الجمهورية الجزائرية الديمقراطية الشعبية
Université Abou Bekr Belkaid -Tlemcen-
Faculté des Sciences
Département Informatique

MEMOIRE

En vue d'obtention du diplôme de Master en informatique

Option : SIC
(Systèmes d'information et connaissances)

Présenté par
DJAZIA BERRAHMA

Application du Machine Learning dans l'optimisation des données massives issus du Web au sein de RDF_QDAG

Présenté le 26/09/2023 devant le jury composé de :

- M. Badr BENMAMMAR — Président (Université de Tlemcen)
- M. Sidi Mohammed CHOUITI — Examinateur (Université de Tlemcen)
- M. Houcine MATALLAH — Encadrant (Université de Tlemcen)
- M. Amin MESMOUDI — Encadrant (Université de Poitiers)

Remerciements

La réalisation de ce mémoire a été une formidable aventure au cours de laquelle plusieurs personnes m'ont apporté leur soutien. Sans leur aide, la tâche aurait été bien plus pénible. C'est alors tout naturellement que je souhaite les remercier.

Tout d'abord, je remercie mes encadrants M. Houcine MATALLAH pour son accompagnement exceptionnel tout au long de mon parcours académique, M. Amin MESMOUDI pour ses conseils avisés et sa patience, qui a été ma source d'inspiration et de motivation inestimable.

Un grand merci également au laboratoire LIAS, pour m'avoir accueilli chaleureusement durant mon stage.

Je remercie également M. Houssem YOUSFI, M. Boumediene SAIDI, M. Nadir GUERMOUDI, M. Soulimane KAMNI, d'avoir partagé vos connaissances, vos perspectives tout au long de cette aventure. Les échanges fructueux que nous avons eus ont éclairé mon parcours et m'ont inspiré à repousser mes limites.

Mes remerciements les plus chaleureux à mes enseignants qui m'ont guidé tout au long de ce voyage académique. Vos connaissances partagées et votre dévouement ont enrichi mon expérience d'apprentissage.

Djazia BERRAHMA

Dédicace

Je dédie ce mémoire, À mes parents, pour l'amour qu'ils m'ont toujours donné, leurs encouragements et toute l'aide qu'ils m'ont apportée durant mes études. Aucun mot, aucune dédicace ne pourrait exprimer mon respect, ma considération, et mon amour pour les sacrifices qu'ils ont consentis pour mon instruction et mon bien-être. Trouvez ici, chère mère et cher père, dans ce modeste travail, le fruit de tant de dévouement et de sacrifices ainsi que l'expression de ma gratitude et de mon profond amour.

À mon grand-père bien-aimé, Abdelhamid. Ta présence et ta sagesse ont été une source inestimable d'inspiration et de soutien tout au long de ma vie, y compris pendant mon parcours académique.

À mes chères sœurs, Sabrina et Ahlem, dont la présence et la complicité m'ont apporté réconfort et motivation à chaque étape de mon parcours.

À la mémoire de Islam pour son soutien indéfectible qu'il m'a offert, même dans les moments les plus exigeants de mes études. En souvenir de sa bienveillance et de son impact profond, je choisis humblement de lui dédier ce travail. Que cette dédicace soit un hommage respectueux à une personne qui restera à jamais dans nos cœurs.

Djazia BERRAHMA

Table des matières

Chapitre 1

Introduction

1.1 Contexte et Problématique

Dans un monde en constante évolution, la gestion efficace des données est devenue cruciale pour la prise de décision, la recherche, et l'innovation. Les graphes de connaissances, en particulier ceux basés sur le modèle RDF (Resource Description Framework), se sont révélés être un outil puissant pour organiser et exploiter les informations de manière sémantique. Ces graphes permettent de représenter des relations complexes entre les données, offrant ainsi un moyen de stocker et de récupérer des connaissances de manière structurée. Au cœur de cette démarche se trouve le système RDF_QDAG[KMG+21], qui se distingue par son approche innovante axée sur la fragmentation et l'exploration des graphes logiques liés aux données.

RDF_QDAG repose sur l'optimiseur GoFAst [ZMG+21], un composant essentiel de son architecture. GoFAst [ZMG+21] utilise un estimateur de coût basé sur des statistiques liées aux données pour optimiser le traitement des requêtes. Récemment, l'équipe de développement de RDF_QDAG a franchi une étape décisive en intégrant un moteur d'évaluation de requêtes spatiales. Cette évolution a ajouté une dimension cruciale au système, en proposant deux stratégies distinctes : RDF_First, qui permet de prendre en charge les opérateurs spatiaux sans perturber la structure fondamentale du système, et Spatial_First, qui exploite habilement des index de type R-Tree pour améliorer les performances des requêtes spatiales. L'optimiseur joue un rôle essentiel en aidant à choisir la meilleure stratégie d'exécution

pour chaque requête de données.

Cependant, malgré ces avancées significatives, la version originale de GoFAst [ZMG+21] présente une limitation majeure. Son estimateur de coût ne tient pas suffisamment compte des particularités des données spatiales, ce qui peut entraîner des inefficacités dans la sélection des stratégies d'exécution des requêtes. Cela soulève une question cruciale : Comment pouvons-nous améliorer l'estimation des coûts dans ce contexte spécifique et trouver la meilleure stratégie pour l'exécution de requêtes de données spatiales ?

1.2 Contribution

Dans ce rapport de master, nous explorons en détail l'utilisation du machine learning pour améliorer l'estimation des coûts dans le système RDF_QDAG[KMG+21]. Notre recherche vise à résoudre les défis liés à l'optimisation des requêtes dans les graphes de connaissances en intégrant des données spatiales. Pour ce faire, nous avons réalisé une série d'expérimentations pour évaluer différentes approches de prédiction. Notre objectif est de déterminer la famille d'approches la plus adaptée pour améliorer l'efficacité de l'optimiseur, en tenant compte des spécificités des données spatiales.

1.3 Organisation du manuscrit

Ce rapport est organisé comme suit :

— Le chapitre : Introduction

  Introduit le sujet de ma recherche afin de vous orienter dans le sujet que je vais traiter, il donne un aperçu sur la problématique rencontrée dans le système actuel.

— Le chapitre : Concepts et prérequis

  Explore les connaissances et les concepts de base nécessaires pour aborder le thème de cette recherche.

— Le chapitre : Intégration du Machine Learning

Présente les concepts de l'apprentissage automatique, l'utilité de l'intégration de ce dernier dans les problèmes d'optimisation, ce chapitre présente aussi la solution que nous avons proposée avec les différents modèles utilisés.

— Le chapitre : Le système RDF_QDAG

Dévoile l'approche RDF_QDAG et son objectif, son architecture et son composant qui nous intéresse dans notre étude.

— Le chapitre : Analyse du besoin

Discute le contexte du sujet réalisé en ce qui concerne système RDF_QDAG, l'objectif principal était de comprendre le fonctionnement de ce dernier et d'extraire ses limitations, une solution a été proposée également pour essayer d'améliorer les performances du système.

— Le chapitre : Expérimentations et résultats

Démontre les différents outils et bibliothèques utilisés, la préparation des données et les expérimentions qu'on a fait sur les modèles d'apprentissage choisis, une dernière section dont nous avons présenté les résultats obtenus.

— Le chapitre : Conclusion générale

Conclue en résumant les points essentiels de notre travail en retirant l'importance et l'impact de la méthode utilisée sur ce type de donnée, finissant par les défis qui peuvent se lever en avenir en continuité de cette étude.

Chapitre 2

Concepts et Prérequis

2.1 Introduction

Ce chapitre introduit les concepts fondamentaux nécessaires pour aborder le Web Sémantique, tout en se penchant sur des aspects spécifiques tels que SPARQL Spatial et les structures de données R-Tree.

2.2 Prérequis

2.2.1 World Wide Web

Le World Wide Web (www) est un service hypermédia reparti, accessible par le réseau Internet. Il permet à tout utilisateur de ce réseau d'accéder à travers un logiciel client à un très grand nombre de documents électroniques qui sont gérés par des serveurs repartis sur toute la planète. Les échanges entre serveurs et clients sont fondés sur une représentation commune des documents HTML. [QV95]

2.2.2 Web Sémantique

Le Web Sémantique est un concept visant à permettre aux machines de comprendre la signification de l'information sur le Web. Le but est ainsi de mettre en place, en plus du réseau des hyperliens entre les pages Web classiques, un réseau de liens entre données structurées. Les machines accèdent plus intelligemment aux différentes sources de données contenues sur le Web et peuvent effectuer des traite-

ments plus précis pour les utilisateurs. Le terme a été inventé par Tim Berners-Lee, directeur du W3C, qui supervise l'élaboration des propositions de standards du Web sémantique. [1]

2.2.3 Le Web des données (Linked Data)

Les Linked Data, également connues sous le nom de données liées, sont un ensemble de principes et de bonnes pratiques permettant de rendre les données sur le Web interconnectées et interopérables. L'idée principale des Linked Data est de créer des liens entre les différentes ressources de données en utilisant des identifiants uniques et en les reliant par des relations sémantiques. Les principes des Linked Data ont été formulés par Tim Berners-Lee [Lee01], le créateur du World Wide Web, et sont basés sur les standards du Web sémantique, tels que RDF, URI et SPARQL. Les données liées peuvent être appliquées dans de nombreux domaines pour améliorer le service comme : Sciences de la Vie et de la Santé, Culture et Patrimoine, Éducation, Industrie, Voyages et Tourisme, E-commerce, etc.

Toutes ces données sont stockées dans une base qui s'appelle Wikidata, il s'agit d'une implémentation concrète des principes de Linked Data, les données de Wikidata peuvent être utilisées pour enrichir d'autres sources de données liées en reliant des informations provenant de domaines différents. [2]

1. https://www.imdeo.com/web-3-0-le-web-semantique
2. https://en.wikipedia.org/wiki/Linked_data

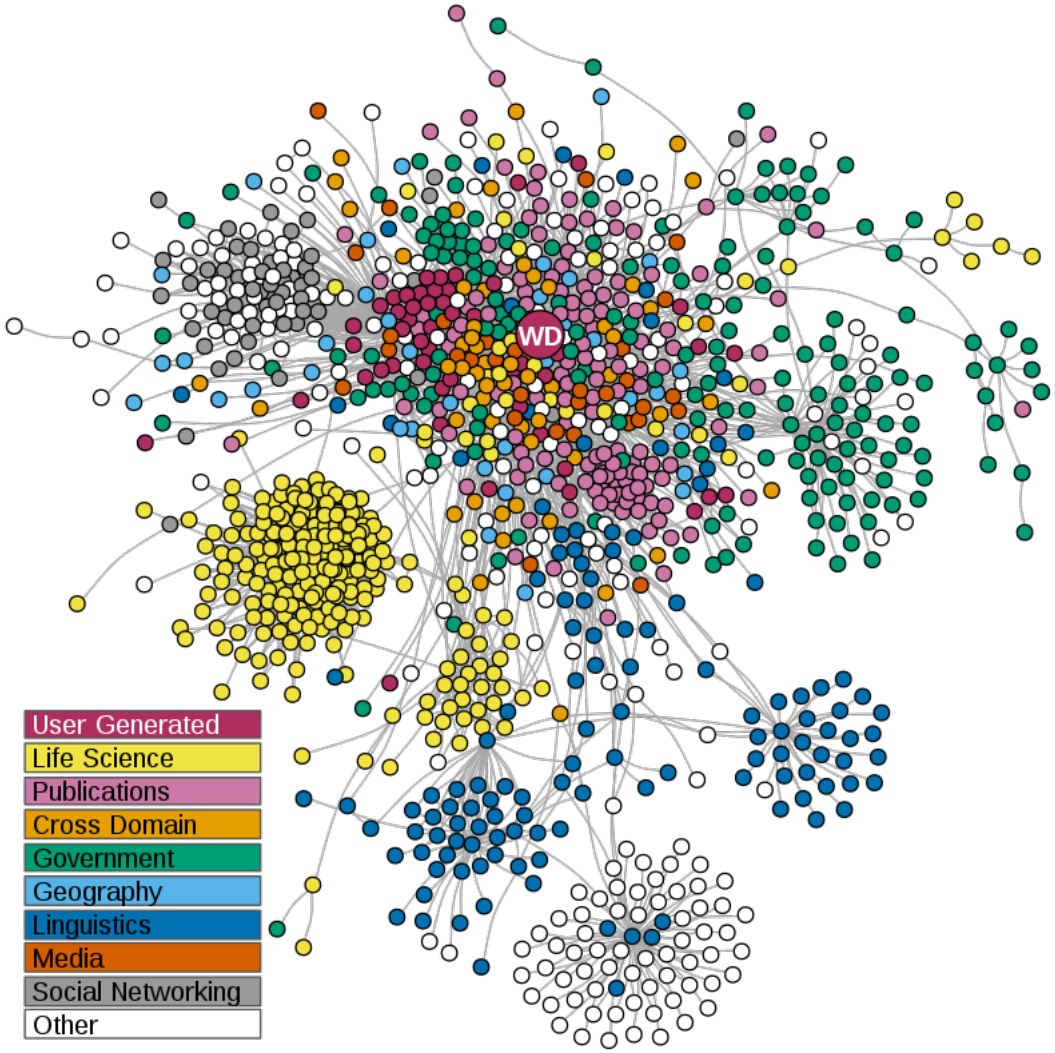


FIGURE 2.1 – la structure du Wikidata

2.2.4 Graphes de connainssances

Aussi connu sous le nom de graphe de connaissances ou graphe sémantique, est une représentation structurée des connaissances qui représente les relations entre différentes entités et concepts. On parle d'un type de graphe orienté où les nœuds représentent des entités et les arêtes représentent les relations entre ces entités.

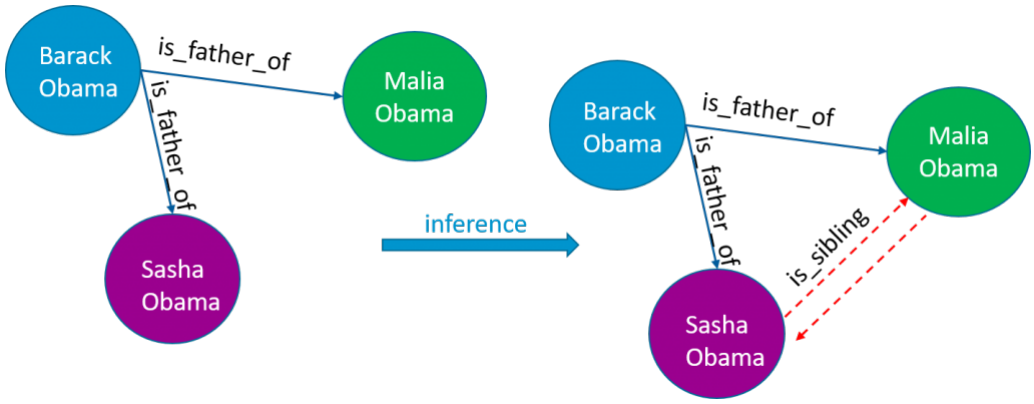


FIGURE 2.2 – Graphe de connaissances

2.2.5 RDF

RDF est un langage de représentation de données qui fournit un cadre pour représenter les données du web sémantique en utilisant des triples sous la forme de (Sujet, Prédicat, Objet). Le langage RDF contient plusieurs composants qui sont : Triples, Ressources, Vocabulaires et ontologies et Requêtes SPARQL.

2.2.6 SPARQL

SPARQL est un langage puissant d'interrogation des données du web sémantique. Il offre une grande flexibilité pour la récupération des informations spécifiques à partir des graphes RDF.

Voici les principales clauses utilisées dans une requête SPARQL :

— Clause SELECT : Cette clause spécifie les variables que vous souhaitez récupérer dans les résultats de la requête.

— Clause WHERE : Cette clause spécifie les motifs de triplets que vous recherchez dans les données RDF.

— Clause OPTIONAL : Cette clause est utilisée pour spécifier des motifs de triplets facultatifs.

— Clause FILTER : Cette clause est utilisée pour appliquer des filtres aux résultats de la requête.

— Clause ORDER BY : Cette clause est utilisée pour trier les résultats de la requête selon une ou plusieurs variables.

— Clause LIMIT : Cette clause permet de limiter le nombre de résultats renvoyés par la requête.

2.2.7 SPARQL Spatial

C'est une extension du langage SPARQL qui permet l'interrogation des données géospatiales, ce langage de requête définit les relations topologiques et les relations spatiales en tant que fonctions spatiales, qui ont de fortes compétences en communication et répondre aux besoins des applications d'information géographique. Pour réaliser une requête SPARQL spatiale, nous devons injecter les notions d'espace ou de géométrie. Plusieurs espaces typiques se développent par des fonctions autodéfinies, comme indiqué dans le tableau suivant [ZHX10] :

SPATIAL EXTENDED SPARQL QUERY

Spatial relations	Function names
Intersection	Udf:SpatialRelIntersects
Minimum bounding rectangle intersection	Udf:SpatialRelEnvelopeIntersects
Tangent	Udf:SpatialRelTouchs
Overlap	Udf:SpatialRelOverlaps
Crossed	Udf:SpatialRelCrosses
Within	Udf:SpatialRelWithin
Contain	Udf:SpatialRelContains
IBE relations	Udf:SpatialRelRelation

FIGURE 2.3 – Requêtes spatiales SPARQL étendues

voici un exemple de requête SPARQL spatiale :

```
SELECT ?p WHERE
{
    ?p <graduatedFrom>?u .
    ?p <worksAt>?w .
    ?u <isLocatedIn>?l .
    ?l <setGeometry>?g .
}
FILTER ?g INTERSECTS "POLYGON((-110 50, -100 50, -100 60, -110 60, -
110 50))";
```

3

3. https://stackoverflow.com/questions/50440311/how-to-run-sparql-spatial-query

2.2.8 R-Trees

Les R-Trees sont des structures de données sous forme d'arbre utilisées comme méthodes d'exploration spatiale pour indexer des informations multidimensionnelles et pour accélérer la recherche, permettant également une minimisation des accès disque. [BS] Dans notre approche, nous aurons besoins de ces structures pour faire l'estimation du nombre d'objets contenant dans un MBR, cette estimation sera faite en fonction d'une requête lancée, sachant que chaque nœud de l'arbre représente un MBR.

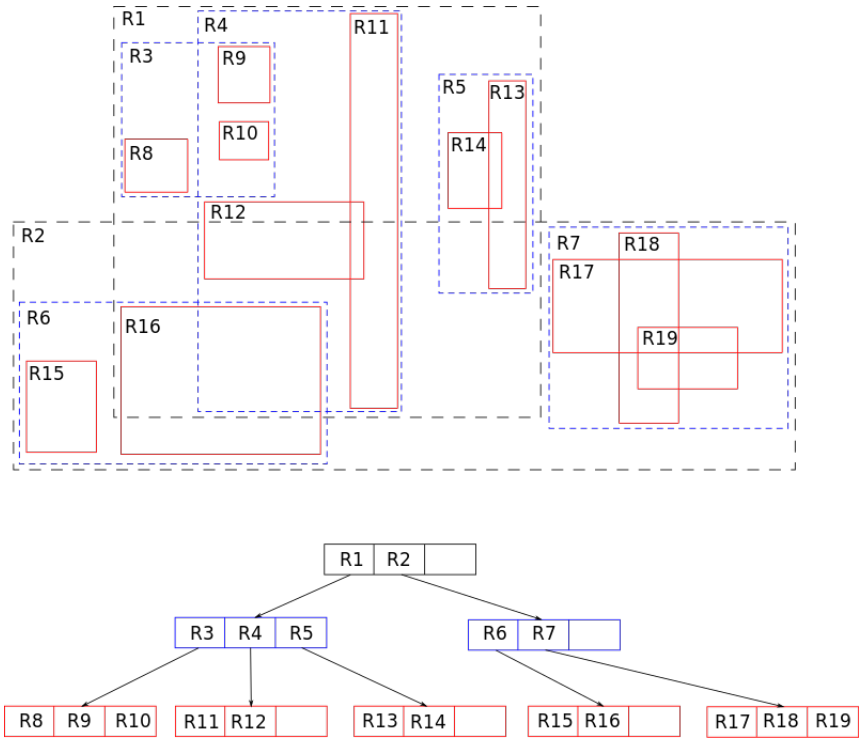


FIGURE 2.4 – R-tree avec représentation graphique

MBR

Un MBR est un rectangle aligné sur les axes x et y qui encadre un objet géométrique donné. Ce rectangle est défini par les coordonnées de son coin inférieur gauche (x_min, y_min) et son coin supérieur droit (x_max, y_max). Il est important de noter que le MBR n'est pas nécessairement le plus petit rectangle qui entoure complètement l'objet géométrique, mais il est suffisamment grand pour contenir l'objet.

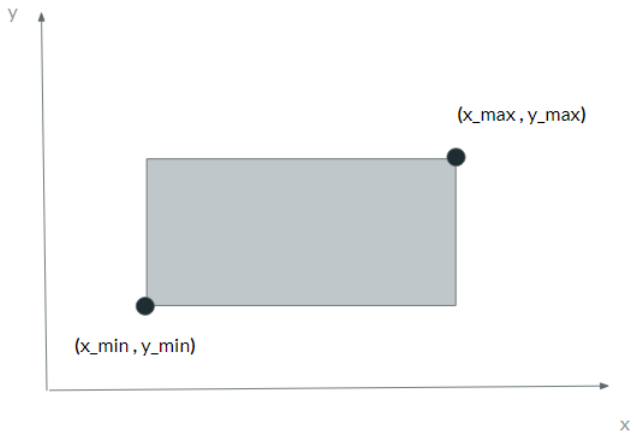


FIGURE 2.5 – Représentation graphique d'un MBR

2.3 Conclusion

En résumé, ce chapitre explore les fondements du World Wide Web, mettant en lumière le Web sémantique et le Web des données. Il présente l'importance des graphes de connaissances, la modélisation avec RDF, les requêtes avec SPARQL, et les requêtes spatiales avec SPARQL Spatial. De plus, il aborde l'utilisation des R-Trees pour l'indexation de données géospatiales. La maîtrise de ces prérequis est essentielle pour naviguer efficacement dans le reste du mémoire.

Chapitre 3

Intégration du Machine Learning (Apprentissage Automatique)

3.1 Introduction

L'explosion de l'Internet, des réseaux sociaux, des appareils connectés et des systèmes numériques a engendré une profusion de données dans de multiples domaines. Dans ce contexte, le machine learning s'est révélé être une approche efficace pour extraire des connaissances pertinentes et prendre des décisions éclairées à partir de ces volumes de données considérables.

Actuellement, plusieurs domaines utilisent le Machine Learning grâce à son efficacité dans diverses tâches, parmi ces domaines, on distingue : traduction automatique, chatbots, reconnaissance faciale, prévisions météorologiques, diagnostic médical, détection de fraudes, création d'agents intelligents dans les jeux vidéos, etc.

3.2 Intelligence Artificielle

Depuis au moins le premier siècle avant notre ère, l'Homme s'est penché sur la création de machines capables d'imiter le raisonnement humain. Le terme « intelligence artificielle » a été créé plus récemment, en 1955, par John McCarthy. En 1956, John McCarthy et ses collaborateurs ont organisé une conférence intitulée « Dartmouth Summer Research Project on Artificial Intelligence » qui a donné naissance

au machine learning, au deep learning, aux analyses prédictives et aux analyses prescriptives. Un nouveau domaine d'étude est également apparu : la science des données. [1]

3.3 Apprentissage automatique

Le Machine Learning, ou apprentissage automatique, est capable de reproduire un comportement grâce à des algorithmes, eux-mêmes alimentés par un grand nombre de données. Confronté à de nombreuses situations, l'algorithme apprend quelle est la décision à adopter et créé un modèle. La machine peut automatiser les tâches en fonction des situations. [2]

Selon les objectifs et les caractéristiques des données, plusieurs types d'apprentissage automatique peuvent être utilisés. Voici les principaux types d'apprentissage automatique :

3.3.1 Apprentissage supervisé (Supervised learning)

Dans ce type, le modèle est entraîné à base d'un ensemble de données étiquetées. Le modèle apprend à associer les caractéristiques des données à leurs étiquettes, ce qui lui permet de prédire les valeurs cibles pour de nouvelles données. Beaucoup d'algorithmes utilisent ce type d'apprentissage parmi : les réseaux de neurones, les machines à vecteurs de support (SVM) et les arbres de décision.

3.3.2 Apprentissage non supervisé (Unsupervised learning)

Contrairement au type précédent, celui-là est entraîné sur un ensemble de données non étiquetées, où aucune information sur les valeurs cibles n'est fournie. Ce type est utilisé dans les tâches de segmentation, classification et réduction de dimension. Les exemples d'algorithmes d'apprentissage non supervisé les plus courants incluent la classification ascendante hiérarchique (CAH), les k-means et les réseaux de neurones auto-encodeurs.

1. https://www.netapp.com/fr/artificial-intelligence/what-is-artificial-intelligence/
2. https://www.oracle.com/fr/artificial-intelligence/machine-learning

Nous nous concentrons sur ce dernier type que nous avons utilisé dans notre étude, plusieurs étapes ont été suivies en appliquant l'apprentissage supervisé, on mentionne les étapes suivantes :

Collecte de données

Cette partie a été réalisé par le M. Amin MESMOUDI et M. Houssem YOUSFI, dont ils ont généré les données massives qu'on a utilisées dans notre étude à partir de la base de connaissances YAGO, ce qui nous a permis de faire nos expérimentations sur de vraies valeurs. [3]

Préparation des données

Avant de commencer l'apprentissage, il faut préparer les données et les prétraiter. Cela peut inclure des étapes telles que l'élimination des valeurs aberrantes, le nettoyage des données, la gestion des valeurs manquantes, la normalisation des caractéristiques, etc.

Séparation des ensembles d'entraînement et de test

Les données sont généralement divisées en ensembles d'entraînement et de test. L'ensemble d'entraînement est utilisé pour entraîner le modèle, tandis que l'ensemble de test est utilisé pour évaluer les performances du modèle sur des données non vues, On prend généralement 30% pour la partie test dans la plupart des cas.

Sélection du modèle

Nous avons utilisé différents modèles et à travers les résultats des tests qu'on va faire, nous allons faire notre sélection.

Entraînement du modèle

Dans cette étape, le modèle est entraîné en ajustant ses paramètres pour minimiser l'erreur entre les prédictions du modèle et les valeurs cibles réelles. L'algorithme d'apprentissage est appliqué sur l'ensemble d'entraînement pour optimiser

3. https://fr.wikipedia.org/wiki/YAGO

les performances du modèle. Nous avons également utilisé la méthode de validation croisée dans l'entrainement des modèles.

Évaluation du modèle

L'évaluation se fait en utilisant l'ensemble de test. Les mesures d'évaluation courantes incluent l'exactitude (accuracy), la précision (precision), le rappel (recall), le score F1 (F1 score), etc. Ces mesures permettent d'évaluer les performances et la capacité de généralisation du modèle.

Métrique Utilisée

Dans notre étude, nous avons utilisé le coefficient de détermination, également appelé R2, plus précisément, le coefficient de détermination R2 est calculé en comparant la variance des prédictions du modèle à la variance des valeurs réelles de la variable dépendante. Voici la formule du R2 Squared :

$$R^2 = 1 - \frac{SSE}{SST}$$

FIGURE 3.1 – Métrique R2 squared

Utilisation du modèle et Prédiction

Une fois que le modèle est entraîné et évalué, il peut être utilisé pour faire des prédictions sur de nouvelles données.

3.3.3 Méthode de validation croisée (Cross Validation)

Le processus de validation croisée implique de diviser l'ensemble de données en plusieurs sous-ensembles, généralement appelés plis (folds). L'une des méthodes les plus couramment utilisées est la validation croisée k-fold, où l'ensemble de données est divisé en k plis de taille égale. Chaque pli est utilisé tour à tour comme ensemble de test, tandis que les k-1 plis restants sont utilisés comme ensemble d'entraînement. Cette procédure est répétée k fois, de sorte que chaque pli est utilisé une fois comme ensemble de test.

Pendant chaque itération de la validation croisée, le modèle est entraîné sur l'ensemble d'entraînement et évalué sur l'ensemble de test. Les performances du modèle, telles que l'exactitude ou l'erreur de prédiction, sont enregistrées pour chaque itération. À la fin du processus, les performances du modèle sont agrégées en utilisant des mesures statistiques, telles que la moyenne ou l'écart type pour obtenir une estimation globale de la performance du modèle. Cette méthode est particulièrement utile lorsque l'ensemble de données est limité et que l'on souhaite tirer le meilleur parti des données disponibles.

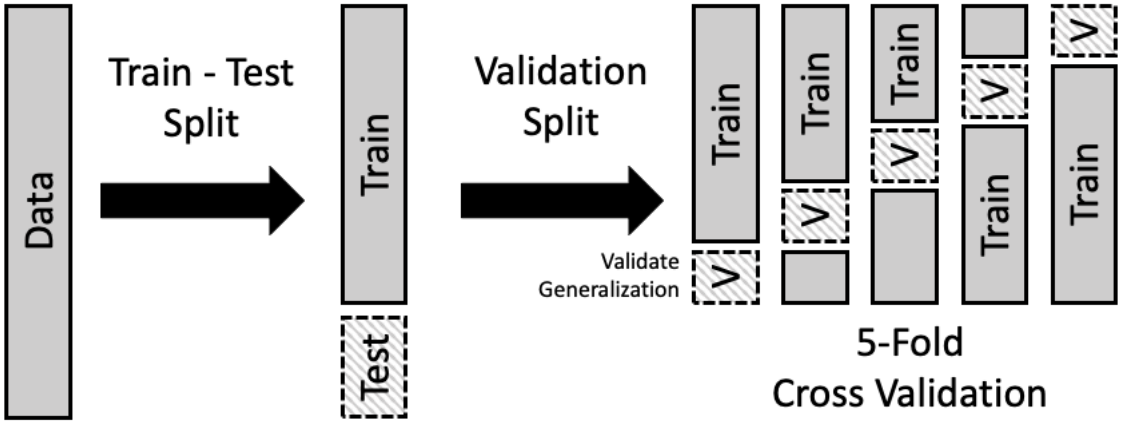


FIGURE 3.2 – Cross Validation Divisions

[noa]

3.3.4 Définition des modèles utilisés

À travers nos données, nous constatons que nous sommes derrière un problème de régression et pas de classification, car nous nous intéressons à faire la prédiction d'une valeur continue en fonction des variables d'entrée et non pas à prédire une classe. Dans cette partie, nous allons définir les modèles utilisés dans notre étude :

Régression linéaire multiple

Il s'agit du modèle de régression le plus simple et le plus couramment utilisé. Il suppose une relation linéaire entre la variable dépendante et les variables indépendantes. La régression linéaire tente de trouver la meilleure ligne droite qui s'ajuste aux données. Un modèle de régression linéaire simple est défini par une équation de la forme : [Guy11]

Les quantités qui viennent du fait que les points ne sont jamais parfaitement alignées sur une droite. On les appelle les erreurs (ou bruits) et elles sont supposées aléatoires.

KNN Régresseur (K-Nearest Neighbors Regressor)

C'est un modèle de régression basé sur l'algorithme des k plus proches voisins. Il est utilisé pour prédire des valeurs numériques continues en se basant sur les données d'entraînement les plus proches dans l'espace des caractéristiques.

L'idée générale du k-NN est de trouver les "k" éléments les plus proches dans l'espace des caractéristiques pour chaque élément de test. Cela se fait par un calcul de distances entre les éléments, généralement la distance euclidienne.

XgBoost Régresseur (Extreme Gradient Boosting)

C'est un algorithme d'apprentissage automatique basé sur le gradient boosting qui est utilisé pour la régression. Il fonctionne en combinant plusieurs modèles de régression simples, appelés "arbres de décision de renforcement", pour former un modèle plus puissant. Il utilise un processus itératif où chaque arbre est ajouté séquentiellement en minimisant l'erreur résiduelle du modèle précédent.

XgBoost calcule le résidu entre la valeur cible réelle et la valeur prédite actuelle. L'arbre suivant sera ensuite formé pour prédire ces résidus afin de réduire progressivement l'erreur dans le modèle global.

— Réglage des hyperparamètres (Tuning) Les modèles d'apprentissage automatique modernes comportent un grand nombre d'hyperparamètres qui doivent être réglés.De même, le XgBoost a au moins cinq hyperparamètres qui doivent être réglés afin d'obtenir un modèle optimal et des résultats plus précis.[SSP19]

Arbre de décision (Decision Tree)

Le modèle d'arbre de décision utilise un arbre construit à partir des données d'entraînement pour prendre des décisions basées sur les caractéristiques des données d'entrée. Chaque nœud de l'arbre représente une caractéristique et chaque branche représente une règle de décision basée sur cette caractéristique. Les nœuds de l'arbre sont construits de manière à minimiser l'erreur de prédiction. Lorsque de

nouvelles données sont présentées au modèle d'arbre de décision, elles sont évaluées en suivant les branches de l'arbre jusqu'à atteindre un nœud de feuille. Le nœud de feuille fournit la prédiction finale.

MLP Regressor (Multilayer Perceptron Regressor)

Basé sur les réseaux de neurones artificiels qui est utilisé pour effectuer des tâches de régression. Il est particulièrement efficace pour capturer des relations non linéaires complexes entre les variables d'entrée et les valeurs cibles. Le MLP Regressor est composé de plusieurs couches de neurones, y compris une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque neurone est connecté aux neurones des couches adjacentes par des poids ajustables. L'apprentissage du modèle consiste à ajuster ces poids pour minimiser l'erreur de prédiction entre les sorties du modèle et les valeurs cibles réelles.

Forêt aléatoire (Random Forest)

Il utilise un ensemble d'arbres de décision pour effectuer des prédictions. C'est une méthode populaire pour les problèmes de classification et de régression, notamment lorsque les données sont complexes et comportent un grand nombre de variables, et c'est le cas dans notre étude, nous utilisons une grande base de données.

Dans cet algorithme, des arbres sont construits partir d'un sous-ensemble aléatoire des données d'entraînement, Une fois que tous les arbres sont construits, les prédictions individuelles de chaque arbre sont combinées pour obtenir une prédiction finale.

SVR (Support Vector Regression)

Contrairement à la régression linéaire traditionnelle, SVR est basé sur le concept des machines à vecteurs de support (SVM) qui est principalement utilisé pour la classification. Cependant, SVR adapte les principes de SVM pour résoudre des problèmes de régression. Il cherche à trouver un hyperplan qui capture au mieux la relation entre les variables d'entrée et les valeurs cibles.

3.4 Conclusion

En conclusion, ce chapitre nous a dotés des outils nécessaires pour comprendre et appliquer l'Apprentissage Automatique dans le contexte de notre travail, en utilisant les connaissances acquises sur la validation croisée, et les modèles pour créer des solutions innovantes et adaptées à notre problématique.

Chapitre 4

Le système RDF_QDAG

4.1 Introduction

L'essor de l'internet a conduit à une explosion de données sur le web, créant un besoin croissant de structurer, d'organiser et de relier ces données de manière intelligente, c'est là qu'intervient le RDF. Cette popularité a fait naitre un grand ensemble de données et a par conséquent conduit à la nécessité pour un traitement efficace de ces données.

Une nouvelle approche qui a été proposée par KHELIL Abdallah et al.(2021) [KMG+21] au sein du laboratoire LIAS, qui a mené à l'adoption de la nouvelle solution RDF_QDAG. Cette dernière combine la fragmentation avec l'exploration des graphes et les techniques d'indexation pour évaluer les requêtes lancées par les utilisateurs. Afin de réduire le coût de cette évaluation, l'équipe a développé un optimiseur Gofast [ZMG+21] qui est basé sur les statistiques. Les travaux ont été dirigés par M. Amin MESMOUDI, Maîtres de conférences à l'Université de Poitiers.

RDF_QDAG a pu surpasser les autres approches en termes de coût et de scalabilité, ce qui l'a rendu plus intéressante.

4.2 Laboratoire de recherches LIAS

Le LIAS (Laboratoire d'Informatique et d'Automatique pour les Systèmes) représente 35 enseignants chercheurs issus des sections CNU 27, 61 et 63 dans les disciplines de l'Automatique, du Génie électrique et de l'Informatique. Il a été créé

depuis le 1er janvier 2012, suite à la fusion des laboratoires du LAII (Laboratoire d'Automatique et d'Informatique Industrielle) et du LISI Laboratoire d'Informatique Scientifique et Industrielle) à Poitiers. [1]

FIGURE 4.1 – Laboratoire Lias/ENSMA

4.3 Aperçu général

Avec l'augmentation des quantités de données utilisées dans le web, ça, c'est devenu impossible d'assurer le passage à l'échelle avec les systèmes traditionnels, Mr Khelil [KMG+21] dans sa thèse a bien défini la nouvelle approche proposée "RDF_QDAG" qui utilise des structures de données scalables pour stocker les données RDF et revisite les techniques d'optimisation traditionnelles comme la fragmentation, l'indexation, la répartition et la gestion de la mémoire pour les adapter à ces structures. La combinaison de la fragmentation physique de données permet de regrouper les nœuds du graphe ayant les mêmes propriétés, ce qui offre des possibilités d'émondage en fonction de la sémantique implicite des données, et le processus d'exploration du graphe de données permet de profiter de la nature graphe des données.

Ce nouveau système supporte plusieurs types de requêtes basées non seulement sur les motifs basiques de graphes, mais aussi qui intègrent des filtres comme le filtre spatial dans notre cas.

L'architecture de QDAG se compose de trois modules principaux : le chargeur, l'optimiseur de requête et le processeur. [KMG+21]

1. https://www.lias-lab.fr/fr/

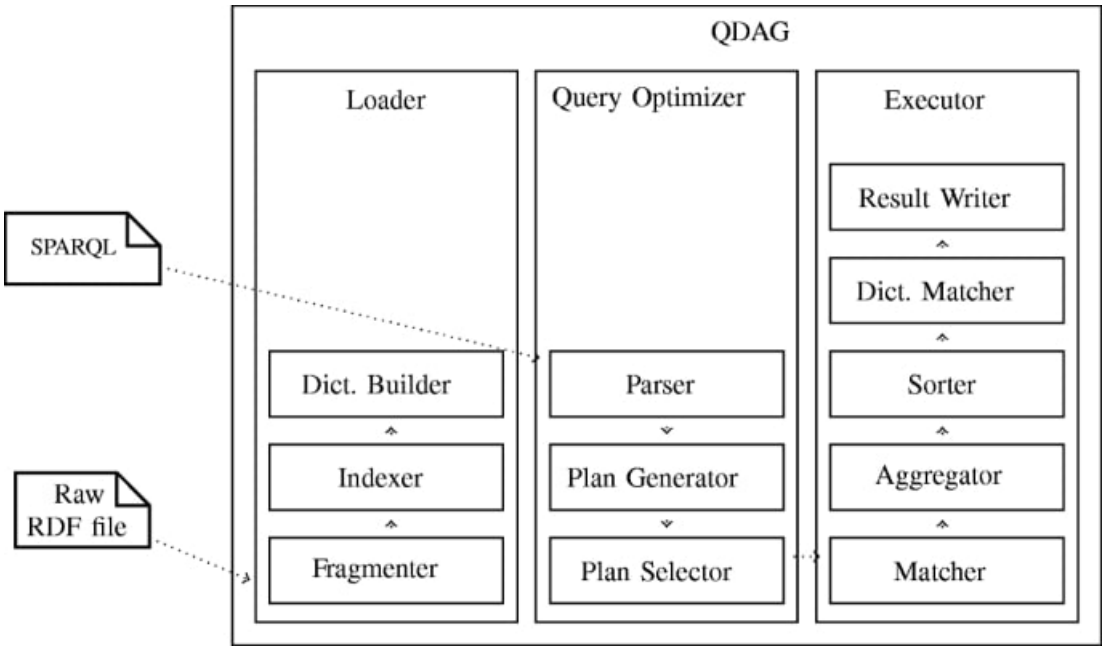


FIGURE 4.2 – Architecture Q_DAG

Dans notre cas nous nous intéressons à la partie optimiseur du système, ce module comprend le plan d'exécution de la requête généré du chargeur, cet optimiseur fait le choix du meilleur plan à l'aide d'un modèle qui calcule le coût,

4.4 l'Optimiseur de requêtes

En effet, l'optimiseur GoFast a été proposé par Zouaghi et al. [ZMG+21] représente un composant qui confère les capacités déclaratives essentielles pour le traitement des requêtes. Ce module traite une représentation logique de la requête en entrée, générant un plan en sortie qui donne l'évaluation de la requête, une même requête n'a pas forcément un seul plan. L'optimiseur doit être capable de répertorier toutes les options envisageables et de comparer leurs coûts afin de choisir le plus efficace ou d'éviter les solutions moins optimales. Ce processus repose sur l'utilisation de statistiques pour calculer le coût de chaque plan envisagé, le coût d'accès disque et le coût d'extensibilité avec les modèles parallèles. Le plan est considéré optimal s'il est moins coûteux en termes de temps d'exécution et en précision.

4.5 RDF_QDAG et données spatiales

L'utilisation fréquente du RDF a conduit à la création de plusieurs types de données (graphiques, temporelles, spatiales), ce qui a mené à une extension du langage SPARQL à fin de représenter ces différentes données. Les données spatiales font référence aux informations géographiques et géospatiales qui sont associées à des emplacements spécifiques sur terre ou dans l'espace. Ces données fournissent des détails sur la position, la forme, la taille, la distance et les relations spatiales entre les objets et les entités géographiques. Comme nous avons déjà parlé dans le chapitre 1 du langage SPARQL spatial, nous tenons maintenant d'expliquer la relation entre ce dernier et RDF_QDAG.

RDF_QDAG [ZMG+21] offre des fonctionnalités avancées pour la représentation des données spatiales, car les entités géographiques sont généralement représentées comme des ressources avec des propriétés spécifiques, ce qui permet de créer un réseau d'informations spatiales interconnectées et de réaliser des requêtes com-

plexes qui exploitent les relations spatiales entre ces entités, ce qui facilite l'interrogation des données spatiales.

4.6 Conclusion

En résumé, l'étude de l'existant nous a offert une vue panoramique de la situation actuelle. Cette étape fondamentale nous permettra de concevoir une solution adaptée et d'optimiser les performances du système actuel.

Chapitre 5

Analyse du besoin

5.1 Introduction

Dans cette section, nous allons passer par plusieurs étapes en commençant par l'évaluation de modèle existant et voir ses limitations en utilisant des métriques appropriées. La deuxième étape consiste à analyser les exigences spécifiques du problème que nous essayons de résoudre et identifiez l'insuffisance du modèle actuel.

5.2 Context

Nous disposons d'une requête SPARQL spatial qui se structure sur deux parties, une "partie graphe" et "partie spatiale" comme on voit sur la figure suivante :

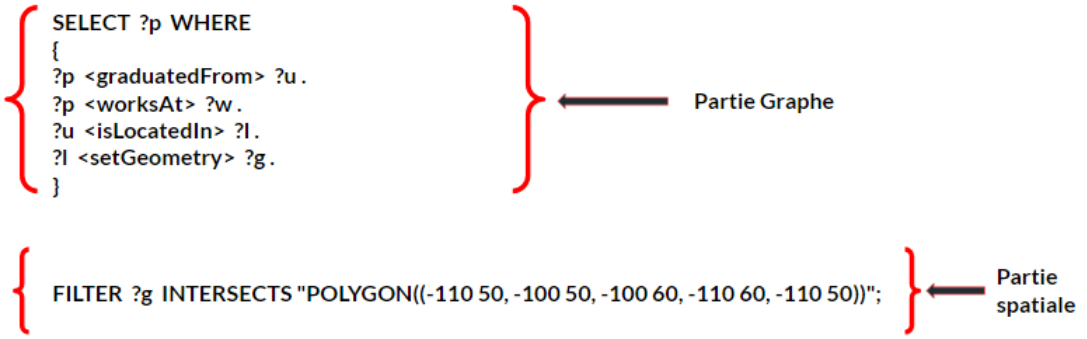


FIGURE 5.1 – Exemple d'une requête SPARQL spatiale

5.2.1 Définition des modèles utilisés

Après le lancement de la requête, on a deux stratégies d'exécution qui sont (en se basant sur l'exemple précédent) :

5.2.2 Plan Spatial First

Ce processus comprend une premiére partie de filtrage dans laquelle il récupère les objets du R-tree et une deuxiéme partie rafinement en cherchant la définition exacte de chaque objet qui satisfassent la requête.

— Trouver les candidats pour ?g

— Utiliser un index R-Tree pour chercher les objets en utilisant les MBRs (Filter)

— Filtrer les objets en utilisant la définition exacte des objets (Refine)

— Trouver les correspondances pour les autres variables

Plan : ?g , ?l , ?u , ?p

Les objets récupérés ont des vraies formes (true shape) qui sont stockés dans un dictionnaire. [YMH+23]

5.2.3 Plan BGP First

— Chercher les candidats pour ?p

— Chercher les candidats pour ?l

— Filtrer en utilisant la définition exacte de ?g

Plan : ?p , ?u , ?l

5.3 Problème rencontré

5.3.1 Limitations du modèle actuel

Après le passage par plusieurs expérimentations sur le modèle actuel, on a rencontré deux problèmes principaux qui sont :

Sur-estimation (overfitting)

Yago-Q7-P3

— Précision : 33%

— Estimation : 270889

— Nombre réel : 56272

Sous-estimation (underfitting)

Yago-Q6-P1

— Précision : 83%

— Estimation : 5901

— Nombre réel : 10639

tel que Q7,Q6 sont des requêtes et P3,P1 sont des plans d'exécution.

On peut constater que ce modèle est limité et on est devant un problème d'instabilité d'où on n'a pas une bonne estimation des objets pertinents.

5.4 Solution Proposée

C'est pour cette raison que nous essayons d'améliorer les performances en utilisant la méthode de la validation croisée avec plusieurs modèles de régression. Nos données sont représentées avec la structure R-tree qu'on a détaillé en premier chapitre 13 tel que chaque nœud de l'arbre contient un MBR et chaque MBR regroupe un nombre d'objets. Notre base d'apprentissage est basée sur les MBR avec le nombre d'objets réels de chacun.

5.5 Structure de la base d'apprentissage

Comme on a déjà mentionné dans la partie précédente, notre base contient quatre variables qui sont "x_min", "y_min", "x_max", 'y_max" d'un MBR et une variable cible qui est "N_objets" c'est le nombre d'objets que contient chaque MBR.

Afin d'enrichir notre base d'apprentissage, nous avons intégré une nouvelle variable appelée "Densité", nous avons été inspirés par l'idée de Theodoridis, Y., Stefanakis, E. et Sellis, T. (2000) [TSS00] dans leur article sur les modèles de coût efficaces pour les requêtes spatiales utilisant R-Tree.

$$D(N,s)=\sum_{N}\prod_{k=1}^{d}s_k=N\cdot\prod_{k=1}^{d}s_k$$

FIGURE 5.2 – Formule de densité

tel que, la densité D : peut être exprimé comme le rapport de la somme des aires de tous les rectangles sur l'aire de l'espace de travail disponible, sachant que nous travaillons sur l'espace suivant : "univ_xmin = -180", "univ_ymin = -90", "univ_xmax = 180", "univ_ymax = 90".

En se basant sur la formule précédente, nous avons ajouté une colonne à notre base d'apprentissage appelé "Densité". Cette variable représente l'aire du MBR divisé par l'aire de l'univers pour chaque MBR.

	x_min	y_min	x_max	y_max	Densité
0	76.0	39.0	123.0	71.0	0.091318
1	-149.0	-27.0	-81.0	13.0	0.165149
2	-54.0	-88.0	-1.0	-51.0	0.119065
3	-102.0	40.0	-54.0	71.0	0.090346
4	-160.0	-13.0	-149.0	-3.0	0.006679
...	...	...	...	...	...
296465	-4.0	-42.0	38.0	19.0	0.155556
296466	70.0	14.0	96.0	70.0	0.088403
296467	-104.0	-73.0	-92.0	-26.0	0.034244
296468	-40.0	-61.0	22.0	-3.0	0.218336
296469	90.0	-47.0	127.0	-28.0	0.042684

296470 rows × 5 columns

[65]:

	n_Objets
0	47260.0
1	11821.0
2	1672.0
3	508786.0
4	84.0
...	...
296465	372177.0
296466	182766.0
296467	145.0
296468	52035.0
296469	5130.0

296470 rows × 1 columns

FIGURE 5.3 – Structure de la base d'apprentissage

En résumé, l'ajout de variables supplémentaires dans un ensemble de données est une étape essentielle pour améliorer les performances d'un modèle d'apprentissage automatique. Cela permet d'enrichir les informations disponibles, de capturer des motifs plus complexes et d'améliorer la capacité du modèle à prendre des décisions précises et informées. Cependant, il est important de sélectionner judicieusement les fonctionnalités supplémentaires et de les évaluer soigneusement pour éviter l'introduction de bruit ou de redondance dans le modèle.

5.6 Conclusion

En résumé, ce chapitre a scruté minutieusement les besoins spécifiques de notre projet, offrant une vision claire et détaillée de l'environnement et des défis à relever. Notre solution, soigneusement élaborée en réponse à ces besoins, représente une avancée significative.

Chapitre 6

Expérimentations et résultats

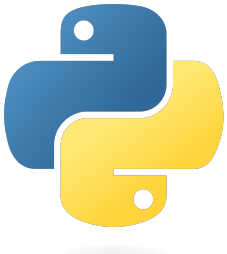
6.1 Introduction

Ce chapitre marque une étape cruciale de notre étude, où nous passons de la théorie à la pratique en mettant en application la solution que nous avons élaborée avec soin. Tout au long de ce chapitre, nous illustrerons les étapes de mise en œuvre et les résultats obtenus.

6.2 Outils utilisés

6.2.1 Python

Python est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative, structurée, fonctionnelle et orienté objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions. [a],il permet de travailler plus rapidement et d'intégrer vos systèmes plus efficacement. [b]



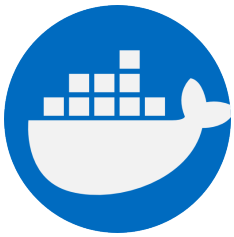
a. https://fr.wikipedia.org/wiki/Python_(langage)
b. https://www.python.org/

6.2.2 Visual Studio

Visual Studio est un outil de Microsoft, qui nous a fournit un environnement de développement riche en fonctionnalités pour développer du code de haute qualité de manière efficace et collaborative. Prise en charge de plusieurs langages : codez en Python, C++, C#, JavaScript, TypeScript, etc. [a]

a. https://visualstudio.microsoft.com/fr/

6.2.3 Docker



Docker est une plate-forme logicielle qui permet de concevoir, tester et déployer des applications rapidement. Docker intègre les logiciels dans des unités normalisées appelées conteneurs, qui rassemblent tous les éléments nécessaires à leur fonctionnement, dont les bibliothèques, les outils système, le code et l'environnement d'exécution. [a]

Nous avons utilisé le docker pour automatiser le déploiement de du script dans n'importe quel environnement. [b]

a. https://aws.amazon.com/fr/docker/
b. https://www.docker.com/

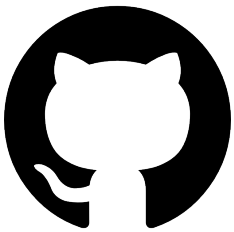
6.2.4 Anaconda

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique [a] , dont nous avons utilisé le module Jupyter dans la partie traitement de données. [b]

a. https://fr.wikipedia.org/wiki/Anaconda_(distribution_Python)
b. https://www.anaconda.com/download

6.2.5 Github

GitHub est une plateforme de développement collaboratif de logiciels qui permet aux développeurs de travailler ensemble, de partager et de collaborer sur des projets. Nous l'avons utilisé afin de partager les projets nécessaires pour notre étude. [a]



a. https://github.com/

6.2.6 GitKraken

GitKraken est un client Git visuel et convivial conçu pour faciliter l'utilisation et la gestion de Git. Il fournit une interface graphique intuitive qui simplifie les opérations courantes de Git, telles que la création de référentiels, la gestion des branches, les confirmations (commits), la fusion (merge), la création de demandes de fusion (pull requests), et bien plus encore.
Nous avons obtenu une licence GitKraken Pro gratuite équivalente à la version payante grâce au programme GitKraken for Students.

6.2.7 Overleaf

Overleaf est une plateforme en ligne qui simplifie la rédaction collaborative de documents scientifiques et techniques, en offrant un éditeur LaTeX en ligne et des fonctionnalités avancées pour faciliter la collaboration, la gestion des références et la publication des documents.
Nous l'avons utilisé pour la rédaction de notre rapport. [a]

a. https://www.overleaf.com/

6.3 Bibliothèques utilisées

6.3.1 NumPy

NumPy est une bibliothèque essentielle pour le calcul scientifique en Python. Elle fournit des structures de données efficaces pour la manipulation des tableaux multidimensionnels, elle permet aussi d'effectuer des opérations numériques à travers des fonctions mathématiques.
Nous l'avons utilisé pour manipuler les différents tableaux de données. [a]

a. https://numpy.org/

6.3.2 Pandas

Pandas est une bibliothèque python qui offre des structures de données et des outils de manipulation de données puissants, Nous l'avons utilisé pour la lecture des fichiers csv, transposé, la construction des DataFrame, etc. [a]

a. https://pandas.pydata.org/

6.3.3 Matplotlib

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et à visualiser des données sous forme de graphiques.
a b

a. https://fr.wikipedia.org/wiki/Matplotlib
b. https://matplotlib.org/

6.3.4 Scikit-Learn

Scikit-learn est l'une des bibliothèques de machine learning les plus populaires en Python. Elle offre une large gamme d'algorithmes d'apprentissage automatique, de prétraitement des données et d'évaluation des modèles, ainsi que des outils pour la sélection des modèles, l'optimisation des hyperparamètres et la validation croisée. Nous l'avons utilisé pour faire nos expérimentations. [a]

a. https://scikit-learn.org/stable/

6.4 Traitement de données

6.4.1 Nettoyage de données

L'élimination des doublons permet de garantir que chaque entrée dans la base est unique, évitant ainsi les répétitions et les redondances.

```
#fusion de deux bases et nettoyage de données

array1 = np.loadtxt('./sdb/yago2-spatial-stat.csv', delimiter=',')
array2 = np.loadtxt('./sdb/yago-spatial-stat.csv', delimiter=',')
array_doublons = np.concatenate((array1, array2))
array = np.unique(array_doublons,axis=0)
```

FIGURE 6.1 – Élimination des doublons

6.4.2 Construction de la base d'apprentissage

Notre objectif est de faire des expérimentations sur plusieurs bases d'apprentissage, chaque base est construite de cinq variables et une variable cible, de base, on a seulement quatre variables et on veut calculer la cinquième à partir des variables qu'on a pour le faire, on va s'inspirer sur la formule dans la figure 5.2 pour calculer la variable densité :

Définition de l'Univers

Comme nous avons déjà expliqué dans le chapitre précédent, nous travaillons sur un espace précis.

```
#Définition de l'univers
Rectangles=array[0:len(array),0:4]
Rectangles.shape
univ_xmin = -180
univ_ymin = -90
univ_xmax = 180
univ_ymax = 90
Surface_univ=(univ_xmax-univ_xmin)*(univ_ymax-univ_ymin)
```

FIGURE 6.2 – Définition de l'univers

Construction du tableau X qui contient les cinq variables

Cette partie contient deux étapes, en premier point on calcule la colonne densité et on divise notre base sur le tableau 'Data_Set' avec les variables, concaténant la colonne densité :

```
#Tableau Data_Set avec les cinq variables
s=0
my_list = []
for i in Rectangles:
    s=abs((i[3]-i[1])*(i[2]-i[0])/Surface_univ)
    my_list.append(s)
D=np.array(my_list)
#colonne  c'est le tableau de densité
L=np.ones(1)
D=D.reshape(D.shape[0],1)
D.shape
Y=array[0:len(array),4:5]
#X_int c'est le tableau avec seulemnt les cordonnées
X_int=array[0:len(array),0:4]
X=np.append(X_int, D, axis=1)
columns = ['x_min','y_min','x_max','y_max','Densité']
Data_Set = pd.DataFrame(data=X,columns=columns)
```

FIGURE 6.3 – Table des variables features "X"

Construction du tableau Target avec la variable cible

Le tableau 'Target' contient seulement la variable cible N_Objets :

```
#Tableau Target avec la variable cible
column=['n_Objets']
Target=pd.DataFrame(data=Y,columns=column)
Target = Target.iloc[:,-1:].values.ravel()
```

FIGURE 6.4 – Table de la variable cible "Target"

6.4.3 Division de la base d'apprentissage

La division d'un jeu de données en ensembles d'entraînement, de validation et de test est une étape importante dans le processus de construction d'un modèle. Nous avons utilisé la méthode "test split" qui divise le jeu de données en un ensemble d'entrainement et un ensemble test, tel que le paramètre "test_size" précise le pourcentage de chaque division, dans notre cas, on a 30% pour le test et 70% pour l'entrainement, le paramètre "randum_state" également permet de fixer une graine aléatoire pour assurer la reproductibilité de la division.

```
#Division de la base 30% pour le test et 70% pour l'entrainement
X_train, X_test, Y_train, Y_test = train_test_split(Data_Set, Target, test_size = 0.3, random_state=42)
```

FIGURE 6.5 – Division de la base en données d'entrainement et données de test

6.4.4 Normalisation des données

En cette étape, nous avons utilisé la bibliothèque "StandardScaler" qui consiste à transformer les variables de manière à ce qu'elles aient une moyenne nulle et un écart-type unitaire, mais cela s'applique seulement sur les X_test et X_train :

```
#Normalisation des données

from sklearn.preprocessing import StandardScaler
#Création d'une instance de StandardScaler qui standarise les variables
scaler = StandardScaler()
#Adapter le scaler aux données d'entraînement
scaler.fit(X_train)
#Mettre à l'échelle les données d'entraînement et de test
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

FIGURE 6.6 – Normalisation de données

6.5 Expérimentations

L'utilisation de plusieurs modèles de machine learning pour mener des expérimentations est une pratique courante dans le domaine de l'apprentissage automatique. Cela permet de comparer les performances, les résultats et les caractéristiques de différents modèles sur un même jeu de données ou pour résoudre un même problème. Pour cela, nous avons choisi différents types de modèles de Machine Learning qui nous sembler pertinents pour notre problème.

6.5.1 Decision Tree

```
#Decision Tree Regressor
print("------------------ Decision Tree------------------")
from sklearn.tree import DecisionTreeRegressor
#instanciation du modèle Arbre de décision
tree_reg = DecisionTreeRegressor(random_state=0)
start1 = process_time()
#Entrainement du modèle
tree_reg.fit(X_train, Y_train)
end1 = process_time()
print("Temps d'entrainement en secondes: " ,end1 - start1)
#Évaluation du modèle avec la validation croisée
reg_result_tree = cross_val_score(tree_reg, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score Decision Tree Regressor : {}".format(np.mean(reg_result_tree)))
print("\n")
#Ajout d'une nouvelle ligne de donnée pour la prédiction
Xnew=[-116,57,-95,79,0.007130]
#Normalisation de Xnew
Xnew = scaler.transform(np.array(Xnew).reshape((1, -1)))
#Affichage de la valeure prédite
print("valeure estimée",round(tree_reg.predict(np.array(Xnew).reshape((1, -1)))[0]))
```

FIGURE 6.7 – Arbre de décision

6.5.2 KNN regressor

```
#KNN Regressor
print("------------------- KNN regressor ------------------")
from sklearn.neighbors import KNeighborsRegressor
#Instanciation du modèle KNN
neigh = KNeighborsRegressor(n_neighbors=2)
start2 = process_time()
#Entrainement du modèle
neigh.fit(X_train, Y_train)
end2 = process_time()
print("Temps d'entrainement en secondes: " ,end2 - start2)
#Évaluation du modèle avec la validation croisée
cv_scores = cross_val_score(neigh, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score KNN Regressor: {}".format(np.mean(cv_scores)))
#Affichage de la valeure prédite
print("valeure estimée",round(neigh.predict(np.array(Xnew).reshape((1, -1)))[0]))
```

FIGURE 6.8 – KNN Regressor

6.5.3 XgBoost Regressor

Partie Tuning

Cela fait référence au processus de recherche des meilleurs hyperparamètres pour un modèle donné, et puisque l'instanciation XgBoost demande l'initialisation de plusieurs hyperparamètres, nous avons utilisé cette technique pour optimiser la performance finale du modèle. Le processus de réglage consiste à expérimenter différentes valeurs d'hyperparamètres pour trouver la combinaison qui donne la meilleure performance du modèle sur les données de validation ou de test.

Apres l'étape du Tuning, on aura les meilleurs hyperparamètres qui peuvent nous mené à un modèle efficace et des résultats qui peuvent être satisfaisantes.

```
#********************Partie tunning XgBoost ************************

#Définition d'une grille d'hyperparamètres
param_grid = {
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 200, 300],
    'gamma': [0, 0.1, 0.2],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0]
}
#Création de l'objet GridSearchCV
grid_search = GridSearchCV(estimator=modelX, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)
#Exécution de la recherche de grille
grid_search.fit(Data_Set, Target)
#Affichage des meilleurs hyperparamètres et du meilleur score
print("Best Hyperparameters: ", grid_search.best_params_)
print("Best MSE: ", -grid_search.best_score_)
#Instanciation du modèle XgBoost avec les hyperparamètres obtenus
modelX = XGBRegressor(grid_search.best_params_)
```

FIGURE 6.9 – Tuning XgBoost

```
#XgBoost

#Instanciation du modèle XgBoost
modelX = XGBRegressor(colsample_bytree=1,max_depth=7,learning_rate=0.1,gamma=0,n_estimators=300,subsample=0.8)
start3 = process_time()
#Entrainement du modèle
modelX.fit(X_train, Y_train)
end3 = process_time()
print("Temps d'entrainement en secondes: " ,end3 - start3)
#Évaluation du modèle avec la validation croisée
cv_scores_x = cross_val_score(modelX, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score Xgboost Regressor: {}".format(np.mean(cv_scores_x)))
#Affichage de la valeure prédite
print("valeure estimée",round(modelX.predict(np.array(Xnew).reshape((1, -1)))[0]))
```

FIGURE 6.10 – XgBoost Regressor

6.5.4 MLP Regressor

```
#MLPRegressor

print("------------------ MLP Regressor ------------------")
#Instanciation du modèle MLP
nn = MLPRegressor(activation='relu',
    hidden_layer_sizes=(10, 100),
    alpha=0.001,
    random_state=20,
    early_stopping=False)
start4 = process_time()
#Entrainement du modèle
nn.fit(X_train, Y_train)
end4 = process_time()
print("Temps d'entrainement en secondes: " ,end4 - start4)
#Évaluation du modèle avec la validation croisée
cv_scores_MLP = cross_val_score(nn, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score MLP Regressor: {}".format(np.mean(cv_scores_MLP)))
#Affichage de la valeure prédite
print("valeure estimée",round(nn.predict(np.array(Xnew).reshape((1, -1)))[0]))
```

FIGURE 6.11 – MLP Regressor

6.5.5 Random Forest

```
#Random Forest regressor

print("------------------ Random Forest ------------------")
from sklearn.ensemble import RandomForestRegressor
#Instanciation du modèle Forêt aléatoire
reg=RandomForestRegressor(max_depth=2, random_state=0)
start5 = process_time()
#Entrainement du modèle
reg.fit(X_train, Y_train)
end5= process_time()
print("Temps d'entrainement en secondes: " ,end5 - start5)
#Évaluation du modèle avec la validation croisée
reg_result = cross_val_score(reg, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score Randum forest : {}".format(np.mean(reg_result)))
#Affichage de la valeure prédite
print("valeure estimée",round(reg.predict(np.array(Xnew).reshape((1, -1)))[0]))
print("\n")
```

FIGURE 6.12 – Random Forest

6.5.6 Linear Regression

```
#regression linéaire

print("------------------ Linear Rergression ------------------")
#Instanciation du modèle Regression linéaire
modell=LinearRegression()
start6 = process_time()
#Entrainement du modèle
modell.fit(X_train, Y_train)
end6 = process_time()
print("Temps d'entrainement en secondes: " ,end6 - start6)
#Évaluation du modèle avec la validation croisée
cv_scores = cross_val_score(modell, Data_Set, Target , cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score Regression lineaire: {}".format(np.mean(cv_scores)))
#Affichage de la valeure prédite
print("valeure estimée",round(modell.predict(np.array(Xnew).reshape(1, -1))[0]))
print("\n")
```

FIGURE 6.13 – Linear regression

6.5.7 SVR

```
#SVR

print("------------------ SVR ------------------")
#Instanciation du modèle SVR
regr_svr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
start7 = process_time()
#Entrainement du modèle
regr_svr.fit(X_train, Y_train)
end7 = process_time()
print("Temps d'entrainement en secondes: " ,end7 - start7)
#Évaluation du modèle avec la validation croisée
cv_scores_svr = cross_val_score(regr_svr, Data_Set, Target, cv=5)
#Affichage de la moyenne des scores de validation croisée
print("Average 5-Fold CV Score SVR : {}".format(np.mean(cv_scores_svr)))
#Affichage de la valeure prédite
print("valeure estimée",round(regr_svr.predict(np.array(Xnew).reshape(1, -1))[0]))
```

FIGURE 6.14 – SVR

6.6 Résultats

Comme on a déjà précisé qu'on va faire des expérimentations sur plusieurs jeux de données avec différent volume, pour cela, nous avons pris cinq bases de données '5000', '50000', '500000', '1 million', '2 millions' de lignes dans les différentes bases d'apprentissage.

6.6.1 5 000 lignes de données

Modèle	**Score R2**	**Temps d'entrainement**
Decision Tree	90.6%	0.015 s
KNN Regressor	93.08%	0 s
XgBoost Regressor	98.2%	5.546 s
MLP Regressor	37.67%	4.453 s
Random Forest	47.99%	0.515 s
Linear Regression	39.55%	0.015 s
SVR	-20%	0.656 s

6.6.2 50 000 lignes de données

Modèle	**Score R2**	**Temps d'entrainement**
Decision Tree	98.37%	0.25 s
KNN Regressor	39.17%	0.078 s
XgBoost Regressor	99.59%	17.671 s
MLP Regressor	62.83%	51.031 s
Random Forest	46.14%	1.843 s
Linear Regression	39.17%	0.015 s
SVR	-19%	106 s

6.6.3 500 000 lignes de données

Modèle	Score R2	Temps d'entrainement
Decision Tree	99.60%	2.265 s
KNN Regressor	99.31%	1 s
XgBoost Regressor	99.75%	203.51 s
MLP Regressor	91.92%	576.85 s
Random Forest	45.05%	14.578 s
Linear Regression	38.47%	0.109 s
SVR	-19%	7236 s

6.6.4 1 million lignes de données

Modèle	Score R2	Temps d'entrainement
Decision Tree	99.6%	2.703 s
KNN Regressor	38.47%	1.203 s
XgBoost Regressor	99.75%	176.93 s
MLP Regressor	91.92%	604.56 s
Random Forest	45.05%	17.531 s
Linear Regression	38.47%	0.140 s
SVR	0%	7981.68 s

6.6.5 2 millions lignes de données environ

Modèle	Score R2	Temps d'entrainement
Decision Tree	59.79%	10.502 s
KNN Regressor	32.52%	2.91 s
XgBoost Regressor	71.49%	930.11 s
MLP Regressor	39.49%	18276 s
Random Forest	25.96%	92.77 s
Linear Regression	16.78%	0.594 s
SVR	%	333128 s

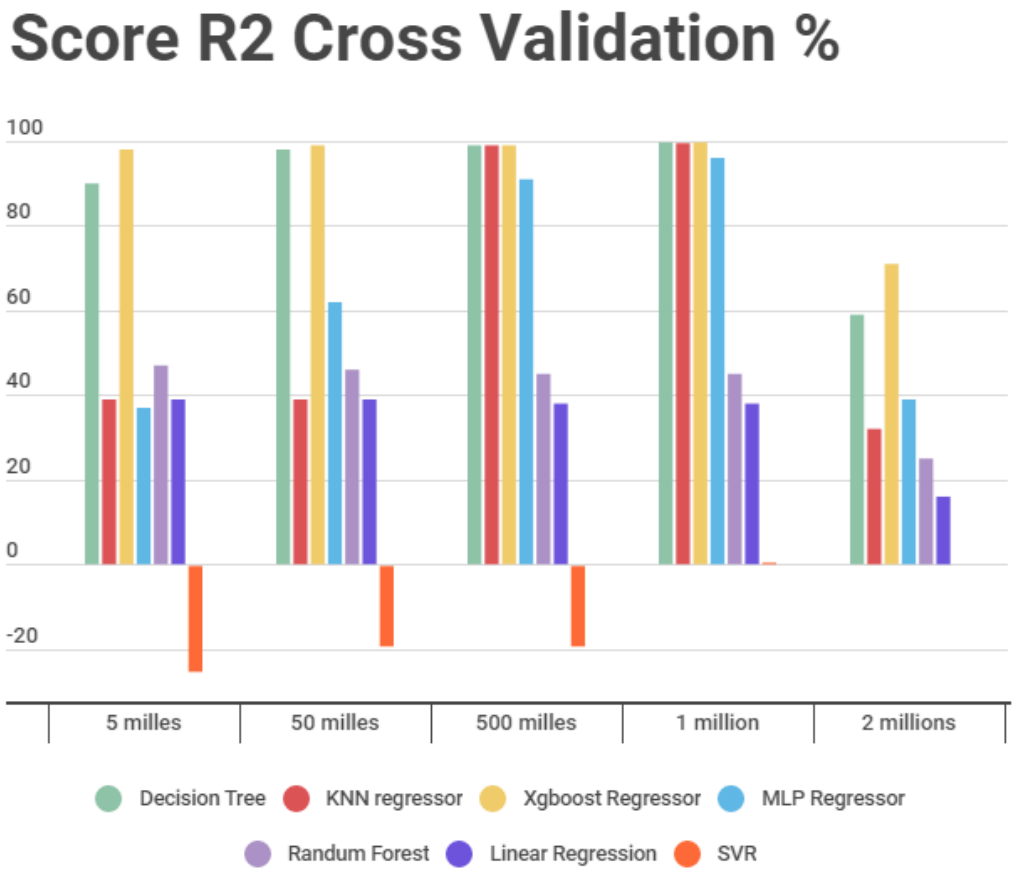


FIGURE 6.15 – Représentation graphique des résultats

Comme on a expliqué déjà l'utilité de calculer le score R2, on peut constater qu'on a quatre modèles (Decision Tree,XgBoost Regressor, MLP Regressor,KNN Regressor) avec des pourcentages elevés, et cela, dans la plupart des partitions de données, on sait au préalable qu'un score R2 proche de 1 signifie que le modèle capture efficacement la variabilité des données ça veut dire que les modèles s'ajustent bien aux données. Un autre point important, c'est qu'un modèle avec un score élevé peut mener à des prédictions fiables sur de nouvelles données, c'est pour cela qu'on a essayé de faire la prédiction de chaque modèle pour confirmer les performances de ces derniers.

6.6.6 Prédiction sur de nouvelles données

Nous avons fait des tests sur une ligne de donnée avec des valeurs réelles qui ne font pas partie de l'ensemble de données entrainées : c'est un MBR avec les coordonnées suivantes : x_min= -116.0, y_min= 57.0, x_max= -95.0, y_max= 79.0, et qui

	Nombre d'objets						
Partitions	Decision T.	KNN	XgBoost	MLP	Random F.	Linear R.	SVR
5000	8594	29640	4350	288646	170600	209379	41189
50000	15198	7726	14721	193871	158504	207489	41307
500000	9194	8341	11267	-13061	148550	206850	42025
1 million	9194	8341	8517	-13061	148550	20685	42435
2 millions	8290	8341	12901	51010	151398	206916	46741

TABLE 6.1 – Table des prédictions

contient 8 790 objets.

6.7 Conclusion

En analysant de près les résultats obtenus, on remarque que les modèles Arbre de décision et KNN regressor ont donnés dans la plupart des partitions des prédictions plus proches des valeurs réelles, ce qui confirme les résultats de la validation croisée pour les deux modèles.

En conclusion, ces modèles-là possèdent de hautes précisions dans leurs prédictions, ce qui indique qu'ils sont bien ajustés aux données d'entraînement, mais il faut savoir que si les données sur lesquelles le modèle est testé différent considérablement de celles sur lesquelles il a été formé, les performances peuvent diminuer, un modèle performant dans un contexte peut ne pas généraliser efficacement vers de nouvelles distributions.

En résumé, un modèle avec de bonnes prédictions est un bon indicateur de fiabilité, mais il est important de considérer la cohérence, la stabilité, l'interopérabilité et d'autres facteurs pour évaluer sa fiabilité globale. La fiabilité découle de la capacité d'un modèle à fonctionner correctement dans une variété de situations et à fournir des résultats cohérents et précis.

Chapitre 7

Conclusion Générale

Ce travail de recherche nous a permis d'explorer en profondeur l'optimisation des requêtes dans les graphes de connaissances basés sur RDF_QDAG, en particulier dans le contexte de l'intégration de données spatiales. Notre objectif principal était d'améliorer l'estimation des coûts et de sélectionner les stratégies d'exécution les plus appropriées pour les requêtes de données spatiales.

Nous avons commencé par mettre en évidence l'importance croissante des graphes de connaissances dans la gestion des données et la recherche d'informations. En intégrant des données spatiales dans ce contexte, nous avons identifié un défi majeur : l'estimation précise des coûts pour les requêtes spatiales. La version originale de GoFAst, bien que robuste pour les requêtes classiques, était insuffisante pour prendre en compte les spécificités des données spatiales, ce qui entraînait des performances sous-optimales.

Pour relever ce défi, nous avons exploré l'utilisation du machine learning pour améliorer l'estimation des coûts. À travers des expérimentations approfondies, nous avons évalué différentes approches de prédiction et avons identifié la famille la plus adaptée à notre contexte. Les résultats ont montré que notre approche basée sur le machine learning permettait une amélioration significative des performances de l'optimiseur, notamment pour les requêtes spatiales complexes. En analysant attentivement les résultats obtenus au cours de notre étude, il est évident que les modèles d'Arbre de Décision et KNN Regressor ont régulièrement généré des prédictions qui se rapprochent le plus des valeurs réelles, confirmant ainsi les performances prometteuses que nous avions observées lors de la validation croisée pour

ces deux modèles.

Cette recherche ouvre la voie à des opportunités passionnantes dans le domaine de l'optimisation des requêtes dans les graphes de connaissances. Elle souligne l'importance croissante des données spatiales dans un monde de plus en plus interconnecté, où la localisation joue un rôle central. Notre travail a montré comment une approche innovante basée sur le machine learning peut être appliquée avec succès pour résoudre des problèmes d'optimisation spécifiques, tout en enrichissant la gestion des connaissances.

Cependant, des défis subsistent, notamment l'exploration de nouvelles techniques de machine learning, l'adaptation à des volumes de données massifs et l'intégration de données spatiales en constante évolution. Nous espérons que cette recherche stimulera davantage d'investigations dans ce domaine et contribuera à une gestion plus efficace des données dans les graphes de connaissances.

En conclusion, notre travail représente une contribution significative à l'amélioration de l'optimisation des requêtes dans les graphes de connaissances et souligne le potentiel prometteur de l'intégration du machine learning dans ce domaine. Nous sommes convaincus que cette recherche ouvre la voie à de nouvelles avancées dans la gestion des connaissances et de l'information, avec des implications potentielles dans de nombreux secteurs d'application.

Table des abréviations

Abréviation	Signification
RDF	Resource Description Framework
QDAG	Querying Data As Graphs
HTML	Hypertext Markup Language
URI	Uniform Resource Identifier
SPARQL	Simple Protocol and Rdf Query Language
MRB	Minimum Bounding Rectangle
KNN	K-Nearest Neighbors
XgBoost	Extreme Gradient Boosting
MLP	Multilayer Perceptron
SVR	Support Vector Regression
SVM	Support Vector Machine

Bibliographie

[BS] K Belmabrouk and Y Slimani. Multimedia mining : P-arbres ou r-arbres pour la représentation des données multimédia.

[Guy11] Arnaud Guyader. Régression linéaire. *Université Rennes*, 2 :60–61, 2011.

[KMG+21] Abdallah Khelil, Amin Mesmoudi, Jorge Galicia, Ladjel Bellatreche, Mohand-Saïd Hacid, and Emmanuel Coquery. Combining graph exploration and fragmentation for scalable rdf query processing. *Information Systems Frontiers*, 23 :165–183, 2021.

[Lee01] Tim Berners Lee. The semantic web. *W3C*, 23, 2001.

[noa] 16.3. cross-validation — learning data science.

[QV95] Vincent Quint and Irene Vatton. L'édition structurée et le world-wide web. *Cahiers GUTenberg*, (19) :85–97, 1995.

[SSP19] Johanna Sommer, Dimitrios Sarigiannis, and Thomas Parnell. Learning to tune xgboost with xgboost. *arXiv preprint arXiv :1909.07218*, 2019.

[TSS00] Yannis Theodoridis, Emmanuel Stefanakis, and Timos Sellis. Efficient cost models for spatial queries using r-trees. *IEEE Transactions on knowledge and data engineering*, 12(1) :19–32, 2000.

[YMH+23] Houssameddine Yousfi, Amin Mesmoudi, Allel Hadjali, Houcine Matallah, and Seif-Eddine Benkabou. Srdf_qdag : An efficient end-to-end rdf data management when graph exploration meets spatial processing. *Computer Science and Information Systems*, (00) :46–46, 2023.

[ZHX10] Xiaofang Zhai, Lei Huang, and Zhifeng Xiao. Geo-spatial query based on extended sparql. In *2010 18th International Conference on Geoinformatics*, pages 1–4. IEEE, 2010.

[ZMG+21] Ishaq Zouaghi, Amin Mesmoudi, Jorge Galicia, Ladjel Bellatreche, and Taoufik Aguili. Gofast : Graph-based optimization for efficient and scalable query evaluation. *Information Systems*, 99 :101738, 2021.

Table des figures

Résumé

Pour améliorer les performances d'exploration des graphes de connaissances, une approche récente a été proposé par l'équipe LIAS intitulé RDF_QDAG. Afin de réduire le coût de cette évaluation, l'équipe a développé un optimiseur Gofast [ZMG+21] qui est basé sur les statistiques. RDF_QDAG a pu surpasser les autres approches en termes de coût et de scalabilité, ce qui l'a rendu plus intéressante. Notre travail est d'essayer d'améliorer l'optimiseur RDF_QDAG en utilisant la technique du Machine Learning et définir un modèle plus efficace pour déterminer le meilleur plan d'exécution d'une requête adapté à nos données.

Mots-clès : Graphe de connaissance, Optimisation, RDF_QDAG, Machine Learning.

Abstract

To enhance knowledge graph exploration performance, a recent approach was proposed the LIAS laboratory, titled RDF_QDAG. In order to reduce the cost of this evaluation, the team developed a statistics-based optimizer called Gofast (Zouaghi et al. 2021). RDF_QDAG has managed to outperform other approaches in terms of cost and scalability, making it more appealing. Our work aims to further improve the RDF_QDAG optimizer by employing Machine Learning techniques and defining a more efficient model to determine the best query execution plan tailored to our data.

Keywords : knowledge graph, Machine Learning, Optimization, RDF_QDAG.

ملخص

قد تم اقتراح النهج لتحسين أداء استكشاف الرسم البياني المعرفي، من قبل فريق LIAS بعنوان RDF_QDAG من أجل تقليل تكلفة هذا التقييم ، المعدات تطوير محسن GoFast الذي يرتكز عليه الإحصائيات. مهمتنا هي المحاولة لتحسين مُحسِنْ RDF_QDAG باستخدام تقنية Machine Learning وتحديد نموذج أكثر كفاءة لتحديد أفضل خطة تنفيذ من استعلام تتكيف مع بياناتنا.

الكلمات المفتاحية : الرسم البياني ، RDF_QDAG ، Machine Learning ، محسن