




Projet Big Data et d'Analyse de la Clientèle d'un Concessionnaire Automobile pour la Recommandation de Modèle 2020-2021



Groupe 7
Cao Nicolas
Guinaldo Vincent
Poirier Corentin

Table des matières

Objectif et contexte	4
1. Chargement Nettoyage des données	5
1.1 Imports des fichiers csv.	5
1.2 Analyse Exploratoire des données (fichier Immatriculation).....	6
1.2.1 Histogramme de la variable longueur	7
1.2.2 Histogramme de la variable nombre de portes	7
1.2.3 Histogramme des variables marque, couleur et occasion	8
1.2.4 Opération post-analyse	8
1.2.5 Conclusion	8
1.3 Analyse Exploratoire des données (fichier Client)	9
1.3.1 Opérations sur la variable Age	10
1.3.2 Opérations sur la variable sexe	10
1.3.3 Opérations sur la variable taux	12
1.3.4 Opérations sur la variable situationFamiliale	12
1.3.5 Opérations sur la variable nombreEnfantsAcharge	13
1.3.6 Conclusion	14
2 Construction des ensembles de tests et d'apprentissage.....	15
2.1 Création des catégories de véhicules.....	15
2.2 Ensemble d'apprentissage et de test	18
3 Test des classifieurs et création de notre modèle de prédiction	22
3.1 Liste des classifieurs	22
3.2 1 ^{er} jeu de test	22
3.2.1 C5.0	22
3.2.2 Naive-Bayes.....	23
3.2.3 SVM.....	24
3.2.4 Kknn	25
3.2.5 Nnet	26
3.2.6 Randomforest	27

3.2.7 Conclusion sur ce 1 ^{er} test	28
3.3 2eme jeu de test	28
3.3.1 C5.0	28
3.3.2 Naive Bayes.....	29
3.3.3 SVM.....	30
3.3.4 Kknn	30
3.3.5 Nnet	31
3.3.6 Random Forest	32
3.3.7 Conclusion.....	33
3.4 Prédiction	33
4 Conclusion	35
Annexe	35
Importation des données via un driver sql	35
Tri des données sur sql.....	36

Objectif et contexte

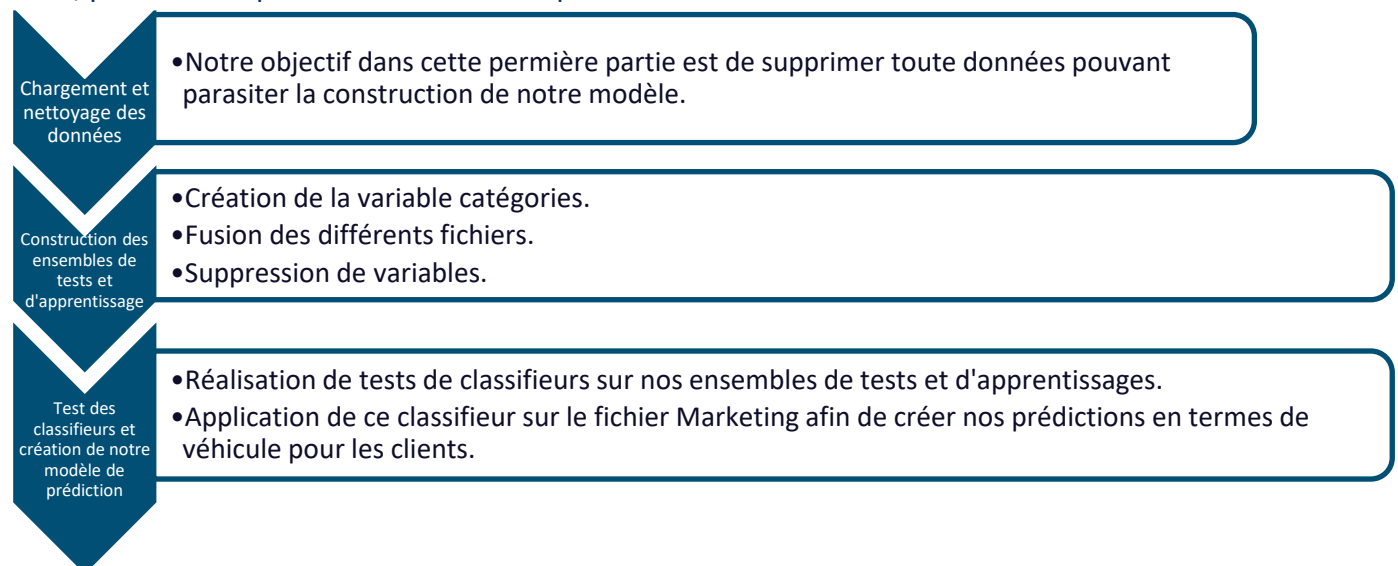
Nous avons été contactés par un concessionnaire automobile afin de l'aider à mieux cibler les véhicules susceptibles d'intéresser les potentiels clients. Ainsi, à travers cette demande, notre but est de construire un modèle prédiction à travers l'utilisation de différentes méthodes de classification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

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

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

- Un fichier, cité précédemment, concernant les immatriculations effectuées au cours de l'année. Ce fichier s'appelant `Immatriculation.csv`
- Un fichier concernant le catalogue de véhicules. Ce fichier s'appelant `Catalogue.csv`
- Un fichier concernant le profil des clients ayant réalisé un achat de voiture sur l'année en cours. Ce fichier s'appelant `Clients_6.csv`
- Enfin un fichier concernant le profil des clients sélectionnés par le service marketing pour créer ce modèle prédictif. Ce fichier s'appelant `Marketing.csv`

Enfin, plusieurs étapes vont être cruciales pour la création de ce modèle :



1. Chargement Nettoyage des données

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

- Par un driver HDFS
- Par un driver Oracle





Ici, nous allons nous intéresser au driver HDFS. Toutes les informations concernant le driver Oracle sont présentes en Annexe.

1.1 Imports des fichiers csv.

Afin de réaliser l'import des données et l'établissement de table Immatriculation, Client, Marketing et Catalogue nous allons utiliser la fonction `read.csv` de la manière suivante :

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

Une fois cette commande exécutée, nous pouvons voir que l'import s'est bien déroulé :

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

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

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

1.2 Analyse Exploratoire des données (fichier Immatriculation)

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

```
summary(immatriculation)

##immatriculation      marque      nom      puissance      longueur      nbPlaces      nbPortes
##Length:2000000      Length:2000000      Length:2000000      Min.   : 55      Length:2000000      Min.   : 5      Min.   :3.000
##Class :character      Class :character      Class :character      1st Qu.: 75      Class :character      1st Qu.:5      1st Qu.:5.000
##Mode  :character      Mode  :character      Mode  :character      Median :150      Mode  :character      Median :5      Median :5.000
                                #Mean :199                                #Mean :5      #Mean :4.868
                                #3rd Qu.:245                                3rd Qu.:5      3rd Qu.:5.000
                                #Max.  :507                                Max.   :5      Max.   :5.000

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

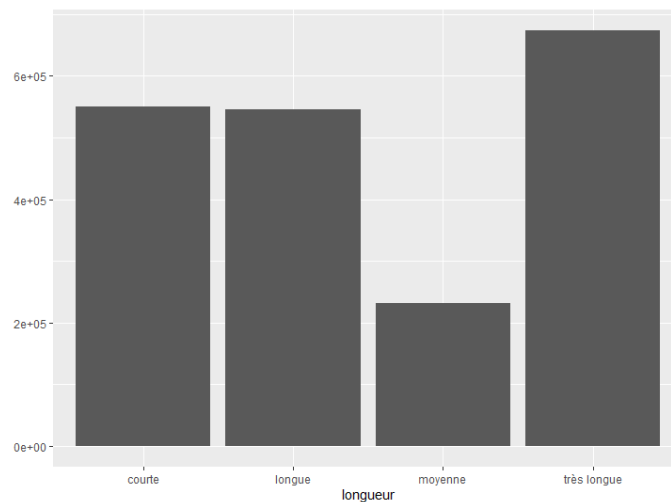
Pour ce qui est de la table immatriculation, on ne remarque aucune valeur de telles sortes. Cependant, nous pouvons voir que dans la table immatriculation, nous n'avons aucune voiture de 7 places.

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

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

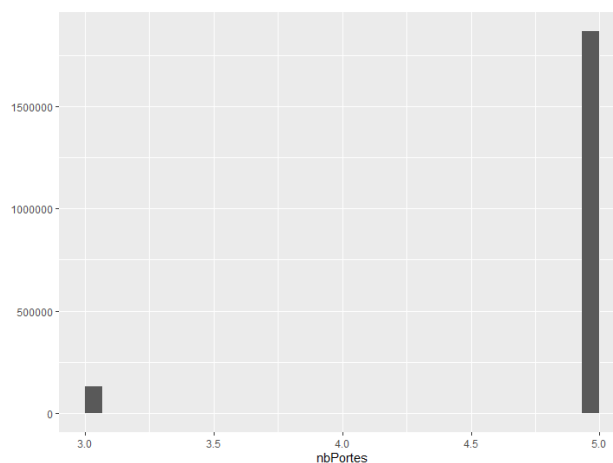
1.2.1 Histogramme de la variable longueur

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



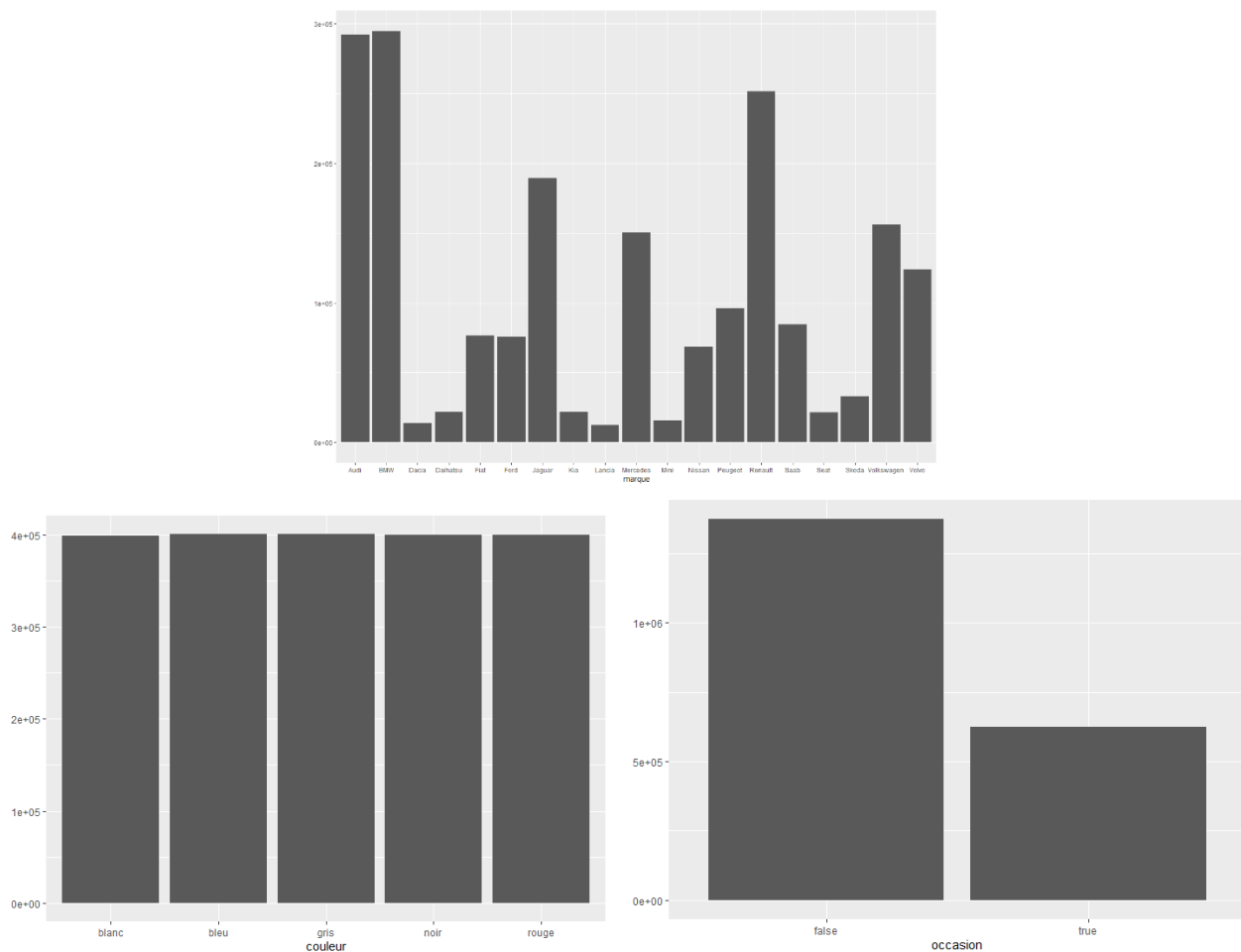
Tout d'abord, nous remarquons que nous n'avons aucune valeur erronée dans cette variable. Cependant, nous avons une répartition à peu près équivalente entre les voitures longues et courtes. Nous avons peu de voitures de longueur moyenne, mais nous avons énormément de voiture très longue (d'un facteur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

1.2.2 Histogramme de la variable nombre de portes



A travers cet histogramme nous remarquons l'absence de valeurs erronées pour cette variable. Mais aussi qu'une majorité sont des voitures 5 portes. Ainsi, lors de l'établissement des règles pour la construction de catégorie de voiture, nous concluons que cette donnée ne sera pas nécessaire.

1.2.3 Histogramme des variables marque, couleur et occasion



Au vu de ces trois histogrammes, aucune de ces trois variables ne contient de valeurs erronées.

1.2.4 Opération post-analyse

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

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

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

1.2.5 Conclusion

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

1.3 Analyse Exploratoire des données (fichier Client)

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

```
#Import fichier clients
clients <- read.csv("Clients_6.csv", header = TRUE, sep = ",", dec = ".")
clients$age <- as.integer(clients$age)
clients$taux <- as.integer(clients$taux)
clients$situationFamiliare <- as.factor(clients$situationFamiliare)
clients$nbEnfantsAcharge <- as.integer(clients$nbEnfantsAcharge)
clients$x2eme.voiture <- as.logical(clients$x2eme.voiture)
clients$sexe <- as.factor(clients$sexe)
```

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

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

Maintenant que ces valeurs sont enlevées, nous affichons un résumé de cette table avec la fonction summary :

```
summary (clients)
```

##age	sexe	taux	situationFamiliare	nbEnfantsAcharge	x2eme.voiture	immatriculation
##Min. : -1.00	M : 67756	Min. : -1.0	En Couple : 63346	Min. : -1.000	Mode : logical	Length: 99194
##1st Qu.: 28.00	F : 29131	1st Qu.: 420.0	Célibataire: 29589	1st Qu.: 0.000	FALSE: 86319	Class : character
##Median : 42.00	Homme : 735	Median : 520.0	Seule : 4960	Median : 1.000	TRUE : 12875	Mode : character
##Mean : 43.73	Masculin: 679	Mean : 607.8	Marié(e) : 652	Mean : 1.245		
##3rd Qu.: 57.00	Féminin : 318	3rd Qu.: 827.0	Seul : 280	3rd Qu.: 2.000		
##Max. : 84.00	Femme : 287	Max. : 1399.0	N/D : 111	Max. : 4.000		
##	(Other) : 288		(other) : 256			

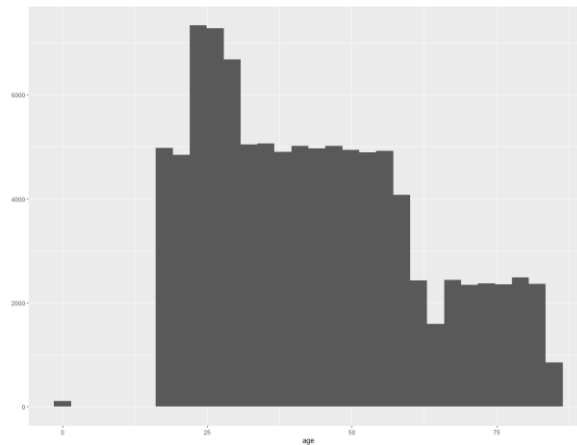
Grace à ce résumé, nous pouvons identifier les variables qui possèdent des valeurs erronées. Ce sont donc les variables :

- Age
- Sexe
- Taux
- SituationFamiliare
- nbEnfantsAcharge

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

1.3.1 Opérations sur la variable Age

Tout d'abord nous réalisons un histogramme de cette variable :



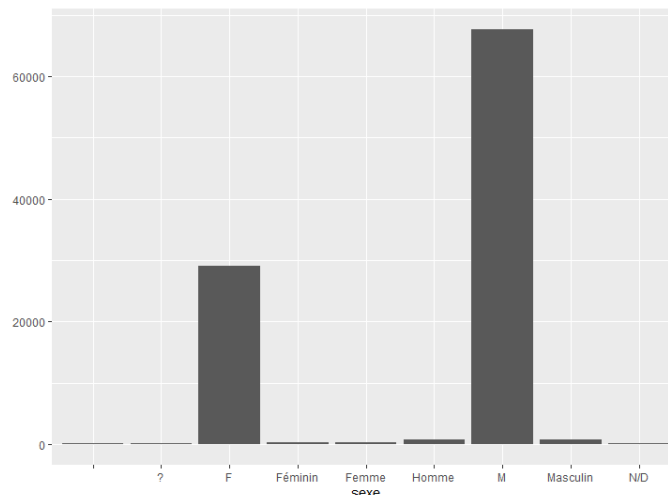
On remarque donc qu'il y a des valeurs outlier pour cette variable. En effet, cet histogramme nous indique que l'âge de certains clients est de 0. Or, pour cette étude, l'âge minimum des clients qui ont acheté des véhicules au cours de l'année est de 18 ans. Nous allons donc utiliser la fonction suivante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

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

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

1.3.2 Opérations sur la variable sexe

Tout comme la variable âge, utilisons ici un histogramme afin de rendre l'analyse des erreurs plus visuelle :



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

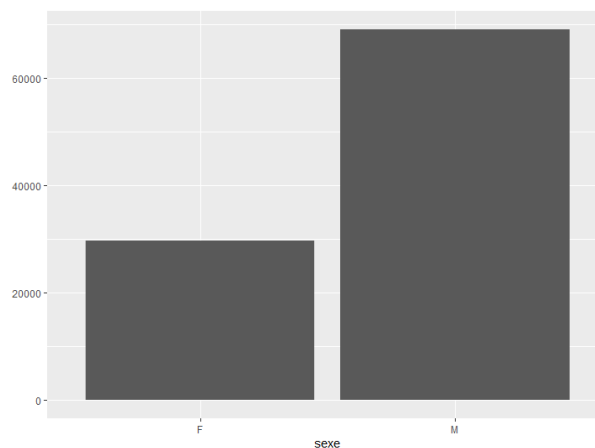
```
clients <- subset(clients, clients$sexe!="?" & clients$sexe!=" " & clients$sexe!="N/D")
```

De plus, le format requis pour cette donnée pour le sexe masculin est M et pour le sexe féminin est F. On va donc utiliser le code suivant :

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

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

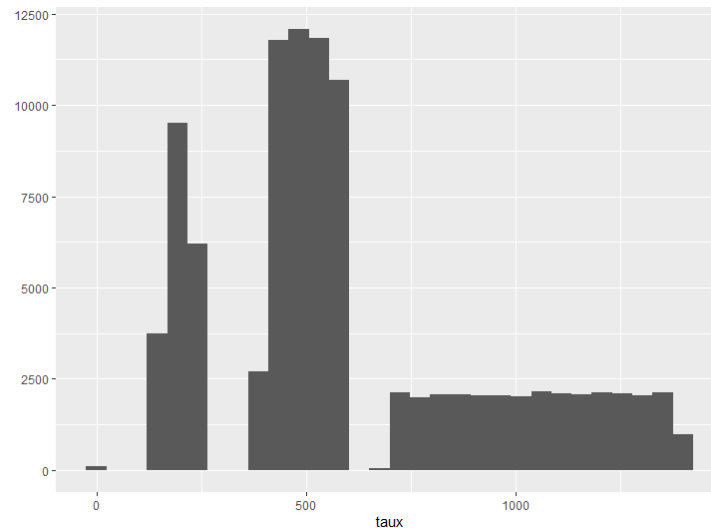
Enfin, nous allons vérifions le résultat de nos manipulations avec un nouvel histogramme :



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

1.3.3 Opérations sur la variable taux

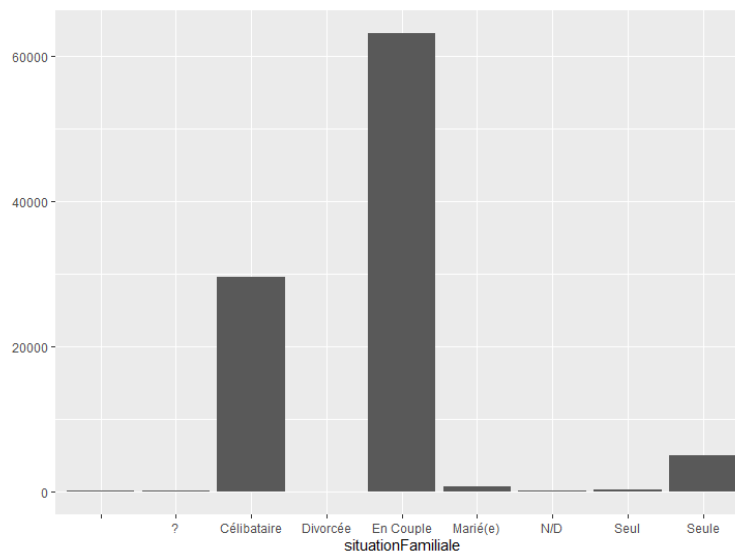
Nous allons donc établir l'histogramme de la variable taux :



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

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

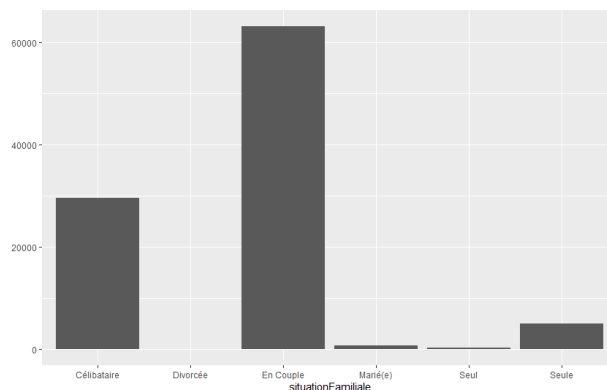
1.3.4 Opérations sur la variable situationFamiliale



Nous remarquons donc la présence de valeurs « », « ? » et « N/D » dans cette variable. Nous les supprimons à l'aide de la ligne de code suivante :

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

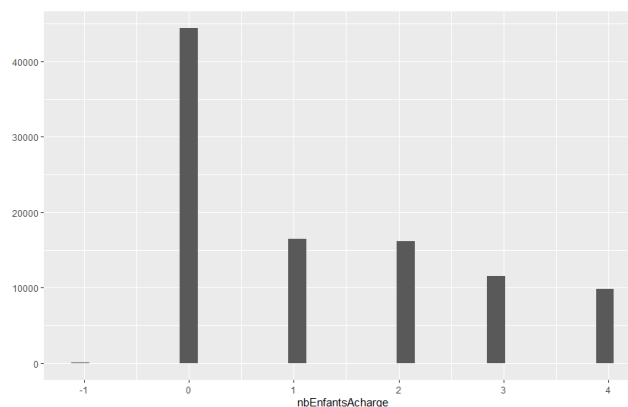
La fonction « droplevels » nous permet d'éliminer les modalités de facteurs qui n'existent plus. Vérifions que nos manipulations ont bien fonctionné.



Nous remarquons donc la disparition de ces valeurs erronées.

1.3.5 Opérations sur la variable nombreEnfantsAcharge

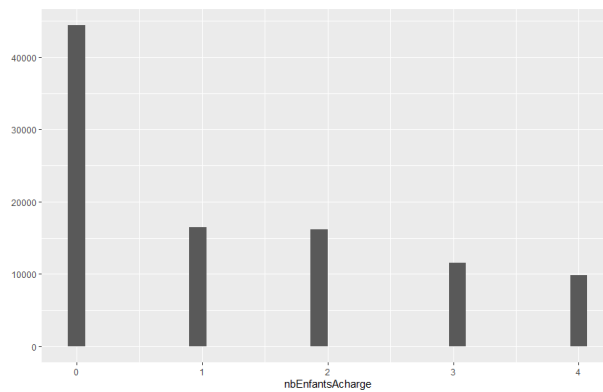
Nous allons donc créer l'histogramme par rapport à cette variable :



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

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

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



Ainsi nous remarquons la disparition des données qui sont hors du domaine de valeurs.

1.3.6 Conclusion

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

```
#Résumé clients
summary(clients)
```

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

Nous remarquons bien, au travers de cette fonction la disparition des valeurs erronées.

Maintenant que nous avons fini le traitement de nos données, l'étape suivante est la création de nos ensembles d'apprentissage et de tests.

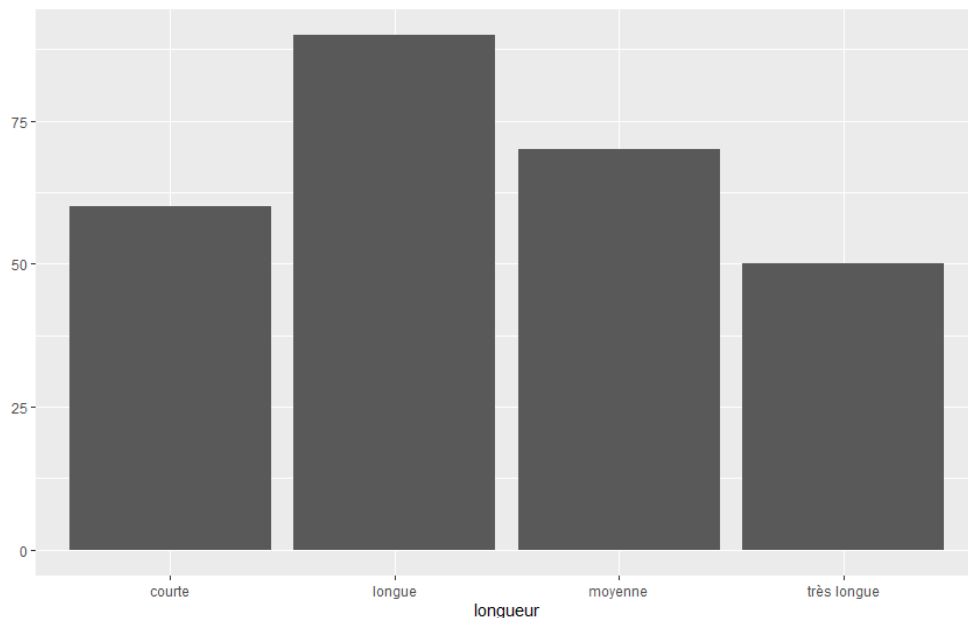
2 Construction des ensembles de tests et d'apprentissage

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

2.1 Création des catégories de véhicules

Pour la création de la variable catégorie, une analyse profonde des variables présente dans la table catégories est essentiel.

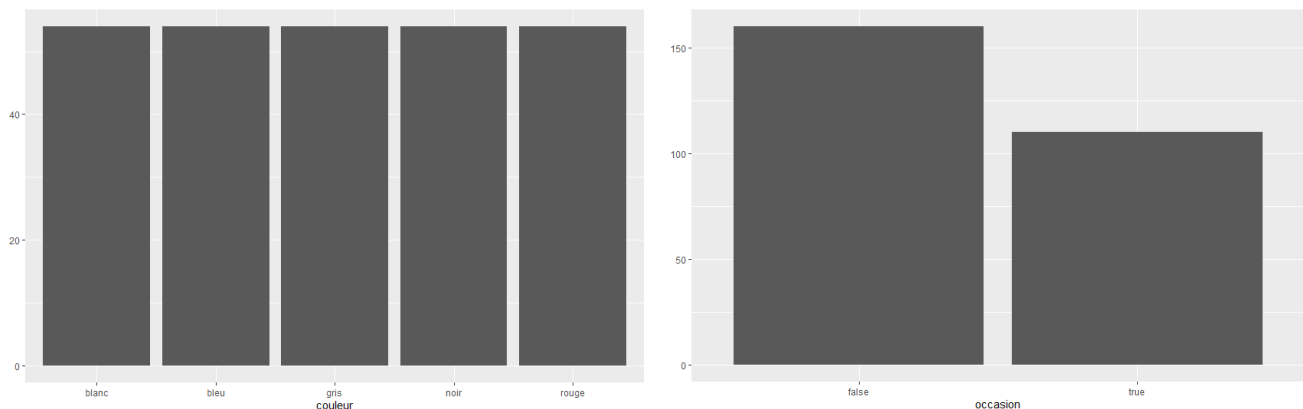
Tout d'abord, intéressons-nous à la variable longueur et sa répartition. Pour cela, nous allons réaliser l'histogramme de cette variable :



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

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

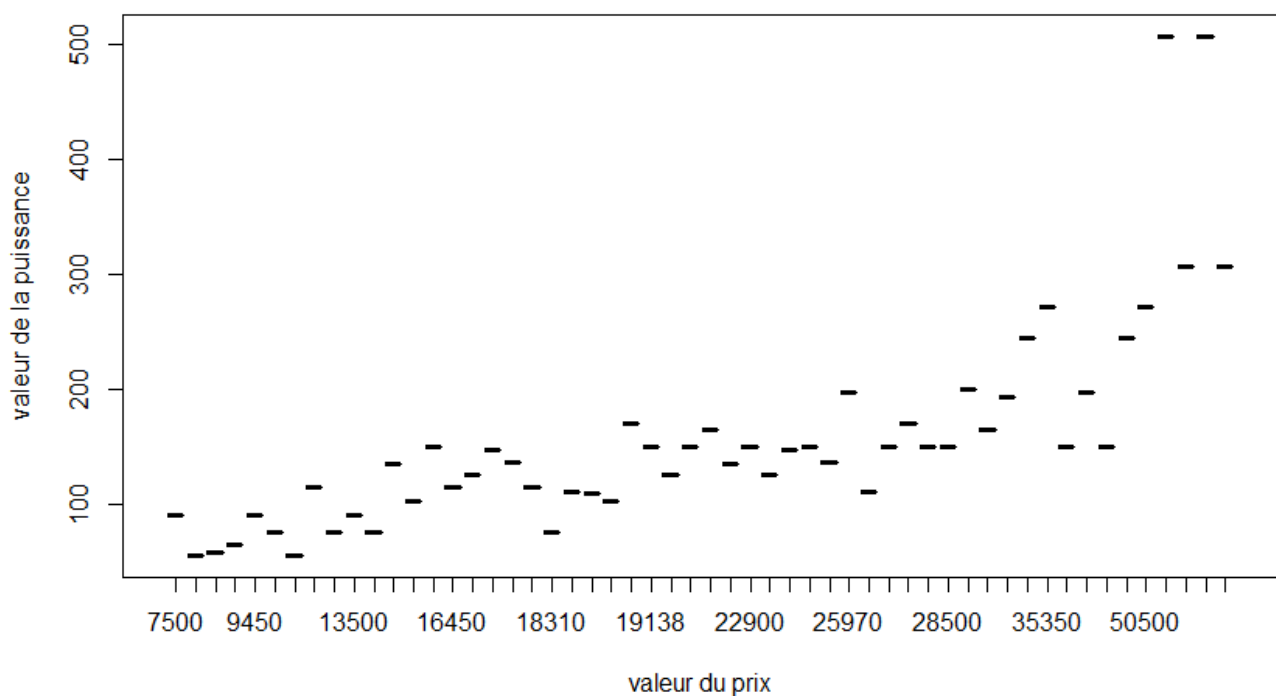
Si nous nous intéressons aux histogrammes des variables occasion et couleur :



Nous remarquons une assez bonne répartition des valeurs possibles pour ces variables. Ainsi, la variable couleur et occasion ne seront pas des critères utiles pour l'établissement de critères pour nos catégories.

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

Distribution de la puissance selon le prix



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

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

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

Ainsi, nous allons créer les catégories suivantes :

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

Afin de créer ces catégories sur R nous utilisons le code suivant :

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

Maintenant, que nous avons créé ces catégories nous pouvons créer la variable « catégorie » dans le data-frame Immatriculation

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

Enfin, l'étape suivante est de réaliser la fusion entre la table Clients et Immatriculation. Nous réalisons le code suivant :

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

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

2.2 Ensemble d'apprentissage et de test

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

La 1^{ère} étape que nous réalisons est la suppression de la variable `immatriculation`, de ce dernier data-frame, car c'est une variable inutile. Nous la supprimons donc avec le code suivant :

```
clients_immatriculations<-clients_immatriculations[,-1]
```

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

De plus, nous avons vu que la puissance est étroitement liée au prix, donc nous faisons aussi le choix de supprimer cette variable.

La distribution de données pour les variables « couleur » et « occasion » étant sensiblement les mêmes, nous pouvons supprimer aussi ces deux variables.

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

En résumé, nous supprimons les variables suivantes de la data-frame « `clients_immatriculations` » :

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

Nous réalisons ces suppressions grâce au code suivant :

```
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)
```

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

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

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

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

Une fois toutes ces étapes réalisées, nous allons réaliser plusieurs jeux de données de test afin d'identifier le meilleur classifieur. Pour cela, nous allons créer 2 jeux de tests.

Nous nous sommes aperçus que durant nos différents tests, la variable taux contenant près de 756 niveaux de facteurs pouvait être à l'origine de nombreux problèmes (Notamment pour les classifieurs : Nnet, kkn et svm). Pour cela, nous faisons le choix que notre premier jeu de test ne comportera pas la variable taux. Pour notre deuxième jeu de données, nous décidons qu'à l'image de la variable catégorie pour les véhicules, nous allons créer une catégorie pour le taux.

Ainsi, nous allons établir nos ensembles d'apprentissage (représentant environ 70% de nos données de la data frame de base) et nos ensembles de tests (représentant les 30% restant).

Nous réalisons donc notre premier jeu de données de la manière suivante :

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

clients_immatriculations_EA <- clients_immatriculations[1:29014,]
|
"clients_immatriculations : sélection des dernières lignes de clients_immatriculations.(30% de données)"

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

Maintenant que nos deux premiers ensembles ont été établis on supprime la variable taux par le code suivant :

"suppression de la colonne taux de ce data-frame"

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

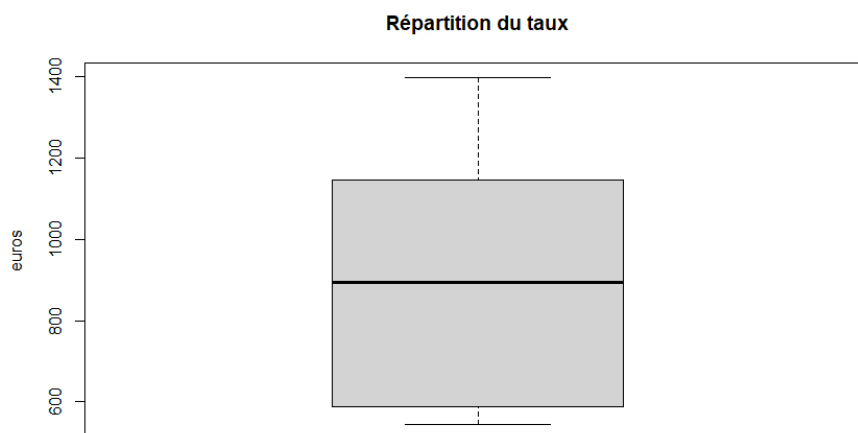
Maintenant que nous avons créé nos 1^{er} jeux de données de test (avec les data-frames clients_immatriculations_EA et clients_immatriculations_ET), nous allons créer notre deuxième jeu de tests.

Nous réalisons le code suivant afin de créer l'ensemble d'apprentissage et l'ensemble de test numéro 2 :

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

clients_immatriculations_ET2 <- clients_immatriculations[29015:43521,]
```

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



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

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

Au vu de ce résultat, nous faisons le choix de créer 4 catégories de taux selon les critères suivants :

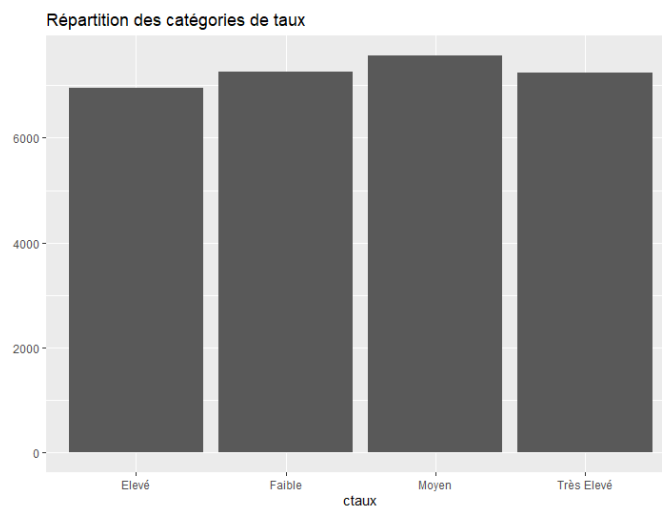
Catégorie de taux	Borne
Faible	[0 ;589[
Moyen	[589 ;901.7[
Elevé	[901.7 ;1147[
Très Elevé	[1147 ; [

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

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

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

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



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

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

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

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

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

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

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

3 Test des classifieurs et création de notre modèle de prédiction

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

3.1 Liste des classifieurs

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

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

Nous importons les packages contenant les classifieurs précédents avec le code suivant :

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

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

3.2 1^{er} jeu de test

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

- clients_immatriculations_EA
- clients_immatriculations_ET

3.2.1 C5.0

Commençons avec C5.0. Tout d'abord nous réalisons l'apprentissage du classifieur par le code suivant :

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

Nous réalisons ensuite les prédictions sur l'ensemble de test.

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

Réalisons maintenant la matrice de confusion.

```
table(clients_immatriculations_ET$categories, C_class)

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

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

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2424	3	0	400	1287	0,589207584		
citadine	2	4632	0	0	1	0,999352751		
compacte	0	1003	0	1	1	0		
routière	30	3	0	744	717	0,497991968		
sportive	475	1	0	0	2783	0,853942927		
Précision	0,827021494	0,820985466	#DIV/0!	0,649781659	0,581123408	0,729509892		
							Taux d'erreur	0,270490108

On mesure donc un taux d'erreur de 0.27.

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

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

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

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

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

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

Ainsi avec C5.0 nous avons un auc de 0.8947 sur ce 1^{er} jeu de test.

3.2.2 Naive-Bayes

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

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

Réaliser les prédictions sur l'ensemble de test.

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

Obtenir la matrice de confusion.

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

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

Calculer les différents indices.

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2484	55	0	400	1175	0,60379193		
citadine	860	3444	330	0	1	0,743042071		
compacte	0	652	351	1	1	0,349253731		
routière	67	35	0	744	648	0,497991968		
sportive	601	113	0	0	2545	0,780914391		
Précision	0,619142572	0,801116539	0,515418502	0,649781659	0,582379863	0,659543669		
							Taux d'erreur	0,340456331

Nous remarquons ici un taux d'erreur de 0.34 pour le classifieur naive-bayes.

Enfin la dernière étape est l'obtention de l'auc.

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

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

Pour le classifieur naive-bayes nous obtenons un auc de 0.8873.

3.2.3 SVM

Pour le classifieur nous obtenons la matrice de confusion suivante :

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

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


A partir de cela on obtient les indicateurs suivant :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2597	5	0	288	1224	0,631259115		
citadine	2	4632	0	0	1	0,999352751		
compacte	1	1003	0	0	1	0		
routière	286	4	0	522	682	0,34939759		
sportive	602	4	0	0	2653	0,814053391		
Précision	0,744552752	0,820113314	#DIV/0!	0,644444444	0,581670686	0,717171021		
							Taux d'erreur	0,282828979

Pour ce classifieur on a un taux de d'erreur de 0.28.

Maintenant déterminons l'auc du classifieur svm.

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

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

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

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

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

#_____#
```

Nous obtenons donc un auc de 0.9019

3.2.4 Kknn

Pour ce classifieur nous obtenons la matrice de confusion suivante :

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

A partir de cette matrice nous pouvons calculer les indices associés :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2597	2	1	396	1136	0,628509197		
citadine	2	4178	454	0	1	0,901402373		
compacte	1	724	279	1	0	0,27761194		
routière	400	3	0	558	533	0,373493976		
sportive	911	0	1	270	2077	0,637312059		
Précision	0,664024546	0,851436723	0,379591837	0,455510204	0,554310115	0,667056799		
							Taux d'erreur	0,332943201

Nous observons donc un taux d'erreur pour ce classifieur de 0.33.

Maintenons déterminons son auc de la manière suivante :

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

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

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

Pour kkn nous obtenons un auc de 0.87.

3.2.5 Nnet

Pour ce classifieur nous obtenons la matrice de confusion suivante :

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

L'étape suivante est le calcul des indices :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2414	3	0	408	1289	0,58677686		
citadine	2	4469	163	0	1	0,964185545		
compacte	0	802	201	1	1	0,2		
routière	19	3	0	755	717	0,505354752		
sportive	482	1	0	0	2776	0,851795029		
Précision	0,827562564	0,846722243	0,552197802	0,64862543	0,580267559	0,731715723		
							Taux d'erreur	0,268284277

Nous obtenons un taux d'erreur de 0.268.

Maintenant déterminons l'auc :

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

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

L'auc du classifieur nnet est de 0.9076.

3.2.6 Randomforest

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

Nous supprimons la variable age de nos ensembles d'apprentissage et de test.

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

Maintenant déterminons la matrice de confusion :

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

A partir de cela déterminons les indices lié à ce classifieurs :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2411	3	0	0	1700	0,586047642		
citadine	2	4632	0	0	1	0,999352751		
compacte	0	1003	0	0	2	0		
routière	1	3	0	0	1490	0		
sportive	475	1	0	0	2783	0,853942927		
Précision	0,834544825	0,820985466	#DIV/0!	#DIV/0!	0,465696118	0,677328186		
							Taux d'erreur	0,322671814

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

Maintenant déterminons l'auc de ce classifieur :

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

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

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

Nous obtenons donc un auc de 0.693

3.2.7 Conclusion sur ce 1^{er} test

Résumons nos derniers résultat à travers ce tableau :

Classifieur	Taux d'erreur	Auc
C5.0	0.27	0.8947
Naive Bayes	0.34	0.8873
SVM	0.28	0.9019
Kknn	0.33	0.87
Nnet	0.268	0.9076
Random Forest	0.323	0.6938

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

3.3 2eme jeu de test

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

- clients_immatriculations_EA2
- clients_immatriculations_ET2

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

3.3.1 C5.0

Pour ce classifieur nous obtenons la matrice de confusions suivante :

```
# Matrice de confusion
table(clients_immatriculations_ET2$categories, C_class2)

##          C_class2
##      berline citadine compacte routière sportive
##berline    2226      2        1      545    1340
##citadine      2    4263      369      0        1
##compacte      0     652      351      1        1
##routière      0      3        0    1352     139
##sportive    105      1        0     461    2692
```

Nous calculons les indices de performance lié à ce classifieur.

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2226	2	1	545	1340	0,541079242		
citadine	2	4263	369	0	1	0,9197411		
compacte	0	652	351	1	1	0,349253731		
routière	0	3	0	1352	139	0,904953146		
sportive	105	1	0	461	2692	0,826020252		
Précision	0,954136305	0,86628734	0,486823856	0,573124205	0,645099449	0,750258496		
							Taux d'erreur	0,249741504

Déterminons l'auc :

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

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

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

Pour ce classifieur nous obtenons un auc de 0.9349.

3.3.2 Naive Bayes

Déterminons la matrice de confusion :

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

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

Calculons le rappel, la précision et le taux d'erreur pour ce classifieur :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2592	246	1	240	1035	0,630043753		
citadine	855	3374	400	0	6	0,72793959		
compacte	0	628	376	1	0	0,374129353		
routière	269	67	0	457	701	0,305890228		
sportive	571	364	1	0	2323	0,712795336		
Précision	0,604618614	0,721094251	0,483290488	0,654727794	0,571463715	0,62879989		
							Taux d'erreur	0,37120011

Ainsi pour ce classifieur nous avons un taux d'erreur de 0.37.

Maintenant déterminons l'auc :

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

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

Pour le classifieur naive bayes on mesure un auc de 0.915.

3.3.3 SVM

Etablissons la matrice de confusion liée à ce classifieur :

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

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

Ensuite, calculons les indicateurs lié à cette matrice :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2788	5	0	217	1104	0,67768595		
citadine	2	4632	0	0	1	0,999352751		
compacte	1	1003	0	0	1	0		
routière	627	9	0	728	130	0,487282463		
sportive	602	4	0	431	2222	0,681804234		
Précision	0,693532338	0,819387936	#DIV/0!	0,529069767	0,642567958	0,714827325		
							Taux d'erreur	0,285172675

On peut mesurer un taux d'erreur de 0.285 pour le classifieur svm.

Maintenant déterminons son auc :

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

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

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

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

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

Ainsi l'auc de svm est de 0.9332.

3.3.4 Kknn

Déterminons la matrice de confusion :

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

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

Calons les indicateurs pour ce classifieur :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2632	3	0	336	1143	0,63976665		
citadine	2	4136	496	0	1	0,892340885		
compacte	0	569	434	0	2	0,431840796		
routière	247	3	0	831	413	0,5562249		
sportive	673	1	1	318	2266	0,695305308		
Précision	0,740574001	0,877758913	0,466165414	0,55959596	0,592418301	0,709933136		
							Taux d'erreur	0,290066864

On note un taux d'erreur de 0.29 pour ce classifieur.

Enfin déterminons l'auc :

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

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

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

Ainsi, nous observons un auc de 0.9164 pour le classifieur kknn.

3.3.5 Nnet

Matrice de confusion :

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

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

Calcul des indicateurs :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2247	2	1	504	1360	0,546183763		
citadine	2	4226	406	0	1	0,91175836		
compacte	0	606	397	1	1	0,395024876		
routière	34	3	0	1259	198	0,84270415		
sportive	120	0	1	421	2717	0,833691316		
Précision	0,935081149	0,873682034	0,493167702	0,576201373	0,635258359	0,747639071		
							Taux d'erreur	0,252360929

Le classifieur Nnet sur ce jeu de test a donc un taux d'erreur de 0.25

Calcul de l'auc :

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

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

Nous observons que l'auc du classifieur nnet est de 0.9463.

3.3.6 Random Forest

De la même manière que dans le 1^{er} je de test nous supprimons la variable age des deux data-frames.

Matrice de confusion :

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

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

Calcul des indicateurs :

	berline	citadine	compacte	routière	sportive	Rappel		
berline	2078	3	0	231	1802	0,505104521		
citadine	2	4632	0	0	1	0,999352751		
compacte	0	1003	0	0	2	0		
routière	0	3	0	766	725	0,512717537		
sportive	90	1	0	461	2707	0,83062289		
Précision	0,957603687	0,820985466	#DIV/0!	0,525377229	0,516898988	0,701936996		
							Taux d'erreur	0,298063004

Calcul de l'auc :

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

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

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

Nous remarquons un auc de 0.8156 pour ce classifieurs.

3.3.7 Conclusion

Résumons nos derniers résultats à travers ce tableau :

Classifieur	Taux d'erreur	Auc
C5.0	0.25	0.9349
Naive Bayes	0.37	0.915
SVM	0.285	0.9332
Kknn	0.29	0.9164
Nnet	0.25	0.9463
Random Forest	0.298	0.8156

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

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

3.4 Prédiction

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

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

Nous créons donc ces deux ensembles :

```
##----- création de deux nouveaux ensembles d'apprentissage et de test
clients_immatriculations_EA3 <- clients_immatriculations[1:29014,]
clients_immatriculations_ET3 <- clients_immatriculations[29015:43521,]
```

Une fois ces ensembles créés appliquons le classifieur C5.0.

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

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

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

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

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

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

Une fois ces étapes réalisées, il nous reste plus qu'à établir les prédictions. Pour cela nous créons la variable `class.treeCpred`, qui contiendra les prédictions de catégories pour chaque profil du data-frame `Marketing`.

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

Enfin nous créons le data-frame `resultat1`. Ce data-frame sera la concaténation des dataframe `marketing` et `class.treeCpred`.

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

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

Ainsi, nous observons qu'une catégorie de véhicule a bien été prédite pour chacun des profils du fichier `marketing`. Nous avons donc établi notre modèle de prédiction. Enfin nous générons, grâce au code suivant, un fichier `csv` regroupant toute les données du data-frame `resultat1`.

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

4 Conclusion

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

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

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

Enfin, une fois que nous avons sélectionné le meilleur classifieur, ici C5.0, nous n'avons plus qu'à réaliser les prédictions de catégorie de véhicules pour le data-frame marketing. Ainsi nous avons réussi à établir notre modèle de prédiction. Une des pistes d'amélioration pourrait être de prédire une marque et le nom d'un véhicule en plus de la catégorie de voiture qui pourrait convenir à chaque profil.

Annexe

Importation des données via un driver sql

Dans cette partie nous allons vous présenter comment nous avons procéder pour réaliser le nettoyage de nos données à travers l'utilisation d'Oracle et de commandes sql.

Pour mener à bien ce nettoyage nous alors d'abord nous connecter à notre base de données par un driver :

```
##classPath : add path to drivers jdbc  
drv <- RJDBC::JDBC(driverClass = "oracle.jdbc.OracleDriver", classPath =  
  sys.glob("C:/Logiciels/oracle/oracle/drivers/*"))  
  
#Connexion OK  
conn <- dbConnect(drv, "jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)  
  (HOST=144.21.67.201)(PORT=1521))(CONNECT_DATA=  
  (SERVICE_NAME=pdbest21.631174089.oraclecloud.internal)))",  
  "GUINALD02B20", "GUINALD02B2001")
```

Une fois ce code exécuté, il ne nous reste plus qu'à importer nos données provenant des csv de la manière suivante :

```

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

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

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

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

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

names(Client)[6] = ("deuxiemevoiture")

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

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

```

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

Enfin une fois que ces fichiers ont bien été importés nous pouvons visualiser cela avec les commandes suivantes :

```

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

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

```

Tri des données sur sql

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

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

Pour la variable age nous effectuons la commande suivante :

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

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

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

Enfin nous supprimons toutes données non-conformes :

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

Ensuite nous supprimons toutes les données identifier comme outlier pour la variable taux :

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

A l'image de la variable sexe, nous supprimons toutes les valeurs non conformes pour la variable situationfamiliale :

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

L'étape suivante est la suppression de toutes les données outlier pour la variable nbEnfantAcharge :

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

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

```
delete from client where deuxiemevoiture !='true' and deuxiemevoiture !='false';
```

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