RAPPORT

Projet Big Data Analytics :

Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèles de Véhicules

Du : 03/02/2024 Au : 30/04/2024

Élaboré par :

JALIL Sohaib

BOUYAZID Ikhlass

MOUSLIH Ikram

KADMIRI Achraf

Encadré par :

MOPOLO Gabriel

PASQUIER Nicolas

SIMONIAN Sergio

Année universitaire : 2023/2024

Résumé

Ce rapport présente le travail réalisé dans le cadre du projet d'analyse de la clientèle d'un concessionnaire automobile pour la recommandation de modèles de véhicules. L'objectif principal était d'exploiter les données disponibles pour améliorer la pertinence des recommandations de véhicules aux clients.

Le projet a impliqué une analyse approfondie des données du concessionnaire, la mise en place d'une architecture de Data Lake, l'utilisation de techniques de Data Mining, Machine Learning et Deep Learning, ainsi que la visualisation des résultats obtenus.

Les principales étapes ont inclus l'analyse exploratoire des données, l'identification des catégories de véhicules, l'application de ces catégories aux données d'immatriculations, et la prédiction des catégories de véhicules pour les clients.

Les résultats ont permis d'optimiser les recommandations de modèles de véhicules pour répondre aux besoins spécifiques des clients.

En conclusion, ce projet a démontré l'efficacité des approches Big Data et Analyse de Données pour améliorer les services d'un concessionnaire automobile.

Abstract

This report presents the work carried out in the project analyzing the clientele of a car dealership for the recommendation of vehicle models.

The main objective was to leverage the available data to enhance the relevance of vehicle recommendations to customers.

The project involved in-depth analysis of the dealership's data, the implementation of a data lake architecture, the use of Data Mining, Machine Learning, and Deep Learning techniques, as well as the visualization of the obtained results.

Key steps included exploratory data analysis, identification of vehicle categories, application of these categories to registration data, and prediction of vehicle categories for customers.

The results optimized the recommendations of vehicle models to meet specific customer needs.

In conclusion, this project demonstrated the effectiveness of Big Data and Data Analysis approaches in enhancing the services of a car dealership.

Contents

Introduction générale

L'industrie automobile est en pleine mutation, portée par les avancées technologiques et l'évolution des préférences des consommateurs. Dans ce contexte, l'analyse des données clients joue un rôle crucial pour comprendre les comportements des consommateurs et adapter les services afin de répondre à leurs besoins individuels. Saviez-vous que 70% des consommateurs préfèrent des recommandations personnalisées lorsqu'ils achètent un véhicule ? C'est dans cette optique que s'inscrit le projet "Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèles de Véhicules", visant à exploiter l'analyse Big Data pour améliorer l'expérience client et optimiser les recommandations de véhicules.

La gestion du Big Data à la visualisation des données et à l'apprentissage automatique ont été réuni pour extraire des informations précieuses des données du concessionnaire. En mettant en place une architecture de Data Lake et en utilisant des techniques analytiques avancées, nous cherchons à améliorer la précision et la pertinence des recommandations de véhicules fournies aux clients.

Dans le cadre de ce projet, nos attentes sont claires : développer des outils et des modèles prédictifs permettant de cibler efficacement les besoins des clients et d'améliorer leur satisfaction tout en stimulant les ventes de la concession. Nos missions principales consistent à contribuer à toutes les phases du projet, de la collecte et du prétraitement des données à la mise en place des modèles prédictifs, en passant par l'analyse exploratoire des données et l'évaluation des performances des modèles.

Le plan de notre rapport de projet est le suivant :

- Présentation du projet.
- Architecture du Data Lake.
- Construction du Data Lake.
- Map/Reduce.
- Visualisation de données.
- Analyse de données.
- Conclusion générale.

Dans les sections suivantes, nous explorerons en détail chaque aspect du projet, mettant en lumière les défis rencontrés, les solutions apportées et les perspectives d'avenir dans le domaine de l'analyse de données pour l'industrie automobile.

1. Présentation du Projet :

Le projet "Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèles de Véhicules" se distingue par son caractère innovant dans l'application de l'analyse des données pour améliorer les recommandations de véhicules. En exploitant les données disponibles, il vise à développer un modèle prédictif précis pour personnaliser les recommandations de véhicules aux clients, ce qui représente une approche novatrice dans le secteur automobile.

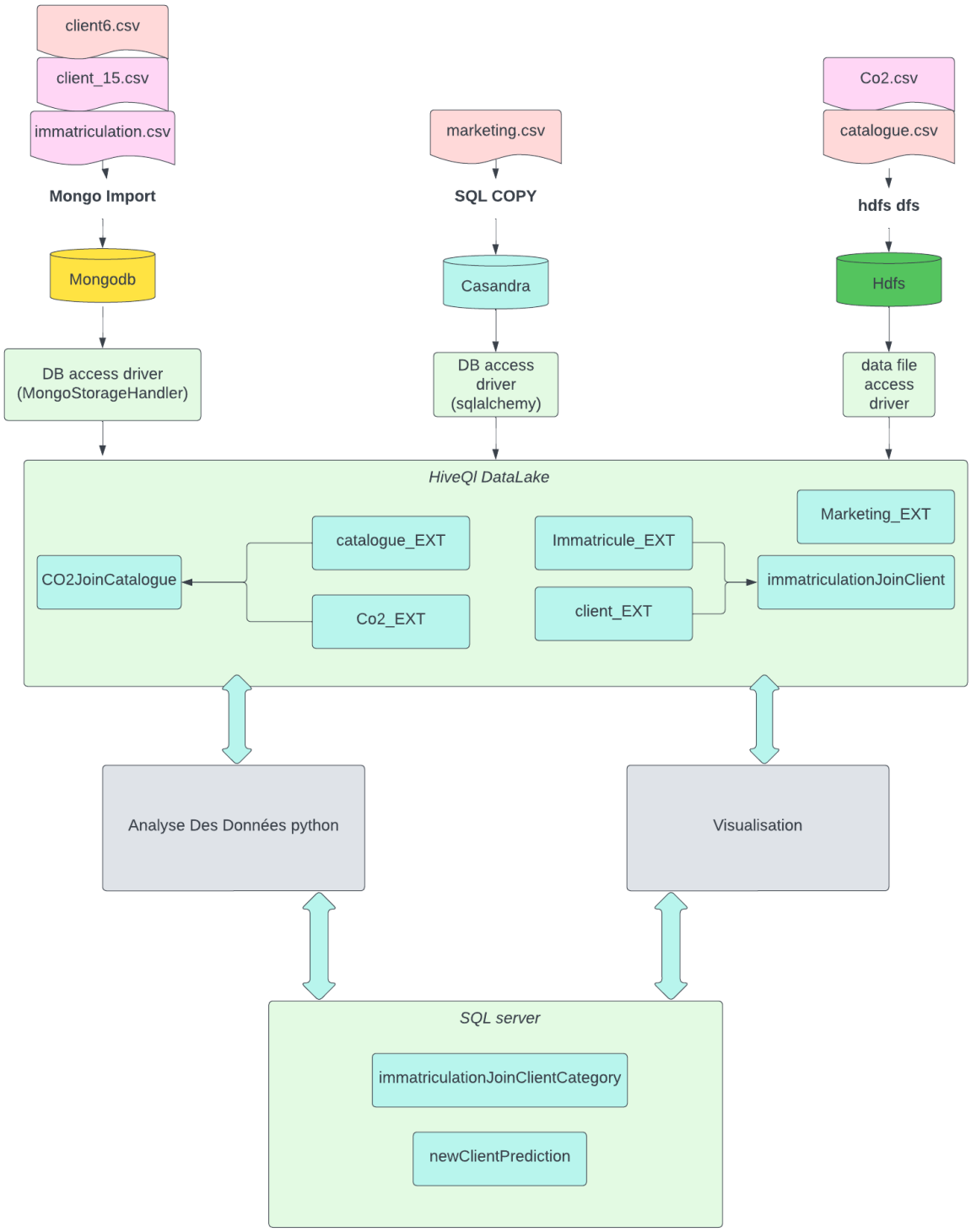
Ce projet intervient dans un contexte métier crucial où la compétition dans le secteur de la vente automobile est de plus en plus intense. La capacité à fournir des recommandations de véhicules précises et personnalisées est devenue un facteur clé de différenciation pour les concessionnaires. Ainsi, le succès de ce projet revêt une importance particulière pour le concessionnaire, car il peut influencer directement sa capacité à fidéliser la clientèle et à augmenter ses ventes.

Les principaux enjeux de ce projet résident dans la qualité des recommandations de véhicules générées par le modèle prédictif. En effet, la précision des recommandations aura un impact direct sur la satisfaction des clients et, par conséquent, sur leur propension à effectuer un achat. De plus, la mise en place d'une architecture de données robuste et évolutive représente également un défi majeur pour assurer le succès à long terme du projet.

Quant aux risques, la qualité des données disponibles constitue un aspect critique. Des données incomplètes, inexactes ou non représentatives pourraient compromettre la performance du modèle prédictif. De plus, la complexité des algorithmes d'apprentissage automatique et de Deep Learning utilisés pour développer le modèle peut entraîner des défis techniques et de compréhension pour l'équipe projet.

En résumé, le projet "Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèles de Véhicules" se positionne comme une initiative innovante et stratégique pour améliorer l'expérience client et stimuler les ventes du concessionnaire. Toutefois, sa réussite dépendra de la capacité de l'équipe projet à surmonter les défis techniques et à garantir la qualité des données et des recommandations générées.

2. Architecture du Data Lake :



Le Data Lake joue un rôle central dans notre projet d'analyse de la clientèle d'un concessionnaire automobile. Il s'agit d'une architecture de stockage de données qui permet d'ingérer, de stocker et de traiter de grands volumes de données provenant de diverses sources, offrant ainsi une plateforme unifiée pour l'analyse et la visualisation.

Dans notre cas, le Data Lake est constitué de plusieurs composants clés :

Sources de données :

Nous avons identifié six fichiers sources contenant des données pertinentes pour notre projet :

- (Client6.csv + Client_15.csv), immatriculation.csv : Stockés dans la base de données MongoDB.
- Marketing.csv : Stocké dans la base de données Cassandra.
- Co2.csv et Catalogue.csv : Stockés dans le système de fichiers distribué HDFS.

Ingestion des données :

Nous avons utilisé des Access drivers pour transférer les données de MongoDB vers le Data Lake HiveQL, tandis qu'un script Python a été employé pour l'ingestion des données provenant de Cassandra.

Tables du Data Lake :

Dans le Data Lake, nous avons créé plusieurs tables externes basées sur les données nettoyées des fichiers sources :

- marketing_ext : Table externe contenant les données de marketing provenant de Cassandra.
- catalogue_ext et co2_ext : Tables externes contenant respectivement les données du catalogue et les données CO2, stockées dans HDFS. Nous avons rejoint ces tables pour former une table appelée Catalogue_X_CO2, qui facilitera l'analyse des données combinées.
- immatriculation_ext et client_ext : Tables externes contenant les données des clients et d'immatriculation, rejointes pour former une autre table appelée client_X_immatriculation.

Analyse et Visualisation :

L'architecture du Data Lake est conçue pour faciliter l'analyse et la visualisation des données. Les tables externes offrent une vue organisée et structurée des données, ce qui facilite leur exploration et leur utilisation dans les outils d'analyse et de visualisation.

3. Construction du Data Lake :

3.1. Nettoyage des données :

Une étape cruciale a été le nettoyage et la transformation des données afin de fusionner les tables CO2 et Catalogue, contrairement aux autres sources de données où les données brutes ont été conservées à leur état initial.

Voici un aperçu des opérations de nettoyage et de transformation effectuées sur chaque fichier :

➢ Fichier Immatriculations.csv (script_for_Immatriculation.ipynb) :

- Remplacement de la valeur "très longue" par "tres longue" dans la colonne "longueur".
- Filtrage des lignes selon le format d'immatriculation spécifié.
- Suppression des doublons basés sur le numéro d'immatriculation, en conservant la première occurrence.
- Ajout d'une colonne 'ID' à partir de 1 pour chaque enregistrement.
- Sauvegarde des données nettoyées dans un nouveau fichier CSV nommé cleaned_Immatriculations.csv".

➢ Fichier Catalogue.csv (script_for_Catalogue.ipynb) :

- Remplacement de la marque "Hyundaï" par "Hyundai".
- Conversion des valeurs de la colonne "marque" en majuscules.
- Remplacement de "très longue" par "tres longue" dans la colonne "longueur".
- Ajout d'une colonne 'ID' contenant un index à partir de 1.
- Réorganisation des colonnes pour placer "ID" en premier.
- Sauvegarde des données nettoyées dans un nouveau fichier CSV nommé "cleaned_Catalogue.csv".

➢ Fichier Marketing.csv (script_for_marketing.ipynb) :

- Remplacement de "é" par "e" dans la colonne "situationFamiliale".
- Ajout d'une colonne 'ID' contenant un index à partir de 1.
- Réorganisation des colonnes pour placer "ID" en premier.
- Conversion de la colonne "2eme voiture" en valeurs booléennes.
- Sauvegarde des données nettoyées dans un nouveau fichier CSV nommé "Cleaned_Marketing.csv".

➢ Fichiers Clients : Concaténation de clients_14.csv et clients_5.csv (script_for_clients.ipynb) :

- Concaténation des ensembles de données clients_14.csv et clients_5.csv pour former un seul ensemble de données.
- Nettoyage des colonnes "sexe", "age", "taux", "situation familiale", "nbEnfantsAcharge" et "2eme voiture" selon les spécifications fournies.
- Filtrage des lignes basé sur le format d'immatriculation spécifié.
- Suppression des doublons basés sur le numéro d'immatriculation, en conservant la première occurrence.
- Ajout d'une colonne 'ID' à partir de 1 pour chaque enregistrement.
- Sauvegarde des données nettoyées dans un nouveau fichier CSV nommé "Cleaned_Clients.csv".

➢ Fichier CO2.csv (script_for_CO2.ipynb) :

- Suppression de la première colonne (en supposant qu'elle est non nommée).
- Extraction de la marque du modèle de voiture à partir de la colonne "Marque / Modele".

- Calcul de la moyenne des émissions de CO2 par marque et remplacement des valeurs de la colonne "Rejets CO2 g/km" par ces moyennes.
- Remplacement des valeurs manquantes dans la colonne "Bonus / Malus" par la moyenne.
- Renommage de la colonne "Cout enerie" en "Cout energie".
- Ajout d'une colonne 'ID' contenant un index à partir de 1.
- Réorganisation des colonnes pour placer "ID" en premier.
- Sauvegarde des données nettoyées dans un nouveau fichier CSV nommé "cleaned_CO2.csv".

Ces opérations de nettoyage et de transformation ont permis d'assurer la qualité et la cohérence des données avant leur utilisation ultérieure dans le processus d'analyse et de modélisation.

3.2. Programme Map/Reduce pour l'adaptation du fichier CO2.csv (script_for_CO2.ipynb):

Le programme Map/Reduce est conçu pour adapter le fichier CO2.csv en intégrant ses informations complémentaires dans la table catalogue du concessionnaire. Voici le déroulement du programme :

Chargement des données :

Les données du fichier CO2.csv sont chargées en tant que DataFrame pandas.

Nettoyage des données :

Les opérations de nettoyage des données du fichier CO2.csv sont effectuées, y compris le formatage des colonnes, le calcul de la moyenne des émissions de CO2 par marque, le remplacement des valeurs manquantes dans la colonne "Bonus / Malus" par la moyenne, et le remplacement des valeurs erronées dans la colonne "Bonus / Malus".

Les données nettoyées sont enregistrées dans un nouveau fichier CSV nommé "cleaned_CO2.csv".

Programme Map/Reduce :

- Map : Dans le contexte de map-reduce, l'étape "map" consiste à diviser les données d'entrée en paires clé-valeur. Ici, nous voulons regrouper les données par la colonne 'Marque / Modele'.
- Reduce : L'étape "reduce" consiste à effectuer une opération d'agrégation sur les données regroupées par les clés. Dans ce cas, nous voulons calculer la moyenne pour chaque groupe.
- Finalize : L'étape "finalize" consiste en toutes les opérations supplémentaires que nous voulons effectuer sur les données réduites. Ici, nous arrondissons également les valeurs moyennes au nombre entier le plus proche.

Export des données :

Les données mises à jour de la table catalogue sont exportées vers un fichier CSV ou une base de données, prêtes à être utilisées dans le processus d'analyse et de modélisation.

3.3. Intégration des données nettoyées dans les Bases de Données correspondantes :

Après avoir nettoyé les données et les avoir préparées pour l'analyse, le processus d'intégration dans les bases de données appropriées a été réalisé avec des scripts. Voici les étapes détaillées pour chaque base de données :

Hadoop Distributed File System (script hdfs.txt) :

- Création du répertoire csv_data dans HDFS pour stocker les fichiers CSV :

✓ *hadoop fs -mkdir csv_data*

Copie des fichiers CSV nettoyés (cleaned_CO2.csv et cleaned_Catalogue.csv) dans le répertoire csv_data:

✓ *hadoop fs -copyFromLocal clean_data/cleaned_CO2.csv csv_data/CO2.csv*

✓ *hadoop fs -copyFromLocal clean_data/cleaned_Catalogue.csv csv_data/Catalogue.csv*

- Vérification des fichiers copiés dans le répertoire csv_data :

✓ *hadoop fs -ls /user/vagrant/csv_data*

Cassandra (script casandra & mongoDB.txt) :

- Démarrage du service Cassandra :

✓ *sudo systemctl start cassandra*

- Connexion à la base de données Cassandra :

✓ *cqlsh localhost*

- Création de l'espace de clés ks et utilisation :
  - *CREATE KEYSPACE IF NOT EXISTS ks WITH replication = {'class': 'NetworkTopologyStrategy'};*
  - *USE ks;*

- Création de la table marketing dans l'espace de clés ks pour stocker les données de marketing :

```
CREATE TABLE IF NOT EXISTS ks.marketing (
    id INT PRIMARY KEY,
    age INT,
    sexe TEXT,
    taux FLOAT,
    situationFamiliale TEXT,
    nbEnfantsAcharge INT,
    deuxiemeVoiture BOOLEAN
);
```

- Import des données nettoyées du fichier CSV Cleaned_Marketing.csv dans la table marketing :
  - *COPY ks.marketing (id, age, sexe, taux, situationFamiliale, nbEnfantsAcharge, deuxiemeVoiture) FROM './clean_data/Cleaned_Marketing.csv' WITH HEADER = true AND DELIMITER=',' AND NULL='NULL' AND ENCODING='latin1';*

- Vérification des données importées dans la table marketing :
  - *SELECT * FROM marketing LIMIT 10;*

- Suppression de la table marketing si nécessaire :
  - *DROP TABLE marketing;*

MongoDB (script casandra & mongoDB.txt) :

- Démarrage du service MongoDB :

✓ *mongo*

- Utilisation de la base de données mbds :
  - *use mbds*

- Création des collections client et immatriculation :
  - *db.createCollection("client");*
  - *db.createCollection("immatriculation");*

- Import des données nettoyées des fichiers CSV (Cleaned_Clients.csv et cleaned_Immatriculations.csv) dans les collections client et immatriculation :
  - *mongoimport --db mbds --collection client --type csv --headerline --file clean_data/Cleaned_Clients.csv*
  - *mongoimport --db mbds --collection immatriculation --type csv --headerline --file clean_data/cleaned_Immatriculations.csv*

- Vérification des données importées dans les collections client et immatriculation :
  - *db.client.find({});*
  - *db.immatriculation.find({});*

- Affichage des collections disponibles :
  - *show collections;*

- Suppression des collections client et immatriculation si nécessaire :
  - *db.client.drop();*
  - *db.immatriculation.drop();*

- Vérification des statistiques des collections client et immatriculation :
  - *db.client.stats();*
  - *db.immatriculation.stats();*

Ces étapes ont permis d'intégrer les données nettoyées dans leurs bases de données respectives, prêtes à être exploitées pour les analyses et les modèles prédictifs.

3.4. Intégration des Données dans le Data Lake :

Le Data Lake joue un rôle crucial dans la centralisation et l'agrégation des données provenant de différentes sources de données. Pour cette tâche, nous avons utilisé des scripts pour extraire les données d'HDFS et des bases de données Cassandra et MongoDB, ainsi que des scripts Hive pour créer des tables externes et réaliser des jointures entre ces données.

Extraction des Données depuis Cassandra vers Hive (dataFromCasandra.py) :

Script Python pour l'extraction des données de Cassandra :

- Le script utilise le module cassandra.cluster pour se connecter à Cassandra et récupérer les données.
- Les données sont ensuite converties en un DataFrame Pandas. vagrant ssh alt.txt
- Enfin, les données sont écrites dans une table Hive en utilisant SQLAlchemy pour la connexion et la méthode to_sql pour l'écriture.

Script Hive pour la création de la table externe marketing :

- Crée une base de données Hive appelée mbds_g5 si elle n'existe pas déjà.
- Crée une table externe marketing avec les mêmes colonnes que la table Cassandra.
- Importe les données depuis HDFS où elles ont été écrites par le script Python.

Création des Tables Externes dans Hive pour les Fichiers CSV dans HDFS (script hive.txt) :

Script Hive pour la création des tables externes CO2 et Catalogue :

- Crée une base de données Hive appelée mbds_g5 si elle n'existe pas déjà.
- Crée deux tables externes, une pour chaque fichier CSV (CO2 et Catalogue).
- Utilise ROW FORMAT DELIMITED pour spécifier le format des données stockées dans les fichiers CSV.
- Définit l'emplacement des fichiers CSV dans HDFS à l'aide de LOCATION.

Création des Tables Externes dans Hive pour MongoDB (script hive.txt):

Script Hive pour la création des tables externes client et immatriculation :

- Crée une base de données Hive appelée mbds.
- Crée deux tables externes, une pour chaque collection MongoDB (client et immatriculation).
- Utilise le format de stockage com.mongodb.hadoop.hive.MongoStorageHandler pour interagir avec MongoDB.

Jointures et Création de Nouvelles Tables (script hive.txt) :

Jointure des Données entre les Tables Catalogue et CO2 :

- Crée une table Catalogue_X_CO2 qui contient des informations provenant des tabs Catalogue et CO2.
- Utilise une jointure interne entre les tables sur la colonne marque.

Jointure des Données entre les Tables Client et Immatriculation :

- Crée une table client_X_immatriculation qui fusionne les données des tables client et immatriculation.
- Utilise une jointure interne entre les tables sur la colonne immatriculation.

Ces étapes d'extraction, de création de tables externes et de jointures dans Hive ont permis d'intégrer les données de différentes sources dans le Data Lake, offrant ainsi un environnement centralisé pour l'analyse et l'exploration des données.

4. Analyse des Données pour la prédiction :

4.1. Clustering et Classification :

Fusion et insertion des données :

Dans cette partie, nous avons créé une nouvelle table appelée immatriculationJoinClient et l'avons peuplée avec les données résultant de la jointure entre la table client et la table immatriculation en utilisant des commandes HiveQL.

- *CREATE TABLE IF NOT EXISTS immatriculationJoinClient (*
  id int, age int, sexe string, taux int, situationFamiliale string, nbEnfantsAcharge int, deuxiemeVoiture string, immatriculation string, nom string, puissance int, longueur string, nbPlaces int, nbPortes int, couleur string, occasion string, prix int
)
- *STORED AS TEXTFILE;*

Cette instruction crée une nouvelle table nommée immatriculationJoinClient et la stocke sous forme de fichier texte.

- *INSERT OVERWRITE TABLE immatriculationJoinClient*
- *SELECT c.id, c.age, c.sexe, c.taux, c.situationFamiliale, c.nbEnfantsAcharge, c.deuxiemeVoiture, c.immatriculation, i.nom, i.puissance, i.longueur, i.nbPlaces, i.nbPortes, i.couleur, i.occasion, i.prix*
  FROM client c
  INNER JOIN immatriculation i ON c.immatriculation = i.immatriculation;

Cette instruction INSERT peuple la table immatriculationJoinClient en sélectionnant les données des tables client et immatriculation et la jointure est effectuée sur la colonne immatriculation des deux tables, puis les colonnes sélectionnées des tables client et immatriculation sont insérées dans la table immatriculationJoinClient.

Clustering (classification_by_category.py) :

Il s'agit d'un script qui effectue une analyse de clustering sur des données provenant d'une base de données Hive. Voici un résumé des étapes :

- Connexion à la base de données Hive en utilisant les informations de connexion fournies (hive_host, hive_port, hive_database).
- Exécution d'une requête SQL pour récupérer les données de la table immatriculationJoinClient.
- Conversion des résultats de la requête en un DataFrame pandas.
- Suppression des lignes contenant des valeurs manquantes dans le DataFrame.
- Sélection des caractéristiques pertinentes pour le clustering (puissance, nbPortes, prix).
- Normalisation des données à l'aide de StandardScaler pour mettre à l'échelle les caractéristiques.
- Test de différentes valeurs de K (nombre de clusters) pour trouver la meilleure, en utilisant la méthode du coude.
- Entraînement du modèle de clustering KMeans avec le nombre optimal de clusters trouvé.
- Ajout des étiquettes de clusters à notre DataFrame (df['cluster']).

Le résultat final est un DataFrame contenant les données d'origine ainsi que les étiquettes de cluster attribuées à chaque ligne.

Classification (classification_by_category.py) :

Il s'agit d'un script qui effectue plusieurs opérations :

- Création d'un dictionnaire de catégories de véhicules associées à chaque cluster : Chaque cluster obtenu à partir de l'analyse de clustering est associé à une catégorie de véhicules spécifique, comme routières, familiales, sport, compactes, luxe, etc.
- Ajout d'une colonne 'Catégorie' dans le DataFrame : Une nouvelle colonne appelée 'Catégorie' est ajoutée au DataFrame df, où chaque ligne se voit attribuer une catégorie basée sur le cluster auquel elle appartient.
- Suppression de la colonne 'cluster' : Une fois la colonne 'Catégorie' ajoutée, la colonne 'cluster' n'est plus nécessaire et est supprimée du DataFrame.
- Exportation du DataFrame vers un fichier CSV : Le DataFrame final est exporté vers un fichier CSV nommé 'immatriculationJoinClientCategory.csv'.
- Établissement d'une connexion à une base de données SQL Server : Les paramètres de connexion au serveur SQL sont définis (adresse IP, base de données, nom d'utilisateur, mot de passe) et une chaîne de connexion est construite.
- Création d'une table dans la base de données SQL Server : Une requête SQL est exécutée pour créer une nouvelle table appelée 'immatriculationJoinClientCategory', avec des colonnes correspondant aux données du DataFrame ainsi qu'à la colonne ajoutée 'Categorie'.
- Insertion des données du DataFrame dans la table SQL Server : Chaque ligne du DataFrame est parcourue, et les données sont insérées dans la table SQL Server nouvellement créée.

- Validation et fermeture de la connexion : Une fois toutes les données insérées avec succès, la transaction est validée et la connexion est fermée.

Enfin, un message de confirmation est affiché pour indiquer que les données ont été insérées avec succès dans la table SQL Server.

4.2. Construction du modèle de prédiction - Classifieur (modelClient_GradientBoostingClassifier.py):

Ce script établit une connexion à une base de données SQL Server, récupère les données des clients à partir d'une table nommée immatriculationJoinClientCategory, prétraite les données, entraîne un classifieur de Gradient Boosting pour prédire les catégories de véhicules en fonction des caractéristiques des clients, évalue la précision du classifieur, puis enregistre le modèle entraîné dans un fichier nommé car_category_prediction_model.pkl. Voici un bref résumé :

- Établissement de la connexion : À l'aide de pyodbc, une connexion à la base de données SQL Server est établie à l'aide de la chaîne de connexion (conn_str).
- Création du curseur : Un objet curseur est créé pour exécuter des requêtes SQL sur la connexion établie.
- Exécution de la requête SQL : Une requête SQL est exécutée pour récupérer les données des clients avec la nouvelle variable "Catégorie" de la table immatriculationJoinClientCategory.
- Chargement des données : Les données des clients récupérées sont chargées dans un DataFrame pandas (client_data).
- Prétraitement des données : Les caractéristiques (X) et la variable cible (y) sont séparées. Les variables catégorielles sont encodées en one-hot.
- Fractionnement des données : Le jeu de données est divisé en ensembles d'entraînement et de test à l'aide de train_test_split.
- Entraînement du modèle : Un classifieur de Gradient Boosting est initialisé et entraîné sur les données d'entraînement.
- Prédiction : Le modèle entraîné est utilisé pour prédire les catégories de véhicules pour l'ensemble de test (X_test).
- Évaluation : La précision du classifieur est calculée à l'aide de accuracy_score qui donne un score de 0.75.
- Sauvegarde du modèle : Le modèle entraîné est sérialisé et enregistré dans un fichier à l'aide de joblib.dump.

Ce processus permet la création et l'utilisation d'un modèle prédictif pour catégoriser les véhicules en fonction des caractéristiques des clients stockées dans une base de données SQL Server.

4.3. Application du modèle de prédiction :

Prédiction à partir de la table marketing (predict_car_category_from_marketing.py) :

Ce script charge un modèle entraîné à partir d'un fichier, établit une connexion à une base de données Hive, récupère des données d'une table Hive, effectue des prédictions sur ces données à l'aide du modèle chargé, puis insère les prédictions ainsi que les données d'origine dans une table SQL Server. Voici une explication succincte :

- Chargement du modèle entraîné : Le modèle préalablement entraîné est chargé à partir d'un fichier.
- Connexion à Hive : Une connexion à une base de données Hive est établie.
- Exécution de la requête SQL pour récupérer les données de la table Hive : Une requête SQL est exécutée pour récupérer les données de la table marketing dans Hive.
- Préparation d'un DataFrame vide pour stocker les prédictions : Un DataFrame vide est créé pour stocker les prédictions.
- Itération sur chaque ligne du DataFrame Hive : Chaque ligne du DataFrame Hive est itérée.
- Préparation des données d'entrée pour la prédiction : Les données d'entrée sont préparées pour la prédiction en les mettant sous la forme attendue par le modèle.
- Prédiction : Les prédictions sont effectuées sur les données préparées à l'aide du modèle chargé.
- Concaténation des prédictions avec le DataFrame Hive : Les prédictions sont concaténées avec le DataFrame Hive original.
- Connexion à la base de données SQL Server : Une connexion à une base de données SQL Server est établie.
- Création d'une nouvelle table dans SQL Server : Une nouvelle table nommée newClientPrediction est créée dans la base de données SQL Server.
- Insertion des données dans la table SQL Server : Les données, y compris les prédictions, sont insérées dans la table newClientPrediction dans SQL Server.

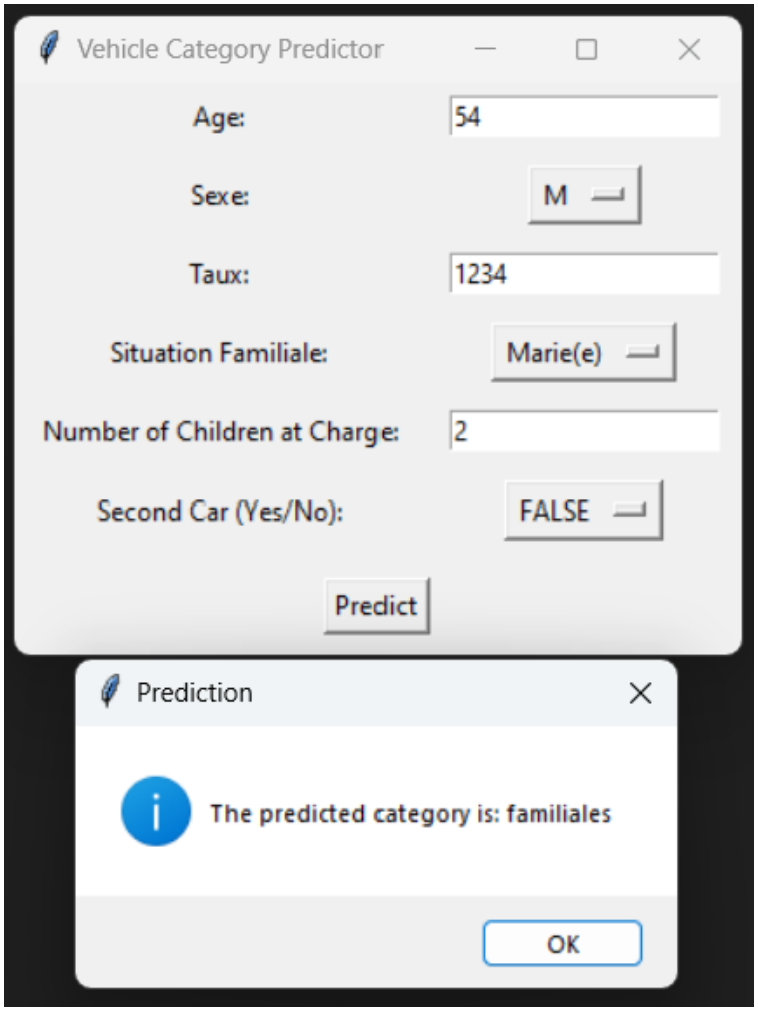
Prédiction à partir des données saisies sur l'interface graphique (predict_car_category.py) :

Ce script charge un modèle préalablement entraîné à partir d'un fichier, puis définit une fonction predict_category() qui utilise ce modèle pour prédire la catégorie d'un véhicule en fonction des valeurs saisies dans une interface graphique utilisateur. Voici une explication succincte :

- Chargement du modèle entraîné : Le modèle entraîné est chargé à partir d'un fichier.
- Définition de la fonction predict_category() : Cette fonction récupère les valeurs saisies dans les champs de texte de l'interface utilisateur, prépare les données d'entrée pour la prédiction, effectue la prédiction à l'aide du modèle chargé, puis affiche le résultat de la prédiction dans une boîte de dialogue.

- Création de l'interface graphique utilisateur : L'interface graphique est créée à l'aide de la bibliothèque Tkinter. Des champs de texte et des menus déroulants sont utilisés pour permettre à l'utilisateur de saisir les données nécessaires à la prédiction.
- Création du bouton de prédiction : Un bouton est ajouté à l'interface utilisateur, qui, lorsqu'il est cliqué, déclenche la fonction predict_category() pour effectuer la prédiction.
- Exécution de l'application : L'application est exécutée en lançant la boucle principale de l'interface graphique utilisateur à l'aide de root.mainloop().

En résumé, ce script fournit une interface utilisateur permettant à l'utilisateur de saisir des données sur un véhicule, puis utilise un modèle préalablement entraîné pour prédire la catégorie de ce véhicule en fonction des données saisies.



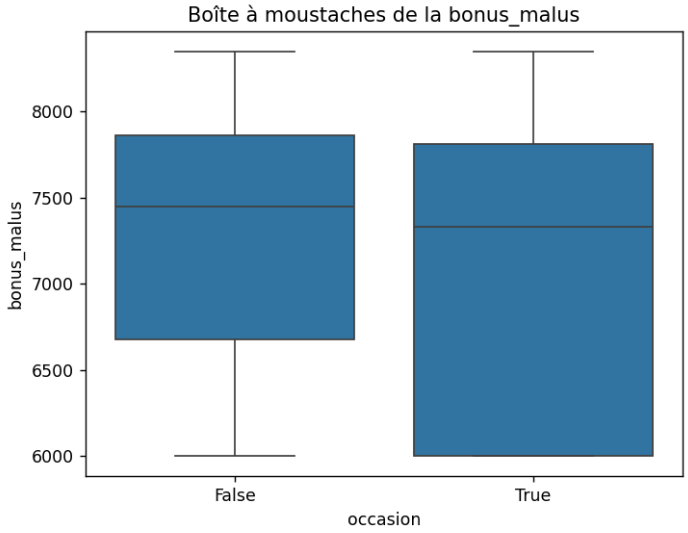
5. Visualisation des Données :

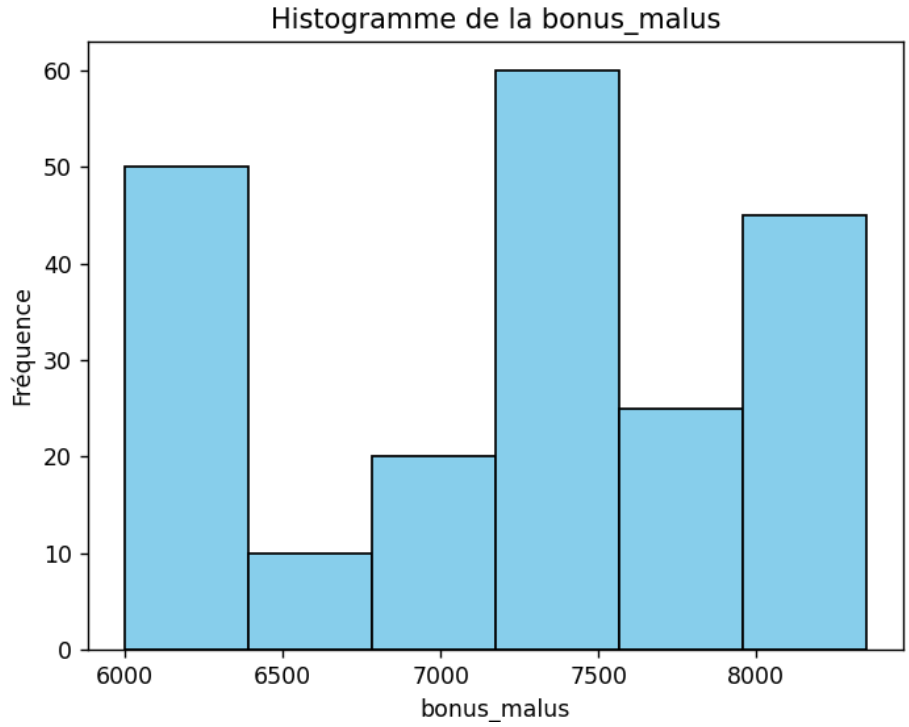
5.1. Visualisation des données à partir de la table Co2joincatalogue :

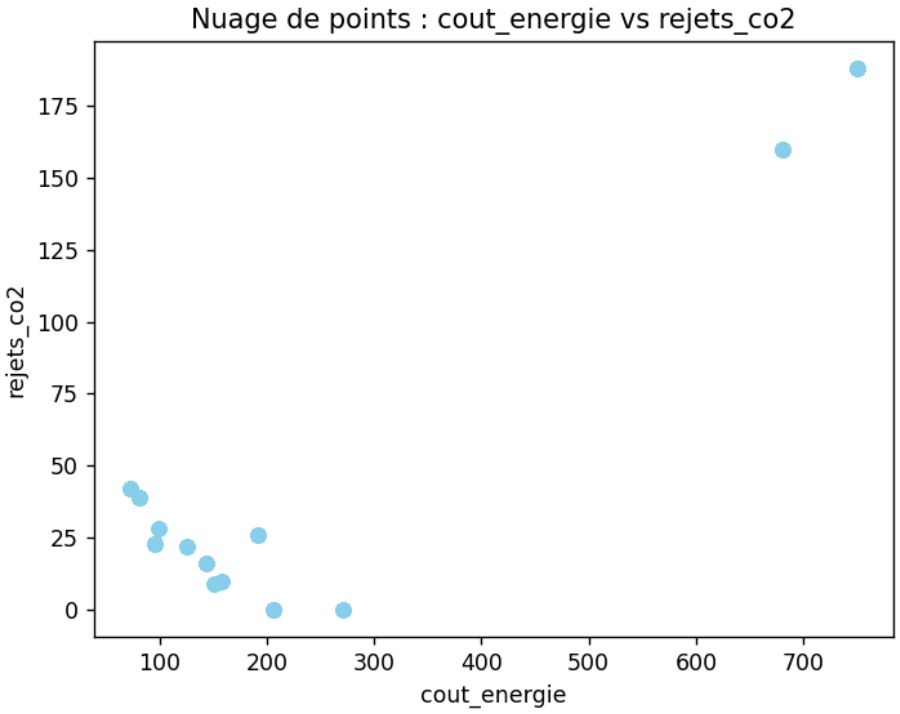
Le script **visualisation_analyse_co2joincatalogue.py** se connecte à une base de données Hive, récupère des données à partir d'une table spécifique, les stocke dans un DataFrame, puis effectue plusieurs types d'analyses visuelles sur ces données à l'aide de bibliothèques telles que Matplotlib et Seaborn. Voici une explication succincte :

- Connexion à la base de données Hive : Le script se connecte à une base de données Hive spécifiée en utilisant les paramètres de connexion tels que l'hôte, le port et la base de données.
- Exécution de la requête SQL pour récupérer les données : Une requête SQL est exécutée pour récupérer toutes les lignes de la table co2joincatalogue. Les résultats sont stockés dans une variable rows.
- Conversion des données en DataFrame : Les données récupérées sont converties en un DataFrame pandas. Les colonnes du DataFrame sont renommées pour plus de lisibilité.
- Analyse visuelle des données :
    - Un histogramme est créé pour visualiser la distribution de la colonne bonus_malus.
    - Un nuage de points est créé pour visualiser la relation entre les colonnes cout_energie et rejets_co2.
    - Une boîte à moustaches est créée pour comparer la distribution de la colonne bonus_malus en fonction de la colonne occasion.
- Fermeture de la connexion et du curseur : Une fois les opérations sur la base de données terminées, la connexion et le curseur sont fermés pour libérer les ressources.

En résumé, ce script récupère des données d'une base de données Hive, effectue diverses analyses visuelles sur ces données, puis ferme la connexion après avoir terminé le traitement.




Histogramme de la bonus_malus
Fréquence
bonus_malus
0
10
20
30
40
50
60
6000
6500
7000
7500
8000

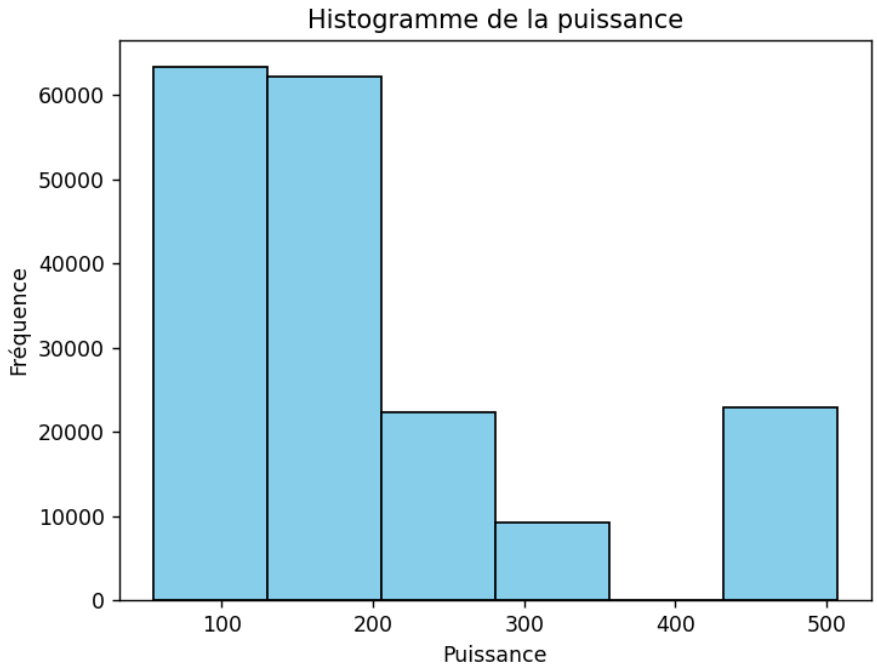

Nuage de points : cout_energie vs rejets_co2
rejets_co2
cout_energie
0
25
50
75
100
125
150
175
100
200
300
400
500
600
700

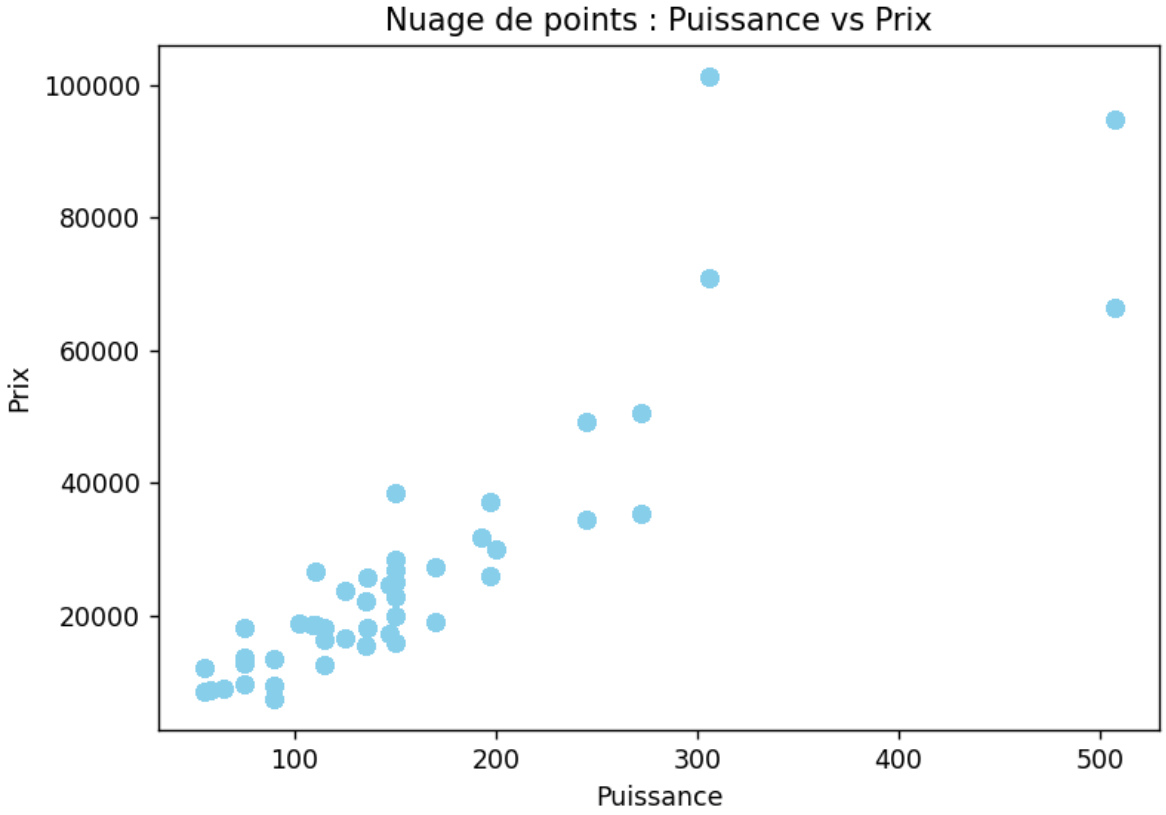
5.2. Visualisation des données à partir de la table immatriculationJoinClient :

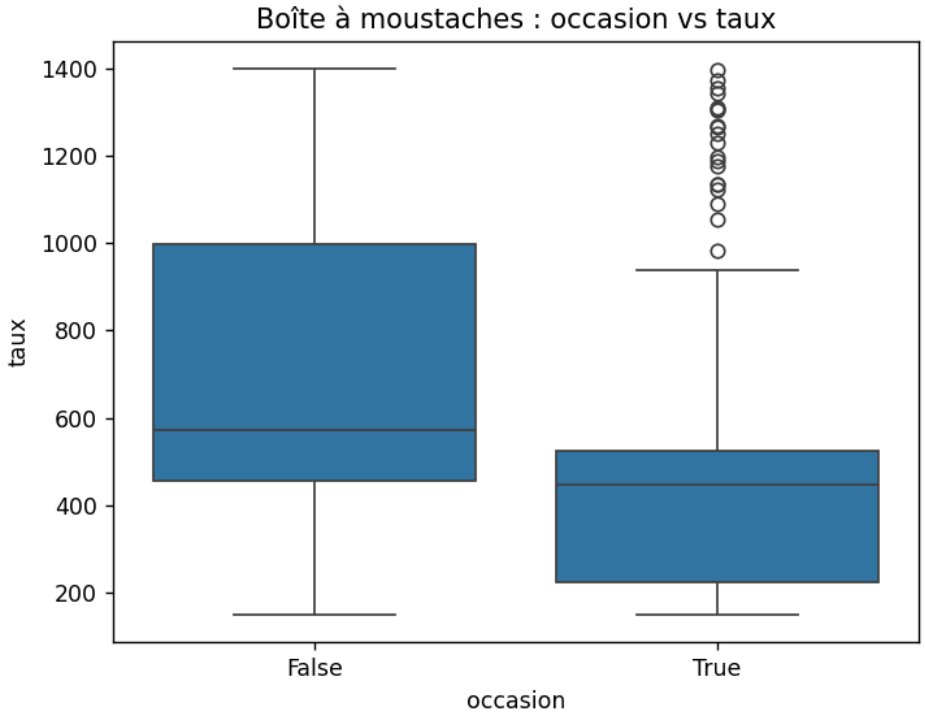
Le script **visualisation_analyse_immatriculationJoinClient.py** se connecte à une base de données Hive spécifiée et récupère des données à partir de la table immatriculationJoinClient. Ensuite, il effectue plusieurs types d'analyses visuelles sur ces données à l'aide de bibliothèques comme Matplotlib et Seaborn. Voici une explication succincte :

- Connexion à la base de données Hive : Le script se connecte à une base de données Hive en utilisant les paramètres de connexion tels que l'hôte, le port et la base de données.
- Exécution de la requête SQL pour récupérer les données : Une requête SQL est exécutée pour récupérer toutes les lignes de la table immatriculationJoinClient. Les résultats sont stockés dans une variable rows.
- Conversion des données en DataFrame : Les données récupérées sont converties en un DataFrame pandas. Les colonnes du DataFrame sont renommées pour plus de lisibilité.
- Analyse visuelle des données :
    - Un histogramme est créé pour visualiser la distribution de la colonne puissance.
    - Un nuage de points est créé pour visualiser la relation entre les colonnes puissance et prix.
    - Une boîte à moustaches est créée pour comparer la distribution de la colonne taux en fonction de la colonne occasion.
- Fermeture de la connexion et du curseur :* Une fois les opérations sur la base de données terminées, la connexion et le curseur sont fermés pour libérer les ressources.

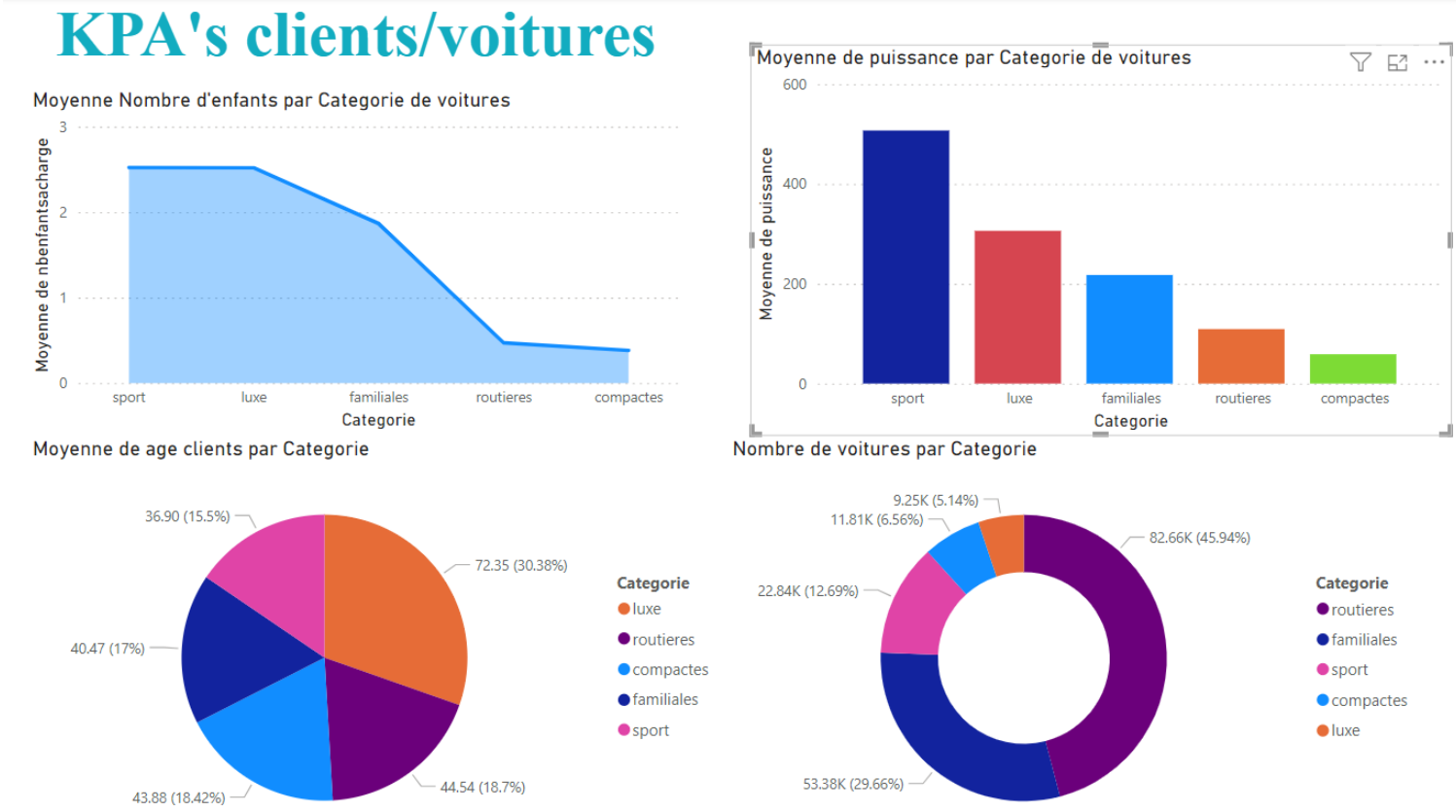
En résumé, ce script récupère des données d'une base de données Hive, effectue diverses analyses visuelles sur ces données, puis ferme la connexion après avoir terminé le traitement.




Nuage de points : Puissance vs Prix
Prix
Puissance
100000
80000
60000
40000
20000
100
200
300
400
500


Boîte à moustaches : occasion vs taux
taux
occasion
1400
1200
1000
800
600
400
200
False
True

5.3. Visualisation des données sur PowerBI de la table immatriculationJoinClientCategory :



Conclusion générale

Ce projet a démontré une méthodologie complète pour le traitement des données, leur intégration dans un environnement de data lake, leur transformation, et enfin leur analyse. Le processus de nettoyage des données a été essentiel pour garantir la qualité des informations. Les scripts Python ont été utilisés pour nettoyer les données dans les fichiers CSV, en supprimant les valeurs incorrectes, en remplissant les valeurs manquantes, et en standardisant les formats.

Les données nettoyées ont été chargées dans différentes bases de données, y compris HDFS, Cassandra et MongoDB. Des scripts spécifiques ont été utilisés pour cette tâche, avec des instructions adaptées à chaque base de données pour assurer une intégration efficace. Les données stockées dans HDFS ont été utilisées pour créer des tables externes dans HIVE, permettant ainsi leur manipulation avec des requêtes SQL. Des tables externes ont également été créées pour les données provenant de MongoDB et Cassandra, facilitant ainsi l'analyse dans un environnement unifié.

Des opérations de jointure ont été effectuées entre différentes tables pour combiner les informations pertinentes. Ces jointures ont permis de créer des ensembles de données plus riches, propices à une analyse plus approfondie. Enfin, des analyses ont été réalisées sur les données combinées pour identifier des tendances, des modèles ou des insights. Des requêtes SQL ont été utilisées pour extraire des informations pertinentes à partir des données stockées dans HIVE.

En conclusion, ce projet illustre l'importance d'une approche systématique pour le traitement des données, depuis leur nettoyage et leur intégration jusqu'à leur analyse. La mise en œuvre de ce processus a permis de créer un environnement robuste pour la gestion et l'analyse de données, ouvrant ainsi la voie à des insights précieux pour l'entreprise.