



UNIVERSITÉ
CÔTE D'AZUR



Projet Big Data et d'Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèle

Cao Nicolas - Guinaldo Vincent - Poirier Corentin
Groupe 7

2020-2021

Table des matières

PARTIE Gestion des Données	4
1	Objectif et contexte de la Gestion des données 4
2	Le chargement de sources de données sur la base MongoDB 6
3	Le chargement de sources de données sur la base Oracle NoSQL 7
3.1	Fichier Immatriculations 7
3.1.1	Création du fichier de chargement des données 7
3.1.2	Chargement des données 8
3.1.3	Vérification du chargement des données 9
3.2	Fichier Marketing 9
3.2.1	Création du fichier de chargement des données 9
3.2.2	Chargement des données 10
3.2.3	Vérification du chargement des données 11
4	Le chargement de source de données via Hadoop HDFS 12
4.1	Fichier Marketing 13
4.1.1	Chargement des données 13
4.1.2	Vérification du chargement des données 13
5	La création de tables externes sur Hive 14
5.1	Connexion à Hive 14
5.2	Depuis le KvStore 14
5.2.1	Table immatriculation 15
5.2.1.1	Création de la table externe 15
5.2.1.2	Vérification de la création de la table externe 15
5.2.2	Table marketing 16
5.2.2.1	Création de la table externe 16
5.2.2.2	Vérification de la création de la table externe 16
5.3	Depuis HDFS 16
5.3.1	Table catalogue 17
5.3.1.1	Création de la table externe 17
5.3.1.2	Vérification de la création de la table externe 17
6	Le chargement de sources de données sur la base Oracle SQL 18
6.1	Création de la table client 18
6.2	Population de la table client 18
6.3	Vérification de la création de la table client 19
7	Création et récupération des tables externes dans SQLPlus 20
7.1	Connexion à la base 20
7.2	Table immatriculation 20
7.2.1	Création de la table 20
7.2.2	Vérification de la création de la table 21
7.3	Table marketing 22
7.3.1	Création de la table 22
7.3.2	Vérification de la création de la table 22
7.4	Table catalogue 23
7.4.1	Création de la table 23
7.4.2	Vérification de la création de la table 23

8	Récupération et affichage des données dans R Studio	24
8.1	Création d'un utilisateur sur la base	24
8.2	Utilisation de R Studio	25
8.2.1	Connexion sur R Studio	25
8.2.2	Création des data-frames sur R Studio	25
PARTIE Analyse de Données		26
9	Objectif et contexte de l'Analyse de données	26
10	Chargement et Nettoyage des données sur R	27
10.1	Import des fichiers .csv	27
10.2	Analyse exploratoire des données sur le fichier Immatriculations	27
10.2.1	Histogramme de la variable longueur	28
10.2.2	Histogramme de la variable nombre de portes	29
10.2.3	Histogrammes des variables marque, couleur et occasion	29
10.2.4	Opération post-analyse	30
10.2.5	Conclusion de l'analyse du fichier Immatriculations	30
10.3	Analyse exploratoire des données sur le fichier Clients	31
10.3.1	Opérations sur la variable Age	32
10.3.2	Opérations sur la variable Sexe	32
10.3.3	Opérations sur la variable Taux	34
10.3.4	Opérations sur la variable situationFamiliale	34
10.3.5	Opérations sur la variable nombreEnfantsAcharge	35
10.3.6	Conclusion de l'analyse du fichier Clients	36
11	Construction des ensembles de tests et d'apprentissages	37
11.1	Création des catégories de véhicules	37
11.2	Ensemble de tests et d'apprentissages	39
11.2.1	Suppression de variables sur le data-frame	40
11.2.2	Modification des variables sur le data-frame	41
11.2.3	Création des jeux de données	41
11.2.4	Création des catégories de taux	42
12	Test des classifieurs et création du modèle de prédiction	45
12.1	Liste des classifieurs	45
12.2	1er jeu de données	45
12.2.1	C5.0	46
12.2.2	Naive-Bayes	48
12.2.3	SVM	49
12.2.4	Kknn	50
12.2.5	Nnet	51
12.2.6	randomForest	52
12.2.7	Conclusion du 1er jeu de données	53
12.3	2eme jeu de données	53
12.3.1	C5.0	54
12.3.2	Naive Bayes	55
12.3.3	SVM	56
12.3.4	Kknn	57
12.3.5	Nnet	58
12.3.6	randomForest	59
12.3.7	Conclusion du 2ème jeu de données	60
13	Prédiction	61
13.1	Création des data-frames	61

13.2	Classifieur prédit	61
13.3	Modification du data-frame	61
13.4	Prédiction sur le data-frame Marketing	62
14	Conclusion de l'Analyse de données	63
PARTIE Analyse de Données		64
15	Objectif et contexte de la partie MapReduce	64
16	Création des scripts Java	65
16.1	Fichier CO2.java	65
16.2	Fichier CO2Map.java	65
16.3	Fichier CO2Reduce.java	66
17	Programme Map Reduce	67
17.1	Import des fichiers Java	67
17.2	Compilation des fichiers Java	67
17.3	Exécution du programme Map Reduce	68
18	Résultat du programme Map Reduce	69
18.1	Résultat sur HDFS	69
18.2	Export du fichier depuis HDFS	69
19	Jonction avec le fichier catalogue	70
19.1	Conversion en csv	70
19.2	Jonction sur R Studio	70
19.2.1	Import des fichiers	71
19.2.2	Modification des colonnes	71
19.2.3	Fusion des data-frames	71
19.2.4	Nettoyage du data-frame	71
19.2.5	Export du data-frame	72
Annexe		73
	Importation des données via un driver SQL Plus	73
	Tri des données sur SQL Plus	74
	Prédiction à l'issue de l'import via la Gestion des Données	75
Références		77

PARTIE Gestion des Données

1 Objectif et contexte de la Gestion des données

Nous avons été contactés par un concessionnaire automobile afin de l'aider à mieux cibler les véhicules susceptibles d'intéresser les potentiels clients.

L'interview du gestionnaire de la concession automobile nous a permis de définir le contexte et les objectifs de l'application : « Les différents véhicules de notre catalogue répondent à des besoins différents. Certains sont petits afin de mieux circuler en ville, d'autres ont de l'espace pour transporter toute une famille tandis que certains sont plus puissants et destinés à une clientèle plus fortunée. Nous souhaitons définir différentes catégories de véhicules afin de mieux comprendre les désirs des clients et proposer aux nouveaux clients le véhicule le plus adapté à leurs besoins. ».

Ainsi, à travers cette demande, notre but est de construire un modèle prédiction à travers l'utilisation de différentes méthodes de classification supervisée. Ainsi, le concessionnaire pourra subvenir plus facilement aux besoins des clients en matière de véhicules. Enfin, ce modèle reposera sur les caractéristiques du client (Par exemple : son âge, le nombre d'enfants, ...).

En plus de construire cela sur les caractéristiques du client, nous nous baserons sur les informations qui concernent les immatriculations effectuées cette année.

La réalisation de ce projet va nécessiter la mise en œuvre de notions sur les thématiques du *BigData*, de la *DataVisualisation* et du *MachineLearning*.

Pour réaliser cela, quatre fichiers nous sont fournis par le concessionnaire :

- Un fichier, cité précédemment, concernant les immatriculations effectuées au cours de l'année, *Immatriculation.csv*
- Un fichier concernant le profil des clients ayant réalisé un achat de voiture sur l'année en cours, *Client_6.csv*
- Un fichier concernant le catalogue de véhicules, *Catalogue.csv*
- Enfin un fichier concernant le profil des clients sélectionnés par le service marketing pour créer ce modèle prédictif, *Marketing.csv*

L'objectif de cette première partie est de récupérer l'ensemble des données à des fins d'analyse en temps réel. Plusieurs méthodes sont réalisables. Pour ce projet, nous allons mettre en place une méthode différente pour chaque fichier que le concessionnaire nous donne.

De ce fait, plusieurs étapes vont être cruciales pour la mise en place de la gestion des données.

1. Le chargement de sources de données sur la base MongoDB.
 - Chargement des données du fichier *Marketing.csv*
2. Le chargement de sources de données sur la base Oracle NoSQL.
 - Chargement des données du fichier *Immatriculations.csv*
 - Chargement des données du fichier *Marketing.csv*
3. Le chargement de sources de données via Hadoop HDFS.
 - Chargement des données du fichier *Catalogue.csv*
4. La création de tables externes sur Hive.
 - Création depuis le KvStore
 - Création depuis HDFS
5. Le chargement de sources de données sur la base Oracle SQL.
 - Chargement des données du fichier *Clients_6.csv*
 - Utilisation de SQL Loader
6. Création et récupération des tables externes dans SQLPlus.
7. Récupération et affichage des données dans R Studio.

2 Le chargement de sources de données sur la base MongoDB

Nous nous sommes intéressé à la base NoSQL MongoDB dans un premier temps. L'utilisation de MongoDB va nous permettre de charger le fichier Marketing. L'un des avantages de MongoDB par rapport au SQL, c'est sa capacité à gérer des systèmes de données complexes, ici nous possédons 4 fichiers complexes, d'où l'intérêt d'utiliser ce type de base de données[1]. On peut avoir des listes, des objets encapsulés sans avoir de soucis. Ce fonctionnement facilite grandement le développement d'applications qui gèrent beaucoup de données, comme notre analyse dans la prochaine partie.

Pour ce faire, nous avons importé le fichier avec la commande suivante, voir Figure(1).

```
C:\Program Files\MongoDB\Server\4.4\bin>mongoimport -d concessionnaire -c marketing --type csv --file "C:/Marketing.csv" --headerline
2021-03-24T10:43:35.231+0100    connected to: mongodb://localhost/
2021-03-24T10:43:35.470+0100    20 document(s) imported successfully. 0 document(s) failed to import.
```

FIGURE 1 – Importation du fichier .csv dans MongoDB.

Afin de pouvoir importer le fichier dans Hive, il nous faut exporter les datas Marketing vers un fichier JSON. L'export se fait de la manière suivante, Figure(2).

```
C:\Program Files\MongoDB\Server\4.4\bin>mongoexport -d concessionnaire -c marketing -o Marketing.json
2021-03-24T10:43:42.037+0100    connected to: mongodb://localhost/
2021-03-24T10:43:42.338+0100    exported 20 records
```

FIGURE 2 – Exportation du fichier JSON depuis MongoDB.

Nous pouvons voir le résultat de l'exportation, voir Figure(3).

```
[{"_id":{"_id":"605b09c7310273dded67748e"},"age":21,"sexe":"F","taux":1396,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded67748f"},"age":26,"sexe":"F","taux":420,"situationFamiliale":"En Couple","nbEnfantsAcharge":3,"2eme voiture":"true"},
{"_id":{"_id":"605b09c7310273dded677490"},"age":80,"sexe":"M","taux":530,"situationFamiliale":"En Couple","nbEnfantsAcharge":3,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677491"},"age":27,"sexe":"F","taux":153,"situationFamiliale":"En Couple","nbEnfantsAcharge":2,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677492"},"age":59,"sexe":"F","taux":572,"situationFamiliale":"En Couple","nbEnfantsAcharge":2,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677493"},"age":43,"sexe":"F","taux":431,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677494"},"age":64,"sexe":"M","taux":559,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677495"},"age":48,"sexe":"M","taux":401,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677496"},"age":55,"sexe":"M","taux":588,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677497"},"age":19,"sexe":"F","taux":212,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677498"},"age":34,"sexe":"F","taux":1112,"situationFamiliale":"En Couple","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded677499"},"age":60,"sexe":"M","taux":524,"situationFamiliale":"En Couple","nbEnfantsAcharge":0,"2eme voiture":"true"},
{"_id":{"_id":"605b09c7310273dded67749a"},"age":22,"sexe":"M","taux":411,"situationFamiliale":"En Couple","nbEnfantsAcharge":3,"2eme voiture":"true"},
{"_id":{"_id":"605b09c7310273dded67749b"},"age":58,"sexe":"M","taux":1192,"situationFamiliale":"En Couple","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded67749c"},"age":22,"sexe":"M","taux":154,"situationFamiliale":"En Couple","nbEnfantsAcharge":1,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded67749d"},"age":79,"sexe":"F","taux":981,"situationFamiliale":"En Couple","nbEnfantsAcharge":2,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded67749e"},"age":35,"sexe":"M","taux":589,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"},
{"_id":{"_id":"605b09c7310273dded67749f"},"age":59,"sexe":"M","taux":748,"situationFamiliale":"En Couple","nbEnfantsAcharge":0,"2eme voiture":"true"},
{"_id":{"_id":"605b09c7310273dded6774a0"},"age":54,"sexe":"F","taux":452,"situationFamiliale":"En Couple","nbEnfantsAcharge":3,"2eme voiture":"true"},
{"_id":{"_id":"605b09c7310273dded6774a1"},"age":35,"sexe":"M","taux":223,"situationFamiliale":"Celibataire","nbEnfantsAcharge":0,"2eme voiture":"false"}]
```

FIGURE 3 – Résultat de l'exportation du fichier JSON depuis MongoDB.

Après réflexion avec l'ensemble des parties, l'import dans MongoDB n'est pas nécessaire dans ce projet. Ce fichier Marketing sera chargé dans la base NoSQL.

3 Le chargement de sources de données sur la base Oracle NoSQL

Après avoir utilisé la base MongoDB, nous nous penchons sur une nouvelle base NoSQL, OracleNoSQL. L'intérêt d'utiliser cette base est de permettre aux développeurs de créer facilement des applications à l'aide de modèles de bases de données de documents, de colonnes et de clés-valeurs, en offrant des temps de réponse très rapide (milliseconde). NoSQL correspond à « not only SQL » et c'est en effet ce que ce modèle de base de données veut être : un enrichissement et complément utile des bases de données SQL relationnelles traditionnelles. De ce fait, les bases de données NoSQL dépassent les limites des systèmes relationnels et exploitent un modèle de base de données différents. Cela ne veut toutefois pas dire qu'aucun système SQL n'est utilisé.

À l'inverse des bases de données SQL relationnelles, les bases de données NoSQL n'utilisent pas de lignes et colonnes pour le stockage des données. Elles organisent les gros volumes de données de façon flexible, tels que des documents, paires de valeurs et colonnes. C'est pourquoi, les systèmes NoSQL sont parfaitement adaptés aux applications exigeant le traitement de larges volumes de données[2].

Nous allons donc chargé deux fichiers sur OracleNoSQL.
Ces deux fichiers sont *Immatriculations.csv* et *Marketing.csv*.

3.1 Fichier Immatriculations

3.1.1 Création du fichier de chargement des données

Dans un premier temps, nous allons utilisé la base de données Oracle Kv afin de charger les données du fichier *Immatriculations.csv*.

Nous réalisons un fichier Java qui nous permet de parcourir l'ensemble du fichier puis de le charger sur la base, voir Figure(4).

```
void loadImmatriculationDataFromFile(String immatriculationDataFileName) {
    InputStream ipsr;
    BufferedReader br = null;
    InputStream ips;
    // Variables pour stocker les données lues d'un fichier.
    String ligne;
    System.out.println("***** Dans : loadImmatriculationDataFromFile *****");
    /* parcourir les lignes du fichier texte et découper chaque ligne */
    try {
        ips = new FileInputStream(immatriculationDataFileName);
        ipsr = new InputStreamReader(ips);
        br = new BufferedReader(ipsr);
        /* open text file to read data */
        //parcourir le fichier ligne par ligne et découper chaque ligne en
        //morceau séparés par le symbole ;
        while ((ligne = br.readLine()) != null) {
            //int situationFamilliale, 2eme voiture, nbPortes, prix;
            //String immatriculation, age, sexe, nbEnfantsAcharge, couleur, occasion;
            ArrayList<String> immatriculationRecord = new ArrayList<String>();
            StringTokenizer val = new StringTokenizer(ligne, ";");
            while (val.hasMoreTokens()) {
                immatriculationRecord.add(val.nextToken().toString());
            }
            String immatriculationID = immatriculationRecord.get(0);
            String marque = immatriculationRecord.get(1);
            String nom = immatriculationRecord.get(2);
            String puissance = immatriculationRecord.get(3);
            String longueur = immatriculationRecord.get(4);
            String nbPlaces = immatriculationRecord.get(5);
            String nbPortes = immatriculationRecord.get(6);
            String couleur = immatriculationRecord.get(7);
            String occasion = immatriculationRecord.get(8);
            String prix = immatriculationRecord.get(9);
            // Add the immatriculation in the KVStore
            this.insertImmatriculationRow(immatriculationID, marque, nom, puissance, longueur, nbPlaces, nbPortes, couleur, occasion, prix);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

FIGURE 4 – Fonction permettant l'importation du fichier sur OracleNoSQL.

La boucle While dans cette fonction permet l'ajout. Elle itère sur le fichier *.csv*, en extrayant les données brutes, puis appelle la fonction `insertAimmatriculationRow` qui permet d'ajouter une ligne à notre table marketing créée préalablement, voir Figure(5) et Figure(6).

```
private void insertAimmatriculationRow(String immatriculationID, String marque, String nom, String puissance,
//TableAPI tableAPI = store.getTableAPI();
StatementResult result = null;
String statement = null;
System.out.println("***** Dans : insertAimmatriculationRow *****")

try {
    TableAPI tableH = store.getTableAPI();
    // The name you give to getTable() must be identical
    // to the name that you gave the table when you created
    // the table using the CREATE TABLE DDL statement.
    Table immatriculationTable = tableH.getTable(tablImmatriculation);
    // Get a Row instance
    Row immatriculationRow = immatriculationTable.createRow();
    // Now put all of the cells in the row.
    // This does NOT actually write the data to
    // the store.

    // Create one row
    immatriculationRow.put("immatriculation", immatriculationID);
    immatriculationRow.put("marque", marque);
    immatriculationRow.put("nom", nom);
    immatriculationRow.put("puissance", puissance);
    immatriculationRow.put("longueur", longueur);
    immatriculationRow.put("nbPlaces", nbPlaces);
    immatriculationRow.put("nbPortes", nbPortes);
    immatriculationRow.put("couleur", couleur);
    immatriculationRow.put("occasion", occasion);
    immatriculationRow.put("prix", prix);

    // Now write the table to the store.
    // "item" is the row's primary key. If we had not set that value,
    // this operation will throw an IllegalArgumentException.
    tableH.put(immatriculationRow, null, null);

} catch (IllegalArgumentException e) {
    System.out.println("Invalid statement:\n" + e.getMessage());
} catch (FaultException e) {
    System.out.println("Statement couldn't be executed, please retry: " + e);
}
}
```

FIGURE 5 – Fonction permettant l'ajout d'une ligne sur la table.

```
public void createImmatriculationTable() {
    String statement = null;
    statement = "Create table " + tabImmatriculation + " ("
        + "IMMATRICULATION STRING,"
        + "MARQUE STRING,"
        + "NOM STRING,"
        + "PUISSANCE STRING,"
        + "LONGUEUR STRING,"
        + "NBPLACES STRING,"
        + "NBPORTES STRING,"
        + "COULEUR STRING,"
        + "OCCASION STRING," //boolean mais on conserve en string
        + "PRIX STRING,"
        + "PRIMARY KEY (IMMATRICULATION))";
    executeDDL(statement);
}
```

FIGURE 6 – Création de la table Immatriculation.

3.1.2 Chargement des données

Afin de charger les données il nous faut ensuite, compiler le fichier. Nous utilisons le terminal pour exécuter les commandes, voir Figure(7).

```
-- Ceci est le chemin vers notre projet sur la machine virtuelle
[oracle@bigdatalite ~]$ export MYPROJECTHOME=/home/CAO/projetMBDS/

-- Compiler le code java pour importer la table Immatriculation à partir du fichier csv
[oracle@bigdatalite ~]$ javac -g -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ $MYPROJECTHOME/voiture/DataImportImmatriculation.java

-- Exécuter le code java pour importer la table Immatriculation à partir du fichier csv
[oracle@bigdatalite ~]$ java -Xmx256m -Xms256m -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ voiture.DataImportImmatriculation
```

FIGURE 7 – Commandes pour exécuter le script Java.

3.1.3 Vérification du chargement des données

Une fois le fichier Java compilé et exécuté, il faut désormais vérifier si les données ont bien été ajoutées. Pour cela, nous nous connectons au KvStore, voir Figure(8). Puis, une fois connecté, nous affichons la table immatriculation, Figure(9).

```
-- Connection à Oracle NoSQL
[oracle@bigdatalite ~]$ java -jar $KVHOME/lib/kvstore.jar runadmin -port 5000 -host bigdatalite.localdomain

kv-> connect store -name kvstore
```

FIGURE 8 – Connection au KvStore

```
-- Vérification du contenu de la table IMMATRICULATION
kv-> get table -name IMMATRICULATION
-- Réponse
{"IMMATRICULATION":"0 AJ 71","MARQUE":"Jaguar","NOM":"X-Type 2.5 V6","PUISSANCE":"197","LONGUEUR":"longue","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 BH 31","MARQUE":"BMW","NOM":"M5","PUISSANCE":"507","LONGUEUR":"très longue","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 DQ 29","MARQUE":"Peugeot","NOM":"1007 1.4","PUISSANCE":"75","LONGUEUR":"courte","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 EA 32","MARQUE":"Jaguar","NOM":"X-Type 2.5 V6","PUISSANCE":"197","LONGUEUR":"longue","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 HF 24","MARQUE":"Peugeot","NOM":"1007 1.4","PUISSANCE":"75","LONGUEUR":"courte","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 IS 25","MARQUE":"Ford","NOM":"Mondeo 1.8","PUISSANCE":"125","LONGUEUR":"longue","NBPLACES":"5","NBPORTES":"5"},
{"IMMATRICULATION":"0 JH 10","MARQUE":"Daihatsu","NOM":"Cuore 1.0","PUISSANCE":"58","LONGUEUR":"courte","NBPLACES":"5","NBPORTES":"5"}
```

FIGURE 9 – Affichage de la table Immatriculation.

Le fichier a bien été chargé dans la base. Nous pouvons donc continuer nos importations de données, et réaliser le même principe pour le fichier Marketing.

3.2 Fichier Marketing

3.2.1 Création du fichier de chargement des données

Dans un second temps, nous utilisons de nouveau la base de données Oracle Kv afin de charger les données du fichier *Marketing.csv*.

Nous réalisons de la même manière un fichier Java qui nous permet de parcourir l'ensemble du fichier puis de le charger sur la base, voir Figure(10).

```
void loadmarketingDataFromFile(String marketingDataFileName) {
    InputStreamReader ipsr;
    BufferedReader br = null;
    InputStream ips;
    // Variables pour stocker les données lues d'un fichier.
    String ligne;
    System.out.println("***** Dans : loadmarketingDataFromFile *****");
    /* parcourir les lignes du fichier texte et découper chaque ligne */
    try {
        ips = new FileInputStream(marketingDataFileName);
        ipsr = new InputStreamReader(ips);
        br = new BufferedReader(ipsr);
        /* open text file to read data */
        //parcourir le fichier ligne par ligne et découper chaque ligne en
        //morceau séparés par le symbole ;
        br.readLine();
        String ligne = null;
        while ((ligne = br.readLine()) != null) {
            //Int situationFamilliale, 2eme voiture, nbPortes, prix;
            //String marketing, age, sexe, nbEnfantsAchange, couleur, occasion, ;
            ArrayList<String> marketingRecord = new ArrayList<String>();
            StringTokenizer val = new StringTokenizer(ligne, ";");
            while (val.hasMoreTokens()) {
                marketingRecord.add(val.nextToken().toString());
            }
            String age = marketingRecord.get(0);
            String sexe = marketingRecord.get(1);
            String taux = marketingRecord.get(2);
            String situationFamilliale = marketingRecord.get(3);
            String nbEnfantsAchange = marketingRecord.get(4);
            String deuxiemeVoiture = marketingRecord.get(5);
            // Add the marketing in the KVStore
            this.insertAMarketingRow(age, sexe, taux, situationFamilliale, nbEnfantsAchange, deuxiemeVoiture);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

FIGURE 10 – Fonction permettant l'importation du fichier sur OracleNoSQL.

Nous procédons de la même manière avec la boucle While. Elle itère sur le fichier *.csv*, en extrayant les données brutes, puis appelle la fonction `insertAmarketingRow` qui permet d'ajouter une ligne à notre table `marketing` créée préalablement, voir Figure(11) et Figure(12).

```
private void insertAmarketingRow(String age, String sexe, String taux, String situationFamilliale,
//TableAPI tableAPI = store.getTableAPI();
StatementResult result = null;
String statement = null;
System.out.println("***** Dans : insertAmarketingRow *****")

try {
    TableAPI tableH = store.getTableAPI();
    // The name you give to getTable() must be identical
    // to the name that you gave the table when you created
    // the table using the CREATE TABLE DDL statement.
    Table marketingTable = tableH.getTable(tabMarketing);
    // Get a Row Instance
    Row marketingRow = marketingTable.createRow();
    // Now put all of the cells in the row.
    // This does NOT actually write the data to
    // the store.

    // Create one row
    marketingRow.put("clientMarketingID", clientID);
    marketingRow.put("age", age);
    marketingRow.put("sexe", sexe);
    marketingRow.put("taux", taux);
    marketingRow.put("situationFamilliale", situationFamilliale);
    marketingRow.put("nbEnfantsAcharge", nbEnfantsAcharge);
    marketingRow.put("deuxiemeVoiture", deuxiemeVoiture);
    // Now write the table to the store.
    // "item" is the row's primary key. If we had not set that value,
    // this operation will throw an IllegalArgumentException.
    tableH.put(marketingRow, null, null);
    clientID++;

} catch (IllegalArgumentException e) {
    System.out.println("Invalid statement:\n" + e.getMessage());
} catch (FaultException e) {
    System.out.println("Statement couldn't be executed, please retry: " + e);
}
}
```

FIGURE 11 – Fonction permettant l'ajout d'une ligne sur la table.

```
public void createMarketingTable() {
    String statement = null;
    statement = "Create table " + tabMarketing + " ("
        + "CLIENTMARKETINGID INTEGER,"
        + "AGE STRING,"
        + "SEXE STRING,"
        + "TAUX STRING,"
        + "SITUATIONFAMILIALE STRING,"
        + "NBENFANTSACHARGE STRING,"
        + "DEUXIEMEVOITURE STRING,"
        + "PRIMARY KEY (CLIENTMARKETINGID))";
    executeDDL(statement);
}
```

FIGURE 12 – Création de la table Marketing.

3.2.2 Chargement des données

Afin de charger les données il nous faut ensuite, compiler le fichier. Nous utilisons le terminal pour exécuter les commandes, voir Figure(13).

```
-- Ceci est le chemin vers notre projet sur la machine virtuelle
[oracle@bigdatalite ~]$ export MYPROJECTHOME=/home/CAO/projetMBDS/

-- Compiler le code java pour importer la table MARKETING à partir du fichier csv
[oracle@bigdatalite ~]$ javac -g -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ $MYPROJECTHOME/voiture/DataImportMarketing.java

-- Executer le code java pour importer la table MARKETING à partir du fichier csv
[oracle@bigdatalite ~]$ java -Xmx256m -Xms256m -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ voiture.DataImportMarketing
```

FIGURE 13 – Commandes pour exécuter le script Java.

3.2.3 Vérification du chargement des données

Une fois le nouveau fichier Java compilé et exécuté, il faut désormais vérifier si les données ont bien été ajoutées. Pour cela, nous nous reconnectons au KvStore, voir Figure(14). Puis, une fois connecté, nous affichons la table marketing, Figure(15).

```
-- Connection à Oracle NoSQL
[oracle@bigdatalite ~]$ java -jar $KVHOME/lib/kvstore.jar runadmin -port 5000 -host bigdatalite.localdomain

kv-> connect store -name kvstore
```

FIGURE 14 – Connection au KvStore

```
-- Vérification du contenu de la table MARKETING
kv-> get table -name MARKETING

-- Réponse :

{"CLIENTMARKETINGID":7,"AGE":"59","SEXE":"F","TAUX":"572","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"2","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":12,"AGE":"55","SEXE":"M","TAUX":"588","SITUATIONFAMILIALE":"C♦libataire","NBENFANTSACHARGE":"0","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":2,"AGE":"35","SEXE":"M","TAUX":"223","SITUATIONFAMILIALE":"C♦libataire","NBENFANTSACHARGE":"0","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":5,"AGE":"80","SEXE":"M","TAUX":"530","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"3","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":15,"AGE":"60","SEXE":"M","TAUX":"524","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"0","DEUXIEMEVOITURE":"true"}
{"CLIENTMARKETINGID":4,"AGE":"26","SEXE":"F","TAUX":"420","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"3","DEUXIEMEVOITURE":"true"}
{"CLIENTMARKETINGID":6,"AGE":"27","SEXE":"F","TAUX":"153","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"2","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":13,"AGE":"19","SEXE":"F","TAUX":"212","SITUATIONFAMILIALE":"C♦libataire","NBENFANTSACHARGE":"0","DEUXIEMEVOITURE":"false"}
{"CLIENTMARKETINGID":18,"AGE":"54","SEXE":"F","TAUX":"452","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":"3","DEUXIEMEVOITURE":"true"}
{"CLIENTMARKETINGID":19,"AGE":"35","SEXE":"M","TAUX":"589","SITUATIONFAMILIALE":"C♦libataire","NBENFANTSACHARGE":"0","DEUXIEMEVOITURE":"false"}
```

FIGURE 15 – Affichage de la table Marketing.

Le fichier a bien été chargé dans la base.

4 Le chargement de source de données via Hadoop HDFS

Nous allons maintenant charger un nouveau fichier. Nous procédons avec une nouvelle méthode. Pour le fichier *Catalogue.csv*, nous avons choisi de le charger sur Hadoop HDFS. Hadoop est un framework Java open source utilisé pour le stockage et traitement des big data. Les données sont stockées sur des serveurs standards qui sont généralement très peu coûteux. Pour ce type de projet, il est très avantageux de choisir ce type de stockage[3].

En effet, pour l'analytique et les big data, Hadoop est une solution très avantageuse. Hadoop aide ainsi à relever le défi de l'énormité des big data grâce à ses différentes qualités :

1. Résilience :
 - Les données sont stockées dans un nœud du cluster qui sont répliquées dans d'autres nœuds, ce qui garantit la tolérance aux incidents de Hadoop. Si un nœud tombe en panne, les autres serveurs du cluster disposent toujours d'une copie de sauvegarde des données.
2. Évolutivité :
 - Contrairement aux systèmes traditionnels qui ont une capacité de stockage limitée, Hadoop est évolutif car il fonctionne dans un environnement distribué. En cas de besoin, la configuration peut être facilement étendue en installant d'autres serveurs, et la capacité de stockage peut ainsi être augmentée.
3. Coût modéré :
 - Hadoop étant un framework open source n'exigeant aucune licence, les coûts de cette solution sont nettement inférieurs à ceux des bases de données relationnelles classiques.
4. Vitesse :
 - Le système de fichiers distribué, les traitements concurrents et le modèle MapReduce permettent d'exécuter les requêtes les plus complexes en quelques secondes.
5. Diversité des données :
 - Le HDFS peut stocker différents formats de données : structurées, non structurées (par exemple, des vidéos) ou semi-structurées (par exemple, des fichiers XML). Lors du stockage des données, il n'est pas nécessaire de valider celles-ci par rapport à un schéma prédéfini : les données peuvent être téléchargées sous n'importe quel format. Lors de leur récupération, les données sont analysées et utilisées en appliquant le ou les schémas requis.

4.1 Fichier Marketing

4.1.1 Chargement des données

Pour ajouter le fichier *Catalogue.csv* sur HDFS, les simples commandes, voir Figure(16), sont nécessaires.

```
export MYPROJECTHOME=/home/CAO/

-- suppression du fichier si on souhaite le remplacer
hadoop fs -rm -r /user/cao

hadoop fs -mkdir /user/cao
hadoop fs -mkdir /user/cao/projetMBDS
hadoop fs -mkdir /user/cao/projetMBDS/Catalogue

hadoop fs -put $MYPROJECTHOME/projetMBDS/Catalogue.csv /user/cao/projetMBDS/Catalogue
```

FIGURE 16 – Commande Hadoop pour importer le fichier.

4.1.2 Vérification du chargement des données

Une fois le fichier *Catalogue.csv* sur HDFS, il suffit d'utiliser la commande *-ls* pour afficher le fichier, Figure(17).

```
hadoop fs -ls /user/cao/projetMBDS/Catalogue
```

FIGURE 17 – Commande Hadoop pour voir le fichier.

5 La création de tables externes sur Hive

Nous devons charger les fichiers *Catalogue.csv*, *Marketing.csv* et *immatriculations.csv* sur Hive.

Son utilité : proposer une abstraction en dessus de MapReduce pour faciliter l'analyse de gros volumes de données. L'avantage de Hive est de définir une structure avec une variété de formats de données facilitant ainsi la possibilité de les requêter. Il est donc bien adapté à un contexte d'analyse de données. Hive propose aussi une fonction de stockage distribué et permet d'accéder à des fichiers stockés dans HDFS. Apache Hive est un datawarehouse pour Hadoop. Il ne s'agit pas d'une base de données relationnelle ni d'un datawarehouse classique. Il s'agit d'un système qui maintient des métadonnées décrivant les données stockées dans HDFS. Il utilise une base de données relationnelle appelée metastore pour assurer la persistance des métadonnées[4].

Nous devons créer des tables externes Hive qui pointent respectivement vers :

- Le KvStore pour les tables immatriculations et marketing
- HDFS pour la table catalogue

5.1 Connexion à Hive

Premièrement, il faut se connecter à Hive, voir Figure(18).

```
[oracle@bigdatalite ~]$ beeline
Beeline version 1.1.0-cdh5.4.0 by Apache Hive

-- Connexion à Hive
beeline> !connect jdbc:hive2://localhost:10000

-- identifiants: oracle/welcome1
```

FIGURE 18 – Commande pour se connecter à Hive.

5.2 Depuis le KvStore

Dans cette étape, procédons à la création de tables externes depuis le KvStore ainsi qu'au chargement pour la table marketing.

5.2.1 Table immatriculation

5.2.1.1 Création de la table externe

Une fois connecté à Hive, nous pouvons créer la table externe qui permet de pointer vers la table immatriculation qui est stockée sur le KvStore. Nous utilisons le code suivant, Figure(19).

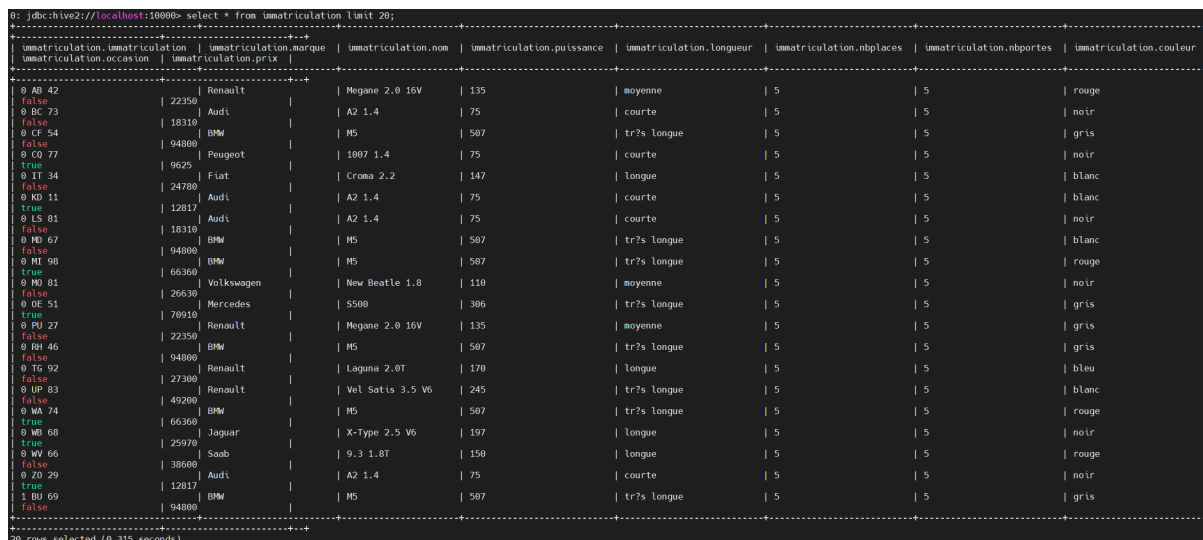
```
-- Supprimer la table IMMATRICULATION si elle existe déjà
jdbc
:hive2://localhost:10000>
drop table IMMATRICULATION;

-- Création de la table externe IMMATRICULATION pointant vers la table IMMATRICULATION de ORACLE NOSQL (kv)
jdbc
:hive2://localhost:10000> CREATE
EXTERNAL TABLE IMMATRICULATION(
    IMMATRICULATION string,
    MARQUE string,
    NOM string,
    PUISSANCE string,
    LONGUEUR string,
    NBPLACES string,
    NBPORTES string,
    COULEUR string,
    OCCASION string,
    PRIX string
)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
    "oracle.kv.kvstore" = "kvstore",
    "oracle.kv.hosts" = "bigdatalite.localdomain:5000",
    "oracle.kv.hadoop.hosts" = "bigdatalite.localdomain/127.0.0.1",
    "oracle.kv.tableName" = "IMMATRICULATION");
```

FIGURE 19 – Création de la table externe immatriculation.

5.2.1.2 Vérification de la création de la table externe

Une fois la table créée, nous vérifions si la table a bien été créé et stocké par la même occasion, voir Figure(20).



```
0: jdbc:hive2://localhost:10000> select * from immatriculation limit 20;
```

immatriculation.immatriculation	immatriculation.marque	immatriculation.nom	immatriculation.puissance	immatriculation.longueur	immatriculation.nbplaces	immatriculation.nbportes	immatriculation.couleur
0 AB 42	Renault	Megane 2.0 16V	135	moyenne	5	5	rouge
0 BC 73	Audi	A2 1.4	75	courte	5	5	noir
0 CF 54	BMW	M5	507	tr?s longue	5	5	gris
0 CO 77	Peugeot	1007 1.4	75	courte	5	5	noir
0 IT 34	Fiat	Croma 2.2	147	longue	5	5	blanc
0 KD 11	Audi	A2 1.4	75	courte	5	5	blanc
0 LS 81	Audi	A2 1.4	75	courte	5	5	noir
0 MD 67	BMW	M5	507	tr?s longue	5	5	blanc
0 MI 98	BMW	M5	507	tr?s longue	5	5	rouge
0 MO 81	Volkswagen	New Beetle 1.8	110	moyenne	5	5	noir
0 OE 51	Mercedes	S500	306	tr?s longue	5	5	gris
0 PU 27	Renault	Megane 2.0 16V	135	moyenne	5	5	gris
0 RH 46	BMW	M5	507	tr?s longue	5	5	gris
0 TG 92	Renault	Laguna 2.0T	170	longue	5	5	bleu
0 UP 83	Renault	Vel Satis 3.5 V6	245	tr?s longue	5	5	blanc
0 WA 74	BMW	M5	507	tr?s longue	5	5	rouge
0 XB 68	Jaguar	X-Type 2.5 V6	197	longue	5	5	noir
0 YV 66	Saab	9.3 1.8T	150	longue	5	5	rouge
0 ZD 29	Audi	A2 1.4	75	courte	5	5	noir
1 BU 69	BMW	M5	507	tr?s longue	5	5	gris

20 rows selected (0.315 seconds)

FIGURE 20 – Vérification de la création de la table externe immatriculation.

On remarque que la table est bien présente.

5.2.2 Table marketing

5.2.2.1 Création de la table externe

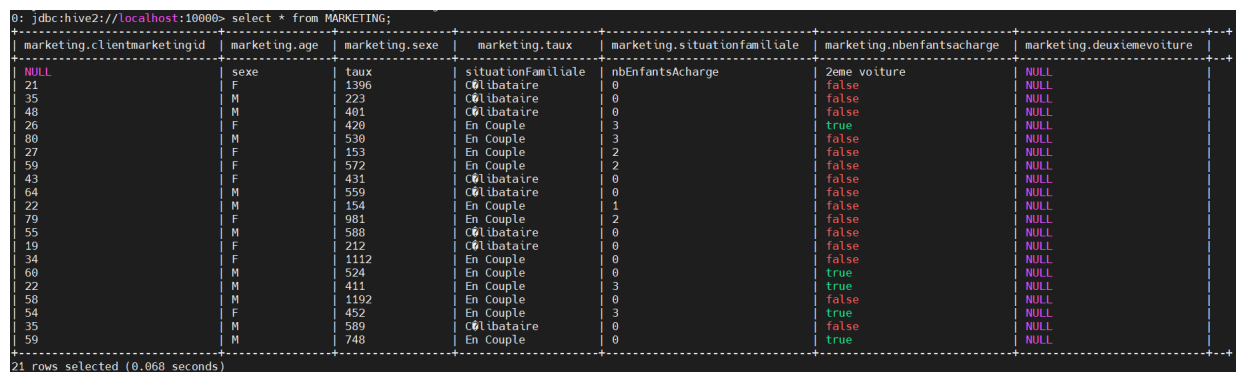
De la même manière, nous pouvons créer la table externe qui permet de pointer vers la table marketing qui est stockée sur le KvStore. Nous utilisons le code suivant, Figure(21).

```
-- Création de la table externe MARKETING pointant vers la table MARKETING de ORACLE NOSQL (kv)
CREATE EXTERNAL TABLE MARKETING(
  CLIENTMARKETINGID int,
  AGE string,
  SEXE string,
  TAUX string,
  SITUATIONFAMILIALE string,
  NBENFANTSACHARGE string,
  DEUXIEMEVOITURE string
)
STORED BY 'oracle.kv.hadoop.hive.table.TableStorageHandler'
TBLPROPERTIES (
  "oracle.kv.kvstore" = "kvstore",
  "oracle.kv.hosts" = "bigdatalite.localdomain:5000",
  "oracle.kv.hadoop.hosts" = "bigdatalite.localdomain/127.0.0.1",
  "oracle.kv.tableName" = "MARKETING");
```

FIGURE 21 – Création de la table externe marketing.

5.2.2.2 Vérification de la création de la table externe

Une fois la table créée, nous vérifions si la table a bien été créé et stocké par la même occasion, voir Figure(22).



marketing.clientmarketingid	marketing.age	marketing.sexe	marketing.taux	marketing.situationfamiliale	marketing.nbenfantsacharge	marketing.deuxiemevoiture
NULL						
21	1396	F	0	0	0	NULL
35	223	M	0	0	0	NULL
48	401	M	0	0	0	NULL
26	420	F	3	3	3	NULL
80	530	M	3	3	3	NULL
27	153	F	2	2	2	NULL
59	572	F	2	2	2	NULL
43	431	F	0	0	0	NULL
64	559	M	0	0	0	NULL
22	154	M	1	1	1	NULL
79	981	F	2	2	2	NULL
55	588	M	0	0	0	NULL
19	212	F	0	0	0	NULL
34	1112	F	0	0	0	NULL
60	524	M	0	0	0	NULL
22	411	M	3	3	3	NULL
58	1192	M	0	0	0	NULL
54	452	F	3	3	3	NULL
35	589	M	0	0	0	NULL
59	748	M	0	0	0	NULL

FIGURE 22 – Vérification de la création de la table externe marketing.

On remarque que la table est bien présente.

5.3 Depuis HDFS

Maintenant lors de cette étape, nous allons créer la table externe depuis le HDFS ainsi que le chargement pour la table catalogue.

5.3.1 Table catalogue

5.3.1.1 Création de la table externe

Nous nous reconnectons à Hive, nous pouvons créer la table externe qui permet de pointer vers la table catalogue qui est stockée sur HDFS. Nous utilisons le code suivant, Figure(23).

```
drop table CATALOGUE;

-- marque,nom,puissance,longueur,nbPlaces,nbPortes,couleur,occasion,prix
CREATE EXTERNAL TABLE CATALOGUE (
    MARQUE string,
    NOM string ,
    PUISSANCE string,
    LONGUEUR string,
    NBPLACES string,
    NBPORTES string,
    COULEUR string,
    OCCASION string,
    PRIX string
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 'hdfs://user/cao/projetMBDS/Catalogue';
```

FIGURE 23 – Création de la table externe catalogue.

5.3.1.2 Vérification de la création de la table externe

Une fois la table créée, nous vérifions si la table a bien été créé et stocké par la même occasion, voir Figure(24).

```
0: jdbc:hive2://localhost:10000> select * from CATALOGUE LIMIT 20;
```

catalogue.marque	catalogue.nom	catalogue.puissance	catalogue.longueur	catalogue.nbplaces	catalogue.nbportes	catalogue.couleur	catalogue.occasion	catalogue.prix
marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix
Volvo	S80 T6	272	très longue	5	5	blanc	false	50500
Volvo	S80 T6	272	très longue	5	5	noir	false	50500
Volvo	S80 T6	272	très longue	5	5	rouge	false	50500
Volvo	S80 T6	272	très longue	5	5	gris	true	35350
Volvo	S80 T6	272	très longue	5	5	bleu	true	35350
Volvo	S80 T6	272	très longue	5	5	gris	false	50500
Volvo	S80 T6	272	très longue	5	5	bleu	false	50500
Volvo	S80 T6	272	très longue	5	5	rouge	true	35350
Volvo	S80 T6	272	très longue	5	5	blanc	true	35350
Volvo	S80 T6	272	très longue	5	5	noir	true	35350
Volkswagen	Touran 2.0 FSI	150	longue	7	5	rouge	false	27340
Volkswagen	Touran 2.0 FSI	150	longue	7	5	gris	true	19138
Volkswagen	Touran 2.0 FSI	150	longue	7	5	bleu	true	19138
Volkswagen	Touran 2.0 FSI	150	longue	7	5	gris	false	27340
Volkswagen	Touran 2.0 FSI	150	longue	7	5	bleu	false	27340
Volkswagen	Touran 2.0 FSI	150	longue	7	5	blanc	true	19138
Volkswagen	Touran 2.0 FSI	150	longue	7	5	noir	true	19138
Volkswagen	Touran 2.0 FSI	150	longue	7	5	rouge	true	19138
Volkswagen	Touran 2.0 FSI	150	longue	7	5	blanc	false	27340

20 rows selected (0.068 seconds)

FIGURE 24 – Vérification de la création de la table externe catalogue.

On remarque que la table est bien présente.

6 Le chargement de sources de données sur la base Oracle SQL

Nous allons maintenant utiliser une nouvelle méthode. Nous allons utiliser la base Oracle SQL pour le dernier fichier.

Oracle Database est un système de gestion de bases de données relationnelles (SGBDR) proposé par Oracle[5]. Aujourd'hui, il est particulièrement performant lorsque l'on a un vrai besoin de performance sur le traitement de gros volumes de données, ce qui est notre cas avec ce projet. Oracle possède de nombreux avantages :

- Possibilité de choisir entre une installation automatique ou paramétrer son installation à 100%.
- Gestion entièrement automatique de la mémoire.
- Gestion avancée de la compression des données.
- Très performant sur des gros volumes de données.
- Vues matérialisées.

Nous allons donc charger le fichier client sur Oracle SQL en utilisant SQL Loader.

Ce fichier est *Clients_6.csv*.

6.1 Création de la table client

Afin d'utiliser SQL Loader, pour ajouter les données, il faut au préalable créer une table dans la database SQL Plus, comme sur la Figure(25). Il faut tout d'abord se connecter sur la base.

```
sqlplus CAOBZ2021@ORCL/CAOBZ202101

DROP TABLE CLIENT;

CREATE TABLE CLIENT(
  AGE varchar2(30),
  SEXE varchar2(30),
  TAUX varchar2(30),
  SITUATIONFAMILIALE varchar2(30),
  NBENFANTSACHARGE varchar2(30),
  XVOITURE varchar2(30),
  IMMATRICULATION varchar2(30)
);
```

FIGURE 25 – Création de la table client.

6.2 Population de la table client

Une fois la table créée, nous pouvons importer les données dedans, depuis le fichier *.csv*. Un fichier de contrôle, voir Figure(26) est nécessaire afin de populer correctement la table en fonction du *.csv* qu'on lui transmet en entrée.

Il suffit ensuite de lancer le fichier de contrôle *.ctl*, Figure(27), dans le terminal de commandes.

```
LOAD DATA
INFILE '$MYPROJECTHOME/projetMBDS/SQLLOADER/Clients_6.csv'
INSERT INTO TABLE client
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
AGE,
SEXE,
TAUX,
SITUATIONFAMILIALE,
NBNFANTSACHARGE,
XVOITURE,
IMMATRICULATION
)
```

FIGURE 26 – Fichier de contrôle pour populer la table client.

```
--Table : client
sqlldr CAOBZ2021@ORCL/CAOBZ202101 control=$MYPROJECTHOME/projetMBDS/SQLLOADER/control_clients.ctl
log=$MYPROJECTHOME/projetMBDS/SQLLOADER/track_clients.log skip=1
```

FIGURE 27 – Commande pour populer la table avec le fichier de commande.

6.3 Vérification de la création de la table client

Enfin, nous vérifions que les données du fichier ont bien été inséré dans la table client, voir Figure(28) et Figure(29).

```
SQL> select * from client fetch first 3 rows only;
```

AGE	SEXE	TAUX	SITUATIONFAMILIALE	NBNFANTSACHARGE	XVOITURE	IMMATRICULATION
25	F					
159	En Couple					
2	false					
3467	SB	72				

AGE	SEXE	TAUX	SITUATIONFAMILIALE	NBNFANTSACHARGE	XVOITURE	IMMATRICULATION
53	M					
594	En Couple					
2	false					
113	LY	42				

AGE	SEXE	TAUX	SITUATIONFAMILIALE	NBNFANTSACHARGE	XVOITURE	IMMATRICULATION
20	F					
949	En Couple					
1	false					
925	WK	87				

FIGURE 28 – Affichage des premières lignes de la table client.

```
SQL> select count(*) from CLIENT;
```

COUNT(*)
2000000

FIGURE 29 – Nombre de ligne dans la table client.

On remarque bien la présence de 2 millions de lignes. La première ligne a bien été sauté pour enlever les titres des colonnes et ne garder que les données.

7 Création et récupération des tables externes dans SQLPlus

Finalement, nous pouvons insérer les 3 tables créées dans HIVE (immatriculation, marketing, catalogue) dans SQL Plus en créant des tables externes.

7.1 Connexion à la base

Comme pour la partie précédente, il faut se connecter à la base pour pouvoir créer des tables, voir Figure(30).

```
sqlplus CAOBZ2021@ORCL/CAOBZ202101;
```

FIGURE 30 – Connexion à la base Oracle.

7.2 Table immatriculation

7.2.1 Création de la table

De la même manière que la table interne, nous créer la table externe immatriculation. Nous utilisons le code suivant, Figure(31).

```
drop table IMMATRICULATION;

CREATE TABLE IMMATRICULATION
(
  IMMATRICULATION varchar2(30),
  MARQUE           varchar2(30),
  NOM              varchar2(30),
  PUISSANCE        varchar2(30),
  LONGUEUR         varchar2(30),
  NBPLACES         varchar2(30),
  NBPORTES         varchar2(30),
  COULEUR          varchar2(30),
  OCCASION         varchar2(30),
  PRIX             varchar2(30)
) ORGANIZATION EXTERNAL (
  TYPE ORACLE_HIVE
  DEFAULT DIRECTORY ORACLE_BIGDATA_CONFIG ACCESS PARAMETERS (
    com.oracle.bigdata.tablename=default.IMMATRICULATION
  )
)
REJECT LIMIT UNLIMITED;
```

FIGURE 31 – Création de la table immatriculation sur la base Oracle.

7.2.2 Vérification de la création de la table

Une fois la table créée, nous vérifions si la table a bien été créée et populé, voir Figure(32).

```
SQL> SELECT * FROM IMMATRICULATION FETCH FIRST 2 ROWS ONLY;
```

IMMATRICULATION	MARQUE
NOM	PUISSANCE
LONGUEUR	NBPLACES
NBPORTES	COULEUR
OCCASION	PRIX
0 AS 74	BMW
120i	150
moyenne	5

IMMATRICULATION	MARQUE
NOM	PUISSANCE
LONGUEUR	NBPLACES
NBPORTES	COULEUR
OCCASION	PRIX
5	bleu
true	25060

IMMATRICULATION	MARQUE
NOM	PUISSANCE
LONGUEUR	NBPLACES
NBPORTES	COULEUR
OCCASION	PRIX
0 AX 73	Ford
Mondeo 1.8	125
longue	5

IMMATRICULATION	MARQUE
NOM	PUISSANCE
LONGUEUR	NBPLACES
NBPORTES	COULEUR
OCCASION	PRIX
5	rouge
false	23900

FIGURE 32 – Vérification de la création de la table immatriculation sur la base Oracle.

On remarque que les données sont bien présentes sur la table.

7.3 Table marketing

7.3.1 Création de la table

De la même manière que la table immatriculation, nous créer la table externe marketing. Nous utilisons le code suivant, Figure(33).

```
drop table MARKETING;

CREATE TABLE MARKETING
(
  CLIENTMARKETINGID INTEGER,
  AGE                varchar2(30),
  SEXE               varchar2(30),
  TAUX               varchar2(30),
  SITUATIONFAMILIALE varchar2(30),
  NBENFANTSACHARGE  varchar2(30),
  DEUXIEMEVOITURE    varchar2(30)
) ORGANIZATION EXTERNAL (
  TYPE ORACLE_HIVE
  DEFAULT DIRECTORY ORACLE_BIGDATA_CONFIG ACCESS PARAMETERS (
    com.oracle.bigdata.tablename=default.MARKETING
  )
)
REJECT LIMIT UNLIMITED;
```

FIGURE 33 – Création de la table marketing sur la base Oracle.

7.3.2 Vérification de la création de la table

Une fois la table créée, nous vérifions si la table a bien été créé et populé, voir Figure(34).

```
SQL> SELECT * FROM MARKETING FETCH FIRST 2 ROWS ONLY;
```

CLIENTMARKETINGID	AGE	SEXE
572	7 59	F
2		En Couple false
981	11 79	F
2		En Couple false

```
CLIENTMARKETINGID AGE SEXE
-----
TAUX SITUATIONFAMILIALE
-----
NBENFANTSACHARGE DEUXIEMEVOITURE
-----
```

FIGURE 34 – Vérification de la création de la table marketing sur la base Oracle.

Nous remarquons bien que les données sont bien présentes sur la table.

7.4 Table catalogue

7.4.1 Création de la table

Enfin, de la même manière que les deux tables précédentes, nous créer la table externe catalogue. Nous utilisons le code suivant, Figure(35).

```
drop table CATALOGUE;  
  
CREATE TABLE CATALOGUE  
(  
  MARQUE    varchar2(30),  
  NOM       varchar2(30),  
  PUISSANCE varchar2(30),  
  LONGUEUR  varchar2(30),  
  NBPLACES  varchar2(30),  
  NBPORTES  varchar2(30),  
  COULEUR   varchar2(30),  
  OCCASION  varchar2(30),  
  PRIX      varchar2(30)  
) ORGANIZATION EXTERNAL (  
  TYPE ORACLE_HIVE  
  DEFAULT DIRECTORY ORACLE_BIGDATA_CONFIG ACCESS PARAMETERS (  
    com.oracle.bigdata.tablename=default.CATALOGUE  
  )  
)  
REJECT LIMIT UNLIMITED;
```

FIGURE 35 – Création de la table catalogue sur la base Oracle.

7.4.2 Vérification de la création de la table

Une fois la table créée, nous vérifions si la table a bien été créé et populé, voir Figure(36).

```
SQL> select * from catalogue offset 2 rows fetch next 3 rows only;
```

MARQUE	NOM
PUISSANCE	LONGUEUR
NBPLACES	NBPORTES
COULEUR	OCCASION
PRIX	
Volvo	S80 T6
272	tr?longue
5	5
MARQUE	NOM
PUISSANCE	LONGUEUR
NBPLACES	NBPORTES
COULEUR	OCCASION
PRIX	
noir	false
50500	
MARQUE	NOM
PUISSANCE	LONGUEUR
NBPLACES	NBPORTES
COULEUR	OCCASION
PRIX	
Volvo	S80 T6
272	tr?longue
5	5

FIGURE 36 – Vérification de la création de la table catalogue sur la base Oracle.

On remarque que les données sont bien présentes sur la table.

Nos 4 tables sont désormais toutes stockées sur la base Oracle. Nous pouvons passer à la récupération des données. Pour ce faire nous utiliserons le logiciel R Studio.

8 Récupération et affichage des données dans R Studio

Enfin, la dernière étape de ce projet et de charger les données présente sur Oracle SQL dans notre machine virtuelle puis sur R Studio.

R Studio est un environnement de développement gratuit, libre et multiplateforme pour le langage R, un langage de programmation utilisé pour le traitement de données et l'analyse statistique. Les commandes sont exécutées grâce à des instructions codées dans un langage relativement simple, les résultats sont affichés sous forme de texte et les graphiques sont visualisés directement[6].

Le logiciel R est particulièrement performant pour la manipulation de données, le calcul et l'affichage de graphiques. Il possède, entre autres choses :

- Un système de documentation intégré très bien conçu
- Des procédures efficaces de traitement des données et des capacités de stockage de ces données
- Une suite d'opérateurs pour des calculs sur des tableaux et en particulier sur des matrices
- Une vaste et cohérente collection de procédures statistiques pour l'analyse de données
- Des capacités graphiques évoluées
- Un langage de programmation simple et efficace intégrant les conditions, les boucles, la récursivité, et des possibilités d'entrée-sortie

Afin de récupérer les données, nous pouvons procéder de deux manières différentes, soit en utilisant les données stockées sur Oracle, ou en important manuellement les données sur R. Nous allons utiliser la première méthode. La seconde méthode, moins optimisée, sera expliquée dans la seconde partie de ce projet.

8.1 Création d'un utilisateur sur la base

Il nous est nécessaire de créer un user afin de pouvoir assurer la connexion à la base de données distante. Pour cela nous réalisons le script suivant, Figure(37).

```
(CAO@bigdatalite ~)$ sqlplus /nolog

SQL*Plus: Release 12.1.0.2.0 Production on Tue Mar 23 11:02:54 2021

Copyright (c) 1982, 2014, Oracle. All rights reserved.

SQL> define MYDBUSER=PROJET
SQL> define MYDB=orcl
SQL> define MYDBUSERPASS=123
SQL> define MYCDBUSER=system
SQL> connect &MYCDBUSER@&MYDB/
Enter password:welcome1

Connected.

SQL> CREATE USER &MYDBUSER IDENTIFIED BY &MYDBUSERPASS default tablespace users temporary tablespace temp;
old 1: CREATE USER &MYDBUSER IDENTIFIED BY &MYDBUSERPASS default tablespace users temporary tablespace temp
new 1: CREATE USER PROJET IDENTIFIED BY 123 default tablespace users temporary tablespace temp

User created.

SQL> grant dba to &MYDBUSER;
old 1: grant dba to &MYDBUSER
new 1: grant dba to PROJET

grant succeeded.

SQL> alter user &MYDBUSER quota unlimited on users;
old 1: alter user &MYDBUSER quota unlimited on users
new 1: alter user PROJET quota unlimited on users

User altered.
```

FIGURE 37 – Création d'un user pour manipuler les données.

8.2 Utilisation de R Studio

8.2.1 Connexion sur R Studio

Une fois notre user projet crée, nous ajoutons les informations correspondant à la base de données distante dans notre fichier tnsnames.ora. Ensuite, nous créons une connexion ODBC, via le panneau de configurations, à notre base de données distante via le user créé. Pour utiliser ce type de connexion, nous décidons de travailler avec le package R « RODB », Figure(38). Ce package nous permettra d'assurer un pont entre nos deux logiciels. Puis, nous créons la connexion avec la commande suivante, Figure(39).

```
install.packages("RODBC")  
  
library(RODBC)
```

FIGURE 38 – Installation du package pour l'import Oracle.

```
##----connexion à la machine virtuelle-----  
connexion <- odbcConnect("ORCLPROJETDB_DNS", uid="PROJET", pwd="123", believeRows=FALSE)
```

FIGURE 39 – Création d'une connexion ODBC.

8.2.2 Création des data-frames sur R Studio

L'étape suivante est la création des data-frames via la commande « sqlQuery », Figure(40).

```
##----création des dataframes-----  
  
immatriculation<-sqlQuery(connexion, "SELECT * FROM IMMATRICULATION")  
marketing<-sqlQuery(connexion, "SELECT * FROM MARKETING")  
catalogue<-sqlQuery(connexion, "SELECT * FROM CATALOGUE")  
clients<-sqlQuery(connexion, "SELECT * FROM CLIENTS")
```

FIGURE 40 – Création des data-frames

La partie Gestion des données est donc terminée, nous pouvons désormais passer à la partie Analyse des Données. Dans cette partie, nous allons utiliser les données des mêmes fichiers .csv. Cependant, nous importerons les données manuellement, expliqué dans la section 10.1. Puis, les données seront nettoyées dans le logiciel R Studio. Cependant, cette étape peut aussi être aussi réalisé sur SQL Plus en amont. Nous expliquons comment réaliser cette étape, en *Annexe*(65). De plus, l'importation des données sur R Studio via un driver y est aussi expliquée.

Enfin, nous monterons à la fin que l'importation manuelle ou via les bases de données (*Partie Gestion des Données*), n'influe pas sur le résultat final, en *Annexe*(66).

PARTIE Analyse de Données

9 Objectif et contexte de l'Analyse de données

Après avoir récupéré l'ensemble des données, il faut passer à la partie analyse de données. Cette partie consiste à utiliser les données connues sur des clients, afin de réaliser une prédiction sur des potentiels clients.

De ce fait, plusieurs étapes vont être cruciales pour la création de ce modèle.

1. Le chargement et le nettoyage des données.
 - Notre objectif dans cette première partie est de supprimer toutes données pouvant parasiter la construction de notre modèle
2. La construction des ensembles de tests et d'apprentissages.
 - Création de la variable "catégories"
 - Fusion des différents fichiers
 - Suppression de variables
3. Le test des classifieurs et la création de notre modèle de prédiction.
 - Réalisation de tests de classifieurs sur nos ensembles de tests et d'apprentissages
 - Sélection du meilleur classifieur
4. Application du classifieur choisi sur le fichier Marketing afin de créer nos prédictions sur les véhicules des clients.
 - Prédire pour chacun des clients, la catégorie de véhicules qui lui correspond le mieux en utilisant le classifieur généré

10 Chargement et Nettoyage des données sur R

Tout d'abord, le chargement des données sur R peut se faire par deux méthodes :

- Par un driver HDFS
- par un driver Oracle

10.1 Import des fichiers .csv

Afin de réaliser l'import des données et l'établissement de table Immatriculation, Client, Marketing et Catalogue, nous allons utiliser la fonction `read.csv` comme sur la Figure(41).

```
##----import des divers fichiers csv-----  
  
immatriculation <- read.csv("Immatriculations.csv", header = TRUE, sep = ",", dec = ".")  
catalogue <- read.csv("Catalogue.csv", header = TRUE, sep = ",", dec = ".")  
marketing <- read.csv("Marketing.csv", header = TRUE, sep = ",", dec = ".")  
clients <- read.csv("Clients_6.csv", header = TRUE, sep = ",", dec = ".")
```

FIGURE 41 – Importation des fichiers .csv.

Une fois cette commande exécutée, nous pouvons voir que l'import s'est bien déroulé avec la Figure(42).





Data		
catalogue	270 obs. of 9 variables	
clients	100000 obs. of 7 variables	
immatriculation	2000000 obs. of 10 variables	
marketing	20 obs. of 6 variables	

FIGURE 42 – Importation réussite des fichiers .csv sur R.

Maintenant que les données sont bien importées nous allons pouvoir débiter l'analyse exploratoire des données.

A travers cette analyse, nous allons nous concentrer sur les données de la table Client et Immatriculation, car ces deux tables sont celles qui comportent les différentes données erronées pouvant parasiter notre étude.

10.2 Analyse exploratoire des données sur le fichier Immatriculations

Tout d'abord, une étude sur les statistiques élémentaires est nécessaire. Nous utilisons donc la fonction « `summary` ». Cela nous permet de voir s'il y a des valeurs outlier présente dans ce data-frame. La Figure(43) nous montre ce data-frame.

```
summary(immatriculation)
```

<u>##immatriculation</u>	<u>marque</u>	<u>nom</u>	<u>puissance</u>	<u>longueur</u>	<u>nbPlaces</u>	<u>n</u>
##Length:2000000	Length:2000000	Length:2000000	Min. : 55	Length:2000000	Min. :5	Min.
##Class :character	Class :character	Class :character	1st Qu.: 75	Class :character	1st Qu.:5	1st
#Mode :character	Mode :character	Mode :character	Median :150	Mode :character	Median :5	Media
			#Mean :199		Mean :5	Mean
			#3rd Qu.:245		3rd Qu.:5	3rd
			#Max. :507		Max. :5	Max.

<u>#couleur</u>	<u>occasion</u>	<u>prix</u>
#Length:2000000	Length:2000000	Min. : 7500
#Class :character	Class :character	1st Qu.: 18310
#Mode :character	Mode :character	Median : 25970
		#Mean : 35783
		#3rd Qu.: 49200
		#Max. :101300

FIGURE 43 – Récapitulatif du data-frame Immatriculation sur R.

Pour ce qui est de la table immatriculation, on ne remarque aucune valeur de telles sortes. Cependant, nous pouvons voir que dans la table immatriculation, nous n'avons aucune voiture de 7 places.

Afin de continuer notre analyse nous allons rendre tout cela plus visuel en utilisant des histogrammes, boîtes à moustaches et nuage de points. Il est nécessaire d'importer la librairie « ggplot2 » pour réaliser ces différents graphiques.

Nous allons donc commencer par afficher un histogramme pour chacune des variables de la table immatriculation.

10.2.1 Histogramme de la variable longueur

En utilisant la fonction « qplot » de la librairie ggplot2 sur la variable longueur de la table Immatriculation, nous obtenons donc, voir Figure(44), la répartition du nombre de véhicules dans les différentes longueurs possibles pour une voiture.

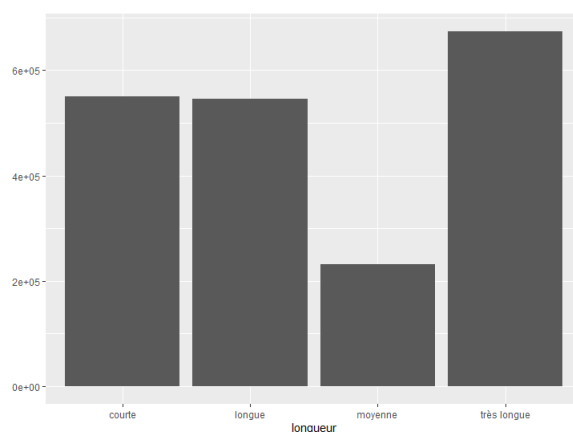


FIGURE 44 – Répartition de la variable longueur.

Tout d'abord, nous remarquons que nous n'avons aucune valeur erronée dans cette variable. Cependant, nous avons une répartition à peu près équivalente entre les voitures longues et courtes. Nous avons peu de voitures de longueur moyenne, mais nous avons énormément de voiture très longue (d'un facteur fois 3 par rapport aux voitures de longueur moyenne).

10.2.2 Histogramme de la variable nombre de portes

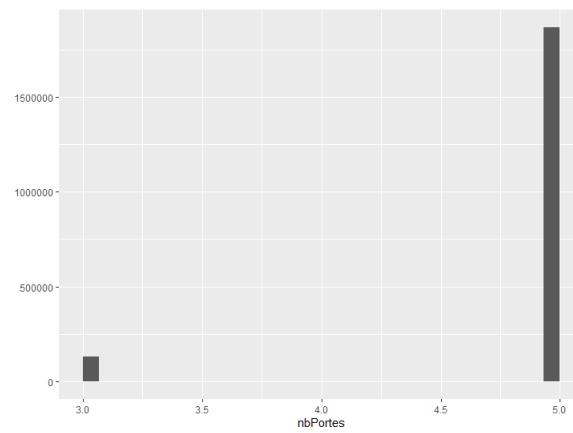
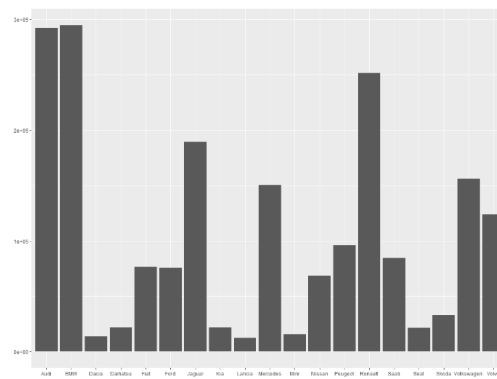


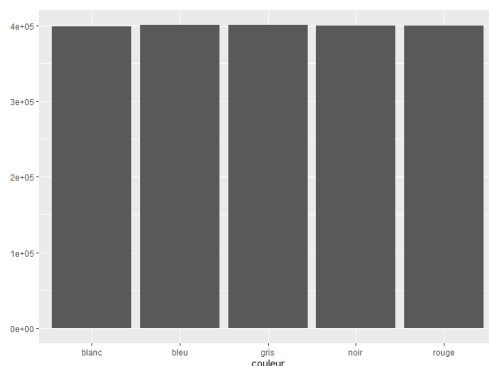
FIGURE 45 – Répartition de la variable nbPortes.

A travers cet histogramme, Figure(45), nous remarquons l'absence de valeurs erronées pour cette variable. Mais aussi qu'une majorité sont des voitures 5 portes. Ainsi, lors de l'établissement des règles pour la construction de catégorie de voiture, nous avons conclu que cette donnée n'est pas nécessaire.

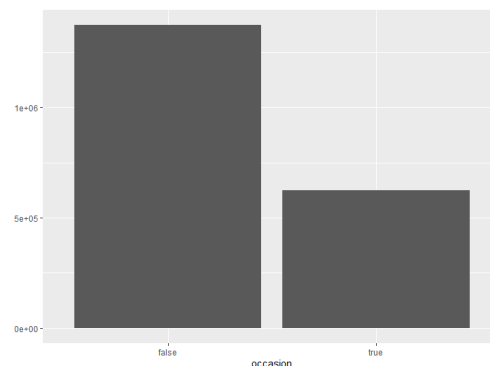
10.2.3 Histogrammes des variables marque, couleur et occasion



(a) Marque



(b) Couleur



(c) Occasion

FIGURE 46 – Répartition des différentes variables.

Au vu de ces trois histogrammes, Figure(46), aucune de ces trois variables ne contient de valeurs erronées.

10.2.4 Opération post-analyse

Un des problèmes que nous pourrions rencontrer avec une grande quantité telle que la table `immatriculation` nous propose est la répétition de données. Pour cela, nous utilisons le code suivant, Figure(47), afin de supprimer tous les doublons.

```
doublons <- which(duplicated(immatriculation$immatriculation))  
immatriculation<-immatriculation[-doublons,]
```

FIGURE 47 – Suppression des données en double.

Par le biais de cette opération, nous supprimons 3368 lignes qui étaient en double.

10.2.5 Conclusion de l’analyse du fichier `Immatriculations`

Afin de conclure cette analyse de la table `immatriculation`, nous pouvons dire que cette table ne comportait pas de valeurs outlier et erronées. Seulement, elle comportait un nombre quand même assez important de données en doublons qui pouvaient parasiter notre analyse.

10.3 Analyse exploratoire des données sur le fichier Clients

Nous remarquons que la table Clients contient uniquement des données sous forme « character ». Il faut donc qu'en amont nous transformions ces données au bon format, voir Figure(48), afin de mener à bien notre analyse exploratoire.

```
#Import fichier clients
clients <- read.csv("Clients_6.csv", header = TRUE, sep = ",", dec = ".")
clients$age <- as.integer(clients$age)
clients$taux <- as.integer(clients$taux)
clients$situationFamiliiale <- as.factor(clients$situationFamiliiale)
clients$nbEnfantsAcharge <- as.integer(clients$nbEnfantsAcharge)
clients$X2eme.voiture <- as.logical(clients$X2eme.voiture)
clients$sexe <- as.factor(clients$sexe)
```

FIGURE 48 – Modification du data-frame Clients.

Cependant lors de l'exécution de ce code, le logiciel nous informe l'ajout de valeurs NA dans la table. Ainsi pour supprimer les lignes contenant des données avec des NA nous exécutons le code suivant, Figure(49).

```
clients <- na.omit(clients)
```

FIGURE 49 – Suppression des datas null du data-frame Clients.

Maintenant que ces valeurs sont enlevées, nous affichons un résumé de cette table avec la fonction summary, voir Figure(50).

```
summary (clients)
```

##age	sexe	taux	situationFamiliiale	nbEnfantsAcharge	X2eme.voiture	immatriculation
###Min. : -1.00	M	:67756	Min. : -1.0	En Couple :63346	Min. : -1.000	Mode :logical Length:99194
###1st Qu.: 28.00	F	:29131	1st Qu.: 420.0	Célibataire:29589	1st Qu.: 0.000	FALSE:86319 Class :character
###Median :42.00	Homme	: 735	Median : 520.0	Seule : 4960	Median : 1.000	TRUE :12875 Mode :character
###Mean :43.73	Masculin	: 679	Mean : 607.8	Marlé(e) : 652	Mean : 1.245	
###3rd Qu.:57.00	Féminin	: 318	3rd Qu.: 827.0	Seul : 280	3rd Qu.: 2.000	
###Max. :84.00	Femme	: 287	Max. :1399.0	N/D : 111	Max. : 4.000	
##	(other)	: 288	(other)	: 256		

FIGURE 50 – Récapitulatif du data-frame Clients sur R.

Grace à ce résumé, nous pouvons identifier les variables qui possèdent des valeurs erronées. Ce sont donc les variables suivantes :

- Age
- Sexe
- Taux
- SituationFamiliiale
- nbEnfantsAcharge

Nous allons donc réaliser des opérations sur ces variables, afin de faire disparaître les erreurs.

10.3.1 Opérations sur la variable Age

Tout d'abord, nous réalisons un histogramme de cette variable, Figure(51).

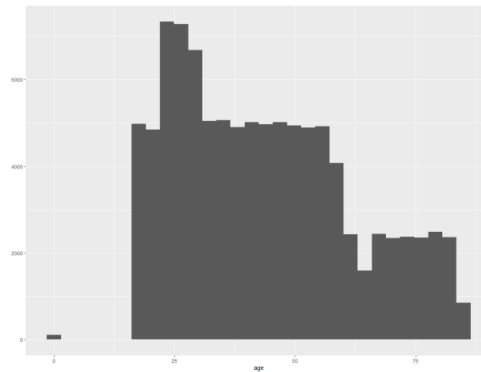


FIGURE 51 – Répartition de la variable Age.

On remarque donc qu'il y a des valeurs outlier pour cette variable. En effet, cet histogramme nous indique que l'âge de certains clients est de 0. Or, pour cette étude, l'âge minimum des clients qui ont acheté des véhicules au cours de l'année est de 18 ans. Nous allons donc utiliser la fonction suivante, Figure(52), pour retirer les valeurs non-autorisées.

```
clients <- subset(clients, clients$age >= 17)
```

FIGURE 52 – Formule de suppression des outlier datas sur l'age.

Ainsi, nous supprimons toutes les lignes où l'âge de l'acheteur est strictement inférieur à 17 ans. Nous n'avons plus de valeurs erronées dans cette variable après cette commande.

10.3.2 Opérations sur la variable Sexe

Tout comme la variable âge, utilisons ici un histogramme afin de rendre l'analyse des erreurs plus visuelle, Figure(53).

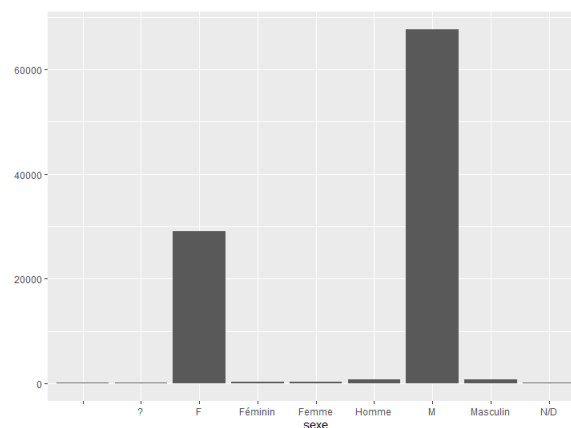


FIGURE 53 – Répartition de la variable Sexe.

Nous remarquons plusieurs erreurs de format et de valeurs. Nous allons procéder par étapes. Tout d'abord, nous supprimons les données contenant les valeurs « », « ? » et « N/D ». Nous allons donc utiliser la fonction suivante, voir Figure(54).

```
clients <- subset(clients, clients$sexe!="?" & clients$sexe!=" " & clients$sexe!="N/D")
```

FIGURE 54 – Formule de suppression des outlier datas sur le sexe.

De plus, le format requis pour cette donnée pour le sexe masculin est M et pour le sexe féminin est F. On va donc utiliser le code suivant, voir Figure(55).

```
clients$sexe <- str_replace(clients$sexe, "Homme", "M")
clients$sexe <- str_replace(clients$sexe, "Masculin", "M")
clients$sexe <- str_replace(clients$sexe, "Féminin", "F")
clients$sexe <- str_replace(clients$sexe, "Femme", "F")
clients$sexe <- as.factor(clients$sexe)
clients$sexe <- droplevels(clients$sexe)
```

FIGURE 55 – Modification des datas sur le sexe.

Tout d'abord, « str_replace » nous permet de mettre les différentes valeurs au bon format et enfin « droplevels » nous permet d'éliminer les modalités des facteurs qui n'existent plus.

Enfin, nous allons vérifions le résultat de nos manipulations avec un nouvel histogramme, Figure(56).

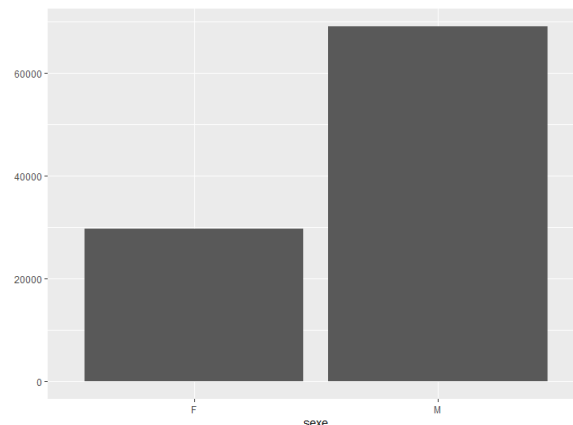


FIGURE 56 – Répartition de la variable Sexe après les opérations.

Ainsi, on remarque bien la disparition des valeurs non-conformes, mais aussi nous n'avons plus que les deux valeurs autorisées « M » et « F ».

10.3.3 Opérations sur la variable Taux

Nous allons établir l'histogramme de la variable taux, Figure(57).

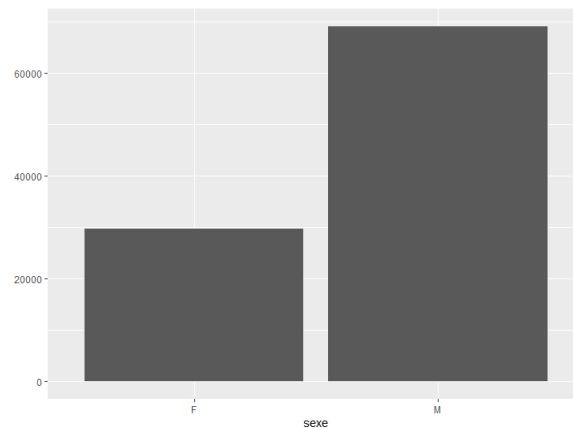


FIGURE 57 – Répartition de la variable Taux.

Nous remarquons qu'un grand nombre de valeurs de taux est inférieur à 544, la valeur minimum autorisée pour un taux. Nous supprimons donc toutes lignes de données avec un taux inférieur à 544 grâce à la commande suivante, voir Figure(58).

```
clients <- subset(clients, clients$taux >= 544)
```

FIGURE 58 – Formule de suppression des outlier datas sur le taux.

10.3.4 Opérations sur la variable situationFamiliale

De la même manière, nous allons établir l'histogramme de la variable situationFamiliale, Figure(59).

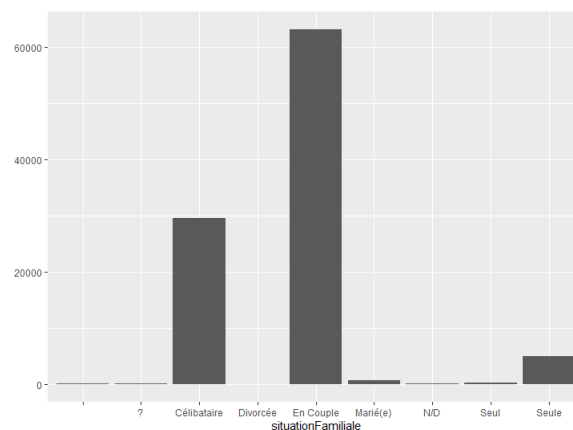


FIGURE 59 – Répartition de la variable situationFamiliale.

Nous remarquons donc la présence de valeurs « », « ? » et « N/D » dans cette variable. Nous les supprimons à l'aide de la ligne de code suivante, voir Figure(60). La fonction «

```
clients <- subset(clients, clients$situationFamiliale != "?" & clients$situationFamiliale != " " & clients$situationFamiliale != "N/D")

clients$situationFamiliale <- droplevels(clients$situationFamiliale)
```

FIGURE 60 – Formule de suppression des outlier datas sur la situation familiale.

droplevels » nous permet d'éliminer les modalités de facteurs qui n'existent plus. Vérifions que nos manipulations ont bien fonctionné, voir Figure(61).

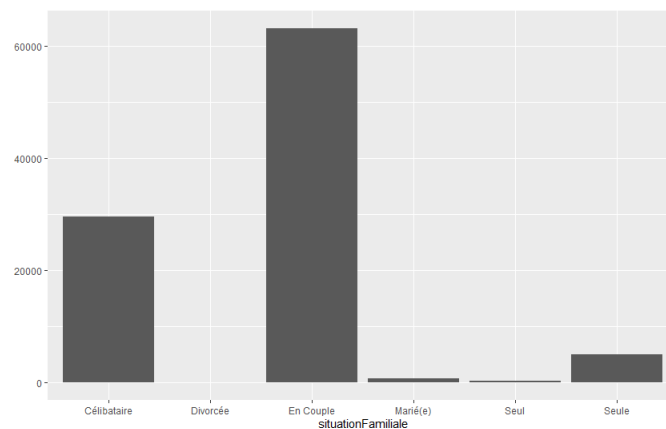


FIGURE 61 – Répartition de la variable situationFamiliale après les opérations.

Nous remarquons donc la disparition de ces valeurs erronées.

10.3.5 Opérations sur la variable nombreEnfantsAcharge

Pour la dernière variable, nous allons établir l'histogramme de la variable nombreEnfantsAcharge, Figure(62).

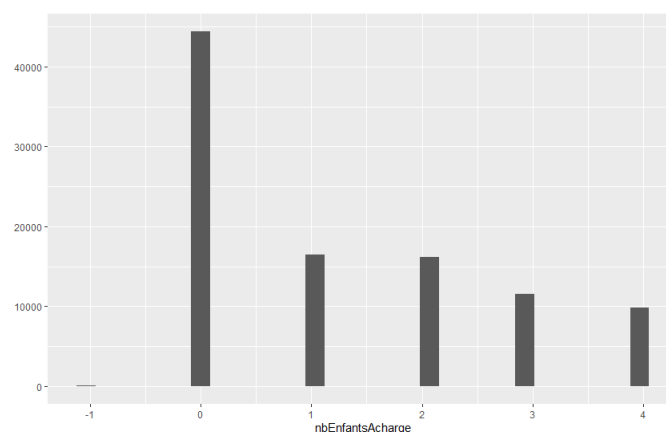


FIGURE 62 – Répartition de la variable nombreEnfantsAcharge.

Nous remarquons qu'un certain nombre de clients possèdent « -1 » enfants à charge. Ceci est impossible, car ce n'est pas en accord avec les domaines de valeurs. Nous allons donc utiliser la ligne de code suivante, voir Figure(63).

```
clients <- subset(clients, clients$nbEnfantsAcharge >= 0)
```

FIGURE 63 – Formule de suppression des outlier datas sur le nombre d'enfants à charge.

Grâce à cela, nous supprimons toutes les valeurs strictement inférieures à 0. Réalisons un nouvel histogramme afin d'afficher le résultat de nos manipulations, Figure(64).

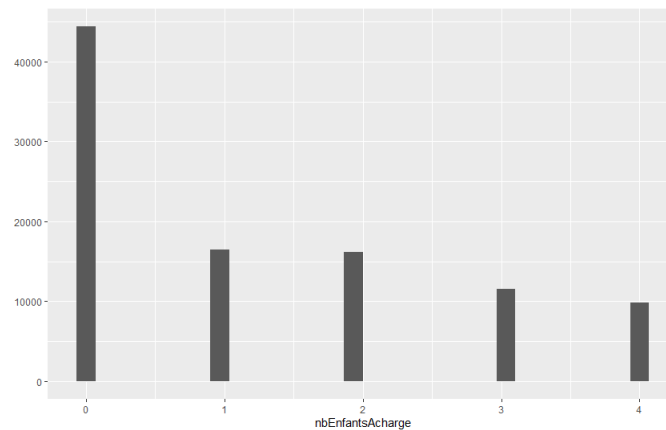


FIGURE 64 – Répartition de la variable situationFamiliare après les opérations.

Nous remarquons donc la disparition de ces valeurs erronées.

10.3.6 Conclusion de l'analyse du fichier Clients

Avant de conclure sur cette table, vérifions qu'il n'y ait plus de valeurs outlier. Pour cela, utilisons de nouveau la fonction « summary » sur cette table, voir Figure(65).

```
#Résumé clients
summary(clients)

#age      sexe      taux      situationFamiliare nbEnfantsAcharge x2eme.voiture immatriculation
#Min.   :18.00   F:13017   Min.   : 544.0   Célibataire:13096   Min.   :0.000   Mode :logical   Length:43521
#1st Qu.:28.00   M:30504   1st Qu.: 589.0   Divorcée   : 22   1st Qu.:0.000   FALSE:37915   Class :character
#Median :42.00   Median : 893.0   En Couple  :27724   Median :1.000   TRUE :5606    Mode :character
#Mean   :43.83   Mean   : 901.7   Marié(e)   : 310   Mean   :1.245
#3rd Qu.:57.00   3rd Qu.:1147.0   Seul      : 124   3rd Qu.:2.000
#Max.   :84.00   Max.   :1399.0   Seul      :2245   Max.   :4.000
```

FIGURE 65 – Récapitulatif du data-frame Clients sur R après les opérations.

Nous remarquons bien, au travers de cette fonction la disparition des valeurs erronées. Maintenant que nous avons fini le traitement de nos données, l'étape suivante est la création de nos ensembles d'apprentissage et de tests.

11 Construction des ensembles de tests et d'apprentissages

Cette étape est cruciale. Nous allons démontrer dans cette partie le cheminement que nous avons suivi afin de mettre en place les différents ensembles et critères pour la création de la variable « catégorie » et les choix que nous avons réalisé pour la création de nos ensembles de tests et d'apprentissages.

11.1 Création des catégories de véhicules

Pour la création de la variable catégorie, une analyse profonde des variables présente dans la table catégories est essentielle. Tout d'abord, intéressons-nous à la variable longueur et sa répartition dans le data-frame des voitures disponibles. Pour cela, nous allons réaliser l'histogramme de cette variable, Figure(66).

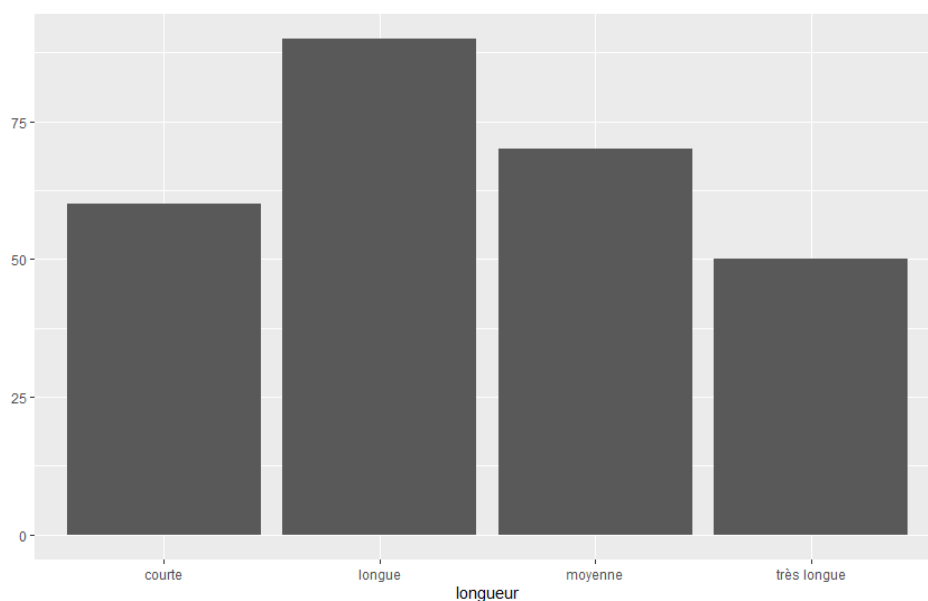


FIGURE 66 – Répartition de la variable Longueur.

Nous pouvons voir que l'on a une proportion assez équivalente pour les voitures de longueur courte, moyenne et très longue. Une des pistes de recherche serait de scinder les voitures de longueur « longue » afin de mieux répartir les véhicules.

Maintenant, intéressons-nous à la variable nbPlaces, deux valeurs sont possibles, soit une voiture comporte 5 places, soit 7 places, donc cette variable pourrait nous intéresser afin de créer la variable « catégorie ». Car une voiture avec 7 places va intéresser une famille avec de nombreux enfants par exemple.

Intéressons-nous aux histogrammes des variables occasion et couleur sur la Figure(67). Nous remarquons une assez bonne répartition des valeurs possibles pour ces variables.

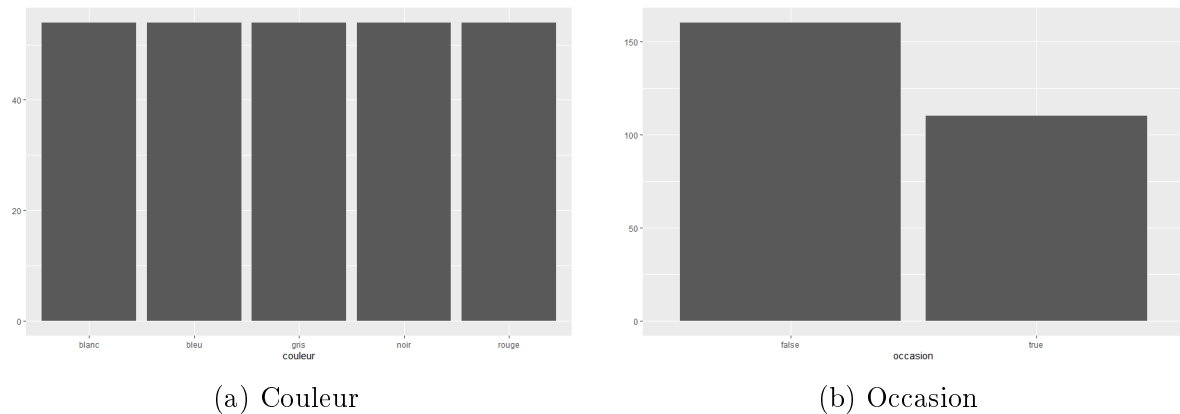


FIGURE 67 – Répartition des différentes variables.

Ainsi, la variable couleur et occasion ne seront pas des critères utiles pour l'établissement de critères pour nos catégories.

Enfin, intéressons-nous aux variables prix et puissance, ce sont deux critères importants pour le choix d'une voiture. En effet, certains profils vont préférer une voiture plutôt luxueuse et puissante quand d'autres vont préférer une voiture moins puissante et donc moins chère. Un nuage de points pourrait permettre de rendre tout cela plus visuel, Figure(68).

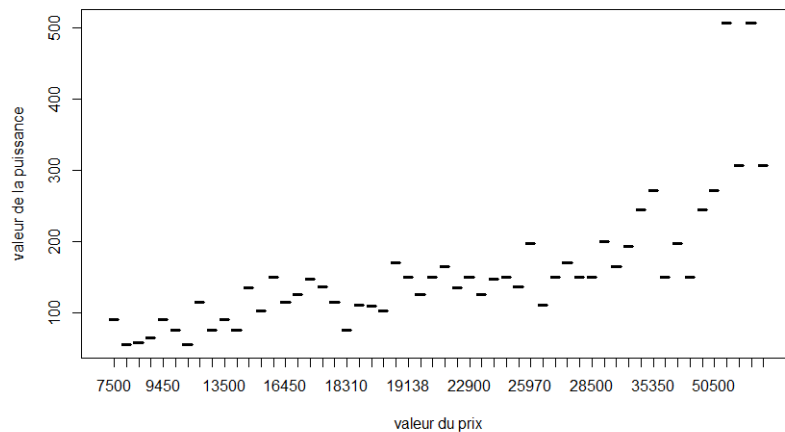


FIGURE 68 – Distribution de la puissance selon le prix

Nous remarquons bien un lien entre la puissance et le prix, ainsi plus la voiture sera puissante plus elle sera chère. Il est donc inutile de mettre un critère de prix et de puissance pour la création de la variable « catégorie », nous aurons juste besoin d'utiliser soit la puissance, soit le prix comme critère.

Au vu de cette analyse, nous pouvons conclure que nos choix de critères pour la variable « catégorie » sont :

- La longueur de la voiture
- La puissance de la voiture
- Le nombre de places

Ainsi, nous allons créer les catégories suivantes de la sorte, Table(1) :

Catégorie de véhicules	Critères
Citadine	Longueur : courte
Compacte	Longueur : moyenne
Routière	Longueur : longue NbPlaces = 5 Puissance < 180
Familiale	Longueur : longue NbPlaces = 7 Puissance < 180
Sportive	Longueur : longue ou très longue 180 < Puissance < 300
Berline	Longueur : très longue Puissance > 300

TABLE 1 – Récapitulatif des catégories de véhicules suivant les critères.

Afin de créer ces catégories sur R nous utilisons le code suivant, Figure(69) :

```
cataloguescategories <- ifelse(catalogueslongueur=="courte","citadine",
                             ifelse(catalogueslongueur=="moyenne","compacte",
                                     ifelse(catalogueslongueur=="longue"& cataloguesnbPlaces== 5& cataloguespuissance<180,"routière",
                                             ifelse(catalogueslongueur=="longue"& cataloguesnbPlaces== 7& cataloguespuissance<180,"familiale",
                                                   ifelse(catalogueslongueur=="longue" | catalogueslongueur=="très longue" & cataloguespuissance >180 & cataloguespuissance <300,"sportive",
                                                         ifelse(catalogueslongueur == "très longue" & cataloguespuissance >300, "berline","rien"))))))))
```

FIGURE 69 – Création des catégories sur R.

Maintenant, que nous avons créé ces catégories nous pouvons créer la variable « catégorie » dans le data-frame Immatriculation, voir Figure(70).

```
immatriculationscategories <- ifelse(immatriculationslongueur=="courte","citadine",
                                     ifelse(immatriculationslongueur=="moyenne","compacte",
                                             ifelse(immatriculationslongueur=="longue"& immatriculationsnbPlaces== 5& immatriculationspuissance<180,"routière",
                                                     ifelse(immatriculationslongueur=="longue"& immatriculationsnbPlaces== 7& immatriculationspuissance<180,"familiale",
                                                           ifelse(immatriculationslongueur=="longue" | immatriculationslongueur=="très longue" & immatriculationspuissance >180 & immatriculationspuissance <300,"sportive",
                                                                 ifelse(immatriculationslongueur == "très longue" & immatriculationspuissance >300, "supercar","rien"))))))))
```

FIGURE 70 – Création de la variable sur le data-frame Immatriculation sur R.

Enfin, l'étape suivante est de réaliser la fusion entre la table Clients et Immatriculation. Nous réalisons le code suivant, voir Figure(71).

```
clients_immatriculations <- merge(immatriculation, clients , by ="immatriculation")
```

FIGURE 71 – Fusion des deux data-frames sur R.

Ce code nous permet de créer une nouvelle table clients_immatriculations, que nous fusionnons par la colonne immatriculation (variable présente dans chacune des data-frames).

11.2 Ensemble de tests et d'apprentissages

Maintenant que nous avons créé le data-frame clients_immatriculations, il nous est nécessaire d'établir un ensemble de tests et un ensemble d'apprentissages. Sans ça, nous ne pourrions pas réaliser une étude des différents classifieurs et donc nous ne pourrions réaliser notre modèle de prédiction.

11.2.1 Suppression de variables sur le data-frame

La 1ère étape que nous réalisons est la suppression de la variable immatriculation, de ce dernier data-frame, car c'est une variable inutile. Nous la supprimons donc avec le code suivant, Figure(72).

```
clients_immatriculations<-clients_immatriculations[,-1]nts , by ="immatriculation")
```

FIGURE 72 – Suppression de la variable immatriculation sur R.

Cependant, nous pouvons supprimer plus de variables. En effet, avec la présence de la variable « catégories », nous considérons qu'il nous est inutile d'avoir encore les variables « longueur », « puissance », « nbPortes ». De plus, il n'y a aucune voiture à 7 places dans notre data-frame immatriculation, donc nous pouvons aussi supprimer la variable « nbPlaces ».

De plus, nous avons vu que la puissance est étroitement liée au prix, donc nous faisons aussi le choix de supprimer cette variable. La distribution de données pour les variables « couleur » et « occasion » étant sensiblement les mêmes, nous pouvons supprimer aussi ces deux variables.

Enfin, nous faisons le choix aussi de supprimer les variables « marque » et « nom », car nous voulons que notre modèle de prédiction nous affiche uniquement la catégorie de véhicules qui pourrait correspondre aux clients.

En résumé, nous supprimons les variables suivantes du data-frame clients_immatriculations :

- Longueur
- Puissance
- nbPortes
- nbPlaces
- Marque
- Nom
- Couleur
- Occasion
- Prix

Nous réalisons ces suppressions grâce au code suivant, sur la Figure(73).

```
clients_immatriculations <- subset(clients_immatriculations, select = -marque) | )
clients_immatriculations <- subset(clients_immatriculations, select = -nom)
clients_immatriculations <- subset(clients_immatriculations, select = -puissance)
clients_immatriculations <- subset(clients_immatriculations, select = -longueur)
clients_immatriculations <- subset(clients_immatriculations, select = -nbPlaces)
clients_immatriculations <- subset(clients_immatriculations, select = -nbPortes)
clients_immatriculations <- subset(clients_immatriculations, select = -couleur)
clients_immatriculations <- subset(clients_immatriculations, select = -occasion)
clients_immatriculations <- subset(clients_immatriculations, select = -prix)
```

FIGURE 73 – Suppression des différentes variables sur le data-frame sur R.

11.2.2 Modification des variables sur le data-frame

Une fois ce code réalisé, la data-frame `clients_immatriculations` ne comporte plus que 7 variables :

- Categories
- Age
- Sexe
- Taux
- situationFamiliare
- nbEnfantsAcharge
- X2eme.voiture

Afin d'appliquer nos différents classifieurs sur ce data-frame il est important de transformer chacune des variables au format « factor ». Nous le réalisons avec le code suivant, Figure(74).

```
clients_immatriculations$categories <- as.factor(clients_immatriculations$categories)
clients_immatriculations$age <- as.factor(clients_immatriculations$age)
clients_immatriculations$sexe <- as.factor(clients_immatriculations$sexe)
clients_immatriculations$taux <- as.factor(clients_immatriculations$taux)
clients_immatriculations$situationFamiliare <- as.factor(clients_immatriculations$situationFamiliare)
clients_immatriculations$nbEnfantsAcharge <- as.factor(clients_immatriculations$nbEnfantsAcharge)
clients_immatriculations$X2eme.voiture <- as.factor(clients_immatriculations$X2eme.voiture)
```

FIGURE 74 – Refactorisation des variable sur le data-frame sur R.

11.2.3 Création des jeux de données

Une fois toutes ces étapes réalisées, nous allons réaliser plusieurs jeux de données pour tester les classifieurs afin d'identifier le meilleur. Pour cela, nous allons créer 2 jeux de données.

Nous nous sommes aperçus que durant nos différents tests, la variable `taux` contenant près de 756 niveaux de facteurs pouvait être à l'origine de nombreux problèmes (Notamment pour les classifieurs : `Nnet`, `kknn` et `svm`). Pour cela, nous faisons le choix que notre premier jeu de tests ne comportera pas la variable `taux`.

Pour notre deuxième jeu de données, nous décidons qu'à l'image de la variable catégorie pour les véhicules, nous allons créer une catégorie pour le `taux`.

Ainsi, nous allons établir nos ensembles d'apprentissage (représentant environ 70% de nos données de la data frame de base) et nos ensembles de tests (représentant les 30% restant). Nous réalisons donc notre premier jeu de données de la manière suivante, Figure(75).

```
"clients_immatriculations : sélection des 29014 premières lignes de clients_immatriculations.(70% de données)"

clients_immatriculations_EA <- clients_immatriculations[1:29014,]

"clients_immatriculations : sélection des dernières lignes de clients_immatriculations.(30% de données)"

clients_immatriculations_ET <- clients_immatriculations[29015:43521,]
```

FIGURE 75 – Création du premier jeu de données sur sur R.

Maintenant que nos deux premiers ensembles ont été établi, on supprime la variable taux par le code suivant, Figure(76).

```
"suppression de la colonne taux de ce data-frame"  
clients_immatriculations_EA <- subset(clients_immatriculations_EA, select = -taux)  
clients_immatriculations_ET <- subset(clients_immatriculations_ET, select = -taux)
```

FIGURE 76 – Suppression de la colonne taux sur le data-frame sur R.

Maintenant que nous avons créé nos 1er jeux de données de tests (avec les data-frames clients_immatriculations_EA et clients_immatriculations_ET), nous allons créer notre deuxième jeu de tests.

Nous réalisons le code suivant afin de créer l'ensemble d'apprentissages et l'ensemble de tests numéro 2, Figure(77).

```
##----- création de deux nouveaux assembles d'apprentissage et de test  
clients_immatriculations_EA2 <- clients_immatriculations[1:29014,]  
clients_immatriculations_ET2 <- clients_immatriculations[29015:43521,]
```

FIGURE 77 – Création du second jeu de données sur sur R.

11.2.4 Création des catégories de taux

L'étape suivante est la création de nos catégories de taux, pour nous aider, nous réalisons une boîte à moustache pour observer la répartition des différentes valeurs du taux, Figure(78).

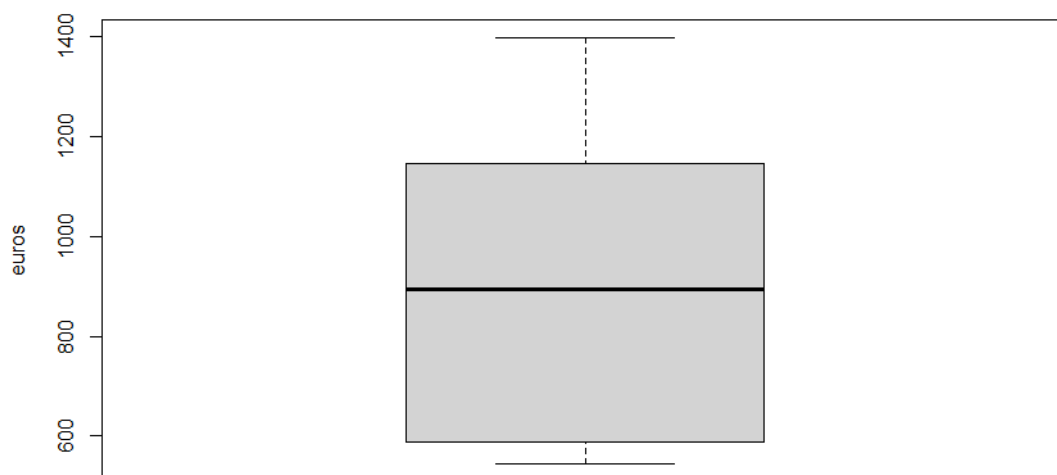


FIGURE 78 – Répartition du taux.

De ce que nous pouvons voir sur ce graphique, c'est que nous avons une assez bonne répartition du taux. Afin de nous aider dans nos choix des catégories de taux. Nous allons utiliser la fonction « summary » pour avoir les valeurs des quartiles, du minimum, du maximum, de la médiane et de la moyenne de cette variable, voir Figure(79).

```
> summary(clients_immatriculations$taux)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 544.0   589.0   893.0   901.7  1147.0  1399.0
```

FIGURE 79 – Suppression de la colonne taux sur le data-frame sur R.

Au vu de ce résultat, nous faisons le choix de créer 4 catégories de taux selon les critères suivants, Table(2).

Catégorie de taux	Bornes
Faible	[0 ;589[
Moyen	[589 ;901.7[
Elevé	[901.7 ;1147[
Très Elevé	[1147 ; [

TABLE 2 – Récapitulatif des catégories de taux en fonction des bornes.

Une fois ces critères établis, nous décidons par le code suivant de créer la variable ctaux, correspondant à la catégorie de chaque taux dans les data-frames clients_immatriculations_EA2 et clients_immatriculations_ET2, Figure(80).

```
clients_immatriculations_EA2$ctaux<- ifelse(clients_immatriculations_EA2$taux <589, "Faible",
                                           ifelse(clients_immatriculations_EA2$taux <901.7, "Moyen",
                                                  ifelse(clients_immatriculations_EA2$taux < 1147, "Elevé", "Très Elevé")))

clients_immatriculations_ET2$ctaux<- ifelse(clients_immatriculations_ET2$taux <589, "Faible",
                                           ifelse(clients_immatriculations_ET2$taux <901.7, "Moyen",
                                                  ifelse(clients_immatriculations_ET2$taux < 1147, "Elevé", "Très Elevé")))
```

FIGURE 80 – Création de la variable sur les data-frames sur R.

Nous réalisons un histogramme de cette dernière variable, afin d'observer si nous avons une bonne répartition, Figure(81).

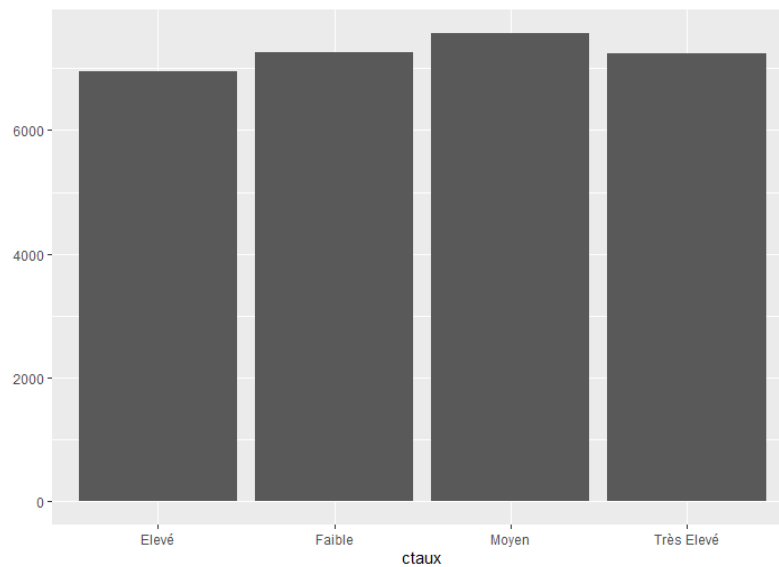


FIGURE 81 – Répartition des catégories de taux.

Nous pouvons voir que nos critères nous ont permis d'avoir une très bonne répartition des catégories de taux dans le data-frame clients_immatriculations_EA2.

Maintenant que nous avons créé ctaux il est nécessaire que l'on passe cette donnée au format factor et que l'on supprime la variable taux. Nous exécutons le code suivant, Figure(82).

```
clients_immatriculations_EA2$ctaux <- as.factor(clients_immatriculations_EA2$ctaux)
clients_immatriculations_ET2$ctaux <- as.factor(clients_immatriculations_ET2$ctaux)

clients_immatriculations_EA2$taux <- as.factor(clients_immatriculations_EA2$taux)
clients_immatriculations_ET2$taux <- as.factor(clients_immatriculations_ET2$taux)

##-----Suppression de la colonne taux

clients_immatriculations_EA2 <- subset(clients_immatriculations_EA2, select = -taux)
clients_immatriculations_ET2 <- subset(clients_immatriculations_ET2, select = -taux)
```

FIGURE 82 – Répartition des catégories de taux.

Ainsi, nous venons de créer nos deux jeux de données au travers de différents ensembles de tests et d'apprentissages. Maintenant, nous allons pouvoir appliquer les différents classifieurs à nos deux jeux de données, afin de créer notre modèle de prédiction.

12 Test des classifieurs et création du modèle de prédiction

Dans cette dernière partie, nous allons vous présenter les tests que nous avons réalisés avec les différents classifieurs. Mais aussi conclure sur le classifieur le plus performant, à travers différentes mesures telles que le rappel, la précision, le taux d'erreur et l'area under curve (auc). Enfin, lorsque nous aurons le classifieur le plus performant, nous construirons notre modèle de prédiction.

12.1 Liste des classifieurs

Tout d'abord, nous allons vous présenter les différents classifieurs que nous allons tester sur nos deux jeux de tests. Voici la liste des classifieurs que nous allons utiliser :

- C5.0
- Naive bayes (Classifieur Bayésien)
- Svm (Machine à vecteurs de support)
- Kknn (K-Nearest Neighbor)
- Nnet (Réseaux de neurones)
- randomForest (Forêts d'Arbres de Décision Aléatoires)

Nous importons les packages contenant les classifieurs précédents avec le code suivant, Figure(83).

```
##### installation des librairies
install.packages("stringr")
install.packages("ggplot2")
install.packages("C50")
install.packages("randomForest")
install.packages("naivebayes")
install.packages("e1071")
install.packages("nnet")
install.packages("kknn")
install.packages("pROC")

library(stringr)
library(ggplot2)
library(C50)
library(randomForest)
library(naivebayes)
library(e1071)
library(nnet)
library(kknn)
library(pROC)
```

FIGURE 83 – Importation des packages sur R.

12.2 1er jeu de données

Nous allons donc effectuer ces tests sur les data-frames :

- clients_immatriculations_EA
- clients_immatriculations_ET

12.2.1 C5.0

Commençons avec C5.0. Tout d'abord nous réalisons l'apprentissage du classifieur par le code suivant, Figure(84).

```
# Apprentissage du classifieur de type arbre de décision
treeC <- C5.0(categories ~., clients_immatriculations_EA)
```

FIGURE 84 – Apprentissage du classifieur C5.0.

Nous réalisons ensuite les prédictions sur l'ensemble de tests, Figure(85).

```
C_class <- predict(treeC, clients_immatriculations_ET, type="class")
```

FIGURE 85 – Prédiction du classifieur C5.0.

Réalisons maintenant la matrice de confusion, voir Figure(86).

```
table(clients_immatriculations_ET$categories, C_class)

####      berline citadine compacte routière sportive
##berline    2424         3         0         400    1287
##citadine      2     4632         0         0         1
##compacte      0     1003         0         1         1
##routière     30         3         0       744       717
##sportive     475         1         0         0     2783
```

FIGURE 86 – Matrice de confusion du classifieur C5.0.

Maintenant que nous avons la matrice de confusion, calculons le taux d'erreur, rappel et précision de ce classifieur, Table(3).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2424	3	0	400	1287	0.589
Citadine	2	4632	0	0	1	0.999
Compacte	0	1003	0	1	1	0
Routière	30	3	0	744	717	0.498
Sportive	475	1	0	0	2783	0.853
Précision	0.827	0.821	x	0.649	0.581	0.729

Erreur	0.270
--------	-------

TABLE 3 – Calculs des différents taux du classifieur C5.0.

On mesure donc un taux d'erreur de 0.27.

Maintenant, nous exécutons le code suivant afin d'obtenir la valeur de l'auc, Figure(87).

```
# Test du classifieur : probabilités pour chaque prediction
c_prob <- predict(treeC, clients_immatriculations_ET, type="prob")

# Calcul de l'AUC
c_auc <- multiclass.roc(clients_immatriculations_ET$categories, c_prob)
print (c_auc)
```

FIGURE 87 – Calcul de la valeur de l'auc du classifieur C5.0.

Ici, nous utilisons la fonction « multiclass.roc » de la bibliothèque pROC, car nous sommes en présence d'une classification « multiclass », Figure(88).

```
Call:
multiclass.roc.default(response = clients_immatriculations_ET$categories,
  predictor = c_prob)

Data: multivariate predictor c_prob with 5 levels of clients_immatriculations_ET$categories: berline, citadine, compacte, routière, sportive.
Multi-class area under the curve: 0.8947
```

FIGURE 88 – Utilisation de la fonction « multiclass.roc » pour le classifieur C5.0.

Ainsi avec C5.0 nous avons un auc de 0.895 sur ce 1er jeu de tests.

12.2.2 Naive-Bayes

Pour chacun des classifieurs, nous allons procéder de la même manière. A savoir réaliser l'apprentissage du classifieur, Figure(89).

```
# Apprentissage du classifieur de type naive bayes
nb <- naive_bayes(categories~., clients_immatriculations_EA)
```

FIGURE 89 – Apprentissage du classifieur Naive-Bayes.

Réaliser les prédictions sur l'ensemble de tests, Figure(90).

```
# Test du classifieur : classe predite
nb_class <- predict(nb, clients_immatriculations_ET, type="class")
```

FIGURE 90 – Prédiction du classifieur Naive-Bayes.

Réalisons maintenant la matrice de confusion, voir Figure(91).

```
# Matrice de confusion
table(clients_immatriculations_ET$categories, nb_class)

##nb_class
##          berline citadine compacte routière sportive
##berline    2484      55         0       400     1175
##citadine    860    3444      330         0         1
##compacte     0     652     351         1         1
##routière     67      35         0      744      648
##sportive    601     113         0         0     2545
```

FIGURE 91 – Matrice de confusion du classifieur Naive-Bayes.

Calculons les différents indices, Table(4).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2484	55	0	400	1175	0.603
Citadine	860	3444	330	0	1	0.743
Compacte	0	652	351	1	1	0.349
Routière	67	35	0	744	648	0.498
Sportive	601	113	0	0	2545	0.781
Précision	0.619	0.801	0.515	0.649	0.582	0.659

Erreur	0.340
--------	-------

TABLE 4 – Calculs des différents taux du classifieur Naive-Bayes.

Nous remarquons ici un taux d'erreur de 0.34 pour le classifieur Naive-Bayes. Enfin la dernière étape est l'obtention de l'auc, Figure(92).

```
Call:
multiclass.roc.default(response = clients_immatriculations_ET$categories, predictor = nb_prob)

Data: multivariate predictor nb_prob with 5 levels of clients_immatriculations_ET$categories: berline, citadine, compacte, routière, sportive.
Multi-class area under the curve: 0.8873
```

FIGURE 92 – Utilisation de la fonction « multiclass.roc » pour le classifieur Naive-Bayes.

Pour le classifieur Naive-Bayes, nous obtenons un auc de 0.887.

12.2.3 SVM

Pour le classifieur nous obtenons la matrice de confusion suivante, Figure(93).

```
# Matrice de confusion
table(clients_immatriculations_ET$categories, svm_class)

##svm_class
##          berline citadine compacte routière sportive
##berline    2597         5         0        288    1224
##citadine      2      4632         0         0         1
##compacte      1     1003         0         0         1
##routière    286         4         0       522     682
##sportive    602         4         0         0    2653
```

FIGURE 93 – Matrice de confusion du classifieur SVM.

A partir de cela on obtient les indicateurs suivants, Table(5).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2597	5	0	288	1224	0.631
Citadine	2	4362	0	0	1	0.999
Compacte	1	1003	0	0	1	0
Routière	286	4	0	522	682	0.349
Sportive	602	4	0	0	2653	0.814
Précision	0.744	0.820	x	0.644	0.582	0.717

Erreur	0.283
--------	-------

TABLE 5 – Calculs des différents taux du classifieur SVM.

Pour ce classifieur on a un taux de d'erreur de 0.283.

Maintenant déterminons l'auc du classifieur svm, Figure(94).

```
# Test du classifieur : probabilités pour chaque prediction
svm_prob <- predict(svm, clients_immatriculations_ET, probability=TRUE)

# Recuperation des probabilités associées aux predictions
svm_prob <- attr(svm_prob, "probabilities")

# Conversion en un data frame
svm_prob <- as.data.frame(svm_prob)

# Calcul de l'AUC
svm_auc <- multiclass.roc(clients_immatriculations_ET$categories, svm_prob)
print (svm_auc)

## sCall:
##multiclass.roc.default(response = clients_immatriculations_ET$categories, predictor = svm_prob)
##
##Data: multivariate predictor svm_prob with 5 levels of clients_immatriculations_ET$categories: citadine, berline, routière, sportive
##Multi-class area under the curve: 0.9019
#_____#
```

FIGURE 94 – Utilisation de la fonction « multiclass.roc » pour le classifieur SVM.

Nous obtenons donc un auc de 0.902.

12.2.4 Kknn

Pour le classifieur nous obtenons la matrice de confusion suivante, Figure(95).

```
##          berline citadine compacte routière sportive
##berline    2579         2         1        396    1136
##citadine     2       4178       454         0         1
##compacte     1        724       279         1         0
##routière    400         3         0       558       533
##sportive    911         0         1       270     2077
```

FIGURE 95 – Matrice de confusion du classifieur Kknn.

A partir de cela on obtient les indicateurs suivants, Table(6).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2597	2	1	396	1136	0.628
Citadine	2	4178	454	0	1	0.999
Compacte	1	724	279	1	0	0.277
Routière	400	3	0	558	533	0.373
Sportive	911	0	1	270	2077	0.637
Précision	0.664	0.851	0.379	0.455	0.554	0.667

Erreur	0.333
--------	-------

TABLE 6 – Calculs des différents taux du classifieur Kknn.

Pour ce classifieur on a un taux de d'erreur de 0.333.

Maintenant déterminons l'auc du classifieur Kknn, Figure(96).

```
# Conversion des probabilités en data frame
knn_prob <- as.data.frame(kknn$prob)

knn_auc <- multiclass.roc(clients_immatriculations_ET$categories, knn_prob)
print(knn_auc)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET$categories, predictor = knn_prob)
##
##data: multivariate predictor knn_prob with 5 levels of clients_immatriculations_ET$categories: berline, citadine, compacte, routière, sportive
##Multi-class area under the curve: 0.8706
```

FIGURE 96 – Utilisation de la fonction « multiclass.roc » pour le classifieur Kknn.

Pour Kknn nous obtenons un auc de 0.870.

12.2.5 Nnet

Pour le classifieur nous obtenons la matrice de confusion suivante, Figure(97).

```
##      berline citadine compacte routière sportive
##berline 2414      3      0      408     1289
##citadine  2     4469     163      0      1
##compacte  0      802     201      1      1
##routière 19      3      0     755     717
##sportive 482      1      0      0     2776
```

FIGURE 97 – Matrice de confusion du classifieur Nnet.

A partir de cela on obtient les indicateurs suivants, Table(7).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2414	3	0	408	1289	0.586
Citadine	2	4469	163	0	1	0.964
Compacte	0	802	201	1	1	0.2
Routière	19	3	0	755	717	0.505
Sportive	482	1	0	0	2776	0.851
Précision	0.827	0.846	0.552	0.648	0.580	0.731

Erreur	0.268
--------	-------

TABLE 7 – Calculs des différents taux du classifieur Nnet.

Pour ce classifieur on a un taux de d'erreur de 0.268.

Maintenant déterminons l'auc du classifieur Nnet, Figure(98).

```
# Test du classifieur : probabilités pour chaque prediction
nn_prob <- predict(nnet, clients_immatriculations_ET, type="raw")
nn_auc <- multiclass.roc(clients_immatriculations_ET$categories, nn_prob)
print(nn_auc)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET$categories, predictor = nn_prob)
##
##Data: multivariate predictor nn_prob with 5 levels of clients_immatriculations_ET$categories: berline, citadine, compacte, routière, sportive
##Multi-class area under the curve: 0.9076
```

FIGURE 98 – Utilisation de la fonction « multiclass.roc » pour le classifieur Nnet.

L'auc du classifieur Nnet est de 0.908.

12.2.6 randomForest

Ce classifieur nécessite un travail en amont, en effet la présence d'un trop grand nombre de facteur pour la variable age, empêche son utilisation.

Nous supprimons la variable age de nos ensembles d'apprentissages et de tests, voir Figure(99).

```
clients_immatriculations_EA <- subset(clients_immatriculations_EA, select = -age)
clients_immatriculations_ET <- subset(clients_immatriculations_ET, select = -age)
```

FIGURE 99 – Suppression de la variable age pour le classifieur randomForest.

Maintenant déterminons la matrice de confusion, Figure(100).

```
##   berline citadine compacte routière sportive
##berline    2411      3      0      0      1700
##citadine     2    4632      0      0      1
##compacte     0    1003      0      0      2
##routière     1      3      0      0    1490
##sportive    475      1      0      0    2783
```

FIGURE 100 – Matrice de confusion du classifieur randomForest.

A partir de cela, déterminons les indices liés à ce classifieurs, Table(8).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2411	3	0	0	1700	0.586
Citadine	2	4632	0	0	1	0.999
Compacte	0	1003	0	0	2	0
Routière	1	3	0	0	1490	0
Sportive	475	1	0	0	2783	0.854
Précision	0.834	0.820	x	x	0.465	0.677

Erreur	0.323
--------	-------

TABLE 8 – Calculs des différents taux du classifieur randomForest.

Nous pouvons remarquer un taux d'erreur de 0.323 pour ce classifieur.

Maintenant déterminons l'auc de ce classifieur, Figure(101).

```
rf_prob <- predict(RF, clients_immatriculations_ET, type="prob")

RF_auc <- multiclass.roc(clients_immatriculations_ET$categories, rf_prob)
print (RF_auc)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET$categories, predictor = rf_prob)
##
##Data: multivariate predictor rf_prob with 5 levels of clients_immatriculations_ET$categories: berline, citadine, compacte, routière, sportive
##Multi-class area under the curve: 0.6938
```

FIGURE 101 – Utilisation de la fonction « multiclass.roc » pour le classifieur randomForest.

Nous obtenons donc un auc de 0.694.

12.2.7 Conclusion du 1er jeu de données

Résumons nos derniers résultat à travers ce tableau, Table(9).

Classifieur	Taux d'erreur	AUC
C5.0	0.270	0.895
Naive Bayes	0.340	0.887
SVM	0.283	0.902
Kknn	0.333	0.870
Nnet	0.268	0.908
Random Forest	0.323	0.694

TABLE 9 – Récapitulatif des classifieurs avec les deux taux.

Dans ce 1er jeu de tests, nous remarquons que le classifieur le plus performant en termes de taux d'erreur et d'auc est Nnet.

12.3 2eme jeu de données

Nous allons donc effectuer ces tests sur les data-frames :

- clients_immatriculations_EA2
- clients_immatriculations_ET2

Nous allons réaliser les mêmes étapes que pour le 1er jeu de données. Ainsi pour chacun des classifieurs nous vous proposerons à chaque fois la matrice de confusion, le calcul des indices et l'auc.

12.3.1 C5.0

Pour ce classifieur nous obtenons la matrice de confusions suivante, Figure(102).

```
# Matrice de confusion
table(clients_immatriculations_ET2$categories, c_class2)

##          c_class2
##          berline citadine compacte routière sportive
##berline      2226         2         1        545      1340
##citadine       2      4263        369         0         1
##compacte       0       652        351         1         1
##routière       0         3         0      1352        139
##sportive      105         1         0       461      2692
```

FIGURE 102 – Matrice de confusion du classifieur C5.0.

Nous calculons les indices de performance liés à ce classifieur, Table(10).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2226	2	1	545	1340	0.541
Citadine	2	4263	369	0	1	0.920
Compacte	0	652	351	1	1	0.349
Routière	0	3	0	1352	139	0.905
Sportive	105	1	0	461	2692	0.826
Précision	0.954	0.866	0.487	0.573	0.645	0.750

Erreur	0.25
--------	------

TABLE 10 – Calculs des différents taux du classifieur C5.0.

Ainsi pour ce classifieur nous avons un taux d'erreur de 0.25.
Déterminons l'auc, voir Figure(103).

```
# Test du classifieur : probabilités pour chaque prediction
c_prob2 <- predict(treec2, clients_immatriculations_ET2, type="prob")

# calcul de l'AUC
c_auc2 <- multiclass.roc(clients_immatriculations_ET2$categories, c_prob2)
print (c_auc2)

## Call:
## multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = c_prob2)
##
##Data: multivariate predictor c_prob2 with 5 levels of clients_immatriculations_ET2$categories: berline, citadine, compacte, routière
##Multi-class area under the curve: 0.9349
```

FIGURE 103 – Utilisation de la fonction « multiclass.roc » pour le classifieur C5.0.

Pour ce classifieur nous obtenons un auc de 0.935.

12.3.2 Naive Bayes

Déterminons la matrice de confusion, Figure(104).

```
# Matrice de confusion
table( clients_immatriculations_ET2$categories, nb_class2)

##nb_class2
##          berline citadine compacte routière sportive
##berline    2592      246         1        240      1035
##citadine    855     3374        400         0         6
##compacte     0      628        376         1         0
##routière    269      67         0       457        701
##sportive    571     364         1         0       2323
```

FIGURE 104 – Matrice de confusion du classifieur Naive Bayes.

Calculons le rappel, la précision et le taux d'erreur pour ce classifieur, Table(11).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2592	246	1	240	1035	0.630
Citadine	855	3374	400	0	6	0.728
Compacte	0	628	376	1	0	0.374
Routière	269	67	0	457	701	0.306
Sportive	571	364	1	0	2323	0.713
Précision	0.605	0.721	0.483	0.655	0.571	0.629

Erreur	0.371
--------	-------

TABLE 11 – Calculs des différents taux du classifieur Naive Bayes.

Ainsi pour ce classifieur nous avons un taux d'erreur de 0.371. Maintenant, déterminons l'auc, voir Figure(105).

```
# calcul de l'AUC
nb_auc2 <-multiclass.roc(clients_immatriculations_ET2$categories, nb_prob2)
print (nb_auc2)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = nb_prob2)
##
##Data: multivariate predictor nb_prob2 with 5 levels of clients_immatriculations_ET2$categories: berline, citadine, compacte, routière, sportive
##Multi-class area under the curve: 0.915
```

FIGURE 105 – Utilisation de la fonction « multiclass.roc » pour le classifieur Naive Bayes.

Pour le classifieur Naive Bayes, on mesure un auc de 0.915.

12.3.3 SVM

Etablissons la matrice de confusion liée à ce classifieur, Figure(106).

```
# Matrice de confusion
table(clients_immatriculations_ET2$categories, svm_class2)

##svm_class2
##          berline citadine compacte routière sportive
##berline    2788         5         0         217      1104
##citadine     2      4632         0         0         1
##compacte     1      1003         0         0         1
##routière    627         9         0        728        130
##sportive    602         4         0        431       2222
```

FIGURE 106 – Matrice de confusion du classifieur SVM.

Ensuite, calculons les indicateurs lié à cette matrice, Table(12).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2788	5	0	217	1104	0.678
Citadine	2	4632	0	0	1	0.999
Compacte	1	1003	0	0	1	0
Routière	627	9	0	728	130	0.487
Sportive	602	4	0	431	2222	0.682
Précision	0.695	0.819	x	0.529	0.642	0.715

Erreur	0.285
--------	-------

TABLE 12 – Calculs des différents taux du classifieur SVM.

On peut mesurer un taux d'erreur de 0.285 pour le classifieur SVM. Maintenant, déterminons l'auc, voir Figure(107).

```
# Test du classifieur : probabilités pour chaque prediction
svm_prob2 <- predict(svm2, clients_immatriculations_ET2, probability=TRUE)

# Recuperation des probabilités associées aux predictions
svm_prob2 <- attr(svm_prob2, "probabilities")

# Conversion en un data frame
svm_prob2 <- as.data.frame(svm_prob2)

# Calcul de l'AUC
svm_auc2 <- multiclass.roc(clients_immatriculations_ET2$categories, svm_prob2)
print (svm_auc2)

##call:
##multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = svm_prob2)
##
##Data: multivariate predictor svm_prob2 with 5 levels of clients_immatriculations_ET2$categories: citadine, berline, routière, sporti
##Multi-class area under the curve: 0.9332
```

FIGURE 107 – Utilisation de la fonction « multiclass.roc » pour le classifieur SVM.

Ainsi l'auc de SVM est de 0.933.

12.3.4 Kknn

Déterminons la matrice de confusion, Figure(108).

```
# Matrice de confusion
table(clients_immatriculations_ET2$categories, kknn2$fitted.values)

##          berline citadine compacte routière sportive
##berline    2632         3         0        336    1143
##citadine      2       4136       496         0         1
##compacte      0        569       434         0         2
##routière    247         3         0       831       413
##sportive    673         1         1       318     2266
```

FIGURE 108 – Matrice de confusion du classifieur Kknn.

Calculons les indicateurs pour ce classifieur, Table(13).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2632	3	0	336	1143	0.640
Citadine	2	4136	496	0	1	0.892
Compacte	0	569	434	0	2	0.432
Routière	247	3	0	831	413	0.556
Sportive	673	1	1	318	2266	0.695
Précision	0.741	0.878	0.467	0.560	0.592	0.71

Erreur	0.290
--------	-------

TABLE 13 – Calculs des différents taux du classifieur Kknn.

On note un taux d'erreur de 0.290 pour ce classifieur.
Enfin déterminons l'auc, voir Figure(109).

```
# Conversion des probabilités en data frame
knn_prob2 <- as.data.frame(kknn2$prob)

knn_auc2 <- multiclass.roc(clients_immatriculations_ET2$categories, knn_prob2)
print(knn_auc2)

## Call:
##multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = knn_prob2)
##
##Data: multivariate predictor knn_prob2 with 5 levels of clients_immatriculations_ET2$categories: berline, citadine, compacte, routière
##Multi-class area under the curve: 0.9164
```

FIGURE 109 – Utilisation de la fonction « multiclass.roc » pour le classifieur Kknn.

Ainsi, nous observons un auc de 0.916 pour le classifieur Kknn.

12.3.5 Nnet

Déterminons la matrice de confusion, Figure(110).

```
# Matrice de confusion
table(clients_immatriculations_ET2$categories, nn_class2)

##nn_class2
##      berline citadine compacte routière sportive
##berline    2247      2        1      504    1360
##citadine      2    4226     406      0        1
##compacte      0     606     397      1        1
##routière     34      3        0    1259     198
##sportive    120      0        1     421    2717
```

FIGURE 110 – Matrice de confusion du classifieur Nnet.

Calculons les indicateurs pour ce classifieur, Table(14).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2247	2	1	504	1360	0.546
Citadine	2	4226	406	0	1	0.912
Compacte	0	606	397	1	1	0.395
Routière	34	3	0	1259	198	0.843
Sportive	120	0	1	421	2717	0.834
Précision	0.935	0.874	0.493	0.576	0.635	0.748

Erreur	0.252
--------	-------

TABLE 14 – Calculs des différents taux du classifieur Nnet.

Le classifieur Nnet sur ce jeu de tests a donc un taux d'erreur de 0.252.
Enfin déterminons l'auc, voir Figure(111).

```
# Test du classifieur : probabilités pour chaque prediction
nn_prob2 <- predict(nnet2, clients_immatriculations_ET2, type="raw")
nn_auc2 <- multiclass.roc(clients_immatriculations_ET2$categories, nn_prob2)
print(nn_auc2)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = nn_prob2)
##
##Data: multivariate predictor nn_prob2 with 5 levels of clients_immatriculations_ET2$categories: berline, citadine, compacte, routière, sportive
##Multi-class area under the curve: 0.9463
```

FIGURE 111 – Utilisation de la fonction « multiclass.roc » pour le classifieur Nnet.

Nous observons que l'auc du classifieur nnet est de 0.946.

12.3.6 randomForest

De la même manière que dans le 1er jeu de tests nous supprimons la variable age des deux data-frames. Déterminons la matrice de confusion, Figure(112).

```
table(clients_immatriculations_ET2$categories, result.RF2)

##result.RF2
##          berline citadine compacte routière sportive
##berline    2078         3         0        231    1802
##citadine      2       4632         0         0         1
##compacte      0       1003         0         0         2
##routière      0         3         0       766       725
##sportive     90         1         0       461     2707
```

FIGURE 112 – Matrice de confusion du classifieur randomForest.

Calculons les indicateurs pour ce classifieur, Table(15).

	Berline	Citadine	Compacte	Routière	Sportive	Rappel
Berline	2078	3	0	231	1802	0.505
Citadine	2	4632	0	0	1	0.999
Compacte	0	1003	0	0	2	0
Routière	0	3	0	766	725	0.513
Sportive	90	1	0	461	2707	0.831
Précision	0.958	0.821	x	0.525	0.517	0.702

Erreur	0.298
--------	-------

TABLE 15 – Calculs des différents taux du classifieur randomForest.

Le classifieur randomForest sur ce jeu de tests a donc un taux d'erreur de 0.298. Enfin déterminons l'auc, voir Figure(113).

```
rf_prob2 <- predict(RF2, clients_immatriculations_ET2, type="prob")

RF2_auc <- multiclass.roc(clients_immatriculations_ET2$categories, rf_prob2)
print (RF2_auc)

##Call:
##multiclass.roc.default(response = clients_immatriculations_ET2$categories, predictor = rf_prob2)
##
##Data: multivariate predictor rf_prob2 with 5 levels of clients_immatriculations_ET2$categories: berline, citadine, compacte, rout
##Multi-class area under the curve: 0.8156
```

FIGURE 113 – Utilisation de la fonction « multiclass.roc » pour le classifieur randomForest.

Nous remarquons un auc de 0.816 pour ce classifieur.

12.3.7 Conclusion du 2ème jeu de données

Résumons nos derniers résultat à travers ce tableau, Table(16).

Classifieur	Taux d'erreur	AUC
C5.0	0.250	0.935
Naive Bayes	0.371	0.915
SVM	0.285	0.933
Kknn	0.290	0.916
Nnet	0.252	0.916
Random Forest	0.298	0.816

TABLE 16 – Récapitulatif des classifieurs avec les deux taux.

Encore une fois sur ce jeu de données, Nnet est le classifieur le plus performant, car il possède le meilleur auc et le plus bas taux d'erreur. Cependant, un problème subsiste avec ce classifieur, son exécution est assez lente. De plus, le classifieur C5.0 est un classifieur tout aussi performant et son exécution est assez rapide.

Au vu des deux jeux de données, nous faisons le choix d'utiliser le classifieur C5.0 pour réaliser nos prédictions.

13 Prédiction

Cette partie concerne donc l'aboutissement de tous nos travaux sur l'établissement de notre modèle de prédiction. Cependant, nous ne pouvons pas encore utiliser les classifieurs « treeC » et « treeC2 » puisque nous avons fait, en amont, la supposition que nous supprimions la colonne taux.

13.1 Création des data-frames

Ainsi, nous allons créer un ultime ensemble d'apprentissages et ensemble de tests, où cette fois-ci, nous inclurons la colonne taux. Car il est essentiel que nos variables coïncident avec celles du marketing.

Nous créons donc ces deux ensembles, voir Figure(114).

```
##----- création de deux nouveaux ensembles d'apprentissage et de test
clients_immatriculations_EA3 <- clients_immatriculations[1:29014,]

clients_immatriculations_ET3 <- clients_immatriculations[29015:43521,]
```

FIGURE 114 – Création des deux ensembles sur le data-frame sur R.

13.2 Classifieur prédit

Une fois ces ensembles créés appliquons le classifieur C5.0, comme sur la Figure(115).

```
# Apprentissage du classifieur de type arbre de décision
treeCFinal <- C5.0(categories ~., clients_immatriculations_EA3)
```

FIGURE 115 – Apprentissage avec le classifieur C5.0 sur le data-frame sur R.

13.3 Modification du data-frame

Une fois l'apprentissage du classifieur réalisée, nous pouvons alors réaliser l'application de C5.0 au data-frame marketing. Cependant, il faut réaliser certaines étapes en amont. En effet, les variables x2èmevoiture, age, taux, nbEnfantsAcharge de Marketing ne sont pas au bon format. Pour cela, nous effectuons les étapes suivantes, voir Figure(116) :

```
####-----transformation des données en facteur-----
marketing$X2eme.voiture <- marketing$X2eme.voiture %>% str_to_upper()

marketing$age <- as.factor(marketing$age)

marketing$taux <- as.factor(marketing$taux)

marketing$nbEnfantsAcharge <- as.factor(marketing$nbEnfantsAcharge)
```

FIGURE 116 – Modification du data-frame sur R.

13.4 Prédiction sur le data-frame Marketing

Une fois ces étapes réalisées, il nous reste plus qu'à établir les prédictions. Nous réalisons aussi la prédiction à partir des données que nous avons importé dans la partie Gestion des Données, voir en *Annexe*(66).

Pour réaliser la prédiction, nous créons la variable `class.treeCpred`, qui contiendra les prédictions de catégories pour chaque profil du data-frame Marketing, voir Figure(117).

```
#=== C5.0 ===#
class.treeCpred <- predict(treecFinal, marketing)
```

FIGURE 117 – Réalisation de la prédiction sur le data-frame sur R.

Enfin nous créons le data-frame *resultat1*. Ce data-frame sera la concaténation des data-frame marketing et `class.treeCpred`, voir Figure(118) et Figure(119) pour le résultat.

```
resultat1 <- data.frame(marketing, class.treeCpred)
```

FIGURE 118 – Création du data-frame résultat.

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	X2eme.voiture	Catégorie prédite
1	21	F	1396	Célibataire	0	FALSE	citadine
2	35	M	223	Célibataire	0	FALSE	compacte
3	48	M	401	Célibataire	0	FALSE	citadine
4	26	F	420	En Couple	3	TRUE	sportive
5	80	M	530	En Couple	3	FALSE	berline
6	27	F	153	En Couple	2	FALSE	routière
7	59	F	572	En Couple	2	FALSE	routière
8	43	F	431	Célibataire	0	FALSE	compacte
9	64	M	559	Célibataire	0	FALSE	compacte
10	22	M	154	En Couple	1	FALSE	routière
11	79	F	981	En Couple	2	FALSE	routière
12	55	M	588	Célibataire	0	FALSE	citadine
13	19	F	212	Célibataire	0	FALSE	citadine
14	34	F	1112	En Couple	0	FALSE	sportive
15	60	M	524	En Couple	0	TRUE	citadine
16	22	M	411	En Couple	3	TRUE	sportive
17	58	M	1192	En Couple	0	FALSE	sportive
18	54	F	452	En Couple	3	TRUE	sportive
19	35	M	589	Célibataire	0	FALSE	compacte
20	59	M	748	En Couple	0	TRUE	citadine

FIGURE 119 – Affichage du data-frame resultat1 avec la catégorie prédite.

Ainsi, nous observons qu'une catégorie de véhicule a bien été prédite pour chacun des profils du fichier marketing. Nous avons donc établi notre modèle de prédiction.

Enfin nous générons, grâce au code suivant, un fichier *.csv* regroupant toutes les données du data-frame *resultat1*, voir Figure(120).

```
# Enregistrement du fichier de resultats au format csv
write.table(resultat1, file='predictions.csv', sep="\t", dec=".", row.names = F)
```

FIGURE 120 – Enregistrement des prédictions sur un fichier .csv.

14 Conclusion de l'Analyse de données

Dans cette partie, nous avons pu vous montrer les différentes étapes que nous avons suivi afin d'établir notre modèle prédiction à travers la classification supervisée.

Pour cela nous avons dû démarrer avec une analyse exploratoire des données, car il est nécessaire de supprimer toutes données qui n'étaient pas conformes.

Une fois ce traitement réalisé, il est important de mettre en place des tests afin d'analyser quel est le meilleur classifieur. Pour cela, nous avons utilisé les indicateurs importants suivant : taux d'erreur et l'auc.

Enfin, une fois que nous avons sélectionné le meilleur classifieur, ici C5.0, nous n'avons plus qu'à réaliser les prédictions de catégorie de véhicules pour le data-frame marketing. Ainsi nous avons réussi à établir notre modèle de prédiction. Nous avons réalisé une prédiction en créant 5 catégories de voitures.

Une des pistes d'amélioration pourrait être de prédire une marque et le nom d'un véhicule en plus de la catégorie de voiture qui pourrait convenir à chaque profil.

PARTIE MapReduce

15 Objectif et contexte de la partie MapReduce

Après avoir réaliser la Gestion des Données, le concessionnaire nous a appelé et nous a fait part que certaines données étaient perdues lorsqu'il nous a transmis les fichiers originaux.

En effet, les détails sur l'émission CO2 / le coût d'énergie / la valeur de Bonus/Malus pour la taxation par marque étaient absente sur le fichier *catalogue.csv* et ceux, pour l'ensemble des modèles de voiture.

En cherchant sur Internet, nous avons trouvé un fichier CO2.csv. C'est une autre base des données qui a certaines informations qui pouvait nous être utile aider mais cependant elle n'est pas parfaite. Elle ne contient pas tous les marques et modèles des voitures qui sont dans le catalogue du concessionnaire. De plus, le format de stockage est différent (la marque et le modèle sont dans une même colonne), il y a des valeurs manquant (colonne Bonus/Malus) et des valeurs erronés (colonne Bonus/Malus par exemple contient -6000€ 1 a la place de -6000€).

Le but est d'écrire un programme map/reduce avec Hadoop qui va permettre d'adapter le fichier CO2.csv pour intégrer les informations complémentaires dans la table catalogue du concessionnaire (ajouter des colonnes "Bonus / Malus", "Rejets CO2 g/km", "Cout Energie").

16 Création des scripts Java

16.1 Fichier CO2.java

Il nous faut créer la classe Driver qui contient le main du programme Hadoop, voir Figure(122).

```
public class CO2 {

    public static void main(String[] args) throws Exception {

        Configuration config = new Configuration();
        String[] ourArgs = new GenericOptionsParser(config, args).getRemainingArgs();

        Job jobMP = Job.getInstance(config, "CO2 MapReduce");

        jobMP.setJarByClass(CO2.class);
        jobMP.setMapperClass(CO2Map.class);
        jobMP.setReducerClass(CO2Reduce.class);

        jobMP.setOutputKeyClass(Text.class);
        jobMP.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(jobMP, new Path(ourArgs[0]));
        FileOutputFormat.setOutputPath(jobMP, new Path(ourArgs[1]));

        if(jobMP.waitForCompletion(true))
            System.exit(0);
        System.exit(-1);
    }
}
```

FIGURE 121 – Programme principal pour lancer le Map Reduce.

Ce programme permet d’instancier le nom des autres class qui vont être utilisées. Le format d’entrée et de sortie est aussi spécifié (Text).

16.2 Fichier CO2Map.java

Ensuite, il nous faut réalisé les deux autres class qu’on a instancier dans le programme main. Il faut donc créer la fonction Map, voir Figure(123).

La fonction Map transforme les entrées du fichier en paires <key,value>, les traite et génère un autre ensemble de paires <key,value> intermédiaires en sortie. Ici, nous allons regrouper les marques comme Key, Figure(124).

```
public class CO2Map extends Mapper<Object, Text, Text, Text> {

    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
```

FIGURE 122 – Programme pour réaliser la fonction Map.

```
context.write(new Text(marque), new Text(new_value));
```

FIGURE 123 – Création du couple Key Value.

16.3 Fichier CO2Reduce.java

Enfin, il nous faut écrire la dernière class du programme Map Reduce. Il faut donc créer la fonction Reduce, voir Figure(125).

La fonction Reduce transforme également les entrées en paires <key,value> et génère une des paires <key,value> en sortie. Ici, nous allons modifier la value en calculant la moyenne des 3 colonnes pour chaque marque(chaque Key), Figure(126).

```
public class CO2Reduce extends Reducer<Text, Text, Text, Text> {  
  
    public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
```

FIGURE 124 – Programme pour réaliser la fonction Reduce.

```
while(i.hasNext()) {  
    String node = i.next().toString();  
  
    System.err.print(key);  
    System.err.print("    ");  
    System.err.println(node);  
    //on sépare les 3 colonnes  
    String[] splitted_node = node.split("\\|");  
    malus_bonus = splitted_node[0];  
    rejet = splitted_node[1];  
    cout = splitted_node[2];  
    //on ajoute les valeurs  
    sumBonus_Malus += Integer.parseInt(malus_bonus);  
    sumRejet += Integer.parseInt(rejet);  
    sumCout += Integer.parseInt(cout);  
    //on compte les voitures par key  
  
    count++;  
}  
//on calcule les moyennes  
avgMalus_Bonus = sumBonus_Malus/count;  
avgRejet = sumRejet/count;  
avgCout = sumCout/count;  
//on écrit les moyennes pour chaque key  
context.write(key, new Text(avgMalus_Bonus + "\t" + avgRejet + "\t" + avgCout));
```

FIGURE 125 – Modification de la Value pour chaque Key.

17 Programme Map Reduce

17.1 Import des fichiers Java

Une fois le programme réalisé. On se connecte sur la machine virtuelle, et on y dépose nos fichiers Java, voir Figure(126).

/home/pcorentin/MapReduce/				
Nom	Taille	Date de modification	Droits	Propriét...
..		26/03/2021 14:25:36	rw-r--r--	pcorentin
data		26/03/2021 14:30:07	rw-rw-r--	pcorentin
CO2Reduce.java	3 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin
CO2Map.java	3 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin
CO2.java	2 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin

FIGURE 126 – Fichiers Java disponible sur la machine virtuelle.

17.2 Compilation des fichiers Java

Ensuite, nous exécutons une liste de commandes qui nous permette de compiler le code Java, voir Figure(127). On remarque bien la création des fichiers, voir Figure(128).

```
Last login: Fri Mar 26 13:20:21 2021 from 88.125.119.248
pcorentin@vps-flf17ale:~$ cd MapReduce
pcorentin@vps-flf17ale:~/MapReduce$ javac CO2*
pcorentin@vps-flf17ale:~/MapReduce$ mkdir -p org/co2
pcorentin@vps-flf17ale:~/MapReduce$ mv CO2*.class org/co2/
pcorentin@vps-flf17ale:~/MapReduce$ jar -cvf CO2.jar -C . org
added manifest
adding: org/(in = 0) (out= 0) (stored 0%)
adding: org/co2/(in = 0) (out= 0) (stored 0%)
adding: org/co2/CO2Reduce.class(in = 2302) (out= 1054) (deflated 54%)
adding: org/co2/CO2Map.class(in = 2470) (out= 1154) (deflated 53%)
adding: org/co2/CO2.class(in = 1549) (out= 815) (deflated 47%)
pcorentin@vps-flf17ale:~/MapReduce$
```

FIGURE 127 – Compilation des fichiers Java.

/home/pcorentin/MapReduce/				
Nom	Taille	Date de modification	Droits	Propriét...
..		26/03/2021 14:25:36	rw-r--r--	pcorentin
org		26/03/2021 16:59:08	rw-rw-r--	pcorentin
data		26/03/2021 14:30:07	rw-rw-r--	pcorentin
CO2Reduce.java	3 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin
CO2Map.java	3 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin
CO2.java	2 KB	26/03/2021 10:56:15	rw-rw-r--	pcorentin
CO2.jar	4 KB	26/03/2021 16:59:18	rw-rw-r--	pcorentin

FIGURE 128 – Création des fichiers d'exécution pour MapReduce.

Avant de procéder à l'exécution du programme MapReduce, nous devons nous assurer que le fichier d'entrée est bien sur HDFS.

Les lignes de commandes, voir Figure(129), nous permettent de mettre le fichier sur HDFS, et de vérifier que le fichier est bien présent.

```
pcorentin@vps-flf17ale:~/MapReduce$ hadoop fs -put data/CO2.csv /user/pcorentin
put: `/user/pcorentin/CO2.csv': File exists
pcorentin@vps-flf17ale:~/MapReduce$ hadoop fs -ls /user/pcorentin/
Found 1 items
-rw-r--r--  1 pcorentin studentgroup    39354 2021-03-26 13:45 /user/pcorentin/CO2.csv
pcorentin@vps-flf17ale:~/MapReduce$
```

FIGURE 129 – Fichier d'entrée pour MapReduce sur HDFS.

17.3 Exécution du programme Map Reduce

Nous pouvons maintenant procéder à l'exécution du programme MapReduce, voir Figure(130). On remarque que la partie Map et Reduce se déroule sans problème.

```
pcorentin@vps-flf17ale:~/MapReduce$ hadoop jar CO2.jar org.co2.CO2 /user/pcorentin/CO2.csv /u
ser/pcorentin/resultsmareduce
2021-03-26 16:05:15,609 INFO client.DefaultNoHARMAFailoverProxyProvider: Connecting to Resourc
eManager at /0.0.0.0:8032
2021-03-26 16:05:15,960 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path
: /tmp/hadoop-yarn/staging/pcorentin/.staging/job_1616340117381_0023
2021-03-26 16:05:16,168 INFO input.FileInputFormat: Total input files to process : 1
2021-03-26 16:05:16,216 INFO mapreduce.JobSubmitter: number of splits:1
2021-03-26 16:05:16,329 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_161634011
7381_0023
2021-03-26 16:05:16,329 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-03-26 16:05:16,494 INFO conf.Configuration: resource-types.xml not found
2021-03-26 16:05:16,494 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-03-26 16:05:16,559 INFO impl.YarnClientImpl: Submitted application application_161634011
7381_0023
2021-03-26 16:05:16,596 INFO mapreduce.Job: The url to track the job: http://vps-flf17ale.vps
.ovh.net:8088/proxy/application_1616340117381_0023/
2021-03-26 16:05:16,597 INFO mapreduce.Job: Running job: job_1616340117381_0023
2021-03-26 16:05:22,673 INFO mapreduce.Job: Job job_1616340117381_0023 running in uber mode :
false
2021-03-26 16:05:22,674 INFO mapreduce.Job:  map 0% reduce 0%
2021-03-26 16:05:26,724 INFO mapreduce.Job:  map 100% reduce 0%
2021-03-26 16:05:31,760 INFO mapreduce.Job:  map 100% reduce 100%
2021-03-26 16:05:31,768 INFO mapreduce.Job: Job job_1616340117381_0023 completed successfully
2021-03-26 16:05:31,859 INFO mapreduce.Job: Counters: 54
```

FIGURE 130 – Fichier d'entrée pour MapReduce sur HDFS.

18 Résultat du programme Map Reduce

18.1 Résultat sur HDFS

Une fois le programme Map Reduce exécuté, nous pouvons tout d'abord consulter si le fichier de résultat est bien présent sur HDFS, puis ensuite l'afficher, voir Figure(131).

```
pcorentin@vps-fl17ale:~/MapReduce$ hadoop fs -ls /user/pcorentin/resultsmapreduce
Found 2 items
-rw-r--r-- 1 pcorentin studentgroup 0 2021-03-26 16:05 /user/pcorentin/resultsmapreduce/_SUCCESS
-rw-r--r-- 1 pcorentin studentgroup 396 2021-03-26 16:05 /user/pcorentin/resultsmapreduce/part-r-00000
pcorentin@vps-fl17ale:~/MapReduce$ hadoop fs -cat /user/pcorentin/resultsmapreduce/part-r-00000
AUDI -2400 26 191
BENTLEY 0 84 102
BMW -631 39 80
CITROEN -6000 0 347
DS -3000 16 159
HYUNDAI -4000 8 151
JAGUAR -6000 0 271
KIA -3000 15 132
LAND 0 69 78
MERCEDES 7790 187 749
MINI -3000 21 126
MITSUBISHI 0 40 98
NISSAN 5802 160 681
PEUGEOT -3000 15 144
PORSCH 0 69 89
RENAULT -6000 0 206
SKODA -666 27 98
SMART -6000 0 191
TESLA -6000 0 245
TOYOTA 0 32 43
VOLKSWAGEN -1714 23 96
VOLVO 0 42 72
```

FIGURE 131 – Resultat du MapReduce sur HDFS.

On remarque bien que trois colonnes ("Bonus / Malus", "Rejets CO2 g/km", "Cout Energie") ont été créées pour chaque marque de voiture.

18.2 Export du fichier depuis HDFS

Une fois le fichier de résultat obtenu, nous devons l'exporter afin de pouvoir le joindre à notre fichier initial *catalogue.csv*. Nous utilisons la commande suivante, voir Figure(132).

```
pcorentin@vps-fl17ale:~/MapReduce$ hadoop fs -get /user/pcorentin/resultsmapreduce/part-r-00000
pcorentin@vps-fl17ale:~/MapReduce$ mv part-r-00000 resultat_mapreduce_co2.txt
```

FIGURE 132 – Export du fichier resultat.

De retour, sur notre machine virtuelle, nous pouvons constater la présence de notre fichier resultat, voir Figure(133).

/home/pcorentin/MapReduce/				
Nom	Taille	Date de modification	Droits	Propriét...
..		26/03/2021 14:25:36	rw-r--r--	pcorentin
org		26/03/2021 16:59:08	rw-r--r--	pcorentin
data		26/03/2021 14:30:07	rw-r--r--	pcorentin
resultat_mapreduce_co2.txt	1 KB	26/03/2021 17:07:39	rw-r--r--	pcorentin
CO2Reduce.java	3 KB	26/03/2021 10:56:15	rw-r--r--	pcorentin
CO2Map.java	3 KB	26/03/2021 10:56:15	rw-r--r--	pcorentin
CO2.java	2 KB	26/03/2021 10:56:15	rw-r--r--	pcorentin
CO2.jar	4 KB	26/03/2021 16:59:18	rw-r--r--	pcorentin

FIGURE 133 – Fichier resultat exporté.

19 Jonction avec le fichier catalogue

Il faut désormais joindre le fichier résultat Map Reduce, avec le fichier catalogue initial. Pour cela nous allons utiliser R Studio pour fusionner les deux fichiers.

19.1 Conversion en csv

Le fichier resultat est un fichier *.txt*, il nous faut tout d'abord le mettre au bon format, *.csv*, puis ensuite qu'il soit complété de la même manière que le fichier catalogue. Pour ce faire, nous allons utiliser quelques commandes dans PowerShell pour convertir le fichier et le mettre au bon format, voir Figure(134) et Figure(135).

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\Users\c.poirier> cd .\Desktop\
PS C:\Users\c.poirier\Desktop> cd .\new\
PS C:\Users\c.poirier\Desktop\new> cd .\ProjetVoiture\
PS C:\Users\c.poirier\Desktop\new\ProjetVoiture> cd .\MapReduce\
PS C:\Users\c.poirier\Desktop\new\ProjetVoiture\MapReduce> import-csv .\resultat_mapreduce_co2.txt -delimiter "`t"
| export-csv csv_mapreduce.csv -NoTypeInformation
PS C:\Users\c.poirier\Desktop\new\ProjetVoiture\MapReduce>
```

FIGURE 134 – Commande PS pour convertir le fichier txt en csv.

```
PS C:\Users\c.poirier\Desktop\new\ProjetVoiture\MapReduce> $Test.Replace("`",",").TrimStart("`").TrimEnd("`")
AUDI,-2400,26,191
BENTLEY,0,84,102
BMW,-631,39,80
CITROEN,-6000,0,347
DS,-3000,16,159
HYUNDAI,-4000,8,151
JAGUAR,-6000,0,271
KIA,-3000,15,132
LAND,0,69,78
MERCEDES,7790,187,749
MINI,-3000,21,126
MITSUBISHI,0,40,98
NISSAN,5802,160,681
PEUGEOT,-3000,15,144
PORSCHE,0,69,89
RENAULT,-6000,0,206
SKODA,-666,27,98
SMART,-6000,0,191
TESLA,-6000,0,245
TOYOTA,0,32,43
VOLKSWAGEN,-1714,23,96
VOLVO,0,42,72
```

FIGURE 135 – Commande PS pour modifier les lignes sans les quotes.

19.2 Jonction sur R Studio

Nous allons utiliser R Studio, et joindre les deux fichiers.

19.2.1 Import des fichiers

Afin de réaliser l'import des fichiers, nous allons utiliser la fonction `read.csv` comme sur la Figure(136).

```
#import des fichiers
co2mapreduce <- read.csv("C:/Users/c.poirier/Desktop/new/ProjetVoiture/DATA/Mapreduce.csv",
                        header = FALSE, sep = ",", dec = ".")
catalogue <- read.csv("C:/Users/c.poirier/Desktop/new/ProjetVoiture/DATA/Catalogue.csv",
                    header = TRUE, sep = ",", dec = ".")
view(co2mapreduce)
view(catalogue)
```

FIGURE 136 – Importation des fichiers dans R Studio.

19.2.2 Modification des colonnes

Une fois l'import réalisé des fichiers, nous allons renommer les colonnes du data-frame. De plus, nous modifions le contenu de la colonne *marque* du data-frame *co2mapreduce* pour qu'elle soit similaire a la colonne du data-frame *catalogue*, voir Figure(137), pour pouvoir les fusionner.

```
#Rename les 3 colonnes
colnames(co2mapreduce) <- c("marque", "Bonus/Malus", "Rejet", "Cout Energie")

#Lower cas pour la colonne marge
co2mapreduce$marque = tolower(co2mapreduce$marque)
catalogue$marque = tolower(catalogue$marque)
```

FIGURE 137 – Modification des colonnes marque.

19.2.3 Fusion des data-frames

Nous pouvons désormais fusionner les deux data-frames sur la colonne *marque* et ainsi créer le data-frame *catalogue_modifie*, voir Figure(138).

```
#Fusion des fichiers Mapreduce.csv et Catalogue.csv :
catalogue_modifie <- merge(catalogue, co2mapreduce, by= "marque", all = TRUE)
```

FIGURE 138 – Fusion des deux data-frames.

19.2.4 Nettoyage du data-frame

Une fois la fusion réalisée, il faut nettoyer le data-frame créer. En effet, des valeurs NA ce sont introduites, pour les marques qui sont contenues dans le fichier catalogue et non dans le fichier CO2 et inversement.

Tout d'abord, nous transformons l'ensemble de ces valeurs par 0. Ensuite, nous supprimons les lignes où la marque n'est pas contenue dans le catalogue du concessionnaire, voir Figure(139). Nous laissons les valeurs à 0 pour les voitures de la concession dont nous n'avons pas de données.

```
#suppression des voitures qui ne sont pas dans le catalogue initialement
catalogue_modifie[is.na(catalogue_modifie)] <- 0
catalogue_modifie[catalogue_modifie[,2]!=0,]
view(catalogue_modifie)
```

FIGURE 139 – Nettoyage du data-frame.

19.2.5 Export du data-frame

Nous observons maintenant le résultat du nouveau catalogue, avec l'ensemble des nouvelles colonnes, voir Figure(140).

Enfin nous générons, grâce au code suivant, un fichier *.csv* regroupant toutes les données du data-frame *catalogue_modifie*, voir Figure(141).

	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	Bonus/Malus	Rejet	Cout Energie
1	audi	A3 2.0 FSI	150	moyenne	5	5	noir	true	19950	-2400	26	191
2	audi	A3 2.0 FSI	150	moyenne	5	5	noir	false	28500	-2400	26	191
3	audi	A3 2.0 FSI	150	moyenne	5	5	rouge	false	28500	-2400	26	191
4	audi	A3 2.0 FSI	150	moyenne	5	5	gris	true	19950	-2400	26	191
5	audi	A3 2.0 FSI	150	moyenne	5	5	blanc	false	28500	-2400	26	191
6	audi	A3 2.0 FSI	150	moyenne	5	5	blanc	true	19950	-2400	26	191
7	audi	A3 2.0 FSI	150	moyenne	5	5	bleu	true	19950	-2400	26	191
8	audi	A3 2.0 FSI	150	moyenne	5	5	bleu	false	28500	-2400	26	191
9	audi	A3 2.0 FSI	150	moyenne	5	5	gris	false	28500	-2400	26	191
10	audi	A3 2.0 FSI	150	moyenne	5	5	rouge	true	19950	-2400	26	191

FIGURE 140 – Affichage du data-frame avec les nouvelles colonnes.

```
#Export du fichier
write.table(catalogue_modifie, file = 'c:/Users/c.poirier/Desktop/new/Projetvoiture/DATA/Catalogue_Modifie.csv',
            sep=";", dec = ".", row.names = F, quote = F)
```

FIGURE 141 – Export du data-frame en fichier *.csv*.

Annexe

Importation des données via un driver SQL Plus

Dans cette partie nous allons vous présenter comment nous avons procédé pour réaliser le nettoyage de nos données à travers l'utilisation d'Oracle et de commandes SQL. Pour mener à bien ce nettoyage, nous allons d'abord nous connecter à notre base de données par un driver, comme sur la Figure(142).

```
##classPath : add path to drivers jdbc

drv <- RJDBC::JDBC(driverClass = "oracle.jdbc.OracleDriver", classPath =
  sys.glob("C:/Logiciels/oracle/oracle/drivers/*"))

#Connexion OK
conn <- dbConnect(drv, "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)
  (HOST=144.21.67.201)(PORT=1521))(CONNECT_DATA=
  (SERVICE_NAME=pdbest21.631174089.oraclecloud.internal)))",
  "GUINALDO2B20", "GUINALDO2B2001")
```

FIGURE 142 – Connexion à la base de données.

Une fois ce code exécuté, il ne nous reste plus qu'à importer nos données provenant des fichiers .csv de la manière suivante, Figure(143).

```
#Enregistrement de la table Marketing dans la DB Oracle
Marketing <- read.csv("C:/Users/v.guinaldo/Desktop/test redécoupage/test/Marketing.csv", header = TRUE,
  sep = ",", dec = ".")
names(Marketing)[6] = ("deuxiemevoiture")

dbwriteTable(conn,"Marketing",Marketing,
  rownames=FALSE, overwrite = TRUE, append = FALSE)

#Enregistrement de la table Catalogue dans la DB Oracle
Catalogue <- read.csv("C:/Users/v.guinaldo/Desktop/test redécoupage/test/Catalogue.csv", header = TRUE,
  sep = ",", dec = ".")

dbwriteTable(conn,"Catalogue",Catalogue,
  rownames=FALSE, overwrite = TRUE, append = FALSE)

#Enregistrement de la table Client dans la DB Oracle
Client <- read.csv("C:/Users/v.guinaldo/Desktop/test redécoupage/test/Clients_6.csv", header = TRUE,
  sep = ",", dec = ".")
names(Client)[6] = ("deuxiemevoiture")

Client$age <- as.integer(Client$age)
Client$taux <- as.integer(Client$taux)
Client$nbEnfantsAcharge <- as.integer(Client$nbEnfantsAcharge)

dbwriteTable(conn,"Client",Client,
  rownames=FALSE, overwrite = TRUE, append = FALSE)
```

FIGURE 143 – Importation des données sur R.

Nous ne pouvons pas importer le fichier Immatriculations, car ce dernier contenant un nombre trop important de données.

Enfin une fois que ces fichiers ont bien été importés nous pouvons visualiser cela avec les commandes suivantes, Figure(144) :

```
#visualisation des tables
allTables <- dbGetQuery(conn, "SELECT owner, table_name FROM all_tables where
                                owner = 'GUINALDO2B20'")

MarketingDB <- dbGetQuery(conn, "select * from Marketing")
CatalogueDB <- dbGetQuery(conn, "select * from Catalogue")
ClientsDB <- dbGetQuery(conn, "select * from Client")
```

FIGURE 144 – Visualisation des données sur R.

Tri des données sur SQL Plus

Enfin, il nous faut réaliser le nettoyage des données sur Oracle via des commandes SQL.

Tout d'abord, nous allons nettoyer la table Client.

Pour la variable age nous effectuons la commande suivante, voir Figure(145) :

```
delete from client where age < 18 ;
```

FIGURE 145 – Suppression des datas suivants l'âge.

Pour la variable sexe, une étape en amont est essentielle. En effet, il est nécessaire de passer toutes les données au même format (à savoir M et F), voir Figure(146) :

```
UPDATE client SET client.sexe = 'M' WHERE sexe = 'Masculin' OR sexe='Homme' ;
UPDATE client SET client.sexe = 'F' WHERE sexe = 'Féminin' OR sexe='Femme' ;
```

FIGURE 146 – Modification des datas suivants le sexe.

Enfin nous supprimons toutes données non-conformes, voir Figure(147) :

```
delete from client where sexe != 'M' And sexe != 'F' ;
```

FIGURE 147 – Suppression des datas suivants le sexe.

Ensuite nous supprimons toutes les données identifier comme outlier pour la variable taux, voir Figure(148) :

```
delete from client where taux < 544 ;
```

FIGURE 148 – Suppression des datas suivants le taux.

A l'image de la variable sexe, nous supprimons toutes les valeurs non conformes pour la variable situationfamiliale, voir Figure(149) :

```
delete from client
where situationfamiliale != 'Célibataire'
AND situationfamiliale != 'Divorcée'
AND situationfamiliale != 'En Couple'
AND situationfamiliale != 'Marié(e)'
AND situationfamiliale != 'Seul'
AND situationfamiliale != 'Seule';
```

FIGURE 149 – Suppression des datas suivants la situation familiale.

L'étape suivante est la suppression de toutes les données outlier pour la variable nbEnfantAcharge, voir Figure(150) :

```
delete from client where nbenfantsacharge <0;
```

FIGURE 150 – Suppression des datas suivants le nombre d'enfants à charge.

Enfin l'ultime étape de notre nettoyage est la suppression de toutes les données non conformes pour la variable DeuxièmeVoiture, pour cela on utilise la commande suivante, voir Figure(151) :

```
delete from client where nbenfantsacharge <0;
```

FIGURE 151 – Suppression des datas suivants le nombre de voitures.

Nous considérons que le nettoyage de données via des commandes s'est bien déroulé puisque nous avons obtenu le même nombre de lignes dans la table Client avec un nettoyage de données avec R ou Oracle.

Prédiction à l'issue de l'import via la Gestion des Données

Dans cette étape, nous allons réaliser la même chose que dans la *section Prediction* 13.4. Le but de cette étape est de montrer que la façon d'importer les données n'influe pas sur le résultat final.

Pour ces données, nous avons nettoyé les données en amont sur SQL Plus, en utilisant la méthode vue en *Annexe*(65), au dessus. Nous avons conclu que le nettoyage c'est bien passée puisqu'après l'import sur R Studio, nous avons le même nombre de données que lorsque nous nettoyons les données directement sur R (43521 clients pour le data-frame clients, ...).

Ensuite, nous réalisons les mêmes étapes sur R que celles traités dans l'autre partie. A savoir :

- La création de nos catégories de véhicules
- La création du data-frame *clients_immatriculations* (résultat de la fusion des deux data-frames clients et immatriculations)
- La création des ensembles d'apprentissages et des ensembles de tests
- L'établissement de nos jeux de tests, afin de vérifier quel est le meilleur classifieur (analyse du taux d'erreur, de la précision, du rappel et de l'indice auc)

Afin d'éviter toute répétition, nous faisons le choix de ne pas remettre le détail de ces actions dans cette partie du rapport.

Nous allons nous concentrer sur la partie application de notre modèle de prédiction. Dans cette partie, nous utiliserons aussi le classifieur C5.0, meilleur classifieur à l'issue des tests. Ensuite, nous réalisons les commandes suivantes, voir Figure(152).

```
#### C5.0 ===#
class.treeCpred <- predict(treecFinal, marketing)

class.treeCpred

resultat11 <- data.frame(marketing, class.treeCpred)

names(resultat11)[7]= ("Catégorie prédite")

view(resultat11)
```

FIGURE 152 – Commandes pour prédire la catégorie avec le classifieur C5.0.

En théorie nous devrions retrouver le même résultat que dans l'autre partie. Vérifions cela avec la fonction « View » Nous observerons le data-frame *resultat11* qui est le résultat des prédictions des catégories de voiture pour les clients sélectionnés par le service marketing. On remarque que les data-frames *resultat1*, voir Figure(119), et le data-frame *resultat11*, voir Figure(153), sont similaires.

On en conclut que l'import, comme prévu, n'a aucun impact sur le résultat final des prédiction. Manipuler les données depuis les bases de données, permet d'avoir une meilleure gestion des données, et peut simplifier grandement un projet avec un nombres très important de données.

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	X2eme.voiture	Catégorie prédite
1	21	F	1396	Célibataire	0	FALSE	citadine
2	35	M	223	Célibataire	0	FALSE	compacte
3	48	M	401	Célibataire	0	FALSE	citadine
4	26	F	420	En Couple	3	TRUE	sportive
5	80	M	530	En Couple	3	FALSE	berline
6	27	F	153	En Couple	2	FALSE	routière
7	59	F	572	En Couple	2	FALSE	routière
8	43	F	431	Célibataire	0	FALSE	compacte
9	64	M	559	Célibataire	0	FALSE	compacte
10	22	M	154	En Couple	1	FALSE	routière
11	79	F	981	En Couple	2	FALSE	routière
12	55	M	588	Célibataire	0	FALSE	citadine
13	19	F	212	Célibataire	0	FALSE	citadine
14	34	F	1112	En Couple	0	FALSE	sportive
15	60	M	524	En Couple	0	TRUE	citadine
16	22	M	411	En Couple	3	TRUE	sportive
17	58	M	1192	En Couple	0	FALSE	sportive
18	54	F	452	En Couple	3	TRUE	sportive
19	35	M	589	Célibataire	0	FALSE	compacte
20	59	M	748	En Couple	0	TRUE	citadine

FIGURE 153 – Affichage du data-frame resultat11 avec la catégorie prédite.

Références

- [1] MongoDB : <https://www.mongodb.com/fr>
- [2] Oracle NoSQL : <https://www.oracle.com/fr/database/nosql-cloud.html>
- [3] Hadoop HDFS : <https://www.talend.com/fr/resources/what-is-hadoop/>
- [4] HIVE : <https://meritis.fr/big-data-analyse-donnees-apache-hive/>
- [5] Oracle SQL : <https://www.next-decision.fr/editeurs-bi/base-de-donnees/oracle-database>
- [6] R Studio : <https://rstudio.com/>

Table des figures

1	Importation du fichier .csv dans MongoDB.	6
2	Exportation du fichier JSON depuis MongoDB.	6
3	Résultat de l'exportation du fichier JSON depuis MongoDB.	6
4	Fonction permettant l'importation du fichier sur OracleNoSQL.	7
5	Fonction permettant l'ajout d'une ligne sur la table.	8
6	Création de la table Immatriculation.	8
7	Commandes pour exécuter le script Java.	8
8	Connection au KvStore	9
9	Affichage de la table Immatriculation.	9
10	Fonction permettant l'importation du fichier sur OracleNoSQL.	9
11	Fonction permettant l'ajout d'une ligne sur la table.	10
12	Création de la table Marketing.	10
13	Commandes pour exécuter le script Java.	10
14	Connection au KvStore	11
15	Affichage de la table Marketing.	11
16	Commande Hadoop pour importer le fichier.	13
17	Commande Hadoop pour voir le fichier.	13
18	Commande pour se connecter à Hive.	14
19	Création de la table externe immatriculation.	15
20	Vérification de la création de la table externe immatriculation.	15
21	Création de la table externe marketing.	16
22	Vérification de la création de la table externe marketing.	16
23	Création de la table externe catalogue.	17
24	Vérification de la création de la table externe catalogue.	17
25	Création de la table client.	18
26	Fichier de contrôle pour populer la table client.	19
27	Commande pour populer la table avec le fichier de commande.	19
28	Affichage des premières lignes de la table client.	19
29	Nombre de ligne dans la table client.	19
30	Connexion à la base Oracle.	20
31	Création de la table immatriculation sur la base Oracle.	20
32	Vérification de la création de la table immatriculation sur la base Oracle.	21
33	Création de la table marketing sur la base Oracle.	22
34	Vérification de la création de la table marketing sur la base Oracle.	22
35	Création de la table catalogue sur la base Oracle.	23
36	Vérification de la création de la table catalogue sur la base Oracle.	23
37	Création d'un user pour manipuler les données.	24
38	Installation du package pour l'import Oracle.	25
39	Création d'une connexion ODBC.	25
40	Création des data-frames	25
41	Importation des fichiers .csv.	27
42	Importation réussite des fichiers .csv sur R.	27

43	Récapitulatif du data-frame Immatriculation sur R.	28
44	Répartition de la variable longueur.	28
45	Répartition de la variable nbPortes.	29
46	Répartition des différentes variables.	29
47	Suppression des données en double.	30
48	Modification du data-frame Clients.	31
49	Suppression des datas null du data-frame Clients.	31
50	Récapitulatif du data-frame Clients sur R.	31
51	Répartition de la variable Age.	32
52	Formule de suppression des outlier datas sur l'age.	32
53	Répartition de la variable Sexe.	32
54	Formule de suppression des outlier datas sur le sexe.	33
55	Modification des datas sur le sexe.	33
56	Répartition de la variable Sexe après les opérations.	33
57	Répartition de la variable Taux.	34
58	Formule de suppression des outlier datas sur le taux.	34
59	Répartition de la variable situationFamiliale.	34
60	Formule de suppression des outlier datas sur la situation familiale.	35
61	Répartition de la variable situationFamiliale après les opérations.	35
62	Répartition de la variable nombreEnfantsAcharge.	35
63	Formule de suppression des outlier datas sur le nombre d'enfants à charge.	36
64	Répartition de la variable situationFamiliale après les opérations.	36
65	Récapitulatif du data-frame Clients sur R après les opérations.	36
66	Répartition de la variable Longueur.	37
67	Répartition des différentes variables.	38
68	Distribution de la puissance selon le prix	38
69	Création des catégories sur R.	39
70	Création de la variable sur le data-frame Immatriculation sur R.	39
71	Fusion des deux data-frames sur R.	39
72	Suppression de la variable immatriculation sur R.	40
73	Suppression des différentes variables sur le data-frame sur R.	40
74	Refactorisation des variable sur le data-frame sur R.	41
75	Création du premier jeu de données sur sur R.	41
76	Suppression de la colonne taux sur le data-frame sur R.	42
77	Création du second jeu de données sur sur R.	42
78	Répartition du taux.	42
79	Suppression de la colonne taux sur le data-frame sur R.	43
80	Création de la variable sur les data-frames sur R.	43
81	Répartition des catégories de taux.	44
82	Répartition des catégories de taux.	44
83	Importation des packages sur R.	45
84	Apprentissage du classifieur C5.0.	46
85	Prédiction du classifieur C5.0.	46
86	Matrice de confusion du classifieur C5.0.	46
87	Calcul de la valeur de l'auc du classifieur C5.0.	47
88	Utilisation de la fonction « multiclass.roc » pour le classifieur C5.0.	47
89	Apprentissage du classifieur Naive-Bayes.	48
90	Prédiction du classifieur Naive-Bayes.	48
91	Matrice de confusion du classifieur Naive-Bayes.	48
92	Utilisation de la fonction « multiclass.roc » pour le classifieur Naive-Bayes.	48

93	Matrice de confusion du classifieur SVM.	49
94	Utilisation de la fonction « multiclass.roc » pour le classifieur SVM.	49
95	Matrice de confusion du classifieur Kknn.	50
96	Utilisation de la fonction « multiclass.roc » pour le classifieur Kknn.	50
97	Matrice de confusion du classifieur Nnet.	51
98	Utilisation de la fonction « multiclass.roc » pour le classifieur Nnet.	51
99	Suppression de la variable age pour le classifieur randomForest.	52
100	Matrice de confusion du classifieur randomForest.	52
101	Utilisation de la fonction « multiclass.roc » pour le classifieur randomForest.	52
102	Matrice de confusion du classifieur C5.0.	54
103	Utilisation de la fonction « multiclass.roc » pour le classifieur C5.0.	54
104	Matrice de confusion du classifieur Naive Bayes.	55
105	Utilisation de la fonction « multiclass.roc » pour le classifieur Naive Bayes.	55
106	Matrice de confusion du classifieur SVM.	56
107	Utilisation de la fonction « multiclass.roc » pour le classifieur SVM.	56
108	Matrice de confusion du classifieur Kknn.	57
109	Utilisation de la fonction « multiclass.roc » pour le classifieur Kknn.	57
110	Matrice de confusion du classifieur Nnet.	58
111	Utilisation de la fonction « multiclass.roc » pour le classifieur Nnet.	58
112	Matrice de confusion du classifieur randomForest.	59
113	Utilisation de la fonction « multiclass.roc » pour le classifieur randomForest.	59
114	Création des deux ensembles sur le data-frame sur R.	61
115	Apprentissage avec le classifieur C5.0 sur le data-frame sur R.	61
116	Modification du data-frame sur R.	61
117	Réalisation de la prédiction sur le data-frame sur R.	62
118	Création du data-frame résultat.	62
119	Affichage du data-frame resultat1 avec la catégorie prédite.	62
120	Enregistrement des prédictions sur un fichier .csv.	62
121	Programme principal pour lancer le Map Reduce.	65
122	Programme pour réaliser la fonction Map.	65
123	Création du couple Key Value.	65
124	Programme pour réaliser la fonction Reduce.	66
125	Modification de la Value pour chaque Key.	66
126	Fichiers Java disponible sur la machine virtuelle.	67
127	Compilation des fichiers Java.	67
128	Création des fichiers d'exécution pour MapReduce.	67
129	Fichier d'entrée pour MapReduce sur HDFS.	68
130	Fichier d'entrée pour MapReduce sur HDFS.	68
131	Resultat du MapReduce sur HDFS.	69
132	Export du fichier resultat.	69
133	Fichier resultat exporté.	69
134	Commande PS pour convertir le fichier txt en csv.	70
135	Commande PS pour modifier les lignes sans les quotes.	70
136	Importation des fichiers dans R Studio.	71
137	Modification des colonnes marque.	71
138	Fusion des deux data-frames.	71
139	Nettoyage du data-frame.	71
140	Affichage du data-frame avec les nouvelles colonnes.	72
141	Export du data-frame en fichier .csv.	72
142	Connexion à la base de données.	73

143	Importation des données sur R.	73
144	Visualisation des données sur R.	74
145	Suppression des datas suivants l'age.	74
146	Modification des datas suivants le sexe.	74
147	Suppression des datas suivants le sexe.	74
148	Suppression des datas suivants le taux.	74
149	Suppression des datas suivants la situation familiale.	75
150	Suppression des datas suivants le nombre d'enfants à charge.	75
151	Suppression des datas suivants le nombre de voitures.	75
152	Commandes pour prédire la catégorie avec le classifieur C5.0.	76
153	Affichage du data-frame resultat11 avec la catégorie prédite.	76

Liste des tableaux

1	Récapitulatif des catégories de véhicules suivant les critères.	39
2	Récapitulatif des catégories de taux en fonction des bornes.	43
3	Calculs des différents taux du classifieur C5.0.	46
4	Calculs des différents taux du classifieur Naive-Bayes.	48
5	Calculs des différents taux du classifieur SVM.	49
6	Calculs des différents taux du classifieur Kknn.	50
7	Calculs des différents taux du classifieur Nnet.	51
8	Calculs des différents taux du classifieur randomForest.	52
9	Récapitulatif des classifieurs avec les deux taux.	53
10	Calculs des différents taux du classifieur C5.0.	54
11	Calculs des différents taux du classifieur Naive Bayes.	55
12	Calculs des différents taux du classifieur SVM.	56
13	Calculs des différents taux du classifieur Kknn.	57
14	Calculs des différents taux du classifieur Nnet.	58
15	Calculs des différents taux du classifieur randomForest.	59
16	Récapitulatif des classifieurs avec les deux taux.	60