



UNIVERSITE ABDELMALEK ESSAADI
FACULTE DES SCIENCES ET TECHNIQUES D'AL-HOCEIMA
DÉPARTEMENT D'INFORMATIQUE
Année universitaire 2023-2024



N° d'ordre : 2024/0019

RAPPORT
De Projet de Fin d'Études

Présenté en vue de l'obtention de
LICENCE SCIENCES ET TECHNIQUES
Spécialité : Ingénierie de données et développement logiciel.

Par :
BAY BAY Badr

Titre
**Système de prédiction basé sur Spark et les algorithmes du
machine learning**

Soutenu le 12 juin 2024 devant le jury compose de :

Pr. Soufi Adil

Président

Pr. ZANNOU Abderrahim

Examineur

Pr. FARISS Mourad

Encadrant

Dédicace

Je dédie ce projet de fin d'études à :

Mes très chers parents,

Pour votre amour sans faille, votre aide constante et les sacrifices que vous avez consentis pour mon éducation. Votre présence et vos encouragements ont été ma force motrice tout au long de ce parcours. Je vous suis éternellement reconnaissant et j'espère vous rendre fiers de mes réalisations.

Mes frères et sœurs,

Pour votre soutien constant, vos conseils précieux et votre amour inébranlable. Vous êtes une source d'inspiration et de motivation pour moi, et je suis reconnaissant d'avoir des liens familiaux aussi forts.

Mes amis proches,

Pour votre présence joyeuse et vos encouragements tout au long de cette aventure. Votre amitié sincère nous a aidés à surmonter les moments difficiles et à célébrer les victoires.

Mes enseignants,

Pour votre expertise, votre dévouement et votre patience dans mon apprentissage. Vos connaissances et vos conseils ont façonné mon parcours académique et professionnel. Et à tous ceux qui ont contribué de près ou de loin à la réalisation de ce projet, je vous exprime ma gratitude pour votre soutien, vos idées et votre collaboration. Votre contribution a été précieuse et a enrichi mon travail. Que cette dédicace soit le reflet de ma profonde reconnaissance envers chacun d'entre vous. Merci d'avoir fait partie de cette expérience et d'avoir joué un rôle essentiel dans ma réussite.

Remerciement

Je débute en exprimant ma profonde gratitude envers le tout-puissant, qui m'a accordé la volonté et le courage de mener à bien mon parcours universitaire malgré les difficultés rencontrées.

Mes sincères remerciements vont à mon précieux encadrant, monsieur Mourad Fariss, dont l'encadrement, le soutien, la rigueur, le suivi et les conseils tout au long de cette période ont été inestimables. Après lui, je souhaite également remercier les membres du jury, notamment les professeurs Soufi Adil et Zannou Abderrahim, qui ont consacré leur précieux temps à l'étude de mon mémoire.

J'exprime également ma reconnaissance et ma gratitude envers mes chers professeurs et enseignants de la Faculté des sciences et techniques d'Al-Hoceima, ainsi qu'envers les étudiants et le personnel avec lesquels j'ai partagé mon parcours universitaire.

Je tiens à exprimer toute ma gratitude envers mes parents bien-aimés, dont le soutien et l'encouragement indéfectibles ont été essentiels tout au long de mes années d'études et sans lesquels je n'aurais jamais pu réussir.

J'adresse mes remerciements chaleureux à mes frères et sœurs pour leur présence, leur soutien moral et leurs encouragements constants.

À toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet, que ce soit par leur aide, leurs conseils ou leur présence, je tiens à exprimer ma profonde reconnaissance envers eux.

Je tiens à remercier sincèrement chacun d'entre vous. Vos efforts, votre soutien et votre présence ont été d'une valeur inestimable pour moi.

Résumé

Ce rapport exhaustif détaille la conception, le développement et l'évaluation d'un système de prédiction sophistiqué utilisant Apache Spark et divers algorithmes de Machine Learning, visant à traiter et analyser de grandes quantités de données de manière efficace pour améliorer la prise de décision dans divers contextes professionnels. Le projet se focalise sur l'intégration de la plateforme Spark, choisie pour sa capacité à effectuer des traitements de données en mémoire, offrant ainsi un avantage significatif par rapport aux approches basées sur disque comme celles utilisées par Hadoop, répondant à un besoin de performances accrues et de scalabilité dans la gestion de Big Data. Les objectifs principaux sont d'améliorer la rapidité et la précision des décisions grâce à des modèles de prédiction fiables et d'optimiser le traitement des données volumineuses. La méthodologie adoptée inclut une phase de planification rigoureuse, le développement d'une architecture système adaptée et une série d'évaluations pour tester la performance des modèles de Machine Learning utilisés, avec une attention particulière portée à la flexibilité et à l'adaptabilité du système. En conclusion, ce projet illustre l'utilisation innovante de technologies avancées pour le traitement et l'analyse de données, posant les bases pour des améliorations futures et l'exploration de nouvelles applications potentielles dans divers secteurs industriels, susceptibles de transformer les pratiques décisionnelles en rendant les processus non seulement plus rapides mais aussi plus intelligents grâce à une compréhension plus profonde et plus analytique des données disponibles.

Mot clés : Système de prédiction, Apache Spark, Machine Learning, Big Data, Analyse des données.

Liste des abréviations

Abréviation	Désignation
ML	Machine Learning
AI	Artificial Intelligence
VM	Virtual Machine
HDFS	Hadoop Distributed File System
RDD	Resilient Distributed Data
DAG	Directed Acyclic Graph
AWS	Amazon Web Services
DBFS	Databricks File System
RAM	Random Access Memory
LR	Linear Regression
DT	Decision Tree
RF	Random Forest
GBTs	Gradient Boosting Trees
XGBOOST	Extreme Gradient Boosting

Liste des figures

Figure 1. Apache Spark Logo	4
Figure 2. Aperçu de l'architecture Spark.....	5
Figure 3. Spark Eco-System.	5
Figure 4. Représentation du RDD.....	7
Figure 5. Fonctionnement d'un RDD.	7
Figure 6. Exemple pour les fonctions Map/Filter.	7
Figure 7. Exemple des fonctions de Transformation/Action.	8
Figure 8. Architecture de Spark.	8
Figure 9. Architecture de HDFS.	10
Figure 11. Les Algorithmes du Machine Learning.....	14
Figure 12. Régression linéaire. Ajustement affine.....	15
Figure 13. Structure d'un arbre de décision.....	16
Figure 14. Gradient Boosting Algorithm.	17
Figure 15. Le fonctionnement du XGBOOST.....	18
Figure 16. Architecture proposée.....	20
Figure 17. Databricks Workspace.....	24
Figure 18. Création d'une session Spark.	25
Figure 19. Le processus du chargement des données dans DBFS.	26
Figure 20. L'importation des fichiers CSV dans Databricks notebook.	26
Figure 21. Dataset choisi.....	27
Figure 22. Prétraitement des données.	28
Figure 23. Taille du dataset après la phase du prétraitement.....	28
Figure 24. DAG exemple de la transformations Union.	29
Figure 25. Extraction des caractéristiques temporelles.....	29
Figure 26. Nouvelle schéma du Data Frame.....	29
Figure 27. Assemblage des caractéristiques	30
Figure 28. La nouvelle colonne des caractéristiques	30
Figure 39. Standardisation des caractéristiques.	30
Figure 30. Séparation des données en ensembles d'entraînement et de test.	32
Figure 31. Séparation des données en ensembles d'entraînement et de test.	32
Figure 32. Les métriques d'évaluation.....	33
Figure 33. L'implémentation du modèle XGBOOST.....	33
Figure 34. Le prix du BTC au fil du temps.....	34

Figure 35. Le prix du BTC au fil du temps.....	34
Figure 36. Matrice de corrélation.....	35
Figure 37. Le prix du BTC au fil du temps.....	36
Figure 38. Les graphiques des valeur réelles et les valeurs prédites.....	37
Figure 39. Les diagrammes à barres de chaque métrique d'évaluation.	38
Figure 40. Configuration du cluster.	38
Figure 41. Les graphiques des valeur réelles et les valeurs prédites des 5 modèles.	40
Figure 42. Résultats d'une machine Locale.	42

Liste des Tableaux

Table 1. Comparaison entre Spark RDD et Hadoop MapReduce	12
Table 2. Comparaison du temp d'exécution des algorithmes, cluster vs machine locale.....	42
Table 3. Comparaison performances des algorithmes, cluster vs machine locale.....	43

Sommaire

Introduction générale	1
Chapitre I : Fondements théoriques	2
1 Introduction	3
2 Qu'est-ce qu'un Système de Prédiction ?	3
2.1 Définition et importance	3
2.2 Fonctionnement général	3
2.3 Défis technologiques et solutions dans l'application des systèmes prédictifs	4
3 Introduction à Apache Spark	4
3.1 Pourquoi Spark et n'est pas Hadoop pour ce projet ?	4
3.1.1 Apache Spark	4
3.1.1.1 Spark Eco-System	5
3.1.1.2 RDD (Resilient Distributed Data) :	6
3.1.1.2.1 Les transformations	7
3.1.1.2.2 Les actions	8
3.1.1.3 Fonctionnement de l'architecture Spark	8
3.1.2 Hadoop	9
3.1.2.1 HDFS	9
3.1.2.2 MapReduce	10
3.1.3 Avantages du traitement en mémoire de Spark	11
3.1.3.1 Rapidité du traitement en mémoire	11
3.1.3.2 Réduction des temps de latence	11
3.1.3.3 Amélioration de la scalabilité	12
3.1.4 Comparaison entre Spark RDD et Hadoop MapReduce	12
4 Principes de Machine Learning	12
4.1 Apprentissage supervisé versus non-supervisé	13
4.2 Les Algorithmes de la bibliothèque d'apprentissage automatique Spark (MLlib)	15
4.2.1 Régression linéaire	15
4.2.1.1 Régression linéaire simple	15
4.2.1.2 Régression linéaire multiple	16
4.2.2 L'arbre de décision et la forêt aléatoire	16
4.2.2.1 Arbre de décision	16
4.2.2.2 Forêt aléatoire	16
4.2.3 Gradient-Boosted Trees (GBTs) :	16
4.2.4 XGBOOST	18
Chapitre II : Développement du système de prédiction : Architecture et implémentation	19
1 Introduction	20
2 Architecture du système	20
2.1 Vue d'ensemble de l'architecture	20

2.1.1	Les composantes de l'architecture	21
2.1.2	Interaction entre les composants	22
2.2	Configuration initiale du cluster Spark sur DataBricks	23
3	Implémentation.....	25
3.1	La création du Maitre et session Spark	25
3.2	Chargement des données.....	25
3.3	Présentation des données	26
3.4	Prétraitement des données.....	28
3.5	Transformation des données	29
3.5.1	Extraction des caractéristiques temporelles.....	29
3.5.2	Assemblage des caractéristiques	30
3.5.3	La standardisation des caractéristiques	30
Chapitre III : Modélisation et Résultats.....		31
3.6	Modélisation.....	32
3.6.1	Les algorithmes LR, RF, DT, GBTs.	32
3.6.1.1	Séparation des données en ensembles d'entraînement et de test et choix des modèles	32
3.6.1.2	Entraînement du modèle	32
3.6.1.3	Évaluation du modèle.....	32
3.6.2	L'algorithme XGBOOST.....	33
3.7	Visualisation et rapport.....	34
3.7.1	Visualisation des données.....	34
3.7.1.1	Graphiques de distribution.....	34
3.7.1.2	Matrice de corrélation.....	35
3.7.1.3	Nuages de points.....	36
3.7.2	Visualisation des résultats de la modélisation :	36
3.7.2.1	Graphiques des performances des modèles	37
3.7.2.2	Comparaison des métriques.....	37
4	Mise en œuvre	38
4.1	Environnement de développement	38
4.2	Défis techniques rencontrés.....	39
4.3	Solutions adoptées	39
5	Résultats.....	40
5.1	Analyse des résultats.....	40
5.1.1	Analyse des performances	40
5.1.2	Analyse des erreurs.....	41
5.2	Comparaison avec d'autres approches	42
5.2.1	Comparaison des temps d'exécution	42
5.2.2	Comparaison des performances des algorithmes.....	43
5.2.3	Discussion des avantages et des inconvénients.....	44
Conclusion et perspectives		45
Bibliographie		46
Webographie.....		47

Introduction générale

Dans un monde où la donnée est de plus en plus importante, la capacité à analyser et anticiper des tendances à partir de grandes quantités de données s'avère être un avantage concurrentiel essentiel pour les entreprises et les organisations. Dans cette situation globale, où il est essentiel de gérer et d'exploiter de manière efficace les données, ce rapport offre un cadre clair et détaillé pour la création et l'évaluation d'un système de prédiction avancé qui utilise Apache Spark et les algorithmes de Machine Learning.

Ce projet a pour objectif de relever le défi majeur de la lenteur et de l'inefficacité des systèmes traditionnels de traitement des données, surtout des situations où le volume et la rapidité des données nécessitent des réponses quasi instantanées. Les objectifs sont donc clairement définis : créer un système capable de traiter rapidement de grandes quantités de données avec une précision de prédiction élevée, optimiser les processus de prise de décision et diminuer les dépenses liées au traitement des données. Le choix de la solution est basé sur l'emploi de Spark, connu pour sa capacité à traiter efficacement les données en mémoire, et une sélection minutieuse d'algorithmes de Machine Learning adaptés aux divers types de données et aux prédictions nécessaires.

Pour atteindre ces objectifs, la procédure suivie comprend la conception de l'architecture du système, son développement, sa mise en œuvre et une évaluation approfondie des performances. Le rapport est structuré de manière à faciliter la compréhension et l'application des informations qu'il contient.

Le premier chapitre, "Fondements Théoriques", débutera par une introduction à Apache Spark, expliquant pourquoi Spark est préféré à Hadoop. Cette section abordera également les principes fondamentaux du Machine Learning, distinguant l'apprentissage supervisé de l'apprentissage non supervisé et détaillant les algorithmes les plus couramment utilisés dans le domaine.

Le deuxième chapitre, "Développement du Système de Prédiction : Architecture et Implémentation", fournira une vue détaillée de l'architecture du système, des composants clés et offrira une vue d'ensemble des différents éléments impliqués.

Le troisième chapitre, "Modélisation et Résultats", analysera la mise en œuvre concrète du projet, évaluant les performances du modèle et les comparant à d'autres approches existantes.

Ainsi, Ce rapport a pour objectif de fournir non seulement une analyse technique de l'utilisation de technologies avancées pour le traitement de données massives, mais aussi un guide pratique pour leur mise en œuvre efficace au sein des entreprises.

Chapitre I : Fondements théoriques

1 Introduction

Dans ce premier chapitre, nous plongeons au cœur des bases théoriques qui sous-tendent notre projet de système de prédiction basé sur Spark et les algorithmes de machine learning. Nous explorerons les concepts essentiels nécessaires à la compréhension de notre démarche, en nous penchant sur la nature même des systèmes de prédiction, l'importance de leur application et les défis technologiques qu'ils soulèvent. Nous examinerons également de près Apache Spark, en nous intéressant à ses caractéristiques uniques et à ses avantages par rapport à d'autres solutions telles que Hadoop. Enfin, nous aborderons les principes fondamentaux du Machine Learning, jetant ainsi les bases nécessaires à la compréhension des algorithmes que nous mettrons en œuvre dans notre projet. Ce chapitre constitue ainsi une introduction essentielle pour appréhender les aspects théoriques qui guident notre démarche et jetteront les bases de nos développements ultérieurs.

2 Qu'est-ce qu'un Système de Prédiction ?

2.1 Définition et importance

Un système de prédiction utilise des méthodes statistiques et de Machine Learning pour analyser des données historiques et actuelles afin de faire des prédictions sur des événements ou des comportements futurs. Cette pratique est souvent appelée analyse prédictive.

L'analyse prédictive [1] consiste à utiliser des données pour prédire les résultats futurs. Ce processus utilise l'analyse des données, le Machine Learning, l'intelligence artificielle et les modèles statistiques pour identifier des tendances susceptibles de prédire les comportements futurs. Les organisations peuvent utiliser des données historiques et actuelles pour prévoir de manière fiable et précise les tendances et les comportements à quelques secondes, jours ou années dans le futur.

2.2 Fonctionnement général

Les étapes de base du processus de création des Framework d'analyse prédictive sont les suivantes :

- **Définir le problème** : une prédiction commence par une bonne problématique et un ensemble d'exigences. Par exemple, un modèle d'analyse prédictive peut-il détecter une fraude ? Déterminer les niveaux d'inventaire optimaux pour la période des fêtes de fin d'année ? Identifier les niveaux d'inondations potentiels en cas d'intempéries ? Un problème à résoudre clairement établi aidera à déterminer la méthode d'analyse prédictive à utiliser.
- **Collecter et organiser les données** : une entreprise peut avoir des décennies de données à exploiter, ou un afflux continu de données provenant des interactions client. Avant de pouvoir développer des modèles d'analyse prédictive, vous devez identifier les flux de données, puis organiser les ensembles de données dans un dépôt, par exemple un entrepôt de données.
- **Prétraiter les données** : pour préparer les données qui seront utilisées dans les modèles d'analyse prédictive, elles doivent auparavant être nettoyées en vue de supprimer les anomalies, les points de

données manquants ou les valeurs aberrantes qui peuvent être le résultat d'erreurs de saisie ou de mesure.

- **Développer des modèles prédictifs** : les data scientists disposent de divers outils et techniques pour développer des modèles prédictifs, en fonction du problème à résoudre et de la nature de l'ensemble de données. Le Machine Learning, les modèles de régression et les arbres de décision comptent parmi les types de modèles prédictifs les plus courants.
- **Valider et déployer les résultats** : vérifiez la justesse du modèle et ajustez-le en conséquence. Après obtention de résultats acceptables, mettez-les à la disposition des data Dashboard via une application, un site web ou un tableau de bord de données.

2.3 Défis technologiques et solutions dans l'application des systèmes prédictifs

Les défis liés à l'analyse prédictive dans le contexte du Big Data comprennent la gestion de volumes massifs de données, la complexité et l'hétérogénéité des données, ainsi que les exigences de traitement en temps réel. Pour répondre à ces défis, des Framework avancés tels qu'Apache Hadoop et Apache Spark ont été développés. Hadoop permet une manipulation et une analyse distribuées grâce à son système de fichiers distribué (HDFS), optimisé pour stocker des données sur plusieurs machines, ce qui augmente la redondance et la résilience tout en facilitant le traitement de grandes quantités de données. En complément, Spark a été développé dans le but de surmonter certaines limitations de Hadoop, telles que la vitesse de traitement des données. Spark offre des calculs beaucoup plus rapides grâce à sa capacité de traitement en mémoire, ce qui le rend particulièrement adapté aux applications qui exigent des analyses en temps réel et des réponses rapides, répondant ainsi de manière efficace aux exigences modernes de l'analyse prédictive dans des environnements à grande échelle.

3 Introduction à Apache Spark

3.1 Pourquoi Spark et n'est pas Hadoop pour ce projet ?

3.1.1 Apache Spark



Figure 1. Apache Spark Logo

Spark (ou Apache Spark) [2] est un Framework open source de calcul distribué **in-memory** pour le traitement et l'analyse de données massives. Il s'agit d'un ensemble d'outils structurés selon une architecture définie (figure 2). Ces composants font de Spark une plate-forme unificatrice riche en fonctionnalités : elle peut être utilisée pour de nombreuses tâches qui devaient auparavant être accomplies avec plusieurs Framework différents.

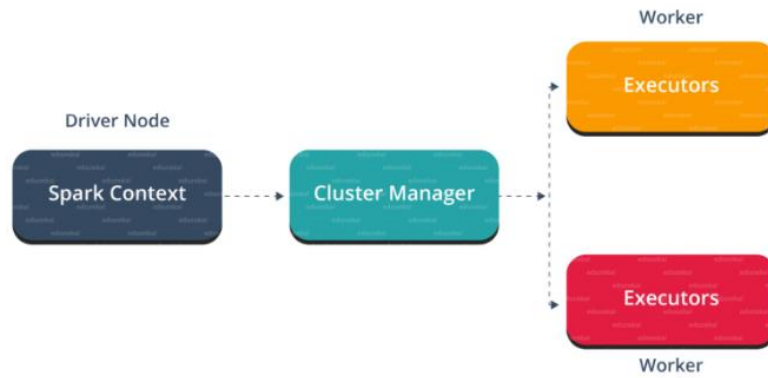


Figure 2. Aperçu de l'architecture Spark.

Apache Spark permet de traiter rapidement de très grandes quantités de données, et il est également capable de répartir des tâches de traitement de données sur plusieurs ordinateurs, soit seul, soit en utilisant d'autres outils informatiques distribués. Ces deux caractéristiques jouent un rôle crucial dans les concepts de Big Data et d'apprentissage automatique, qui requièrent une puissance de calcul considérable pour explorer les vastes collections de données.

3.1.1.1 Spark Eco-System

Spark peut être implémenté avec différents langages de programmation fonctionnels Scala (qui s'exécute dans Java VM), java, python et R. [2]

Il inclut également diverses bibliothèques pour soutenir le développement d'applications de Machine Learning (MLlib), le traitement de flux (Spark Streaming) et le traitement graphique (GraphX). Apache Spark se compose de Spark Core qui forme le cœur d'Apache Spark : c'est lui qui se charge de la transmission des tâches distribuées, de planifier et de gérer les fonctionnalités E/S. Le moteur Spark Core utilise le concept de RDD (Resilient Distributed Dataset) comme type de données de base.



Figure 3. Spark Eco-System.

- **Spark Streaming**

Spark streaming est le composant de Spark qui est utilisé pour traiter les données en continu et en temps réel. Il s'agit donc d'un complément utile à l'API de base de Spark. Il permet un traitement à haut débit et tolérant aux pannes des flux de données en direct.

- **Spark SQL**

Spark SQL est un nouveau module de Spark qui intègre le traitement relationnel à l'API de programmation fonctionnelle de Spark. Il permet d'interroger les données soit via SQL, soit via le langage de requête Hive. Pour ceux d'entre vous qui sont familiers avec les SGBDR, Spark SQL sera une transition facile à partir de vos outils précédents où vous pouvez repousser les limites du traitement traditionnel des données relationnelles.

- **GraphX**

Graphx est l'API Spark pour les graphes et les calculs parallèles aux graphes. Ainsi, il étend le Spark RDD avec un Resilient Distributed Property Graph (graphe de propriétés résilientes distribuées). À un niveau élevé, GraphX étend l'abstraction Spark RDD en introduisant le Resilient Distributed Property Graph (un multigraphe dirigé avec des propriétés attachées à chaque sommet et à chaque arête).

- **MLlib** (Machine Learning Library)

MLlib est utilisée pour effectuer de l'apprentissage automatique dans Apache Spark.

- **SparkR**

Il s'agit d'un package R qui fournit une implémentation de cadre de données distribué. Il prend également en charge des opérations telles que la sélection, le filtrage et l'agrégation, mais sur de grands ensembles de données.

3.1.1.2 **RDD (Resilient Distributed Data) :**

Les RDD sont une collection de données immuables et organisées sur lesquelles des opérations peuvent être réalisées simultanément. À la différence des Data frames, où chaque enregistrement est une ligne structurée comprenant des champs avec un schéma connu, dans les RDD, les enregistrements ne sont que des objets Java, Scala ou Python sélectionnés par le programmeur. [2]

RDD signifie :

- **Resilient** : Tolérance aux pannes et la capacité à reconstruire les données en cas de défaillance.
- **Distributed** : Distribution des données entre les différents nœuds d'un cluster.
- **Dataset** : Collection de données partitionnées avec des valeurs.

Les données d'un RDD sont divisées en morceaux sur la base d'une clé. Les RDD sont très résistants, c'est-à-dire qu'ils sont capables de se remettre rapidement d'un problème, car les mêmes blocs de données sont répliqués sur plusieurs nœuds d'exécution. Ainsi, même si un nœud d'exécution tombe en panne, un autre traitera toujours les données. Cela vous permet d'effectuer très rapidement vos calculs fonctionnels sur votre ensemble de données en exploitant la puissance de plusieurs nœuds.

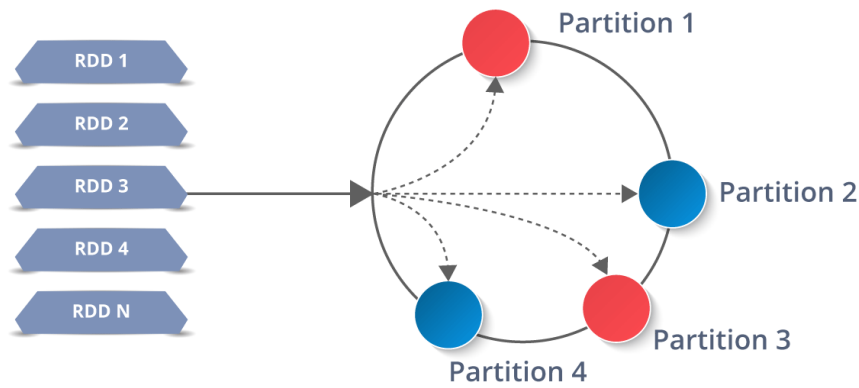


Figure 4. Représentation du RDD.

Les données d'entrée qui forment un RDD sont divisées en parties et réparties sur tous les nœuds du cluster Spark, chaque nœud effectuant ensuite un calcul en parallèle. Il est possible de réaliser une série d'opérations parallèles sur les RDD en utilisant l'API Spark Core. On peut classer ces opérations en deux catégories distinctes : les **Transformations** et les **actions**.

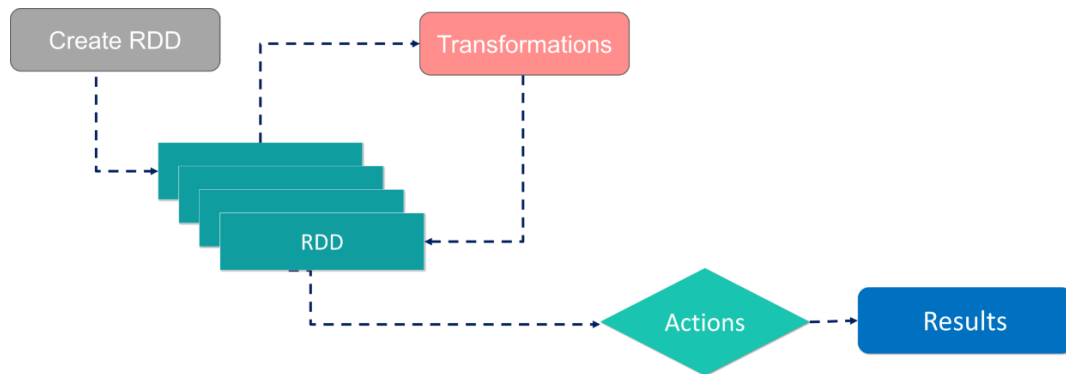


Figure 5. Fonctionnement d'un RDD.

3.1.1.2.1 Les transformations

Une **transformation** Spark est une fonction qui produit de nouveaux RDD à partir des RDD existants. Elle prend un RDD en entrée et produit un ou plusieurs RDD en sortie. A chaque fois qu'une transformation est appliquée, elle crée de nouveaux RDD. Ainsi, les RDD en entrée ne peuvent pas être modifiés puisque les RDD sont immuables par nature.

- L'application de la transformation a permis de construire une lignée de RDD, avec l'ensemble des RDD parents du ou des RDD finaux. Le lignage RDD est également connu sous le nom de graphe d'opérateur RDD ou de graphe de dépendance RDD. Il s'agit d'un plan d'exécution logique, c'est-à-dire d'un graphe acyclique dirigé (**DAG**) de l'ensemble des RDD parents du RDD. [2]
- Les transformations sont paresseuses par nature (calculées à la demande), c'est-à-dire qu'elles sont exécutées lorsque nous appelons une action. Elles ne sont pas exécutées immédiatement.[2]

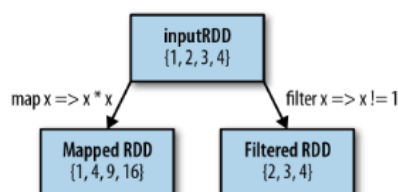


Figure 6. Exemple pour les fonctions Map/Filter.

3.1.1.2.2 Les actions

Les transformations créent des RDD les uns à partir des autres, mais lorsque nous voulons travailler avec le jeu de données actuel, c'est à ce moment-là qu'une **action** est effectuée. Lorsque l'action est déclenchée après le résultat, un nouveau RDD n'est pas formé comme dans le cas d'une transformation. **Les actions** sont donc des opérations Spark RDD qui donnent des valeurs non RDD. Les valeurs de l'action sont stockées dans les pilotes ou dans le système de stockage externe. Cela met en mouvement la paresse du RDD. [2]

- Une action est l'un des moyens d'envoyer des données de l'exécuter au driver.

Le tableau ci-dessous présente quelques-unes des fonctions d'action/transformation :

Transformations				
map	join	union	distinct	repartition
mapPartitions	flatMap	intersection	pipe	coalesce
cartesian	cogroup	filter	sample	
sortByKey	groupByKey	reduceByKey	aggregateByKey	
mapPartitionsWithIndex		repartitionAndSortWithinPartitions		
Actions				
reduce	take	collect	takeSample	count
takeOrdered	countByKey	first	foreach	saveAsTextFile
saveAsSequenceFile		saveAsObjectFile		

Figure 7. Exemple des fonctions de Transformation/Action.

3.1.1.3 Fonctionnement de l'architecture Spark

Nous avons déjà vu l'aperçu de l'architecture de base d'Apache Spark (figure 3), nous allons maintenant nous plonger plus profondément dans son fonctionnement.

Dans le nœud principal (**Master Node**), il y a le programme pilote (**Driver Program**) qui pilote l'application. Le code que vous écrivez se comporte comme un programme pilote ou, si vous utilisez l'interpréteur de commandes (Shell) interactif, ce dernier agit comme le programme pilote.

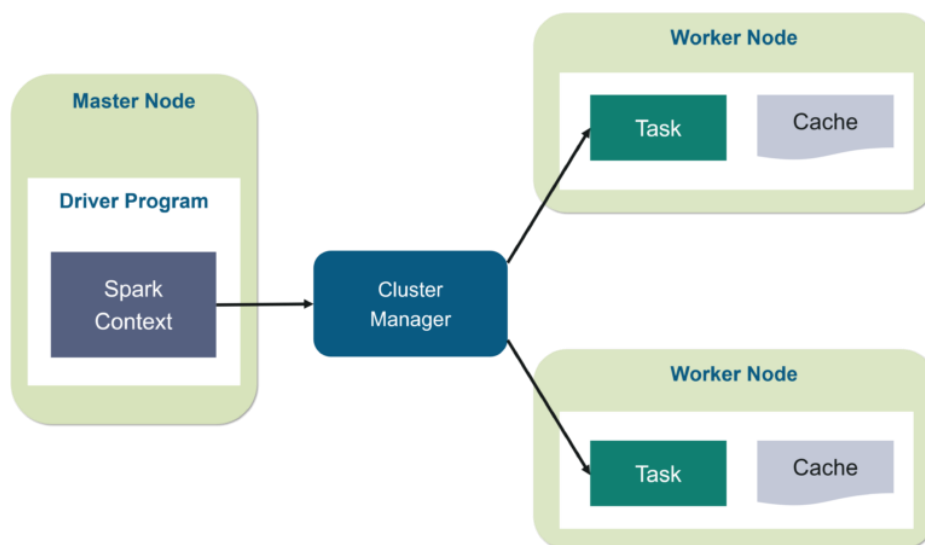


Figure 8. Architecture de Spark.

- Dans le programme du pilote, la première chose à faire est de créer un contexte Spark (**Spark Context**). Supposons que le contexte Spark est une passerelle vers toutes les fonctionnalités Spark. Il est similaire à la connexion à votre base de données. Toute commande exécutée dans votre base de données passe par la connexion à la base de données. De même, tout ce que vous faites sur Spark passe par le contexte Spark.
- Le contexte Spark travaille avec le gestionnaire de cluster (**Cluster Manager**) pour gérer les différents travaux. Le programme pilote et le contexte Spark s'occupent de l'exécution des tâches (**Tasks**) au sein du cluster. Un job est divisé en plusieurs tâches qui sont distribuées sur le nœud de travail (**Worker Node**). Chaque fois qu'un RDD est créé dans le contexte Spark, il peut être distribué sur différents nœuds et y être mis en cache.
- Les nœuds de travail sont les nœuds esclaves dont le travail consiste essentiellement à exécuter les tâches. Ces tâches sont ensuite exécutées sur les RDD partitionnés dans le nœud travailleur et renvoient le résultat au contexte Spark.
- Le contexte Spark prend le travail, le divise en tâches et les distribue aux nœuds de travail. Ces tâches travaillent sur les RDD partitionnés, effectuent des opérations, collectent les résultats et retournent au contexte Spark principal.
- Si vous augmentez le nombre de travailleurs, vous pouvez diviser les tâches en plusieurs partitions et les exécuter en parallèle sur plusieurs systèmes. Ce sera beaucoup plus rapide. Avec cette augmentation, la taille de la mémoire augmentera également et vous pouvez mettre en cache les travaux pour les exécuter plus rapidement.

3.1.2 Hadoop

Apache Hadoop [3] est un Framework qui permet le traitement distribué de grands ensembles de données sur des grappes d'ordinateurs à l'aide de modèles de programmation simples. Elle est conçue pour passer d'un simple serveur à des milliers de machines, chacune offrant des capacités de calcul et de stockage locales. Plutôt que de s'appuyer sur le matériel pour assurer une haute disponibilité, la bibliothèque elle-même est conçue pour détecter et gérer les défaillances au niveau de la couche applicative, offrant ainsi un service hautement disponible au-dessus d'une grappe d'ordinateurs, dont chacun peut être sujet à des défaillances.

- HDFS est un élément clé de nombreux systèmes Hadoop, car il permet de gérer les données volumineuses et de prendre en charge l'analyse des données volumineuses.

3.1.2.1 HDFS

HDFS est l'acronyme de Hadoop Distributed File System (système de fichiers distribués Hadoop). HDFS fonctionne comme un système de fichiers distribués conçu pour fonctionner sur du matériel de base.

- HDFS est tolérant aux pannes et conçu pour être déployé sur du matériel de base à faible coût. HDFS fournit un accès à haut débit aux données d'application et convient aux applications qui ont de grands ensembles de données et permet un accès en continu aux données du système de fichiers dans Apache Hadoop.

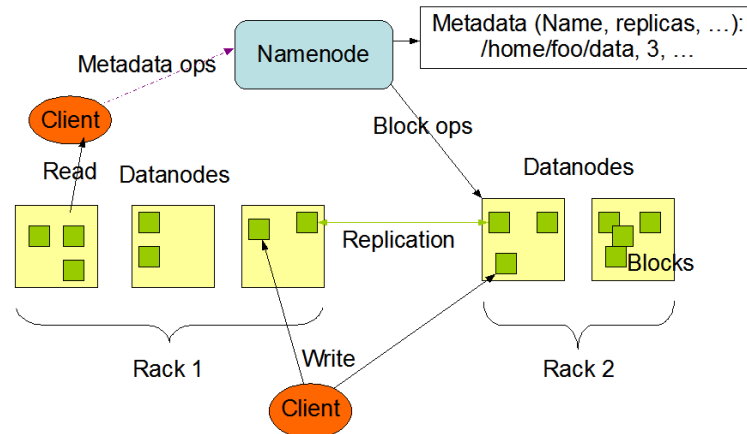


Figure 9. Architecture de HDFS.

HDFS a une architecture maître/esclave. Un cluster HDFS se compose d'un NameNode unique, un serveur maître qui gère l'espace de noms du système de fichiers et régule l'accès aux fichiers par les clients. En outre, il existe un certain nombre de DataNodes, généralement un par nœud dans le cluster, qui gèrent le stockage attaché aux nœuds sur lesquels ils s'exécutent. HDFS expose un espace de noms de système de fichiers et permet aux données utilisateur d'être stockées dans des fichiers. En interne, un fichier est divisé en un ou plusieurs blocs et ces blocs sont stockés dans un ensemble de DataNodes. Le NameNode exécute les opérations de l'espace de noms du système de fichiers, telles que l'ouverture, la fermeture et le renommage des fichiers et des répertoires. Il détermine également la correspondance entre les blocs et les DataNodes. Les DataNodes sont chargés de répondre aux demandes de lecture et d'écriture des clients du système de fichiers. Les DataNodes effectuent également la création, la suppression et la réplication de blocs sur instruction du NameNode.[3]

3.1.2.2 MapReduce

MapReduce est une technique de traitement et un modèle de programme pour le calcul distribué basé sur Java. L'algorithme MapReduce comprend deux tâches importantes, à savoir Map et Reduce. Map prend un ensemble de données et le convertit en un autre ensemble de données, où les éléments individuels sont décomposés en tuples (paires clé/valeur). Deuxièmement, la tâche de réduction, qui prend la sortie d'une carte comme entrée et combine ces tuples de données en un ensemble plus petit de tuples. Comme l'indique la séquence du nom MapReduce, la tâche de réduction est toujours exécutée après la tâche de mappage.[3]

Le paradigme MapReduce consiste à envoyer l'ordinateur là où sont stockées les données. Il se divise en trois étapes : le mappage, le mélange et la réduction.

- L'étape de mappage consiste à traiter les données d'entrée et à les diviser en morceaux plus petits.

- L'étape de réduction combine le mélange et la réduction des données provenant du mappage, créant ainsi une nouvelle sortie stockée dans le système de fichiers Hadoop (HDFS).

Lorsqu'un travail MapReduce est effectué, les tâches Map et Reduce sont envoyées aux serveurs appropriés du cluster, et le framework gère le transfert de données, la vérification de l'achèvement des tâches et la copie des données entre les nœuds du cluster. La plupart des calculs se font sur les nœuds, les données étant stockées sur des disques locaux, réduisant ainsi le trafic réseau. Une fois les tâches terminées, le cluster collecte et réduit les données pour obtenir un résultat, qui est renvoyé au serveur Hadoop.

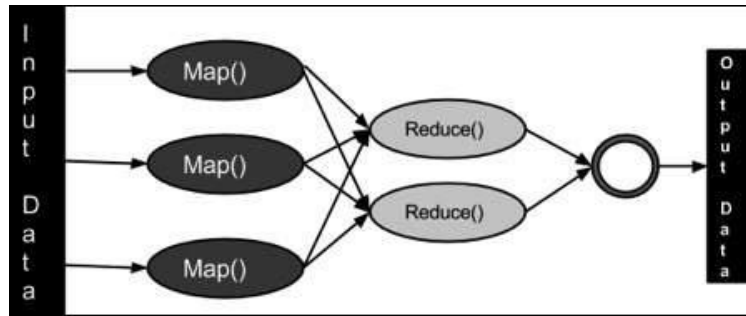


Figure 10. Exemple de fonctionnement MapReduce.

3.1.3 Avantages du traitement en mémoire de Spark

3.1.3.1 Rapidité du traitement en mémoire

Le traitement en mémoire de Spark permet des opérations plus rapides en minimisant les accès coûteux au disque, ce qui réduit considérablement les temps de latence. Les systèmes traditionnels basés sur le disque, comme Hadoop, s'appuient fortement sur la lecture et l'écriture de données sur le disque, ce qui ralentit les performances en raison du délai inhérent aux opérations d'E/S (Entrée/Sortie) sur disque. En revanche, Spark conserve les données en mémoire dans la mesure du possible, ce qui lui permet d'accéder aux données et de les manipuler beaucoup plus rapidement. Cette approche exploite la nature rapide de la RAM, accélérant ainsi les tâches de traitement des données et améliorant l'efficacité globale du système. En évitant les accès fréquents au disque, Spark peut exécuter des algorithmes itératifs et des analyses de données en temps réel beaucoup plus rapidement, ce qui permet d'obtenir des informations et des réponses plus rapides.

3.1.3.2 Réduction des temps de latence

La réduction de la latence avec Spark améliore considérablement la réactivité des systèmes, en particulier pour les applications en temps réel. En traitant les données en mémoire, Spark minimise les délais liés à la lecture et à l'écriture sur disque, ce qui permet un accès et une manipulation des données beaucoup plus rapides. Cette réduction de la latence permet à Spark de traiter rapidement de grandes quantités de données ainsi que des calculs complexes, ce qui permet aux applications de répondre presque instantanément aux requêtes des utilisateurs ou aux événements en temps réel. Dans des scénarios tels que les systèmes de recommandation en ligne, la détection de fraudes ou l'analyse interactive des données, des temps de réponse rapides sont essentiels. La capacité de Spark à fournir un traitement quasi-instantané

permet à ces applications de fournir des résultats pertinents en temps voulu, améliorant ainsi l'expérience de l'utilisateur et l'efficacité opérationnelle. Par conséquent, la réduction de la latence obtenue grâce au traitement en mémoire est un facteur clé pour maintenir des performances élevées et une réactivité rapide dans les environnements en temps réel.

3.1.3.3 Amélioration de la scalabilité

La capacité de Spark à évoluer de manière linéaire en ajoutant des nœuds supplémentaires lui permet de s'adapter efficacement aux changements dans les charges de travail. Cela signifie que lorsque le volume de données et la complexité des calculs augmentent, des nœuds supplémentaires peuvent être ajoutés au cluster sans nécessiter de reconfiguration majeure. Chaque nouvel ajout apporte plus de mémoire et de puissance de calcul, ce qui maintient les performances élevées du système. Cette évolutivité est particulièrement avantageuse dans les environnements dynamiques où les charges de données fluctuent rapidement. En élargissant le cluster, Spark peut répartir efficacement les tâches sur toutes les ressources disponibles, assurant une répartition équilibrée des charges de travail et évitant les ralentissements. Cela permet à Spark de répondre efficacement aux besoins de traitement et d'analyse de données à grande échelle (Big Data).

3.1.4 Comparaison entre Spark RDD et Hadoop MapReduce

Facteurs	HADOOP	SPARK
Difficulté	Il est plus difficile à utiliser.	Spark est plus facile à utiliser
Coût	Hadoop est une option moins chère	Nécessite beaucoup de RAM pour fonctionner en mémoire, ce qui augmente la taille du cluster et donc les coûts.
Vitesse	Plus rapide que les systèmes traditionnels	Apache Spark est 100 fois plus vite en mémoire et 10 fois plus vite sur disque que Hadoop.[2]
Latence	Latence élevée	Faible latence
Sécurité	Hadoop dispose de solides fonctions de sécurité, de cryptage du stockage et de contrôle d'accès.	Spark n'est pas sécurisé, il repose sur l'intégration avec Hadoop pour atteindre le niveau de sécurité nécessaire.
Diffusion (Streaming)	Hadoop est conçu pour gérer efficacement le traitement par lots (Batch processing).	Spark est conçu pour traiter efficacement les données en temps réel

Table 1. Comparaison entre Spark RDD et Hadoop MapReduce

4 Principes de Machine Learning

Le Machine Learning (ML) [4] est une forme d'intelligence artificielle (IA) qui est axée sur la création de systèmes qui apprennent, ou améliorent leurs performances, en fonction des données qu'ils traitent. L'intelligence artificielle est un terme large qui désigne des systèmes ou des machines simulant une forme

d'intelligence humaine. Le ML et l'IA sont souvent abordés ensemble et ces termes sont parfois utilisés de manière interchangeable bien qu'ils ne renvoient pas exactement au même concept. Une distinction essentielle est que, bien que l'ensemble de la ML soit basé sur l'intelligence artificielle, l'intelligence artificielle n'est pas limitée à la ML.

4.1 Apprentissage supervisé versus non-supervisé

L'Apprentissage supervisé [4] se définit par l'utilisation d'ensembles de données étiquetées pour former des algorithmes permettant de classer les données ou de prédire les résultats avec précision. Au fur et à mesure que les données d'entrée sont introduites dans le modèle, celui-ci ajuste ses poids jusqu'à ce qu'il soit correctement ajusté. Cette opération s'inscrit dans le cadre du processus de validation croisée, qui permet de s'assurer que le modèle n'est pas sur- ou sous-adapté. L'apprentissage supervisé aide les organisations à résoudre une variété de problèmes réels à grande échelle, tels que la classification des spams dans un dossier distinct de votre boîte de réception. Parmi les méthodes utilisées dans l'apprentissage supervisé figurent la régression linéaire simple ou multiple, la régression logistique, la forêt aléatoire et le support vector machine (la machine à vecteur de support SVM).

Apprentissage non-supervisé [4] également connu sous le nom d'apprentissage automatique non supervisé, utilise des algorithmes d'apprentissage automatique pour analyser et regrouper des ensembles de données non étiquetés (sous-ensembles appelés clusters). Ces algorithmes découvrent des modèles cachés ou des regroupements de données sans nécessiter d'intervention humaine. La capacité de cette méthode à découvrir des similitudes et des différences dans les informations la rend idéale pour l'analyse exploratoire des données, les stratégies de vente croisée, la segmentation de la clientèle et la reconnaissance des images et des formes. Elle est également utilisée pour réduire le nombre de caractéristiques d'un modèle grâce au processus de réduction de la dimensionnalité. L'analyse en composantes principales (PCA) et la décomposition en valeurs singulières (SVD) sont deux approches courantes. Parmi les autres algorithmes utilisés dans l'apprentissage non supervisé figurent les réseaux neuronaux, le regroupement par k-moyennes et les méthodes de regroupement probabilistes.

La principale différence entre l'apprentissage supervisé et l'apprentissage non supervisé est la nécessité de disposer de données de formation étiquetées. L'apprentissage automatique supervisé repose sur des données d'entrée et de sortie étiquetées, tandis que l'apprentissage non supervisé traite des données brutes ou non étiquetées. Dans l'apprentissage automatique supervisé, le modèle apprend la relation entre les données d'entrée et de sortie étiquetées. Les modèles sont affinés jusqu'à ce qu'ils puissent prédire avec précision les résultats de données inédites. Cependant, la création de données de formation étiquetées nécessite souvent des ressources importantes. L'apprentissage automatique non supervisé, quant à lui, apprend à partir de données d'apprentissage brutes non étiquetées. Un modèle non supervisé apprend les relations et les modèles au sein de cet ensemble de données non étiquetées, et est donc souvent utilisé pour découvrir les tendances inhérentes à un ensemble de données donné.

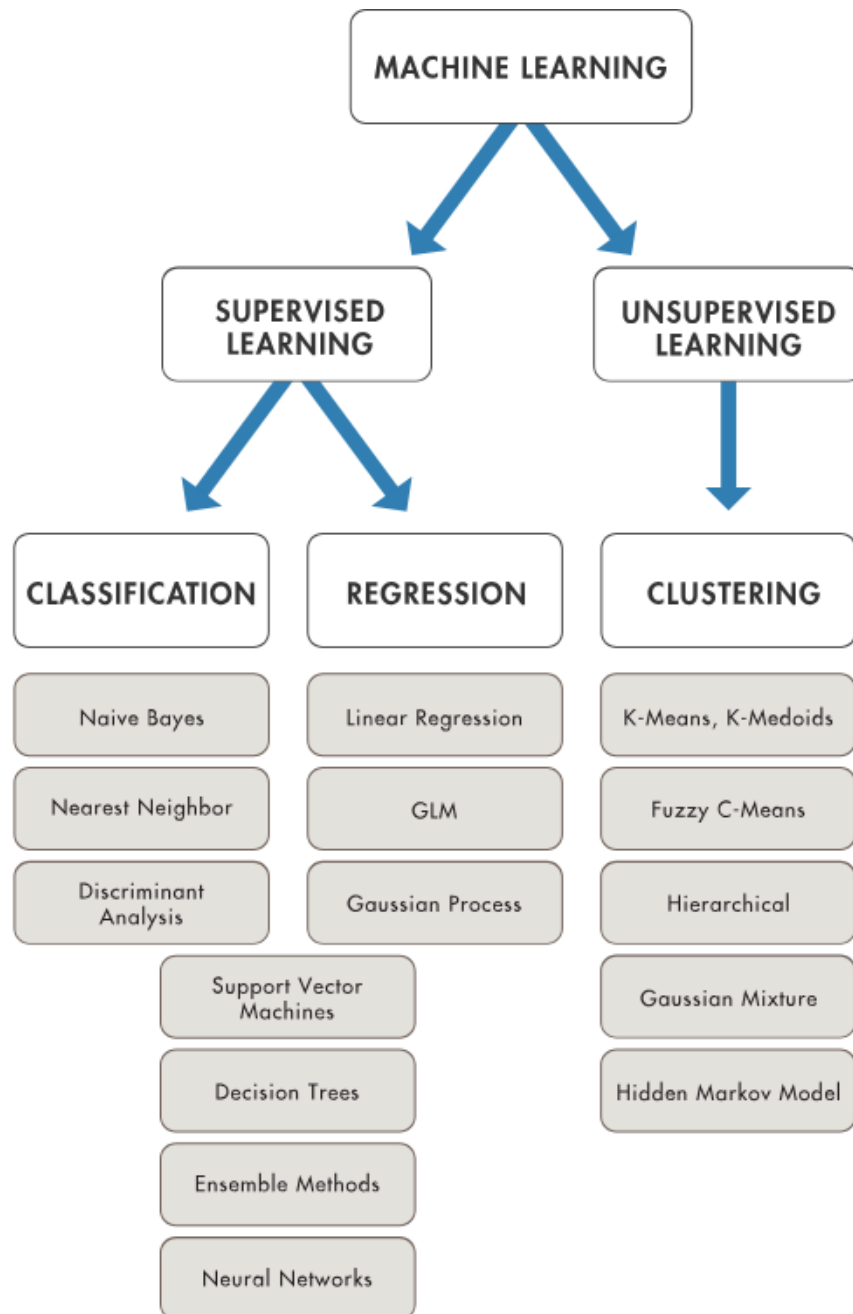


Figure 11. Les Algorithmes du Machine Learning

En résumé Les principales différences entre l'apprentissage supervisé et l'apprentissage non supervisé sont les suivantes :

- La nécessité de disposer de données étiquetées dans l'apprentissage automatique supervisé.
- Le problème que le modèle est censé résoudre. L'apprentissage automatique supervisé est généralement utilisé pour classer les données ou faire des prédictions, tandis que l'apprentissage non supervisé est généralement utilisé pour comprendre les relations au sein des ensembles de données.
- L'apprentissage automatique supervisé est beaucoup plus gourmand en ressources en raison de la nécessité de disposer de données étiquetées.
- Dans l'apprentissage automatique non supervisé, il peut être plus difficile d'atteindre des niveaux adéquats d'explicabilité en raison d'une supervision humaine moindre.

4.2 Les Algorithmes de la bibliothèque d'apprentissage automatique Spark (MLlib)

Pour mon projet, qui porte sur le développement d'un système de prédiction, je me concentrerai sur les algorithmes d'apprentissage automatique supervisé.

- L'objectif des algorithmes du Machine Learning est de trouver la fonction cible (f) qui mappe le mieux les variables d'entrée (X) à une variable de sortie (Y) : $Y=f(X)$
- Dans ce cas, nous pourrions faire des prévisions dans l'avenir (Y) à partir de nouveaux exemples de variables d'entrée (X).
- Un algorithme ne peut à lui seul résoudre tous les problèmes d'apprentissage automatique, des problèmes différents nécessitent des approches et des techniques différentes. C'est pourquoi il est essentiel de comprendre une variété d'algorithmes d'apprentissage automatique.

4.2.1 Régression linéaire

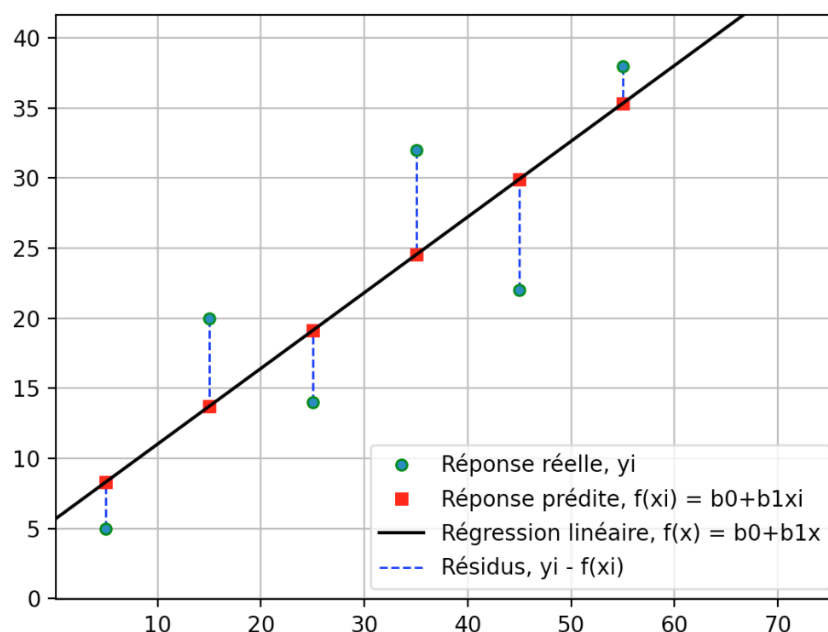
La régression linéaire [5] est une technique de modélisation statistique utilisée pour décrire une variable de réponse continue en fonction d'une ou plusieurs variables prédictives

4.2.1.1 Régression linéaire simple

Dans la régression linéaire simple, il n'y a qu'une seule variable indépendante (variable explicative X) utilisée pour prédire la variable dépendante Y .

- Le modèle est de la forme $Y = a + bX$, où Y est la variable dépendante, X est la variable indépendante, a est l'intercept (le point où la ligne de régression coupe l'axe des ordonnées) et b est la pente (la relation linéaire entre X et Y).
- Donc, l'objectif de la régression linéaire simple est de trouver l'équation de la droite $Y = a + bX$, c'est-à-dire de trouver a et b .

L'objectif est de minimiser la distance entre la droite (le modèle) et les points de données comme il est montré dans la **figure 12**.



4.2.1.2 Régression linéaire multiple

- Dans la régression linéaire multiple, il y a plusieurs variables indépendantes (variables explicatives) utilisées pour prédire la variable dépendante.
- Le modèle est de la forme $Y = a + b_1x_1 + b_2x_2 + \dots + b_nx_n$, où Y est la variable dépendante, x_1, x_2, \dots, x_n sont les variables indépendantes, a est l'intercept et b_1, b_2, \dots, b_n sont les coefficients de régression correspondant à chaque variable indépendante.
- L'objectif est de trouver a, b_1, b_2, \dots , et b_n .

4.2.2 L'arbre de décision et la forêt aléatoire

4.2.2.1 Arbre de décision

Un arbre de décision (Decision Tree) est une structure arborescente similaire à un organigramme, où chaque nœud interne (nœud non feuille) correspond à un test sur un attribut, chaque branche représente un résultat du test et chaque nœud feuille (ou nœud terminal) possède une étiquette de classe.

- Le nœud qui se situe au sommet de l'arbre est appelé racine.

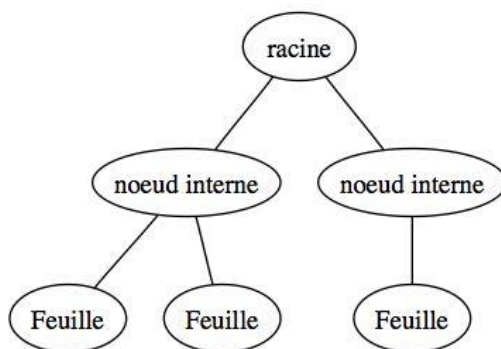


Figure 13. Structure d'un arbre de décision.

4.2.2.2 Forêt aléatoire

Dans une forêt aléatoire, au lieu qu'un seul arbre de décision prenne toutes les décisions, nous créons une « forêt » entière d'arbres de décision. Chaque arbre donne son « opinion » ou sa prédiction sur la base des données qu'il a vues. Le résultat final est ensuite déterminé en tenant compte des résultats de tous les arbres de la forêt.

- Cette « collecte d'opinions » est rendue plus excitante en donnant à chaque arbre un ensemble légèrement différent de points de données et de caractéristiques à partir desquels il peut apprendre. C'est un peu comme si chacun de vos amis connaissait des choses légèrement différentes sur votre situation. Cette technique est connue sous le nom de « bagging » et de « feature randomness », ce qui rend le modèle plus robuste car il est moins susceptible de mémoriser les données d'apprentissage et plus susceptible de bien se généraliser à de nouvelles données.

4.2.3 Gradient-Boosted Trees (GBTs) :

Le Gradient Boosting est une méthodologie appliquée au-dessus d'un autre algorithme d'apprentissage automatique. De manière informelle, le gradient boosting implique deux types de modèles :

- Un modèle d'apprentissage automatique « faible », qui est généralement un arbre de décision.

- Un modèle d'apprentissage automatique « fort », composé de plusieurs modèles faibles.

Dans le gradient boosting, à chaque étape, un nouveau modèle faible est formé pour prédire l'« erreur » du modèle fort actuel (que l'on appelle la pseudo-réponse). Pour l'instant, nous supposons que l'« erreur » est la différence entre la prédiction et une étiquette régressive. Le modèle faible (c'est-à-dire l'« erreur ») est ensuite ajouté au modèle fort avec un signe négatif pour réduire l'erreur du modèle fort.

Le renforcement du gradient est itératif. Chaque itération fait appel à la formule suivante :

$$F_{i+1} = F_i - f_i$$

- F_i : est le modèle fort à l'étape i .
- f_i : est le modèle faible à l'étape i .

Cette opération se répète jusqu'à ce qu'un critère d'arrêt soit rempli, tel qu'un nombre maximum d'itérations ou si le modèle (fort) commence à se sur-adapter, tel que mesuré sur un ensemble de données de validation séparé.

Dans les problèmes de régression, il est logique de définir l'erreur signée comme la différence entre la prédiction et l'étiquette. Cependant, dans d'autres types de problèmes, cette stratégie conduit souvent à des résultats médiocres. Une meilleure stratégie utilisée dans le Gradient Boosting consiste à :

- Définir une fonction de perte similaire aux fonctions de perte utilisées dans les réseaux neuronaux. Par exemple, l'entropie (également connue sous le nom de perte logarithmique) pour un problème de classification.
- Entraîner le modèle faible à prédire le gradient de la perte en fonction de la sortie du modèle fort.

En général, le Gradient Boosting est une puissante technique d'apprentissage automatique qui excelle dans la précision des prédictions en combinant les forces de plusieurs apprenants faibles, souvent des arbres de décision, dans un processus itératif. Elle ajuste continuellement son modèle en fonction des erreurs des prédictions précédentes, ce qui permet d'obtenir des modèles très précis et adaptables à diverses tâches. Cependant, il peut être intensif en termes de calcul, nécessite un réglage minutieux des hyperparamètres et peut manquer d'interprétabilité en raison de sa nature d'ensemble.

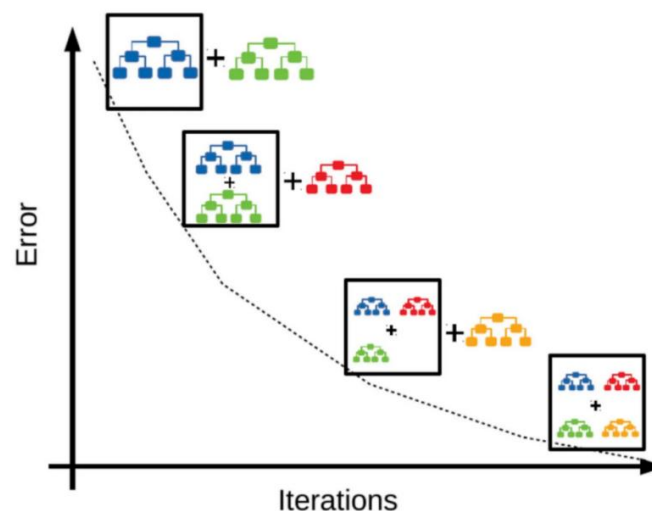


Figure 14. Gradient Boosting Algorithm.

4.2.4 XGBOOST

XGBoost est une bibliothèque distribuée optimisée de gradient boosting conçue pour être hautement efficace, flexible et portable. Elle implémente des algorithmes d'apprentissage automatique dans le cadre du Gradient Boosting, également connu sous le nom de GBDT ou GBM. XGBoost fournit un arbre parallèle de boosting qui résout de nombreux problèmes de science des données de manière rapide et précise. Les principaux avantages de XGBoost résident dans sa capacité à gérer efficacement de grands ensembles de données, à produire des modèles hautement précis et à être robuste face à des données bruitées ou incomplètes.[6]

Contrairement au Random Forest, l'algorithme XGBoost travaille de manière séquentielle, ce qui améliore le boosting de gradient en matière d'échelle et de vitesse de calcul. Le boosting de gradient est basé sur un arbre de décision d'ensemble dont le principe est de combiner les résultats d'un ensemble de modèles dans le but de fournir une meilleure prédiction. Il s'agit en réalité d'une approche pragmatique qui permet de gérer aussi bien des problèmes de classification que de régression.

L'algorithme de boosting XGBoost est capable de traiter un volume important de jeux de données, ce qui le rend particulièrement utile pour les applications de Big Data. Les principales caractéristiques du boosting de gradient extrême sont le calcul distribué, la parallélisation, le traitement hors du cœur et l'optimisation du cache.[6]

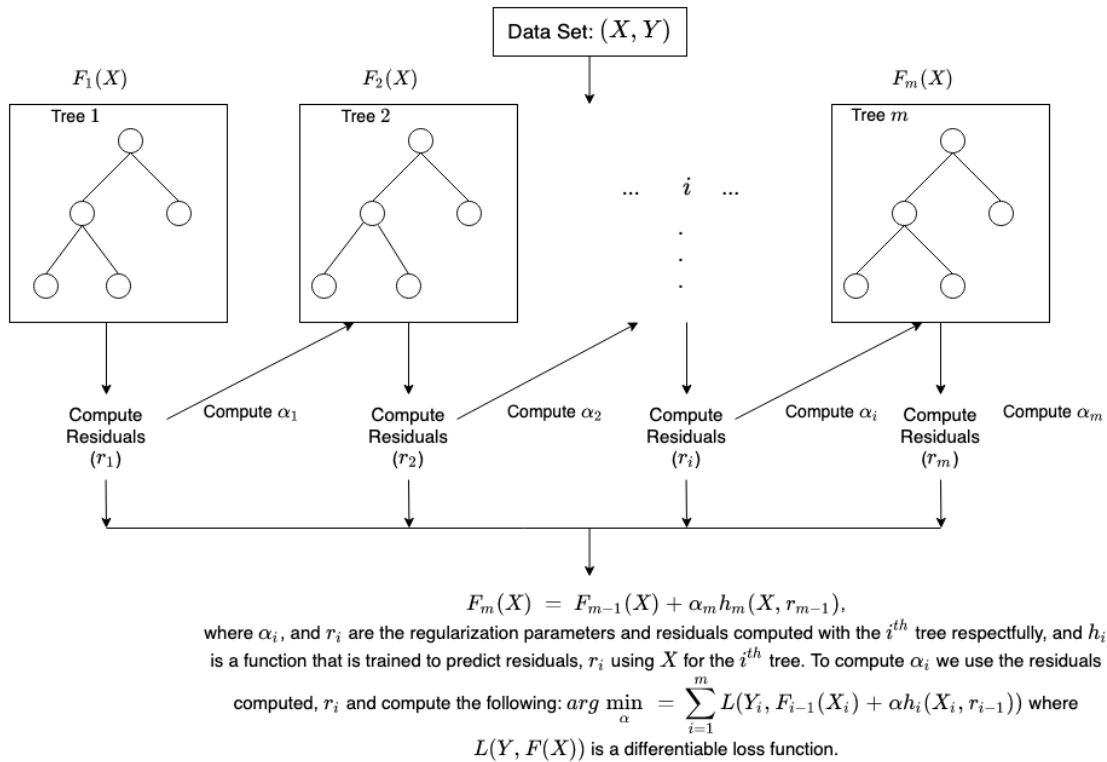


Figure 15. Le fonctionnement du XGBOOST.

Chapitre II : Développement du système de prédiction : Architecture et implémentation

1 Introduction

Dans la continuité des fondements théoriques établis dans le Chapitre 1, ce chapitre se concentre sur la concrétisation des concepts abordés à travers le développement et l'évaluation d'un système de prédiction. Nous passons ainsi de la théorie à la pratique, en explorant les détails de l'architecture, de l'implémentation et des performances du système. Avant d'entrer dans les détails de ce chapitre, il est important de préciser que le système de prédiction sera déployé sur un cluster, avec Spark comme outil principal. Ce choix de déploiement sur un cluster est essentiel pour garantir des performances optimales, une scalabilité efficace et une gestion efficace des ressources. Pour bénéficier pleinement des capacités de Spark, il est crucial de le déployer dans un environnement distribué, tel qu'une machine virtuelle dans un cluster. Cela permet une répartition efficace des tâches de calcul sur plusieurs nœuds, exploitant ainsi pleinement la puissance de traitement parallèle offerte par Spark. Ce chapitre vise à fournir une vue d'ensemble complète du processus de développement d'un système de prédiction, en mettant en avant les aspects pratiques de la mise en œuvre et en soulignant les meilleures pratiques pour garantir des performances optimales.

2 Architecture du système

2.1 Vue d'ensemble de l'architecture

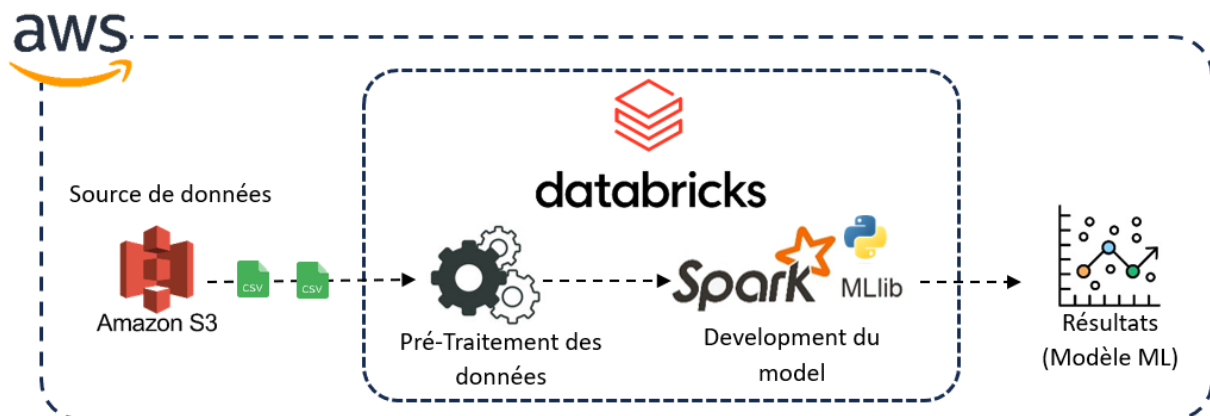


Figure 16. Architecture proposée.

Pour mieux comprendre l'architecture proposée, examinons de plus près ses différents composants et leur fonctionnement. La figure illustre visuellement la disposition des éléments clés de l'architecture, offrant ainsi un aperçu de la manière dont ils interagissent pour former un système de prédiction complet et fonctionnel.

Ce système est déployé sur Databricks sur AWS (Amazon Web Services). Il s'agit d'un service géré proposé par Databricks en collaboration avec AWS. Il simplifie le processus de configuration car Databricks et votre environnement Spark s'exécutent tous deux sur l'infrastructure AWS.

- **Amazon Web Services** est une division du groupe américain de commerce électronique Amazon, spécialisée dans les services de cloud computing à la demande pour les entreprises et particuliers.

AWS fournit une infrastructure cloud robuste et évolutive, garantissant une haute disponibilité et une performance optimale pour notre système de prédiction.

- **Databricks** est une société mondiale de données, d'analyse et d'intelligence artificielle fondée par les créateurs originaux d'Apache Spark. Databricks offre une plateforme intégrée qui facilite le traitement des données et le développement de modèles de Machine Learning. En utilisant Databricks, nous pouvons tirer parti de ses fonctionnalités avancées pour gérer les tâches de traitement de données, entraîner nos modèles de prédiction et analyser les résultats de manière efficace.

Dans le contexte de notre projet, le déploiement de Spark sur des clusters Databricks sur AWS permet de bénéficier pleinement des capacités de traitement parallèle et distribué de Spark. Les clusters permettent une répartition efficace des tâches de calcul sur plusieurs nœuds, assurant ainsi des performances optimales et une scalabilité efficace. Cela est crucial pour gérer de grandes quantités de données et exécuter des algorithmes de Machine Learning complexes.

En résumé, cette architecture distribuée et gérée par Databricks sur AWS nous offre un environnement performant et flexible pour développer et déployer notre système de prédiction, tout en simplifiant les aspects de configuration et de gestion des ressources.

2.1.1 Les composantes de l'architecture

- **Amazon S3** (Source de données)

Amazon S3 joue un rôle crucial en tant que source de données. Il fournit un stockage cloud évolutif et sécurisé pour les données brutes, les données prétraitées et les modèles déployés. Ce stockage garantit que les données sont facilement accessibles pour le Databricks Workspace et Spark MLlib, tout en offrant une collaboration fluide entre les différents composants du système. Les données peuvent être directement stockées et accessibles sur Amazon S3 pour les tâches d'entraînement et de prédiction, assurant ainsi une évolutivité et une accessibilité optimales.

- **Prétraitement des données**

Le prétraitement des données est une étape cruciale où les données brutes sont nettoyées et préparées pour être utilisées par les modèles d'apprentissage automatique. Cela inclut la gestion des valeurs manquantes, la suppression des doublons et la transformation des données en formats adéquats.

- **Databricks Workspace**

Le Databricks Workspace est le cœur du système de prédiction. C'est une plateforme cloud d'analyse et d'apprentissage automatique offrant un environnement sécurisé et évolutif pour le développement, l'exécution, et le déploiement de modèles de prédiction.

- **Spark MLlib**

Databricks utilise Spark MLlib pour développer et entraîner des modèles de prédiction à grande échelle sur les données stockées dans S3.

- **Développement du modèle**

La phase de développement du modèle implique :

- La création de modèles de prédiction.
- L'entraînement de ces modèles.
- L'évaluation de différents modèles en utilisant Spark MLlib.

Les algorithmes d'apprentissage automatique appropriés sont sélectionnés en fonction de la nature des données et du type de prédiction souhaité.

- **Evaluation du modèle**

Les modèles de prédiction développés sont évalués sur un ensemble de données de validation pour mesurer leur précision et leur performance. Les métriques d'évaluation couramment utilisées incluent :

- La précision.
- Le rappel.
- L'erreur quadratique moyenne.

Ces métriques permettent de comparer les modèles et de sélectionner le meilleur pour la prédiction finale.

- **Déploiement du modèle**

Le modèle de prédiction sélectionné est déployé dans l'environnement de production. Cette étape consiste à emballer le modèle dans un format exécutable et à le rendre accessible aux applications ou services pour générer des prédictions.

- **Génération de prédictions**

Les nouvelles données, ou données non vues, sont traitées et prétraitées de la même manière que les données d'entraînement. Le modèle de prédiction déployé est ensuite exécuté sur ces données prétraitées pour générer des prédictions.

- **Résultats et visualisations**

Les prédictions générées par le modèle sont présentées aux utilisateurs de manière claire et compréhensible. Cela peut inclure la création de visualisations, de tableaux de bord, ou de rapports pour communiquer efficacement les résultats et les informations exploitables.

2.1.2 Interaction entre les composants

L'interaction entre les composants du système de prédiction est un processus crucial qui assure le bon fonctionnement de l'ensemble du système. Voici une analyse détaillée de la manière dont chaque composant interagit avec les autres :

- **Databricks Workspace et Amazon S3**

Databricks Workspace utilise Amazon S3 comme une source de données principale. Il interagit avec Amazon S3 pour :

- Importer les données brutes : Databricks Workspace récupère les données brutes depuis Amazon S3 afin de les prétraiter avant leur utilisation dans le processus de modélisation.

- Stocker les données prétraitées : Une fois les données prétraitées, Databricks Workspace les stocke de nouveau sur Amazon S3. Cela garantit que les données sont accessibles et sécurisées pour l'entraînement des modèles.

- **Databricks Workspace et Spark MLlib**

Databricks Workspace utilise Spark MLlib pour le développement, l'entraînement et l'évaluation des modèles de prédiction. L'interaction entre Databricks Workspace et Spark MLlib se déroule comme suit :

- Entraînement des modèles : Les données prétraitées stockées sur Amazon S3 sont utilisées par Databricks Workspace pour entraîner les modèles de prédiction à l'aide de Spark MLlib. Cela implique l'utilisation d'algorithmes d'apprentissage automatique pour construire des modèles à partir des données disponibles.
- Évaluation des modèles : Une fois les modèles entraînés, Databricks Workspace utilise Spark MLlib pour évaluer la performance de ces modèles en utilisant des techniques telles que la validation croisée ou la répartition des données en ensembles d'entraînement et de validation.

- **Déploiement des Modèles sur Amazon S3**

Une fois que les modèles sont entraînés et évalués avec succès, Databricks Workspace les déploie sur Amazon S3 pour un accès futur et une utilisation dans la génération de prédictions. Cela garantit que les modèles sont stockés de manière sécurisée et peuvent être facilement récupérés lorsque de nouvelles données doivent être prédites.

- **Génération de Nouvelles Prédictions**

Lorsqu'il est temps de générer de nouvelles prédictions, Databricks Workspace utilise les modèles préalablement déployés sur Amazon S3. Il récupère ces modèles et les utilise pour traiter les nouvelles données, en s'appuyant à la fois sur les capacités de stockage d'Amazon S3 et sur les algorithmes performants de Spark MLlib pour garantir des prédictions précises et opportunes.

Chaque interaction entre les composants du système de prédiction contribue de manière essentielle à l'objectif global de transformation des données brutes en informations prédictives précieuses. Cette intégration harmonieuse assure une performance optimale et une exploitation efficace des différentes composantes de l'architecture.

2.2 Configuration initiale du cluster Spark sur DataBricks

Avant de créer un cluster Spark sur Databricks, il est important de comprendre que le choix du type de cluster et des paramètres de configuration dépendent des besoins spécifiques en termes de calcul, de mémoire et de stockage. Databricks propose une variété de types d'instances AWS pour répondre à différents besoins de performance. Les paramètres du cluster peuvent être ajustés après sa création si nécessaire. Il est recommandé de surveiller l'utilisation des ressources du cluster et d'ajuster les paramètres en conséquence pour optimiser les performances et les coûts.

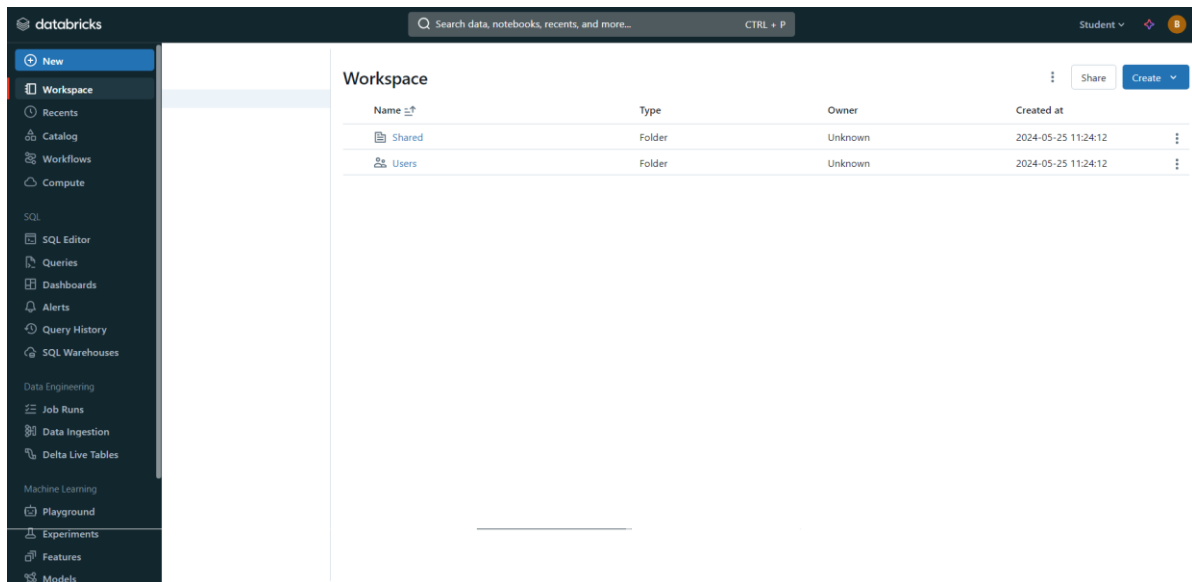


Figure 17. Databricks Workspace.

Les étapes de création d'un cluster :

Compute > New compute >

PFE Cluster

Policy ⓘ

Unrestricted

☒ Multi node ☐ Single node

Access mode ⓘ Single user access ⓘ

Single user BAY BAY Badr

Performance

Databricks runtime version ⓘ

Runtime: 13.3 LTS (Scala 2.12, Spark 3.4.1)

☒ Use Photon Acceleration ⓘ

Worker type ⓘ

r6id.xlarge 32 GB Memory, 4 Cores

Min workers 2 Max workers 8

Driver type

Same as worker 32 GB Memory, 4 Cores

☒ Enable autoscaling ⓘ

☐ Enable autoscaling local storage ⓘ

☒ Terminate after 120 minutes of inactivity ⓘ

Create compute Cancel

Finalement appuyez

- Liste des clusters déjà créés
- Choisissez entre un cluster multi-nœuds ou un seul nœud.
- Mode d'accès
- Sélectionnez la version de Spark
- Choisissez le type de machine pour les nœuds du cluster. Dans cet exemple, "r6id.xlarge" avec 32 Go de mémoire et 4 cœurs est utilisé.
- Spécifiez le nombre minimum et maximum de nœuds dans le cluster.
- Choisissez le type de machine pour le nœud du driver.
- Spécifiez la durée d'inactivité du cluster après laquelle il doit être arrêté.

Une fois ces étapes complétées, le cluster sera créé et disponible pour être utilisé dans l'environnement Databricks.

3 Implémentation

On décrit ci-dessous les étapes d'exécutions sur Spark :

3.1 La création du Maître et session Spark

Normalement à chaque application Spark, la première opération consiste à se connecter au Maître Spark et à obtenir une session Spark, mais dans un environnement Databricks, vous disposez déjà d'une infrastructure configurée pour utiliser Spark donc la session Spark est généralement créée automatiquement, mais vous pouvez toujours la référencer comme suit :

```
# Créer un objet SparkSession
spark = SparkSession.builder.appName("Système de prédiction basé\
sur Spark et les algorithmes de machine learning").getOrCreate()
spark
```

SparkSession - hive
SparkContext
[Spark UI](#)
Version
v3.4.1
Master
spark://10.165.237.104:7077
AppName
Databricks Shell

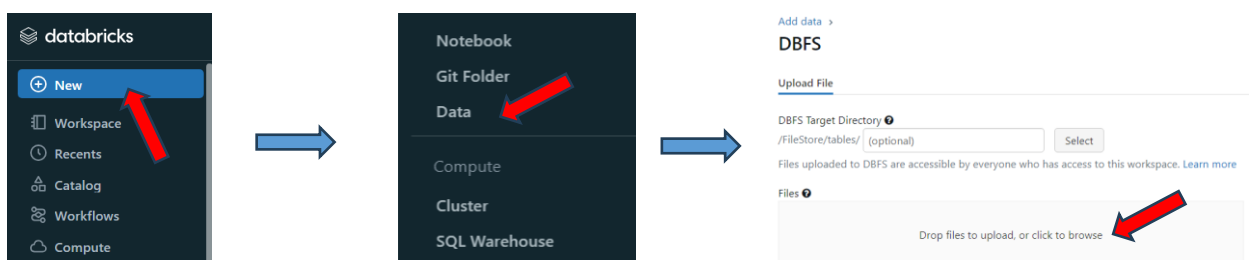
Figure 18. Création d'une session Spark.

Dans la figure, on voit les informations suivantes :

- **SparkSession - hive** : Indique que la session Spark utilise le support Hive.
- **SparkContext** : Le contexte Spark associé à cette SparkSession.
- **Spark UI** : L'interface web fournie par Spark pour surveiller les jobs, les stages, les tâches, le DAG, et plus encore, ce qui permet de comprendre les étapes de traitement des données. Elle est accessible via l'URL fournie.
- **Version** : Version de Spark utilisée.
- **Master** : URL du maître Spark (le nœud principal du cluster).
- **AppName** : Nom de l'application Spark.

3.2 Chargement des données

Databricks facilite le chargement des données à partir de divers formats et emplacements, y compris le DBFS (Databricks File System). Cependant, pour importer des données dans un notebook Databricks, vous devez d'abord les charger sur DBFS, nous décrirons le processus ci-dessus :



DBFS

Upload File

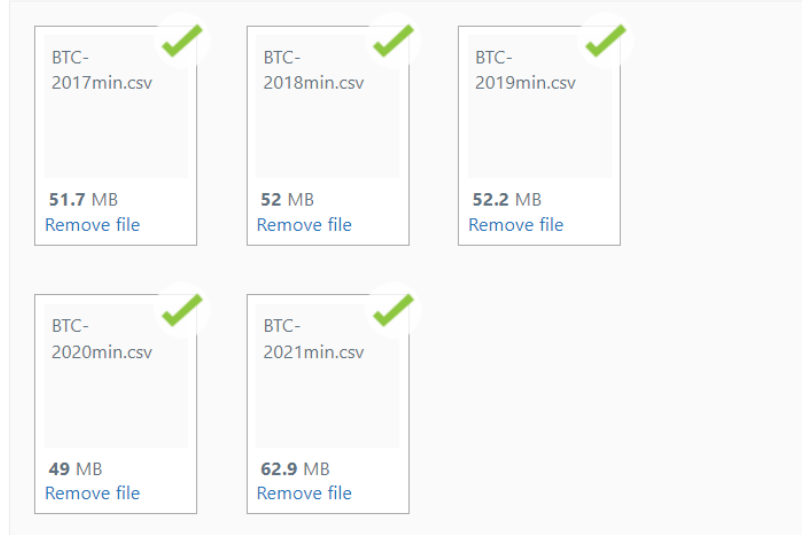
DBFS Target Directory ?

/FileStore/tables/ (optional)

Select

Files uploaded to DBFS are accessible by everyone who has access to this workspace. [Learn more](#)

Files ?



- ✓File uploaded to /FileStore/tables/BTC_2017min.csv
- ✓File uploaded to /FileStore/tables/BTC_2018min.csv
- ✓File uploaded to /FileStore/tables/BTC_2020min.csv
- ✓File uploaded to /FileStore/tables/BTC_2019min.csv
- ✓File uploaded to /FileStore/tables/BTC_2021min.csv

Figure 19. Le processus du chargement des données dans DBFS.

Une fois les fichiers téléversés dans DBFS, ils peuvent être utilisés dans notre notebooks Databricks. Voici le code Python pour lire ces fichiers CSV dans un DataFrame :

```
# Charger les fichiers CSV
df_2017 = spark.read.csv("/FileStore/tables/BTC_2017min.csv", header=True, inferSchema=True)
df_2018 = spark.read.csv("/FileStore/tables/BTC_2018min.csv", header=True, inferSchema=True)
df_2019 = spark.read.csv("/FileStore/tables/BTC_2019min.csv", header=True, inferSchema=True)
df_2020 = spark.read.csv("/FileStore/tables/BTC_2020min.csv", header=True, inferSchema=True)
df_2021 = spark.read.csv("/FileStore/tables/BTC_2021min.csv", header=True, inferSchema=True)
#Concaténer les DataFrames
df = df_2017.union(df_2018).union(df_2019).union(df_2020).union(df_2021)
```

► (10) Spark Jobs

Figure 20. L'importation des fichiers CSV dans Databricks notebook.

« (10) Spark jobs » indique que le processus de lecture des 5 fichiers csv est réparti en 10 jobs Spark. Cela signifie que Spark a divisé la tâche de lecture des cinq fichiers en dix tâches indépendantes plus petites et les exécute en parallèle sur un cluster de machines.

3.3 Présentation des données

Le **Bitcoin** (BTC) est une cryptomonnaie décentralisée, fondée en 2009 par une entité pseudonyme connue sous le nom de Satoshi Nakamoto. En tant que première cryptomonnaie, le Bitcoin a introduit la blockchain, garantissant la transparence, la sécurité et l'immutabilité des transactions. Fonctionnant sans autorité centrale, il repose sur un réseau de pairs pour valider et enregistrer les transactions. La prédiction du prix du Bitcoin est cruciale en raison de sa volatilité, de son adoption croissante et de son impact

économique. Les technologies comme le Machine Learning et Apache Spark permettent d'analyser des données complexes pour des prévisions précises.

Pour notre projet, nous utilisons un ensemble de données historiques sur le prix du Bitcoin en dollar américain, comprenant 2,6 millions de lignes. Cet ensemble inclut les prix de chaque minute de 2017 à 2022, les volumes de transactions, et d'autres indicateurs. En appliquant des algorithmes de Machine Learning à ces données, nous pouvons modéliser et prévoir les fluctuations futures du prix du Bitcoin, exploitant la puissance de Spark pour traiter et analyser de grands volumes de données efficacement.

unix	date	symbol	open	high	low	close	Volume BTC	Volume USD
1646106180	2022-03-01 03:43:00	BTC/USD	43046.58	43046.58	43046.58	43046.58	0.0	0.0
1646106060	2022-03-01 03:41:00	BTC/USD	43018.23	43046.59	43018.23	43046.58	0.14297705	6154.673020989
1646106000	2022-03-01 03:40:00	BTC/USD	43022.24	43022.24	43016.03	43016.03	0.00923	397.0379569
1646105940	2022-03-01 03:39:00	BTC/USD	43035.16	43035.16	42999.44	42999.44	0.82095	35300.390268
1646105880	2022-03-01 03:38:00	BTC/USD	43077.82	43077.82	43049.46	43049.46	0.02221034	956.1431434163999
1646105820	2022-03-01 03:37:00	BTC/USD	43078.73	43092.09	43078.73	43088.92	1.59309482	68644.73525139439
1646105760	2022-03-01 03:36:00	BTC/USD	43095.57	43095.57	43068.85	43090.55	0.06888172	2968.1511997459997
1646105700	2022-03-01 03:35:00	BTC/USD	43098.34	43098.42	43083.17	43083.17	0.60903	26238.943025099998
1646105640	2022-03-01 03:34:00	BTC/USD	43091.15	43098.42	43053.76	43098.42	0.09774787	4212.7787553654
1646105580	2022-03-01 03:33:00	BTC/USD	43073.99	43098.42	43073.99	43098.42	0.0287456	1238.8899419519998
1646105520	2022-03-01 03:32:00	BTC/USD	43107.67	43107.67	43107.67	43107.67	0.0	0.0
1646105460	2022-03-01 03:31:00	BTC/USD	43089.57	43107.67	43089.57	43107.67	0.01657	714.2940919
1646105400	2022-03-01 03:30:00	BTC/USD	43065.82	43106.72	43065.82	43093.52	0.31304999	13490.426005064799
1646105340	2022-03-01 03:29:00	BTC/USD	43085.12	43085.12	43042.51	43065.82	0.01437843	619.2188782626
1646105280	2022-03-01 03:28:00	BTC/USD	43002.0	43047.74	42999.7	43047.74	0.13463	5795.517236199999
1646105220	2022-03-01 03:27:00	BTC/USD	42953.79	42976.12	42937.05	42976.12	1.4995318	64444.058580616
1646105160	2022-03-01 03:26:00	BTC/USD	43002.2	43002.2	42960.8	42960.8	0.10754105	4620.04954084

Figure 21. Dataset choisi.

À propos de chaque colonne :

- **Unix Timestamp** : Il s'agit du moment précis où une donnée a été enregistrée, exprimé en nombre de secondes écoulées depuis le 1er janvier 1970, 00:00:00 UTC. C'est une référence temporelle fondamentale pour analyser l'évolution des prix du Bitcoin au fil du temps.
- **Date** : Cette colonne indique la date et l'heure où la donnée a été enregistrée, en utilisant le fuseau horaire UTC (Temps Universel Coordonné). Elle permet de visualiser les données chronologiquement dans leur contexte temporel.
- **Symbole** : Ce champ spécifie le symbole associé à la paire de trading du Bitcoin. Par exemple, "BTC/USD" indique que les prix sont exprimés en dollars américains par unité de Bitcoin. Il permet de différencier les données provenant de différentes paires de trading.
- **Open** : Ce prix représente la valeur du Bitcoin au début de la période considérée. Il s'agit du premier prix enregistré dans l'intervalle de temps spécifié.
- **High** : Il s'agit du prix le plus élevé atteint par le Bitcoin au cours de la période considérée. Cela indique le pic de valeur maximal atteint pendant cette période.
- **Low** : Ce prix représente le niveau le plus bas atteint par le Bitcoin au cours de la période considérée. Il illustre le creux de valeur minimal observé pendant cette période.
- **Close** : Ce prix représente la valeur du Bitcoin à la fin de la période considérée. Il s'agit du dernier prix enregistré dans l'intervalle de temps spécifié.

- **Volume BTC** : Ce chiffre indique la quantité de Bitcoin échangée pendant la période considérée. Il permet d'évaluer l'activité de trading et la liquidité du marché.
- **Volume USD** : Il s'agit de la valeur totale des transactions effectuées en dollars américains pendant la période considérée. Cela fournit une mesure de l'ampleur des échanges en termes de valeur fiduciaire.

3.4 Prétraitement des données

```
# Prétraitement des données
df = df.dropna().dropDuplicates()
quantiles = df.approxQuantile("close", [0.25, 0.75], 0.05)
Q1, Q3 = quantiles[0], quantiles[1]
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df.filter((col("close") >= lower_bound) & (col("close") <= upper_bound))
```

Figure 22. Prétraitement des données.

Le prétraitement des données a été effectué pour nettoyer et préparer l'ensemble de données en vue de l'analyse et de la modélisation. Les étapes suivantes ont été réalisées :

- **Suppression des valeurs manquantes et des doublons** : Les lignes contenant des valeurs manquantes ont été supprimées à l'aide de **dropna()**, et les doublons ont été supprimés avec **dropDuplicates()** pour garantir l'intégrité des données.
- **Détection et filtrage des valeurs aberrantes** (outliers) :
 - Les quantiles (Q1 et Q3) de la colonne "close" ont été calculés en utilisant **approxQuantile()**.
 - L'écart interquartile (IQR) a été déterminé pour identifier les bornes inférieure et supérieure des valeurs normales.
 - Les lignes dont les valeurs "close" étaient en dehors de ces bornes (définies comme $Q1 - 1.5 * IQR$ et $Q3 + 1.5 * IQR$) ont été filtrées pour éliminer les valeurs aberrantes.

Cette étape de prétraitement a permis d'améliorer la qualité des données, réduisant le bruit et les anomalies susceptibles d'affecter négativement la performance des modèles de Machine Learning.

```
df.count()

▼ (3) Spark Jobs
  ▶ Job 14 View (Stages: 1/1)
  ▶ Job 15 View (Stages: 1/1, 1 skipped)
  ▶ Job 16 View (Stages: 1/1, 2 skipped)

2056739
```

Figure 23. Taille du dataset après la phase du prétraitement

Lorsque vous exécutez des opérations telles que « **union** » ou « **filter** » dans Spark, vous créez une transformation sur les DataFrames sans déclencher immédiatement une action comme on a déjà discuté dans le chapitre précédent. Les transformations telles que « **union** » ne sont pas évaluées immédiatement mais sont ajoutées au plan **DAG** qui représente la séquence d'opérations à exécuter.

Cependant, lorsque vous exécutez une action comme « **count** », vous déclenchez l'évaluation des transformations et le calcul du résultat. C'est à ce moment-là que Spark construit le plan DAG complet et l'exécute sur le cluster pour obtenir le résultat.

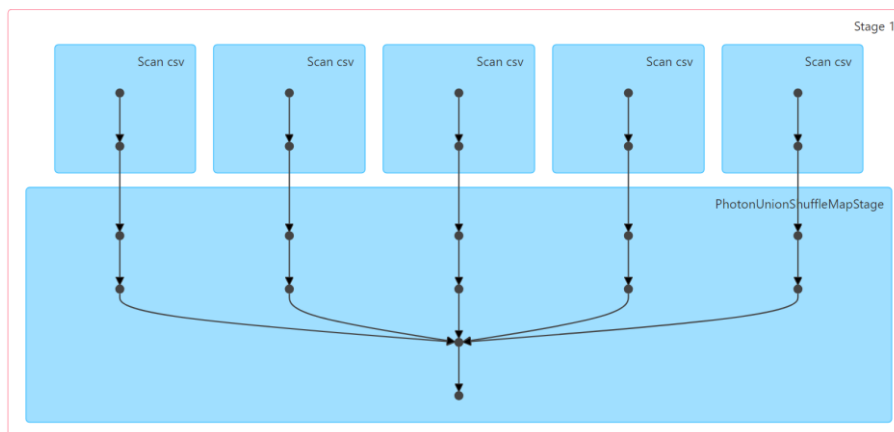


Figure 24. DAG exemple de la transformations Union.

3.5 Transformation des données

3.5.1 Extraction des caractéristiques temporelles

Pour exploiter les informations temporelles contenues dans nos données, nous avons transformé et extrait plusieurs caractéristiques à partir de la colonne **date**. Ces caractéristiques incluent l'heure, le jour de la semaine, le mois et l'année. Ces transformations permettent d'améliorer la précision des modèles de prédiction en capturant les tendances et les patterns temporels.

```
# Conversion et extraction des caractéristiques temporelles
df = df.withColumn("date", to_timestamp(col("date"), "yyyy-MM-dd HH:mm:ss"))
df = df.withColumn("hour", hour(col("date")))
df = df.withColumn("dayofweek", dayofweek(col("date")))
df = df.withColumn("month", month(col("date")))
df = df.withColumn("year", year(col("date")))
df = df.drop("date", "symbol")
```

Figure 25. Extraction des caractéristiques temporelles

Après cette transformation les colonnes de notre data frame devient

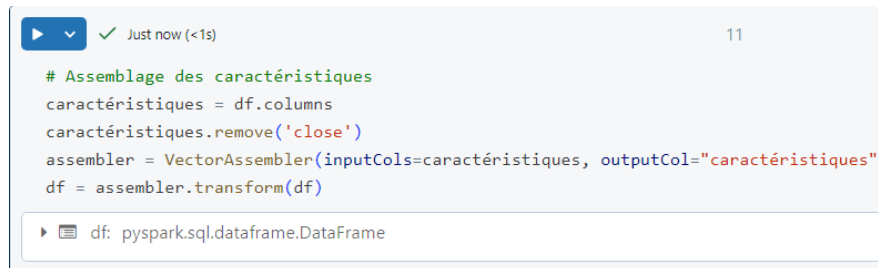
```
df.printSchema()

root
|-- unix: integer (nullable = true)
|-- open: double (nullable = true)
|-- high: double (nullable = true)
|-- low: double (nullable = true)
|-- close: double (nullable = true)
|-- Volume BTC: double (nullable = true)
|-- Volume USD: double (nullable = true)
|-- hour: integer (nullable = true)
|-- dayofweek: integer (nullable = true)
|-- month: integer (nullable = true)
|-- year: integer (nullable = true)
```

Figure 26. Nouvelle schéma du Data Frame

3.5.2 Assemblage des caractéristiques

En Spark, les algorithmes de traitement de données et de ML fonctionnent généralement avec des vecteurs plutôt qu'avec des données brutes. Par conséquent, avant d'appliquer des algorithmes sur un DataFrame, il est nécessaire de convertir les données brutes en vecteurs, car Spark ne comprend pas directement les données brutes mais peut comprendre et manipuler des vecteurs. C'est là que le **VectorAssembler()** entre en jeu : il transforme les caractéristiques individuelles en un format que Spark peut traiter efficacement pour l'analyse et l'apprentissage automatique.



```
# Assemblage des caractéristiques
caractéristiques = df.columns
caractéristiques.remove('close')
assembler = VectorAssembler(inputCols=caractéristiques, outputCol="caractéristiques")
df = assembler.transform(df)
```

df: pyspark.sql.dataframe.DataFrame

Figure 27. Assemblage des caractéristiques

Après l'exécution de ce code, toutes les colonnes du DataFrame sont listées la variable '**caractéristiques**', à l'exception de la colonne close qui est la variable **cible**. Un 'VectorAssembler' est ensuite créé avec ces colonnes comme entrées (**inputCols**) et la colonne de sortie est nommée "caractéristiques".

Le DataFrame est transformé en utilisant cet assembler, ajoutant ainsi une nouvelle colonne "caractéristiques" contenant des vecteurs regroupant les valeurs des colonnes spécifiées.

```
+-----+
| caractéristiques |
+-----+
|[1.51476462E9,139...|
|[1.51475748E9,139...|
|[1.51474602E9,138...|
|[1.51474056E9,137...|
|[1.51471308E9,126...|
|[1.5147123E9,1278...|
```

Figure 28. La nouvelle colonne des caractéristiques

3.5.3 La standardisation des caractéristiques

Dans cette étape, nous avons standardisé les caractéristiques présentes dans notre DataFrame. La standardisation est une technique qui permet de rendre les caractéristiques comparables en les mettant sur une même échelle. Pour cela, nous avons utilisé la classe 'StandardScaler' de PySpark.

```
# Standardiser les caractéristiques
scaler = StandardScaler(inputCol="caractéristiques", outputCol="features", withMean=True, withStd=True)
df = scaler.fit(df).transform(df)
```

Figure 39. Standardisation des caractéristiques.

La standardisation des caractéristiques vise à :

- Centrer les données en soustrayant la moyenne de chaque caractéristique.
- Mettre à l'échelle les données en divisant chaque caractéristique par son écart-type.

Cette opération permet d'améliorer la performance et la convergence des algorithmes de Machine Learning.

Chapitre III : Modélisation et Résultats

3.6 Modélisation

3.6.1 Les algorithmes LR, RF, DT, GBTs.

3.6.1.1 Séparation des données en ensembles d'entraînement et de test et choix des modèles

```
#Séparation des données en ensembles d'entraînement et de test
train_df, test_df = df.randomSplit([0.8, 0.2], seed=42)

# Liste des algorithmes
algorithmes = ['Régression Linéaire', 'Arbre de Décision', 'Forêt Aléatoire', 'Gradient Boosting']

# Liste des modèles correspondants aux algorithmes
modeles = [LinearRegression(featuresCol='features', labelCol='close'),
            DecisionTreeRegressor(featuresCol='features', labelCol='close'),
            RandomForestRegressor(featuresCol='features', labelCol='close'),
            GBRegressor(featuresCol='features', labelCol='close')]
```

Figure 30. Séparation des données en ensembles d'entraînement et de test.

La division des données en ensembles d'entraînement et de test est une étape cruciale dans la construction et l'évaluation de modèles prédictifs. Cela permet de :

- **Entraîner** le modèle sur une portion des données (train_df), afin qu'il puisse apprendre les relations sous-jacentes entre les caractéristiques et les cibles.
- **Tester** le modèle sur une portion distincte des données (test_df), pour évaluer sa capacité à généraliser ses prédictions sur des données qu'il n'a pas vues auparavant. Cela aide à détecter les problèmes de surapprentissage (overfitting).

3.6.1.2 Entraînement du modèle

```
# Boucle pour entraîner les modèles et obtenir les prédictions
for modele, nom in zip(modeles, algorithmes):
    start_time = time.time()
    modele_entraine = modele.fit(train_df)
    temps_entrainement = time.time() - start_time

    start_time = time.time()
    predictions = modele_entraine.transform(test_df)
    temps_prediction = time.time() - start_time
```

Figure 31. Séparation des données en ensembles d'entraînement et de test.

La boucle parcourt chaque modèle et son nom associé. Pour chaque modèle, elle enregistre le temps avant et après l'entraînement pour calculer le temps total d'entraînement (temps_entrainement). Ensuite, elle enregistre le temps avant et après la prédiction pour calculer le temps total de prédiction (temps_prediction). Les prédictions sont stockées dans la variable predictions.

3.6.1.3 Évaluation du modèle

Utiliser l'ensemble de test pour évaluer la performance du modèle. Les métriques d'évaluation incluront :

- **RMSE (Root Mean Squared Error)** : La racine carrée de l'erreur quadratique moyenne, qui mesure la différence entre les valeurs prédites par le modèle et les valeurs réelles.

- **MAE (Mean Absolute Error)** : L'erreur absolue moyenne, qui calcule la moyenne des différences absolues entre les valeurs prédites et les valeurs réelles.
- **R² (Coefficient de détermination)** : Une mesure statistique de la proportion de la variance dans la variable dépendante qui est prédite par les variables indépendantes.

```
# Calcul des métriques de performance
rmse = RegressionEvaluator(labelCol="close", predictionCol="prediction", metricName="rmse").evaluate(predictions)
mae = RegressionEvaluator(labelCol="close", predictionCol="prediction", metricName="mae").evaluate(predictions)
r2 = RegressionEvaluator(labelCol="close", predictionCol="prediction", metricName="r2").evaluate(predictions)
```

Figure 32. Les métriques d'évaluation.

3.6.2 L'algorithme XGBOOST

XGBoost nécessite une préparation des données légèrement différente par rapport aux autres algorithmes de Spark MLlib en raison des raisons suivantes :

- **Format des données** : XGBoost ne travaille pas directement avec les DataFrames Spark. Il nécessite que les données soient sous forme de matrices NumPy ou de DataFrames Pandas. Les algorithmes de Spark MLlib, quant à eux, travaillent directement avec les DataFrames Spark où les caractéristiques (features) sont généralement stockées sous forme de vecteurs dans une seule colonne.
- **Interface d'entraînement et de prédiction** : La manière d'entraîner un modèle et de faire des prédictions avec XGBoost diffère légèrement. Par exemple, avec XGBoost, vous devez appeler explicitement les méthodes fit pour l'entraînement et predict pour les prédictions, en utilisant les données sous forme de matrices NumPy ou de DataFrames Pandas. Avec Spark MLlib, les méthodes d'entraînement et de prédiction sont intégrées dans l'API des DataFrames Spark.

Voilà comment on peut implémenter le model xgboost :

```
#Modèle XGBoost
#Préparation des données
# Convertir les données Spark en Pandas
train_pandas = train_df.toPandas()
test_pandas = test_df.toPandas()

# Préparer les matrices de caractéristiques et les cibles
X_train = np.array(train_pandas[features].values.tolist())
y_train = train_pandas['close']
X_test = np.array(test_pandas[features].values.tolist())
y_test = test_pandas['close']
#Construction et Entraînement du Modèle XGBoost
# Entraîner le modèle XGBoost
start_time = time.time()
xgb_model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100)
xgb_model.fit(X_train, y_train)
xgb_train_time = time.time() - start_time

# Faire des prédictions
start_time = time.time()
xgb_predictions = xgb_model.predict(X_test)
xgb_prediction_time = time.time() - start_time

# Calculer RMSE, MAE, R2
xgb_rmse = np.sqrt(mean_squared_error(y_test, xgb_predictions))
xgb_mae = mean_absolute_error(y_test, xgb_predictions)
xgb_r2 = r2_score(y_test, xgb_predictions)
```

Figure 33. L'implémentation du modèle XGBOOST.

3.7 Visualisation et rapport

3.7.1 Visualisation des données

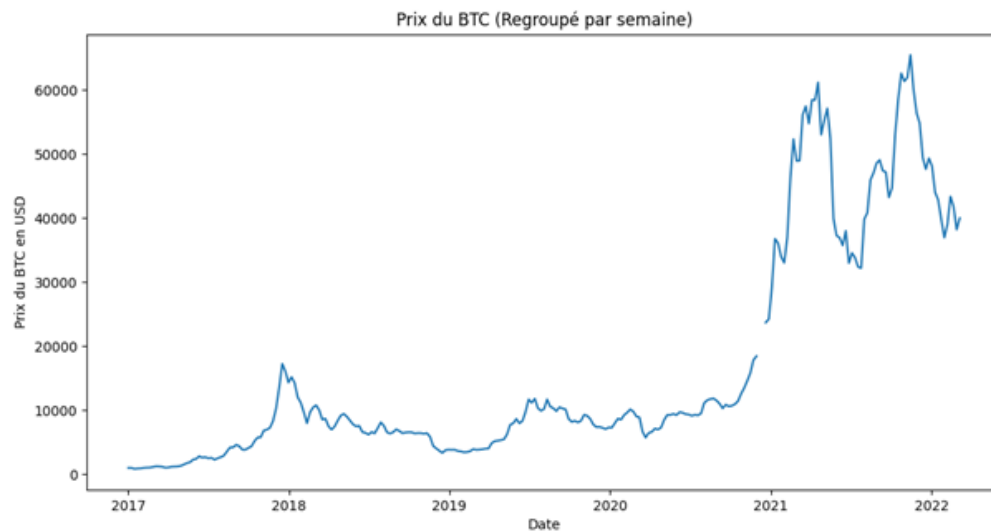


Figure 34. Le prix du BTC au fil du temps.

En examinant l'évolution du prix du Bitcoin au fil du temps, deux tendances majeures ressortent :

- **Une augmentation significative** : Le prix du bitcoin a connu une augmentation substantielle tout au long de cette période. Il a démarré aux alentours de 10 000 USD début 2017 et a atteint un pic d'environ 20 000 USD début 2018. À la fin de l'année 2021, il tournait autour de 60 000 USD.
- **Une forte volatilité** : Le prix a connu d'importantes fluctuations tout au long de cette période. Il y a eu de fortes hausses suivies de corrections et de périodes de consolidation.

3.7.1.1 Graphiques de distribution

Pour chaque variable continue, nous allons créer des histogrammes afin de visualiser la distribution des données.

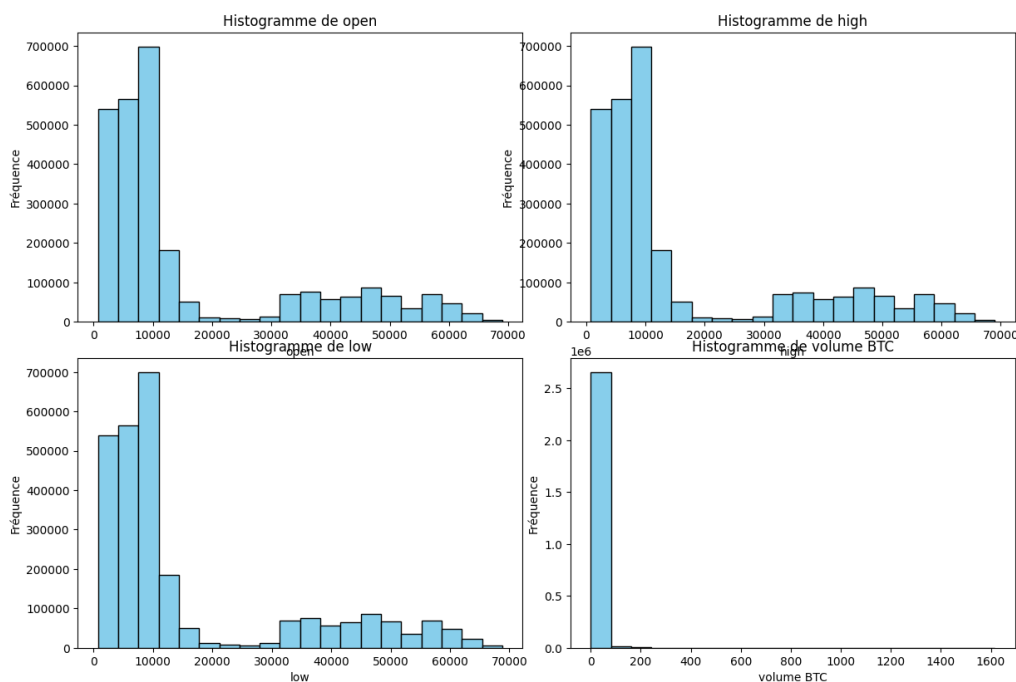


Figure 35. Le prix du BTC au fil du temps.

Les histogrammes permettent de visualiser la distribution des prix d'ouverture (Open), des prix maximums (High), des prix minimums (Low) et du volume de trading (Volume) du Bitcoin. Ils indiquent la fréquence à laquelle chaque plage de prix ou de volume a été observée.

Les distributions des prix d'ouverture, maximums et minimums sont **relativement similaires**, ce qui suggère que les prix du Bitcoin évoluaient généralement dans une fourchette relativement cohérente tout au long de la journée.

3.7.1.2 Matrice de corrélation

Nous allons créer une matrice de corrélation pour montrer les relations entre les variables numériques. Cela nous permettra d'identifier les relations linéaires entre ces variables.

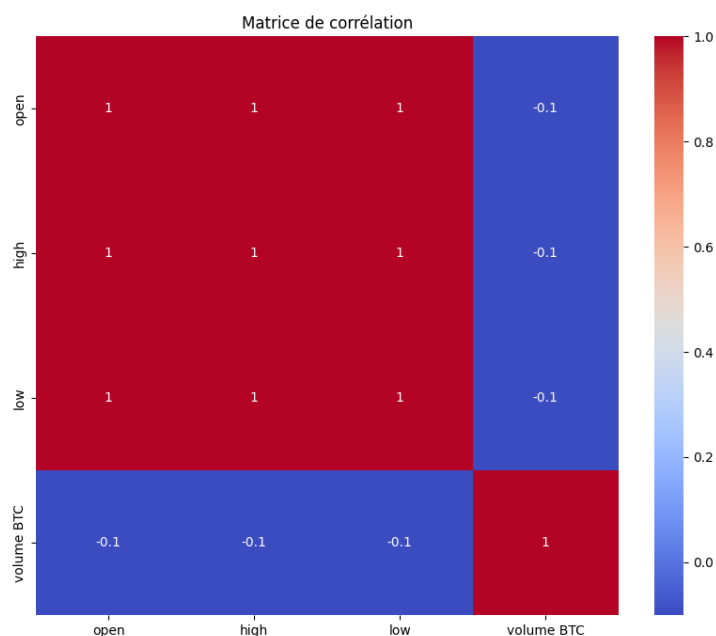


Figure 36. Matrice de corrélation.

D'après la matrice de corrélation, on observe des corrélations positives fortes entre les prix d'ouverture, le plus haut et le plus bas du Bitcoin. Cela signifie que ces variables ont tendance à évoluer dans la même direction. Par exemple, si le prix d'ouverture du Bitcoin est élevé, alors le prix le plus haut de la journée sera probablement élevé également, et le prix le plus bas sera vraisemblablement élevé lui aussi.

À l'inverse, la corrélation négative entre le volume et les trois autres variables suggère une relation inverse. Lorsque le volume de trading augmente fortement, les prix d'ouverture, maximum et minimum tendent à être inférieurs à la moyenne. Cela signifie que des périodes d'activité de trading élevée ne se traduisent pas nécessairement par des pics de prix plus élevés.

Cependant, il est important de reconnaître que le coefficient de corrélation de -0,1 entre le volume et les prix représente une faible corrélation négative. Cela suggère que la relation n'est pas toujours simple et n'implique pas un mouvement inverse strict. Divers facteurs peuvent influencer l'interaction entre le volume et le prix, nécessitant une analyse approfondie.

3.7.1.3 Nuages de points

Nous allons créer des nuages de points pour visualiser les relations entre les variables indépendantes et la variable dépendante

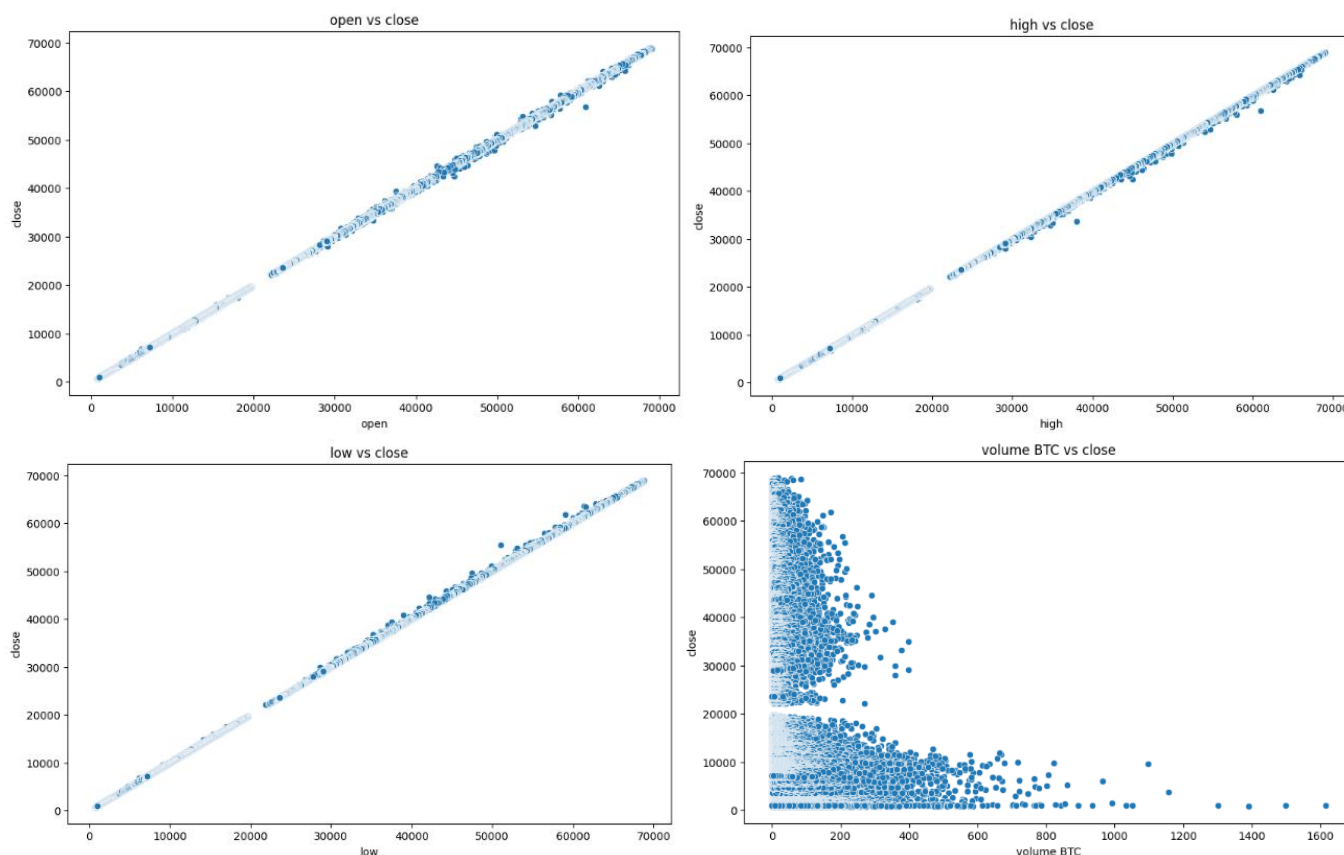


Figure 37. Le prix du BTC au fil du temps.

Les trois graphiques de nuage de points montrent une corrélation extrêmement forte et positive entre les prix d'ouverture (open), le plus haut (high), le plus bas (low) et le prix de clôture (close) du Bitcoin, illustrée par l'alignement quasi parfait des points le long d'une ligne droite pour chacun des graphiques. Cela signifie que les prix de clôture sont étroitement liés aux prix d'ouverture, aux plus hauts et aux plus bas, avec des écarts minimes entre eux, suggérant que le Bitcoin ne connaît pas de variations significatives au cours de ces périodes. Ces relations linéaires indiquent une stabilité relative des prix du Bitcoin tout au long de la journée, où les variations de prix entre l'ouverture, les plus hauts, les plus bas et la clôture sont proportionnelles et prévisibles. Par ailleurs, le quatrième graphique montre une corrélation négative entre le volume des transactions en Bitcoin (volume BTC) et le prix de clôture, bien que la faiblesse de cette corrélation suggère une relation non linéaire et complexe, influencée par divers autres facteurs du marché.

3.7.2 Visualisation des résultats de la modélisation :

Cette section est dédiée à la présentation des résultats obtenus par les différents modèles de régression appliqués.

3.7.2.1 Graphiques des performances des modèles

Cette section se concentre sur la représentation visuelle des performances des différents modèles de prédiction à travers des graphiques des valeurs prédites versus les valeurs réelles. Ces graphiques permettent de visualiser directement la précision des prédictions des modèles par rapport aux valeurs observées. Ces visualisations sont cruciales pour évaluer la performance des modèles et identifier les modèles qui produisent les prédictions les plus précises.

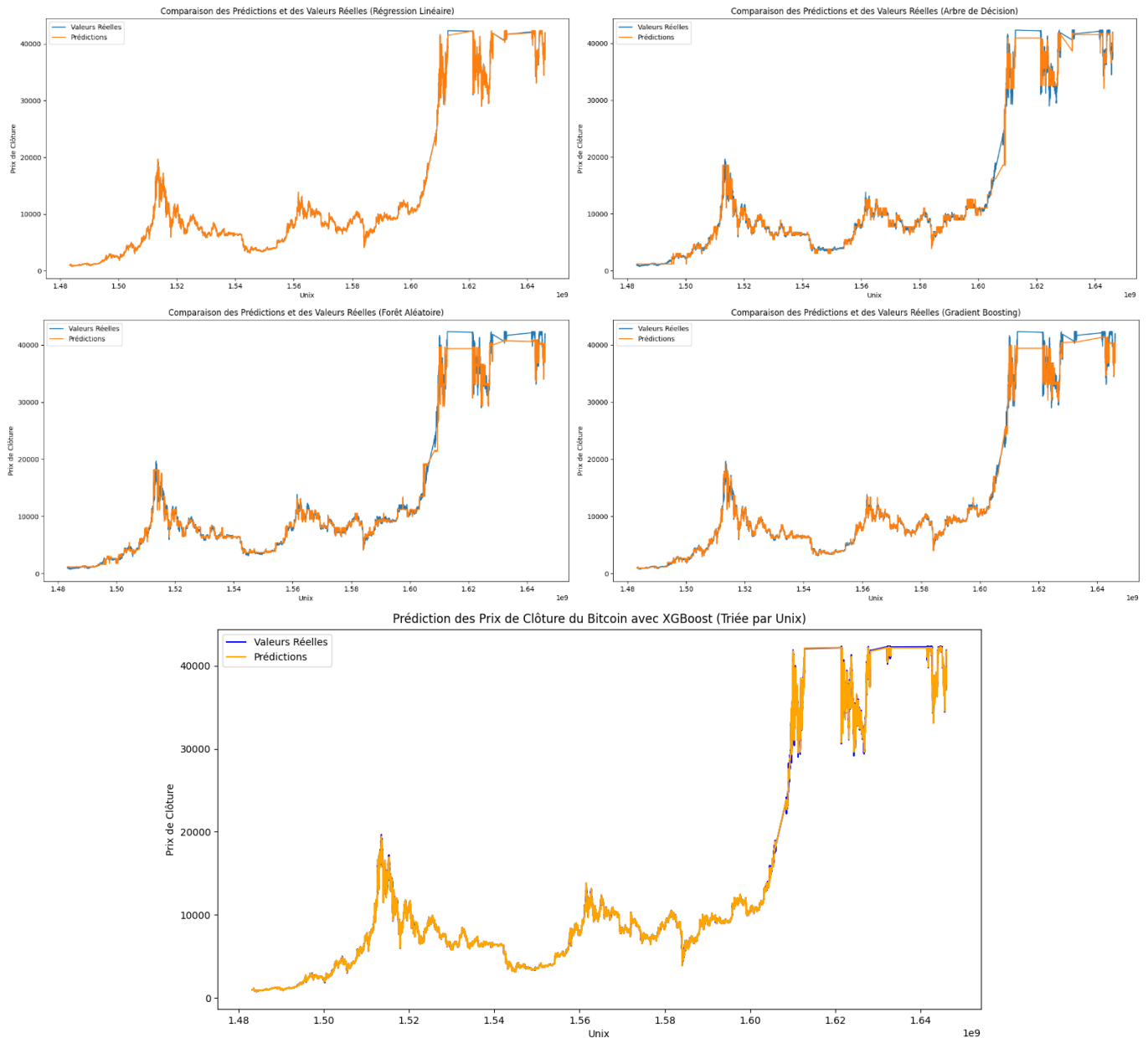


Figure 38. Les graphiques des valeur réelles et les valeurs prédites.

3.7.2.2 Comparaison des métriques

Dans cette section, les différentes métriques de performance des modèles sont comparées de manière détaillée. Cela inclut les métriques telles que l'erreur quadratique moyenne (RMSE), l'erreur absolue moyenne (MAE), le coefficient de détermination (R^2)

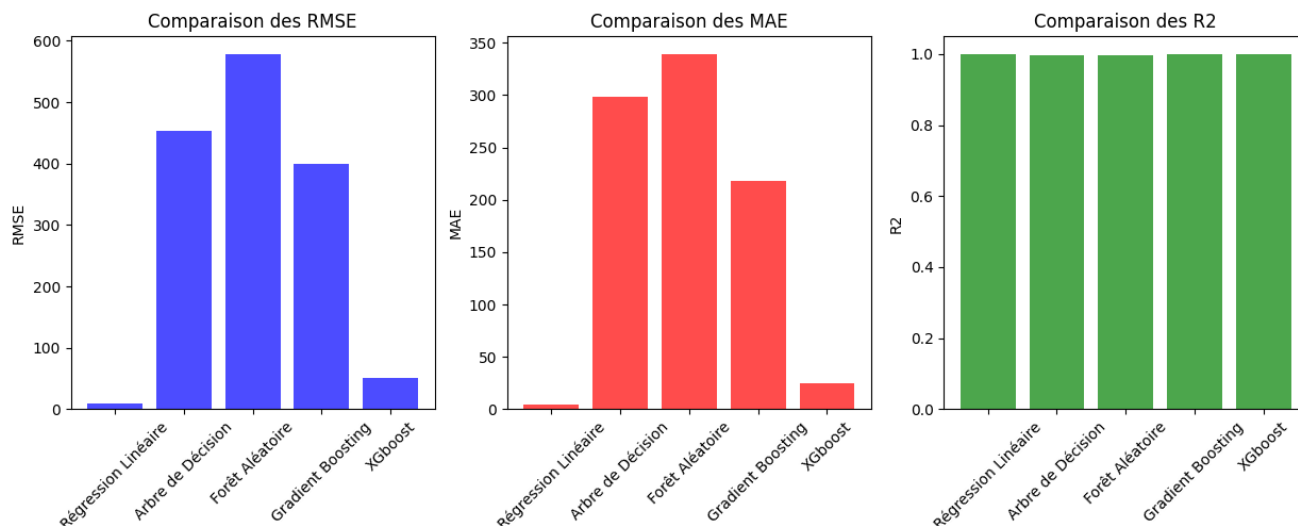


Figure 39. Les diagrammes à barres de chaque métrique d'évaluation.

Les résultats sont souvent présentés dans des diagrammes à barres pour visualiser clairement les différences de performance entre les modèles. Cette comparaison permet de déterminer quel modèle offre la meilleure performance globale et dans quelles conditions chaque modèle est le plus efficace.

4 Mise en œuvre

4.1 Environnement de développement

- **Logiciels et frameworks:**
 - Spark 3.4.1
 - Databricks Runtime 13.3 LTS
 - Scala 2.12
 - Databricks Notebook
- **La configuration du notre cluster :**

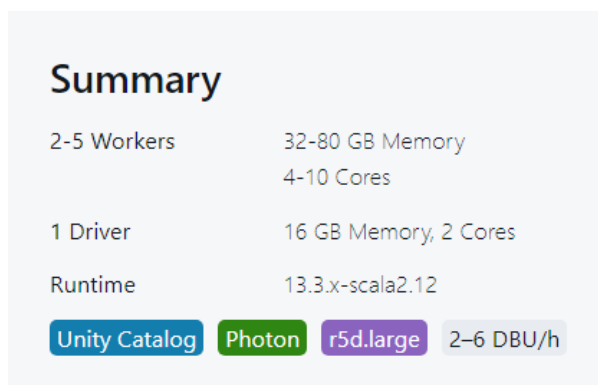


Figure 40. Configuration du cluster.

- La configuration comprend entre 2 et 5 travailleurs, chacun disposant de 32 à 80 Go de mémoire et de 4 à 10 cœurs. Le driver, quant à lui, est doté de 16 Go de mémoire et de 2 cœurs. L'environnement d'exécution utilisé est la version 13.3.x-scala2.12.
- Catalogues et options :

- **Unity Catalog** : Offre une gestion unifiée des métadonnées pour toutes les données, améliorant la gouvernance et la sécurité.
- **Photon** : Un moteur de requêtes compatible avec Apache Spark, optimisé pour offrir de meilleures performances.
- **r5d.large** : Type d'instance AWS EC2 avec des disques NVMe locaux pour des performances de stockage élevées.
- **2-6 DBU/h** : Unités de Databricks utilisées pour mesurer les coûts de traitement. Le coût varie en fonction de l'utilisation des ressources.

4.2 Défis techniques rencontrés

Lors de la mise en œuvre du système, nous avons été confrontés à plusieurs défis techniques de taille. Tout d'abord, notre machine locale s'est avérée insuffisante pour exécuter efficacement des tâches complexes, ce qui a rapidement mis en évidence la nécessité de passer à une infrastructure de calcul plus robuste, à savoir un cluster. Cependant, migrer vers un cluster n'est pas sans ses propres défis. Le principal obstacle rencontré était lié aux coûts associés à l'utilisation d'un cluster, en particulier lorsqu'il est hébergé sur une plateforme comme AWS. Les frais d'utilisation des ressources peuvent rapidement s'accumuler, ce qui a exigé de nous une approche prudente et stratégique pour gérer nos dépenses tout en répondant à nos besoins de traitement de données.

Pour faire face à cette situation, nous avons initialement exploré les essais gratuits proposés par différentes plateformes de cloud, y compris AWS et Microsoft Azure. Ces essais gratuits nous ont permis d'expérimenter les fonctionnalités et les performances des clusters sans supporter de coûts initiaux élevés. Cependant, il est important de noter que ces essais gratuits sont souvent assortis de limitations en termes de ressources disponibles, ce qui a posé des défis supplémentaires en matière de scalabilité et de gestion de la mémoire.

4.3 Solutions adoptées

Pour surmonter les défis techniques rencontrés, nous avons adopté plusieurs solutions méthodologiques et techniques. En réponse aux limitations des essais gratuits, nous avons choisi d'utiliser Databricks, qui propose des essais gratuits de 14 jours avec une configuration de cluster suffisante pour nos besoins initiaux. En outre, nous avons mis en œuvre des techniques d'optimisation spécifiques à Spark pour améliorer les performances du système. Cela inclut le tuning des paramètres de Spark et le partitionnement des données pour une meilleure répartition de la charge de travail. Ces optimisations nous ont permis d'améliorer significativement la scalabilité et la gestion de la mémoire, rendant notre système plus performant et stable.

5 Résultats

5.1 Analyse des résultats

5.1.1 Analyse des performances

En traçant les cinq courbes du (**Figure 39**) des valeurs prédites versus les valeurs réelles dans un même graphique, cela offre une visualisation directe de la performance comparative des différents modèles de prédiction. Cette approche permet de comparer facilement les prédictions de chaque modèle avec les valeurs réelles de manière simultanée, ce qui facilite l'évaluation de la précision des prédictions.

Il est important de noter qu'on n'a pas tracé l'intégralité des données historiques, mais on a plutôt choisi une seule journée pour une meilleure visualisation. En se concentrant sur une journée spécifique (**01-01-2018 de 00 :00 :00 à 00 :30 :00**), cela permet de réduire la complexité visuelle du graphique tout en mettant en évidence les performances relatives des différents modèles pour cette journée précise.

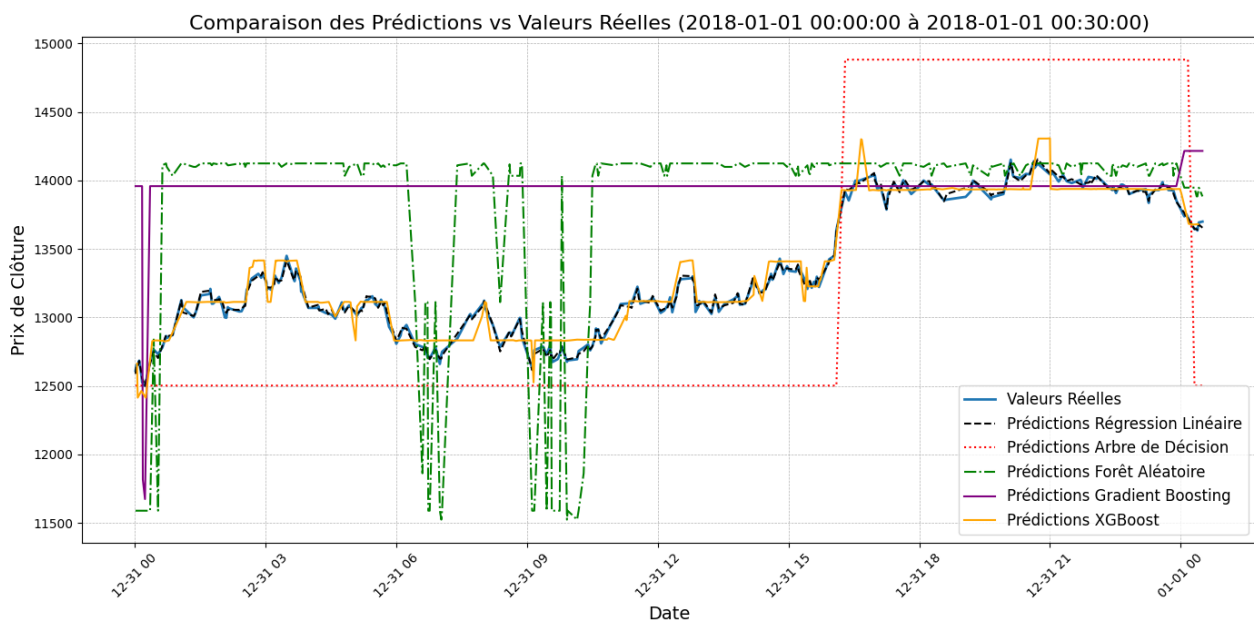


Figure 41. Les graphiques des valeur réelles et les valeurs prédites des 5 modèles.

En observant le graphique, il est évident que les modèles XGBoost et de régression linéaire sont les plus performants. Leurs prédictions se rapprochent le plus des valeurs réelles et suivent la même tendance que ces dernières.

La régression linéaire excelle souvent lorsque les relations entre les variables sont linéaires, comme l'illustrent les graphiques de nuages de points (**Figure 38**) où les prix d'ouverture, les plus hauts, les plus bas et les prix de clôture du Bitcoin sont alignés de manière presque parfaite le long d'une ligne droite. Cette capacité à modéliser correctement les tendances linéaires permet à la régression linéaire de produire des prédictions précises.

D'autre part, **XGBoost**, en tant que méthode d'ensemble basée sur des arbres de décision, est réputé pour sa capacité à capturer des relations non linéaires entre les variables. Cette flexibilité est essentielle dans des domaines comme la prédiction financière, où les relations peuvent être complexes et non linéaires. De plus, XGBoost est adapté à la gestion de grands ensembles de données complexes, ce qui est courant dans le domaine financier.

En revanche, les autres algorithmes présentent des performances inférieures dans ce contexte. Par exemple, **l'arbre de décision et la forêt aléatoire** produisent des prédictions rectangulaires périodiques qui ne prennent pas en compte la complexité des données financières.

5.1.2 Analyse des erreurs

Le graphique de la (**Figure 39**) montre l'erreur quadratique moyenne (MSE), l'erreur absolue moyenne (MAE) et les scores R^2 pour chaque modèle. Plus le MSE et le MAE sont bas, meilleure est la performance du modèle. Plus le score R^2 est élevé, meilleure est l'adéquation du modèle aux données.

- Performance globale
 - D'après le graphique, il est clair que les modèles XGBoost et Régression Linéaire sont les meilleurs modèles. Ils ont les scores MSE et MAE les plus bas, ainsi que les scores R^2 les plus élevés.
 - Les modèles Arbre de Décision, Forêt Aléatoire et Gradient Boosting sont moins performants. Ils ont des scores MSE et MAE plus élevés, ainsi que des scores R^2 plus bas.
- Performance des modèles individuels
 - **Régression Linéaire** : La Régression Linéaire est un modèle performant. Elle affiche un RMSE de 6.147 et un MAE de 3.234, ce qui indique une précision élevée dans ses prédictions. De plus, son coefficient de détermination R^2 est très proche de 1, ce qui signifie que le modèle explique presque parfaitement la variance des données.
 - **XGBoost** : XGBoost se démarque comme un modèle de premier plan avec un RMSE de 29.734 et un MAE de 15.079, ce qui sont des scores légèrement moins performants que ceux de la régression linéaire. Cependant, son coefficient de détermination R^2 reste très élevé, ce qui confirme une bonne adéquation entre les prédictions du modèle et les valeurs réelles.
 - **Arbre de Décision** : Le modèle Arbre de Décision est le moins performant des cinq modèles. Il a les scores MSE et MAE les plus élevés, ainsi que le score R^2 le plus bas.
 - **Forêt Aléatoire** : Le modèle Forêt Aléatoire est légèrement meilleure que le modèle Arbre de Décision. Il a des scores MSE et MAE légèrement plus bas, ainsi qu'un score R^2 légèrement plus élevé.
 - **Gradient Boosting** : Le modèle Gradient Boosting est légèrement meilleur que le modèle Forêt Aléatoire. Il a des scores MSE et MAE légèrement plus bas, ainsi qu'un score R^2 légèrement plus élevé.
- Conclusion

Sur la base de cette analyse, les modèles XGBoost et Régression Linéaire sont les meilleurs choix pour prédire les prix des actions. Les modèles Arbre de Décision, Forêt Aléatoire et Gradient Boosting sont moins performants, mais ils peuvent encore être utiles pour d'autres tâches.

5.2 Comparaison avec d'autres approches

Nous allons comparer les performances de notre système développé en utilisant différents algorithmes de machine learning dans deux contextes différents : un cluster Spark et une machine locale mais en utilisant le même code avec scikit-learn (sklearn). Les critères de comparaison incluent le temps d'exécution et les performances des algorithmes, mesurées par le RMSE, le MAE et le coefficient de détermination R^2 .

La configuration de la machine locale est la suivante :

- Processeur : i7-1065G7 1.30 GHz
- Mémoire : 8 Go 3200Mhz
- Disque dur :500 GB SSD NVMe.

Les résultats obtenus sont les suivantes :

```
Régression Linéaire - RMSE: 15.26521174531684
MAE: 6.377286967193567
R2: 0.9999993499345984
Training Time: 1.403620958328247s
Prediction Time: 0.02601027488708496s
Arbre de Décision - RMSE: 21.54079376257935
MAE: 8.387733660026152
R2: 0.9999987055812595
Training Time: 80.88173794746399s
Prediction Time: 0.5948262214660645s
Forêt Aléatoire - RMSE: 16.650373303459535
MAE: 6.610918924184157
R2: 0.9999992266086037
Training Time: 5925.115368127823s
Prediction Time: 272.00717973709106s
Gradient Boosting - RMSE: 85.99575923831375
MAE: 57.850165025468726
R2: 0.9999793697041095
Training Time: 1650.6199834346771s
Prediction Time: 2.1559176445007324s
XGBoost - RMSE: 76.56210061160753
MAE: 42.96313706041271
R2: 0.9999836476916568
Temps d'Entraînement: 9.382939100265503 s
Temps de Prédiction: 0.439228126525879 s
```

Figure 42. Résultats d'une machine Locale.

5.2.1 Comparaison des temps d'exécution

Algorithme	Machine locale (sklearn)		Cluster Spark	
	Entrainment (s)	Prediction (s)	Entrainment (s)	Prediction (s)
Régression linéaire	1.4036	0.0260	14.6476	0.0252
Arbre de Décision	80.8817	0.5948	10.5544	0.0229
Forêt Aléatoire	5925.1153	272.0071	13.7217	1.5246
Gradient Boosting	1650.6199	2.1559	35.9342	0.5741
XGBoost	9.3829	0.4392	4.8192	0.2334

Table 2. Comparaison du temps d'exécution des algorithmes, cluster vs machine locale.

Comme le montre le tableau, les temps d'exécution sont généralement plus courts sur le cluster Spark, grâce à la capacité de Spark à paralléliser les tâches de calcul sur plusieurs nœuds. Cela est particulièrement avantageux pour des algorithmes complexes et gourmands en ressources comme la forêt aléatoire, gradient boosting et XGBoost.

5.2.2 Comparaison des performances des algorithmes

Algorithme	Machine locale (sklearn)			Cluster Spark		
	RMSE	MAE	R ²	RMSE	MAE	R ²
Régression linéaire	15.2652	6.3772	0.99999934	6.1472	3.2344	0.999997
Arbre de Décision	21.5407	8.3877	0.9999987	454.7799	197.5155	0.9876
Forêt Aléatoire	16.6503	6.6109	0.99999922	536.5295	198.0316	0.9828
Gradient Boosting	85.9957	57.8501	0.999979	264.4708	135.0041	0.9958
XGBoost	76.5621	42.9631	0.999983	29.7344	15.0787	0.999947

Table 3. Comparaison performances des algorithmes, cluster vs machine locale.

En termes de performances, les résultats montrent que les différences entre l'utilisation de scikit-learn sur une machine locale et Spark dans un cluster sont significatives dans certains cas. Les algorithmes de Régression Linéaire et XGBoost présentent de meilleures performances sur le cluster Spark en termes de RMSE et MAE.

Les algorithmes d'arbre de décision, la forêt aléatoire et Gradient Boosting montrent des erreurs plus élevées (RMSE et MAE) lorsqu'ils sont exécutés sur un cluster Spark. Voici quelques raisons possibles pour ces différences :

- **Parallélisation et Distribution des Données :**

Lors de l'utilisation de Spark, les données sont réparties sur plusieurs nœuds du cluster. Cette fragmentation peut parfois entraîner une perte d'information contextuelle qui est plus facilement disponible lorsque les données sont traitées sur une seule machine. La communication entre les nœuds d'un cluster peut introduire des latences et des décalages, affectant ainsi les performances des algorithmes qui nécessitent des calculs séquentiels ou synchronisés.

- **Complexité des Algorithmes :**

Les algorithmes Arbre de Décision et Forêt Aléatoire dépendent fortement des interrelations complexes entre les données. La distribution des données sur plusieurs nœuds peut affecter la construction de ces arbres, entraînant des performances dégradées. Bien que les algorithmes Gradient Boosting et XGBoost soient conçus pour être parallélisés, ils bénéficient souvent d'un traitement plus cohérent lorsqu'ils sont exécutés sur une seule machine, en raison de la nécessité de plusieurs itérations et d'étapes de calibration.

5.2.3 Discussion des avantages et des inconvénients

Avantages du système développé avec Spark dans un cluster :

- **Scalabilité :** Spark permet de traiter de grandes quantités de données en parallèle sur plusieurs nœuds, ce qui réduit considérablement le temps d'exécution pour les algorithmes complexes.
- **Efficacité :** Le traitement distribué de Spark optimise l'utilisation des ressources et améliore l'efficacité globale du système.
- **Gestion des Big Data :** Spark est mieux adapté à la manipulation de grands ensembles de données qui dépasseraient les capacités d'une machine locale.

Inconvénients :

- **Complexité de déploiement :** La configuration et la gestion d'un cluster Spark peuvent être complexes et nécessitent une expertise technique avancée.
- **Coût :** Utiliser un cluster de calcul peut être plus coûteux en termes de ressources matérielles et de maintenance comparé à une machine locale.

Avantages du système développé avec scikit-learn sur une machine locale :

- **Simplicité :** La mise en œuvre de modèles avec scikit-learn est simple et rapide, idéal pour des tâches de machine learning à petite échelle.
- **Coût :** L'utilisation d'une machine locale est généralement moins coûteuse et plus facile à gérer que l'infrastructure d'un cluster.

Inconvénients :

- **Limites de scalabilité :** Les performances peuvent être limitées par les ressources de la machine locale, ce qui peut être un obstacle pour traiter de grandes quantités de données.
- **Temps d'exécution :** Pour des tâches complexes, le temps d'exécution peut être significativement plus long sur une machine locale par rapport à un cluster Spark.

En conclusion, le choix entre l'utilisation de Spark dans un cluster et scikit-learn sur une machine locale dépend des besoins spécifiques du projet, notamment en termes de volume de données, de ressources disponibles et de budget. Notre système développé montre que les deux approches ont leurs avantages et peuvent être utilisées de manière complémentaire selon les circonstances.

Conclusion et perspectives

En conclusion, ce rapport offre une exploration exhaustive de la conception, du développement et de l'évaluation d'un système de prédiction avancé reposant sur Apache Spark et les algorithmes de Machine Learning. À travers une démarche méthodique, nous avons scruté les fondements théoriques sous-tendant notre projet, détaillé l'architecture complexe du système, et finalement, évalué ses performances à travers diverses expériences de modélisation.

Nos résultats confirment non seulement la fiabilité mais aussi l'efficacité de cette approche dans le traitement rapide et précis de vastes ensembles de données. Cependant, dans notre quête vers l'excellence, nous ne pouvons ignorer les défis et les opportunités qui se profilent à l'horizon.

Parmi les pistes d'amélioration et les axes de recherche pour l'avenir, nous envisageons plusieurs voies prometteuses :

- Poursuivre l'exploration de nouvelles techniques et algorithmes de Machine Learning afin d'optimiser davantage les performances du système de prédiction. Cette quête incessante d'amélioration continue d'être un moteur crucial pour rester à la pointe de l'innovation.
- L'intégration de techniques avancées de traitement du langage naturel (NLP) représente une opportunité majeure pour élargir la portée du système et traiter efficacement les données non structurées et textuelles, ouvrant ainsi la voie à de nouvelles applications et insights.
- Étendre la fonctionnalité du système pour prendre en charge le traitement en temps réel des flux de données en continu constitue une exigence de plus en plus pressante dans un monde où la rapidité des décisions est essentielle. Cela nous permettrait de fournir des prédictions en temps réel, ouvrant de nouvelles opportunités dans des domaines tels que la finance, le commerce électronique et la logistique.
- Améliorer la scalabilité et la résilience du système est impératif pour répondre aux exigences croissantes en termes de volume de données et pour s'adapter à des environnements dynamiques. Cela nécessitera des efforts continus pour optimiser les performances du système et garantir sa stabilité à grande échelle.

En investissant dans ces domaines et en exploitant les avancées technologiques à venir, nous pouvons non seulement renforcer les capacités des entreprises et des organisations à exploiter leurs données de manière stratégique, mais également ouvrir de nouvelles perspectives pour l'innovation et la croissance future. En fin de compte, ce rapport ne marque pas la fin de notre voyage, mais plutôt le début d'une nouvelle ère d'exploration et de progrès dans le domaine passionnant de l'analyse prédictive et du Machine Learning

Bibliographie

- [1] Im. DAHANE and Sa. MEDINE, “L’analyse prédictive dans un contexte de Big Data,” 2020.
- [2] B. Chambers and M. Zaharia, *Spark: The definitive guide: Big data processing made simple*. O’Reilly Media, Inc., 2018.
- [3] T. White, *Hadoop: The definitive guide*. O’Reilly Media, Inc., 2012.
- [4] Z.-H. Zhou, *Machine learning*. Springer nature, 2021.
- [5] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.
- [6] T. Chen *et al.*, “Xgboost: extreme gradient boosting,” *R Package Version 04-2*, vol. 1, no. 4, pp. 1–4, 2015.

Webographie

- <https://www.veonum.com/apache-spark-pour-les-nuls/>, consulté le 11 mai 2024.
- <https://www.databricks.com/fr/glossary/what-is-apache-spark#:~:text=Le%20moteur%20Spark%20Core%20utilise,au%20regard%20de%20l'utilisateur>, consulté le 13 mai 2024.
- <https://www.edureka.co/blog/spark-architecture/>, consulté le 13 mai 2024.
- <https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/>, consulté le 13 mai 2024.
- <https://data-flair.training/blogs/spark-rdd-operations-transformations-actions/> consulté le 14 mai 2024.
- https://www.researchgate.net/figure/Transformations-and-actions-in-Apache-Spark_tbl1_274716564, consulté le 14 mai 2024.
- <https://hadoop.apache.org/>, consulté le 14 mai 2024.
- <https://www.databricks.com/glossary/hadoop-distributed-file-system-hdfs#:~:text=Hadoop%20HDFS%20%2D%20Hadoop%20Distributed%20File,resource%20management%20component%20of%20Hadoop>, consulté le 14 mai 2024.
- [https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm#:~:text=The%20MapReduce%20algorithm%20contains%20two,\(key%2Fvalue%20pairs\)](https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm#:~:text=The%20MapReduce%20algorithm%20contains%20two,(key%2Fvalue%20pairs)), consulté le 14 mai 2024.
- <https://www.next-decision.fr/wiki/les-bonnes-pratiques-spark#:~:text=Spark%20est%20jusqu'%C3%A0%20100,m%C3%A9moire%20est%20donc%20fortement%20acc%C3%A9l%C3%A9r%C3%A9>, consulté le 15 mai 2024.
- <https://www.oracle.com/ch-fr/artificial-intelligence/machine-learning/what-is-machine-learning/>, consulté le 15 mai 2024.
- <https://www.ibm.com/topics/machine-learning>, consulté le 15 mai 2024.
- <https://www.mathworks.com/discovery/machine-learning-models.html>, consulté le 17 mai 2024.
- <https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea>, consulté le 20 mai 2024