

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

Nom du membre du Groupe 6 :

Gao LIU

Pierric SCHOENDORF

Yimin CAO

Ziheng WANG

Table des matières

Partie DATALAKE	7
1. L'objectif :	7
2. Le chargement de sources de données sur la base MongoDB	9
3. Le chargement de sources de données sur Hadoop HDFS	12
4. Le chargement de sources de données sur la base Oracle NoSQL	15
4.1 Fichier Catalogue.java	15
4.2 Fichier Marketing.java	17
4.3 Table Interne sur Hive	17
5. Conclusion data Lake	21
Partie HADOOP MAP REDUCE	22
1. Objectif	22
2. Scripts java	22
2.1 Class Driver	22
2.2 Class Map	23
2.3 Class REDUCE	23
3. Exécution du programme Map Reduce	24
4. Résultat du programme	25
5. Export du fichier	25
6. Jonction entre Catalogue et Co2	25
6.1 Import des fichiers	25
6.2 Modification des data-frames	25
6.3 Jointure des data-frames	26
6.4 Complétion des données dans le data-frame	26
6.5 Nettoyage du data-frame	27
7. Résultat	27
Partie Machine Learning	28
1. RStudio et langage R	28
2. Nettoyage des données	30
2.1 Nettoyage des données : Clients	30
2.2 Nettoyage des données : Catalogue	31
2.3 Nettoyage des données : Immatriculations	33
3. Analyse exploratoire des données vues	34
4. Identification des catégories de véhicules et application aux Immatriculations	39
5. Fusion des données Clients et Immatriculations	40
6. Modèle de classification et les tests dans différents algorithmes	41

6.1	Situation 1 : normal	41
6.1.1	<i>Algorithme : C50</i>	41
6.1.2	<i>Algorithme : naivebayes</i>	42
6.1.3	<i>Algorithme : Random Forest</i>	43
6.1.4	<i>Algorithme : NNET</i>	43
6.1.5	<i>Algorithme : Régression logistique multinomiale</i>	44
6.2	Situation 2 : supprimer la colonne taux	44
6.2.1	<i>Algorithme : C50</i>	44
6.2.2	<i>Algorithme : Naivebayes</i>	45
6.2.3	<i>Algorithme : Random Forest</i>	45
6.2.4	<i>Algorithme : NNET</i>	46
6.3	Situation 3 : supprimer la colonne taux et la colonne âge	47
6.3.1	<i>Algorithme : C50</i>	47
6.3.2	<i>Algorithme : Naviebayes</i>	48
6.3.3	<i>Algorithme : Random Forest</i>	49
6.3.4	<i>Algorithme : NNET</i>	50
6.4	Situation 4 : remplacer la colonne taux à la colonne taux_classe	51
6.4.1	<i>Algorithme : C50</i>	52
6.4.2	<i>Algorithme : Naviebayes</i>	53
6.4.3	<i>Algorithme : Random Forest</i>	53
6.4.4	<i>Algorithme : NNET</i>	54
6.5	Situation 5 : remplacer la colonne taux et la colonne âge à la colonne taux_classe et âge_classe	55
6.5.1	<i>Algorithme : C50</i>	56
6.5.2	<i>Algorithme : Naviebayes</i>	56
6.5.3	<i>Algorithme : Random Forest</i>	57
6.5.4	<i>Algorithme : NNET</i>	58
7.	Conclusion des quatre algorithmes	59
8.	Nettoyage de Marketing	60
9.	Prédiction	61

Table des figures

Figure 1 Architecture de Datalake	8
Figure 2 La création d'une database dans MongoDB	9
Figure 3 La création d'une collection Client	9
Figure 4 Importer des fichier csv dans le VM	9
Figure 5 Importer le fichier client_5.csv dans la collection Client	9
Figure 6 Importer le fichier client_13.csv dans la collection Client	10
Figure 7 Exporter la collection Client dans un fichier Clientsmongodb.csv	10
Figure 8 Créer une table Client sur Hive	10
Figure 9 Charger les données dans la table Client crée	10
Figure 10 Vérifier la présence de donnée de Client	11
Figure 11 Déposer le fichier Immatriculation.csv sur HDFS	12
Figure 12 Déposer le fichier CO2.csv sur HDFS	12
Figure 13 Créer une table externe Immatriculation sur Hive	13
Figure 14 Créer une table externe CO2 sur Hive	13
Figure 15 Lists des tables externes créées sur Hive	14
Figure 16 Vérifier la présence des données dans la table Immatriculation	14
Figure 17 Vérifier la présence des données dans la table CO2	14
Figure 18 La fonction importer les données Catalogue dans Oracle NoSQL	15
Figure 19 Les commande à exécuter le fichier Ctalogue.java pour importer des données	16
Figure 20 Lancer le service KV Store	16
Figure 21 Vérifier la présence de la table Catalogue sur Oracle NoSQL	16
Figure 22 La fonction importer les données Marketing dans Oracle NoSQL	17
Figure 23 Les commande à exécuter le fichier Marketing.java pour importer des données	17
Figure 24 Vérifier la présence de la table Marketing sur Oracle NoSQL	17
Figure 25 Créer une table Marketing_ext sur Hive	18
Figure 26 Vérifier la présence des données dans la table externe Marketing_ext	18
Figure 27 Créer une table interne Marketing à partir de la table externe Marketing_ext	19
Figure 28 Vérifier la présence des données dans la table interne Marketing	19
Figure 29 Créer la table externe Catalogue_ext sur Hive	19
Figure 30 Vérifier la présence des données dans la table externe Catalogue_ext	20
Figure 31 Créer une table interne Catalogue à partir de la table externe Catalogue_ext	20
Figure 32 Vérifier la présence des données dans la table interne Catalogue	20
Figure 33 Liste des tales dans Data Lake	21
Figure 34 Class Driver du programme Map Reduce	22
Figure 35 Fonction Map du programme	23
Figure 36 Calcul des moyennes dans la fonction Reduce	23
Figure 37 Fichier java dans la vm	24
Figure 38 Compilation des fichier java et création du .Jar	24
Figure 39 Résultat de l'exécution du programme	25
Figure 40 Import des fichiers dans R	25
Figure 41 Modification des data-frames	25
Figure 42 Jointure des deux data-frames	26
Figure 43 Ajout des moyennes	26
Figure 44 Moyenne des colonnes de CO2	26
Figure 45 Nettoyage du data-frame	27
Figure 46 Résultat de la jonction des deux fichiers	27
Figure 47 Connexion entre Hive et RStudio pour l'importation de tables	28

Figure 48 Récapitulatif des Clients	29
Figure 49 Récapitulatif des Immatriculations	29
Figure 50 Récapitulatif des Marketing	29
Figure 51 Récapitulatif des Catalogue	29
Figure 52 Récapitulatif des Co2	29
Figure 53 Changement du nom et suppression de nulles	30
Figure 54 Respect de la contrainte de l'âge	30
Figure 55 Corrige du nom de colonne et valeur erronées	30
Figure 56 Respect de la contrainte de taux	30
Figure 57 Corrige du nom de colonne de client	31
Figure 58 Respect de la contrainte de nbEnfantsacharge	31
Figure 59 Suppression de double	31
Figure 60 Récapitulatif du tableau des Clients nettoyés	31
Figure 61 Renommer les colonnes	31
Figure 62 Convert to format demandé	32
Figure 63 Suppression de nulles	32
Figure 64 Nettoyage de caractère incorrect	32
Figure 65 Récapitulatif du tableau du Catalogue nettoyé	32
Figure 66 Corrige du nom de colonne	33
Figure 67 Convert to format demandé	33
Figure 68 Suppression ID, convert format, suppression double, et corrige valeur erronée	33
Figure 69 Récapitulatif du tableau des Immatriculations nettoyés	33
Figure 70 Code de dessin de couleur de catalogue	34
Figure 71 La répartition de la variable couleur	34
Figure 72 Code de dessin de longueur de catalogue	34
Figure 73 La répartition de la variable longueur	34
Figure 74 Code de dessin de nbPlaces de catalogue	35
Figure 75 La répartition de la variable nbPlaces	35
Figure 76 Code de dessin de nbPortes de catalogue	35
Figure 77 La répartition de la variable nbPortes	35
Figure 78 Code de dessin de puissance de catalogue	36
Figure 79 La répartition de la variable puissance	36
Figure 80 La répartition de la variable nbPortes de chaque type de longueur	36
Figure 81 La répartition de la variable nbPlaces de chaque type de longueur	37
Figure 82 La répartition de la variable puissance de chaque type de longueur	38
Figure 83 Categories à partir de Catalogue	39
Figure 84 Application de catégories aux Immatriculations	39
Figure 85 Fusion de data frames et suppression de colonnes inutiles	40
Figure 86 Convert to format	41
Figure 87 Situation1_Séparation de l'échantillonnage d'apprentissage et test	41
Figure 88 Formation de jeux d'apprentissage avec C5.0	41
Figure 89 Prédiction de de jeu du test avec C50 et affichage des résultats prédis	41
Figure 90 Situation1_C50 : Dessin de la matrice de mélange	42
Figure 91 Situation1_C50 : calcul de l'AUC	42
Figure 92 Formation de jeux d'apprentissage avec naivebayes	42
Figure 93 Prédiction de de jeu du test avec naivebayes	42
Figure 94 Situation1_naivebayes : Dessin de la matrice de mélange	43
Figure 95 Situation1_Naivebayes : calcul de l'AUC	43

Figure 96 Formation de jeux d'apprentissage avec random forest	43
Figure 97 Formation de jeux d'apprentissage avec NNET	43
Figure 98 Formation de jeux d'apprentissage avec Régression logistique multinomiale	44
Figure 99 Situation2_Séparation de l'échantillonnage d'apprentissage et test.....	44
Figure 100 Situation2_C50 résultat de prédiction	44
Figure 101 Situation2_C50 : calcul de l'AUC.....	44
Figure 102 Situation2_Naivebayes résultat de prédiction	45
Figure 103 Situation2_Naivebayes: calcul de l'AUC	45
Figure 104 Formation de jeux d'apprentissage avec random forest	45
Figure 105 Formation de jeux d'apprentissage avec NNET	46
Figure 106 Prédiction de de jeu du test avec NNET	46
Figure 107 Situation2_NNET : Dessin de la matrice de mélange	46
Figure 108 Situation2_NNET : calcul de l'AUC	47
Figure 109 Situation3_Séparation de l'échantillonnage d'apprentissage et test	47
Figure 110 Situation3_C50 résultat de prédiction	47
Figure 111 Situation3_C50 : calcul de l'AUC.....	48
Figure 112 Situation3_Naivebayes résultat de prédiction	48
Figure 113 Situation3_Naivebayes : calcul de l'AUC	48
Figure 114 Situation3_Random Forest résultat de prédiction	49
Figure 115 Situation3_Random Forest : calcul de l'AUC	49
Figure 116 Situation3_NNET résultat de prédiction	50
Figure 117 Situation3_NNET : calcul de l'AUC	50
Figure 118 Le boxplot de taux	51
Figure 119 Récapitulatif du taux	51
Figure 120 Récapitulatif du Clients_Immatriculations_traite	51
Figure 121 Situation4_Séparation de l'échantillonnage d'apprentissage et test	51
Figure 122 Situation4_C50 résultat de prédiction	52
Figure 123 Situation4_C50 : calcul de l'AUC	52
Figure 124 Situation4_Naivebayes : résultat de prediction	53
Figure 125 Situation4_Naivebayes : calcul de l'AUC	53
Figure 126 Formation de jeux d'apprentissage avec random forest	53
Figure 127 Situation4_NNET résultat de prédiction	54
Figure 128 Situation4_NNET : calcul de l'AUC	54
Figure 129 Le boxplot d'âge	55
Figure 130 Récapitulatif d'âge	55
Figure 131 Récapitulatif du Clients_Immatriculations_traite	55
Figure 132 Situation5_Séparation de l'échantillonnage d'apprentissage et test	56
Figure 133 Situation5_C50 code	56
Figure 134 Situation5_Naivebayes résultat de prédiction	56
Figure 135 Situation5_Naivebayes : calcul de l'AUC	57
Figure 136 Situation5_Random Forest résultat de prédiction	57
Figure 137 Situation5_Random Forest : calcul de l'AUC	57
Figure 138 Situation5_NNET résultat de prédiction	58
Figure 139 Situation5_NNET : calcul de l'AUC	58
Figure 140 Renommer des colonnes de Marketing	60
Figure 141 Convert to format demandé	60
Figure 142 Création de class pour taux, corrige de valeurs, et convert to format demandé	60
Figure 143 Récapitulatif du tableau des Immatriculations nettoyés	60

Figure 144 Prédiction de Marketing avec NNET	61
Figure 145 Prédiction finale de Marketing	61

Liste des tableaux

Tableau 1 Situation1_C50 : Calcul du rappel, de la précision et des taux d'erreur	42
Tableau 2 Situation1_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur	43
Tableau 3 Situation2_C50 : Calcul du rappel, de la précision et des taux d'erreur	44
Tableau 4 Situation2_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur	45
Tableau 5 Situation2_NNET : Calcul du rappel, de la précision et des taux d'erreur	46
Tableau 6 Situation3_C50 : Calcul du rappel, de la précision et des taux d'erreur	47
Tableau 7 Situation3_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur	48
Tableau 8 Situation3_Random Forest : Calcul du rappel, de la précision et des taux d'erreur	49
Tableau 9 Situation3_NNET : Calcul du rappel, de la précision et des taux d'erreur	50
Tableau 10 Situation4_C50 : Calcul du rappel, de la précision et des taux d'erreur	52
Tableau 11 Situation4_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur	53
Tableau 12 Situation4_NNET : Calcul du rappel, de la précision et des taux d'erreur	54
Tableau 13 Situation5_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur	56
Tableau 14 Situation5_Random Forset : Calcul du rappel, de la précision et des taux d'erreur	57
Tableau 15 Situation5_NNET : Calcul du rappel, de la précision et des taux d'erreur	58
Tableau 16 Comparaison de l'AUC	59
Tableau 17 Comparaison des taux d'erreur	59

Partie DATALAKE

1. L'objectif :

Nous avons été sollicités par un concessionnaire automobile qui souhaite améliorer la pertinence de sa stratégie de vente en ciblant mieux les besoins des clients potentiels.

Au cours de l'entretien avec le responsable de la concession, nous avons pu clarifier le contexte et les objectifs de notre mission. Les différents types de véhicules proposés répondent à des besoins variés : certains sont conçus pour la conduite en ville, d'autres sont plus spacieux pour les familles, tandis que d'autres encore s'adressent à une clientèle aisée en quête de puissance.

L'objectif est donc de catégoriser ces véhicules en fonction des désirs des clients pour proposer des modèles qui correspondent le mieux à leurs attentes. Pour cela, nous allons utiliser différentes méthodes de classification supervisée afin de créer un modèle de prédiction. Ce dernier sera basé sur les caractéristiques du client (telles que l'âge, le nombre d'enfants, etc.)

Ainsi que sur les immatriculations effectuées cette année. Nous disposerons de six fichiers fournis par le concessionnaire :

Immatriculation.csv,

Client_5.csv et Client_13.csv,

Catalogue.csv

Marketing.csv

CO2.csv

Pour récupérer et traiter toutes ces données, nous mettrons en place différentes méthodes pour ces fichiers. Nous devrons donc charger les sources de données sur la base MongoDB et Oracle NoSQL, ainsi que via Hadoop HDFS. Nous créerons également des tables externes sur Hive pour visualiser des données sur Hadoop par exemple. Nous créons notre data Lake en basant sur Hive, on importer toutes les données de différentes sources dans Hive pour faciliter la requête sur ces données. Enfin, nous faisons la connexion entre Hive et R pour récupérer et afficher les données dans RStudio afin d'analyse en temps réel. Ce projet nécessitera donc la mise en œuvre de techniques de Big Data et de Machine Learning.

La gestion des données est une étape cruciale pour mener à bien ce projet. Nous allons donc mettre en place plusieurs méthodes pour charger les différentes sources de données à notre disposition. Voici les étapes que nous allons suivre :

- 1.Chargement des données du fichier Client_5.csv et Client_13.csv sur la base MongoDB.
- 2.Chargement des données des fichiers Catalogue.csv et Marketing.csv sur la base Oracle NoSQL.
- 3.Chargement des données du fichier Immatriculation.csv et CO2.csv via Hadoop HDFS.
- 4.Création de tables externes sur Hive à partir de l'Oracle NoSQL ou de HDFS.
- 5.Importer le fichier Client_5.csv et Client_13.csv sur Hive par créer le table interne.

6. Enfin, nous faisons la connexion entre Hive et R pour récupérer et afficher les données dans RStudio afin de procéder à leur analyse. L'image ci-dessous est l'architecture globale de notre Data Lake.

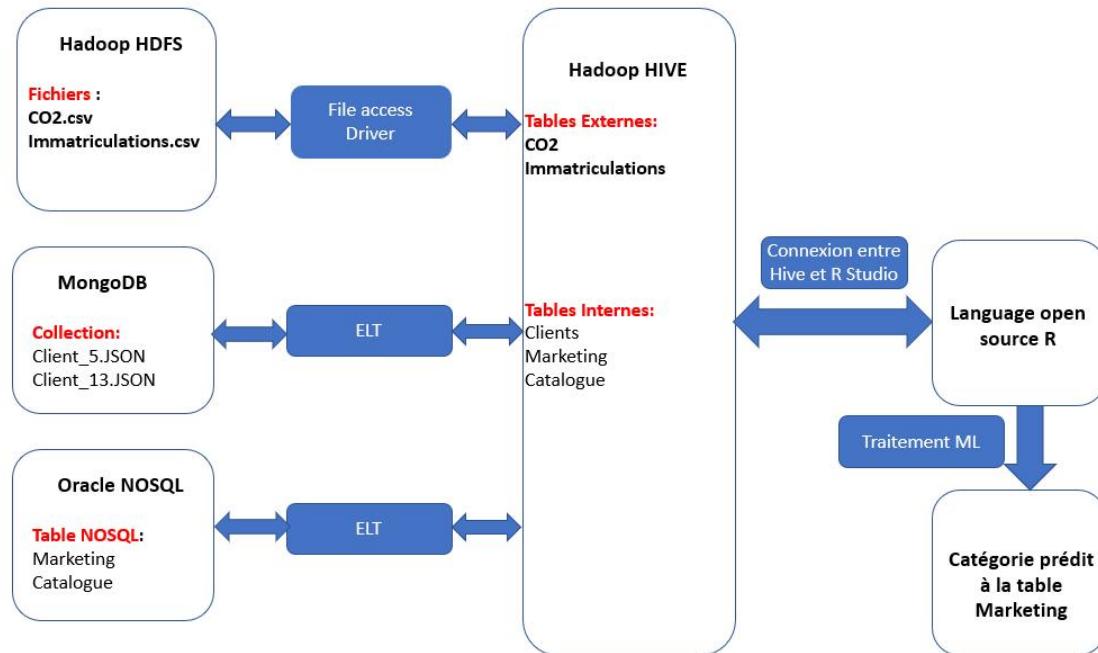


Figure 1 Architecture de Datalake

2. Le chargement de sources de données sur la base MongoDB

Créer un database par la commande 'use concessionnaire'

```
> show dbs
admin          0.000GB
concessionnaire 0.010GB
config          0.000GB
local           0.000GB
test            0.000GB
> db
test
> use concessionnaire
switched to db concessionnaire
> db
concessionnaire
```

Figure 2 La création d'une database dans MongoDB

Après on crée une collection Client par la commande ci-dessous

```
db.createCollection("client")
```

```
> db
concessionnaire
> show collections
client
```

Figure 3 La création d'une collection Client

Avant d'importer le fichier client dans mongodb on va mettre nos 6 fichier csv dans le chemin ci-dessous

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ pwd
/home/vagrant/Projet_Big_Data
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ ls
Catalogue.csv   Clients_5.csv   Immatriculations.csv  nohup.out  voiture
Clients_13.csv   CO2.csv       Marketing.csv        README.md
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ |
```

Figure 4 Importer des fichier csv dans le VM

Pour importer le fichier Client_5.csv et Client13.csv dans la collection client, on doit utiliser l'outil mongoimport :

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ mongoimport -d concessionnaire -c
clients --type csv --file "/home/vagrant/Projet_Big_Data/Clients_5.csv" --header
line
2023-03-02T19:02:41.316+0100      connected to: mongodb://localhost/
2023-03-02T19:02:42.558+0100      100000 document(s) imported successfully. 0 docu
ment(s) failed to import.
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ |
```

Figure 5 Importer le fichier client_5.csv dans la collection Client

Pareil pour le fichier client_13.

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ mongoimport -d concessionnaire -c clients --type csv --file "/home/vagrant/Projet_Big_Data/Clients_13.csv" --headerline  
2023-03-02T19:04:16.796+0100      connected to: mongodb://localhost/  
2023-03-02T19:04:18.037+0100      100000 document(s) imported successfully. 0 document(s) failed to import.  
[vagrant@oracle-21c-vagrant Projet_Big_Data]$
```

Figure 6 Importer le fichier client_13.csv dans la collection Client

Après on importe les fichier client dans mongodb, on a une collection client qui contient 20000 lignes de client, donc pour créer notre datalake, je vais exporter la collection client vers un fichier client.csv, après je crée un table client sur hive, et importer le fichier csv dans cette table.

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ mongoexport --db=concessionnaire --collection=client --type=csv --fields=age,sexe,taux,situationFamiliale,nbEnfantsACharge,2emevoiture,immatriculation --out=./Clientsmongodb.csv
2023-03-02T19:09:10.934+0100      connected to: mongodb://localhost/
2023-03-02T19:09:11.935+0100      [#####] concessionnaire.client 176000/200000 (88.0%)
2023-03-02T19:09:12.015+0100      [#####] concessionnaire.client 200000/200000 (100.0%)
2023-03-02T19:09:12.015+0100      exported 200000 records
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ ls
Catalogue.csv  Clientsmongodb.csv  Marketing.csv  voiture
clients_13.csv  CO2.csv           nohup.out
Clients_5.csv   Immatriculations.csv  README.md
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ |
```

Figure 7 Exporter la collection Client dans un fichier Clientsmongodb.csv

J'exporte la collection client vers le fichier Clientsmongodb.csv, ce fichier contient 20000 lignes de client.

Après je vais me connecter sur Hive pour créer une table client :

```
CREATE TABLE if not exists client (age INT, sexe STRING, taux INT, situationFamiliale STRING, nbEnfantsAcharge INT, 2emevoiture STRING, immatriculation STRING)
```

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ''

LINES TERMINATED BY '\n':

Figure 8 Crée une table Client sur Hive

Pour importer le fichier clientmondb.csv dans la table client sur Hive, on doit d'abord mets le fichier sur HDFS, après on vas utiliser la commande load.

```
0: jdbc:hive2://localhost:10000> load data inpath '/client/Clientsmongodb.csv' overw  
rite into table client;
```

Figure 9 Charger les données dans la table Client créée

Après on va vérifier sur hive

Select * from client ;

client.age	client.sexé	client.taux enfantsacharge	client.Zemevoiture	client.situationfamiliale	client.nb immatriculation
28	Masculin	744		En Couple	2
59	F	1163		Celibataire	0
79	M	478		Celibataire	0
32	M	519		Seul	0
54	M	496		En Couple	1
41	F	429		En Couple	1

200,001 rows selected (15.317 seconds)					
0: jdbc:hive2://localhost:10000>					

Figure 10.Vérifier la présence de donnée de Client

3. Le chargement de sources de données sur Hadoop HDFS

Nous allons maintenant charger deux nouveaux fichiers, le Immatriculations.csv et CO2.csv, en utilisant une méthode différente. Nous avons choisi de stocker les données sur Hadoop HDFS, un Framework Java open source conçu pour stocker et traiter des big data. Cette solution est particulièrement adaptée à notre projet pour plusieurs raisons :

Résilience : les données sont stockées sur des nœuds du cluster et sont répliquées sur d'autres nœuds, garantissant ainsi la tolérance aux incidents. Si un nœud tombe en panne, les autres serveurs du cluster disposent toujours d'une copie de sauvegarde des données.

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.

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.

Vitesse : le système de fichiers distribué, les traitements concurrents et le modèle MapReduce permettent d'exécuter les requêtes les plus complexes en quelques secondes.

Diversité des données : le HDFS peut stocker différents formats de données, qu'elles soient structurées, non structurées (comme des vidéos) ou semi-structurées (comme des fichiers XML). Il n'est pas nécessaire de valider les données par rapport à un schéma prédéfini lors de leur stockage, ce qui permet de télécharger des données sous n'importe quel format. Lors de leur récupération, les données sont analysées et utilisées en appliquant le ou les schémas requis.

Créer un dossier sur HDFS qui s'appelle Datalake_Immatriculation pour stocker le fichier Immatriculation.csv

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -mkdir /Datalake_Immatri
culation
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -put home/vagrant/Projet
_Big_Data/Immatriculation.csv /Datalake_Immatri
cation
put: `home/vagrant/Projet_Big_Data/Immatriculation.csv': No such file or directo
ry
[vagrant@oracle-21c-vagrant Projet_Big_Data]$
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -put /home/vagrant/Proje
t_Big_Data/Immatriculation.csv /Datalake_Immatri
cation
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -ls /Datalake_Immatri
cation
Found 1 items
-rw-r--r-- 1 vagrant supergroup      14114 2023-03-11 00:10 /Datalake_Immatri
cation/Immatriculation.csv
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ |
```

Figure 11 Déposer le fichier Immatriculation.csv sur HDFS

Pareil on va créer un dossier sur HDFS qui s'appelle Datalake_CO2 pour stocker le fichier CO2.csv

```
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -mkdir /Datalake_CO2
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -put ./CO2.csv /Datalake_
_CO2
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ hadoop fs -ls /Datalake_CO2
Found 1 items
-rw-r--r-- 1 vagrant supergroup      38916 2023-03-03 19:31 /Datalake_CO2/CO2.
csv
[vagrant@oracle-21c-vagrant Projet_Big_Data]$ |
```

Figure 12 Déposer le fichier CO2.csv sur HDFS

Après je vais créer une table externe sur Hive

Apache Hive est un datawarehouse pour Hadoop qui permet de proposer une abstraction au-dessus de MapReduce pour faciliter l'analyse de gros volumes de données. L'un des avantages de Hive est qu'il permet de définir une structure avec une variété de formats de données, ce qui facilite la requête de ces données et le travail analytique. Il est donc particulièrement adapté à un contexte d'analyse de données. En outre, Hive propose une fonction de stockage distribué et permet d'accéder aux fichiers stockés dans HDFS.

Il est important de noter que Hive ne fonctionne ni comme une base de données relationnelle, ni comme un datawarehouse classique. En effet, il s'agit d'un système qui maintient des métadonnées décrivant les données stockées dans HDFS. Pour assurer la persistance des métadonnées, Hive utilise une base de données relationnelle appelée metastore.

Afin de pouvoir travailler avec Hive, il est nécessaire de créer des tables externes qui pointent respectivement vers HDFS pour la table catalogue et CO2.

D'abord on va créer une table externe CATALOGUE pour le fichier catalogue.csv stocké sur HDFS

```
drop table Immatriculation
CREATE EXTERNAL TABLE Immatriculation (
    IMMATRICULATION STRING,
    MARQUE STRING,
    NOM STRING,
    PUISSANCE INT,
    LONGUEUR STRING,
    NBPLACES INT,
    NBPORTES INT,
    COULEUR STRING,
    OCCASION BOOLEAN,
    PRIX INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 'hdfs:/Datalake_Immatriculation';
```

Figure 13 Créer une table externe Immatriculation sur Hive

Après on crée une table externe CO2 pour le fichier co2.csv stocké sur HDFS

```
drop table CO2;
CREATE EXTERNAL TABLE CO2 (
    ordre int,
    MARQUEModele string,
    BonusMalus string ,
    RejetsCO2 int,
    Coutenerie string)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION 'hdfs:/Datalake_CO2';
```

Figure 14 Créer une table externe CO2 sur Hive

On exécute ces deux commandes sur Hive

Après on va obtenir deux tables comme ci-dessous.

```
23/03/10 18:08:49 INFO ql.Driver: Completed executing command(queryId=vagrant_230310180848_1e1e0485-e1ed-45cd-a6dc-845034c29d0c); Time taken: 0.141 seconds
23/03/10 18:08:49 INFO ql.Driver: OK
+-----+
| tab_name |
+-----+
| catalogue |
| client |
| co2 |
| immatriculation |
+-----+
4 rows selected (0.207 seconds)
```

Figure 15 Lists des tables externes créées sur Hive

Après je vais vérifier la présence de donnée pour le table externe Immatriculation et co2

```
+-----+-----+-----+
| immatriculation.immatriculation | immatriculation.marque | immatriculation.n
| nom | immatriculation.puissance | immatriculation.longueur | immatriculation.n
| bplaces | immatriculation.nbportes | immatriculation.couleur | immatriculatio
| n.occasion | immatriculation.prix |
+-----+-----+-----+
+-----+-----+-----+
| 6343 VT 29 | Renault | Vel Satis 3.5 V6
| 245 | très longue | 5
| 5 | bleu | true
| 3184 BU 69 | Mercedes | 5500
| 306 | très longue | 5
| 5 | blanc | false
| 4725 UR 96 | Skoda | Superb 2.8 V6
| 193 | très longue | 5
| 5 | bleu | false
| 788 FH 14 | Saab | 9.3 1.8T
| 150 | longue | 5
+-----+-----+-----+
```

Figure 16 Vérifier la présence des données dans la table Immatriculation

```
+-----+-----+-----+
| co2.ordre | co2.marquemodele | co2.bonusmalus | co2
| rejetsco2 | co2.coutenerie | |
+-----+-----+-----+
+-----+-----+-----+
| NULL | Marque / Modele | Bonus / Malus | NUL
| | Cout enerie | |
| 2 | AUDI E-TRON SPORTBACK 55 (408ch) quattro | -6 000€ 1 | 0
| | 319 € | |
| 3 | AUDI E-TRON SPORTBACK 50 (313ch) quattro | -6 000€ 1 | 0
| | 356 € | |
| 4 | AUDI E-TRON 55 (408ch) quattro | -6 000€ 1 | 0
| | 357 € | |
| 5 | AUDI E-TRON 50 (313ch) quattro | -6 000€ 1 | 0
| | 356 € | |
| 6 | BMW i3 120 Ah | -6 000€ 1 | 0
| | 204 € | |
| 7 | BMW i3s 120 Ah | -6 000€ 1 | 0
| | 204 € | |
+-----+-----+-----+
```

Figure 17 Vérifier la présence des données dans la table CO2

4. Le chargement de sources de données sur la base Oracle NoSQL

Après avoir utilisé la base MongoDB et HDFS, nous nous tournons vers Oracle NoSQL, une autre base de données NoSQL. Cette base de données offre aux développeurs la possibilité de créer facilement des applications en utilisant des modèles de bases de données de documents, de colonnes et de clés-valeurs, ce qui permet d'obtenir des temps de réponse très rapides (de l'ordre de la milliseconde). Le terme NoSQL signifie "not only SQL" et désigne un modèle de base de données qui veut enrichir et compléter de manière utile les bases de données SQL relationnelles traditionnelles. En effet, les bases de données NoSQL dépassent les limites des systèmes relationnels en exploitant un modèle de base de données différent. Cependant, cela ne signifie pas qu'aucun système SQL n'est utilisé. Contrairement aux bases de données SQL relationnelles, les bases de données NoSQL n'utilisent pas de lignes et de colonnes pour stocker les données. Elles organisent les gros volumes de données de manière flexible, par exemple sous forme de documents, de paires de valeurs ou de colonnes. Les systèmes NoSQL sont donc parfaitement adaptés aux applications qui nécessitent le traitement de larges volumes de données.

Dans notre projet, nous allons charger deux fichiers, Catalogue.csv et Marketing.csv, dans Oracle NoSQL. Cette étape nous permettra de profiter des avantages offerts par cette base de données NoSQL pour manipuler efficacement ces données.

4.1 Fichier Catalogue.java

Pour importer un fichier CSV dans une table Oracle NoSQL, on peut utiliser l'interface de ligne de commande (CLI) ou l'API de votre langage de programmation préféré ici on choisit java.

On écrit un fichier Catalogue.java pour importer les données sur kvstore.

Ci-dessous c'est la fonction permettant l'importation du fichier Catalogue.csv sur Oracle NoSQL.

```
private void insertACatalogueRow(String marque, String nom, String puissance, String longueur, String nbPlaces, String nbPortes, String couleur, String occasion, String prix) {
    //TableAPI tableAPI = store.getTableAPI();
    StatementResult result = null;
    String statement = null;
    System.out.println("***** Dans : insertACatalogueRow *****");
    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 CatalogueTable = tableH.getTable(tabCatalogue);
        // Get a Row instance
        Row CatalogueRow = CatalogueTable.createRow();
        // Now put all of the cells in the row.
        // This does NOT actually write the data to
        // the store.

        // Create one row
        CatalogueRow.put("catalogueID", catalogueID);
        CatalogueRow.put("marque", marque);
        CatalogueRow.put("nom", nom);
        CatalogueRow.put("puissance", puissance);
        CatalogueRow.put("longueur", longueur);
        CatalogueRow.put("nbPlaces", nbPlaces);
        CatalogueRow.put("nbPortes", nbPortes);
        CatalogueRow.put("couleur", couleur);
        CatalogueRow.put("occasion", occasion);
        CatalogueRow.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(CatalogueRow, null, null);
        catalogueID++;
    } 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 18 La fonction importer les données Catalogue dans Oracle NoSQL

On écrit d'abord ce fichier et après on va mettre ce fichier Catalogue.java dans le chemin
/home/vagrant/Projet_Big_Data/voiture

Après on stocker ce fichier java sur le vs, on va exécuter ce script java pour charger les données sur Oracle NoSQL.

```
[Asdrisutgorsje-S1c-Agdrisun -]à [Svls -Xwxs2ew -Xws2ew -cb $KHOME\J\p\KACJ\enf.let:éMABEOLCEHOME\lotfni.Çafajodne
--executer je cjsas tmmatcijpjeous.cjsas
[Asdrisutgorsje-S1c-Agdrisun lotfni.let:éSvls -d -cb $KHOME\J\p\KACJ\enf.let:éMABEOLCEHOME\lotfni.Çafajodne.Svls
--combyjjet je tscyot tmmatcijpjeous.cjsas
[Asdrisutgorsje-S1c-Agdrisun lotfni.let:éSvls -exbotr MABEOLCEHOME=\nowe\Agdrisun\btolof-Bid-Dafra
--exbotr je cyswun
```

Figure 19 Les commandes à exécuter le fichier Catalogue.java pour importer des données

On va lancer la service Oracle NoSQL pour vérifier si on a réussi à importer les données sur kvstore

```
--Start KVStore using KVLite utility
nohup java -Xmx256m -Xms256m -jar $KHOME/lib/kvstore.jar kvlite --secure-config disable -root $KVROOT > kvstore.log 2>&1 &
--Ping KVStore
java -Xmx256m -Xms256m -jar $KHOME/lib/kvstore.jar ping -host localhost -port 5000
--Start KVStoreAdminClient
java -Xmx256m -Xms256m -jar $KHOME/lib/kvstore.jar runadmin -host localhost -port 5000
```

Figure 20 Lancer le service KV Store

Après on est connecté sur KV store, on peut voir que la table IMMATRICULATION est créé et les données sont réussi à importer.

```
kv-> connect store -name kvstore
Connected to kvstore at localhost:5000.
kv-> get table -name CATALOGUE
>{"CATALOGUEID":8,"MARQUE":"Volvo","NOM":"S80 T6","PUISSEANCE":"272","LONGUEUR":"très longue","NBPLACES":"5","NBPORTES":"5","COULEUR":"bleu","OCCASION":"false","PRIX":"50500"}
>{"CATALOGUEID":15,"MARQUE":"Volkswagen","NOM":"Touran 2.0 FSI","PUISSEANCE":"150","LONGUEUR":"longue","NBPLACES":"7","NBPORTES":"5","COULEUR":"gris","OCCASION":"false","PRIX":"27340"}
>{"CATALOGUEID":22,"MARQUE":"Volkswagen","NOM":"Polo 1.2 6V","PUISSEANCE":"55","LONGUEUR":"courte","NBPLACES":"5","NBPORTES":"3","COULEUR":"blanc","OCCASION":"true","PRIX":"8540"}
>{"CATALOGUEID":32,"MARQUE":"Volkswagen","NOM":"New Beatle 1.8","PUISSEANCE":"110","LONGUEUR":"moyenne","NBPLACES":"5","NBPORTES":"5","COULEUR":"blanc","OCCASION":"true","PRIX":"18641"}
>{"CATALOGUEID":39,"MARQUE":"Volkswagen","NOM":"New Beatle 1.8","PUISSEANCE":"110","LONGUEUR":"moyenne","NBPLACES":"5","NBPORTES":"5","COULEUR":"bleu","OCCASION":"false","PRIX":"26630"}
```

Figure 21 Vérifier la présence de la table Catalogue sur Oracle NoSQL

Le Fichier a été importé dans la base. Nous pouvons donc continuer importer données sur le fichier Marketing.csv.

4.2 Fichier Marketing.java

Ci-dessous c'est la fonction permette de l'importation du fichier marketing.csv sur Oracle NoSQL.

```
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 ;
        while ((ligne = br.readLine()) != null) {
            //int situationFamiliale, 2eme voiture, nbPortes, prix;
            //String marketing age, sexe, nbEnfantsAcharge, 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 situationFamiliale = marketingRecord.get(3);
            String nbEnfantsAcharge = marketingRecord.get(4);
            String deuxiemeVoiture = marketingRecord.get(5);
            // Add the marketing in the KVStore
            this.insertAMarketingRow(age, sexe, taux, situationFamiliale, nbEnfantsAcharge, deuxiemeVoiture);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 22 La fonction importer les données Marketing dans Oracle NoSQL

Pour importer le fichier marketing.csv on fait le même chose que Catalogue.java

```
--exporter le chemin
[vagrant@oracle-21c-vagrant voiture]$ export MYPROJECTHOME=/home/vagrant/Projet_Big_Data/
--compiler le fichier Marketing.java
[vagrant@oracle-21c-vagrant voiture]$ javac -g -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ $MYPROJECTHOME/voiture/Marketing.java
--executer le class Marketing.class
[vagrant@oracle-21c-vagrant ~]$ java -Xmx256m -Xms256m -cp $KVHOME/lib/kvclient.jar:$MYPROJECTHOME/ voiture.Marketing
```

Figure 23 Les commande à exécuter le fichier Marketing.java pour importer des données

Après on va vérifier la présence des données sur Oracle NoSQL.

```
[vagrant@oracle-21c-vagrant voiture]$ java -Xmx256m -Xms256m -jar $KVHOME/lib/kv
store.jar runadmin -host localhost -port 5000
kv-> connect store -name kvstore
Connected to kvstore at localhost:5000.
kv-> get table -name MARKETING
[{"CLIENTMARKETINGID":7,"AGE":"27","SEXE":"F","TAUX":"153","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":2,"DEUXIEMEVOITURE":false}
 {"CLIENTMARKETINGID":8,"AGE":"59","SEXE":"F","TAUX":"572","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":2,"DEUXIEMEVOITURE":false}
 {"CLIENTMARKETINGID":9,"AGE":"43","SEXE":"F","TAUX":"431","SITUATIONFAMILIALE":"Libataire","NBENFANTSACHARGE":0,"DEUXIEMEVOITURE":false}
 {"CLIENTMARKETINGID":19,"AGE":"54","SEXE":"F","TAUX":"452","SITUATIONFAMILIALE":"En Couple","NBENFANTSACHARGE":3,"DEUXIEMEVOITURE":true}
 {"CLIENTMARKETINGID":5,"AGE":"26","SEXE":"F","TAUX":"420","SITUATIONFAMILIALE":}
```

Figure 24 Vérifier la présence de la table Marketing sur Oracle NoSQL

Les données aussi ont bien été importer dans NoSQL.

4.3 Table Interne sur Hive

Après je vais créer deux tables externes sur Hive pour la table MARKETING ET CATALOGUE. Apache Hive est un datawarehouse pour Hadoop qui permet de proposer une abstraction au-dessus de MapReduce pour faciliter l'analyse de gros volumes de données. L'un des avantages de Hive est qu'il permet de définir une structure avec une variété de formats de données, ce qui facilite la requête de ces données et le travail analytique. Il est donc particulièrement adapté à un contexte d'analyse de

données. En outre, Hive propose une fonction de stockage distribué et permet d'accéder aux fichiers stockés dans Oracle NoSQL.

Afin de pouvoir travailler avec Hive, il est nécessaire de créer des tables externes qui pointent respectivement vers KVSTORE pour la table CATALOGUE et MARKETING.

4.3.1 Table Interne Marketing

D'abord on va créer une table externe Marketing_ext.

```
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE Marketing_ext(
    . . . . . > CLIENTMARKETINGID INT,
    . . . . . > AGE STRING,
    . . . . . > SEXE STRING,
    . . . . . > TAUX STRING,
    . . . . . > SITUATIONFAMILIALE STRING,
    . . . . . > NBENFANTSACHARGE STRING,
    . . . . . > DEUXIEMEVOITURE STRING)
    . . . . . > STORED BY 'oracle.kv.hadoop.hive.tablesto
rageHandler'
    . . . . . > TBLPROPERTIES (
    . . . . . . > "oracle.kv_kvstore" = "kvstore",
    . . . . . . > "oracle.kv_hosts" = "localhost:5000",
    . . . . . . > "oracle.kv_tableName" = "MARKETING"
    . . . . . > );
23/03/10 19:55:49 INFO ql.Driver: Compiling command(queryId=vagrant_202303101955
18_172_0_151_E51_1925_1_1_2_120_1000000000)
```

Figure 25 Créer une table Marketing_ext sur Hive

Après on va voir si la table externe est bien créée sur hive par la commande :

```
Select * from Marketing_ext;
```

On peut voir que c'est bien créé.

marketing_ext.clientmarketingid	marketing_ext.age	marketing_ext.sexé	
marketing_ext.taux	marketing_ext.situationfamiliale	marketing_ext.nbenfant	
sacharge	marketing_ext.deuxiemevoiture		
+-----+-----+-----+	+-----+-----+	+-----+	+-----+
11	22	M	
154	En couple	1	
false			
20	35	M	
589	c*libataire	0	
false			
4	48	M	
401	c*libataire	0	
false			
5	26	F	
420	En Couple	3	
true			
6	80	M	
530	En Couple	3	
false			

Figure 26 Vérifier la présence des données dans la table externe Marketing_ext

Mais pour importer les données sur hive, ce n'est pas suffisant de créer une table externe, on doit créer une table interne sur Hive. Il est important de noter que la table interne créée par la commande CREATE TABLE AS SELECT est une table de stockage interne sur Hive. Les données sont stockées dans le stockage Hadoop et ne sont pas accessibles en dehors de Hive. Si on souhaite exporter les données de la table interne dans un autre système, vous pouvez utiliser Sqoop pour extraire les données et les charger dans une autre base de données. Donc Pour créer une table interne sur Hive à partir d'une requête sur une table externe, on peut utiliser la clause CREATE TABLE AS SELECT de Hive. Voici les codes. On crée une table interne Marketing en basant sur la table externe Marketing_ext

Figure 27 Créez une table interne Marketing à partir de la table externe Marketing_ext

Après on vérifie aussi si les données sont bien importées dans la table Marketing ou pas ?

Select * from Marketing ;

	marketing.clientmarketingid	marketing.age	marketing.sexé	marketing.taux uxiemevoiture
4	Celibataire	48	0	M false
5	En couple	26	3	F true
6	En couple	80	3	M false
10	Celibataire	64	0	M false

Figure 28 Vérifier la présence des données dans la table interne Marketing

4.3.2 Table Interne Catalogue

Pareil pour la table Catalogue.on crée une table externe Catalogue_ext sur Hive

```
drop table Catalogue;
CREATE EXTERNAL TABLE Catalogue_ext(
CATALOGUEID INT,
MARQUE STRING,
NOM STRING,
PUISSEANCE 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" = "localhost:5000",
    "oracle.kv.tableName" = "CATALOGUE"
);
```

Figure 29 Créez la table externe Catalogue_ext sur Hive

Après on vérifie si les données sur kV store sont accessible ou pas ? on peut voir que les données sont sur la table Catalogue_ext.

catalogue_ext.catalogueid	catalogue_ext.marque	catalogue_ext.nom	catalogue_ext.prix	
puissance	catalogue_ext.longueur	catalogue_ext.nbpplaces	catalogue_ext.nbportes	
	catalogue_ext.couleur	catalogue_ext.occasion	catalogue_ext.prix	
+-----+-----+-----+-----+	+-----+-----+-----+	+-----+-----+	+-----+-----+	+-----+
13	volkswagen	Touran 2.0 FSI	150	
gris	longue	7	5	
18	true	19138		
	volkswagen	Touran 2.0 FSI	150	
noir	longue	7	5	
24	true	19138		
	volkswagen	Polo 1.2 6V	55	
noir	courte	5	3	
27	false	12200		
	volkswagen	Polo 1.2 6V	55	
bleu	courte	5	3	
41	false	12200		
	volkswagen	New Beatle 1.8	110	
	moyenne	5	5	

Figure 30 Vérifier la présence des données dans la table externe Catalogue_ext

Pour la même raison, on va créer aussi la table interne Catalogue en basant sur la table externe.

```
CREATE TABLE Catalogue  
ASSELECT CATALOGUEID, MARQUE, NOM, PUISSANCE, LONGUEUR, NBPLACES, NBPORTES, COULEUR, OCCASION, PRIX  
FROM Catalogue_ext;
```

Figure 31 Créer une table interne Catalogue à partir de la table externe Catalogue_ext

Après on vas voir que les données sont parfaitement stockées dans la table Catalogue, c'est une table interne.

Figure 32 Vérifier la présence des données dans la table interne Catalogue

5. Conclusion data Lake

La partie Data Lake est donc terminée, on peut voir toutes les tables interne et externe qu'on crée sont bien sur Hive. Alors on a bien créé notre data Lake, on va faire la connexion sur notre data Lake pour récupérer toutes les tables qu'on a besoin pour la partie machine Learning.

```
42_55f0dd09-c451-4143-8532-50110b1ed9e9, Time taken: 0.226 seconds
23/03/10 21:09:42 INFO ql.Driver: OK
+-----+
| tab_name |
+-----+
| catalogue |
| catalogue_ext |
| client |
| co2 |
| immatriculation |
| marketing |
| marketing_ext |
+-----+
7 rows selected (0.226 seconds)
```

Figure 33 Liste des tales dans Data Lake

Partie HADOOP MAP REDUCE

1. Objectif

Après avoir construit notre DATA LAKE le Concessionnaire nous appelle et nous fait part que certaines données étaient perdues avant notre intervention – notamment les détails sur l'émission CO2 / le coût d'énergie / la valeur de Bonus/Malus pour la taxation par marque et modèle de voiture.

Il est possible que ces données seraient utiles pour améliorer la qualité de vos modèles prédictifs. En cherchant sur Internet nous avons trouvé un fichier CO2.csv. C'est une autre base des données qui a certaines informations qui peuvent nous aider mais 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 manquantes (colonne Bonus/Malus) et des valeurs erronées (colonne Bonus/Malus par exemple contient '-6 000€ 1' à la place de '-6 000€').

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

2. Scripts java

2.1 Class Driver

Cette class permet d'instancier les classes Map et Reduce qui vont être utilisées plus loin dans cette partie.

```
public class Co2 {  
    Run|Debug  
    public static void main(String[] args) throws Exception {  
        // Créer un objet de configuration Hadoop.  
        Configuration conf = new Configuration();  
        // Permet à Hadoop de lire ses arguments génériques, récupère les arguments  
        // restants dans ourArgs.  
        String[] ourArgs = new GenericOptionsParser(conf, args).getRemainingArgs();  
        // Obtient un nouvel objet Job: une tâche Hadoop. On fournit la configuration  
        // Hadoop ainsi qu'une description  
        // textuelle de la tâche.  
        Job job = Job.getInstance(conf, "Map Reduce sur le fichier Co2");  
  
        // Définie les classes driver, map et reduce  
        job.setJarByClass(Co2.class);  
        job.setMapperClass(Co2Map.class);  
        job.setReducerClass(Co2Reduce.class);  
  
        // Définie les couples clefs/valeur  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(Text.class);  
  
        // Définie les fichiers d'entrée et de résultat  
        FileInputFormat.addInputPath(job, new Path(ourArgs[0]));  
        FileOutputFormat.setOutputPath(job, new Path(ourArgs[1]));  
  
        if (job.waitForCompletion(true))  
            System.exit(status: 0);  
        System.exit(-1);  
    }  
}
```

Figure 34 Class Driver du programme Map Reduce

2.2 Class Map

La class map permet de transformée toutes les entrées qui sont sous la forme clé, valeur et les renvoie après un traitement sous la forme clé, valeur.

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

    // La fonction MAP.
    // On récupère la ligne en entrée, on la découpe en fonction des virgules
    // On récupère la marque, le malus/bonus, le cout énergie et le rejet CO2
    // On crée le couple key value avec la marque comme key
    // On écrit le couple key value dans le contexte
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {
```

Figure 35 Fonction Map du programme

2.3 Class REDUCE

La class reduce prend en entrée le couple clé, valeur transformée par la class map et pour chaque clé nous calculons la moyenne des trois colonnes (Malus/Bonus, Rejets Co2, Cout energie)

```
Iterator<Text> i = values.iterator();
while (i.hasNext()) {
    String value = i.next().toString();

    String[] splitted_value = value.split("\\|");
    malus_bonus = splitted_value[0];
    rejet = splitted_value[1];
    cout = splitted_value[2];

    sommeBonus_Malus += Integer.parseInt(malus_bonus);
    sommeRejet += Integer.parseInt(rejet);
    sommeCout += Integer.parseInt(cout);

    count++;
}

moy_Malus_Bonus = sommeBonus_Malus / count;
moy_Rejet = sommeRejet / count;
moy_Cout = sommeCout / count;
```

Figure 36 Calcul des moyennes dans la fonction Reduce

3. Exécution du programme Map Reduce

Nous allons déplacer les différents fichiers nécessaires pour faire fonctionner le programme MapReduce (CO2.java, CO2Map.java, CO2Reduce.java) dans la machine virtuelle dans le chemin suivant : /vagrant/MapReduce

```
[vagrant@oracle-21c-vagrant MapReduce]$ ls  
co2.java  co2Map.java  CO2Reduce.java
```

Figure 37 Fichier java dans la vm

Ensuite nous allons compiler les fichiers .java en fichier .class pour pouvoir par la suite créer un JAR pour exécuter le programme

```
[vagrant@oracle-21c-vagrant MapReduce]$ javac co2* -cp $(hadoop classpath)  
[vagrant@oracle-21c-vagrant MapReduce]$ ls  
co2.class  co2.java  co2Map.class  CO2Map.java  CO2Reduce.class  CO2Reduce.java  
[vagrant@oracle-21c-vagrant MapReduce]$ mkdir -p org/co2  
[vagrant@oracle-21c-vagrant MapReduce]$ mv CO2*.class org/co2/  
[vagrant@oracle-21c-vagrant 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/co2.class(in = 1549) (out= 815)(deflated 47%)  
adding: org/co2/CO2Map.class(in = 2470) (out= 1153)(deflated 53%)  
adding: org/co2/CO2Reduce.class(in = 2108) (out= 950)(deflated 54%)  
[vagrant@oracle-21c-vagrant MapReduce]$ ls  
co2.jar  co2.java  co2Map.java  CO2Reduce.java  org  
[vagrant@oracle-21c-vagrant MapReduce]$ |
```

Figure 38 Compilation des fichier java et création du .Jar

Nous exécutons ensuite le programme avec la ligne suivante :

```
hadoop jar CO2.jar org.co2.CO2 /Datalake_CO2/CO2.csv /CO2/results
```

4. Résultat du programme

Une fois que le programme Map Reduce a été exécuté nous pouvons voir si le résultat est bien présent et que le programme a bien sorti les bons résultats.

```
[vagrant@oracle-21c-vagrant ~]$ hadoop fs -cat /co2/results/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
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 39 Résultat de l'exécution du programme

5. Export du fichier

Une fois que nous avons vérifié le fichier nous pouvons exporter le fichier pour pouvoir l'utiliser pour faire la suite. La ligne de commande utilisée est la suivante :

```
hadoop fs -get /CO2/results/part-r-00000 $MYDATAHOME/co2_result.csv
```

6. Jonction entre Catalogue et Co2

6.1 Import des fichiers

Pour importer les fichiers nous utilisons la fonction `read.csv`.

```
co2mapreduce <- read.csv("C:/MBDS/VM/vagrant-projects-staging/oracleDatabase/21.3.0/Groupe_TPT_6/co2_result.csv",
                         header = FALSE, sep = "\t", dec = ".")
catalogue <- read.csv("C:/MBDS/VM/vagrant-projects-staging/oracleDatabase/21.3.0/Groupe_TPT_6/catalogue.csv",
                      header = TRUE, sep = ",", dec = ".")
```

Figure 40 Import des fichiers dans R

6.2 Modification des data-frames

Nous modifions les data-frames pour que l'on soit sûr que la colonne commune entre les deux data-frames contienne bien les marques avec la même case.

```
#Modifications des data-frames
colnames(co2mapreduce) <- c("marque", "Bonus_Malus", "Rejet", "Cout_Energie")

co2mapreduce$marque = tolower(co2mapreduce$marque)
catalogue$marque = tolower(catalogue$marque)
```

Figure 41 Modification des data-frames

6.3 Jointure des data-frames

Nous allons donc fusionner les deux data-frames avec la colonne commune marque et donc créer un data-frame avec les données de catalogue et les trois autres colonnes de Co2.

```
#Jointure entre les deux data-frames
catalogue_modifie <- merge(catalogue,co2mapreduce, by= "marque", all = TRUE)
```

Figure 42 Jointure des deux data-frames

6.4 Complétion des données dans le data-frame

Nous allons maintenant remplacer les valeurs manquantes pour les colonnes Bonus_Malus, Rejet et Cout_energie. Pour chaque colonne nous remplaçons la valeur NA par la moyenne de toutes les autres valeurs de la colonne.

```
#Moyenne des colonnes pour remplacer les NA
moy_Bonus_Malus <- mean(co2mapreduce$Bonus_Malus)
moy_rejet <- mean(co2mapreduce$Rejet)
moy_cout <- mean(co2mapreduce$cout_Energie)

#Remplacement des NA dans les colonnes
catalogue_modifie$Bonus_Malus[is.na(catalogue_modifie$`Bonus_Malus`)] <- moy_Bonus_Malus
catalogue_modifie$Rejet[is.na(catalogue_modifie$Rejet)] <- moy_rejet
catalogue_modifie$cout_Energie[is.na(catalogue_modifie$cout_Energie)] <- moy_cout
```

Figure 43 Ajout des moyennes

Nous avons les moyennes suivantes pour les trois colonnes :

moy_Bonus_Malus	-1719.04545454545
moy_cout	197.681818181818
moy_rejet	39.6818181818182

Figure 44 Moyenne des colonnes de CO2

6.5 Nettoyage du data-frame

Il nous faut maintenant nettoyer le nouveau data-frame car le fichier Co2 a des marques de voiture qui ne sont pas disponible dans le catalogue et nous ne les voulons pas qu'il y a des valeurs NA dans le data-frame et donc nous les enlevons.

```
#Enlever les marques pour lequel nous n'avons pas de voiture
catalogue_modifie <- na.omit(catalogue_modifie)
```

Figure 45 Nettoyage du data-frame

7. Résultat

Nous pouvons maintenant voir le résultat du catalogue avec les nouvelles colonnes que nous allons utiliser pour la suite et ensuite exporter ce data-frame sous forme de csv pour l'utiliser plus tard.

	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	-2400.000	26.00000	191.0000
2	audi	A3 2.0 FSI	150	moyenne	5	5	noir	false	28500	-2400.000	26.00000	191.0000
3	audi	A3 2.0 FSI	150	moyenne	5	5	rouge	false	28500	-2400.000	26.00000	191.0000
4	audi	A3 2.0 FSI	150	moyenne	5	5	gris	true	19950	-2400.000	26.00000	191.0000
5	audi	A3 2.0 FSI	150	moyenne	5	5	blanc	false	28500	-2400.000	26.00000	191.0000
6	audi	A3 2.0 FSI	150	moyenne	5	5	blanc	true	19950	-2400.000	26.00000	191.0000
7	audi	A3 2.0 FSI	150	moyenne	5	5	bleu	true	19950	-2400.000	26.00000	191.0000
8	audi	A3 2.0 FSI	150	moyenne	5	5	bleu	false	28500	-2400.000	26.00000	191.0000
9	audi	A3 2.0 FSI	150	moyenne	5	5	gris	false	28500	-2400.000	26.00000	191.0000
10	audi	A3 2.0 FSI	150	moyenne	5	5	rouge	true	19950	-2400.000	26.00000	191.0000
11	audi	A2 1.4	75	courte	5	5	noir	true	12817	-2400.000	26.00000	191.0000
12	audi	A2 1.4	75	courte	5	5	bleu	false	18310	-2400.000	26.00000	191.0000
13	audi	A2 1.4	75	courte	5	5	gris	false	18310	-2400.000	26.00000	191.0000
14	audi	A2 1.4	75	courte	5	5	bleu	true	12817	-2400.000	26.00000	191.0000
15	audi	A2 1.4	75	courte	5	5	gris	true	12817	-2400.000	26.00000	191.0000
16	audi	A2 1.4	75	courte	5	5	noir	false	18310	-2400.000	26.00000	191.0000
17	audi	A2 1.4	75	courte	5	5	rouge	false	18310	-2400.000	26.00000	191.0000
18	audi	A2 1.4	75	courte	5	5	blanc	true	12817	-2400.000	26.00000	191.0000

Figure 46 Résultat de la jonction des deux fichiers

Partie Machine Learning

1. RStudio et langage R

Enfin, la dernière étape de ce projet consiste à charger les données présentes sur HIVE dans notre machine virtuelle à RStudio sous Windows. R est un langage de programmation utilisé pour le traitement de données et l'analyse statistique. Les commandes sont exécutées à l'aide d'instructions codées dans un langage relativement simple, les résultats sont affichés sous forme de texte et les graphiques sont visualisés directement. Le RStudio est particulièrement performant pour la manipulation de données, le calcul et la création de graphiques. Il possède notamment :

Un système de documentation intégré très bien conçu

Des procédures efficaces pour le traitement et le stockage de données

Une suite d'opérateurs pour des calculs sur des tableaux, notamment sur des matrices

Une vaste collection de procédures statistiques pour l'analyse de données

Des capacités graphiques évoluées

Un langage de programmation simple et efficace, incluant des conditions, des boucles, de la récursivité, ainsi que des possibilités d'entrée-sortie.

Afin de récupérer les données sur Hive, nous allons faire la connexion entre hive et RStudio.

On peut voir que les tables sur hive sont bien affichées sur RStudio.

```
> library(RJDBC)
> hive_jdbc_jar <- "c:/vagrant-projects/OracleDatabase/21.3.0/hive-jdbc-3.1.3-standalone.jar"
> hive_driver <- "org.apache.hive.jdbc.HiveDriver"
> hive_url <- "jdbc:hive2://localhost:10000"
> drv <- JDBC(hive_driver, hive_jdbc_jar)
> conn <- dbConnect(drv, hive_url, "vagrant", "")
> show_tables <- dbGetQuery(conn, "show tables")
> print(show_tables);
   tab_name
1 catalogue
2 catalogue_ext
3 client
4 co2
5 immatriculation
6 marketing
7 marketing_ext
```

Figure 47 Connexion entre Hive et RStudio pour l'importation de tables

On peut voir que les tables Clients, Immatriculations, Marketing, Catalogue et Co2 sont bien import sur RStudio à partir de notre Data Lake Hive.

```
> Clients <- dbGetQuery(conn, "select * from Client")
> summary(Clients)
  client.age   client.sexé   client.taux   client.situationfamiliale client.nbenfantsacharge client.2emevoiture client.immatriculation
Min. :-1.00   Length:200001  Min. : -1.0  Length:200001  Min. :-1.000  Length:200001  Length:200001
1st qu.:27.00 Class :character 1st qu.: 421.0  Class :character 1st Qu.: 0.000  Class :character  Class :character
Median :41.00 Mode :character Median : 522.0  Mode :character Median : 1.000  Mode :character  Mode :character
Mean  :43.68 Mean  : 608.2  Mean  : 1.248
3rd qu.:56.00 3rd Qu.: 827.0  3rd Qu.: 2.000
Max. :84.00  Max. :1399.0  Max. : 4.000
NA's :441    NA's :443    NA's :433
```

Figure 48 Récapitulatif des Clients

```
> Immatriculations <- dbGetQuery(conn, "select * from Immatriculation")
> summary(Immatriculations)
immatriculation.immatriculation immatriculation.marque immatriculation.nom immatriculation.puissance immatriculation.longueur
Length:2000001  Length:2000001  Length:2000001  Min. : 55  Length:2000001
Class :character  Class :character  Class :character  1st Qu.: 75  Class :character
Mode :character  Mode :character  Mode :character  Median :150  Mode :character
                                         Mean :199
                                         3rd Qu.:245
                                         Max. :507
                                         NA's :1
immatriculation.nbplaces immatriculation.nbportes immatriculation.couleur immatriculation.occasion immatriculation.prix
Min. :5  Min. :3.000  Length:2000001  Length:2000001  Min. : 7500
1st Qu.:5  1st Qu.:5.000  Class :character  Class :character  1st Qu.: 18310
Median :5  Median :5.000  Mode :character  Mode :character  Median : 25970
Mean  :5  Mean  :4.868  Mode :character  Mode :character  Mean  : 35783
3rd Qu.:5  3rd Qu.:5.000  Mode :character  Mode :character  3rd Qu.: 49200
Max. :5  Max. :5.000  Mode :character  Mode :character  Max. :101300
NA's :1  NA's :1  ...  ...  NA's :1
```

Figure 49 Récapitulatif des Immatriculations

```
> Marketing <- dbGetQuery(conn, "select * from Marketing")
> summary(Marketing)
marketing.clientmarketingid marketing.age   marketing.sexé   marketing.taux   marketing.situationfamiliale marketing.nbenfantsacharge
Min. : 1  Length:21  Length:21  Length:21  Length:21  Length:21
1st Qu.: 6  Class :character  Class :character  Class :character  Class :character  Class :character
Median :11  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Mean  :11
3rd Qu.:16
Max. :21
marketing.deuxiemevoiture
Length:21
Class :character
Mode :character
```

Figure 50 Récapitulatif des Marketing

```
> Catalogue <- dbGetQuery(conn, "select * from Catalogue")
> summary(Catalogue)
catalogue.catalogueid catalogue.marque  catalogue.nom  catalogue.puissance catalogue.longueur catalogue.nbplaces catalogue.nbportes
Min. : 1.0  Length:271  Length:271  Length:271  Length:271  Length:271  Length:271
1st Qu.: 68.5  Class :character  Class :character  Class :character  Class :character  Class :character  Class :character
Median :136.0  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character  Mode :character
Mean  :136.0
3rd Qu.:203.5
Max. :271.0
catalogue.couleur  catalogue.occasion catalogue.prix
Length:271  Length:271  Length:271
Class :character  Class :character  Class :character
Mode :character  Mode :character  Mode :character
```

Figure 51 Récapitulatif des Catalogue

```
> Co2 <- dbGetQuery(conn, "select * from Co2")
> summary(Co2)
  co2.id      co2.marque_modele  co2.bonus_malus  co2.rejects_co2_g_km co2.cout_energie
Min. : 2.0  Length:438  Length:438  Min. : 0.0  Length:438
1st Qu.:116.0  Class :character  Class :character  1st Qu.: 42.5  Class :character
Median :230.0  Mode :character  Mode :character  Median :182.0  Mode :character
Mean  :230.4
3rd Qu.:345.0
Max. :459.0
NA's :1
```

Figure 52 Récapitulatif des Co2

2. Nettoyage des données

2.1 Nettoyage des données : Clients

Tout d'abord, nous renommons des colonnes. Ensuite, nous indiquons qu'il existe des lignes qui contient des valeurs manquantes (sous forme NA's), donc nous les supprimons. Nous remarquons certaines données sont sous forme character. Pour afficher des détails, nous les transformons au bon format.

```
> names(clients)[1] <- "age"
> names(clients)[2] <- "sexe"
> names(clients)[3] <- "taux"
> names(clients)[4] <- "situationfamiliale"
> names(clients)[5] <- "nbenfantsacharge"
> names(clients)[6] <- "deuxiemevoiture"
> names(clients)[7] <- "immatriculation"
> clients_sans_na <- na.omit(clients)
> clients_sans_na <- droplevels(clients_sans_na)
> clients_sans_na$sexe <- as.factor(clients_sans_na$sexe)
> clients_sans_na$situationfamiliale <- as.factor(clients_sans_na$situationfamiliale)
> clients_sans_na$deuxiemevoiture <- as.logical(clients_sans_na$deuxiemevoiture)
> summary(clients_sans_na)
  age           sexe          taux      situationfamiliale nbenfantsacharge
Min. :-1.00   M       :135274   Min.  :-1.0  En Couple    :127206   Min.  :-1.000
1st Qu.:27.00  F       : 58475   1st Qu.: 421.0  Célibataire: 58795   1st Qu.: 0.000
Median :41.00  Masculin: 1398   Median : 522.0  Seule       : 9706   Median : 1.000
Mean   :43.69  Homme   : 1356   Mean   : 608.2  Marié(e)   : 1263   Mean   : 1.249
3rd Qu.:56.00  Femme   : 596    3rd Qu.: 827.0  Seul        : 602    3rd Qu.: 2.000
Max.   :84.00  Féminin:  580   Max.   :1399.0  ?          : 219   Max.   : 4.000
                           (Other)   : 600   (Other)     : 488
deuxiemevoiture immatriculation
Mode :logical Length:198279
FALSE:172493  Class :character
TRUE :25786   Mode  :character
```

Figure 53 Changement du nom et suppression de nulles

A partir de résumé de data frame Clients_sans_na, nous indiquons que le min de l'âge est -1, mais on a la contrainte [18, 84], donc nous récupérons l'âge est supérieur ou égal à 18 ans.

```
> clients_traitel <- subset(clients_sans_na, age >= 18 )
> library(ggplot2)
> ggplot(clients_traitel, aes(x=sexe))+geom_bar()
```

Figure 54 Respect de la contrainte de l'âge

La valeur de colonne de sexe a des formes différentes, nous pouvons remplacer Masculin et Homme par M, Femme et Féminin par F. Nous supprimons les lignes qui contiennent les valeurs comme "?", " " et "N/D". En utilisant la function ggplot(), nous pouvons faire un diagramme de la répartition du variable sexe.

```
> clients_traitel$sexe <- gsub("Masculin", "M", clients_traitel$sexe)
> clients_traitel$sexe <- gsub("Homme", "M", clients_traitel$sexe)
> clients_traitel$sexe <- gsub("Femme", "F", clients_traitel$sexe)
> clients_traitel$sexe <- gsub("Féminin", "F", clients_traitel$sexe)
> clients_traitel <- subset(Clients_traitel, Clients_traitel$sexe!="N/D"&Clients_traitel$sexe!="?"&Clients_traitel$sexe!="")
> clients_traitel$sexe <- as.factor(Clients_traitel$sexe)
> ggplot(clients_traitel, aes(x=sexe))+geom_bar()
```

Figure 55 Corrige du nom de colonne et valeur erronées

Et pour la variable taux, il faut se réunir dans cet intervalle [544, 74185], donc nous gardons ceux qui est supérieur ou égal à 544. Et nous corrigions les noms des colonnes au bon format.

```
> clients_traitel <- subset(clients_traitel, taux >= 544 )
> qqplot(clients_traitel, aes(x=situationfamiliale))+geom_bar()
```

Figure 56 Respect de la contrainte de taux

Nous remarquons qu'il existe des valeurs erronées ou manquantes(N/D), donc nous les supprimons. Et nous corrigions les noms de colonnes qui ont des caractères erronés.

```
> clients_trate1 <- subset(Clients_trate1, Clients_trate1$situationfamiliale!="N/D" & Clients_trate1$situationfamiliale!="?" & Clients_trate1$situationfamiliale!="")  
> Clients_trate1$situationfamiliale <- gsub("Celibataire", "Celibataire", Clients_trate1$situationfamiliale)  
> Clients_trate1$situationfamiliale <- gsub("Divorce", "Divorce", Clients_trate1$situationfamiliale)  
> Clients_trate1$situationfamiliale <- gsub("Marié", "Marie", Clients_trate1$situationfamiliale)  
> Clients_trate1$situationfamiliale <- as.factor(Clients_trate1$situationfamiliale)
```

Figure 57 Corrige du nom de colonne de client

Nous indiquons que la contrainte de nbEnfantsacharge est [0,4], mais il existe des valeurs négatives, donc nous les supprimons. Et nous gardons ceux dont le texte au format « 9999 AA 99 » pour la colonne « immatriculation ».

```
> #nbenfantsacharge >=0  
> Clients_trate1 <- subset(cclients_trate1, nbenfantsacharge >= 0 )  
>  
> #garder que des lignes en format 9999 AA 99 dans la colonne immatriculation  
> Clients_trate1<- Clients_trate1[grep("^\d{4} [A-Z]{2} \d{2}$", Clients_trate1$immatriculation), ]  
summary(Clients_trate1)  
age sexe taux situationfamiliale nbenfantsacharge deuxiemevoiture immatriculation  
Min. :18.00 F:23655 Min. : 544.0 Celibataire:23390 Min. :0.000 Mode :logical Length:78740  
1st Qu.:27.00 M:55085 1st Qu.: 588.0 Divorce : 36 1st Qu.:0.000 FALSE:68482 Class :character  
Median :41.00 Median : 888.0 En Couple :50716 Median :1.000 TRUE :10258 Mode :character  
Mean :43.72 Mean : 899.2 Marie(e) : 530 Mean :1.252  
3rd Qu.:57.00 3rd Qu.:1144.0 Seul : 212 3rd Qu.:2.000  
Max. :84.00 Max. :1399.0 Seule : 3856 Max. :4.000
```

Figure 58 Respect de la contrainte de nbEnfantsacharge

Parce que l'immatriculation est unique, donc nous supprimons les doubles.

```
> doublons <- which(duplicated(Clients_trate1$immatriculation))  
> Clients_trate1 <- Clients_trate1[-doublons, ]
```

Figure 59 Suppression de double

Le tableau final des clients nettoyés est présenté ci-dessous.

```
> doublons <- which(duplicated(Clients_trate1$immatriculation))  
> Clients_trate1 <- Clients_trate1[-doublons, ]  
summary(Clients_trate1)  
age sexe taux situationfamiliale nbenfantsacharge deuxiemevoiture immatriculation  
Min. :18.00 F:23653 Min. : 544.0 Celibataire:23389 Min. :0.000 Mode :logical Length:78734  
1st Qu.:27.00 M:55081 1st Qu.: 588.0 Divorce : 36 1st Qu.:0.000 FALSE:68476 Class :character  
Median :41.00 Median : 888.0 En Couple :50711 Median :1.000 TRUE :10258 Mode :character  
Mean :43.72 Mean : 899.2 Marie(e) : 530 Mean :1.252  
3rd Qu.:57.00 3rd Qu.:1144.0 Seul : 212 3rd Qu.:2.000  
Max. :84.00 Max. :1399.0 Seule : 3856 Max. :4.000
```

Figure 60 Récapitulatif du tableau des Clients nettoyés

2.2 Nettoyage des données : Catalogue

Tout d'abord, nous changions les noms des colonnes.

```
> ## renommer des colonnes  
> names(Catalogue)[1] <- "id"  
> names(Catalogue)[2] <- "marque"  
> names(Catalogue)[3] <- "nom"  
> names(Catalogue)[4] <- "puissance"  
> names(Catalogue)[5] <- "longueur"  
> names(Catalogue)[6] <- "nbplaces"  
> names(Catalogue)[7] <- "nbportes"  
> names(Catalogue)[8] <- "couleur"  
> names(Catalogue)[9] <- "occasion"  
> names(Catalogue)[10] <- "prix"  
>
```

Figure 61 Renommer les colonnes

Ensuite, pour afficher la quantité de données, nous transformons les données à vecteurs de données.

```
> Catalogue$marque <- as.factor(Catalogue$marque)
> Catalogue$nom <- as.factor(Catalogue$nom)
> Catalogue$longueur <- as.factor(Catalogue$longueur)
> Catalogue$couleur <- as.factor(Catalogue$couleur)
> Catalogue$occasion <- as.logical(Catalogue$occasion)
> Catalogue$puissance<- as.integer(Catalogue$puissance)
> Catalogue$nbplaces<- as.integer(Catalogue$nbplaces)
> Catalogue$prix<- as.integer(Catalogue$prix)
> Catalogue$nbportes<- as.integer(Catalogue$nbportes)
```

Figure 62 Convert to format demandé

Et nous supprimons les lignes de données qui contiennent des valeurs manquantes (NA).

```

> Catalogue_sans_na <- na.omit(catalogue)
> Catalogue_sans_na <- droplevels(Catalogue_sans_na)
>
> summary(Catalogue_sans_na)
      id           marque        nom       puissance    longueur
Min. : 2.00   Renault : 40   1007 1.4 : 10   Min. : 55.0   courte   :60
1st Qu.: 69.25 Volkswagen: 40   120i     : 10   1st Qu.:109.0   longue   :90
Median :136.50 Audi      : 20   9.3 1.8T : 10   Median :147.0   moyenne  :70
Mean   :136.50 BMW      : 20   A2 1.4   : 10   Mean   :157.6   très longue:50
3rd Qu.:203.75 Mercedes: 20   A200     : 10   3rd Qu.:170.0
Max.   :271.00 Nissan   : 15   A3 2.0 FSI: 10   Max.   :507.0
                           (Other)  :115  (Other)  :210

      nbplaces    nbportes    couleur    occasion      prix
Min.   :5.000   Min.   :3.000   blanc:54   Mode :logical   Min.   : 7500
1st Qu.:5.000   1st Qu.:5.000   bleu :54   FALSE:160    1st Qu.:16029
Median :5.000   Median :5.000   gris :54   TRUE :110    Median : 20598
Mean   :5.222   Mean   :4.815   noir :54
3rd Qu.:5.000   3rd Qu.:5.000   rouge:54
Max.   :7.000   Max.   :5.000

```

Figure 63 Suppression de nulles

Nous aussi corrigions les valeurs des colonnes qui sont erronées.

```
> Catalogue_sans_na$longueur <- gsub("très longue", "tres longue", catalogue_sans_na$longueur)
> Catalogue_sans_na$longueur <- as.factor(catalogue_sans_na$longueur)
```

Figure 64 Nettoyage de caractère incorrect

Le tableau final du Catalogue nettoyé est présenté ci-dessous.

```
> summary(catalogue_traité)
      id           marque        nom       puissance      longueur
Min. : 2.00  Renault : 40  1007 1.4 : 10  Min. : 55.0  courte   :60
1st Qu.: 69.25 Volkswagen: 40  120i     : 10  1st Qu.:109.0  longue   :90
Median :136.50 Audi      : 20  9.3 1.8T : 10  Median :147.0 moyenne  :70
Mean   :136.50 BMW      : 20  A2 1.4   : 10  Mean   :157.6 très longue:50
3rd Qu.:203.75 Mercedes : 20  A200    : 10  3rd Qu.:170.0
Max.   :271.00 Nissan    : 15  A3 2.0 FSI: 10  Max.   :507.0
                           (Other) :115 (Other) :210

      nbplaces      nbportes     couleur     occasion      prix
Min. :5.000  Min. :3.000  blanc:54  Mode :logical  Min. : 7500
1st Qu.:5.000 1st Qu.:5.000  bleu :54  FALSE:160   1st Qu.:16029
Median :5.000  Median :5.000  gris :54  TRUE :110   Median : 20598
Mean   :5.222  Mean   :4.815  noir :54
3rd Qu.:5.000 3rd Qu.:5.000  rouge:54
Max.   :7.000  Max.   :5.000
```

Figure 65 Récapitulatif du tableau du Catalogue nettoyé

2.3 Nettoyage des données : Immatriculations

La première étape est comme avant de changer les noms des colonnes.

```
> ## renommer des colonnes
> names(Immatriculations)[1] <- "immatriculation"
> names(Immatriculations)[2] <- "marque"
> names(Immatriculations)[3] <- "nom"
> names(Immatriculations)[4] <- "puissance"
> names(Immatriculations)[5] <- "longueur"
> names(Immatriculations)[6] <- "nbplaces"
> names(Immatriculations)[7] <- "nbportes"
> names(Immatriculations)[8] <- "couleur"
> names(Immatriculations)[9] <- "occasion"
> names(Immatriculations)[10] <- "prix"
>
```

Figure 66 Corrige du nom de colonne

```
> Immatriculations$immatriculation <- as.factor(Immatriculations$immatriculation)
> Immatriculations$marque <- as.factor(Immatriculations$marque)
> Immatriculations$nom <- as.factor(Immatriculations$nom)
> Immatriculations$longueur <- as.factor(Immatriculations$longueur)
> Immatriculations$couleur <- as.factor(Immatriculations$couleur)
> Immatriculations$occasion <- as.logical(Immatriculations$occasion)
>
> Immatriculations_sans_na <- na.omit(Immatriculations)
> Immatriculations_sans_na <- droplevels(Immatriculations_sans_na)
>
> summarv(Immatriculations sans na)
```

Figure 67 Convert to format demandé

Pour la colonne « immatriculation », à partir du domaine de valeurs, nous gardons ceux dont le texte au format « 9999 AA 99 ».

```
> Immatriculations_sans_na <- Immatriculations_sans_na[,-1]
> Immatriculations_traité <- Immatriculations_sans_na
> Immatriculations_traité<- Immatriculations_traité[grep("^\d{4} [A-Z]{2} \d{2}$", Immatriculations_traité$immatriculation), ]
#immatriculation double?
> doublons <- which(duplicated(Immatriculations_traité$immatriculation))
> Immatriculations_traité <- Immatriculations_traité[-doublons,]
> Immatriculations_traité$longueur <- gsub("très longue", "tres longue", Immatriculations_traité$longueur)
> Immatriculations_traité$longueur <- as.factor(Immatriculations_traité$longueur)
```

Figure 68 Suppression ID, convert format, suppression double, et corrige valeur erronée

Le tableau final des Immatriculations nettoyés est présenté ci-dessous.

```
> summary(Immatriculations_traité)
      marque          nom        puissance      longueur      nbplaces      nbportes      couleur
BMW      :264567  A2 1.4       :255022  Min.   :55  courte    :494211  Min.   :5  Min.   :3.000  blanc:358278
Audi     :262265  M5          :230523  1st Qu.:75  longue   :489744  1st Qu.:5  1st Qu.:5.000  bleu :360023
Renault  :225923  X-Type 2.5 V6  :169630  Median :150 moyenne  :207526  Median :5  Median :5.000  gris :360327
Jaguar   :169630  S80 T6       :11244  Mean   :199 tres longue:605371  Mean   :5  Mean   :4.868  noir :359381
Völkswagen:140117 vel Satis 3.5 V6:110671  3rd Qu.:245               3rd Qu.:5  3rd Qu.:5.000  rouge:358843
Mercedes :134842  S500         :93180  Max.   :507               Max.   :5  Max.   :5.000
(Other)   :599508  (Other)      :826582
      occasion      prix      immatriculation
Mode :logical  Min.   : 7500  1000 AD 49:    1
FALSE:1234892  1st Qu.: 18310 1000 AD 66:    1
TRUE :561960   Median : 25970 1000 AE 54:    1
              Mean   : 35778 1000 AG 15:    1
              3rd Qu.: 49200 1000 AM 17:    1
              Max.   :101300 1000 AQ 88:    1
              (Other)  :1796846
```

Figure 69 Récapitulatif du tableau des Immatriculations nettoyés

3. Analyse exploratoire des données vues

Dans cette étape, nous allons faire des dessins pour afficher les données en vues. Les figures suivantes sont la répartition de la variable couleur, longueur, nbPlaces, nbPortes.

A partir du diagramme de couleur, nous remarquons que le besoin du client pour la couleur est équilibrant. Donc ce n'est pas nécessaire de considérer cette variable quand nous créons des catégories des voitures.

```
> ggplot(Catalogue_traité, aes(x=couleur))+geom_bar()
```

Figure 70 Code de dessin de couleur de catalogue

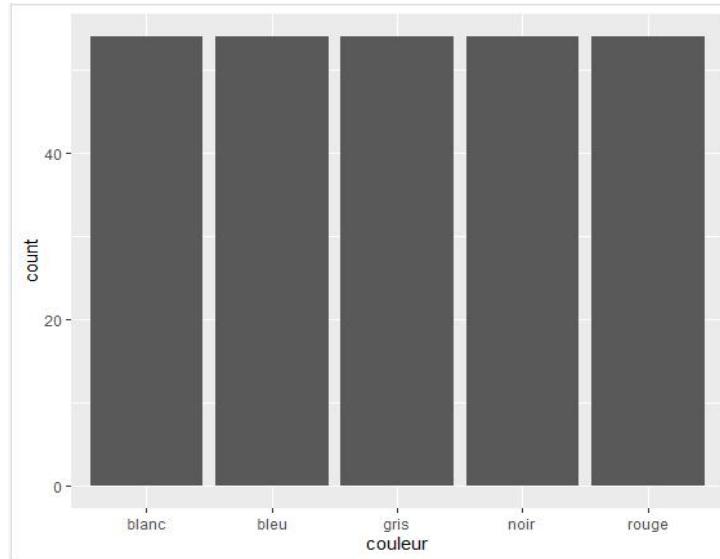


Figure 71 La répartition de la variable couleur

A partir du diagramme de longueur, le client qui choisit la voiture de longueur « longue » est plus que ceux qui choisissent les autres. Et nous considérons cette variable pour créer « categories » dans l'étape suivante.

```
> ggplot(catalogue_traité, aes(x=longueur))+geom_bar()
```

Figure 72 Code de dessin de longueur de catalogue

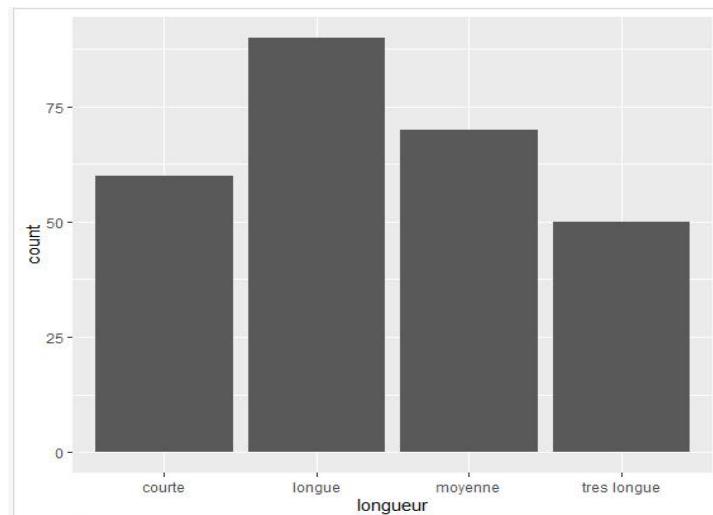


Figure 73 La répartition de la variable longueur

A partir du diagramme de nbPlaces, nous remarquons qu'il existe deux possibilités de nbPlaces : 5 et 7 et nbPlaces = 7 correspond à une situation spéciale pour la famille qui a plus de 5 personnes. Donc cette variable aura été utilisé pour créer des catégories de véhicules.

```
> ggplot(Catalogue_traité, aes(x=nbPlaces))+geom_bar()
```

Figure 74 Code de dessin de nbPlaces de catalogue

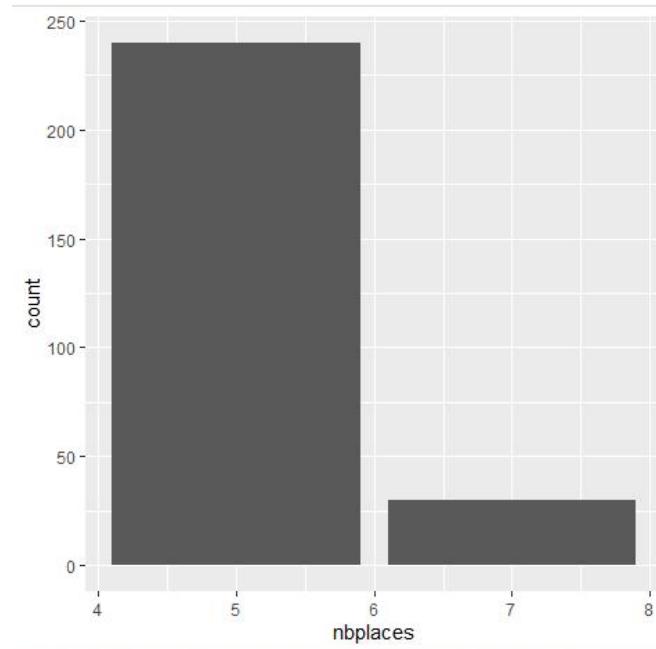


Figure 75 La répartition de la variable nbPlaces

A partir du diagramme de nbPortes, il existe une grande différence dans nbPortes. Donc nous pouvons utiliser cette variable pour créer des catégories de véhicules.

```
> ggplot(Catalogue_traité, aes(x=nbportes))+geom_bar()
```

Figure 76 Code de dessin de nbPortes de catalogue

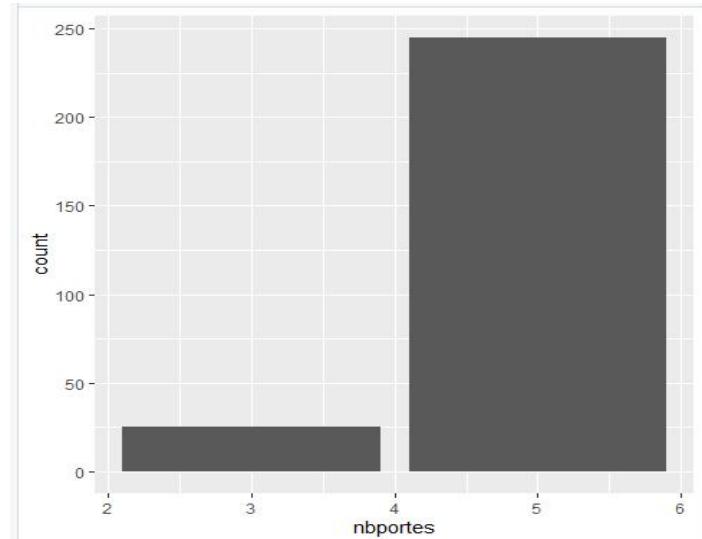


Figure 77 La répartition de la variable nbPortes

A partir du diagramme de puissance, la puissance varie beaucoup pour les voitures différentes. Donc nous le considérons pour évaluer les catégories de voitures.

```
> ggplot(catalogue_traite, aes(x=puissance))+geom_bar()
```

Figure 78 Code de dessin de puissance de catalogue

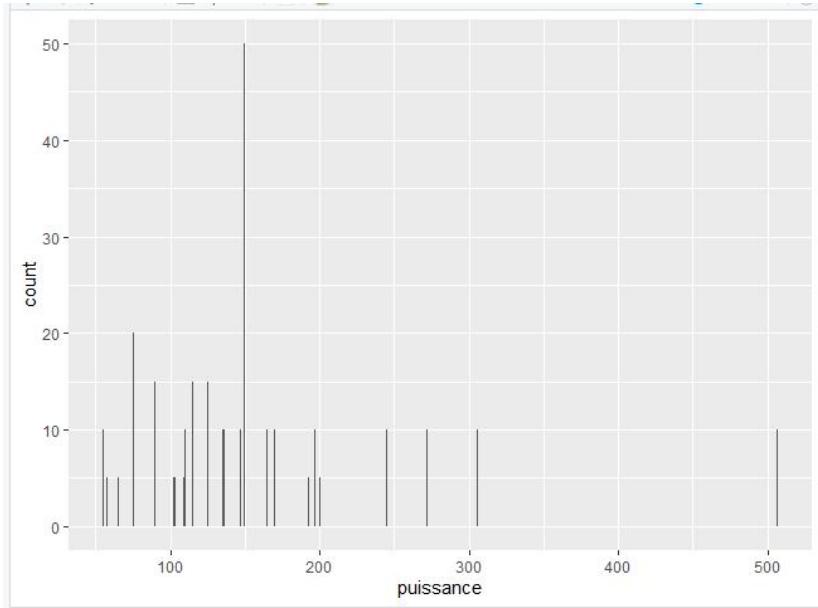


Figure 79 La répartition de la variable puissance

Nous pouvons aussi dessiner pour chaque type de longueur la répartition de la variable nbPortes, nbPlaces, et puissance.

Le diagramme de corrélation entre la longueur et nbPortes montre que les voitures à 5 portes existent pour toutes les longueurs, et que les voitures à 3 portes n'existent que pour la longueur="courte", ce qui nous permet de classer les voitures courtes dans une catégorie distincte.

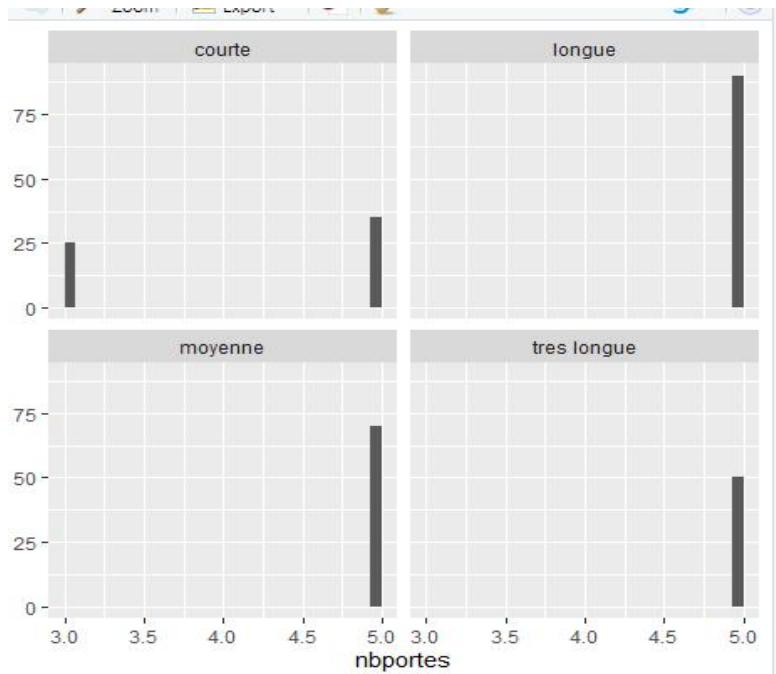


Figure 80 La répartition de la variable nbPortes de chaque type de longueur

La corrélation entre longueur et nbPlaces montre qu'il n'y a que des voitures de 5 et 7 places pour toutes les voitures, et que les voitures de 7 places ne se trouvent que dans les voitures longues. Nous envisageons donc de diviser les voitures longues en deux catégories : les voitures longues à 5 places et les voitures longues à 7 places.

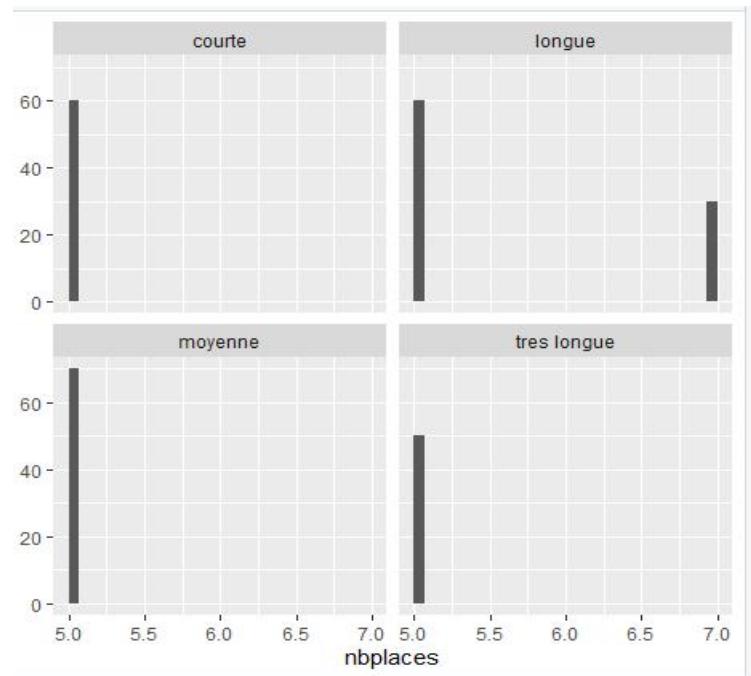


Figure 81 La répartition de la variable nbPlaces de chaque type de longueur

Enfin, nous considérons la relation entre la puissance et la longueur. Lorsque la puissance est supérieure à 250, seuls les modèles très longs sont disponibles, c'est pourquoi nous avons divisé ce cas en une catégorie distincte.

Nous observons ensuite une distribution plus concentrée de la puissance inférieure à 185, de sorte que nous utilisons la puissance 185 comme autre ligne de démarcation. Celle-ci est divisée en puissance inférieure à 185, puissance comprise entre 180 et 250 et puissance supérieure à 250 (qui a été divisée en une catégorie distincte).

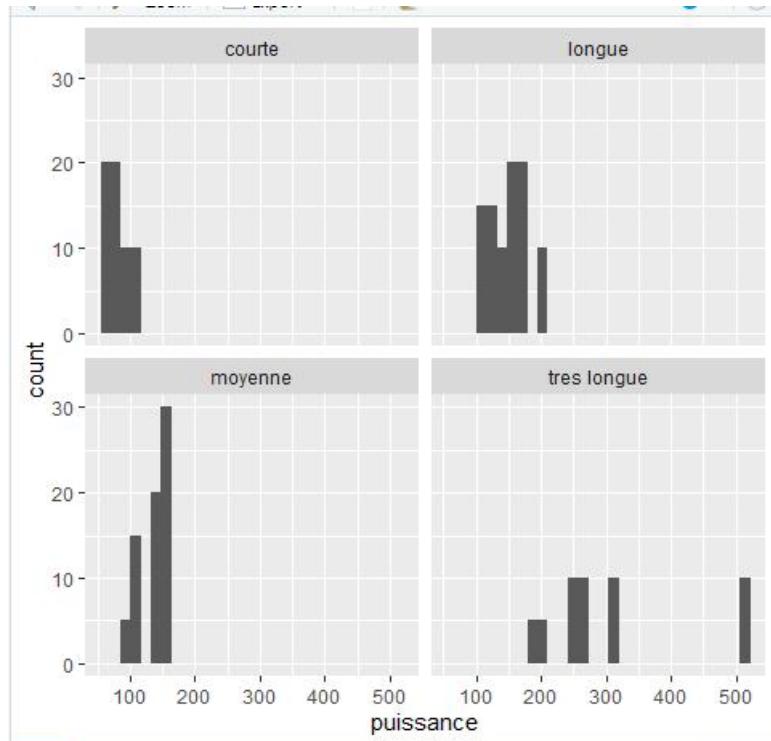


Figure 82 La répartition de la variable puissance de chaque type de longueur

Sur la base de la répartition de la puissance entre les modèles et de l'analyse précédente, nous avons finalement créé les catégories.

4. Identification des catégories de véhicules et application aux Immatriculations

Les 3 critères pour créer la variable categories sont cités au-dessous :

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

Nous créons categories de minicar (longueur= courte), compacte (longueur= moyenne), routière (longueur=longue, puissance<185, nbplaces=5), suv (longueur=longue, puissance<185,nbplaces=7), sportive (longueur=longue/très longue, 185<=puissance<250), berline (longueur=très longue, puissance>=250). Le code est comme dans la figure :

```
> Catalogue_traite$categories <- ifelse(Catalogue_traite$longueur=="courte","minicar",
+                                         ifelse(Catalogue_traite$longueur=="moyenne","compacte",
+                                         ifelse(Catalogue_traite$longueur=="longue" & Catalogue_traite$puissance < 185 & Catalogue_traite$nbplaces == 5 , "Routiere",
+                                         ifelse(Catalogue_traite$longueur=="longue" & Catalogue_traite$puissance < 185 & Catalogue_traite$nbplaces == 7 , "suv",
+                                         ifelse(Catalogue_traite$longueur=="longue" | Catalogue_traite$longueur=="tres longue" & Catalogue_traite$puissance >=185 & Catalogue_traite$puissance <250 , "sportive",
+                                         ifelse(Catalogue_traite$longueur == "tres longue" & Catalogue_traite$puissance >= 250 , "berline","Rien")))))
>
> Catalogue_traite$categories <- as.factor(Catalogue_traite$categories)
> summary(Catalogue_traite)
```

Figure 83 Categories à partir de Catalogue

```
> Immatriculations_traite$categories <- ifelse(Immatriculations_traite$longueur=="courte","minicar",
+                                         ifelse(Immatriculations_traite$longueur=="moyenne","compacte",
+                                         ifelse(Immatriculations_traite$longueur=="longue" & Immatriculations_traite$puissance < 185 & Immatriculations_traite$nbplaces == 5,"Routiere",
+                                         ifelse(Immatriculations_traite$longueur=="longue" & Immatriculations_traite$puissance < 185 & Immatriculations_traite$nbplaces == 7,"suv",
+                                         ifelse(Immatriculations_traite$longueur=="longue" | Immatriculations_traite$longueur=="tres longue" & Immatriculations_traite$puissance >=185 & Immatriculations_traite$puissance <250,"sportive",
+                                         ifelse(Immatriculations_traite$longueur == "tres longue" & Immatriculations_traite$puissance >= 250 , "berline","Rien")))))
>
> Immatriculations_traite$categories <- as.factor(Immatriculations_traite$categories)
> summary(Immatriculations_traite)
```

Figure 84 Application de catégories aux Immatriculations

5. Fusion des données Clients et Immatriculations

D'abord nous créons un nouveau donnée appelé « Clients_Immatriculations » en fusionnant deux autres données « Immatriculations_traite » et « Clients_traite », en utilisant la colonne « immatriculation » comme clé de fusion.

Après nous supprimons la première colonne « immatriculation » et les colonnes qui ne sont pas pertinentes avec les prévisions ou qui font doubles colonnes de données.

```
> clients_Immatriculations <- merge(immatriculations_traite, clients_traite, by = "immatriculation")
>
> clients_Immatriculations <- clients_Immatriculations[,-1]
>
> 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)
>
> summary(clients_Immatriculations)
   categories      age       sexe      taux      situationfamiliale nbenfantsacharge deuxiemevoiture
berline :24326  Min.   :18.00  F:123653  Min.   :544.0  Celibataire:23389  Min.   :0.000  Mode :logical
compacte: 5706  1st Qu.:27.00  M:55081   1st Qu.: 588.0  Divorce    : 36   1st Qu.:0.000  FALSE:68476
minicar :24963  Median :41.00          Median : 888.0  En Couple :50711  Median :1.000  TRUE :10258
Routiere: 8350  Mean   :43.72          Mean   : 899.2  Marie(e)  : 530   Mean   :1.252
sportive:15389  3rd Qu.:57.00          3rd Qu.:1144.0  Seul     : 212   3rd Qu.:2.000
                           Max.   :84.00          Max.   :1399.0  Seule    : 3856  Max.   :4.000
>
```

Figure 85 Fusion de data frames et suppression de colonnes inutiles

6. Modèle de classification et les tests dans différents algorithmes

6.1 Situation 1 : normal

Nous changeons tous les colonnes de données en facteurs et résume les données Clients_Immatriculations. Nous pouvons voir qu'il contient les colonnes suivantes : categories, age, sexe, taux, situationfamiliale, nbenfantsacharge et deuxiemevoiture.

```
> 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$situationfamiliale <- as.factor(Clients_Immatriculations$situationfamiliale)
> Clients_Immatriculations$nbenfantsacharge <- as.factor(Clients_Immatriculations$nbenfantsacharge)
> Clients_Immatriculations$deuxiemevoiture <- as.factor(Clients_Immatriculations$deuxiemevoiture)
> Clients_Immatriculations$categories <- as.factor(Clients_Immatriculations$categories)
> summary(Clients_Immatriculations)
   categories      age       sexe      taux      situationfamiliale nbenfantsacharge deuxiemevoiture 
berline :24326  25 : 2053  F:23653  550 : 506  Celibataire:23389  0:35312          FALSE:68476 
compacte: 5706  29 : 2018  M:55081  564 : 483  Divorce : 36  1:13216          TRUE :10258 
minicar :24963  23 : 1988           581 : 480  En Couple :50711  2:13129 
Routiere: 8350  18 : 1986           565 : 478  Marie(e) : 530  3: 9192 
sportive:15389  21 : 1979           577 : 468  Seul : 212  4: 7885 
                     28 : 1978           557 : 467  seule : 3856 
                     (other):66732           (other):75852
```

Figure 86 Convert to format

Enfin, nous créons deux jeux de données, un jeu d'apprentissage à 70 % et un jeu de test à 30 %, donc nous sélectionnons les lignes 1 à 55114 pour le jeu d'apprentissage et les lignes 55115 à 78734 pour le jeu de test.

```
> Clients_Immatriculations_traite_EA <- Clients_Immatriculations[1:55114,]
> Clients_Immatriculations_traite_ET <- Clients_Immatriculations[55115:78734,]
```

Figure 87 Situation1_Séparation de l'échantillonnage d'apprentissage et test

6.1.1 Algorithme : C50

Nous utilisons l'algorithme C5.0 pour la classification de catégories à partir de données Clients_Immatriculations.

D'abord nous créons un objet treec pour classifier les catégories en fonction des variables de Clients_Immatriculations_traite_EA. Nous pouvons voir le résultat suivant.

```
> treec <- C5.0(categories ~., Clients_Immatriculations_traite_EA)
> print(treec)

call:
C5.0.formula(formula = categories ~ ., data = clients_Immatriculations_traite_EA)

Classification Tree
Number of samples: 55114
Number of predictors: 6

Tree size: 12

Non-standard options: attempt to group attributes
```

Figure 88 Formation de jeux d'apprentissage avec C5.0

Après nous prédisons les catégories pour le jeu de données Clients_Immatriculations_traite_ET à l'aide du modèle treec. Et nous obtenons le résultat suivant.

```
> test_treec <- predict(treec, Clients_Immatriculations_traite_ET, type = "class")
> table(test_treec)
test_treec
berline compacte minicar Routiere sportive
    4701     1172     8075     4248     5424
```

Figure 89 Prédiction de de jeu du test avec C50 et affichage des résultats prédits

Nous créons une matrice de confusion pour comparer les catégories prédites par le modèle aux valeurs réelles, puis nous calculons manuellement son rappel, sa précision et son taux d'erreur et nous les avons présentés sous forme de tableau.

```
> table(Clients_Immatriculations_traite_ET$categories, test_treec)
   test_treec
berline compacte minicar Routiere sportive
berline    4310      0      2    1178    1792
compacte      0    593    1102      0      1
minicar       4    578    6970      0      1
Routiere      1      0      1    2437     51
sportive    386      1      0     633   3579
```

Figure 90 Situation1_C50 : Dessin de la matrice de mélange

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4310	0	2	1178	1792	0,591870365
compacte	0	593	1102	0	1	0,349646226
minicar	4	578	6970	0	1	0,922812128
Routiere	1	0	1	2437	51	0,978714859
sportive	386	1	0	633	3579	0,778212655
Précision	0,9168262	0,5059727	0,86315789	0,57368173	0,65984513	
taux d'erreur	0,2426334					

Tableau 1 Situation1_C50 : Calcul du rappel, de la précision et des taux d'erreur

Enfin, nous créons un objet c_prob qui stocke les probabilités de classification des catégories pour chaque observation dans l'ensemble de données Clients_Immatriculations_traite_ET. Et nous calculons l'aire sous la courbe (AUC) pour le modèle multiclass en utilisant la fonction multiclass.roc pour évaluer la capacité du modèle à classer correctement les catégories. Nous pouvons obtenir que l'AUC est égale à 0,9418 pour ce test de l'algorithme C50.

```
> c_prob <- predict(treec, Clients_Immatriculations_traite_ET, type = "prob")
> c_auc <- multiclass.roc(Clients_Immatriculations_traite_ET$categories, c_prob)
> print(c_auc)

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

Data: multivariate predictor c_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar,
Routiere, sportive.
Multi-class area under the curve: 0.9418
> |
```

Figure 91 Situation1_C50 : calcul de l'AUC

6.1.2 Algorithme : naivebayes

Nous utilisons l'algorithme naive bayes pour prédire les catégories à partir de données Clients_Immatriculations. De la même manière que pour l'algorithme C50, nous obtenons les résultats prédits par l'algorithme naive bayes, qui sont présentés dans le tableau (nb_classe).

```
> ## ---- ---- naivebayes -----
> nb <- naive_bayes(categories~, clients_Immatriculations_traite_EA)
```

Figure 92 Formation de jeux d'apprentissage avec naivebayes

```
> nb_class <- predict(nb, clients_Immatriculations_traite_ET, type = "class")
```

Figure 93 Prédiction de de jeu du test avec naivebayes

Nous obtenons également la matrice de confusion et calculons les taux de rappel, de précision et d'erreur.

```
> table( clients_Immatriculations_traite_ET$categories, nb_class)
   nb_class
   berline compacte minicar Routiere sportive
berline    4669      0     364     535    1714
compacte      0    876     819      0      1
minicar    1430    927    5195      0      1
Routiere     416      1    254    1078    741
sportive     713      1    551    157    3177
```

Figure 94 Situation1_naivebayes : Dessin de la matrice de mélange

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4669	0	364	535	1714	0,641170008
compacte	0	876	819	0	1	0,516509434
minicar	1430	927	5195	0	1	0,68780617
Routiere	416	1	254	1078	741	0,432931727
sportive	713	1	551	157	3177	0,690802348
Précision	0,6459602	0,48531856	0,72323542	0,60903955	0,56389776	
taux d'erreur	0,3651566					

Tableau 2 Situation1_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9201 pour ce test de l'algorithme naviebayes.

```
> nb_prob <- predict(nb, clients_Immatriculations_traite_ET, type="prob")
> nb_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, nb_prob)
> print(nb_auc)

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

Data: multivariate predictor nb_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar,
Routiere, sportive.
Multi-class area under the curve: 0.9201
```

Figure 95 Situation1_Naivebayes : calcul de l'AUC

6.1.3 Algorithme : Random Forest

Nous avons essayé d'utiliser l'algorithme Random Forest, mais il était impossible de le détecter en raison du nombre de taux et de catégories d'âge.

```
> ## ---- Random Forest ----
> clients_Immatriculations_traite_EA3 <- clients_Immatriculations_traite_EA
> clients_Immatriculations_traite_ET3 <- clients_Immatriculations_traite_ET
>
> RF <- randomForest(categories ~ ., clients_Immatriculations_traite_EA3)
Error in randomForest.default(m, y, ...) :
  Can not handle categorical predictors with more than 53 categories.
```

Figure 96 Formation de jeux d'apprentissage avec random forest

6.1.4 Algorithme : NNET

Nous avons essayé d'utiliser l'algorithme NNET, mais il était impossible de le détecter en raison du nombre de taux et de catégories d'âge.

```
> ## ---- NNET ----
> nnet <- nnet(categories ~ ., clients_Immatriculations_traite_EA, size=6, maxit=180, act.fct = "softmax")
Error in nnet.default(x, y, w, softmax = TRUE, ...) :
  trop (5033) de pondérations
```

Figure 97 Formation de jeux d'apprentissage avec NNET

6.1.5 Algorithme : Régression logistique multinomiale

Nous avons essayé d'utiliser l'algorithme Régression logistique multinomiale, mais il était impossible de le détecter en raison du nombre de taux et de catégories d'âge.

```
> ## ---- un modèle de régression logistique multinomiale ----
> # Entrainer un modèle de régression logistique multinomiale
> model <- multinom(categories ~., data = clients_Immatriculations_traite_EA)
Error in nnet.default(x, Y, w, mask = mask, size = 0, skip = TRUE, softmax = TRUE, :
  trop (4170) de pondérations
```

Figure 98 Formation de jeux d'apprentissage avec Régression logistique multinomiale

6.2 Situation 2 : supprimer la colonne taux

Nous considérons le cas où nous ne pouvions pas utiliser plus d'un algorithme en raison du nombre de catégories de variables. Nous avons décidé de diviser les cas de classification multiples, de comparer les résultats de détection et de choisir l'algorithme le plus approprié. Par conséquent, pour la situation 2, nous choisissons de supprimer la colonne taux directement de ce jeu d'apprentissage et de ce jeu de test.

```
> clients_Immatriculations_traite_EA <- subset(clients_Immatriculations_traite_EA , select = -taux)
> clients_Immatriculations_traite_ET <- subset(clients_Immatriculations_traite_ET , select = -taux)
```

Figure 99 Situation2_Séparation de l'échantillonnage d'apprentissage et test

6.2.1 Algorithme : C50

Nous exécutons à nouveau l'algorithme C50 et obtenons les résultats suivants :

```
> table(test_tréec)
test_tréec
berline compacte minicar Routiere sportive
 4701     775   8472   1911    7761
>
> table(clients_Immatriculations_traite_ET$categories, test_tréec)
            test_tréec
berline compacte minicar Routiere sportive
berline      4310      0      2     648    2322
compacte      0     382    1313      0      1
minicar        4     393    7155      0      1
Routiere       1      0      1    1263    1225
sportive      386      0      1      0    4212
...           ...     ...    ...    ...    ...
```

Figure 100 Situation2_C50 résultat de prédiction

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4310	0	2	648	2322	0,591870365
compacte	0	382	1313	0	1	0,225235849
minicar	4	393	7155	0	1	0,947305706
Routiere	1	0	1	1263	1225	0,507228916
sportive	386	0	1	0	4212	0,915851272
Précision	0,9168262	0,49290323	0,84454674	0,66091052	0,54271357	
taux d'erreur	0,2666384					

Tableau 3 Situation2_C50 : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9039 pour ce test de l'algorithme C50.

```
> c_prob <- predict(tréec, clients_Immatriculations_traite_ET, type = "prob")
>
> c_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, c_prob)
> print(c_auc)

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

Data: multivariate predictor c_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar,
Routiere, sportive.
Multi-class area under the curve: 0.9039
```

Figure 101 Situation2_C50 : calcul de l'AUC

6.2.2 Algorithme : Naivebayes

Nous exécutons à nouveau l'algorithme Naivebayes et obtenons les résultats suivants :

```
> table(nb_class)
nb_class
berline compacte minicar Routiere sportive
 6562    1340    7165    1832    6721
> table( clients_Immatriculations_traite_ET$categories, nb_class)
nb_class
berline compacte minicar Routiere sportive
berline   4441      0    228     620    1993
compacte      0    670   1025      0      1
minicar    1465    669   5418      0      1
Routiere     103      1    119   1212    1055
sportive     553      0    375      0    3671
```

Figure 102 Situation2_Naivebayes résultat de prédiction

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4441	0	228	620	1993	0,609859929
compacte	0	670	1025	0	1	0,39504717
minicar	1465	669	5418	0	1	0,717330862
Routiere	103	1	119	1212	1055	0,486746988
sportive	553	0	375	0	3671	0,798217004
Précision	0,6767754	0,5	0,75617585	0,66157205	0,54619848	
taux d'erreur	0,3475021					

Tableau 4 Situation2_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,8908 pour ce test de l'algorithme Naivebayes.

```
> nb_prob <- predict(nb, clients_Immatriculations_traite_ET, type="prob")
Warning message:
predict.naive_bayes(): more features in the newdata are provided as there are probability tables in the object. Calculation is performed based on features to be found in the tables.
> # Installation du package
> nb_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, nb_prob)
> print(nb_auc)

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

Data: multivariate predictor nb_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar,
Routiere, sportive.
Multi-class area under the curve: 0.8908
```

Figure 103 Situation2_Naivebayes: calcul de l'AUC

6.2.3 Algorithme : Random Forest

L'algorithme Random Forest n'a pas pu être mis en œuvre en raison du trop grand nombre de catégories d'âge.

```
> ## ---- Random Forest ----
> Clients_Immatriculations_traite_EA3 <- Clients_Immatriculations_traite_EA
> Clients_Immatriculations_traite_ET3 <- Clients_Immatriculations_traite_ET
>
> RF <- randomForest(categories ~ ., Clients_Immatriculations_traite_EA3)
Error in randomForest.default(m, y, ...) :
  Can not handle categorical predictors with more than 53 categories.
```

Figure 104 Formation de jeux d'apprentissage avec random forest

6.2.4 Algorithme : NNET

Nous utilisons l'algorithme NNET pour prédire les catégories à partir de données Clients_Immatriculations. De la même manière que pour l'algorithme C50, nous obtenons les résultats prédis par l'algorithme NNET, qui sont présentés dans le tableau (nn_classe).

```
> ## ---- NNET ----
> nnet <- nnet(categories ~., clients_Immatriculations_traite_EA, size=6, maxit=180, act.fct = "softmax")
# weights:  503
initial value 95597.927624
iter  10 value 44649.881309
iter  20 value 36902.785769
iter  30 value 35251.401558
iter  40 value 33823.381033
iter  50 value 33354.436900
iter  60 value 33025.379932
iter  70 value 32866.801962
iter  80 value 32777.690782
iter  90 value 32713.553218
iter 100 value 32674.754380
iter 110 value 32631.095213
iter 120 value 32591.460197
iter 130 value 32494.472288
iter 140 value 32435.962512
iter 150 value 32412.244723
iter 160 value 32391.396351
iter 170 value 32375.580918
iter 180 value 32366.681955
final  value 32366.681955
stopped after 180 iterations
```

Figure 105 Formation de jeux d'apprentissage avec NNET

```
> nn_class <- predict(nnet, clients_Immatriculations_traite_ET, type="class")
> table(nn_class)
nn_class
berline compacte minicar Routiere sportive
    4769      764     8483     1873     7731
```

Figure 106 Prédiction de de jeu du test avec NNET

Nous obtenons également la matrice de confusion et calculons les taux de rappel, de précision et d'erreur.

```
> table(clients_Immatriculations_traite_ET$categories, nn_class)
   nn_class
   berline compacte minicar Routiere sportive
berline    4329      0      2     638    2313
compacte      0    366    1329      0      1
minicar       4    398    7150      0      1
Routiere     32      0      1    1235    1222
sportive    404      0      1      0    4194
```

Figure 107 Situation2_NNET : Dessin de la matrice de mélange

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4329	0	2	638	2313	0,594479539
compacte	0	366	1329	0	1	0,215801887
minicar	4	398	7150	0	1	0,946643718
Routiere	32	0	1	1235	1222	0,495983936
sportive	404	0	1	0	4194	0,911937378
Précision	0,9077375	0,47905759	0,84286219	0,65936999	0,54249127	
taux d'erreur	0,2686706					

Tableau 5 Situation2_NNET : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9104 pour ce test de l'algorithme NNET.

```
> nn_prob <- predict(nnet, clients_Immatriculations_traité_ET, type="raw")
> nn_auc <- multiclass.roc(clients_Immatriculations_traité_ET$categories, nn_prob)
> print(nn_auc)

Call:
multiclass.roc.default(response = clients_Immatriculations_traité_ET$categories, predictor = nn_prob)

Data: multivariate predictor nn_prob with 5 levels of clients_Immatriculations_traité_ET$categories: berline, compacte, minicar,
Routière, sportive.
Multi-class area under the curve: 0.9104
```

Figure 108 Situation2_NNET : calcul de l'AUC

6.3 Situation 3 : supprimer la colonne taux et la colonne âge

Nous considérons le cas où l'algorithme Random Forest ne fonctionne pas en raison d'un trop grand nombre de catégories d'âge et, dans la situation 3, nous supprimons à la fois la colonne de taux et la colonne des âges.

```
> clients_Immatriculations_traité_EA <- subset(clients_Immatriculations_traité_EA , select = -taux)
> clients_Immatriculations_traité_ET <- subset(clients_Immatriculations_traité_ET , select = -taux)

<-->
> clients_Immatriculations_traité_EA <- subset(clients_Immatriculations_traité_EA , select = -age)
> clients_Immatriculations_traité_ET <- subset(clients_Immatriculations_traité_ET , select = -age)
```

Figure 109 Situation3_Séparation de l'échantillonnage d'apprentissage et test

6.3.1 Algorithme : C50

Nous exécutons à nouveau l'algorithme C50 et obtenons les résultats suivants :

```
> treec <- c5.0(categories ~ ., clients_Immatriculations_traité_EA)
> print(treec)

Call:
c5.0.formula(formula = categories ~ ., data = clients_Immatriculations_traité_EA)

Classification Tree
Number of samples: 55114
Number of predictors: 4

Tree size: 6

Non-standard options: attempt to group attributes

>
> test_treec <- predict(treec, clients_Immatriculations_traité_ET, type = "class")
> table(test_treec)
test_treec
berline compacte minicar Routière sportive
    4701      0    9247      0    9672
>
> table(clients_Immatriculations_traité_ET$categories, test_treec)
test_treec
berline compacte minicar Routière sportive
berline    4310      0      2      0    2970
compacte      0      0   1695      0      1
minicar        4      0   7548      0      1
Routière       1      0      1      0    2488
sportive     386      0      1      0    4212
```

Figure 110 Situation3_C50 résultat de prédiction

	berline	compacte	minicar	Routière	sportive	Rappel
berline	4310	0	2	0	2970	0,591870365
compacte	0	0	1695	0	1	0
minicar	4	0	7548	0	1	0,999338011
Routière	1	0	1	0	2488	0
sportive	386	0	1	0	4212	0,915851272
Précision	0,9168262	1	0,81626473	1	0,43548387	
taux d'erreur	0,3196444					

Tableau 6 Situation3_C50 : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,8686 pour ce test de l'algorithme C50.

```
> c_prob <- predict(treeC, clients_Immatriculations_Traite_ET, type = "prob")
> c_auc <- multiclass.roc(Clients_Immatriculations_Traite_ET$categories, c_prob)
> print(c_auc)

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

Data: multivariate predictor c_prob with 5 levels of clients_Immatriculations_Traite_ET$categories: berline, compacte, minicar, Routiere, sportive.
Multi-class area under the curve: 0.8686
```

```

Figure 111 Situation3\_C50 : calcul de l'AUC

### 6.3.2 Algorithme : Naviebayes

Nous exécutons à nouveau l'algorithme Naviebayes et obtenons les résultats suivants :

```
> table(nb_class)
nb_class
berline compacte minicar Routiere sportive
 6562 0 10576 0 6482
> table(Clients_Immatriculations_Traite_ET$categories, nb_class)
nb_class
berline compacte minicar Routiere sportive
berline 4441 0 848 0 1993
compacte 0 0 1695 0 1
minicar 1465 0 6087 0 1
Routiere 103 0 714 0 1673
sportive 553 0 1 0 2814
```

```

Figure 112 Situation3_Naivebayes résultat de prédiction

	berline	compacte	minicar	Routiere	sportive	Rappel
berline	4441	0	848	0	1993	0,609859929
compacte	0	0	1695	0	1	0
minicar	1465	0	6087	0	1	0,805904938
Routiere	103	0	714	0	1673	0
sportive	553	0	1	0	2814	0,835510689
Précision	0,6767754	1	0,65136437	1	0,43412527	
taux d'erreur	0,4040824					

Tableau 7 Situation3_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,8507 pour ce test de l'algorithme Naivebayes.

```
> nb_prob <- predict(nb, clients_Immatriculations_Traite_ET, type="prob")
Warning message:
predict.naive_bayes(): more features in the newdata are provided as there are probability tables in the object. calculation is performed based on features to be found in the tables.
> # Installation du package
> nb_auc <- multiclass.roc(Clients_Immatriculations_Traite_ET$categories, nb_prob)
> print(nb_auc)

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

Data: multivariate predictor nb_prob with 5 levels of clients_Immatriculations_Traite_ET$categories: berline, compacte, minicar, Routiere, sportive.
Multi-class area under the curve: 0.8507
```

```

Figure 113 Situation3\_Naivebayes : calcul de l'AUC

### 6.3.3 Algorithme : Random Forest

Nous exécutons à nouveau l'algorithme Random Forest et obtenons les résultats suivants :

```
> table(result.RF)
result.RF
berline compacte minicar Routiere sportive
 4707 0 9247 0 9666
> table(Clients_Immatriculations_traite_ET3$categories, result.RF)
 result.RF
berline compacte minicar Routiere sportive
berline 4311 0 2 0 2969
compacte 0 0 1695 0 1
minicar 4 0 7548 0 1
Routiere 3 0 1 0 2486
sportive 389 0 1 0 4209
```

Figure 114 Situation3\_Random Forest résultat de prédiction

|                      | berline          | compacte | minicar           | Routiere | sportive          | Rappel             |
|----------------------|------------------|----------|-------------------|----------|-------------------|--------------------|
| berline              | 4311             | 0        | 2                 | 0        | 2969              | <b>0,59200769</b>  |
| compacte             | 0                | 0        | 1695              | 0        | 1                 | <b>0</b>           |
| minicar              | 4                | 0        | 7548              | 0        | 1                 | <b>0,999338011</b> |
| Routiere             | 3                | 0        | 1                 | 0        | 2486              | <b>0</b>           |
| sportive             | 389              | 0        | 1                 | 0        | 4029              | <b>0,911744739</b> |
| <b>Précision</b>     | <b>0,91587</b>   | <b>1</b> | <b>0,81626473</b> | <b>1</b> | <b>0,42473118</b> |                    |
| <b>taux d'erreur</b> | <b>0,3221843</b> |          |                   |          |                   |                    |

Tableau 8 Situation3\_Random Forest : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,7011 pour ce test de l'algorithme Random Forest.

```
> rf_prob <- predict(RF, Clients_Immatriculations_traite_ET3, type="prob")
> RF_auc <- multiclass.roc(Clients_Immatriculations_traite_ET$categories, rf_prob)
> print (RF_auc)

call:
multiclass.roc.default(response = Clients_Immatriculations_traite_ET$categories, predictor = rf_prob)

Data: multivariate predictor rf_prob with 5 levels of Clients_Immatriculations_traite_ET$categories: berline, compacte, minicar, Routiere, sportive.
Multi-class area under the curve: 0.7011
>
```

Figure 115 Situation3\_Random Forest : calcul de l'AUC

### 6.3.4 Algorithme : NNET

Nous exécutons à nouveau l'algorithme Random Forest et obtenons les résultats suivants :

```

> ## ---- NNET ----
> nnet <- nnet(categories ~., Clients_Immatriculations_traite_EA, size=6, maxit=180, act.fct = "softmax")
weights: 107
initial value 111678.138098
iter 10 value 72020.396085
iter 20 value 41313.154000
iter 30 value 39407.564823
iter 40 value 37907.374832
iter 50 value 37528.422492
iter 60 value 37309.993639
iter 70 value 37183.601768
iter 80 value 37116.386144
iter 90 value 37026.968712
iter 100 value 36979.299039
iter 110 value 36965.294672
iter 120 value 36957.266693
iter 130 value 36953.987839
iter 140 value 36951.238544
iter 150 value 36933.625655
iter 160 value 36917.091182
iter 170 value 36899.380610
iter 180 value 36891.172077
final value 36891.172077
stopped after 180 iterations
> nn_class <- predict(nnet, Clients_Immatriculations_traite_ET, type="class")
> table(nn_class)
nn_class
berline minicar sportive
 4702 9247 9671
> table(Clients_Immatriculations_traite_ET$categories, nn_class)
 nn_class
 berline minicar sportive
berline 4306 2 2974
compacte 0 1695 1
minicar 4 7548 1
Routiere 3 1 2486
sportive 389 1 4209

```

Figure 116 Situation3\_NNET résultat de prédiction

|                      | berline          | minicar           | sportive         | Rappel            |
|----------------------|------------------|-------------------|------------------|-------------------|
| berline              | 4306             | 2                 | 2974             | <b>0,59132107</b> |
| compacte             | 0                | 1695              | 1                |                   |
| minicar              | 4                | 7548              | 1                | <b>0,99933801</b> |
| Routiere             | 3                | 1                 | 2486             |                   |
| sportive             | 389              | 1                 | 4209             | <b>0,91519896</b> |
| <b>Précision</b>     | <b>0,9157805</b> | <b>0,81626473</b> | <b>0,4352187</b> |                   |
| <b>taux d'erreur</b> | <b>0,3199407</b> |                   |                  |                   |

Tableau 9 Situation3\_NNET : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,871 pour ce test de l'algorithme NNET.

```
> nn_prob <- predict(nnet, clients_Immatriculations_traite_ET, type="raw")
> nn_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, nn_prob)
> print(nn_auc)

Call:
multiclass.roc.default(response = clients_Immatriculations_traite_ET$categories, predictor = nn_prob)

Data: multivariate predictor nn_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar, R0
utiere, sportive.
Multi-class area under the curve: 0.871
```

Figure 117 Situation3\_NNET : calcul de l'AUC

#### 6.4 Situation 4 : remplacer la colonne taux à la colonne taux\_classe

Dans la situation 4, nous reclassons les taux et remplaçons la colonne originale des taux pour la détection.

En examinant la distribution de taux dans le boxplot et les valeurs minimum, premier quartile, médiane, moyenne, troisième quartile et maximum résumées dans le résumé, nous décidons de créer une nouvelle colonne taux\_classe et diviser taux en 4 catégories : tauxbas : taux < 588 ; tauxmoyen : 588 <= taux <= 899 ; tauxeleve : 899 < taux <= 1144 ; tauxtreseleve : taux > 1144.

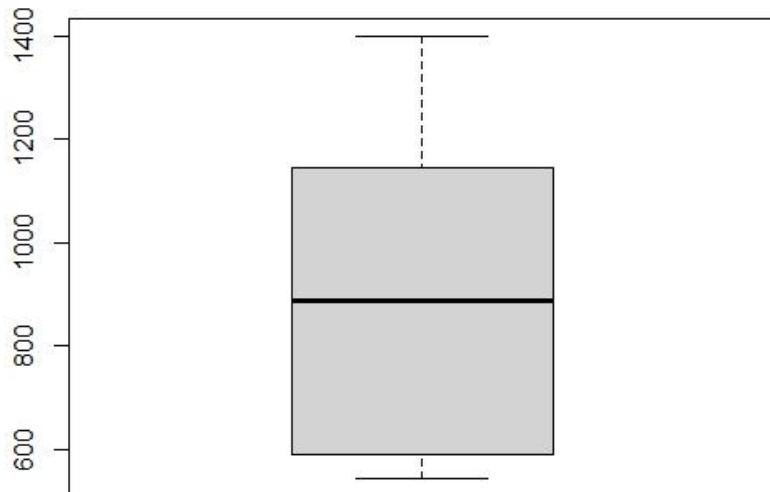


Figure 118 Le boxplot de taux

```
> summary(Clients_Immatriculations$taux)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
 544.0 588.0 888.0 899.2 1144.0 1399.0
```

Figure 119 Récapitulatif du taux

Nous supprimons ensuite la colonne taux d'origine et renommons taux\_classe en taux. Les données finales Clients\_Immatriculations, sont présentées ci-dessous.

```
> Clients_Immatriculations_traité <- Clients_Immatriculations
> Clients_Immatriculations_traité <- subset(Clients_Immatriculations_traité , select = -taux)
> names(Clients_Immatriculations_traité)[7] <- "taux"
> summary(Clients_Immatriculations_traité)
 categories age sexe situationfamiliale nbenfantsacharge deuxiemevoiture taux
berline :24326 25 : 2053 F:23653 celibataire:23389 0:35312 FALSE:68476 tauxbas :19894
compacte: 5706 29 : 2018 M:55081 Divorce : 36 1:13216 TRUE :10258 tauxeleve :18793
minicar :24963 23 : 1988 En Couple :50711 2:13129
Routiere: 8350 18 : 1986 Marie(e) : 530 3: 9192 tauxmoyen :20378
sportive:15389 21 : 1979 Seul : 212 4: 7885 tauxtreseleve :19669
 28 : 1978 Seule : 3856
(other):66732
```

Figure 120 Récapitulatif du Clients\_Immatriculations\_traité

Nous créons ensuite deux ensembles de données pour l'apprentissage automatique.

```
> Clients_Immatriculations_traité_EA <- clients_Immatriculations_traité[1:55114,]
> Clients_Immatriculations_traité_ET <- clients_Immatriculations_traité[55115:78734,]
```

Figure 121 Situation4\_Séparation de l'échantillonnage d'apprentissage et test

#### 6.4.1 Algorithme : C50

Nous exécutons à nouveau l'algorithme C50 et obtenons les résultats suivants :

```
> treec <- C5.0(categories ~., clients_Immatriculations_traite_EA)
> print(treec)

Call:
C5.0.formula(formula = categories ~ ., data = Clients_Immatriculations_traite_EA)

Classification Tree
Number of samples: 55114
Number of predictors: 6

Tree size: 50

Non-standard options: attempt to group attributes

>
> test_treec <- predict(treec, clients_Immatriculations_traite_ET, type = "class")
> table(test_treec)
test_treec
berline compacte minicar Routiere sportive
 4701 1586 7661 3875 5797
>
> table(clients_Immatriculations_traite_ET$categories, test_treec)
 test_treec
berline compacte minicar Routiere sportive
berline 4310 0 2 1076 1894
compacte 0 793 902 0 1
minicar 4 792 6756 0 1
Routiere 1 0 1 2267 221
sportive 386 1 0 532 3680
```

Figure 122 Situation4\_C50 résultat de prédiction

|                      | berline          | compacte   | minicar           | Routiere          | sportive          | Rappel             |
|----------------------|------------------|------------|-------------------|-------------------|-------------------|--------------------|
| berline              | 4310             | 0          | 2                 | 1076              | 1894              | <b>0,591870365</b> |
| compacte             | 0                | 793        | 902               | 0                 | 1                 | <b>0,467570755</b> |
| minicar              | 4                | 792        | 6756              | 0                 | 1                 | <b>0,894479015</b> |
| Routiere             | 1                | 0          | 1                 | 2267              | 221               | <b>0,910441767</b> |
| sportive             | 386              | 1          | 0                 | 532               | 3680              | <b>0,800173951</b> |
| <b>Précision</b>     | <b>0,9168262</b> | <b>0,5</b> | <b>0,88186921</b> | <b>0,58503226</b> | <b>0,63481111</b> |                    |
| <b>taux d'erreur</b> | <b>0,2461473</b> |            |                   |                   |                   |                    |

Tableau 10 Situation4\_C50 : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9399 pour ce test de l'algorithme C50.

```
> c_prob <- predict(treec, clients_Immatriculations_traite_ET, type = "prob")
>
> c_auc <- multiclass.roc(Clients_Immatriculations_traite_ET$categories, c_prob)
> print(c_auc)

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

Data: multivariate predictor c_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar, Routiere, sportive.
Multi-class area under the curve: 0.9399
```

Figure 123 Situation4\_C50 : calcul de l'AUC

#### 6.4.2 Algorithme : Naviebayes

Nous exécutons à nouveau l'algorithme Naviebayes et obtenons les résultats suivants :

```
> table(nb_class)
nb_class
berline compacte minicar Routiere sportive
 6901 1536 7128 1486 6569
> table(clients_Immatriculations_traité_ET$categories, nb_class)
nb_class
berline compacte minicar Routiere sportive
berline 4554 0 275 473 1980
compacte 0 746 949 0 1
minicar 1465 788 5299 0 1
Routiere 329 1 254 962 944
sportive 553 1 351 51 3643
```

Figure 124 Situation4\_Naivebayes : résultat de prediction

|                      | berline          | compacte          | minicar           | Routiere         | sportive          | Rappel             |
|----------------------|------------------|-------------------|-------------------|------------------|-------------------|--------------------|
| berline              | 4554             | 0                 | 275               | 473              | 1980              | <b>0,625377644</b> |
| compacte             | 0                | 746               | 949               | 0                | 1                 | <b>0,439858491</b> |
| minicar              | 1465             | 788               | 5299              | 0                | 1                 | <b>0,701575533</b> |
| Routiere             | 329              | 1                 | 254               | 962              | 944               | <b>0,386345382</b> |
| sportive             | 553              | 1                 | 351               | 51               | 3643              | <b>0,792128724</b> |
| <b>Precision</b>     | <b>0,6599044</b> | <b>0,48567708</b> | <b>0,74340629</b> | <b>0,6473755</b> | <b>0,55457452</b> |                    |
| <b>taux d'erreur</b> | <b>0,3563082</b> |                   |                   |                  |                   |                    |

Tableau 11 Situation4\_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9201 pour ce test de l'algorithme Naivebayes.

```
> nb_prob <- predict(nb, clients_Immatriculations_traité_ET, type="prob")
Warning message:
predict.naive_bayes(): more features in the newdata are provided as there are probability tables in the object. Calculation is performed
based on features to be found in the tables.
> # Installation du package
> nb_auc <- multiclass.roc(clients_Immatriculations_traité_ET$categories, nb_prob)
> print(nb_auc)

call:
multiclass.roc.default(response = clients_Immatriculations_traité_ET$categories, predictor = nb_prob)

Data: multivariate predictor nb_prob with 5 levels of clients_Immatriculations_traité_ET$categories: berline, compacte, minicar, Routier
e, sportive.
Multi-class area under the curve: 0.9201
```

Figure 125 Situation4\_Naivebayes : calcul de l'AUC

#### 6.4.3 Algorithme : Random Forest

L'algorithme Random Forest n'a pas pu être mis en œuvre en raison du trop grand nombre de catégories d'âge.

```
> ## ---- Random Forest ----
> clients_Immatriculations_traité_EA3 <- clients_Immatriculations_traité_EA
> clients_Immatriculations_traité_ET3 <- clients_Immatriculations_traité_ET
>
> RF <- randomForest(categories ~ ., clients_Immatriculations_traité_EA3)
Error in randomForest.default(m, y, ...) :
 Can not handle categorical predictors with more than 53 categories.
>
```

Figure 126 Formation de jeux d'apprentissage avec random forest

#### 6.4.4 Algorithme : NNET

Nous exécutons à nouveau l'algorithme NNET et obtenons les résultats suivants :

```
> ## ---- NNET ----
> net <- nnet(categories ~., Clients_Immatriculations_traité_EA, size=6, maxit=180, act.fct = "softmax")
weights: 521
initial value 107681.037570
iter 10 value 59255.075602
iter 20 value 38955.983072
iter 30 value 33287.453567
iter 40 value 31237.873370
iter 50 value 30114.015874
iter 60 value 29413.239679
iter 70 value 28528.228451
iter 80 value 27969.449176
iter 90 value 27491.079269
iter 100 value 27257.625170
iter 110 value 27043.181621
iter 120 value 26935.629881
iter 130 value 26888.758009
iter 140 value 26863.095594
iter 150 value 26838.638776
iter 160 value 26811.224149
iter 170 value 26793.678786
iter 180 value 26781.898261
final value 26781.898261
stopped after 180 iterations
> nn_class <- predict(nnet, Clients_Immatriculations_traité_ET, type="class")
> table(nn_class)
nn_class
berline compacte minicar Routiere sportive
 4735 1647 7602 3851 5785
> table(Clients_Immatriculations_traité_ET$categories, nn_class)
 nn_class
berline compacte minicar Routiere sportive
berline 4316 0 4 1074 1888
compacte 0 803 892 0 1
minicar 4 842 6705 1 1
Routiere 29 2 0 2237 222
sportive 386 0 1 539 3673
```

Figure 127 Situation4\_NNET résultat de prédiction

|                      | berline          | compacte          | minicar           | Routiere          | sportive          | Rappel             |
|----------------------|------------------|-------------------|-------------------|-------------------|-------------------|--------------------|
| berline              | <b>4316</b>      | 0                 | 4                 | 1074              | 1888              | <b>0,592694315</b> |
| compacte             | 0                | 803               | 892               | 0                 | 1                 | <b>0,473466981</b> |
| minicar              | 4                | 842               | 6705              | 1                 | 1                 | <b>0,887726731</b> |
| Routiere             | 29               | 2                 | 0                 | 2237              | 222               | <b>0,898393574</b> |
| sportive             | 386              | 0                 | 1                 | 539               | 3673              | <b>0,798651881</b> |
| <b>Précision</b>     | <b>0,911151</b>  | <b>0,48755313</b> | <b>0,88200474</b> | <b>0,58088808</b> | <b>0,63491789</b> |                    |
| <b>taux d'erreur</b> | <b>0,2491956</b> |                   |                   |                   |                   |                    |

Tableau 12 Situation4\_NNET : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9486 pour ce test de l'algorithme NNET.

```
> nn_prob <- predict(nnet, Clients_Immatriculations_traité_ET, type="raw")
> nn_auc <- multiclass.roc(Clients_Immatriculations_traité_ET$categories, nn_prob)
> print(nn_auc)

Call:
multiclass.roc.default(response = Clients_Immatriculations_traité_ET$categories, predictor = nn_prob)

Data: multivariate predictor nn_prob with 5 levels of Clients_Immatriculations_traité_ET$categories: berline, compacte, minicar, Routiere, sportive.
Multi-class area under the curve: 0.9486
```

Figure 128 Situation4\_NNET : calcul de l'AUC

## 6.5 Situation 5 : remplacer la colonne taux et la colonne âge à la colonne taux\_classe et âge\_classe

Dans la situation 5, nous reclassons les taux et remplaçons la colonne originale des taux et reclassons les âges et remplaçons la colonne originale des âges pour la détection.

La classification du taux\_classe reste inchangée par rapport à la situation4.

En examinant la distribution de âge dans le boxplot et les valeurs minimum, premier quartile, médiane, moyenne, troisième quartile et maximum résumées dans le résumé, nous décidons de créer une nouvelle colonne âge\_classe et diviser taux en 4 catégories : jeune : âge <27 ; âge moyen : 27<= âge <=44; adulte : 44< âge <= 51 et Aînés : âge > 51.

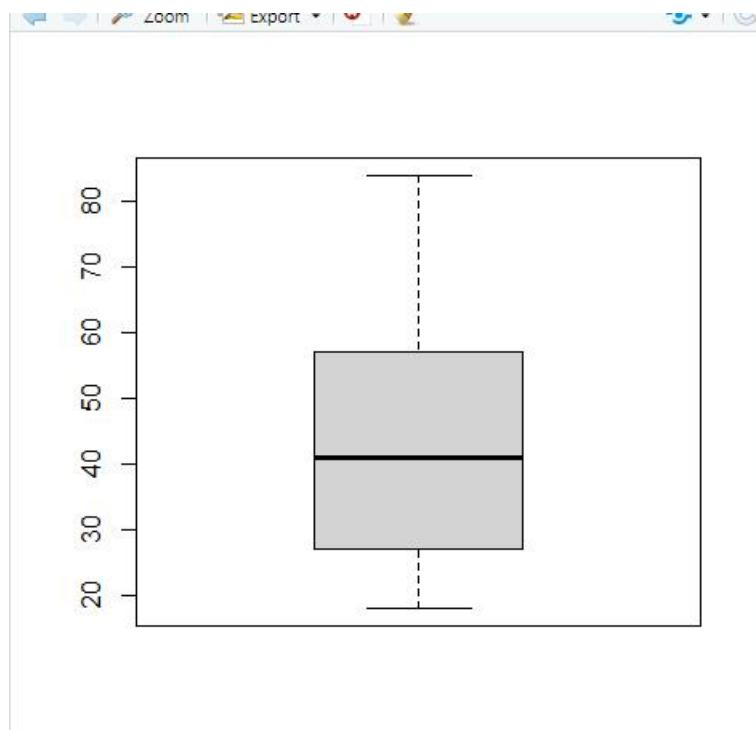


Figure 129 Le boxplot d'âge

```
> summary(Clients_Immatriculations$age)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
18.00 27.00 41.00 43.72 57.00 84.00
```

Figure 130 Récapitulatif d'âge

Nous supprimons ensuite la colonne taux et âge d'origine, renommons taux\_classe en taux et renommons age\_classe en age. Les données finales Clients\_Immatriculations, sont présentées ci-dessous.

```
> Clients_Immatriculations_traité <- Clients_Immatriculations
> Clients_Immatriculations_traité <- subset(Clients_Immatriculations_traité , select = -age)
> Clients_Immatriculations_traité <- subset(Clients_Immatriculations_traité , select = -taux)
> names(Clients_Immatriculations_traité)[6] <- "age"
> names(Clients_Immatriculations_traité)[7] <- "taux"
> summary(Clients_Immatriculations_traité)
 categories sexe situationfamiliale nbenfantsacharge deuxiemevoiture age taux
berline:24326 F:23653 Celibataire:23389 0:35312 FALSE:68476 adulte :17023 tauxbas :19894
compacte: 5706 M:55081 Divorce : 36 1:13216 TRUE :10258 agemoyen:23608 tauxeleve :18793
minicar:24963 En Couple :50711 2:13129 Aînés :18413 tauxmoyen :20378
Routiere: 8350 Marie(e) : 530 3: 9192 jeune :19690 tauxtreseleve :19669
sportive:15389 Seul : 212 4: 7885
seule : 3856
```

Figure 131 Récapitulatif du Clients\_Immatriculations\_traité

Nous créons ensuite deux ensembles de données pour l'apprentissage automatique.

```
|> Clients_Immatriculations_traite_EA <- Clients_Immatriculations_traite[1:55114,]
|> Clients_Immatriculations_traite_ET <- Clients_Immatriculations_traite[55115:78734,]
```

Figure 132 Situation5\_Séparation de l'échantillonnage d'apprentissage et test

### 6.5.1 Algorithme : C50

Nous exécutons à nouveau l'algorithme C50 mais nous pouvons voir que la taille de l'arbre = 0 et ne peut donc pas être prédite.

```
|> treec <- C5.0(categories ~ ., Clients_Immatriculations_traite_EA)
C50 code called exit with value 1
> print(treec)

Call:
C5.0.formula(formula = categories ~ ., data = Clients_Immatriculations_traite_EA)

Classification Tree
Number of samples: 55114
Number of predictors: 6

Tree size: 0

Non-standard options: attempt to group attributes

>
> test_treec <- predict(treec, Clients_Immatriculations_traite_ET, type = "class")
Error in predict.C5.0(treec, Clients_Immatriculations_traite_ET, type = "class") :
either a tree or rules must be provided
```

Figure 133 Situation5\_C50 code

### 6.5.2 Algorithme : Naviebayes

Nous exécutons à nouveau l'algorithme Naviebayes et obtenons les résultats suivants :

```
|> table(nb_class)
nb_class
berline compacte minicar Routiere sportive
 7171 964 7559 1389 6537
> table(Clients_Immatriculations_traite_ET$categories, nb_class)
nb_class
berline compacte minicar Routiere sportive
berline 4652 0 228 445 1957
compacte 0 455 1240 0 1
minicar 1465 507 5580 0 1
Routiere 418 1 246 843 982
sportive 636 1 265 101 3596
```

Figure 134 Situation5\_Naivebayes résultat de prédiction

| Naivebayes           |                  |                  |                   |                   |                   |                    |
|----------------------|------------------|------------------|-------------------|-------------------|-------------------|--------------------|
|                      | berline          | compacte         | minicar           | Routiere          | sportive          | Rappel             |
| berline              | 4652             | 0                | 228               | 445               | 1957              | <b>0,638835485</b> |
| compacte             | 0                | 455              | 1240              | 0                 | 1                 | <b>0,268278302</b> |
| minicar              | 1465             | 507              | 5580              | 0                 | 1                 | <b>0,738779293</b> |
| Routiere             | 418              | 1                | 246               | 843               | 982               | <b>0,338554217</b> |
| sportive             | 636              | 1                | 265               | 101               | 3596              | <b>0,781909111</b> |
| <b>Précision</b>     | <b>0,648724</b>  | <b>0,4719917</b> | <b>0,73819288</b> | <b>0,60691145</b> | <b>0,55009943</b> |                    |
| <b>taux d'erreur</b> | <b>0,3596105</b> |                  |                   |                   |                   |                    |

Tableau 13 Situation5\_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9156 pour ce test de l'algorithme Naivebayes.

```
> nb_prob <- predict(nb, clients_Immatriculations_traite_ET, type="prob")
Warning message:
predict.naive_bayes(): more features in the newdata are provided as there are probability tables in the object. Calculation is performed
based on features to be found in the tables.
> # Installation du package
> nb_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, nb_prob)
> print(nb_auc)

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

Data: multivariate predictor nb_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar, Routiere,
sportive.
Multi-class area under the curve: 0.9156
```

Figure 135 Situation5\_Naivebayes : calcul de l'AUC

### 6.5.3 Algorithme : Random Forest

Nous exécutons à nouveau l'algorithme Random Forest et obtenons les résultats suivants :

```
> table(result.RF)
result.RF
berline compacte minicar Routiere sportive
 4704 507 8740 4176 5493
> table(clients_Immatriculations_traite_ET$categories, result.RF)
 result.RF
 berline compacte minicar Routiere sportive
berline 4311 0 2 1177 1792
complate 0 250 1445 0 1
minicar 4 257 7291 0 1
Routiere 2 0 1 2280 207
sportive 387 0 1 719 3492
```

Figure 136 Situation5\_Random Forest résultat de prédiction

|                      | berline          | compacte          | minicar           | Routiere          | sportive          | Rappel             |
|----------------------|------------------|-------------------|-------------------|-------------------|-------------------|--------------------|
| berline              | 4311             | 0                 | 2                 | 1177              | 1792              | <b>0,59200769</b>  |
| compacte             | 0                | 250               | 1445              | 0                 | 1                 | <b>0,14740566</b>  |
| minicar              | 4                | 257               | 7291              | 0                 | 1                 | <b>0,965311797</b> |
| Routiere             | 2                | 0                 | 1                 | 2280              | 207               | <b>0,915662651</b> |
| sportive             | 387              | 0                 | 1                 | 719               | 3492              | <b>0,759295499</b> |
| <b>Précision</b>     | <b>0,9164541</b> | <b>0,49309665</b> | <b>0,83421053</b> | <b>0,54597701</b> | <b>0,63571819</b> |                    |
| <b>taux d'erreur</b> | <b>0,2538527</b> |                   |                   |                   |                   |                    |

Tableau 14 Situation5\_Random Forset : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,8698 pour ce test de l'algorithme Random Forest.

```
> rf_prob <- predict(RF, clients_Immatriculations_traite_ET, type="prob")
> RF_auc <- multiclass.roc(clients_Immatriculations_traite_ET$categories, rf_prob)
> print (RF_auc)

Call:
multiclass.roc.default(response = clients_Immatriculations_traite_ET$categories, predictor = rf_prob)

Data: multivariate predictor rf_prob with 5 levels of clients_Immatriculations_traite_ET$categories: berline, compacte, minicar, Routiere,
sportive.
Multi-class area under the curve: 0.8698
```

Figure 137 Situation5\_Random Forest : calcul de l'AUC

#### 6.5.4 Algorithme : NNET

Nous exécutons à nouveau l'algorithme NNET et obtenons les résultats suivants :

```
> ## ---- NNET ----
> nnet <- nnet(categories ~., clients_Immatriculations_traité_EA, size=6, maxit=180, act.fct = "softmax")
weights: 143
initial value 100381.863621
iter 10 value 50318.237937
iter 20 value 32443.574739
iter 30 value 30779.020228
iter 40 value 30172.668262
iter 50 value 29674.017134
iter 60 value 29309.269150
iter 70 value 28993.239366
iter 80 value 28774.197561
iter 90 value 28700.145272
iter 100 value 28641.475347
iter 110 value 28538.366939
iter 120 value 28476.199835
iter 130 value 28369.065421
iter 140 value 28295.596384
iter 150 value 28282.339549
iter 160 value 28271.813289
iter 170 value 28263.805416
iter 180 value 28257.130589
final value 28257.130589
stopped after 180 iterations
> nn_class <- predict(nnet, clients_Immatriculations_traité_ET, type="class")
> table(nn_class)
nn_class
berline compacte minicar Routiere sportive
 4701 1067 8180 4175 5497
> table(clients_Immatriculations_traité_ET$categories, nn_class)
 nn_class
berline compacte minicar Routiere sportive
berline 4310 0 2 1177 1793
compacte 0 521 1174 0 1
minicar 4 545 7003 0 1
Routiere 1 0 1 2280 208
sportive 386 1 0 718 3494
```

Figure 138 Situation5\_NNET résultat de prédiction

|                      | berline          | compacte          | minicar           | Routiere          | sportive          | Rappel             |
|----------------------|------------------|-------------------|-------------------|-------------------|-------------------|--------------------|
| berline              | 4310             | 0                 | 2                 | 1177              | 1793              | <b>0,591870365</b> |
| compacte             | 0                | 521               | 1174              | 0                 | 1                 | <b>0,307193396</b> |
| minicar              | 4                | 545               | 7003              | 0                 | 1                 | <b>0,927181252</b> |
| Routiere             | 1                | 0                 | 1                 | 2280              | 208               | <b>0,915662651</b> |
| sportive             | 386              | 1                 | 0                 | 718               | 3494              | <b>0,759730376</b> |
| <b>Précision</b>     | <b>0,9168262</b> | <b>0,48828491</b> | <b>0,85611247</b> | <b>0,54610778</b> | <b>0,63561943</b> |                    |
| <b>taux d'erreur</b> | <b>0,2545301</b> |                   |                   |                   |                   |                    |

Tableau 15 Situation5\_NNET : Calcul du rappel, de la précision et des taux d'erreur

Nous pouvons obtenir que l'AUC est égale à 0,9453 pour ce test de l'algorithme NNET.

```
> nn_prob <- predict(nnet, clients_Immatriculations_traité_ET, type="raw")
> nn_auc <- multiclass.roc(clients_Immatriculations_traité_ET$categories, nn_prob)
> print(nn_auc)

Call:
multiclass.roc.default(response = clients_Immatriculations_traité_ET$categories, predictor = nn_prob)

Data: multivariate predictor nn_prob with 5 levels of Clients_Immatriculations_traité_ET$categories: berline, compacte, minicar, Routier
e, sportive.
Multi-class area under the curve: 0.9453
```

Figure 139 Situation5\_NNET : calcul de l'AUC

## 7. Conclusion des quatre algorithmes

Nous comparons les valeurs AUCs et les taux d'erreur des quatre algorithmes.

Nous examinons les 5 classifications. La situation 1 ne supprime ni ne modifie les données ; la situation 2 supprime la colonne taux ; la situation 3 supprime les colonnes taux et âge ; la situation 4 reclasse le taux et la situation 5 reclasse le taux et l'âge. Nous combinons tous les cas pour produire un tableau, et les deux tableaux suivants montrent que l'algorithme NNET dans la situation 4 a la valeur AUC la plus élevée, c'est 0,9486. La différence entre le taux d'erreur de l'algorithme NNET et le taux d'erreur le plus bas dans ce cas est très faible. Par conséquent, nous choisissons l'algorithme NNET dans la situation 4 pour prédire les données de marketing.

| situation  | 1      | 2      | 3      | 4      | 5      |
|------------|--------|--------|--------|--------|--------|
| C50        | 0,9418 | 0,9039 | 0,8686 | 0,9399 |        |
| naivebayes | 0,9201 | 0,8908 | 0,8507 | 0,9201 | 0,9156 |
| randomFor  |        |        | 0,7011 |        | 0,8698 |
| nnet       |        | 0,9104 | 0,871  | 0,9486 | 0,9453 |

Tableau 16 Comparaison de l'AUC

| situation   | 1         | 2         | 3         | 4         | 5         |
|-------------|-----------|-----------|-----------|-----------|-----------|
| C50         | 0,2426334 | 0,2666384 | 0,3196444 | 0,2461473 |           |
| naivebayes  | 0,3651566 | 0,3475021 | 0,4040824 | 0,3563082 | 0,3596105 |
| randomFores |           |           | 0,3221843 |           | 0,2538527 |
| nnet        |           | 0,2686706 | 0,3199407 | 0,2491956 | 0,2545301 |

Tableau 17 Comparaison des taux d'erreur

## 8. Nettoyage de Marketing

Tout d'abord, nous renommons les noms des colonnes de Marketing.

```
> ## renommer des colonnes
> names(Marketing)[1] <- "id"
> names(Marketing)[2] <- "age"
> names(Marketing)[3] <- "sexe"
> names(Marketing)[4] <- "taux"
> names(Marketing)[5] <- "situationfamiliale"
> names(Marketing)[6] <- "nbenfantscharge"
> names(Marketing)[7] <- "deuxiemevoiture"
```

Figure 140 Renommer des colonnes de Marketing

Ensuite, nous transformons les valeurs au format correct.

```
> Marketing$age <- as.factor(Marketing$age)
> Marketing$sex <- as.factor(Marketing$sex)
> Marketing$taux <- as.integer(Marketing$taux)
> Marketing$situationfamiliale <- as.factor(Marketing$situationfamiliale)
> Marketing$nbenfantscharge <- as.factor(Marketing$nbenfantscharge)
> Marketing$deuxiemevoiture<- as.factor(Marketing$deuxiemevoiture)
>
```

Figure 141 Convert to format demandé

Afin de pouvoir utiliser l'algorithme NNET de Situation4 pour prévoir le marketing, nous devons maintenir la cohérence des types et des structures de données. Nous créons donc une colonne Taux\_classe pour le marketing afin de reclasser le taux, qui doit être cohérent avec la classification Taux de Situation4. Nous supprimons ensuite la colonne Taux d'origine et renommons la colonne Taux\_classe en taux.

Enfin, supprimez la première colonne id, corrigez les caractères erronés et changez la forme des données en facteur.

```
> Marketing_traite$taux_classe<- ifelse(Marketing_traite$taux <= 588,"tauxbas",
+ ifelse(Marketing_traite$taux > 588& Marketing_traite$taux <=899,"tauxmoyen",
+ ifelse(Marketing_traite$taux > 899& Marketing_traite$taux <=1144,"tauxeleve",
+ ifelse(Marketing_traite$taux > 1144,"tauxtreseleve ","Rien"))))
>
> Marketing_traite$taux_classe <- as.factor(Marketing_traite$taux_classe)
> Marketing_traite <- Marketing_traite[,4]
> names(Marketing_traite)[7] <- "taux"
> Marketing_traite$situationfamiliale <- gsub("Celibataire", "Celibataire", Marketing_traite$situationfamiliale)
> Marketing_traite$age <- as.factor(Marketing_traite$age)
> Marketing_traite$sex <- as.factor(Marketing_traite$sex)
> Marketing_traite$nbenfantscharge <- as.factor(Marketing_traite$nbenfantscharge)
> Marketing_traite$taux <- as.factor(Marketing_traite$taux)
```

Figure 142 Crédation de class pour taux, corrigé de valeurs, et convert to format demandé

Le tableau final de Marketing nettoyés est présenté ci-dessous.

```
> summary(Marketing_traite)
 age sexe situationfamiliale deuxiemevoiture nbenfantscharge taux
 22 : 2 F: 9 Length:20 FALSE:15 0:12 tauxbas :14
35 : 2 M:11 Class :character TRUE : 5 1: 1 tauxeleve : 2
59 : 2 Mode :character 2: 3 2: 3 tauxmoyen : 2
19 : 1 3: 4 tauxtreseleve : 2
21 : 1
26 : 1
(other):11
```

Figure 143 Récapitulatif du tableau des Immatriculations nettoyés

## 9. Prédiction

Après avoir nettoyé les données de marketing, nous avons utilisé l'algorithme NNET de Situation4 pour faire des prédictions. Les prédictions sont ensuite affichées dans un tableau (classpred).

```
> classpred <- predict(nnet, Marketing_traite, type = "class")
> ##classpred <- predict(nnet, Marketing_traite)
> table(classpred)
classpred
berline compacte minicar Routiere sportive
 4 4 6 4 2
```

Figure 144 Prédiction de Marketing avec NNET

Enfin, les prédictions sont fusionnées avec le tableau Marketing original, comme le montre la figure.

```
> resultat <- data.frame(Marketing_traite, classpred)
>
> view(resultat)
> show(resultat)
 age sexe situationfamiliale deuxiemevoiture nbenfantsacharge taux classpred
1 48 M celibataire FALSE 0 tauxbas minicar
2 26 F En Couple TRUE 3 tauxbas berline
3 80 M En Couple FALSE 3 tauxbas berline
4 64 M celibataire FALSE 0 tauxbas compacte
5 22 M En Couple TRUE 3 tauxbas berline
6 54 F En Couple TRUE 3 tauxbas berline
7 59 M En Couple TRUE 0 tauxmoyen minicar
8 22 M En Couple FALSE 1 tauxbas Routiere
9 35 M celibataire FALSE 0 tauxmoyen minicar
11 34 F En Couple FALSE 0 tauxeleve sportive
12 60 M En Couple TRUE 0 tauxbas minicar
13 59 F En Couple FALSE 2 tauxbas Routiere
14 58 M En Couple FALSE 0 tauxtreseleve sportive
15 21 F celibataire FALSE 0 tauxtreseleve minicar
16 55 M celibataire FALSE 0 tauxbas compacte
17 19 F celibataire FALSE 0 tauxbas compacte
18 27 F En Couple FALSE 2 tauxbas Routiere
19 43 F celibataire FALSE 0 tauxbas minicar
20 35 M celibataire FALSE 0 tauxbas compacte
21 79 F En Couple FALSE 2 tauxeleve Routiere
```

Figure 145 Prédiction finale de Marketing