Projet Big Data Analytics : Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèles de Véhicules

INALIARIJAONA Ony Mirana

RAKOTOARIMANANA Manoa Iharivola

RAKOTOARISON Tojo Fandresena Flavien

RAKOTOARIVONY Tendry Hery ny Aina

Avril 2024 – 09 Juin 2024

Enseignant Encadreur :

Gabriel MOPOLO

Sergio SIMONIAN

Nicolas PASQUIER

Résumé du projet

Le rapport présente une étude sur la concurrence dans l'industrie du commerce de véhicules, visant à évaluer les types de véhicules et à fournir une documentation appropriée aux clients en fonction de leurs besoins.

L'envergure du travail inclut l'analyse de grandes quantités de données et la mise en place d'un DataLake pour le traitement efficace de ces données.

Les méthodes utilisées pour mener cette étude comprennent Access Driver, Alimentation ELT, Hadoop Map Reduce et le langage R pour l'analyse des données.

Les principaux résultats de ce travail sont la manipulation efficace de vastes ensembles de données, l'établissement d'un DataLake et le traitement optimal des données pour répondre aux besoins des clients.

En conclusion, ce projet permettra aux concessionnaires automobiles de mieux cibler les véhicules adaptés à chaque type de client grâce aux diverses méthodes d'analyse de données employées.

Mots clés : Big Data, Analyse de données, Data Lake, ELT, Hadoop Map Reduce, R, Recommandation de véhicules.

Abstract

This report presents a study on competition in the vehicle trade industry, aiming to evaluate vehicle types and provide appropriate documentation to customers based on their needs.

The scope of the work includes analyzing large amounts of data and setting up a Data Lake for efficient data processing.

The methods used in this study include Access Drivers, ELT feeding, Hadoop Map Reduce, and the R language for data analysis.

The main results of this work are the efficient manipulation of large datasets, the establishment of a Data Lake, and the optimal processing of data to meet customer needs.

In conclusion, this project will enable car dealerships to better target vehicles suited to each type of customer through the various data analysis methods employed.

Keywords: Big Data, Data Analysis, Data Lake, ELT, Hadoop Map Reduce, R, Vehicle Recommendation.

Liste des figures

Liste des acronymes

C5.0 - Un algorithme de classification utilisé pour la création de modèles décisionnels

DBI - Database Interface

DBMS - Database Management System

HDFS - Hadoop Distributed File System

Hive - Projet logiciel d'entrepôt de données construit sur Apache Hadoop permettant de faire des requêtes SQL et des analyses de données

KNN - K-Nearest Neighbors, un algorithme de classification

NoSQL - Not Only SQL, une catégorie de systèmes de gestion de bases de données non relationnelles

ODBC - Open Database Connectivity

R - Langage de programmation et environnement logiciel pour l'analyse statistique

RJDBC - R package for connecting to databases via JDBC

SQL - Structured Query Language

Table des matières

1. Introduction générale

" Saviez-vous que l'analyse de données peut augmenter les ventes automobiles de 15 % en ciblant mieux les besoins des clients ? Dans ce projet, nous avons mis en pratique cette réalité fascinante."

L'objectif principal de ce projet est de maîtriser les techniques du Big Data, notamment à travers la mise en place d'un DataLake, l'utilisation de Map Reduce et l'analyse de données avec le langage R, afin de répondre efficacement aux besoins des clients du concessionnaire automobile.

Pour atteindre ces objectifs, nous avons mis en œuvre plusieurs missions clés :

1. **Mise en place d'un DataLake centralisé** pour stocker toutes les données du concessionnaire.
2. **Utilisation de Hadoop Map Reduce** pour effectuer des tâches complexes de traitement et d'analyse de données.
3. **Analyse des données avec le langage** R pour fournir des résultats pertinents aux clients.

Le plan de notre rapport est structuré comme suit :

1. **Répartition des Tâches :** Discussion sur la distribution des responsabilités parmi les membres du groupe.
2. **Architecture du DataLake :** Description de la structure du DataLake avec HIVEQL en tant que frontal.
3. **Construction du DataLake :** Étapes détaillées de la mise en place du DataLake.
4. **Mise en Œuvre de Hadoop Map Reduce :** Explication des tâches et des processus réalisés avec Hadoop Map Reduce.
5. **Analyse des Données avec R :** Utilisation du langage R pour analyser les données et répondre aux besoins des clients.
6. **Conclusion et Réalisations :** Résumé des résultats obtenus et évaluation de la réussite du projet par rapport aux objectifs fixés.

Pour conclure, nous avons réussi à mettre en place un DataLake et utilisé des méthodes telles que Hadoop Map Reduce et le langage R pour traiter et analyser de grandes quantités de données, répondant ainsi efficacement aux besoins des clients en matière de concession automobile. Le rapport se termine par les références, les dossiers des scripts et une vidéo présentant notre projet.

2. Présentation du projet

Le projet se distingue par son approche innovante de l'intégration et de l'analyse de données volumineuses (Big Data) dans l'industrie automobile. En mettant en place un DataLake, nous avons créé une infrastructure capable de centraliser et de gérer efficacement des quantités massives de données, permettant ainsi une meilleure compréhension des tendances et des comportements des consommateurs. L'utilisation de technologies avancées telles que Hadoop Map Reduce et le langage R pour l'analyse de ces données apporte une valeur ajoutée significative en termes de rapidité et de précision des insights obtenus. Cette

combinaison de technologies représente une avancée majeure par rapport aux méthodes traditionnelles de gestion des données.

Dans le secteur de la vente de véhicules, les concessionnaires doivent traiter et analyser une multitude de données provenant de diverses sources : historiques de vente, préférences des clients, inventaires de véhicules, etc. Ces données sont cruciales pour prendre des décisions éclairées, améliorer l'expérience client, et optimiser les opérations commerciales. Cependant, la gestion de ces données pose des défis importants en raison de leur volume et de leur diversité. Le projet vise à résoudre ces défis en fournissant une solution centralisée et scalable pour la gestion et l'analyse des données, permettant aux concessionnaires d'exploiter pleinement le potentiel des Big Data.

La criticité du projet réside dans son potentiel à transformer la manière dont les concessionnaires automobiles gèrent leurs données et prennent des décisions stratégiques. Les principaux enjeux du projet incluent :

- **Optimisation de la Gestion des Données :** La capacité à centraliser et à traiter de grandes quantités de données permet une meilleure gestion des ressources et des stocks, ainsi qu'une personnalisation accrue des offres pour les clients.
- **Amélioration de l'Expérience Client :** En analysant les données des clients de manière plus approfondie, les concessionnaires peuvent mieux comprendre leurs besoins et préférences, ce qui conduit à une satisfaction et une fidélité accrue.
- **Gain de Compétitivité :** L'adoption de technologies Big Data permet aux concessionnaires de rester compétitifs dans un marché en évolution rapide en prenant des décisions basées sur des données précises et actualisées.

Les principaux risques associés à ce projet incluent :
- **Complexité Technique :** La mise en place d'un DataLake et l'intégration de technologies avancées comme Hadoop et R requièrent des compétences techniques spécialisées et une gestion rigoureuse.
- **Sécurité des Données :** La centralisation des données implique des risques accrus en matière de sécurité et de confidentialité des informations, nécessitant des mesures de protection robustes.
- **Adoption par les Utilisateurs :** La réussite du projet dépend également de l'acceptation et de l'adoption par les utilisateurs finaux, qui doivent être formés aux nouvelles technologies et processus.

En gros, ce projet représente une initiative innovante et stratégique pour les concessionnaires automobiles, visant à exploiter les avantages des Big Data pour améliorer la gestion des données, l'expérience client et la compétitivité globale.

3. Répartition du travail en membre du groupe

Nous allons voir dans le tableau ci-dessous la contribution de chaque membre du groupe 5.

	Membre 1	**Membre 2**	**Membre 3**	**Membre 4**
Recherche mise en place du DataLake	100%			
Mise en place du Datalake	15%	55%	15%	15%
Formation sur Map Reduce	100%			
Application du Map Reduce	12%	12%	64%	12%
Formation Data Analytics et Data Science	100%			
Analyse exploratoire des données	55%	20%	5%	20%
Identification des catégories de véhicules	40%	40%	10%	10%
Création d'un modèle de classification supervisée pour la prédiction de la catégorie de véhicules	12%	12%	12%	64%
Rédaction	25%	25%	25%	25%

Notons que :

Membre 1: RAKOTOARIVONY Tendry Hery Ny Aina

Membre 2 : RAKOTOARISON Tojo Fandresena Flavien

Membre 3 : RAKOTOARIMANANA Manoa Iharivola

Membre 4 : INALIARIJAONA Ony Mirana

4. Architecture du data lake

L'architecture de notre Data Lake se compose de plusieurs éléments clés, intégrant diverses sources de données et technologies pour assurer un stockage, un traitement et une analyse efficaces des données. Voici une description détaillée de chaque composant :

a. Fichiers de Données

- CO2.csv et Immatriculations.csv : Ces fichiers CSV contiennent des données sur les émissions de CO2 et les immatriculations de véhicules.
- Catalogue.csv : Ce fichier CSV contient des informations sur le catalogue des véhicules disponibles.
- Clients_4.csv, Clients_15.csv et Marketing.csv : Ces fichiers CSV contiennent des données client et des informations marketing.

b. Source de données

- Utilitaire HDFS : Les fichiers CO2.csv et Immatriculations.csv sont importés dans le système de fichiers distribué Hadoop (HDFS) à l'aide d'un utilitaire HDFS.
- Mongo Import : Le fichier Catalogue.csv est importé dans MongoDB.
- Java Program Extractor : Les fichiers Clients_4.csv, Clients_15.csv et Marketing.csv sont importés dans Oracle NoSQL à l'aide d'un programme d'extraction en Java.

c. Systèmes de Stockage

- HDFS : Stocke les données des fichiers CO2.csv et Immatriculations.csv. Spark MapReduce est utilisé pour traiter les données dans CO2.csv, produisant des sorties stockées de nouveau dans HDFS.
- MongoDB : Stocke les données du fichier Catalogue.csv.
- Oracle NoSQL : Stocke les données des fichiers Clients_4.csv, Clients_15.csv et Marketing.csv.

d. Data Lake HiveQL

- Tables Externes HDFS : Les données de HDFS sont accessibles via des tables externes HiveQL, telles que co2_ext et immatriculation_ext.
- Tables Externes MongoDB : Les données de MongoDB sont accessibles via des tables externes HiveQL, telles que catalogue_CO2.
- Tables Externes Oracle NoSQL : Les données d'Oracle NoSQL sont accessibles via des tables externes HiveQL, telles que clients_ext et marketing_ext.

e. Analyse et Visualisation

- Cloudera ODBC Driver for Apache Hive : Utilisé pour connecter les tables externes du Data Lake HiveQL à des outils d'analyse et de visualisation.
- Langage R : Utilisé pour l'analyse et la visualisation des données. Le langage R se connecte au Data Lake via le driver ODBC Cloudera pour accéder et analyser les données stockées dans les tables externes HiveQL.

f. Base des Résultats

- Les résultats de l'analyse des données sont stockés et visualisés à l'aide du langage R, permettant de répondre aux besoins spécifiques des concessionnaires en termes de gestion des données et d'optimisation des opérations commerciales.

Cette architecture intégrée permet de centraliser les données provenant de diverses sources, de les traiter efficacement avec des technologies avancées comme Spark MapReduce et de les analyser de manière approfondie avec des outils comme le langage R, fournissant ainsi des insights précieux pour les concessionnaires automobiles.

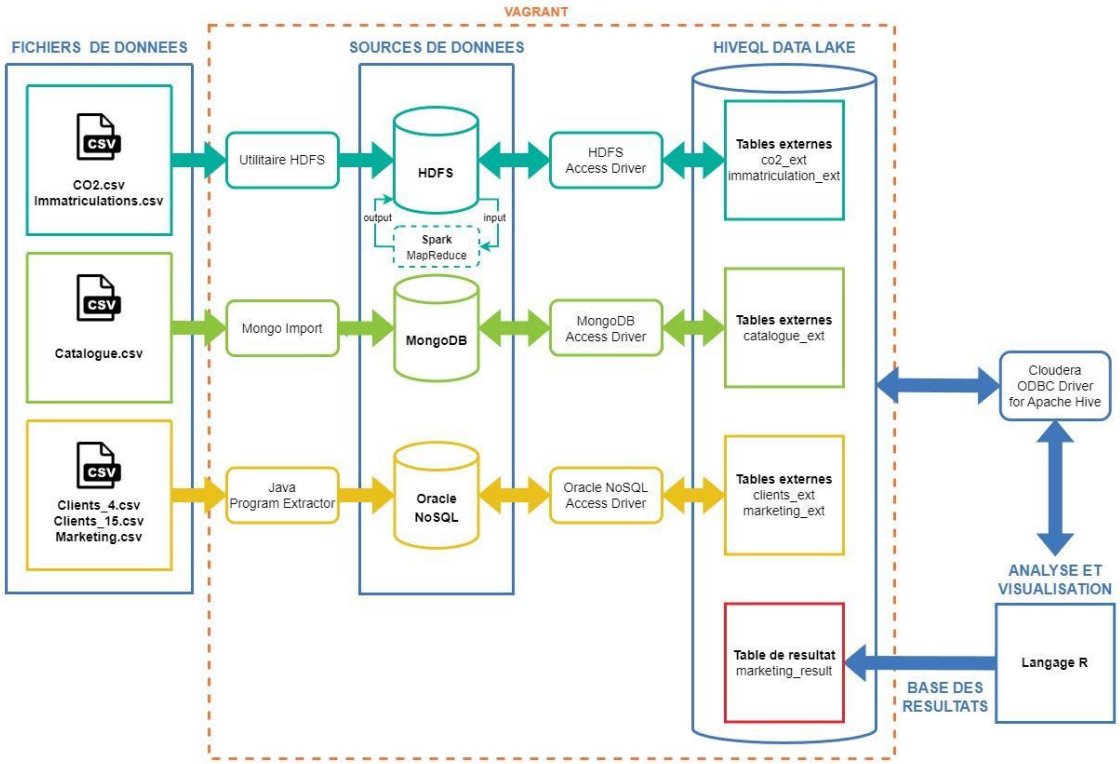

Figure 1: Architecture du data Lake

5. Construction du data lake par étape

Dans cette section, nous détaillons chaque étape de la mise en œuvre de notre architecture de Data Lake, de la collecte des données jusqu'à leur exploitation dans des outils d'analyse et de visualisation tels que R. Pour des instructions plus détaillées, vous pouvez vous référer au fichier README dans notre dépôt Git.

1. Collecte et Importation des Données

A. Source Oracle NoSQL

- Importation des Données
  - Fichiers : `Clients_4.csv`, `Clients_15.csv`, `Marketing.csv`
  - Programmes d'extraction : `Clients.java`, `Marketing.java`
  - Procédure :
    i. Connectez-vous à la machine Vagrant :
    - vagrant ssh

    ii. Définissez les chemins des répertoires :
    - export MYTPHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/programmesExtraction/
    - export DATAHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/dataSources

    iii. Démarrez le serveur Oracle NoSQL :
    - nohup java -Xmx64m -Xms64m -jar $KVHOME/lib/kvstore.jar kvlite -secure-config disable -root $KVROOT &

    iv. Compilez et exécutez les programmes d'extraction pour importer les données :
      - Pour Marketing.java :
    - javac -g -cp "$KVHOME/lib/kvclient.jar:$MYTPHOME:." "$MYTPHOME/Marketing.java"
    - java -cp "$KVHOME/lib/kvclient.jar:$MYTPHOME:." Marketing

      - Pour Clients.java :
    - javac -g -cp "$KVHOME/lib/kvclient.jar:." "$MYTPHOME/Clients.java"
    - java -cp "$KVHOME/lib/kvclient.jar:$MYTPHOME" Clients

B. Source MongoDB

Importation des Données

- Fichier : `Catalogue.csv`
- Procédure :
    i. Démarrez MongoDB
        - sudo systemctl start mongod

    ii. Créez la base de données et la collection :
        - mongo
        - use TPA
        - db.createCollection("Catalogue")
        - quit()

    iii. Importez les données :
        - mongoimport -d TPA -c Catalogue --type=csv --file="$DATAHOME/Catalogue.csv" --headerline

C. Source Hadoop Distributed File System (HDFS)

Importation des Données

- Fichiers : `CO2.csv`, `Immatriculations.csv`
- Procédure :
    i. Définissez le chemin du répertoire HDFS :
        - export HDFSHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/hdfs

    ii. Créez les répertoires sur HDFS et transférez les fichiers
        - Pour CO2
        - hadoop fs -mkdir -p /user/vagrant/input
        - hadoop fs -put $DATAHOME/CO2.csv /user/vagrant/input/CO2.csv

        - Pour Immatriculations
        - hadoop fs -put $DATAHOME/immatriculations.csv /user/vagrant/data/immatriculations.csv

        - Lancez le script Spark pour nettoyer et transformer les données de CO2
        - spark-submit $HDFSHOME/clean_map_reduce_co2.py

2. Création des Tables Externes sur Hive

Pour toutes les sources de données, commencez par démarrer les services Hive et accéder à la console Hive :

```
start-dfs.sh
start-yarn.sh
nohup hive --service metastore > /dev/null &
nohup hiveserver2 > /dev/null &
beeline -u jdbc:hive2://localhost:10000 vagrant
USE DEFAULT;
```

- **Tables Externes pour Oracle NoSQL**
  - Table Marketing :

```
CREATE EXTERNAL TABLE IF NOT EXISTS MARKETING_EXT (
MARKETINGID INTEGER,
AGE INTEGER,
SEXE STRING,
TAUX INTEGER,
SITUATION_FAMILIALE STRING,
NBR_ENFANT INTEGER,
VOITURE_2 STRING
)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
"oracle.kv.kvstore" = "kvstore",
"oracle.kv.hosts" = "localhost:5000",
"oracle.kv.tableName" = "Marketing"
);
```

  - Table Clients :

```
CREATE EXTERNAL TABLE IF NOT EXISTS CLIENTS_EXT (
ClIENTID INTEGER,
AGE INTEGER,
SEXE STRING,
TAUX INTEGER,
SITUATION_FAMILIALE STRING,
NBR_ENFANT INTEGER,
VOITURE_2 STRING,
IMMATRICULATION STRING
)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
```

```
"oracle.kv.kvstore" = "kvstore",
"oracle.kv.hosts" = "localhost:5000",
"oracle.kv.tableName" = "clients"
);
```

Tables Externes pour MongoDB

- Table Catalogue :

```
CREATE EXTERNAL TABLE catalogue_ext (
id STRING,
Marque STRING,
        Nom STRING,
Puissance DOUBLE,
Longueur STRING,
NbPlaces INT,
NbPortes INT,
Couleur STRING,
Occasion STRING,
Prix DOUBLE )
STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
WITH SERDEPROPERTIES('mongo.columns.mapping'='{"id":"_id", "marque":"marque", "nom" :
"nom", "puissance": "puissance", "longueur" : "longueur", "nbPlaces" : "nbPlaces", "nbPortes" :
"nbPortes", "couleur" : "couleur", "occasion" : "occasion", "prix" : "prix"}')
TBLPROPERTIES('mongo.uri'='mongodb://localhost:27017/TPA.Catalogue');
```

Tables Externes pour HDFS

- Table Immatriculation_ext :

```
CREATE EXTERNAL TABLE immatriculation_ext (
Immatriculation STRING,
Marque STRING,
Nom STRING,
Puissance DOUBLE,
Longueur STRING,
NbPlaces INT,
NbPortes INT,
Couleur STRING,
Occasion STRING,
Prix DOUBLE )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/vagrant/data'
TBLPROPERTIES ("skip.header.line.count" = "1");
```

3. Exploitation des Données dans R

Installation du Driver Hive pour R

- Windows :
  1. Téléchargez et installez le Driver ODBC Hive depuis le site de Cloudera.
  2. Configurez le Data Source (DSN) ODBC dans l'administrateur de sources de données ODBC en suivant les instructions pour ajouter le Cloudera ODBC Driver.

- Connexion à Hive depuis R
  1. Installez et chargez le package RJDBC dans R :

```
install.packages(c("RJDBC", "DBI", "rJava", "odbc"))
library(odbc)
library(DBI)
library(rJava)
library(RJDBC)
```

  2. Connectez-vous à Hive en utilisant le driver ODBC :

```
hiveDB <- dbConnect(odbc::odbc(), "Hive Driver")
```

  3. Testez la connexion en exécutant une requête SQL simple :

```
dbGetQuery(hiveDB,"select * from catalogue_ext")
```

Avec ces étapes, nous avons mis en œuvre une architecture de Data Lake, collecté des données de diverses sources, créé des tables externes dans Hive, et configuré l'accès aux données via l'outils d'analyse et de visualisation R.

6. Hadoop Map Reduce

Cette étape détaille la démarche pour l'adaptation et l'intégration du fichier CO2.csv dans la table catalogue d'un concessionnaire automobile. L'objectif principal de cette intégration est d'enrichir la table catalogue avec de nouvelles informations sur les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus pour chaque marque. Cependant, pour obtenir ces données, il est nécessaire de nettoyer et de transformer les informations contenues dans CO2.csv, qui sont initialement réparties par marque et modèle de voiture, et non par marque. De plus, le fichier présente des données manquantes et des problèmes de format.

Nous avons choisi d'utiliser Apache Spark pour le nettoyage et la transformation des données (via MapReduce), ainsi que Hive pour intégrer ces données dans la table catalogue.

a. Chargement des Données

- **Démarrage de HDFS :**

Pour démarrer HDFS, exécuter :

```
start-dfs.sh
```

- **Définition du chemin du répertoire HDFS :**

Pour définir le répertoire où sont stockées les sources de données ainsi que le répertoire de destination des fichiers dans HDFS, utiliser la commande suivante :

```
export DATAHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/dataSources
export HDFSHOME=/vagrant/MBDS_GRP5/DATA_EXTRACTOR/hdfs
```

- **Création d'un répertoire "input" dans HDFS et transfert du fichier CO2.csv :**

Créer un répertoire dans HDFS pour stocker le fichier CO2.csv et le transférer depuis le répertoire local :

```
hadoop fs -mkdir -p /user/vagrant/input
hadoop fs -put $DATAHOME/CO2.csv /user/vagrant/input/CO2.csv
```

b. Nettoyage et transformation des données (Map Reduce) avec Spark

Le script Spark suivant nettoie les caractères spéciaux, corrige les marques mal orthographiées, remplace les valeurs manquantes et prépare les données pour l'intégration en utilisant un traitement de Map Reduce.

Code source du programme Spark : `clean_map_reduce_co2.py` (Chemin dans le projet : `/DATA_EXTRACTOR/hdfs/clean_map_reduce_co2.py`)

Pour adapter le fichier CO2.csv avec la table catalogue, il doit être nettoyé et transformé via Map Reduce pour obtenir les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus par marque.

- **Chargement des données :** Les données sont chargées depuis un fichier CSV spécifié dans un RDD à l'aide de la méthode **textFile** de Spark.

```
from pyspark import SparkContext

sc = SparkContext("local[*]", "NettoyageMapReduceCO2")

# Charger les données depuis le fichier CSV
co2_rdd = sc.textFile('/user/vagrant/input/CO2.csv')
```

- **Nettoyage des caractères spéciaux :** Une fonction est définie pour corriger les caractères spéciaux et les fautes dans le fichier CSV. Chaque ligne nettoyée est ensuite divisée en colonnes en utilisant la virgule (‘,’) comme séparateur.

```
# Fonction pour nettoyer les caractères spéciaux et corriger les marques mal
orthographiées
def nettoyer_caracteres(line):
    # Remplacer les caractères spéciaux et les fautes typographiques
    cleaned = line.replace('Ã\xa0 aimant permanent,', 'à aimant permanent')
    cleaned = cleaned.replace('\xa0', '')  # Enlever les espaces non
conventionnels
    cleaned = cleaned.replace('Ã©', 'é')  # Corriger les caractères
accentués
    cleaned = cleaned.replace('€1', '€')  # Corriger le symbole de l'euro
    cleaned = cleaned.replace('€', '')  # Retirer le symbole de l'euro
    cleaned = cleaned.replace('Ã¨', 'è')  # Corriger les caractères
accentués
    return cleaned

# Nettoyer les lignes du CSV et les diviser en colonnes
lignes_propres_rdd = co2_rdd.map(nettoyer_caracteres).map(lambda line:
line.split(","))
```

- **Correction des marques et suppression des lignes vides :** Une fonction est créée pour corriger les noms de certaines marques et filtrer les lignes vides du jeu de données.

```
# Fonction pour corriger les marques spécifiques et retirer les lignes vides
def corriger_marque(ligne):
    marque = ligne[1].split(' ')[0]  # Utiliser le premier mot de la marque
    if marque == 'LAND':
        marque = 'LAND ROVER'
    elif marque in ['"VOLKSWAGEN', '"KIA']:
        marque = marque.replace('"', '')  # Enlever les guillemets
    return [ligne[0], marque, ligne[2], ligne[3], ligne[4]]

# Corriger les marques
marques_corrigees_rdd =
lignes_propres_rdd.map(corriger_marque).filter(lambda x: x[0] != '')
```

- **Conversion des valeurs en entiers :** Une fonction est définie pour convertir les valeurs numériques en entiers, en remplaçant les valeurs manquantes par 0.

```
# Fonction pour convertir une chaîne en entier ou renvoyer 0 s'il y a une
valeur manquante
def convertir_en_entier(value):
    return int(value) if value != '-' else 0

# Convertir les valeurs appropriées en entiers
entiers_rdd = marques_corrigees_rdd.map(lambda x: (x[1],
convertir_en_entier(x[2]), convertir_en_entier(x[3]), int(x[4])))
```

- **Regroupement par marque et calcul des moyennes finales (Map Reduce) :** Les données sont regroupées par marque (Map Reduce) en sommant les colonnes pertinentes (malusBonus, rejetsCO2, coutEnergie) et en comptant le nombre d'occurrences. Ensuite, les moyennes finales sont calculées pour chaque marque, en divisant chaque colonne par le nombre d'occurrences de la marque.

```
# Réduire par clé (marque) pour obtenir les sommes des colonnes
aggregated_rdd = entiers_rdd.map(lambda x: (x[0], (x[1], x[2], x[3], 1)))
reduced_rdd = aggregated_rdd.reduceByKey(lambda x, y: (x[0] + y[0], x[1] +
y[1], x[2] + y[2], x[3] + y[3]))

# Calculer les moyennes finales pour chaque marque
moyennes_rdd = reduced_rdd.map(lambda x: (x[0], x[1][0] / x[1][3], x[1][1] /
x[1][3], x[1][2] / x[1][3]))
```

- **Conversion des résultats et enregistrement :** Les résultats finaux sont convertis en chaînes de caractères et enregistrés dans un fichier texte à l'emplacement spécifié (**/user/vagrant/output/clean_co2**).

```
# Convertir les résultats en string pour l'enregistrement
resultats_string = moyennes_rdd.map(lambda x:
f"{x[0]},{x[1]},{x[2]},{x[3]}")

# Enregistrer les résultats dans des fichiers texte
resultats_string.saveAsTextFile('/user/vagrant/output/clean_co2')
```

Exécution du script Spark :

Pour exécuter ce script et obtenir le résultat attendu, utiliser la commande suivante :

```
spark-submit $HDFSHOME/clean_map_reduce_co2.py
```

Ainsi, le résultat, contenant les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus par marque, est stocké dans le dossier **/user/vagrant/output/clean_co2** dans HDFS.

c. Création de Table Externe dans Hive

Après avoir nettoyé et transformé les données, nous allons créer des tables externes et intégrer ces données dans le catalogue dans Hive.

Création de la table externe `co2_ext` :

Cette table externe pointe vers les données nettoyées et transformées stockées dans HDFS.

```
CREATE EXTERNAL TABLE IF NOT EXISTS co2_ext (
    marque STRING,
    malusBonus FLOAT,
    rejetsCO2 FLOAT,
    coutEnergie FLOAT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/vagrant/output/clean_co2';
```

d. Intégration avec la Table Catalogue

- **Nettoyage et préparation de la table catalogue_ext :**

Pour intégrer la table **co2_ext** avec la table **catalogue_ext**, il est nécessaire de commencer par nettoyer certaines données de la table catalogue existante pour créer une table intermédiaire nommée **cleaned_catalogue_ext**.

```
CREATE TABLE cleaned_catalogue_ext AS
SELECT
    id,
    CASE
        WHEN marque LIKE '%Hyunda%' THEN 'Hyundai'
        ELSE marque
    END AS marque,
    nom,
    puissance,
    longueur,
    nbplaces,
    nbportes,
    couleur,
    occasion,
    prix
FROM catalogue_ext;
```

- **Création de la table catalogue_co2 et intégration des données :**

Ensuite, effectuer une jointure entre **cleaned_catalogue_ext** et **co2_ext** pour intégrer les données des émissions de CO2, des coûts d'énergie, et des valeurs de Bonus/Malus par marque avec les données du catalogue.

Lors de la jointure, pour les marques de voitures qui ne sont pas présentes dans **co2_ext**, insérer la moyenne des colonnes **malusBonus**, **rejetsCO2** et **coutEnergie** de toutes les marques de véhicules présentes dans les deux tables (**cleaned_catalogue_ext** et **co2_ext**).

Pour ce faire, voici les étapes à suivre :

1. Identifier les marques communes aux deux tables.
2. Calculer les moyennes des colonnes correspondantes dans **co2_ext** pour ces marques.
3. Utiliser ces moyennes pour remplacer les valeurs des marques manquantes dans **cleaned_catalogue_ext**.
4. Créer la table **catalogue_co2** en intégrant ces données.

```
CREATE TABLE IF NOT EXISTS catalogue_co2
AS
WITH marques_communs AS (
    SELECT DISTINCT co2.marque, co2.malusBonus, co2.rejetsCO2,
co2.coutEnergie
    FROM co2_ext co2
    JOIN cleaned_catalogue_ext cat
    ON LOWER(co2.marque) = LOWER(cat.Marque)
),
moyennes_co2 AS (
    SELECT
        AVG(co2.malusBonus) AS avg_malusBonus,
        AVG(co2.rejetsCO2) AS avg_rejetsCO2,
        AVG(co2.coutEnergie) AS avg_coutEnergie
    FROM co2_ext co2
    JOIN marques_communs communs
    ON LOWER(co2.marque) = LOWER(communs.marque)
)
SELECT
    cat.*,
    COALESCE(co2.malusBonus, moyennes_co2.avg_malusBonus) AS malusBonus,
    COALESCE(co2.rejetsCO2, moyennes_co2.avg_rejetsCO2) AS rejetsCO2,
    COALESCE(co2.coutEnergie, moyennes_co2.avg_coutEnergie) AS coutEnergie
FROM cleaned_catalogue_ext cat
LEFT JOIN co2_ext co2
ON LOWER(co2.marque) = LOWER(cat.Marque)
CROSS JOIN moyennes_co2;
```

- **Résultat :**

Cette démarche pour l'adaptation et l'intégration des données du fichier CO2.csv a permis d'enrichir la table catalogue (**catalogue_co2**) avec les informations sur les émissions de CO2, les coûts d'énergie et les valeurs de Bonus/Malus de chaque marque. Ces données pourraient potentiellement améliorer la qualité des modèles prédictifs lors des étapes ultérieures d'analyse de données.

7. Visualisation de données avec des outils de DataViz

Nous avons fait l'impasse à cette étape.

8. Analyse de données avec des outils de Machine Learning (R)

Dans le cadre de ce projet, plusieurs choix stratégiques ont été faits pour optimiser la gestion et l'analyse des données.

Nous avons opté pour **le langage R** en raison de ses puissantes capacités en analyse statistique, en machine learning, et en visualisation. R offre une vaste bibliothèque de packages spécialisés, ce qui le rend particulièrement adapté à des projets d'analyse de données de grande envergure.

Pour établir des connexions robustes et efficaces aux bases de données SQL, nous avons utilisé les packages **ODBC** et **DBI**. Ces outils nous ont permis d'extraire les données nécessaires de manière fluide et sécurisée. La centralisation des données dans Hive, facilite l'analyse des données à grande échelle avec des requêtes SQL.

L'exploration préliminaire des données a été réalisée à l'aide de statistiques descriptives et de visualisations. Cette étape est cruciale pour comprendre la distribution des variables et identifier les anomalies potentielles, ce qui permet de préparer les données de manière optimale pour les étapes ultérieures de modélisation.

Ces choix méthodologiques et technologiques ont été guidés par la nécessité d'assurer une analyse précise et efficiente des données.

a. Connexion aux Sources de Données :

Pour accéder aux données nécessaires à notre analyse, nous avons utilisé R pour établir une connexion aux tables externes présentes dans Hive, comme décrit dans le script **« 1-DriverConnection.R ».** Ce script utilise les bibliothèques « RJDBC », « DBI », « rJava » et « odbc » pour se connecter à Hive et importer les données directement dans R.

Cette configuration nous a permis de centraliser les données dans Hive et de les importer directement dans R pour une analyse efficace.

b. Analyse exploratoire des données :

L'exploration des données est une étape cruciale dans le processus d'analyse, car elle permet de comprendre la structure, la qualité et les caractéristiques des données avant d'appliquer des techniques de modélisation avancées. Cette phase vise à identifier les anomalies, les valeurs manquantes et les tendances générales dans les jeux de données. Voici comment nous avons procédé pour explorer les données de notre projet (**2-DataExploration.R).**

i. Chargement des Données :

Nous avons commencé par charger les bibliothèques nécessaires et les données à partir des tables Hive dans notre environnement R.

ii. Inspection Initiale des Datasets :

Une fois les données chargées, nous avons effectué une inspection initiale pour comprendre la structure et le contenu des jeux de données. Cette inspection comprend l'utilisation de fonctions telles que `str()`, `names()` et `summary()` pour obtenir un aperçu des données.

iii. Nettoyage et Préparation des Données :

Lors de l'exploration des données, nous avons identifié certaines anomalies et valeurs incorrectes que nous avons corrigées pour assurer la qualité des données avant l'analyse. Les étapes de nettoyage incluent la mise à jour des colonnes avec des valeurs incorrectes, la correction des âges et taux négatifs, et le nettoyage des valeurs de sexe et de situation familiale.

Voici un extrait des principales opérations de nettoyage effectuées :

```
# Mettre à jour la colonne 'marketing.situation familiale'
marketing$situation_familiale <- tolower(marketing$situation_familiale)
marketing$situation_familiale <- with(marketing, ifelse(situation_familiale
=="c�libataire", "celibataire",situation_familiale))
marketing$situation_familiale <- with(marketing, ifelse(situation_familiale
=="célibataire", "celibataire",situation_familiale))


# concernant l'age nous avons des anomalies avec des ages negatifs. Pour
corriger cette anomalie nous allons remplacer les ages negatifs par la
mediane : 41
clients$age <- with(clients, ifelse(age < 0, 41 ,age))
# de meme pour le taux avec une mediane à 521
clients$taux <- with(clients, ifelse(taux < 0, 521 ,taux))

#pour le sexe du client
# Supprimer les caracteres speciaux
```

```r
clients$sexe <- gsub("[^A-Za-z]", "", clients$sexe)

# Mettre en majuscules
clients$sexe <- toupper(clients$sexe)

# Supprimer les espaces vides
clients$sexe <- gsub("\\s+", "", clients$sexe)

# Remplacer les valeurs
clients$sexe <- ifelse(clients$sexe %in% c("MASCULIN", "HOMME"), "M",
clients$sexe)
clients$sexe <- ifelse(clients$sexe %in% c("FEMININ", "FEMME"), "F",
clients$sexe)
clients$sexe <- ifelse(clients$sexe %in% c("FEMININ", "FEMME"), "F",
clients$sexe)
```

iv. Visualisation des Données :

Pour mieux comprendre la distribution et les caractéristiques des données, nous avons créé des visualisations à l'aide de la bibliothèque ggplot2. Nous avons réalisé les principales visualisations suivantes : un histogramme des âges des clients montrant une répartition uniforme avec une majorité moins de 50 ans, un barplot des marques de voitures indiquant que certaines marques sont plus populaires que d'autres, un boxplot de la puissance en fonction de la longueur révélant que les voitures plus longues tendent à avoir plus de puissance, et un scatter plot de la relation entre puissance et prix montrant que les voitures plus longues et très longues sont plus puissantes et plus chères, tandis que les voitures courtes sont moins puissantes et moins chères.

La **heatmap** des corrélations entre les variables numériques révèle des relations importantes entre les variables numériques. Nous avons observé une forte corrélation positive entre la puissance et le prix des voitures, ainsi qu'entre les rejets de CO2 et le coût énergétique. Une très forte corrélation positive a été notée entre MalusBonus et RejetsCO2, suggérant que ces deux variables sont presque interchangeables en termes de valeur informative. En revanche, il y a une faible corrélation négative entre le nombre de places et le prix, et une corrélation positive modérée entre le nombre de portes et le prix, ainsi qu'entre la puissance et le nombre de portes. Ces observations nous aident à réduire la redondance et à identifier les variables importantes pour le clustering.

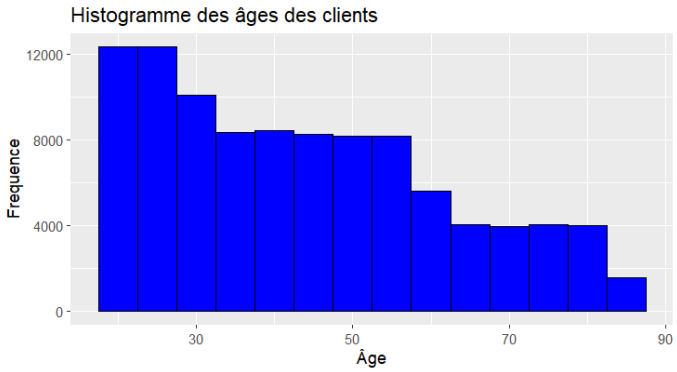

Figure 3: Histogramme des âges des clients

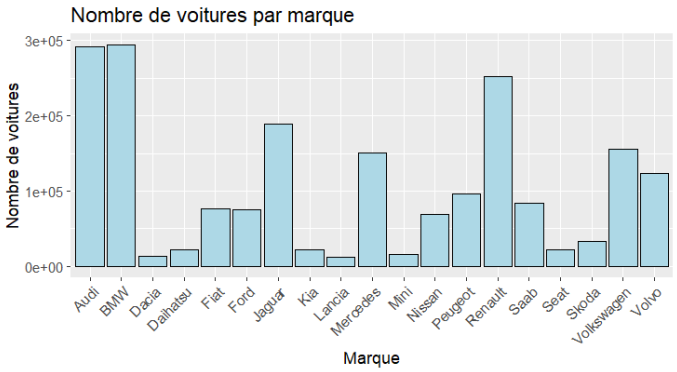

Figure 2: Barplot du nombre de voitures par marque

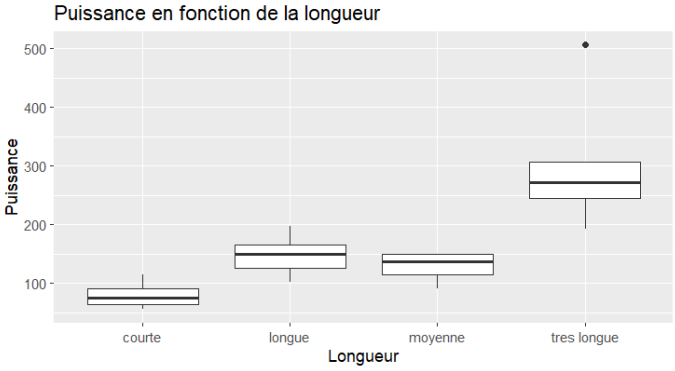

Figure 4: Boxplot de la puissance en fonction de la longueur

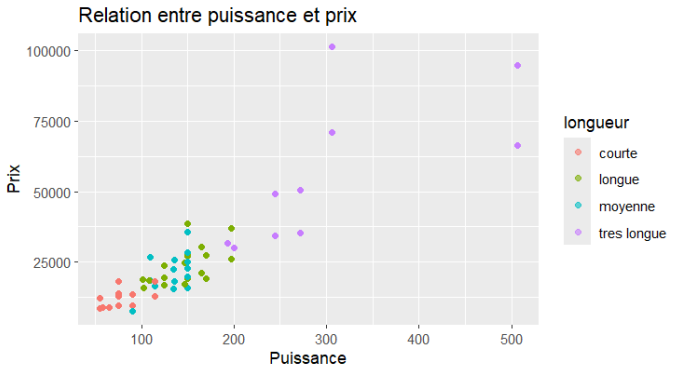

Figure 5: Scatter plot de la relation entre puissance et prix

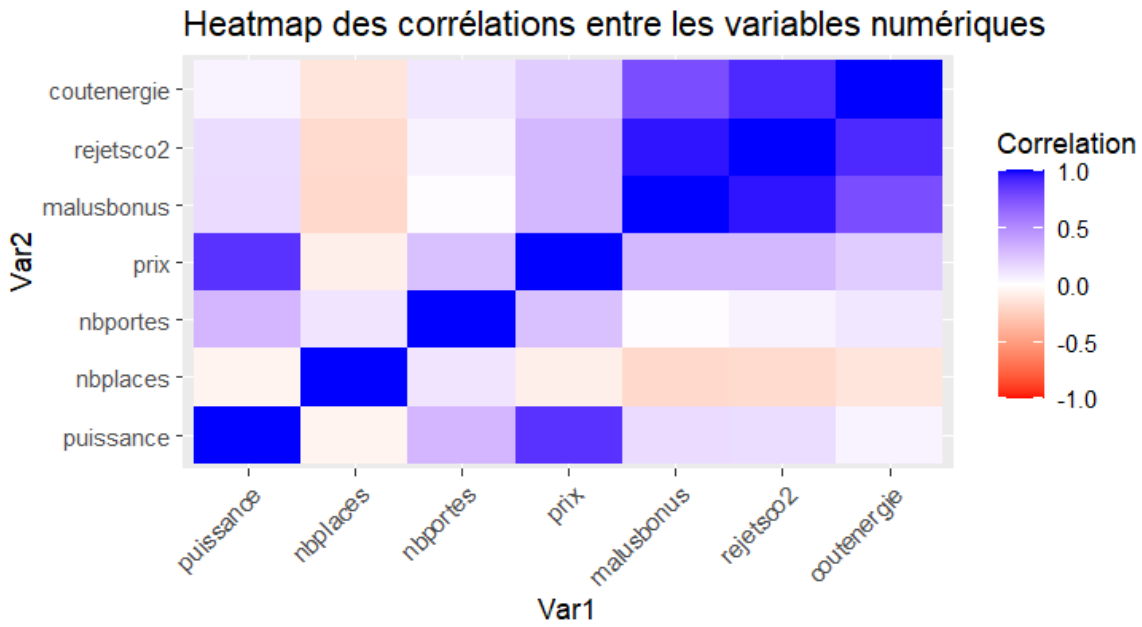

Figure 6: Heatmap des corrélations entre les variables numériques

Pour plus de détails sur les étapes spécifiques de chargement, inspection, visualisation et nettoyage des données, vous pouvez vous référer au script complet **« 2-DataExploration.R` ».**

Ces étapes nous ont permis d'identifier et de corriger les anomalies dans les données, garantissant ainsi leur qualité et leur fiabilité pour les analyses ultérieures.

c. Identification des catégories de véhicules :

L'objectif de cette section est de catégoriser les véhicules en différentes catégories pertinentes afin de mieux comprendre les segments de marché et d'optimiser les analyses ultérieures. Le fichier **« 3-categoriesVehicules.R »** décrit le processus de classification des véhicules en fonction de plusieurs critères clés. Voici une explication détaillée du contenu et des étapes suivies dans ce script.

i. Introduction à la Catégorisation des Véhicules

Nous avons utilisé une approche de clustering hiérarchique pour définir les catégories de véhicules basées sur des critères tels que la puissance, la longueur, le nombre de places et de portes, le malus/bonus, les rejets de CO2, le coût énergétique, et le prix. L'objectif est de regrouper les véhicules en segments homogènes pour des analyses plus ciblées.

ii. Principales Étapes du Processus

- **Préparation des Données :** Les données pertinentes ont été sélectionnées et normalisées pour préparer le clustering.
- **Clustering Hiérarchique :** Un clustering hiérarchique a été effectué en utilisant la méthode `ward.D` pour regrouper les véhicules en clusters.
- **Analyse des Clusters :** Les clusters ont été analysés pour déterminer les caractéristiques dominantes de chaque groupe.
- **Définition des Catégories :** Les clusters ont été mappés aux catégories spécifiques telles que citadine, sportive, familiale, confort, longue, luxe, et écologique, en fonction des caractéristiques analysées.

iii. Résultats du Clustering

Les véhicules ont été catégorisés en fonction des caractéristiques dominantes des clusters. Nous avons appliqué nos observations sur la visualisation des données pour définir ces catégories.

Voici un résumé des catégories définies :

- **Sportive :** Véhicules avec une puissance élevée.
- **Luxe :** Véhicules avec un prix moyen élevé.
- **Familiale :** Véhicules avec un nombre de places élevé.
- **Citadine :** Véhicules avec une puissance moyenne faible.
- **Écologique :** Véhicules avec de faibles rejets de CO2.
- **Confort** : Véhicules avec plus de trois portes.

d. Application des catégories de véhicules définies aux données des Immatriculations :

Après avoir défini les catégories de véhicules, ces catégories ont été appliquées aux données des immatriculations pour enrichir les informations disponibles. Cela a permis de relier chaque immatriculation à une catégorie de véhicule spécifique, facilitant ainsi une analyse plus détaillée des segments de marché. Les étapes détaillées de ce processus sont également décrites dans le fichier **« 3-categoriesVehicules.R ».**

```
# Convertir toutes les valeurs de la colonne "marque" en minuscules pour la
fusion insensible à la casse
immatriculation$marque <- tolower(immatriculation$marque)
catalogue$marque <- tolower(catalogue$marque)
catalogue$couleur <- tolower(catalogue$couleur)
immatriculation$couleur <- tolower(immatriculation$couleur)

# Fusionner les données
immatrCatalog <- merge(x = immatriculation, by = c(
"marque","nom","puissance", "longueur", "nbplaces", "nbportes","couleur",
"prix"), y = catalogue )
immatrCatalog <- unique(immatrCatalog)
View(immatrCatalog)
```

e. Fusion des données Clients et Immatriculations :

Ensuite, nous avons fusionné les données des clients avec les données des immatriculations enrichies. Cela nous a permis de créer un ensemble de données complet, intégrant les informations clients et véhicules, ce qui est essentiel pour une analyse cohérente et complète. Ce processus est détaillé dans le fichier **« 3-categoriesVehicules.R »**.

```
clientsImmat <- merge(x = clients, by = c( "immatriculation"), y =
immatrCatalog )
clientsImmat <- unique(clientsImmat)

# Vérifier le nombre d'éléments dans chaque catégorie
print(table(catalogue$categorie))
print(table(immatrCatalog$categorie))
print(table(clientsImmat$categorie))
```

Cette fusion des données a facilité une analyse détaillée des préférences et des comportements des clients en fonction des catégories de véhicules. En ayant un ensemble de données intégrant les informations sur les clients et les véhicules, nous avons pu effectuer des analyses plus précises et pertinentes pour le marché cible.

f. Création d'un modèle de classification supervisée pour la prédiction de la catégorie de véhicules à partir de la fusion des données clients et immatriculations :

Dans cette section, nous avons appliqué plusieurs modèles de classification supervisée pour prédire les catégories de véhicules en fonction des caractéristiques des clients. Les scripts **« 4-ClassificationTree.r »**, **« 5-ClassificationRandomForest.R »**, **« 6-ClassificationNeuralNetworks.R »**, **« 7-ClassificationNaiveBayes.R »**, **«8-ClassificationKNN.R »** décrivent l'utilisation de différentes techniques de classification.

i. Choix des Colonnes Utilisées

Lors de la mise en place des modèles de classification, nous avons sélectionné les colonnes suivantes :

- **Classe à Prédire :** La catégorie des véhicules (`categorie`).
- **Variables Prédictives :** Âge (`age`), Sexe (`sexe`), Taux (`taux`), Situation familiale (`situation_familiale`), Nombre d'enfants à charge (`nbr_enfant`), Deuxième voiture (`voiture_2`).
- **Variables Ignorées :** Immatriculation (`immatriculation`), Marque (`marque`), Nom (`nom`), Puissance (`puissance`), Longueur (`longueur`), Nombre de places (`nbplaces`), Nombre de portes (`nbportes`), Couleur (`couleur`), Occasion (`occasion`), Prix (`prix`).

Cette sélection a été guidée par l'objectif de prédire la catégorie de véhicule en fonction des caractéristiques des clients.

ii. Étapes Communes aux Modèles de Classification

Pour chaque modèle de classification, les étapes suivantes ont été suivies :

1. **Préparation des Données** : Division des données en ensembles d'entraînement et de test.
2. **Construction du Modèle :** Utilisation des bibliothèques appropriées pour construire le modèle.
3. **Prédiction** : Application du modèle aux données de test pour effectuer des prédictions.

iii. Spécificités des Modèles

1. Arbre de Décision (`4-ClassificationTree.r`) :

L'arbre de décision est un modèle simple et interprétable qui divise les données en segments basés sur les variables prédictives. Le script 4-ClassificationTree.r utilise les bibliothèques rpart et caret pour construire et évaluer l'arbre de décision. Le modèle est construit en utilisant la fonction rpart, et les prédictions sont effectuées avec la fonction predict.

2. **Forêt Aléatoire (`5-ClassificationRandomForest.R`) :**

La forêt aléatoire améliore la précision en combinant plusieurs arbres de décision. Le script 5-ClassificationRandomForest.R utilise les bibliothèques randomForest et caret. Le modèle est construit en utilisant la fonction randomForest, et les prédictions sont faites en utilisant predict.

3. **Réseaux de Neurones (`6-ClassificationNeuralNetworks.R`) :**

Les réseaux de neurones capturent des relations complexes entre les variables grâce à une architecture de réseau. Le script 6-ClassificationNeuralNetworks.R utilise la bibliothèque neuralnet pour construire et évaluer le modèle. Le modèle est construit en utilisant la fonction neuralnet, et les prédictions sont obtenues avec la fonction compute.

4. **Naive Bayes (`7-ClassificationNaiveBayes.R`) :**

Naive Bayes utilise des probabilités pour classer les données de manière rapide et efficace. Le script 7-ClassificationNaiveBayes.R utilise la bibliothèque e1071 pour construire et évaluer le modèle. Le modèle est construit en utilisant la fonction naiveBayes, et les prédictions sont effectuées avec predict.

5. **K-Nearest Neighbors (KNN) (`8-ClassificationKNN.R`):**

KNN classe les données en fonction de la proximité des points de données dans l'espace des caractéristiques. Le script 8-ClassificationKNN.R utilise la bibliothèque class pour appliquer l'algorithme KNN. Les prédictions sont faites en utilisant la fonction knn.

iv. Comparatif des Résultats

Pour mieux visualiser et comparer les performances des différents modèles, nous avons résumé les résultats obtenus dans le tableau ci-dessous :

Modèles	Tests	Prédiction
Arbre de Décision C5.0	taux_C51	0.745079
	taux_C52	0.745814
	taux_C53	0.745079
	taux_C54	0.745814
K-Nearest Neighbors	taux_knn1	0.672660
	taux_knn2	0.696153
	taux_knn3	0.673061
	taux_knn4	0.696087
Naive Bayes	taux_nb1	0.704408
	taux_nb2	0.704441
Réseaux de Neurones	taux_nn1	0.719112
	taux_nn2	0.706480
	taux_nn3	0.666310
	taux_nn4	0.718611
Forêt Aléatoire	taux_rf1	0.744912
	taux_rf2	0.745179
	taux_rf3	0.735421
	taux_rf4	0.735220

Arbre de Décision rpart	taux_rp1	0.742606
	taux_rp2	0.742606
	taux_rp3	0.742606
	taux_rp4	0.742606
Arbre de Décision tree	taux_tr1	0.742606
	taux_tr2	0.742606

Pour plus de détails sur la mise en œuvre et l'évaluation de chaque modèle, vous pouvez vous référer aux scripts complets respectifs.
Après avoir évalué les différents modèles de classification, nous avons constaté que l'algorithme C5.0 a obtenu le meilleur taux de succès. Nous avons donc utilisé C5.0 pour prédire les catégories des données marketing, car il a démontré la précision la plus élevée parmi tous les modèles testés.

g. Application du modèle de prédiction aux données Marketing :

Dans cette section, nous avons appliqué le modèle de prédiction aux données marketing afin de prédire les catégories des véhicules pour les clients potentiels. Nous avons utilisé le modèle C5.0, qui a montré le meilleur taux de succès lors de notre évaluation.

- **Prétraitement des Données Marketing :**

Avant d'appliquer le modèle, nous avons prétraité les données marketing pour nous assurer qu'elles étaient dans le bon format pour la prédiction.

```
# Prétraitement des données marketing
marketing$sexe <- as.factor(marketing$sexe)
marketing$situation_familiale <- as.factor(marketing$situation_familiale)
```

- **Application du Modèle C5.0 :**

Nous avons utilisé le modèle C5.0 pour prédire les catégories des véhicules en fonction des caractéristiques des clients présents dans les données marketing.

```
# Utilisation du modèle C5.0 avec le meilleur taux de succès pour prédire
les catégories des données marketing
marketing_predictions <- predict(tree_C54, marketing)
```

- **Ajout des Prédictions aux Données Marketing :**

Les prédictions obtenues ont été ajoutées aux données marketing, ce qui nous a permis de voir quelle catégorie de véhicule était attribuée à chaque client potentiel.

```
# Ajout des prédictions aux données marketing
marketing$predicted_categorie <- marketing_predictions

# Affichage des résultats
print(marketing)
```

- **Résultats :**

Les résultats montrent les catégories de véhicules prédites pour chaque client potentiel dans les données marketing. Cela nous permet de cibler plus efficacement les clients avec des offres de véhicules qui correspondent à leurs caractéristiques et préférences.

	marketing_result.id	marketing_result.age	marketing_result.sexe	marketing_result.taux	marketing_result.situation_familiale	marketing_result.nbr_enfant	marketing_result.voiture_2	marketing_result.predicted_categorie
1	4	26	F	420	en couple	3	TRUE	confort
2	20	59	M	748	en couple	0	TRUE	confort
3	15	60	M	524	en couple	0	TRUE	confort
4	2	35	M	223	celibataire	0	FALSE	confort
5	8	43	F	431	celibataire	0	FALSE	confort
6	12	55	M	588	celibataire	0	FALSE	confort
7	13	19	F	212	celibataire	0	FALSE	confort
8	6	27	F	153	en couple	2	FALSE	confort
9	10	22	M	154	en couple	1	FALSE	confort
10	1	21	F	1396	celibataire	0	FALSE	confort
11	7	59	F	572	en couple	2	FALSE	confort
12	16	22	M	411	en couple	3	TRUE	confort
13	17	58	M	1192	en couple	0	FALSE	confort
14	3	48	M	401	celibataire	0	FALSE	confort
15	9	64	M	559	celibataire	0	FALSE	confort
16	14	34	F	1112	en couple	0	FALSE	confort
17	18	54	F	452	en couple	3	TRUE	confort
18	5	80	M	530	en couple	3	FALSE	luxe
19	19	35	M	589	celibataire	0	FALSE	confort
20	11	79	F	981	en couple	2	FALSE	confort

Figure 7: Résultat prédiction marketing

h. Insertion du résultat dans Hive :

Après avoir prédit les catégories de véhicules pour les clients potentiels, l'étape suivante consiste à insérer ces résultats dans la base de données Hive pour une utilisation ultérieure. Voici comment nous avons procédé :

- Connexion à Hive
- Création de la Table `marketing_result`
- Préparation des Données pour l'Insertion
- Créer la requête SQL d'insertion
- Vérification des Données Insérées

9. Conclusion générale

Bilan des Résultats Obtenus pour le Projet

Le projet Big Data Analytics, visant à analyser la clientèle d'un concessionnaire automobile pour recommander des modèles de véhicules, a été une réussite sur plusieurs fronts. Nous avons mis en place un Data Lake robuste intégrant diverses sources de données telles que HDFS, MongoDB, et Oracle NoSQL. Les méthodes avancées de traitement des données, notamment l'utilisation de Hadoop Map Reduce et du langage R, nous ont permis de manipuler efficacement de vastes ensembles de données. L'analyse des données a conduit à des insights précieux qui ont aidé à mieux comprendre les comportements des consommateurs et à optimiser les recommandations de véhicules. L'utilisation de modèles de classification supervisée, avec une préférence pour l'algorithme C5.0 en raison de son taux de succès élevé, a permis de prédire avec précision les catégories de véhicules adaptées aux clients.

Bilan des Problèmes Rencontrés et des Solutions Apportées

Au cours du projet, nous avons rencontré plusieurs défis techniques et opérationnels. L'un des principaux problèmes était la gestion et le nettoyage des données provenant de sources diverses et disparates. Pour résoudre ce problème, nous avons utilisé des scripts de nettoyage et des méthodes de transformation de données robustes, notamment avec Spark pour les traitements Map Reduce. Un autre défi majeur a été l'intégration des données dans un environnement centralisé (Hive) et la connexion de ces données à nos outils d'analyse (R). Nous avons surmonté ces obstacles en configurant des connexions stables via ODBC et en utilisant des packages spécialisés pour l'importation et la manipulation des données. En conclusion, ce projet a démontré le potentiel des technologies Big Data et des méthodes d'analyse avancées pour transformer la gestion des données et améliorer les opérations commerciales dans le secteur automobile. Les résultats obtenus, les solutions apportées aux défis rencontrés montrent que ce projet est une base solide pour des développements futurs et des applications plus larges.

10. Références et Bibliographie

Documents de Nicolas PASQUIER

[1] Nicolas Pasquier, Diaporama 01 - Introduction au Logiciel R
[2] Nicolas Pasquier, Diaporama 02 - Introduction à la Classification Supervisée
[3] Nicolas Pasquier, Diaporama 03 - Apprentissage d'Arbres de Décision
[4] Nicolas Pasquier, Diaporama 04 - Représentations des Arbres de Décision
[5] Nicolas Pasquier, Diaporama 05 - Paramétrage de l'Apprentissage
[6] Nicolas Pasquier, Diaporama 06 - Matrices de Confusion
[7] Nicolas Pasquier, Diaporama 07 - Analyse Exploratoire des Données
[8] Nicolas Pasquier, Diaporama 08 - Introduction au Clustering de Données
[9] Nicolas Pasquier, Diaporama 09 - Clustering par Partitionnement et Hiérarchique

Documents de Gabriel MOPOLO-MOKE

[10] Gabriel Mopolo-Moke, M4.1 Concepts Big Data Et SGBD NoSql
[11] Gabriel Mopolo-Moke, M4.1 Concepts Big Data Et SGBD NoSql_modified
[12] Gabriel Mopolo-Moke, M4.2_4.3 Bd Du BigData Cas ORACLE NOSQL
[13] Gabriel Mopolo-Moke, M4.2_4.3 Bd Du BigData Cas ORACLE NOSQL_modified
[14] Gabriel Mopolo-Moke, M4.4_4.5 Bd Du BigData INTRODUCTION A MONGODB
[15] Gabriel Mopolo-Moke, M4.6 Architectures BigData et Lacs de Données Avec Big Data SQL par la pratique
[16] Gabriel Mopolo-Moke, M4.6 Architectures BigData et Lacs de Données Avec Big Data SQL par la pratique_modified

Documents de Sergio SIMONIAN

[17] Sergio Simonian, HadoopSparkMapReduce_1
[18] Sergio Simonian, HadoopSparkMapReduce_2
[19] Sergio Simonian, HadoopSparkMapReduce_3

Sites internets

[20] A propos de R, What is R? - The Statistical Computing Powerhouse | DataCamp
[21] Programmation R Aide mémoire R (duclert.org)
[22] Tout savoir sur Hadoop et ses avantages, https://www.talend.com/fr/resources/what-is-hadoop/ ,talend
[23] What is R? - The Statistical Computing Powerhouse, https://www.datacamp.com/blog/all-about-r , dataCamp
[24] classification par arbres décisionnels , http://perso.ens-lyon.fr/lise.vaudor/classification-par-arbres-decisionnels/#:~:text=L'arbre%20de%20d%C3%A9cision%20est,%C3%A0%20une%20mesure%20appel%C3%A9e%20impuret%C3%A9. , R-atique ,date de parution : 15/12/2014

11. Annexes

a. Vidéo de présentation

Le lien de la vidéo de présentation du projet est le suivant : https://drive.google.com/file/d/1gbPjelBLLrzj_ZK0fE58_MR2ZlnhHpDc/view?usp=sharing

b. Dossier contenant les scripts et programmes de construction du lac de données

Le dossier contenant les scripts et les programmes pour la construction du lac de donnée se trouve dans le dossier datalake : MBDS_GRP5/DATA_EXTRACTOR at main · Flavien008/MBDS_GRP5 (github.com)

c. Dossier contenant les scripts et programmes Hadoop Map Reduce

Le programme concernant Hadoop Map Reduce se trouve dans le dossier mapreduce MBDS_GRP5/DATA_EXTRACTOR/hdfs at main · Flavien008/MBDS_GRP5 (github.com)

d. Dossier contenant les scripts et programmes d' analyse de données

Le dossier contentant les scripts se trouve dans le dossier datascience(MBDS_GRP5/AnalyseR at main · Flavien008/MBDS_GRP5 (github.com)