizazion algorithm finally find the greatest Sharpe Ratio
#because it will find the most far from zeroes as it's negative and without the -1, it will be the greatest
opt_results_strict = minimize (neg_sharpe, init_guess, method="SLSQP", bounds=bounds, constraints=cons_strict)

#-----------OPTIMISATION - DISPLAY THE BEST PORTFOLIO DATA--------
print(opt_results_strict)

allocation_strict = pd.DataFrame(opt_results_strict.x ,index=col_names,columns=['allocation'])
allocation_strict.allocation = [round(i*100,2)for i in allocation_strict.allocation]

ret_opt_strict = round(get_ret_vol_sr(opt_results_strict.x)[0],2)
vol_opt_strict = round(get_ret_vol_sr(opt_results_strict.x)[1],2)
sr_opt_strict = round(get_ret_vol_sr(opt_results_strict.x)[2],2)

print("-"*80)
print("Maximum Sharpe Ratio Portfolio Allocation\n")
print("Annualised Return:", ret_opt_strict)
print("Annualised Volatility:", vol_opt_strict)
print("Annualised Sharpe Ratio:",sr_opt_strict)
print("\n")
print(allocation_strict.T)
print("-"*80)
```

```
fig = plt.figure(figsize=(15,10))
fig.add_subplot(1,1,1)
plt.scatter(vol_arr, ret_arr, c=sharpe_arr, cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')
plt.scatter(max_sr_vol, max_sr_ret,c='red', s=50) # simulation max sharpe ratio
plt.scatter(min_vol_vol, min_vol_ret,c='orange', s=50) # simulation minimum vol
plt.scatter(vol_opt_strict/100, ret_opt_strict/100,c='fuchsia', s=50) #optimisation with minimum allocation
plt.scatter(vol_opt/100, ret_opt/100,c='cyan', s=50) # optmization
plt.show()
```

```
#============================MARKOVITZ EFFICIENT FRONTIER=============================================================
#Source : Fábio Neves - https://towardsdatascience.com/python-markowitz-optimization-b5e1623060f5

frontier_y = np.linspace(0,0.35,50) #0,35 max returns define with a previsualisation of simulated portfolio return
                                    #it appears that 0.30 is the max value graphically
                                    #50 returns are calculated. 50 is choosed, just to have enough point to plot the frontier
frontier_x = []

#Calculate the minimum vol (x abscisse) for 200 possible return between 0 and 0,3
for possible_return in frontier_y:
    cons = ({'type':'eq', 'fun':check_sum},
            {'type':'eq', 'fun': lambda w: get_ret_vol_sr(w)[0] - possible_return})
    result = minimize(minimize_volatility,init_guess,method='SLSQP', bounds=bounds, constraints=cons)
    frontier_x.append(result['fun']) #retrieve the sharpe ratio for the x axis

fig = plt.figure(figsize=(15,10))
fig.add_subplot(1,1,1)
plt.scatter(vol_arr, ret_arr, c=sharpe_arr, cmap='viridis')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility')
plt.ylabel('Return')
plt.scatter(max_sr_vol, max_sr_ret,c='red', s=50) # simulation max sharpe ratio
plt.scatter(min_vol_vol, min_vol_ret,c='orange', s=50) # simulation minimum vol
plt.scatter(vol_opt_strict/100, ret_opt_strict/100,c='fuchsia', s=50) #optimisation with minimum allocation
plt.scatter(vol_opt/100, ret_opt/100,c='cyan', s=50) # optmization
plt.plot(frontier_x,frontier_y, 'r--', linewidth=3)
plt.show()
```

```
#============================SIMULATE A PORTFOLIO WITH THE OPTIMALE ALLOCATION==============================

init_invest = 1000000

#-----------CALCULATE THE PORTFOLIO VALUATION EVOLUTION----------------

#Calculate the value in ($) of each position by applying the optimal ponderation to the initial investment
ini_val_pos = []
myWeights = opt_results_strict.x/100
for c in myWeights:
    ini_val_pos.append(int(init_invest * c))

#Don't erase the price df by duplicate it
portfolio_master = pd_price_master

#Get the first price S0 of each ticker in T = 0
S0 = portfolio_master.iloc[0,:]
#Calculate the number of position required to get the optimal allocation value with the S0 price
N = np.floor(ini_val_pos / S0)
#Multiply each ticker by his weight across the whole period
portfolio_master = portfolio_master.multiply(N)
#Sum each position value to get the prtfolio val
portfolio_master['value'] = portfolio_master.sum(axis = 1, skipna = True)


#-----------INDICATOR CALCULATION----------------

#Compound Annual Growth Rate CAGR
portfolio_master['date'] = portfolio_master.index.to_series()
days =  (portfolio_master.date.iloc[-1] - portfolio_master.date.iloc[0]).days
cagr = (portfolio_master['value'].iloc[-1] / portfolio_master['value'].iloc[0]) ** (252.0/days) - 1
print ('CAGR =',str(round(cagr,4)*100)+"%")
mu = cagr

#returns
portfolio_master['daily_returns'] = portfolio_master['value'].pct_change()

#Moving Average
portfolio_master['rollmean_30d'] = pd.to_numeric(portfolio_master['value'].rolling(30).mean())
portfolio_master['rollmean_15d'] = pd.to_numeric(portfolio_master['value'].rolling(15).mean())

# Compute the logarithmic returns using the Closing price
portfolio_master['daily_log_returns'] = np.log(portfolio_master['value'] / portfolio_master['value'].shift(1))

#Volatility
portfolio_master['monthly_vol'] = portfolio_master['daily_log_returns'].rolling(30).std()
vol = portfolio_master['daily_returns'].std()*math.sqrt(252)
print ('Annual Vol =',str(round(vol,4)*100)+"%")
# Compute Volatility using the pandas rolling standard deviation function
#stock['daily_vol'] = pd.rolling_std(stock['Log_Ret'], window=252) * np.sqrt(252)
portfolio_master['daily_vol'] = portfolio_master['daily_log_returns'].rolling(2).std()
```

```
%matplotlib inline

#----------PLOTTING INDICATORS--------------
register_matplotlib_converters()

#Plot Portfolio Valuation + Moving Avg
fig = plt.figure(figsize=(15, 10))
ax = fig.add_subplot(1,1,1)
ax.plot(portfolio_master.date, portfolio_master.value, label='Portfolio Valuation')
ax.plot(portfolio_master.date, portfolio_master.rollmean_30d, label='Mov. Avg 30d')
ax.plot(portfolio_master.date, portfolio_master.rollmean_15d, label='Mov. Avg 15d')
ax.set_xlabel('Date')
ax.set_ylabel('Portfolio Value')
ax.legend(loc='best')
plt.title('Portfolio Valuation')
plt.show();

%matplotlib inline

#--------------------Plot Max DrawDown-----------------------------------------------------

# Calculate the max drawdown in the past window days for each day in the series.
# Use min_periods=1 if you want to let the first 252 days data have an expanding window
Roll_Max = portfolio_master.value.rolling(252,min_periods=1).max()
Daily_Drawdown = portfolio_master.value/Roll_Max - 1.0

# Next we calculate the minimum (negative) daily drawdown in that window.
# Again, use min_periods=1 if you want to allow the expanding window
Max_Daily_Drawdown = Daily_Drawdown.rolling(252, min_periods=1).min()

#Plot the results
plt.figure(figsize=(15,10))
plt.plot(portfolio_master.date, Daily_Drawdown, label='Daily Drawdown')
plt.plot(portfolio_master.date, Max_Daily_Drawdown, label='Max Daily Drawdown')
plt.xlabel('Date')
plt.ylabel('Drawdown in %')
plt.show()

Max_Daily_Drawdown = pd.DataFrame(Max_Daily_Drawdown)
max_dd = Max_Daily_Drawdown.min()
max_dd_day = Max_Daily_Drawdown.idxmin()

print('The Maximum drawdown was {} %'.format(round(max_dd[0] * 100,2)))
print('The day when the max drawdown was reached : {}'.format(max_dd_day[0]))
%matplotlib inline

#--------------------Plot Daily Log Returns-----------------------------------------------------
fig = plt.figure(figsize=(15, 10))
ax = fig.add_subplot(1,1,1)
ax.plot(portfolio_master.date, portfolio_master.daily_log_returns, label='Daily Log Returns')
plt.axhline(y = portfolio_master.daily_log_returns.mean(), color = "red", lw = 1,linestyle = '--')

# Add labels to the plot
style = dict(size=15, color='red')
ax.text('2019-05-01', portfolio_master.daily_log_returns.mean()+0.001, '{}%'.format(round(portfolio_master.daily_log_returns
ax.set_xlabel('Date')
ax.set_ylabel('Returns')
ax.legend(loc='best')
plt.title('Portfolio Returns')
plt.show()

print('Maximum Daily returns Date {}'.format(portfolio_master.daily_log_returns.idxmax()))
print('Maximum Daily returns {} %'.format(round(portfolio_master.daily_log_returns.max()*100,2)))
print('Minimum Daily returns Date {}'.format(portfolio_master.daily_log_returns.idxmin()))
print('Minimum Daily returns {} %'.format(round(portfolio_master.daily_log_returns.min()*100,2)))
print('Average Daily returns {} %'.format(round(portfolio_master.daily_log_returns.mean()*100,2)))
print('Std Daily returns {} %'.format(round(portfolio_master.daily_log_returns.std()*100,2)))
```

```
%matplotlib inline

#----------------------Plot Daily Volatility-----------------------------------------------------
fig = plt.figure(figsize=(15, 10))
ax = fig.add_subplot(1,1,1)
ax.plot(portfolio_master.date, portfolio_master.daily_vol, label='Daily Volatility')
plt.axhline(y = portfolio_master.daily_vol.mean(), color = "red", lw = 1,linestyle = '--')

# Add labels to the plot
style = dict(size=15, color='red')
ax.text('2019-05-01', portfolio_master.daily_vol.mean()+0.001, '{}%'.format(round(portfolio_master.daily_vol.mean()*100,2)),

ax.set_xlabel('Date')
ax.set_ylabel('Vol')
ax.legend(loc='best')
plt.title('Portfolio Volatility')
plt.show()

print('Maximum Daily vol Date {}'.format(portfolio_master.daily_vol.idxmax()))
print('Maximum Daily vol {} %'.format(round(portfolio_master.daily_vol.max()*100,2)))
print('Minimum Daily vol Date {}'.format(portfolio_master.daily_vol.idxmin()))
print('Minimum Daily vol {} %'.format(round(portfolio_master.daily_vol.min()*100,2)))
print('Average Daily vol {} %'.format(round(portfolio_master.daily_vol.mean()*100,2)))
print('Std Daily vol {} %'.format(round(portfolio_master.daily_vol.std()*100,2)))
#==========================MONTE CARLO SIMULATION=============================================================
# Source : Wenqiang Feng -https://runawayhorse001.github.io/LearningApacheSpark/preface.html

#GEnerate an empty figure where all simluation will be plotted
fig = plt.figure(figsize=(15, 10))

#set up empty list to hold our ending values for each simulated price series
result = []

#Define Variables
S = portfolio_master['value'].iloc[-1] #starting stock price (i.e. last available real stock price)
T = 252 #Number of trading days

#Compute the number of days "from when during how many times". From the last day to 252 working day
days = pd.date_range(portfolio_master['date'].iloc[-1], periods= T+1,freq='B').date

#Select the number of simulation
simulation_nb = 10000

for i in range(simulation_nb):
    #create list of daily returns using random normal distribution
    #The random normal list has for parameters : mean = CAGR / T - Std = Volatility / Squre root of the 252 days - the nb of
    daily_returns=np.random.normal(mu/T,vol/math.sqrt(T),T)+1

    #set starting price and create price series generated by above random daily returns
    price_list = [S]

    for x in daily_returns:
        price_list.append(price_list[-1]*x)

    #plot data from each individual run which we will plot at the end
    plt.plot(days, price_list)

    #append the ending value of each simulated run to the empty list we created at the beginning
    result.append(price_list[-1])


plt.title('Monte Carlo Simulation of the portfolio valuation')
plt.xlabel('Days (252d)')
plt.ylabel('Value ($)')
plt.axhline(S, color='red', linestyle='dashed', linewidth=2, label='S0')
plt.legend(loc='upper left')
plt.show()
```

```
%matplotlib inline

#use numpy mean function to calculate the mean / percentile of the all simulations
result_mean = round(np.mean(result),2)
result_5p = np.percentile(result,5)
result_95p = np.percentile(result,95)
print("mean =", result_mean)
print("5% quantile =",result_5p)
print("95% quantile =",result_95p)

#Plot an histogram of all simulation
plt.figure(figsize=(20,15))
sns.distplot(result, bins=100, color='purple',hist=True, kde=False)
plt.axvline(result_5p , color='blue', linestyle='dashed', linewidth=1)
plt.axvline(result_mean, color='blue', linestyle='dashed', linewidth=1)
plt.axvline(result_95p, color='blue', linestyle='dashed', linewidth=1)
plt.axvline(S, color='red', linestyle='dashed', linewidth=1)
plt.title('Histogram of simulated Portfolio Value')
plt.xlabel('Portfolio Valuation')
plt.ylabel('Number of Simulation')
plt.show()
```

```
ini = portfolio_master['value'].iloc[-1]
gain_loss=[]
for x in result:
    gain_loss.append((x-ini)/ini)

gain_loss_mean = round(np.mean(gain_loss),2)
gain_loss_5p = np.percentile(gain_loss,5)
gain_loss_95p = np.percentile(gain_loss,95)

print("mean =", gain_loss_mean)
print("5% quantile =",gain_loss_5p)
print("95% quantile =",gain_loss_95p)

#Plot an histogram of all simulation
plt.figure(figsize=(20,15))
sns.distplot(gain_loss, bins=100, color='purple',hist=True, kde=False)
plt.axvline(gain_loss_5p , color='blue', linestyle='dashed', linewidth=1)
plt.axvline(gain_loss_mean, color='blue', linestyle='dashed', linewidth=1)
plt.axvline(gain_loss_95p, color='blue', linestyle='dashed', linewidth=1)
plt.title('Histogram of Gain/Loss simulated')
plt.xlabel('Gain / Loss')
plt.ylabel('Number of Simulation')
plt.show()
```

```
var =  np.percentile(gain_loss, .95)
var1 =  np.percentile(gain_loss, .99)
var2 =  np.percentile(gain_loss, .9999)

print("-"*80)
print("Annual VaR")
print("VaR at 95% Confidence: {}%".format(round(var*100,2)))
print("VaR at 99% Confidence: {}%".format(round(var1*100,2)))
print("VaR at 99.99% Confidence: {}%".format(round(var2*100,2)))
print("-"*80)
print("Monthly VaR")
print("VaR at 95% Confidence: {}%".format(round(var*100/np.sqrt(20),2)))
print("VaR at 99% Confidence: {}%".format(round(var1*100/np.sqrt(20),2)))
print("VaR at 99.99% Confidence: {}%".format(round(var2*100/np.sqrt(20),2)))
print("-"*80)
print("Daily VaR")
print("VaR at 95% Confidence: {}%".format(round(var*100/np.sqrt(252),2)))
print("VaR at 99% Confidence: {}%".format(round(var1*100/np.sqrt(252),2)))
print("VaR at 99.99% Confidence: {}%".format(round(var2*100/np.sqrt(252),2)))
#============================CAPM IMPLEMENTATION=================================================
#Source : Bernard Brenyah - https://medium.com/python-data/capm-analysis-calculating-stock-beta-as-a-regression-in-python-c8

path = "C:/Users/huniv/TB/datasets/full_history/SPY.csv"
#Get the raw data in DataFrame from the csv
spy = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path)
#Cast the date
spy = spy.withColumn('myDate', F.to_date(spy.date, 'yyyy-MM-dd'))
#Sort
spy = spy.orderBy('myDate')
#Select the begining of your investment. Preferably a date where all ticker was public yet
spy = spy.filter(spy["myDate"] >= F.lit('2016-01-01'))
#convert the dim table into pandas df to be able to iterate each ticker in a loop.
pd_spy = spy.select('myDate', 'adjclose').toPandas()
pd_spy['adjclose'] = pd.to_numeric(pd_spy['adjclose'])
#returns
pd_spy['daily_returns'] = pd_spy['adjclose'].pct_change()
#Cast the date
pd_spy["myDate"]= pd.to_datetime(pd_spy["myDate"], infer_datetime_format=True)
#Set the date as the row index (Primary Key)
pd_spy = pd_spy.set_index('myDate')

#Create a DataFrame
capm = {
    'SPY':pd_spy['daily_returns'],
    'Portfolio':portfolio_master['daily_returns']}

capm = pd.DataFrame(capm,columns=['SPY','Portfolio'])
capm = capm.dropna(axis=0)  # drop first missing row
%matplotlib inline

#Plot an histogram of all simulation
plt.figure(figsize=(15,10))
capm['SPY'].plot(label="SPY")
capm['Portfolio'].plot(label="Portfolio")
plt.title('Daily return - Portfolio VS SPY')
plt.xlabel('Date')
plt.ylabel('Returns')
plt.legend(loc='best')
plt.show()
```

```
# split dependent and independent variable
X = capm['SPY']
Y = capm['Portfolio']

#scipy linear regression
slope, intercept, r_value, p_value, std_err = stats.linregress(X, Y)

#Cast the date
pd_spy.index= pd.to_datetime(pd_spy.index, infer_datetime_format=True)

#Create a DataFrame
capm = {
    'SPY':pd_spy['daily_returns'],
    'Portfolio':portfolio_master['daily_returns']}

capm = pd.DataFrame(capm,columns=['SPY','Portfolio'])
capm = capm.dropna(axis=0)  # drop first missing row

# split dependent and independent variable
X = capm['SPY']
Y = capm['Portfolio']

#scipy linear regression
slope, intercept, r_value, p_value, std_err = stats.linregress(X, Y)


plt.figure(figsize=(15,10))
plt.plot(X, Y, 'o', label='Data Point')
plt.plot(X, intercept + slope*X, 'r', label='CAPM line')
plt.xlabel('SPY Daily Returns %')
plt.ylabel('Porfolio Daily Returns %')
plt.legend(loc='best')
plt.show()

print('alpha: ' + str(intercept))
print('beta: ' + str(slope))

#Always end your Spark Session
spark.stop()
```

Annexe 3 : Batch CSV Reading Comparison code (full)

```
#==================================IMPORT LIBRAIRIES=================================================
import os
import math


import findspark
findspark.init()

from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import Row
from pyspark.sql import functions as F
from pyspark.sql.types import StructType
from pyspark.sql.window import Window
from pyspark.mllib.stat import Statistics

import pandas as pd
from pandas.plotting import register_matplotlib_converters

import numpy as np

from scipy.stats import norm
from scipy.optimize import minimize

import seaborn as sns

import matplotlib.pyplot as plt
```

```
#==================================SPARK SESSION BEGIN================================================
path = "C:/Users/huniv/TB/datasets/full_history/"

#Create a SparkSession (the config bit is only for Windows!)
spark = SparkSession.builder.config("spark.sql.warehouse.dir", "file:///C:/temp").appName("batch_simu").getOrCreate()

#==================================SPARK - IMPORT ALL TICKER==================================

stock = spark.read.format("csv").option("header","true").option("mode","DROPMALFORMED").load(path + "*.csv")
```

```
#==================================SPARK COUNT=================================================
stock.count()
```

```
#==================================SPARK AGGREGATION=================================================
stock.createOrReplaceTempView("stock")
stock_grouped = spark.sql("SELECT YEAR(date) as year, \
                                  MONTH(date) as month, \
                                  SUM(volume) as volume, \
                                  SUM(open) as open, \
                                  SUM(close) as close, \
                                  SUM(high) as high, \
                                  SUM(low) as low, \
                                  SUM(adjclose) as adjclose \
                           FROM stock \
                           GROUP BY year, month")
```

```
#--------------------------------VIZUALIZE THE SPARK DF-----------------------------------------
stock_grouped.createOrReplaceTempView("stock_grouped")
spark.sql("SELECT * FROM stock_grouped LIMIT 24 ORDER BY YEAR ASC").toPandas().head(2)
```

```
#==================================PANDAS - IMPORT ALL TICKER==================================
path = "C:/Users/huniv/TB/datasets/full_history/"

files = os.listdir(path)
df = pd.concat([pd.read_csv(path + f) for f in os.listdir(path)], ignore_index = True)
```

```
#==================================PANDAS COUNT=================================================
df.count()
```

```
#==================================PANDAS AGGREGATION=================================================
#Cast the date
df["date"]= pd.to_datetime(df["date"], infer_datetime_format=True)
#Set the date as the row index (Primary Key)
df = df.set_index('date')
 #Cast the date
df["volume"] = pd.to_numeric(df["volume"])
df["year"]=df.index.year
df["month"]=df.index.month
df_grouped = df.groupby(['year', 'month']).sum()
```

```
#--------------------------------VIZUALIZE THE PANDAS DF-----------------------------------------
df_grouped.head(24)
```

```
#Always end your Spark Session
spark.stop()
```