

# Rapport et descriptif de la thèse d'innovation (TPA) intitulée :

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

Réalisé par le **groupe 3** dont les membres sont:

- BERTRAMO Quentin
- DRIDI Nour
- MALVESIN Anthony
- VESNAT Jules

Supervisé par :

Mr MOPOLO Gabriel  
Mr PASQUIER Nicolas  
Mr SIMONIAN Sergio  
Mr WINCKLER Marco  
Mr GALLI Gregory

Année universitaire: 2022/2023  
**M2 MIAGE MBDS**

## CONTEXTE DU PROJET

Ce projet vient répondre au besoin exprimé par un concessionnaire automobile. En effet, nous avons été contactés par ce dernier dans le but de l'aider à mieux cibler les véhicules susceptibles d'intéresser ses clients en fonction de divers critères (Age, sexe, situation familiale, taux d'endettement etc.).

Notre client sera satisfait si nous lui proposons un moyen afin :

- Qu'un vendeur puisse en quelques secondes évaluer le type de véhicule le plus susceptible d'intéresser des clients qui se présentent dans la concession.
- Qu'il puisse envoyer une documentation précise sur le véhicule le plus adéquat pour des clients sélectionnés par son service marketing.

Les données fournies étant de taille significative, nous avons en ce sens fait appel à nos connaissances en traitement des Big Data avec Hive, Hadoop et les notions de mapper et reducer, l'analyse des données avec des algorithmes R et finalement la visualisation des données sous des représentations graphiques.

La solution que nous proposons par la présente, repose sur quatre piliers à savoir:

- La réalisation d'un datalake à partir des fichiers .csv fournis par le concessionnaire. Cette partie concerne l'import de ces données à la machine virtuelle, leur insertion dans différents SGBD complètement indépendants et les regrouper au final avec Hive.
- L'adaptation du fichier co2 avec un programme map/reduce avec Hadoop
- L'analyse de ces données avec R
- La datavisualisation des données pour une meilleure prise de décision, et une représentation graphique claire et interprétable.

## 1. Projet Big Data Analytics avec les Lacs de Données et les tables externes : BDA/DL

*Au sein de cette partie, nos travaux ont été réalisés au sein de la machine virtuelle Vagrant conçue et développée par Mr. SIMONIAN.*

### Import des données et la création des tables externes:

Pour mener à bien notre projet et répondre au besoin du concessionnaire, il fallait construire dans un premier temps un lac de données. Plus prosaïquement dit, le lac de données fait référence à un moyen de stockage volumineux des données massives et hétérogènes dont nous disposons. Nous avons, en ce sens, opté pour le lac de données HIVEQL. Ici, le moteur SQL HIVE d'Hadoop représente le frontal de notre lac.

Sachant que nos données brutes sont les fichiers .csv fournis par le concessionnaire, il fallait utiliser ces fichiers et assurer leur persistance à l'aide des SGBD. Ainsi, nous avons réparti notre data sur trois bases de données différentes comme suit :

1. Les données clients et immatriculations sont stockées dans le serveur **Oracle NoSQL**.
2. Les données catalogue sont stockées dans le serveur **MongoDB**
3. Les données Co2 et Marketing sont placées dans le système de fichiers distribué **HDFS** comme demandé.

A la suite de la création des données et de leur persistance au sein des différentes bases de données, nos efforts ont été dédiés à la création des tables externes depuis Hive. Ces cinq tables pointent, respectivement, vers les différentes données qui existent sur les différents SGBD. Plus simplement dit, Hive va pouvoir accéder à l'ensemble des données depuis un seul endroit (virtuel) malgré leur différents emplacements, grâce au hive access driver.

La figure ci-dessous permet de traduire ce qui a été mentionné précédemment en illustrant l'architecture adoptée pour le présent projet:

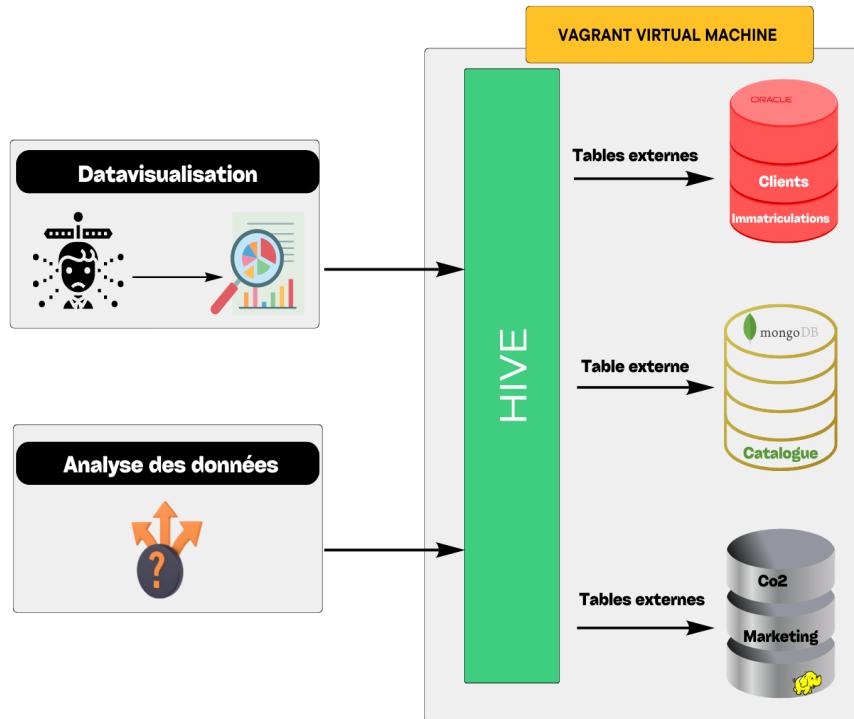


Figure 1: Architecture du projet

Le [readme](#) de la section datalake de notre repository, énumère l'ensemble des étapes suivies ainsi que les scripts lancés, menant à la mise en place du lac des données, son alimentation et son extraction.

## Difficultés rencontrées:

Nous souhaitions, à travers cette section, faire remonter les difficultés que nous avons rencontrées au sein de cette partie dans une optique d'amélioration.

Dans un premier lieu, et malgré la performance de la machine virtuelle que nul ne remet en question, réussir à installer cette dernière et l'ensemble de ses composants, représentait une tâche laborieuse. Celle-ci nous a coûté approximativement 21 jh. Voici quelques exemples des points bloquants rencontrés:

- *Vagrant n'a pas pu monter les dossiers partagés de VirtualBox:* Cette erreur est généralement liée à l'extension de VirtualBox. Malgré l'installation de celle-ci, le problème n'était pas résolu. La seule solution de notre côté était de détruire la VM et de la réinstaller.

```
=> oracle-21c-vagrant: Mounting shared folders...
    oracle-21c-vagrant: /vagrant => C:/vagrant/vagrant-projects/OracleDatabase/21.3.0
Vagrant was unable to mount VirtualBox shared folders. This is usually
because the filesystem "vboxsf" is not available. This filesystem is
made available via the VirtualBox Guest Additions and kernel module.
Please verify that these guest additions are properly installed in the
guest. This is not a bug in Vagrant and is usually caused by a faulty
Vagrant box. For context, the command attempted was:

mount -t vboxsf -o uid=1000,gid=1000,_netdev vagrant /vagrant

The error output from the command was:

/sbin/mount.vboxsf: mounting failed with the error: No such device
```

- Mysql,dont Hive a besoin pour le metastore, ne démarre pas en lançant hadoop. La solution proposée étant d'exécuter le provisioning des prérequisities, mais en vain. Il a fallu détruire la VM et la réinstaller.

```
-- 
-- The unit mysqld.service has entered the 'failed' state with result 'exit-code'.
Nov 27 00:03:59 oracle-21c-vagrant systemd[1]: Failed to start MySQL 8.0 database server.
-- Subject: Unit mysqld.service has failed
-- Defined-By: systemd
-- Support: https://support.oracle.com
-- 
-- Unit mysqld.service has failed.
```

- L'installation de Mongo est incomplète.

```
Last login: Tue Nov 22 00:48:23 2022 from 10.0.2.2
[vagrant@oracle-21c-vagrant ~]$ sudo systemctl start mongod
Failed to start mongod.service: Unit mongod.service not found.
[vagrant@oracle-21c-vagrant ~]$ sudo systemctl start mongod
Failed to start mongod.service: Unit mongod.service not found.
[vagrant@oracle-21c-vagrant ~]$
```

- L'installation de Hadoop est incomplète et comme le montre la capture ci-dessous, les commandes de provisioning ne permettent pas de résoudre le problème.

```
E:\final\vagrant-projects\OracleDatabase\21.3.0>start-dfs.sh
'start-dfs.sh' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

E:\final\vagrant-projects\OracleDatabase\21.3.0>start-dfs.sh
'start-dfs.sh' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

E:\final\vagrant-projects\OracleDatabase\21.3.0>vagrant provision --provision-with scripts/03_install_hadoop.sh
=> oracle-21c-vagrant: Running provisioner: scripts/03_install_hadoop.sh (shell)...
  oracle-21c-vagrant: Running: inline script
    oracle-21c-vagrant: [03_install_hadoop.sh:15]  [INFO] Remove previous Hadoop installation
    oracle-21c-vagrant: /vagrant/scripts/03_install_hadoop.sh: line 16: HADOOP_HOME: parameter null or not set
The SSH command responded with a non-zero exit status. Vagrant
assumes that this means the command failed. The output for this command
should be in the log above. Please read the output to determine what
went wrong.

E:\final\vagrant-projects\OracleDatabase\21.3.0>start-dfs.sh
'start-dfs.sh' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.
```

- Hive ne trouve pas le .jar du connecteur MongoDB malgré la présence des dossiers .zip dans le répertoire. La solution était de les ajouter à la main avec des commandes spécifiques que nous avons mentionnées dans la partie [In case of error due to MongoStorageHandler](#):

```
Error: Error while compiling statement: FAILED: SemanticException Cannot find class 'com.mongodb.hadoop.hive.MongoStorageHandler' (state=42000,code=40000)
0: jdbc:hive2://localhost:10000> CREATE EXTERNAL TABLE IF NOT EXISTS CATALOGUE_EXTERNE_MG (
  > CATALOGUEID INTEGER,
  > MARQUE STRING,
  > NOM STRING,
  > PUISSANCE INTEGER,
  > LONGUEUR STRING,
  > NB_PLACES INTEGER,
  > NB_PORTES INTEGER,
  > COULEUR STRING,
  > OCCASION BOOLEAN,
  > PRIX FLOAT,
  > MOYENNE_COUT_ENERGIE FLOAT,
  > MOYENNE_REJET FLOAT,
  > MOYENNE_BONUS_MALUS FLOAT
  > )
  > STORED BY 'com.mongodb.hadoop.hive.MongoStorageHandler'
  > WITH SERDEPROPERTIES('mongo.columns.mapping'='{"CATALOGUEID ":"_CATALOGUEID ", "MARQUE ":"MARQUE ", "NOM ":"NOM ", "PUISSEANCE ":"PUISSEANCE ", "LONGUEUR ":"LONGUEUR ", "NB_PLACES ":"NB_PLACES ", "NB_PORTES ":"NB_PORTES ", "COULEUR ":"COULEUR ", "OCCASION ":"OCCASION ", "Prix ":"PRIX ", "MOYENNE_COUT_ENERGIE ":"MOYENNE_COUT_ENERGIE ", "MOYENNE_REJET ":"MOYENNE_REJET ", "MOYENNE_BONUS_MALUS ":"MOYENNE_BONUS_MALUS "}')
  > TBLPROPERTIES('mongo.url'='mongodb://localhost:27017/mbds_22.catalogue');
Error: Error while compiling statement: FAILED: SemanticException Cannot find class 'com.mongodb.hadoop.hive.MongoStorageHandler' (state=42000,code=40000)
0: jdbc:hive2://localhost:10000>
```

Une fois l'installation réussie à 100%, une autre anomalie apparaît. En effet, certains composants de la machine virtuelle cessent de fonctionner notamment la connexion aux serveurs locaux hive et kv qui n'aboutissait pas.

```
[vagrant@oracle-21c-vagrant ~]$ beeline -u jdbc:hive2://localhost:10000 vagrant
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 2.3.9)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 2.3.9 by Apache Hive
0: jdbc:hive2://localhost:10000> |
```

Après plusieurs tentatives, la seule solution est de les réinstaller en ré-exécutant leurs scripts de provisioning correspondants. Dans notre cas ça concernait les scripts d'installation de Hive, Hadoop, Spark et KV store. Par

conséquent, nous perdons toutes nos tables créées sur hive, kv et hdfs. Il fallait donc tout refaire à plusieurs reprises.

En outre, nous tenons à exprimer nos profonds remerciements envers Mr SIMONIAN pour sa disponibilité, sa patience et son aide tout au long de la réalisation de cette partie.

## 2. HADOOP MAP REDUCE

### Introduction

En ce qui concerne le module Hadoop, deux parties distinctes s'ouvrent à nous notamment:

- La manipulation/refactoring du fichier CO2.csv
- L'ajout des moyennes calculées par marque dans Catalogue.csv.

Pour la première partie, un programme Hadoop map/reduce en java a été implémenté, utilisant de toute évidence le fichier CO2.csv ainsi qu'une liste de Marques (makes.txt) qui nous permettra de séparer la marque du modèle. Le programme permettant d'intégrer ces moyennes dans le fichier Catalogue.csv a été réalisé, dans un second temps, avec un script R.

### Adaptation du fichier CO2.csv

*Plusieurs manipulations ont été effectuées sur le fichier CO2.csv*

Tout d'abord, nous avons séparé la première colonne qui auparavant contenait la marque et le modèle fusionnés. Désormais, nous disposons de deux colonnes contenant respectivement la marque et le modèle séparément.

Pour ce faire, le programme Hadoop prend en 2ème argument un fichier qui liste toutes les marques de voiture à travers le monde : makes.txt (le 1er argument étant le fichier CO2.csv). Ce fichier a été récupéré sur internet [à cette adresse](#).

La première manipulation JAVA visait à mettre en majuscule toutes les marques de manière à s'aligner sur les marques du fichier CO2.csv.

Le fichier CO2.csv comporte plusieurs erreurs de syntaxe dont :

- Des virgules et des doubles quotes en trop:

```
1 usage
public CarWritable(String string) { //AUDI E-TRON SPORTBACK 55 (408ch) quattro,-6NBSP000€NBS
    string = string.replaceAll( regex: ", ", replacement: " " ); //fixed: virgule en trop
    string = string.replaceAll( regex: "\"", replacement: "" ); //fixed: double quote en trop
```

- Des valeurs manquantes pour le bonus/malus remplacées par 0:

```
public int get_bonus_malus_serialized(String bon_mal) { //"-6 000€ 1"
    String str = bon_mal.split( regex: "€" )[0]; //+/-....
    try {
        str = str.split( regex: "\\" )[1].trim(); //Positive value
    } catch (Exception e) { }
    str = str.replaceAll( regex: "[^0-9-]", replacement: "" ); //"-6000" //Accept only number and minus symbol

    if (str.equals("-") || str.equals("")) return 0; //Empty values
    return Integer.parseInt(str);
}
```

- Des valeurs manquantes pour le rejet CO2 remplacées par 0

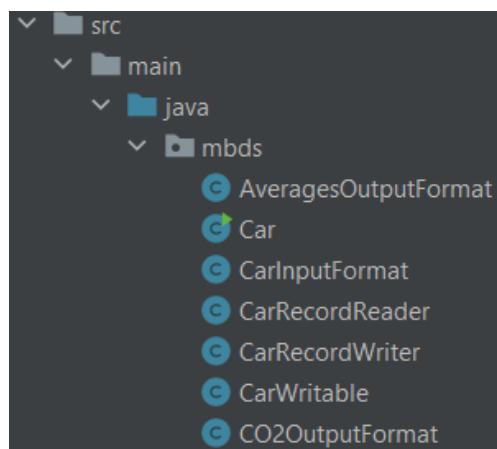
```
rejet = 0;
try {
    rejet = Integer.parseInt(elems[2]);
} catch (Exception e) { }
```

# Programme et exécution Hadoop

Le programme va avoir en sortie 2 fichiers :

- average\_per\_brand.csv -> Toutes les marques avec leurs moyennes
- co2\_reformatted.csv -> Le fichier CO2 corrigé

Voici l'architecture du programme map/reduce :



Dans le driver nous utilisons les MultipleOutputs objects afin de séparer les 2 fichiers en sortie:

```

FileInputFormat.addInputPath(job, new Path(co2File));
FileOutputFormat.setOutputPath(job,new Path( pathString: "/car-map-reduce-output"));
MultipleOutputs.addNamedOutput(job, namedOutput: "averages", AveragesOutputFormat.class , LongWritable.class, Text.class);
MultipleOutputs.addNamedOutput(job, namedOutput: "co2", CO2OutputFormat.class, LongWritable.class, Text.class);
    
```

Pour récupérer les marques du fichier makes.txt nous utilisons les outils du filesystem puis les intégrons dans la configuration map reduce pour pouvoir les réutiliser dans le map.

```

FileSystem fs = new Path(makes_txt).getFileSystem(conf);
Path hdfsreadpath = new Path(makes_txt); // p-e le folder path ?
FSDataInputStream inputStream = fs.open(hdfsreadpath);
String makesList = IOUtils.toString(inputStream, encoding: "UTF-8");

conf.set("makesList", makesList); //Store makes.txt list in configuration
    
```

Pour séparer la marque et le modèle, pour chaque row (Voiture) dans le map, nous comparons la marque du fichier CO2.csv et les marques du fichier makes.txt.

```
public void setRightProperties(List<String> makesList) {
    for (String brand : makesList) { //For each make of makes.txt file
        brand = brand.trim(); //Remove white space before and after
        if (marqueModele.contains(brand + " ") && marqueModele.length() > 0) {
            this.brand = brand;
            this.modele = marqueModele.replaceFirst(brand, replacement: "").trim();
        }
    }
}
```

Nous avons opté pour une solution fiable et bien plus lisible. Ainsi, au lieu de transférer des chaînes de caractères entre le map/shuffle/reduce, nous avons créé un objet voiture.

```
@Override
public void readFields(DataInput in) throws IOException {
    String line = in.readUTF();
    System.out.println("READING LINE " + line); //DS,DS3 CROSSBACK E-Tense (136ch),-6000,0,251
    String[] elems = line.split( regex: "\t");
    brand = elems[0]; //MERCEDES AMG...
    System.out.println("MARQUE: " + brand); //DS
    modele = elems[1]; //MERCEDES AMG...
    System.out.println("MODELE: " + modele);
    bonus_malus = Integer.parseInt(elems[2]); //-6000
    System.out.println("BONUS: " + bonus_malus);
    rejet = Integer.parseInt(elems[3]);
    System.out.println("REJET: " + rejet);
    cout_energie = Integer.parseInt(elems[4]);
    System.out.println("COUT: " + cout_energie);
}
```

Une fois l'objet Voiture complété avec les bonnes valeurs, nous utilisons comme clé la marque de voiture et en valeur l'objet voiture :

```
context.write(new Text(car.brand), car);
```

Quant au reduce, nous obtenons tous les modèles par marques. Nous calculons donc la moyenne et en profitons pour write tous les objets *Car* proprement dans un autre fichier en sortie.

```
public void reduce(Text key, Iterable<CarWritable> values, Context context) throws IOException, InterruptedException {
    // ex. key = "MERCEDES"
    // ex. values = Object CarWritable
    if (key.equals(new Text("999"))) {
        mos.write(namedOutput: "co2", new Text(string: "Marque"), columns_co2, baseOutputPath: "co2_reformatted");
        mos.write(namedOutput: "averages", new Text(string: "Marque"), columns_averages, baseOutputPath: "averages_per_brand");
    } else {
        System.out.println("Key: " + key + "\\n");
        int count = 0;
        int total_cout_energie = 0;
        int total_rejet = 0;
        int total_bonus_malus = 0;

        Iterator<CarWritable> i = values.iterator();
        while (i.hasNext()) {
            CarWritable car = i.next();
            total_cout_energie += car.cout_energie;
            total_rejet += car.rejet;
            total_bonus_malus += car.bonus_malus;
            System.out.println("Car: " + car.get_serialized());
            count++;
            mos.write(namedOutput: "co2", key, new Text(car.get_attributes()), baseOutputPath: "co2_reformatted"); //Reformatted file
        }
        int moyenne_cout_energie = total_cout_energie / count;
        int moyenne_rejet = total_rejet / count;
        int moyenne_bonus_malus = total_bonus_malus / count;
        mos.write(namedOutput: "averages", key, new Text(string: String.valueOf(moyenne_cout_energie) + "," + String.valueOf(moyenne_rejet) + "," + String.valueOf(moyenne_bonus_malus)));
    }
}
```

## Programme R

Dans le but d'intégrer les moyennes par marque dans le fichier Catalogue, nous avons décidé d'écrire un programme en R, langage qui correspond tout à fait à la manipulation et au traitement de données.

Dans un premier temps, le script prend en considération la première ligne comme le nom des colonnes pour éviter de faire des calculs inutiles :

`df -> Dataframe average_per_brand.csv`

```
header.true <- function(df) {
  names(df) <- as.character(unlist(df[1,]))
  df[-1,]
}
averages_per_brand <- header.true(averages_per_brand)
```

Nous créons les nouvelles colonnes à ajouter dans le fichier dataframe Catalogue :

```
empty_cols <- c("Moyenne cout energie", "Moyenne Rejet CO2", "Moyenne Bonus Malus")
Catalogue[ , empty_cols] <- NA
```

Pour finir, nous parcourons toutes les lignes du fichier Catalogue et nous ajoutons, pour chaque marque, la moyenne associée venant du fichier `average_per_brand.csv`. Nous exportons par conséquent ce dernier sous le nom de `Catalogue_with_averages.csv` :

```
for(i in 1:nrow(Catalogue)) {
  #Get row
  row <- Catalogue[i,]

  #Add value from averages_per_brand file to the good column id field
  row[10] <- averages_per_brand$"Moyenne cout energie"[averages_per_brand$"Marque"==toupper(row[1])]
  row[11] <- averages_per_brand$"Moyenne rejet CO2"[averages_per_brand$"Marque"==toupper(row[1])]
  row[12] <- averages_per_brand$"Moyenne Bonus Malus"[averages_per_brand$"Marque"==toupper(row[1])]

  #Replace modified row into Catalogue dataframe
  Catalogue[i,] <- row
}

write.csv(Catalogue,"D:\\Users\\Anthony\\Desktop\\M2_miage\\S1\\tpa\\Catalogue_with_averages.csv", row.names = FALSE, quote=FALSE)
```

### 3. Techniques de Data Visualisation

#### Triple-axis graph

**Library utilisée:** ChartJS

**Web documentation:** <https://www.chartjs.org/docs/latest/>

**Chaîne de traitement:**

- Récupération des données de Catalogue.csv avec les catégories de voitures
- Conversion CSV -> JSON
- Suppression des accents et/ou suppression des caractères spéciaux

**Utilisateurs visés:**

- Conseiller
- Client

**Objectif de la visualisation:**

L'objectif de cette visualisation est la mise en parallèle de plusieurs types de données utiles pour le choix d'un véhicule. Les données sont regroupées par catégorie de véhicules et par marque.

**Tâches utilisateurs:**

L'utilisateur peut choisir la catégorie de voiture à mettre en évidence dans le graphique

**Attributs utilisés:**

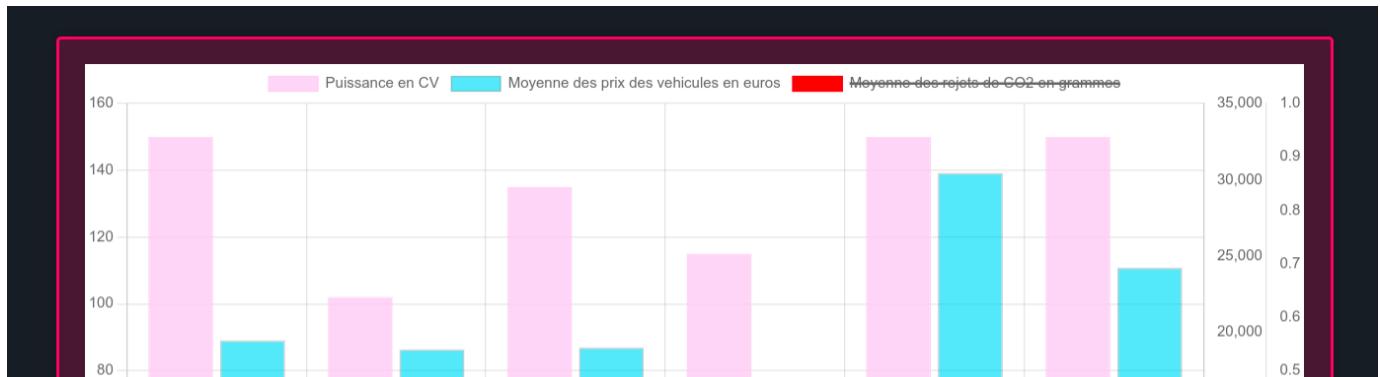
- Moyenne des rejets CO2 par marque (calculer avec Hadoop)
- Prix des véhicules (pour le calcul de la moyenne)
- Puissance du véhicule

**Filtres:**

Le graphique est dynamiquement adapté à la catégorie choisie par le client. En effet le type de catégorie est placé dans les paramètres URL:

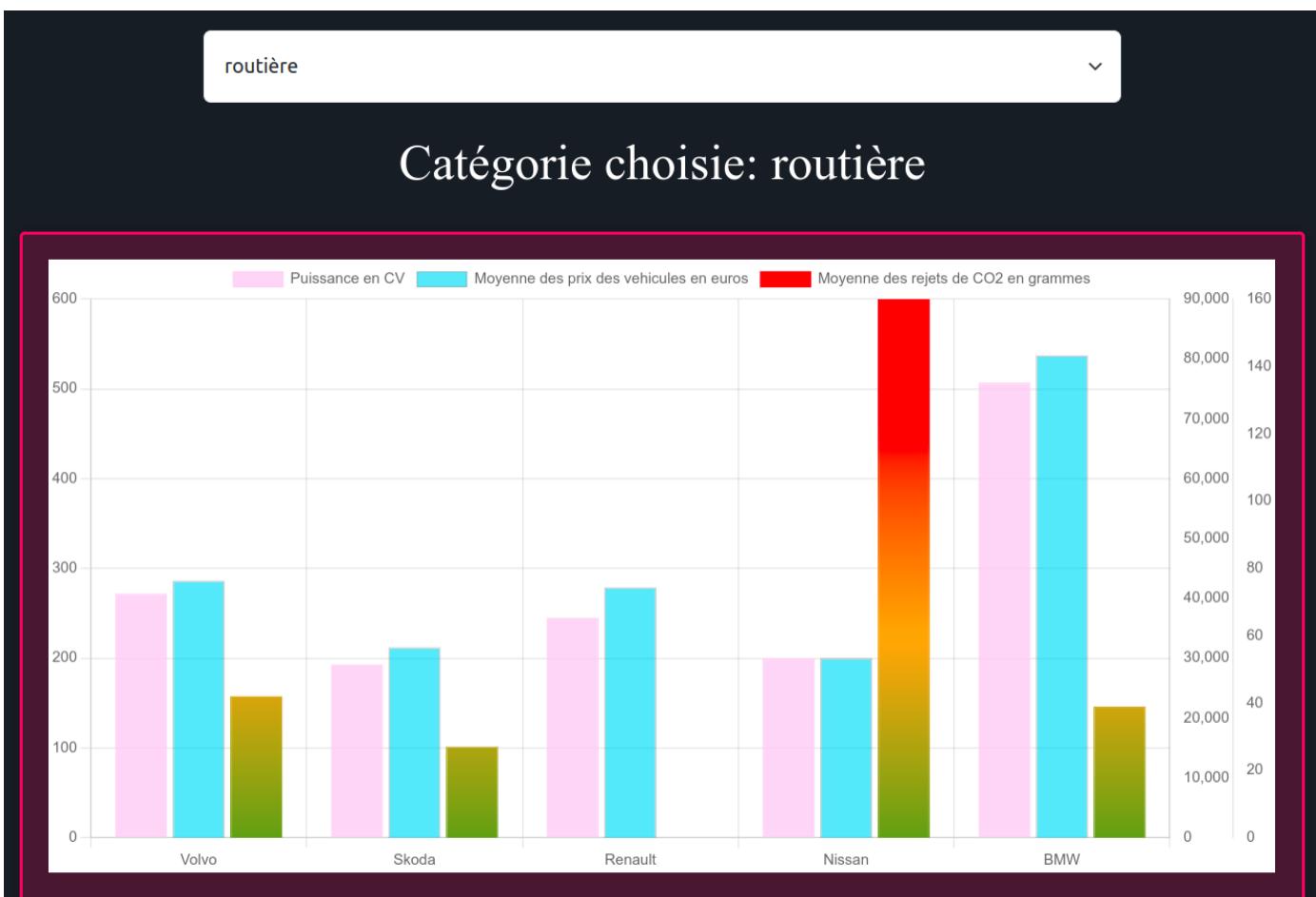


La représentation autorise l'utilisateur à supprimer ou ajouter certaines colonnes qui intéressent ou non le client. A titre d'exemple, un client qui ne prend pas en compte le rejet CO2 de son prochain véhicule peut appliquer ce filtre:



### Explication:

Ici le graphique fait un parallèle entre la consommation, le prix et la puissance des véhicules. En effet ces attributs sont conjointement liés. Plus un véhicule est puissant, plus il est cher et son rejet CO2 sera élevé. Ce graphique est d'autant plus intéressant pour les marques qui promeuvent l'utilisation de l'électricité plutôt que l'essence. Le rejet CO2 sera faible (pour les hybrides) et nul pour les véhicules 100% électrique.





## Tree

**Library utilisée:** TreantJS

**Web documentation:** <https://fperucic.github.io/treant-js/>

**Chaîne de traitement:**

- Récupération des donnée de Catalogue.csv avec les catégories de voitures
- Conversion CSV -> JSON
- Suppression des accents et/ou suppression des caractère spéciaux
- Récupération des logos des images via une banque d'image locale:  
<https://github.com/filippofilip95/car-logos-dataset/tree/master/logos/thumb>
- Regroupement par marque puis par catégorie

**Utilisateurs visés:**

- Conseiller

**Objectif de la visualisation:**

L'objectif de cette visualisation est d'avoir une vue globale de toutes les marques et modèles que compose le catalogue, pouvoir faire des parallèles entre les modèles de différentes catégories et marques sur la même page.

**Tâches utilisateurs:**

L'utilisateur déplie l'arbre de sélection

**Attributs utilisés:**

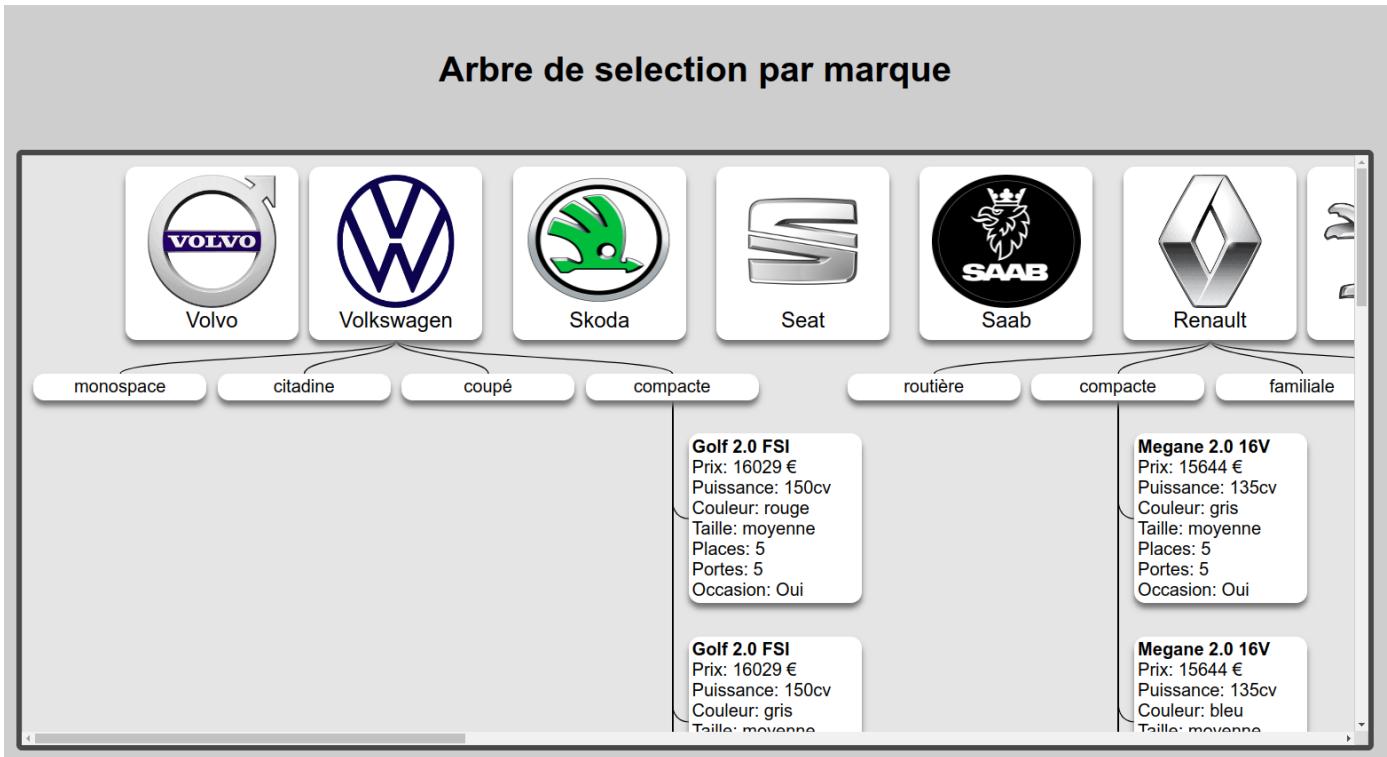
- Nom des marques (Noeud parent)
- Catégorie par marque
- Nom des modèles disponibles
- Tous les attributs des modèles (prix, couleur, longueur, etc...)

**Navigation:**

Déploiement de l'arbre en cliquant sur la marque ou le modèle.

De la vue global puis on précise (Marque -> Catégorie -> Modèles)

Capture d'écran de l'application:



## Matrix Table

**Library utilisée:** Koolchart

**Web documentation:** <https://www.koolchart.com/>

### Chaîne de traitement:

- Récupération des données de Catalogue.csv avec les catégories de voitures
- Conversion CSV -> JSON
- Suppression des accents et/ou suppression des caractères spéciaux
- Regroupement par marque puis par catégorie

### Utilisateurs visés:

- Conseiller

### Objectif de la visualisation:

L'objectif de cette visualisation est d'avoir une vue globale de toutes les marques et modèles que compose le catalogue, d'avoir une idée des prix et du nombre de modèles existant par marque et catégorie.

### Attributs utilisés:

- Nom des marques
- Nom des catégories
- Nombre des modèles disponibles
- Tous les attributs des modèles (prix, couleur, longueur, etc...)

### Navigation

L'utilisateur accède au détail des modèles en survolant la cellule à visualiser.

Marque	Modèle	Puissance	Longueur	Nb de places	Nb de portes	Couleur	Occasion	Prix
Nissan	Primera	1.6 109	longue	5	5	noir	false	18650
Nissan	Primera	1.6 109	longue	5	5	bleu	false	18650
Nissan	Primera	1.6 109	longue	5	5	rouge	false	18650
Nissan	Primera	1.6 109	longue	5	5	gris	false	18650
Nissan	Primera	1.6 109	longue	5	5	blanc	false	18650

**10 cars**

<b>10 cars</b>	<b>10 cars</b>		<b>5 cars</b>	
----------------	----------------	--	---------------	--

## Capture d'écran de la data visualisation:



# Parallel Coordinates Plot

**Library utilisée:** D3JS

**Web documentation:** <https://d3-graph-gallery.com/>

**Chaîne de traitement:**

- Récupération des donnée de Catalogue.csv avec les catégories de voitures
- Conversion CSV -> JSON
- Suppression des accents et/ou suppression des caractère spéciaux

**Utilisateurs visés:**

- Conseiller
- Vendeur

**Objectif de la visualisation:**

L'objectif de cette visualisation est d'avoir une vision globale de la comparaison des modèles de voitures et de leurs prix selon les attributs de la voiture : puissance, nombre de places, nombre de portes, couleurs...

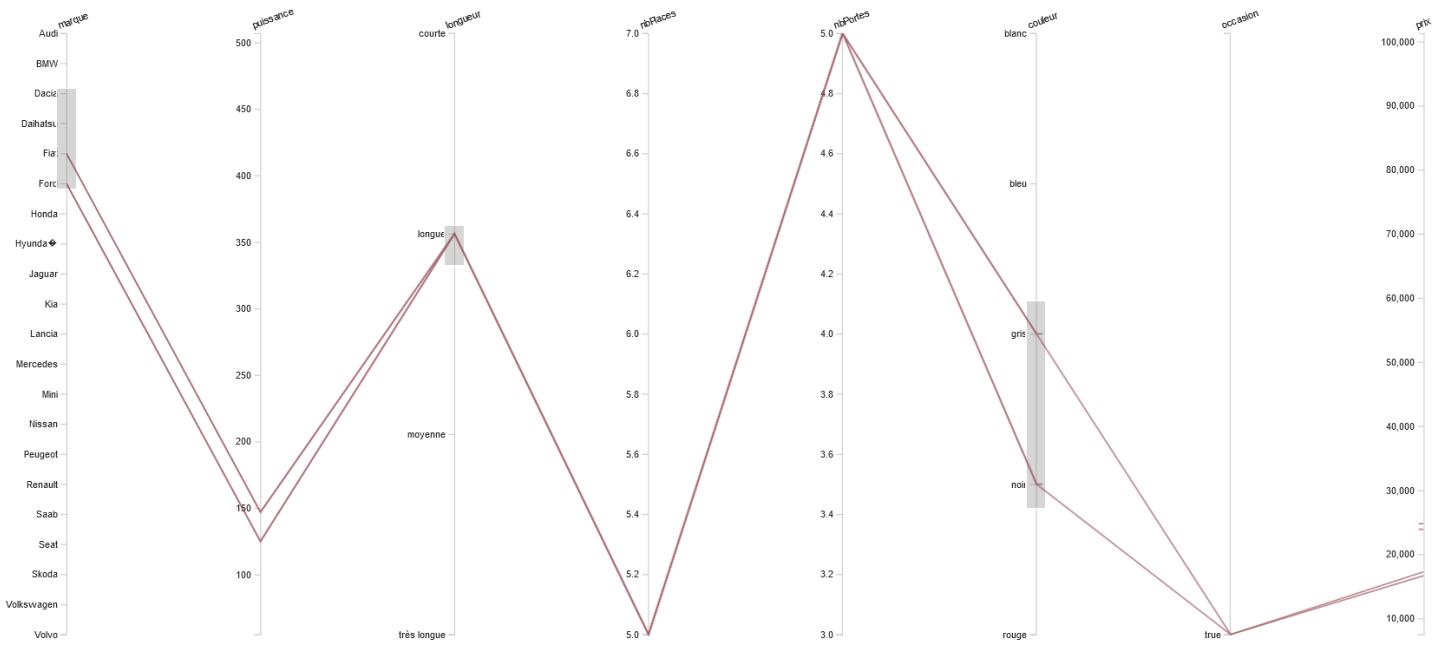
**Attributs utilisés:**

- Nom des marques
- Prix
- Tous les attributs des modèles (couleur, longueur, etc...)

**Navigation:**

Il est possible de cibler uniquement les éléments/attributs qui nous intéressent en créant des curseurs sur les colonnes avec la souris. On peut sélectionner plusieurs attributs différents sur différentes colonnes pour peaufiner la sélection, les curseurs des colonnes créés avec la souris peuvent être déplacés de haut en bas et être agrandis et rétrécis avec la souris.

## Capture d'écran de la data visualisation:



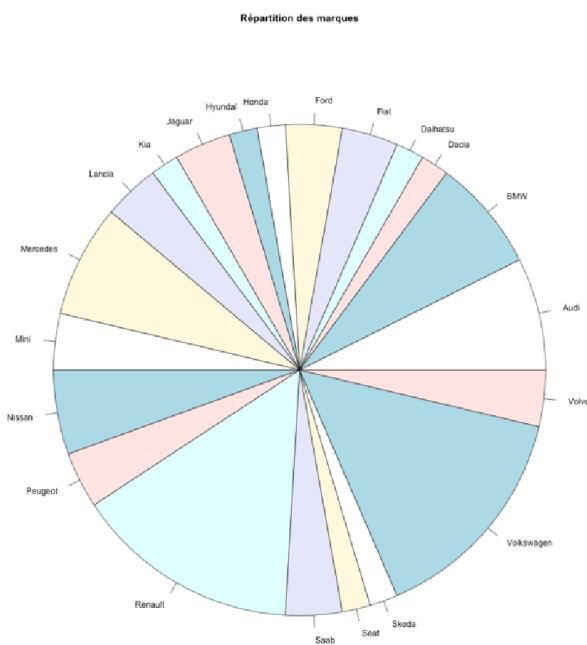
## 4. Analyse des Données

Cette partie est dédiée à l'analyse des données massives dont nous disposons et vise à aider le concessionnaire dans sa recherche de voiture susceptible de correspondre au profil de son client.

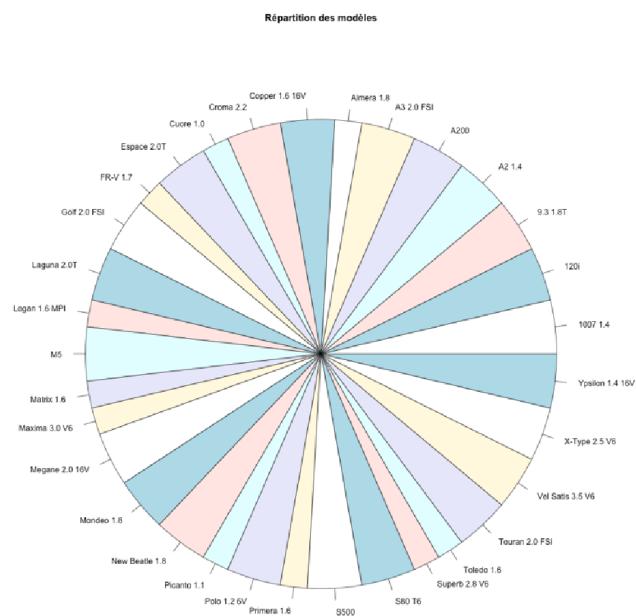
### Affichage des effectifs

Tout d'abord, nous avons affiché les effectifs grâce à des pie charts :

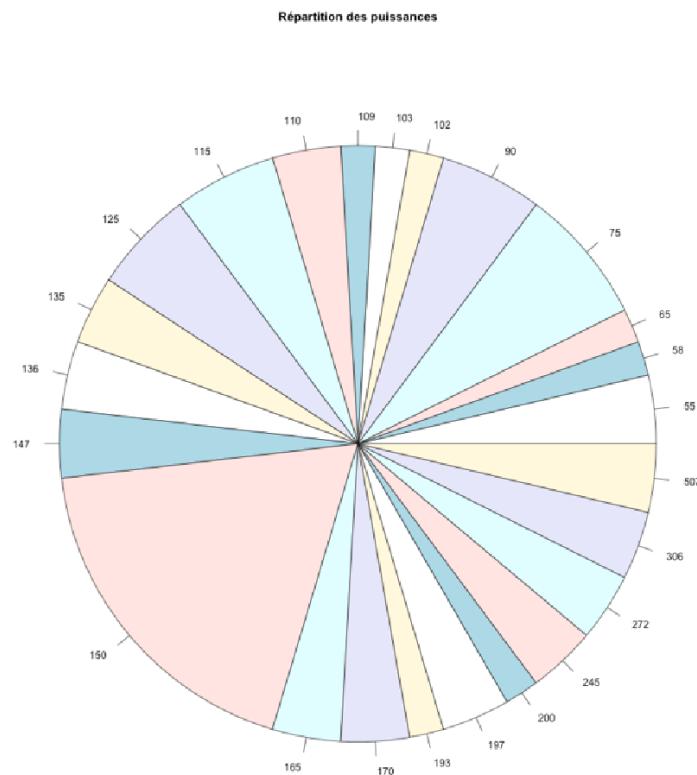
1. Marques :



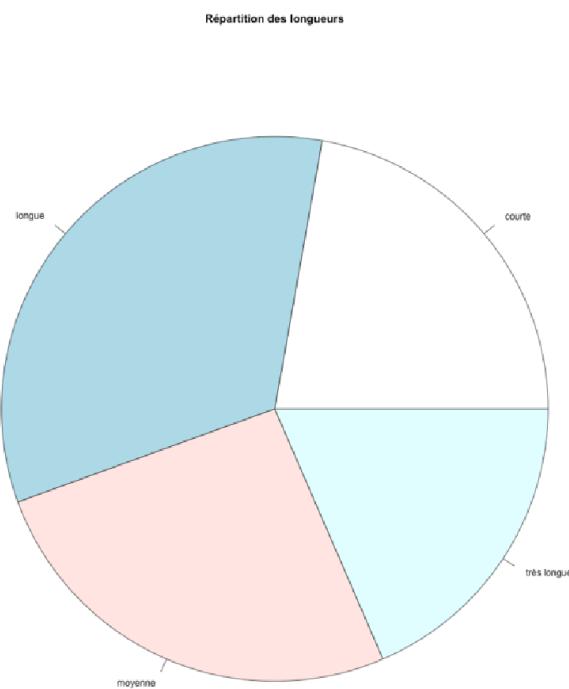
2. Modèles :



### 3. Puissances :

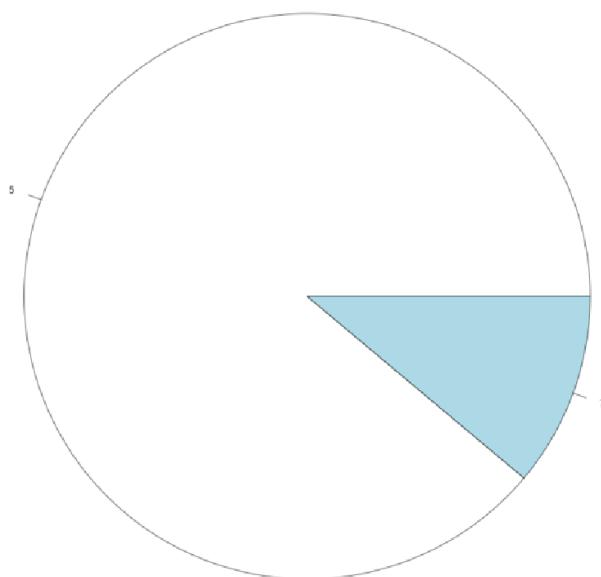


### 4. Longueurs :



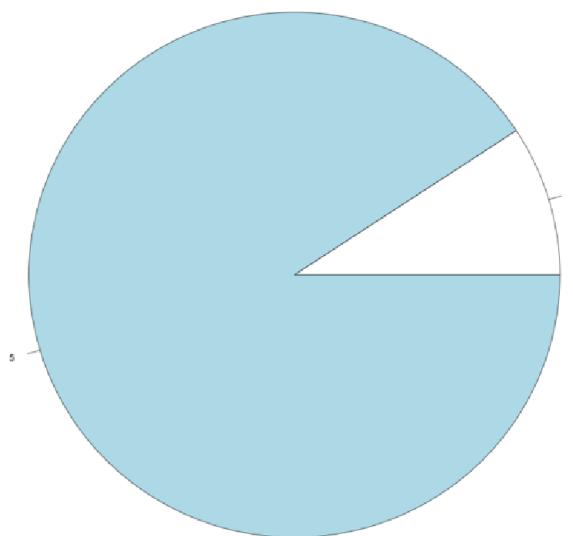
5. Nombre de places :

Répartition du nombre de places

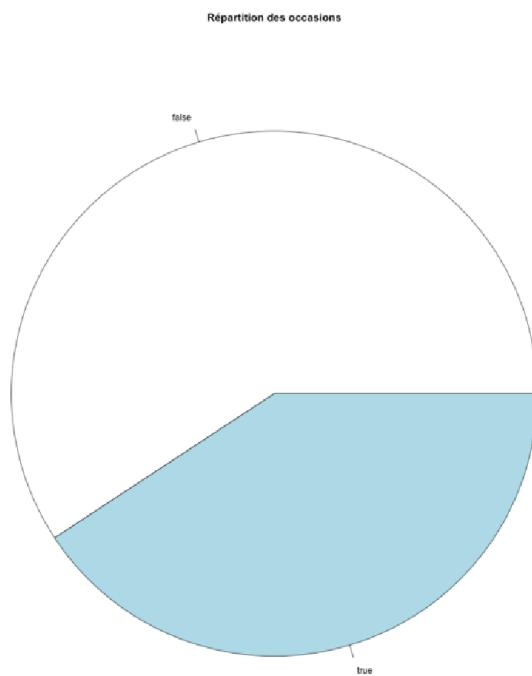


6. Nombre de portes :

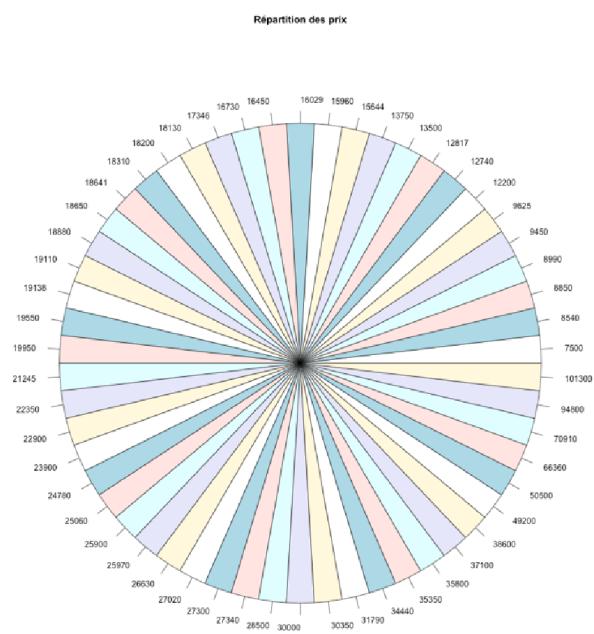
Répartition du nombre de portes



## 7. Occasions :



## 8. Prix :

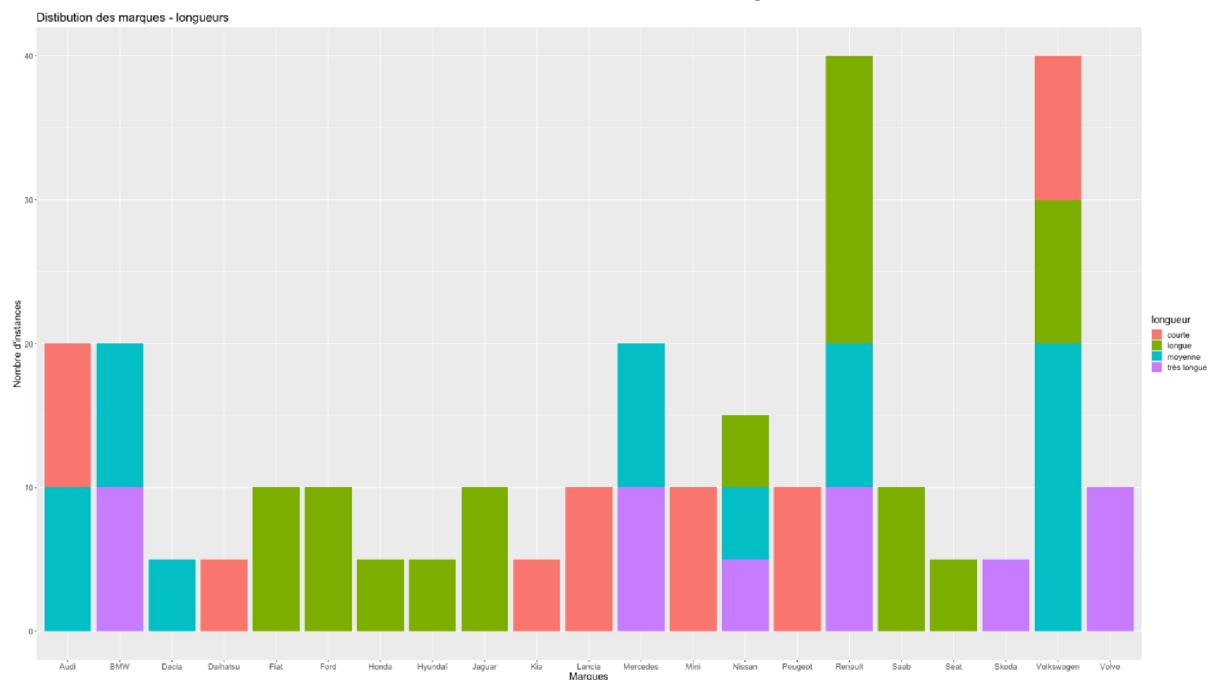


## Variables liées

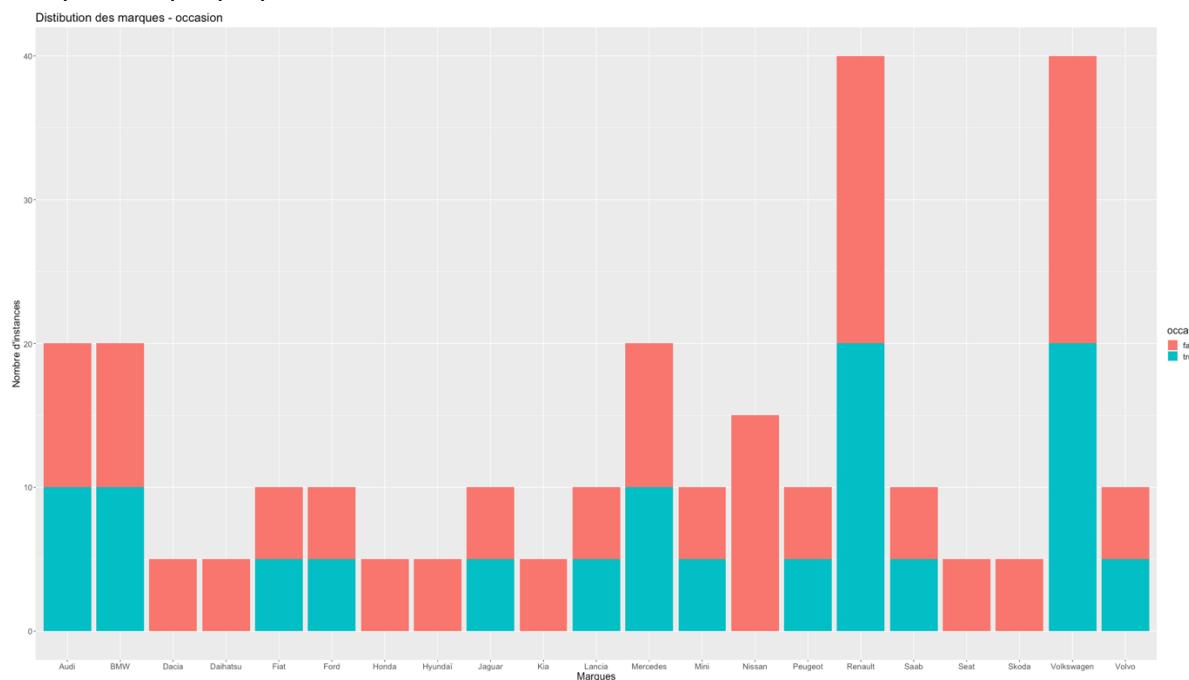
Nous avons ensuite cherché à identifier quelles variables étaient liées, grâce à des histogrammes d'effectif des variables.

### 1. Marques

On voit clairement que chaque marque propose diverses gammes de taille de voiture :



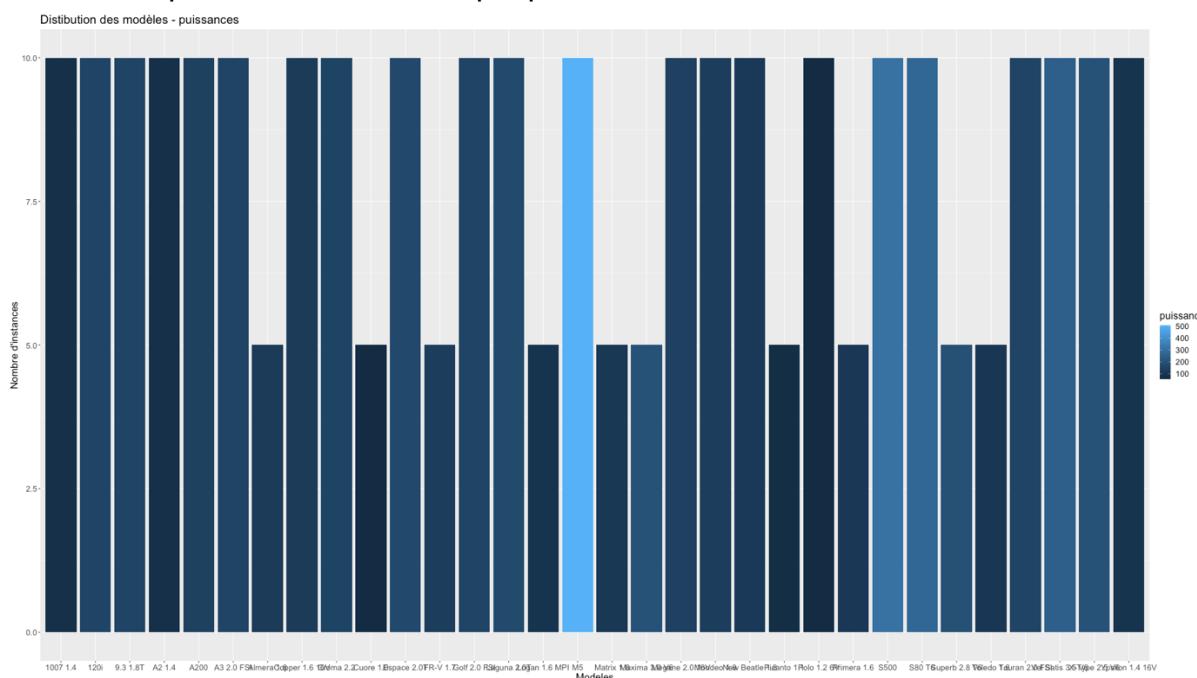
On remarque, de surcroît, que la colonne occasion n'a pas d'intérêts particuliers puisque chaque marque propose des modèles neufs et d'occasion :



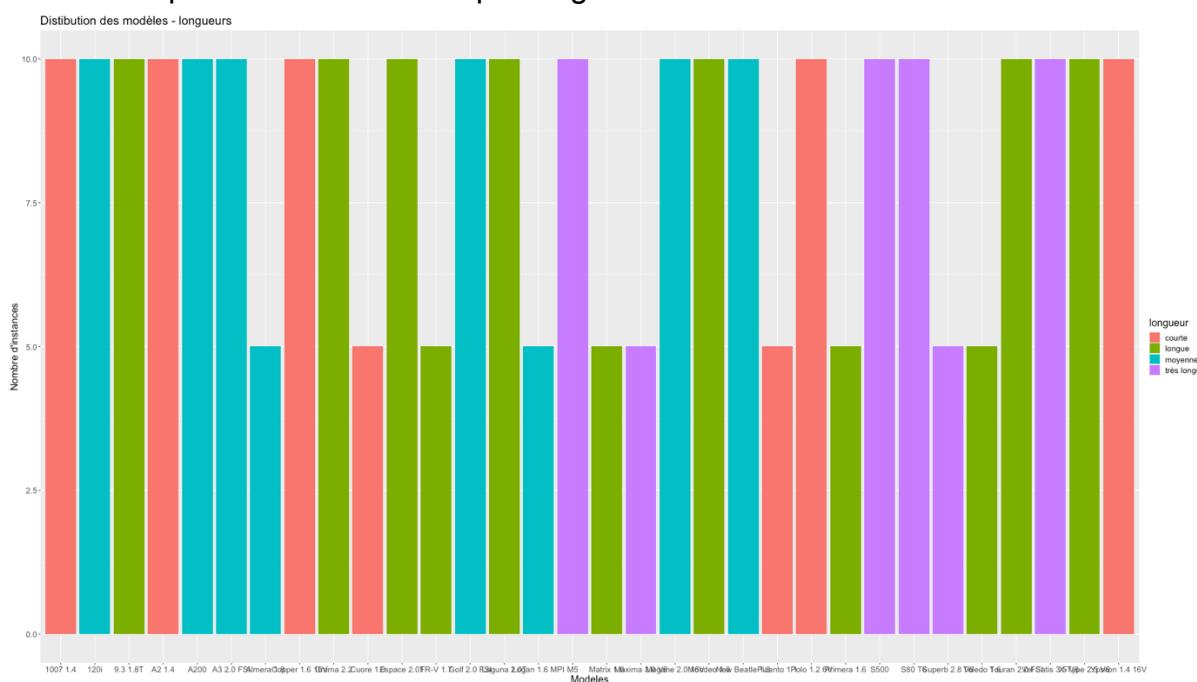
## 2. Modèles :

Grâce à ces différentes distributions, on voit bien que :

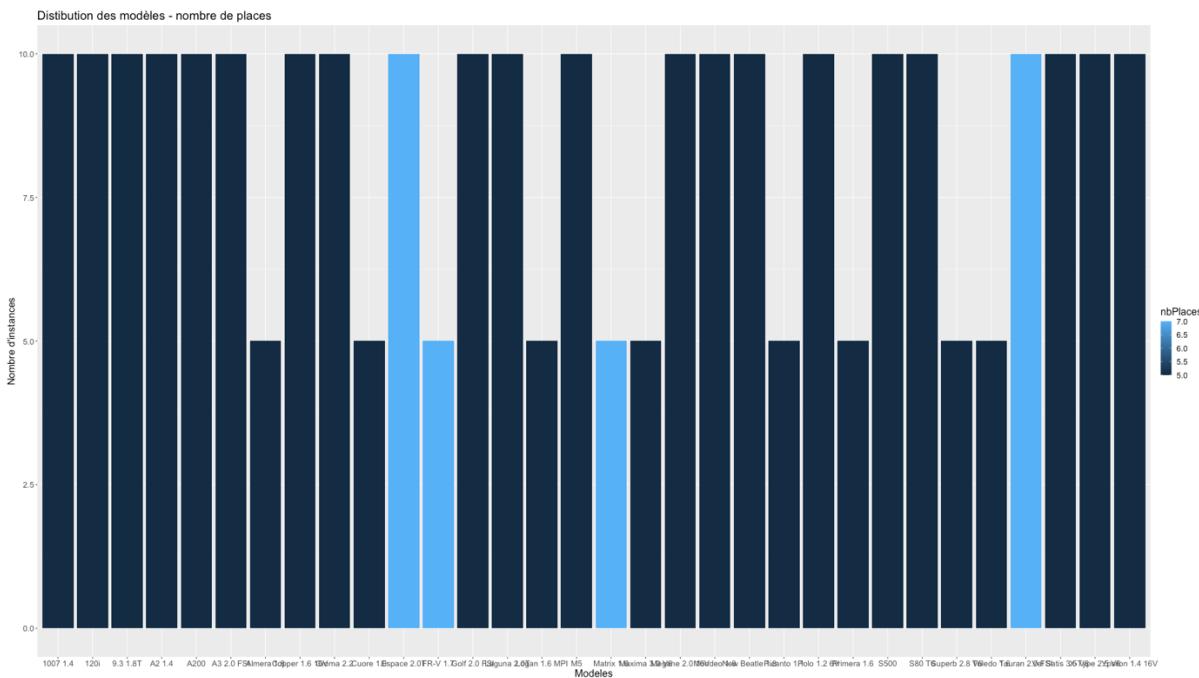
- Chaque modèle a une unique puissance :



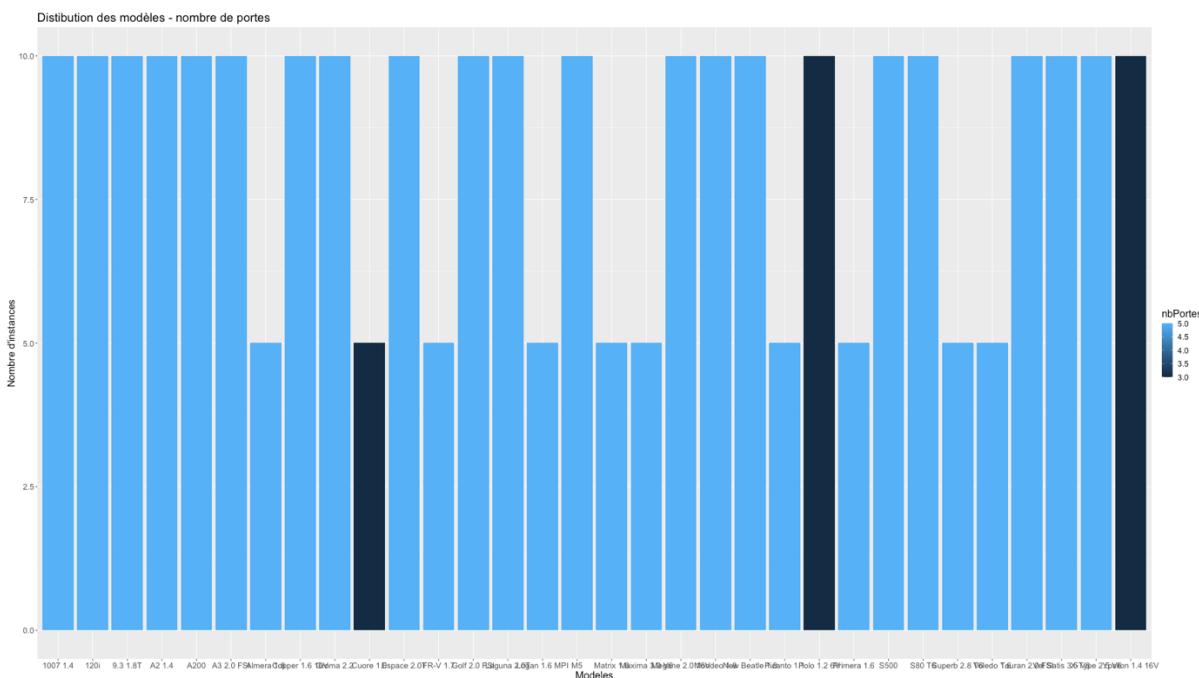
- Chaque modèle a une unique longueur :



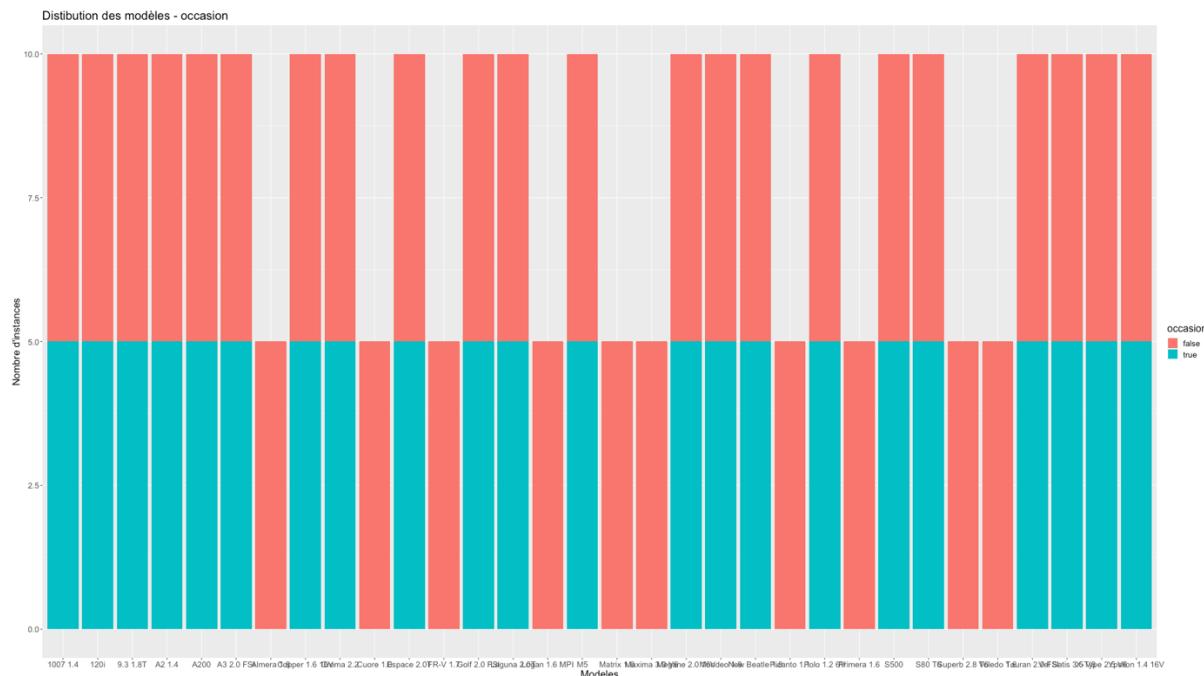
- Chaque modèle a un nombre de places unique :



- Chaque modèle a un nombre de portes unique :

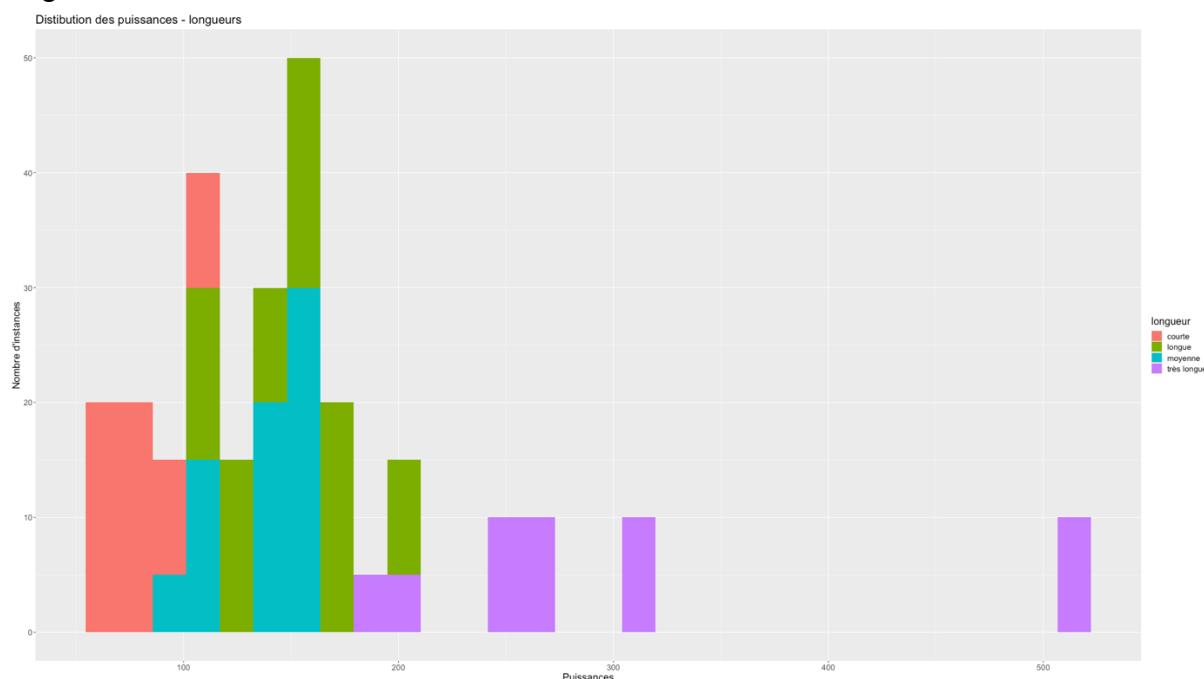


- Chaque modèle peut être neuf ou d'occasion, cela nous conforte dans l'idée que cette colonne n'est pas utile :



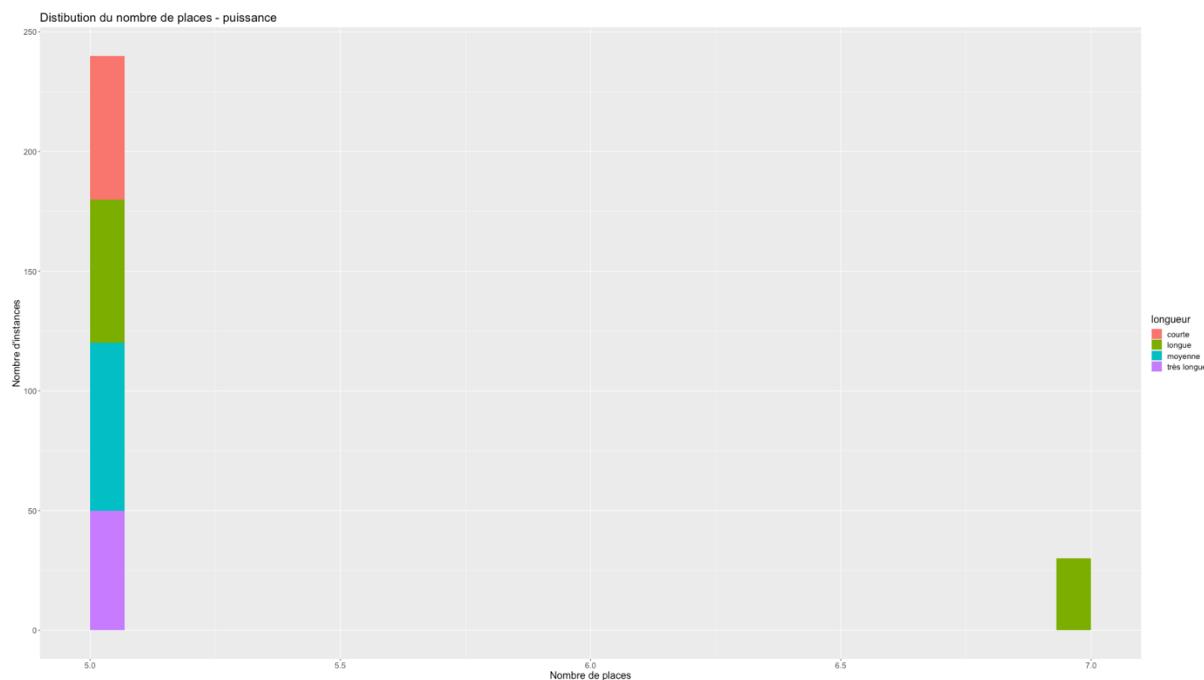
### 3. Puissance

Grâce à cette distribution, on voit bien que plus la voiture est puissante, plus sa taille augmente :



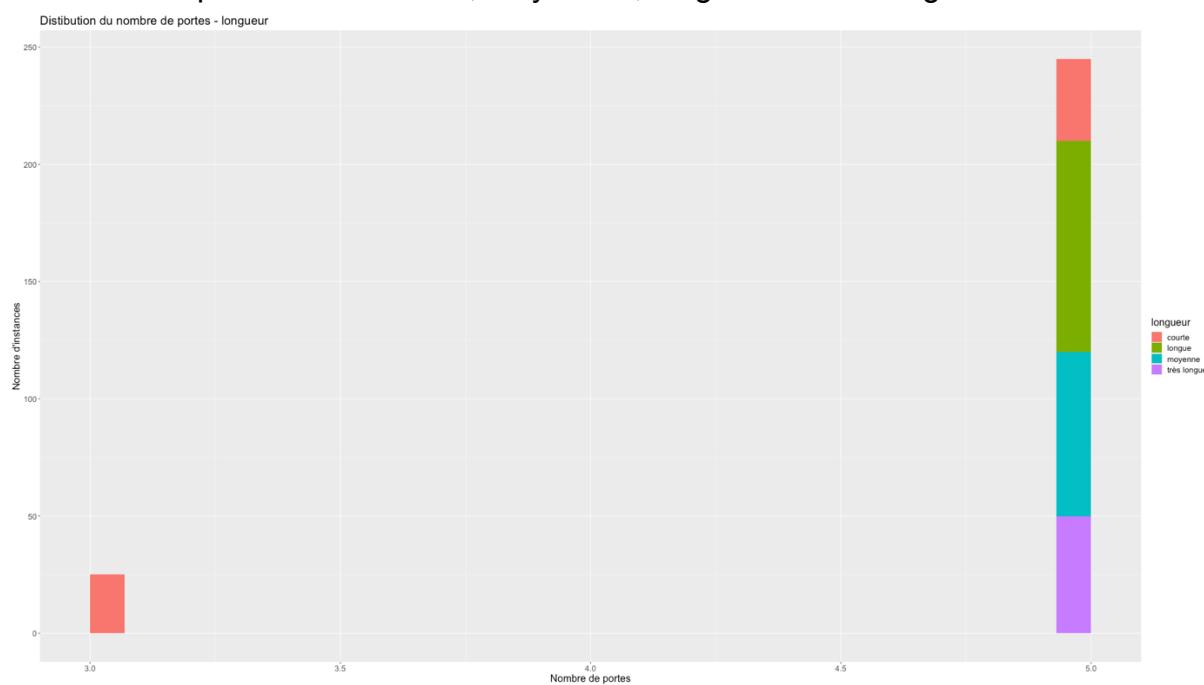
#### 4. Nombre de places

Grâce à cette distribution, on voit que les voitures de 7 places sont longues et que les voitures de 5 places sont courtes, moyennes, longues ou très longues.



#### 5. Nombre de portes

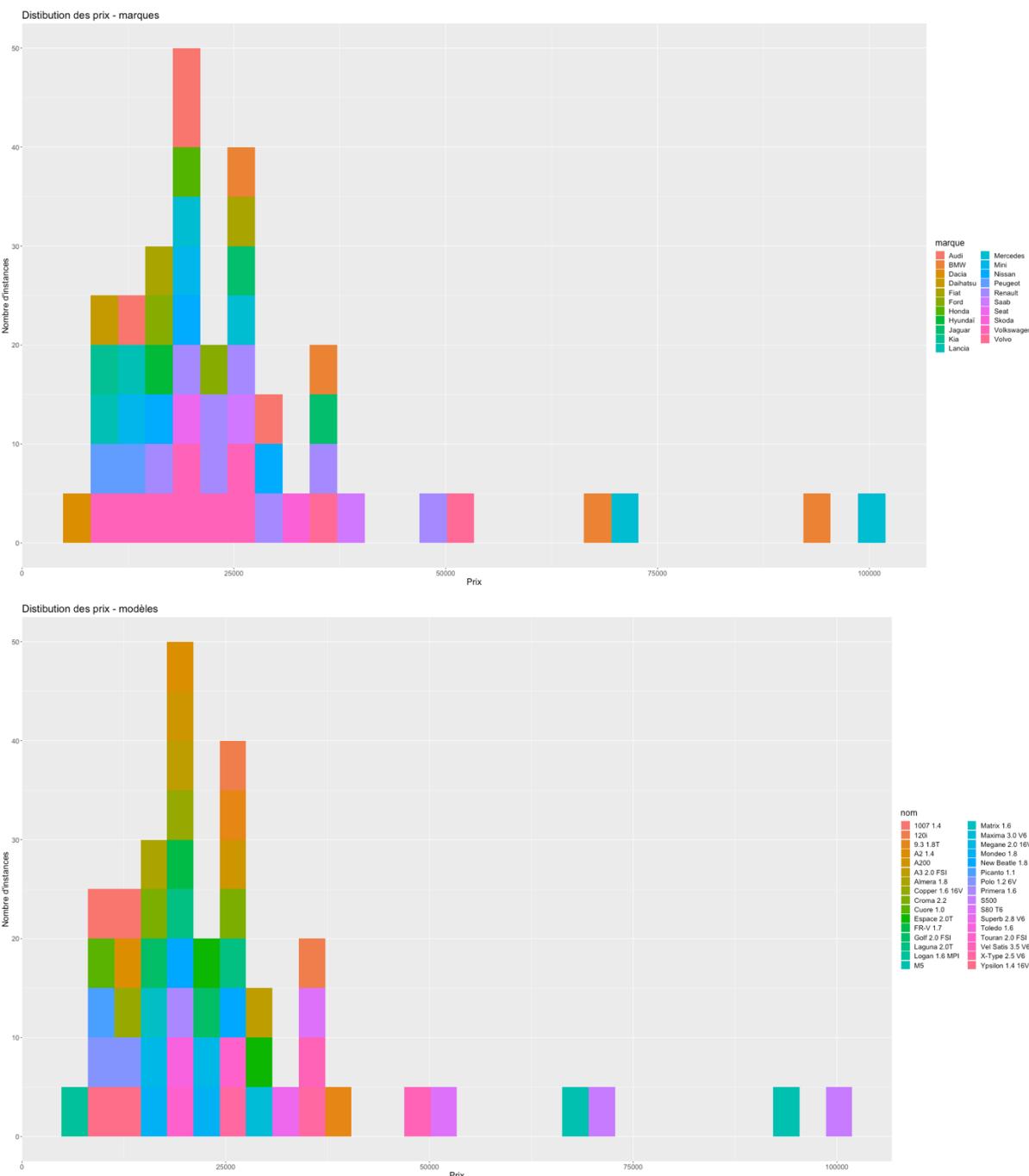
Grâce à cette distribution, on voit que les voitures de 3 portes sont courtes et que les voitures de 5 portes sont courtes, moyennes, longues ou très longues.



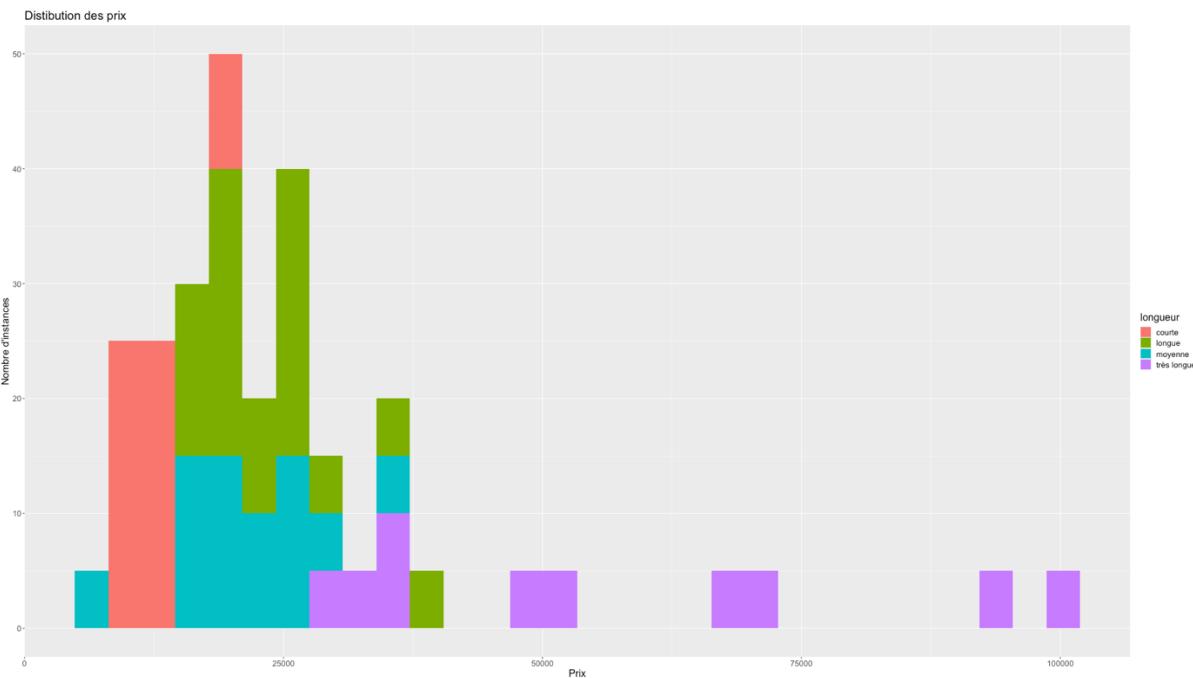
## 6. Prix

Grâce à ces distributions, on identifie clairement que :

- Chaque marque / modèle se situe dans une gamme de prix bien définie :



- Chaque longueur se situe aussi dans une gamme de prix bien définie :



## Variables d'importance particulière ou bien inutiles

Nous avons aussi fait des nuages de point, des boîtes à moustache ainsi que des tables de contingence de qui sont disponibles dans le code source mais que nous ne présenterons pas ici car ils corroborent les analyses faites ci-dessus.

Nous nous retrouvons donc avec des variables d'importance particulières et des variables inutiles :

Variables d'importance particulières	Variables inutiles
Puissance	Couleur
Longueur	Occasion
Nombre de places	
Nombre de portes	
Prix	

## Identification des catégories de voiture

Pour chaque modèle, nous avons ajouté la catégorie du véhicule grâce à des données trouvées sur internet.

Nous nous retrouvons donc avec le fichier « Catalogue » contenant une colonne « catégorie » en plus.

Nous avons créé un fichier « Catégories » qui détaille, pour chaque modèle, la catégorie attribuée. En voici le contenu :

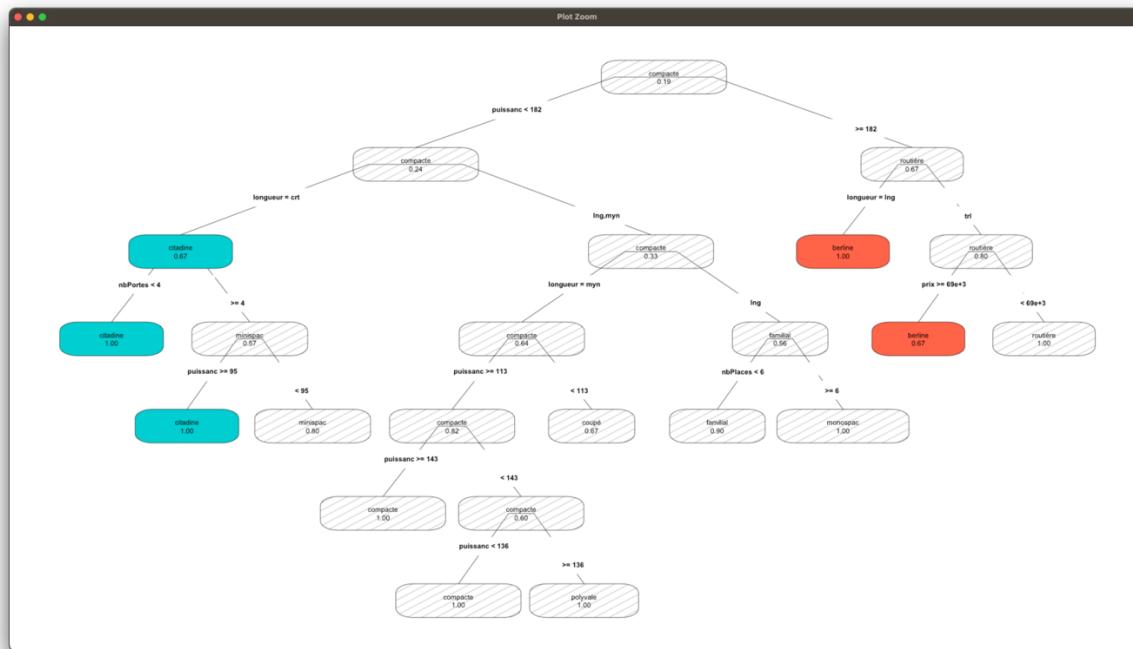
	nom	catégorie
1	S80 T6	routière
2	Touran 2.0 FSI	monospace
3	Polo 1.2 6V	citadine
4	New Beetle 1.8	coupé
5	Golf 2.0 FSI	compacte
6	Superb 2.8 V6	routière
7	Toledo 1.6	compacte
8	9.3 1.8T	familiale
9	Vel Satis 3.5 V6	routière
10	Megane 2.0 16V	compacte
11	Laguna 2.0T	familiale
12	Espace 2.0T	monospace
13	1007 1.4	minospace
14	Primera 1.6	familiale
15	Maxima 3.0 V6	routière
16	Almera 1.8	compacte
17	Copper 1.6 16V	citadine
18	S500	berline
19	A200	polyvalente
20	Ypsilon 1.4 16V	citadine
21	Picanto 1.1	citadine
22	X-Type 2.5 V6	berline
23	Matrix 1.6	monospace
24	FR-V 1.7	monospace
25	Mondeo 1.8	familiale
26	Croma 2.2	familiale
27	Cuore 1.0	citadine
28	Logan 1.6 MPI	polyvalente
29	M5	routière
30	120i	compacte
31	A3 2.0 FSI	compacte
32	A2 1.4	minospace

Nous avons créé un arbre « RPart » afin de tester si l'algorithme arrive à associer la bonne catégorie sur ce petit ensemble.

Pour cela, nous avons supprimé les colonnes qu'il ne faut pas prendre en compte pour créer l'arbre :

- Marque
- Modèle
- Couleur
- Occasion

Nous avons ensuite générée l'arbre :



Puis nous avons appliquée cet arbre à l'ensemble de données.

Nous obtenons ainsi un dataframe avec les probabilités que chaque modèle soit de telle ou telle catégorie :

	berline	citadine	compacte	coupé	familiale	minispace	monospace	polyvalente	routière
1	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
2	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
3	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
4	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
5	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
6	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
7	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
8	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
9	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
10	0.0000000	0.0	0.0	0.0000000	0.0	0.0	0	0.0000000	1.0000000
11	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
12	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
13	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
14	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
15	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
16	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
17	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
18	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
19	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
20	0.0000000	0.0	0.0	0.0000000	0.0	0.0	1	0.0000000	0.0000000
21	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
22	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
23	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
24	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
25	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
26	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
27	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
28	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
29	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
30	0.0000000	1.0	0.0	0.0000000	0.0	0.0	0	0.0000000	0.0000000
31	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
32	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
33	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
34	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
35	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
36	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000
37	0.0000000	0.0	0.0	0.6666667	0.0	0.0	0	0.3333333	0.0000000

Enfin, nous ajoutons au dataframe initial, pour chaque modèle, la catégorie prédictive qui correspond à la probabilité maximale obtenue dans le dataframe ci-dessus.

En comparant la catégorie initiale avec la catégorie prédictive, nous obtenons un taux de succès de 93%, ce qui est très performant.

Voici la liste des échecs :

▲	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	catégorie	catégoriePrédite
1	Seat	Toledo 1.6	102	longue	5	5	blanc	false	18880	compacte	familiale
2	Seat	Toledo 1.6	102	longue	5	5	noir	false	18880	compacte	familiale
3	Seat	Toledo 1.6	102	longue	5	5	rouge	false	18880	compacte	familiale
4	Seat	Toledo 1.6	102	longue	5	5	gris	false	18880	compacte	familiale
5	Seat	Toledo 1.6	102	longue	5	5	bleu	false	18880	compacte	familiale
6	Kia	Picanto 1.1	65	courte	5	5	blanc	false	8990	citadine	minispace
7	Kia	Picanto 1.1	65	courte	5	5	gris	false	8990	citadine	minispace
8	Kia	Picanto 1.1	65	courte	5	5	noir	false	8990	citadine	minispace
9	Kia	Picanto 1.1	65	courte	5	5	rouge	false	8990	citadine	minispace
10	Kia	Picanto 1.1	65	courte	5	5	bleu	false	8990	citadine	minispace
11	Dacia	Logan 1.6 MPI	90	moyenne	5	5	blanc	false	7500	polyvalente	coupé
12	Dacia	Logan 1.6 MPI	90	moyenne	5	5	rouge	false	7500	polyvalente	coupé
13	Dacia	Logan 1.6 MPI	90	moyenne	5	5	noir	false	7500	polyvalente	coupé
14	Dacia	Logan 1.6 MPI	90	moyenne	5	5	gris	false	7500	polyvalente	coupé
15	Dacia	Logan 1.6 MPI	90	moyenne	5	5	bleu	false	7500	polyvalente	coupé
16	BMW	M5	507	très longue	5	5	gris	false	94800	routière	berline
17	BMW	M5	507	très longue	5	5	blanc	false	94800	routière	berline
18	BMW	M5	507	très longue	5	5	noir	false	94800	routière	berline
19	BMW	M5	507	très longue	5	5	bleu	false	94800	routière	berline
20	BMW	M5	507	très longue	5	5	rouge	false	94800	routière	berline

## Application des catégories de véhicules définies aux données des Immatriculations

Comme pour la question 2, on génère un arbre de prédiction avec les données du Catalogue et on l'applique aux données des Immatriculations.

On obtient ainsi un nouveau fichier Immatriculations contenant une nouvelle colonne « catégorie » :

	immatriculation	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	catégorie
1	3176 TS 67	Renault	Laguna 2.0T	170	longue	5	5	blanc	false	27300	familiale
2	3721 QS 49	Volvo	S80 T6	272	très longue	5	5	noir	false	50500	routière
3	9099 UV 26	Volkswagen	Golf 2.0 FSI	150	moyenne	5	5	gris	true	16029	compacte
4	3563 LA 55	Peugeot	1007 1.4	75	courte	5	5	blanc	true	9625	minispace
5	6963 AX 34	Audi	A2 1.4	75	courte	5	5	gris	false	18310	minispace
6	5592 HQ 89	Skoda	Superb 2.8 V6	193	très longue	5	5	bleu	false	31790	routière
7	674 CE 26	Renault	Megane 2.0 16V	135	moyenne	5	5	gris	false	22350	compacte
8	1756 PR 31	Mercedes	A200	136	moyenne	5	5	noir	true	18130	polyvalente
9	6705 GX 50	BMW	120i	150	moyenne	5	5	noir	true	25060	compacte
10	4487 DR 75	Saab	9.3 1.8T	150	longue	5	5	gris	true	27020	familiale
11	7080 NW 34	Jaguar	X-Type 2.5 V6	197	longue	5	5	blanc	true	25970	berline
12	9626 HF 36	Audi	A2 1.4	75	courte	5	5	rouge	false	18310	minispace
13	2401 PA 98	Volvo	S80 T6	272	très longue	5	5	bleu	true	35350	routière
14	826 YF 89	Renault	Laguna 2.0T	170	longue	5	5	rouge	false	27300	familiale
15	8216 GR 23	Skoda	Superb 2.8 V6	193	très longue	5	5	bleu	false	31790	routière
16	8076 YM 23	Jaguar	X-Type 2.5 V6	197	longue	5	5	noir	false	37100	berline
17	9277 JN 49	BMW	M5	507	très longue	5	5	rouge	true	66360	routière
18	4231 HC 31	Audi	A2 1.4	75	courte	5	5	rouge	false	18310	minispace
19	2319 IQ 28	Ford	Mondeo 1.8	125	longue	5	5	gris	false	23900	familiale
20	148 RS 75	BMW	M5	507	très longue	5	5	blanc	true	66360	routière
21	6786 JV 36	Skoda	Superb 2.8 V6	193	très longue	5	5	gris	false	31790	routière
22	8049 KN 17	Renault	Megane 2.0 16V	135	moyenne	5	5	blanc	false	22350	compacte
23	9610 BR 52	Volkswagen	New Beetle 1.8	110	moyenne	5	5	blanc	true	18641	coupé
24	8745 KJ 12	Volkswagen	Polo 1.2 6V	55	courte	5	3	gris	false	12200	citadine
25	5805 YN 37	BMW	M5	507	très longue	5	5	gris	true	66360	routière
26	7341 QB 17	BMW	M5	507	très longue	5	5	bleu	false	94800	berline
27	9925 TY 41	Audi	A2 1.4	75	courte	5	5	rouge	false	18310	minispace
28	6238 TQ 16	Audi	A2 1.4	75	courte	5	5	noir	false	18310	minispace
29	4395 AS 40	BMW	M5	507	très longue	5	5	noir	true	66360	routière
30	1295 WZ 85	Peugeot	1007 1.4	75	courte	5	5	gris	false	13750	minispace
31	3228 PI 22	Audi	A2 1.4	75	courte	5	5	gris	false	18310	minispace
32	3367 NG 10	Volvo	S80 T6	272	très longue	5	5	gris	false	50500	routière
33	5784 HC 14	Audi	A2 1.4	75	courte	5	5	bleu	false	18310	minispace
34	6685 TE 75	Volkswagen	Golf 2.0 FSI	150	moyenne	5	5	rouge	false	22900	compacte
35	6461 RY 26	Renault	Laguna 2.0T	170	longue	5	5	noir	false	27300	familiale
36	1498 MN 80	Fiat	Croma 2.2	147	longue	5	5	blanc	false	24780	familiale
37	4400 IT 29	BMW	M5	507	très longue	5	5	gris	false	94800	berline



## Fusion des données Clients et Immatriculations

Dans les données fournies, nous avons 2 fichiers clients.

Chacun de ces fichiers « Clients » contenait des lignes partiellement vides ou erronées donc nous les avons supprimées.

De plus :

- La colonne « sexe » contenait :
  - « Masculin » remplacé par « M »
  - « Homme » remplacé par « M »
  - « Féminin » remplacé par « F »
  - « Femme » remplacé par « F »
- La colonne « situationFamiliale » contenait :
  - « Seule » remplacé par « Célibataire »
  - « Seul » remplacé par « Célibataire »
  - « Divorcée » remplacé par « Célibataire »

Puis, nous faisons un inner\_join sur les dataframes « clients » et « immatriculations » sur la colonne « immatriculation » qu'ils ont en commun afin de garder que les lignes qu'ils ont en communs.

Enfin, nous écrivons ce dataframe dans le fichier « ClientsImmatriculations.csv ».

# Création d'un modèle de classification supervisée pour la prédiction de la catégorie de véhicules

Le but était de voir si on arrivait à établir un modèle afin de savoir si on pouvait déterminer quelle catégorie de véhicule un client était le plus susceptible d'acheter en fonction de son profil.

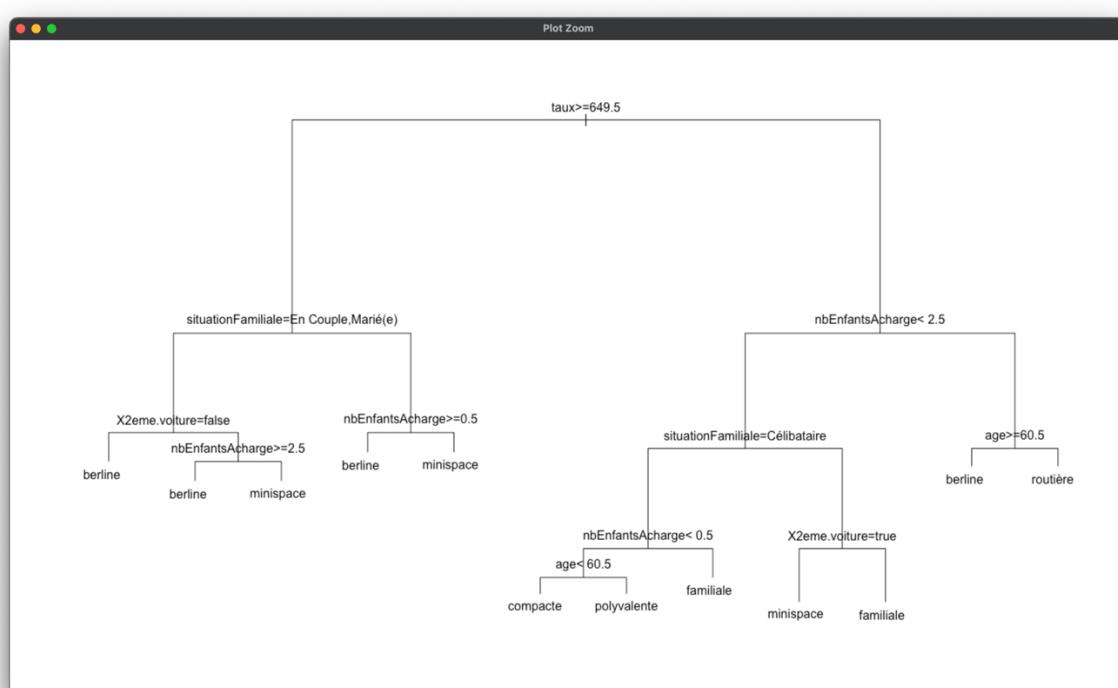
## Arbre de décision

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

On réalise un apprentissage avec un arbre RPart sur l'ensemble du dataframe :



Et comme pour la question 3, ajoute la catégorie ayant eu le maximum de probabilité au dataframe global.

On obtient donc le dataframe initial avec une colonne « catégoriePrédite » en plus :

	age	sexe	taux	situationFamiliale	nbEnfantsAcharge	X2eme.voiture	immatriculation	marque	nom	puissance	longueur	nbPlaces	nbPortes	couleur	occasion	prix	catégorie	catégoriePrédite
1	37	M	457	Célibataire	0	false	4920 VZ 62	Volkswagen	Polo 1.2 6V	55	courte	5	3	blanc	false	12200	citadine	compacte
2	26	F	462	En Couple	2	false	8605 FC 57	Ford	Mondeo 1.8	125	longue	5	5	bleu	false	23900	familiale	familiale
3	23	M	211	En Couple	2	false	2515 UN 16	Renault	Laguna 2.0T	170	longue	5	5	blanc	true	19110	familiale	familiale
4	68	F	563	Marié(e)	0	false	1580 JN 60	Saab	9.3 1.8T	150	longue	5	5	rouge	true	27020	familiale	familiale
5	47	F	986	En Couple	2	false	9723 ID 97	Jaguar	X-Type 2.5 V6	197	longue	5	5	blanc	false	37100	berline	berline
6	22	M	225	Célibataire	0	false	767 DB 60	Kia	Picanto 1.1	65	courte	5	5	gris	false	8990	minispace	compacte
7	43	M	725	En Couple	3	false	8375 OM 73	BMW	M5	507	très longue	5	5	gris	false	94800	berline	berline
8	57	M	923	En Couple	0	true	6803 JL 63	Audi	A2 1.4	75	courte	5	5	gris	false	18310	minispace	minispace
9	36	M	1392	En Couple	1	false	7275 YF 19	BMW	M5	507	très longue	5	5	noir	false	94800	berline	berline
10	28	F	817	En Couple	1	true	8105 ZW 94	Mini	Copper 1.6 16V	115	courte	5	5	rouge	false	18200	citadine	minispace
11	49	F	426	En Couple	1	false	993 EP 20	Volvo	S80 T6	272	très longue	5	5	gris	false	50500	routière	familiale
12	39	M	522	En Couple	2	false	6177 KO 47	Ford	Mondeo 1.8	125	longue	5	5	rouge	false	23900	familiale	familiale
13	53	M	187	En Couple	1	false	2419 U 11	Skoda	Superb 2.8 V6	193	très longue	5	5	rouge	false	31790	routière	familiale
14	47	M	213	En Couple	3	false	895 TW 48	Nissan	Maxima 3.0 V6	200	très longue	5	5	rouge	false	30000	routière	routière
15	21	M	461	En Couple	2	false	253 PK 36	Fiat	Croma 2.2	147	longue	5	5	gris	false	24780	familiale	familiale
16	34	M	531	En Couple	1	false	2793 MC 12	Ford	Mondeo 1.8	125	longue	5	5	blanc	false	23900	familiale	familiale
17	38	M	574	En Couple	1	false	3935 JT 36	Fiat	Croma 2.2	147	longue	5	5	bleu	false	24780	familiale	familiale
18	84	F	407	En Couple	4	true	6022 CZ 51	Mercedes	S500	306	très longue	5	5	noir	true	70910	berline	berline
19	24	M	471	Célibataire	0	false	8265 CT 57	Volkswagen	Polo 1.2 6V	55	courte	5	3	bleu	false	12200	citadine	compacte
20	84	M	166	En Couple	2	false	9552 XQ 98	Nissan	Primera 1.6	109	longue	5	5	gris	false	18650	familiale	familiale
21	23	M	1327	En Couple	1	false	7805 FP 75	Jaguar	X-Type 2.5 V6	197	longue	5	5	rouge	false	37100	berline	berline
22	73	F	1192	En Couple	2	false	9666 ZF 38	Saab	9.3 1.8T	150	longue	5	5	rouge	false	38600	familiale	berline
23	33	F	984	En Couple	1	false	4208 XB 39	Jaguar	X-Type 2.5 V6	197	longue	5	5	rouge	false	37100	berline	berline
24	65	M	1194	En Couple	0	false	2027 QN 51	Saab	9.3 1.8T	150	longue	5	5	blanc	false	38600	familiale	berline
25	21	F	468	En Couple	0	true	7129 VZ 31	Volkswagen	Polo 1.2 6V	55	courte	5	3	gris	false	12200	citadine	minispace
26	22	M	1147	Célibataire	0	false	1081 HL 80	Audi	A2 1.4	75	courte	5	5	blanc	false	18310	minispace	minispace
27	20	M	933	En Couple	1	true	1744 EB 38	Audi	A2 1.4	75	courte	5	5	rouge	false	18310	minispace	minispace
28	56	M	517	Célibataire	0	false	6526 QS 60	Audi	A2 1.4	75	courte	5	5	bleu	true	12817	minispace	compacte
29	29	M	1052	En Couple	1	false	4848 QX 54	BMW	M5	507	très longue	5	5	bleu	false	94800	berline	berline
30	59	M	597	Célibataire	0	false	5982 EQ 97	Audi	A2 1.4	75	courte	5	5	rouge	true	12817	minispace	compacte
31	62	M	881	Célibataire	0	false	836 OK 19	Audi	A2 1.4	75	courte	5	5	blanc	false	18310	minispace	minispace
32	22	F	166	En Couple	1	false	7774 WL 27	Renault	Vel Satis 3.5 V6	245	très longue	5	5	rouge	true	34440	routière	familiale
33	38	M	534	En Couple	4	false	5373 OE 10	BMW	M5	507	très longue	5	5	rouge	true	66360	routière	routière
34	38	M	494	En Couple	2	false	374 AJ 86	Volvo	S80 T6	272	très longue	5	5	bleu	false	50500	routière	familiale
35	23	M	851	En Couple	4	false	5306 AK 80	BMW	M5	507	très longue	5	5	bleu	false	94800	berline	berline
36	24	M	752	En Couple	0	false	1360 AB 34	Jaguar	X-Type 2.5 V6	197	longue	5	5	rouge	false	37100	berline	berline
37	23	F	782	En Couple	4	false	8509 YK 82	BMW	M5	507	très longue	5	5	blanc	false	94800	berline	berline

On calcul ensuite un taux de succès qui est pour cette méthode ci de 69%.

## Random forests

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

Nous créons ensuite un ensemble d'entraînement correspondant à 30% de l'ensemble global et donc un ensemble de test correspondant à 70% à l'ensemble global.

On applique la méthode randomForest à l'ensemble d'entraînement et, comme pour les autres méthodes, on obtient un datafram montrant, pour chaque ligne de l'ensemble de test, quelle est la probabilité que le client X achète une catégorie Y.

	berline	citadine	compacte	coupé	familiale	minispace	polyvalente	routière
2	0.058	0.000	0.000	0.000	0.880	0.000	0.000	0.062
3	0.018	0.000	0.000	0.000	0.834	0.000	0.000	0.148
4	0.164	0.002	0.012	0.000	0.776	0.002	0.024	0.020
7	0.968	0.000	0.000	0.000	0.000	0.000	0.004	0.028
8	0.032	0.000	0.000	0.000	0.002	0.966	0.000	0.000
9	0.974	0.000	0.000	0.000	0.024	0.000	0.000	0.002
10	0.016	0.002	0.000	0.000	0.000	0.980	0.000	0.002
11	0.068	0.000	0.000	0.000	0.754	0.000	0.000	0.178
12	0.038	0.000	0.000	0.000	0.872	0.000	0.000	0.090
13	0.010	0.000	0.000	0.000	0.874	0.000	0.000	0.116
16	0.020	0.000	0.000	0.000	0.880	0.000	0.000	0.100
17	0.014	0.000	0.000	0.000	0.860	0.000	0.000	0.126
18	0.870	0.010	0.000	0.000	0.000	0.032	0.000	0.088
19	0.000	0.000	0.828	0.000	0.000	0.172	0.000	0.000
20	0.024	0.002	0.000	0.000	0.716	0.000	0.000	0.258
22	0.160	0.000	0.000	0.000	0.840	0.000	0.000	0.000
23	0.980	0.000	0.000	0.000	0.016	0.000	0.000	0.004
24	0.256	0.000	0.000	0.000	0.698	0.036	0.010	0.000
25	0.000	0.390	0.000	0.000	0.014	0.578	0.000	0.018
28	0.000	0.000	0.798	0.000	0.000	0.202	0.000	0.000
29	0.978	0.000	0.000	0.000	0.020	0.000	0.000	0.002
31	0.004	0.000	0.000	0.000	0.006	0.508	0.482	0.000
32	0.018	0.000	0.000	0.000	0.838	0.002	0.000	0.142
33	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.998
34	0.020	0.000	0.000	0.000	0.916	0.000	0.000	0.064
35	0.972	0.000	0.000	0.000	0.000	0.000	0.000	0.028
37	0.964	0.000	0.000	0.000	0.000	0.000	0.000	0.036
38	0.978	0.000	0.000	0.000	0.020	0.000	0.000	0.002
39	0.924	0.000	0.000	0.000	0.030	0.038	0.000	0.008
45	0.092	0.000	0.000	0.000	0.008	0.000	0.000	0.900
48	0.030	0.002	0.000	0.000	0.002	0.966	0.000	0.000
49	0.978	0.000	0.000	0.000	0.020	0.000	0.000	0.002
51	0.004	0.000	0.000	0.000	0.000	0.996	0.000	0.000
52	0.924	0.000	0.000	0.000	0.032	0.042	0.000	0.002
53	0.004	0.006	0.000	0.000	0.004	0.406	0.580	0.000
55	0.060	0.000	0.000	0.000	0.934	0.000	0.000	0.006

Nous ajoutons donc une nouvelle colonne au dataframe de test afin de comparer la prédiction avec le résultat attendu.

Dans ce cas-ci, nous obtenons un taux de succès de 71%.

En voici la table de confusion :

		Reference							
Prediction		berline	citadine	compacte	coupé	familiale	minispace	polyvalente	routière
berline	25056	0	7	3	28	14	2	19	
citadine	6	1499	15	8	3	1400	0	5	
compacte	14	5289	10964	1473	14	7063	0	16	
coupé	0	0	0	0	0	0	0	0	
familiale	6711	4	9	1	24338	30	1	11303	
minispace	18	2631	294	1	14	17594	780	14	
polyvalente	5	867	5	1	3	1562	2486	1	
routière	14	6	3	1	12	12	0	16277	

Overall Statistics									
Accuracy : 0.7122									
95% CI : (0.7098, 0.7146)									
No Information Rate : 0.2308									
P-Value [Acc > NIR] : < 2.2e-16									
Kappa : 0.6539									
McNemar's Test P-Value : < 2.2e-16									
Statistics by Class:									
Class: berline Class: citadine Class: compacte Class: coupé Class: familiale Class: minispace Class: polyvalente Class: routière									
Sensitivity	0.7873	0.14559	0.97052	0.00000	0.9970	0.6357	0.76048	0.5890	
Specificity	0.9993	0.98874	0.89045	1.00000	0.8409	0.9660	0.98185	0.9996	
Pos Pred Value	0.9971	0.51056	0.44151	NaN	0.5741	0.8242	0.50426	0.9971	
Neg Pred Value	0.9400	0.93482	0.99705	0.98921	0.9992	0.9135	0.99411	0.9066	
Prevalence	0.2308	0.07466	0.08192	0.01079	0.1770	0.2007	0.02371	0.2004	
Detection Rate	0.1817	0.01087	0.07951	0.00000	0.1765	0.1276	0.01803	0.1180	
Detection Prevalence	0.1822	0.02129	0.18008	0.00000	0.3075	0.1548	0.03575	0.1184	
Balanced Accuracy	0.8933	0.56716	0.93049	0.50000	0.9189	0.8008	0.87116	0.7943	

## Support vector machines

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

Nous créons ensuite un ensemble d'entraînement correspondant à 30% de l'ensemble global et donc un ensemble de test correspondant à 70% à l'ensemble global.

Nous avons 4 types de Support Vector Machines à notre disposition :

- Linear
- Polynomial
- Radial
- Sigmoid

Nous entraînons donc ces 4 types avec l'ensemble d'entraînement puis on applique le modèle obtenu à l'ensemble de test.

En termes de taux de succès, nous obtenons :

Modèle	Taux de succès
Linear	65%
Polynomial	69%
Radial	70%
Sigmoid	51%

Le modèle radial apporte donc les meilleures performances. On en extrait donc la matrice de confusion :

		Reference							
Prediction		berline	citadine	compacte	coupé	familiale	minispace	polyvalente	routière
berline	24851	0	7	3	438	14	1	126	
citadine	2	540	130	79	1	366	0	2	
compacte	13	5061	10360	1396	14	6673	60	16	
coupé	0	0	0	0	0	0	0	0	
familiale	6602	4	9	1	23937	30	1	11310	
minispace	23	3957	784	7	15	18774	640	16	
polyvalente	5	728	4	1	4	1806	2566	2	
routière	328	6	3	1	3	12	1	16163	

Overall Statistics									
Accuracy : 0.7048									
95% CI : (0.7024, 0.7072)									
No Information Rate : 0.2308									
P-Value [Acc > NIR] : < 2.2e-16									
Kappa : 0.6439									
McNemar's Test P-Value : < 2.2e-16									

Statistics by Class:									
	Class: berline	Class: citadine	Class: compacte	Class: coupé	Class: familiale	Class: minispace	Class: polyvalente	Class: routière	
Sensitivity	0.7809	0.052448	0.91706	0.00000	0.9805	0.6784	0.78495	0.5849	
Specificity	0.9944	0.995455	0.89547	1.00000	0.8418	0.9506	0.98106	0.9968	
Pos Pred Value	0.9768	0.482143	0.43911	NaN	0.5714	0.7753	0.50156	0.9786	
Neg Pred Value	0.9380	0.928672	0.99180	0.98921	0.9951	0.9217	0.99471	0.9055	
Prevalence	0.2308	0.074665	0.08192	0.01079	0.1770	0.2007	0.02371	0.2004	
Detection Rate	0.1802	0.003916	0.07513	0.00000	0.1736	0.1361	0.01861	0.1172	
Detection Prevalence	0.1845	0.008122	0.17109	0.00000	0.3038	0.1756	0.03710	0.1198	
Balanced Accuracy	0.8877	0.523951	0.90627	0.50000	0.9112	0.8145	0.88300	0.7908	

## Réseau de neurones

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

Nous créons ensuite un ensemble d'entraînement correspondant à 30% de l'ensemble global et donc un ensemble de test correspondant à 70% à l'ensemble global.

Nous entraînons donc le modèle avec différents paramètres de taille, de décroissance ainsi que de nombre d'itération maximum afin de voir là où on tire les meilleures performances :

<b>Size</b>	<b>Decay</b>	<b>Maxit</b>	<b>Taux de succès</b>
50	0.01	100	56%
50	0.01	300	64%
25	0.01	100	37%
25	0.01	300	68,4%
50	0.001	100	56%
50	0.001	300	68%
25	0.001	100	23%
25	0.001	300	23%

Ce modèle avec ces paramètres (25, 0.01, 30) nous donne les meilleures performances. On en extrait donc la matrice de confusion :

nnClass	
berline	
berline	31824
citadine	10296
compacte	11297
coupé	1488
familiale	24412
minispace	27675
polyvalente	3269
routière	27635

## KNN

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

Nous créons ensuite un ensemble d'entraînement correspondant à 30% de l'ensemble global et donc un ensemble de tests correspondant à 70% à l'ensemble global.

Nous entraînons donc le modèle avec différents paramètres de nombre de voisins ainsi que de distance afin de voir là où on tire les meilleures performances :

<b>K</b>	<b>Distance</b>	<b>Taux de succès</b>
10	1	67%
10	2	67%
20	1	68%
20	2	69%

Ce modèle avec ces paramètres (20, 2) nous donne les meilleures performances (69%).

## Naive Bayes

Pour cette méthode, on part du fichier « ClientsImmatriculations » créé lors de la question 4.

On supprime les colonnes communiquant des informations sur le véhicule :

- Immatriculation
- Marque
- Modèle
- Puissance
- Longueur
- Nombre de places
- Nombre de portes
- Couleur
- Occasion
- Prix

Nous créons ensuite un ensemble d'entraînement correspondant à 30% de l'ensemble global et donc un ensemble de tests correspondant à 70% à l'ensemble global.

Nous entraînons donc le modèle avec différents paramètres de lissage de laplace ainsi que d'utilisation du noyau afin de voir là où on tire les meilleures performances :

Laplace	Usekernel	Taux de succès
0	FALSE	60.3%
20	FALSE	60.26%
0	TRUE	59%
20	TRUE	59%

Ce modèle avec ces paramètres (20, TRUE) nous donne les meilleures performances (69%). On en extrait donc la matrice de confusion :

Confusion Matrix and Statistics

		Reference							
Prediction		berline	citadine	compacte	coupé	familiale	minispace	polyvalente	routière
berline	22173	137	6	2	1978	1390	1	22	
citadine	37	1184	1	0	170	1278	0	93	
compacte	16	4517	9747	796	66	8483	556	41	
coupé	3	1047	1256	665	11	845	5	6	
familiale	4650	3	6	0	16020	21	0	7617	
minispace	1740	998	227	1	760	11214	40	371	
polyvalente	6	689	46	21	6	2049	2665	4	
routière	3199	1721	8	3	5401	2395	2	19481	

Overall Statistics

Accuracy : 0.603

95% CI : (0.6004, 0.6056)

No Information Rate : 0.2308

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.5242

McNemar's Test P-Value : < 2.2e-16

Statistics by Class:

	Class: berline	Class: citadine	Class: compacte	Class: coupé	Class: familiale	Class: minispace	Class: polyvalente	Class: routière
Sensitivity	0.6967	0.114996	0.86280	0.446909	0.6562	0.40520	0.81523	0.7049
Specificity	0.9667	0.987625	0.88566	0.976739	0.8916	0.96247	0.97905	0.8846
Pos Pred Value	0.8625	0.428520	0.40240	0.173267	0.5657	0.73051	0.48578	0.6048
Neg Pred Value	0.9140	0.932570	0.98636	0.993861	0.9234	0.86567	0.99544	0.9228
Prevalence	0.2308	0.074665	0.08192	0.010791	0.1770	0.20069	0.02371	0.2004
Detection Rate	0.1608	0.008586	0.07068	0.004822	0.1162	0.08132	0.01933	0.1413
Detection Prevalence	0.1864	0.020037	0.17565	0.027833	0.2054	0.11132	0.03978	0.2336
Balanced Accuracy	0.8317	0.551311	0.87423	0.711824	0.7739	0.68383	0.89714	0.7947

## Récapitulatif

Méthode	Taux de succès
Arbre de décision	69%
Random forests	71%
Support vector machines	70%
Réseau de neurones	68,4%
KNN	69%
Naive Bayes	60,3%

Ainsi, on va donc appliquer le modèle Random forests aux données Marketing afin de tenter de prédire quelle catégorie de voiture ces clients sont les plus susceptibles d'acheter.

# Application du modèle de prédiction aux données Marketing

On obtient donc :

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



## 5. Conclusion

A l'issue de ce projet, nous avons réussi à répondre au besoin du concessionnaire en mettant à sa disposition un outil lui permettant de viser directement une catégorie de voiture susceptible de correspondre au profil de son client. Nous lui offrons, de surcroît, un rendu visuel pertinent qui permet de diffuser des informations importantes en les traduisant dans un contexte visuel plus simple à se faire comprendre par n'importe quel utilisateur.

Au fil de la réalisation du projet, nous avons appris à gérer des données de taille significative, à les nettoyer et à y accéder à l'aide des tables externes. Le projet nous a invité à s'organiser, prévoir des réunions chaque semaine pour suivre nos avancées. De plus, la mise en place d'un repository [Github](#) a permis un meilleur suivi des tâches et gestion des versions de codes/scripts.

Finalement, nous tenons à exprimer nos remerciements les plus sincères envers vous, les lecteurs de ce présent rapport, pour votre temps, pour votre attention et pour le temps que vous nous accordez en acceptant de juger ce travail. Nous vous remercions surtout, pour l'ensemble des informations vues en cours et dont ce projet demeure le fruit.