Analyse des Données du sous projet BDA/DL par les Techniques de Data Mining, Machine Learning et Deep Learning et Activités Attendues par N. PASQUIER ou un le professeur qui le remplace

Nom du membre du Groupe 6:

Gao LIU

Pierric SCHOENDORF

Yimin CAO

Ziheng WANG

Table des matières

Table des figures

Liste des tableaux

Partie Machine Learning

1. RStudio et langage R

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

Un système de documentation intégré très bien conçu

Des procédures efficaces pour le traitement et le stockage de données

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

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

Des capacités graphiques évoluées

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

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

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

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

Figure 1 Connexion entre Hive et RStudio pour l'importation de tables

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

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

Figure 2 Récapitulatif des Clients

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

Figure 3 Récapitulatif des Immatriculations

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

Figure 4 Récapitulatif des Marketing

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

Figure 5 Récapitulatif des Catalogue

```
> Co2 <- dbGetQuery(conn, "select * from Co2")
> summary(Co2)
     co2.id      co2.marque_modele  co2.bonus_malus    co2.rejets_co2_g_km co2.cout_enerie
 Min.   :  2.0   Length:438         Length:438         Min.   :  0.0       Length:438
 1st Qu.:116.0   Class :character   Class :character   1st Qu.: 42.5       Class :character
 Median :230.0   Mode  :character   Mode  :character   Median :182.0       Mode  :character
 Mean   :230.4                                        Mean   :139.9
 3rd Qu.:345.0                                        3rd Qu.:197.0
 Max.   :459.0                                        Max.   :262.0
 NA's   :1                                            NA's   :3
```

Figure 6 Récapitulatif des Co2

2. Nettoyage des données

2.1 Nettoyage des données : Clients

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

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

Figure 7 Changement du nom et suppression de nulles

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

```
> Clients_traite1 <- subset(Clients_sans_na, age >= 18 )
> library(ggplot2)
> ggplot(Clients_traite1, aes(x=sexe))+geom_bar()
```

Figure 8 Respect de la contrainte de l'âge

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

```
> Clients_traite1$sexe <- gsub("Masculin", "M", Clients_traite1$sexe)
> Clients_traite1$sexe <- gsub("Homme", "M", Clients_traite1$sexe)
> Clients_traite1$sexe <- gsub("Femme", "F", Clients_traite1$sexe)
> Clients_traite1$sexe <- gsub("F?minin", "F", Clients_traite1$sexe)
> Clients_traite1 <- subset(Clients_traite1, Clients_traite1$sexe!="N/D"&Clients_traite1$sexe!="?"&Clients_traite1$sexe!="")
> Clients_traite1$sexe <- as.factor(Clients_traite1$sexe)
> ggplot(Clients_traite1, aes(x=sexe))+geom_bar()
```

Figure 9 Corrige du nom de colonne et valeur erronées

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

```
> Clients_traite1 <- subset(Clients_traite1, taux >= 544 )
> ggplot(Clients_traite1, aes(x=situationfamiliale))+geom_bar()
```

Figure 10 Respect de la contrainte de taux

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

```
> Clients_traite1 <- subset(Clients_traite1, Clients_traite1$situationfamiliale!="N/D" & Clients_traite1$situationfamiliale!="?" & Clients_traite1$sit
uationfamiliale!="")
> Clients_traite1$situationfamiliale <- gsub("C<U+FFFD>libataire", "Celibataire", Clients_traite1$situationfamiliale)
> Clients_traite1$situationfamiliale <- gsub("Divorc<U+FFFD>e", "Divorce", Clients_traite1$situationfamiliale)
> Clients_traite1$situationfamiliale <- gsub("Mari<U+FFFD>", "Marie", Clients_traite1$situationfamiliale)
> Clients_traite1$situationfamiliale <- as.factor(Clients_traite1$situationfamiliale)
```

Figure 11 Corrige du nom de colonne de client

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

```
> #nbenfantsacharge >=0
> Clients_traite1 <- subset(Clients_traite1, nbenfantsacharge >= 0 )
>
> #garder que des lignes en format 9999 AA 99 dans la colonne immatriculation
> Clients_traite<- Clients_traite1[grep("^\\d{4} [A-Z]{2} \\d{2}$", Clients_traite1$immatriculation), ]
> summary(Clients_traite)
      age          sexe           taux          situationfamiliale nbenfantsacharge deuxiemevoiture immatriculation
 Min.   :18.00   F:23655   Min.   : 544.0   Celibataire:23390    Min.   :0.000    Mode :logical   Length:78740
 1st Qu.:27.00   M:55085   1st Qu.: 588.0   Divorce    :   36    1st Qu.:0.000    FALSE:68482     Class :character
 Median :41.00             Median : 888.0   En Couple  :50716    Median :1.000    TRUE :10258     Mode  :character
 Mean   :43.72             Mean   : 899.2   Marie(e)   :  530    Mean   :1.252
 3rd Qu.:57.00             3rd Qu.:1144.0   Seul       :  212    3rd Qu.:2.000
 Max.   :84.00             Max.   :1399.0   Seule      : 3856    Max.   :4.000
```

Figure 12 Respect de la contrainte de nbEnfantsacharge

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

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

Figure 13 Suppression de double

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

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

Figure 14 Récapitulatif du tableau des Clients nettoyés

2.2 Nettoyage des données : Catalogue

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

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

Figure 15 Renommer les colonnes

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

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

Figure 16 Convert to format demandé

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

```
> Catalogue_sans_na <- na.omit(Catalogue)
> Catalogue_sans_na <- droplevels(Catalogue_sans_na)
>
> summary(Catalogue_sans_na)
       id                marque            nom        puissance            longueur
 Min.   :  2.00   Renault   : 40   1007 1.4  : 10   Min.   : 55.0   courte     :60
 1st Qu.: 69.25   Volkswagen: 40   120i      : 10   1st Qu.:109.0   longue     :90
 Median :136.50   Audi      : 20   9.3 1.8T  : 10   Median :147.0   moyenne    :70
 Mean   :136.50   BMW       : 20   A2 1.4    : 10   Mean   :157.6   tr<U+FFFD>s longue:50
 3rd Qu.:203.75   Mercedes  : 20   A200      : 10   3rd Qu.:170.0
 Max.   :271.00   Nissan    : 15   A3 2.0 FSI: 10   Max.   :507.0
                  (Other)   :115   (Other)   :210
    nbplaces        nbportes      couleur    occasion           prix
 Min.   :5.000   Min.   :3.000   blanc:54   Mode :logical   Min.   :  7500
 1st Qu.:5.000   1st Qu.:5.000   bleu :54   FALSE:160       1st Qu.: 16029
 Median :5.000   Median :5.000   gris :54   TRUE :110       Median : 20598
 Mean   :5.222   Mean   :4.815   noir :54                   Mean   : 26668
 3rd Qu.:5.000   3rd Qu.:5.000   rouge:54                   3rd Qu.: 30000
 Max.   :7.000   Max.   :5.000                              Max.   :101300
```

Figure 17 Suppression de nulles

Nous aussi corrigions les valeurs des colonnes qui sont erronées.

```
> Catalogue_sans_na$longueur <- gsub("tr<U+FFFD>s longue", "tres longue", Catalogue_sans_na$longue
ur)
> Catalogue_sans_na$longueur <- as.factor(Catalogue_sans_na$longueur)
```

Figure 18 Nettoyage de caractère incorrect

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

```
> summary(Catalogue_traite)
       id                marque            nom        puissance            longueur
 Min.   :  2.00   Renault   : 40   1007 1.4  : 10   Min.   : 55.0   courte     :60
 1st Qu.: 69.25   Volkswagen: 40   120i      : 10   1st Qu.:109.0   longue     :90
 Median :136.50   Audi      : 20   9.3 1.8T  : 10   Median :147.0   moyenne    :70
 Mean   :136.50   BMW       : 20   A2 1.4    : 10   Mean   :157.6   tres longue:50
 3rd Qu.:203.75   Mercedes  : 20   A200      : 10   3rd Qu.:170.0
 Max.   :271.00   Nissan    : 15   A3 2.0 FSI: 10   Max.   :507.0
                  (Other)   :115   (Other)   :210
    nbplaces        nbportes      couleur    occasion           prix
 Min.   :5.000   Min.   :3.000   blanc:54   Mode :logical   Min.   :  7500
 1st Qu.:5.000   1st Qu.:5.000   bleu :54   FALSE:160       1st Qu.: 16029
 Median :5.000   Median :5.000   gris :54   TRUE :110       Median : 20598
 Mean   :5.222   Mean   :4.815   noir :54                   Mean   : 26668
 3rd Qu.:5.000   3rd Qu.:5.000   rouge:54                   3rd Qu.: 30000
 Max.   :7.000   Max.   :5.000                              Max.   :101300
```

Figure 19 Récapitulatif du tableau du Catalogue nettoyé

2.3 Nettoyage des données : Immatriculations

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

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

Figure 20 Corrige du nom de colonne

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

Figure 21 Convert to format demandé

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

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

Figure 22 Suppression ID, convert format, suppression double, et corrige valeur erronée

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

```
>
> summary(Immatriculations_traite)
       marque                  nom          puissance          longueur        nbplaces    nbportes      couleur
 BMW       :264567   A2 1.4          :255022   Min.   : 55   courte     :494211   Min.   :5   Min.   :3.000   blanc:358278
 Audi      :262265   M5              :230523   1st Qu.: 75   longue     :489744   1st Qu.:5   1st Qu.:5.000   bleu :360023
 Renault   :225923   X-Type 2.5 V6   :169630   Median :150   moyenne    :207526   Median :5   Median :5.000   gris :360327
 Jaguar    :169630   S80 T6          :111244   Mean   :199   tres longue:605371   Mean   :5   Mean   :4.868   noir :359381
 Volkswagen:140117   Vel Satis 3.5 V6:110671   3rd Qu.:245                       3rd Qu.:5   3rd Qu.:5.000   rouge:358843
 Mercedes  :134842   S500            : 93180   Max.   :507                       Max.   :5   Max.   :5.000
 (Other)   :599508   (Other)         :826582
  occasion            prix          immatriculation
 Mode :logical   Min.   :  7500   1000 AD 49:      1
 FALSE:1234892   1st Qu.: 18310   1000 AD 66:      1
 TRUE :561960    Median : 25970   1000 AE 54:      1
                 Mean   : 35778   1000 AG 15:      1
                 3rd Qu.: 49200   1000 AM 17:      1
                 Max.   :101300   1000 AQ 88:      1
                                  (Other)   :1796846
```

Figure 23 Récapitulatif du tableau des Immatriculations nettoyés

3. Analyse exploratoire des données vues

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

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

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

Figure 24 Code de dessin de couleur de catalogue

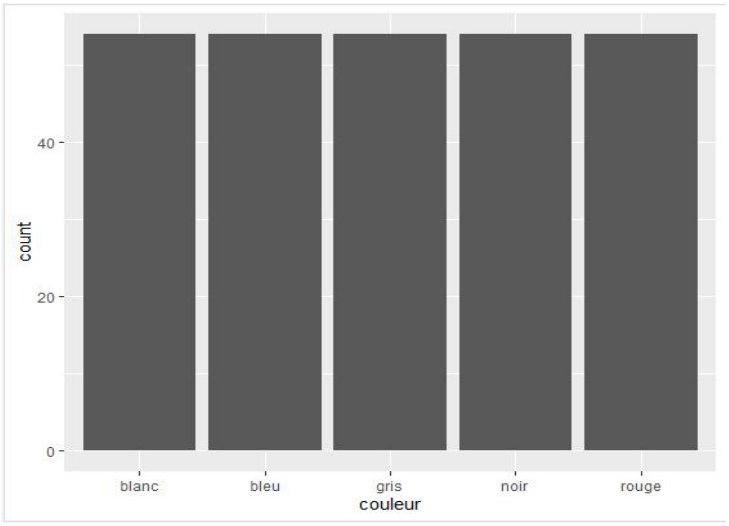

Figure 25 La répartition de la variable couleur

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

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

Figure 26 Code de dessin de longueur de catalogue

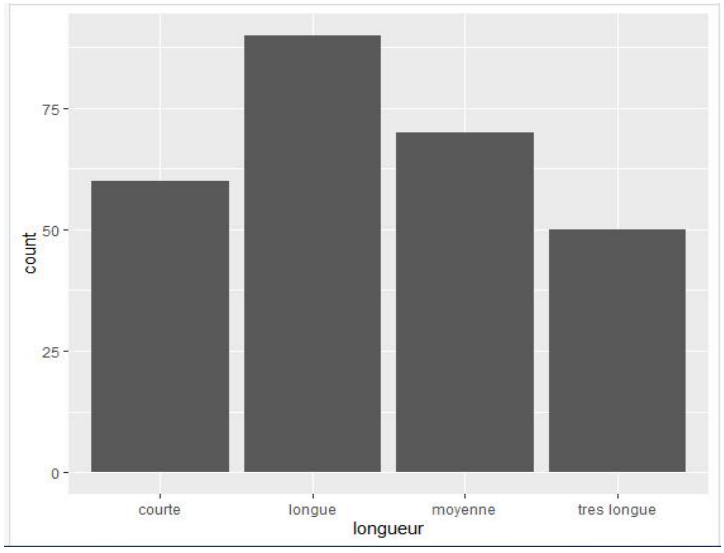

Figure 27 La répartition de la variable longueur

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

```
> ggplot(Catalogue_traite, aes(x=nbplaces))+geom_bar()
```

Figure 28 Code de dessin de nbplaces de catalogue

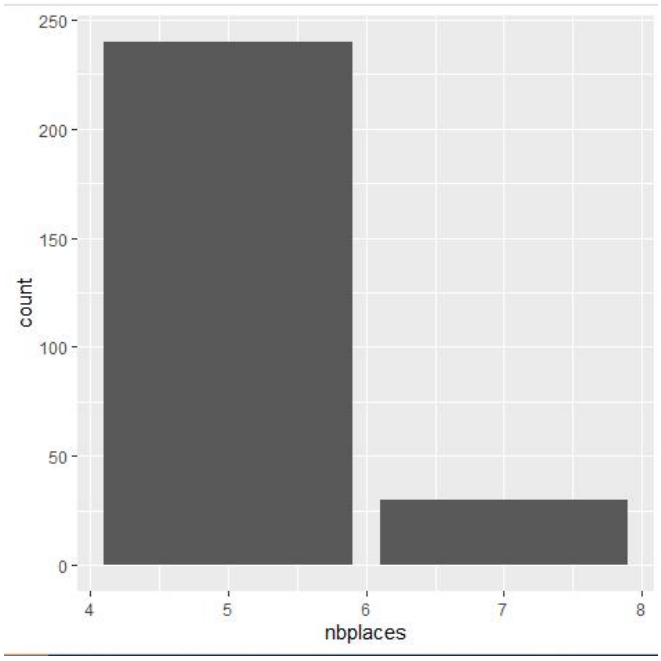

Figure 29 La répartition de la variable nbPlaces

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

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

Figure 30 Code de dessin de nbPortes de catalogue

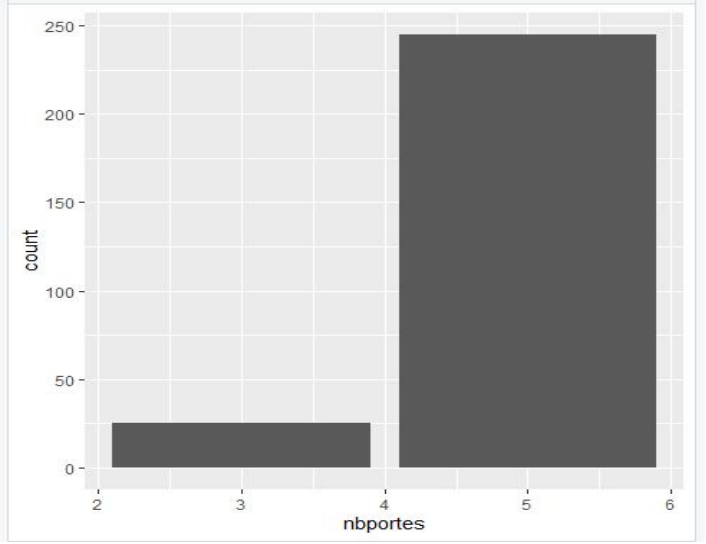

Figure 31 La répartition de la variable nbPortes

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

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

Figure 32 Code de dessin de puissance de catalogue

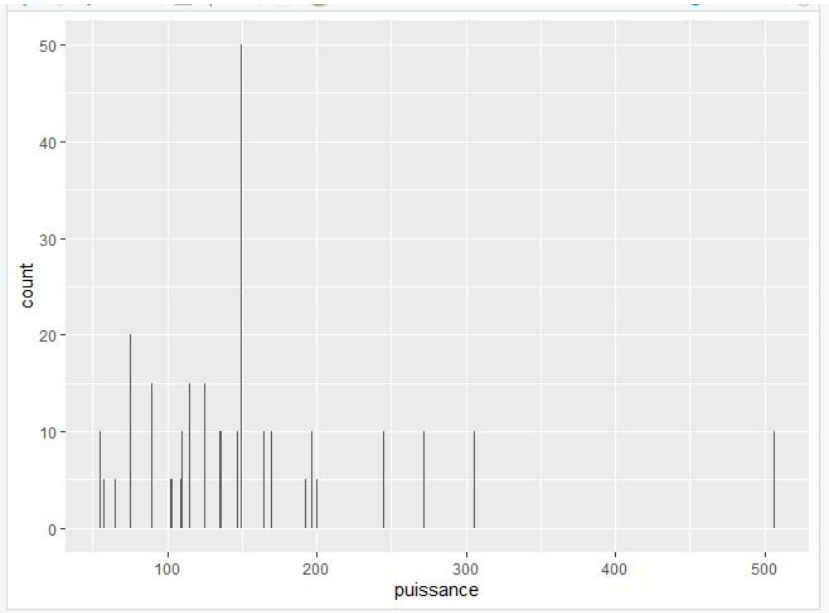

Figure 33 La répartition de la variable puissance

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

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

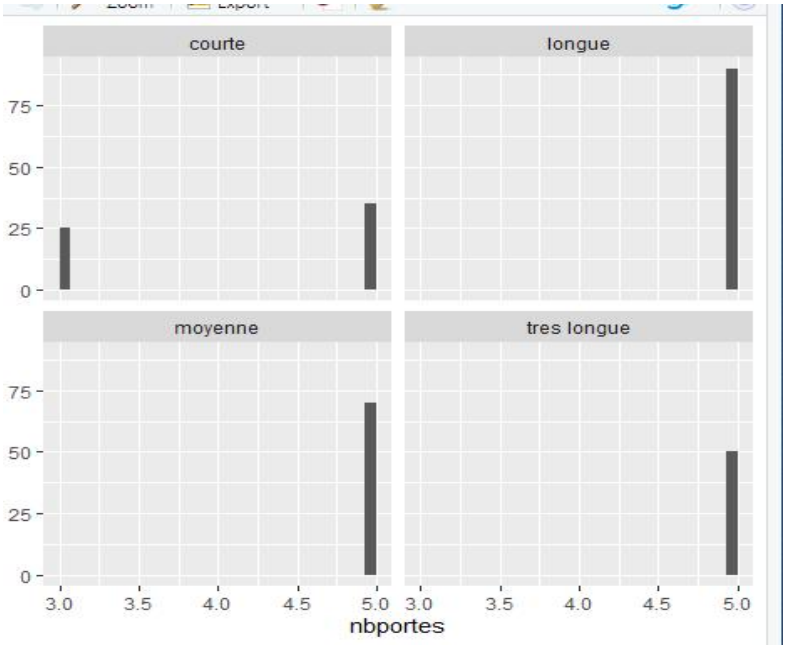

Figure 34 La répartition de la variable nbPortes de chaque type de longueur

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

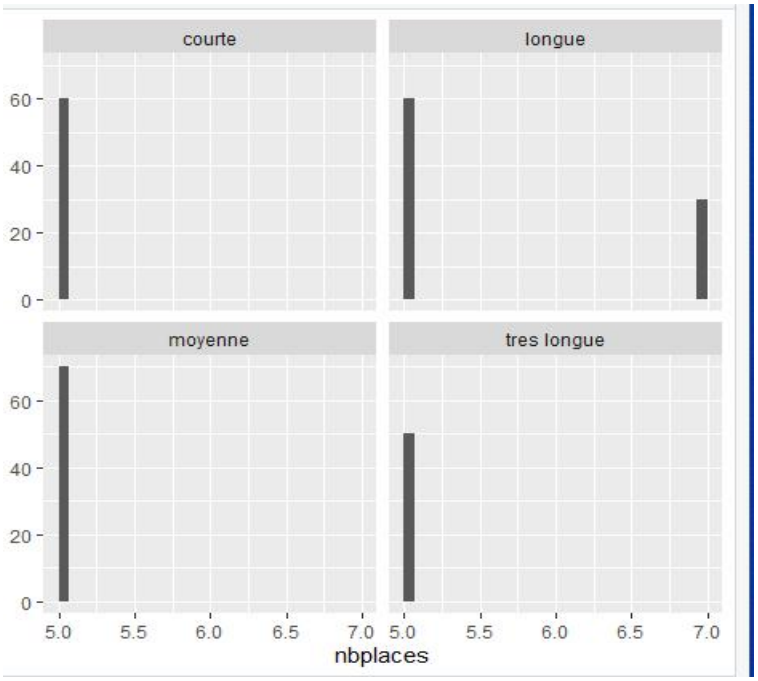

Figure 35 La répartition de la variable nbPlaces de chaque type de longueur

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

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

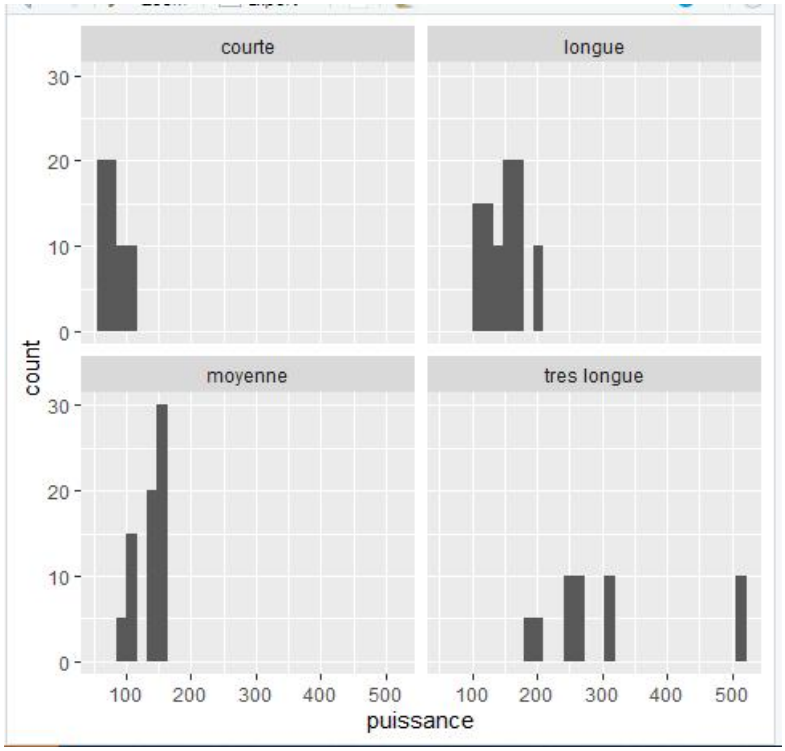

Figure 36 La répartition de la variable puissance de chaque type de longueur

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

4. Identification des catégories de véhicules et application aux Immatriculations

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

- La longueur de la voiture

- La puissance de la voiture

- Le nombre de places

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

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

Figure 37 Categories à partir de Catalogue

```
>
> Immatriculations_traite$categories <- ifelse(Immatriculations_traite$longueur=="courte","minicar",
+                                              ifelse(Immatriculations_traite$longueur=="moyenne","compacte",
+                                                     ifelse(Immatriculations_traite$longueur=="longue"& Immatriculations_traite$puissance < 1
85 & Immatriculations_traite$nbplaces == 5,"Routiere",
+                                                            ifelse(Immatriculations_traite$longueur=="longue"& Immatriculations_traite$puissance <
 185 & Immatriculations_traite$nbplaces == 7,"suv",
+                                                                   ifelse(Immatriculations_traite$longueur=="longue" | Immatriculations_traite$longu
eur=="tres longue" & Immatriculations_traite$puissance >=185 & Immatriculations_traite$puissance <250,"sportive",
+                                                                          ifelse(Immatriculations_traite$longueur == "tres longue" & Immatriculation
s_traite$puissance >= 250, "berline","Rien"))))))
>
>
> Immatriculations_traite$categories <- as.factor(Immatriculations_traite$categories)
> summary(Immatriculations_traite)
```

Figure 38 Application de catégories aux Immatriculations

5. Fusion des données Clients et Immatriculations

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

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

```
> Clients_Immatriculations <- merge(Immatriculations_traite, Clients_traite, by ="immatriculation")
>
> Clients_Immatriculations <- Clients_Immatriculations[,-1]
>
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -marque)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -nom)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -puissance)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -longueur)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -nbplaces)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -nbportes)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -couleur)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -occasion)
> Clients_Immatriculations <- subset(Clients_Immatriculations, select = -prix)
>
> summary(Clients_Immatriculations)
    categories         age        sexe          taux          situationfamiliale nbenfantsacharge deuxiemevoiture
 berline :24326   Min.   :18.00   F:23653   Min.   : 544.0   Celibataire:23389    Min.   :0.000    Mode :logical
 compacte: 5706   1st Qu.:27.00   M:55081   1st Qu.: 588.0   Divorce    :   36    1st Qu.:0.000    FALSE:68476
 minicar :24963   Median :41.00             Median : 888.0   En Couple  :50711    Median :1.000    TRUE :10258
 Routiere: 8350   Mean   :43.72             Mean   : 899.2   Marie(e)   :  530    Mean   :1.252
 sportive:15389   3rd Qu.:57.00             3rd Qu.:1144.0   Seul       :  212    3rd Qu.:2.000
                  Max.   :84.00             Max.   :1399.0   Seule      : 3856    Max.   :4.000
>
```

Figure 39 Fusion de data frames et suppression de colonnes inutiles

6. Modèle de classification et les tests dans différents algorithmes

6.1 Situation 1 : normal

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

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

Figure 40 Convert to format

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

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

Figure 41 Situation1_Séparation de l'échantillonnage d'apprentissage et test

6.1.1 Algorithme : C50

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

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

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

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

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

Tree size: 12

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

Figure 42 Formation de jeux d'apprentissage avec C5.0

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

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

Figure 43 Prédiction de de jeu du test avec C50 et affichage des résultats prédits

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

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

Figure 44 Situation1_C50 : Dessin de la matrice de mélange

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

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

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

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

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

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

Figure 45 Situation1_C50 : calcul de l'AUC

6.1.2 Algorithme : naivebayes

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

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

Figure 46 Formation de jeux d'apprentissage avec naivebayes

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

Figure 47 Prédiction de de jeu du test avec naivebayes

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

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

Figure 48 Situation1_naivebayes : Dessin de la matrice de mélange

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

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

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

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

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

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

Figure 49 Situation1_Naivebayes : calcul de l'AUC

6.1.3 Algorithme : Random Forest

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

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

Figure 50 Formation de jeux d'apprentissage avec random forest

6.1.4 Algorithme : NNET

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

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

Figure 51 Formation de jeux d'apprentissage avec NNET

6.1.5 Algorithme : Régression logistique multinomiale

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

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

Figure 52 Formation de jeux d'apprentissage avec Régression logistique multinomiale

6.2 Situation 2 : supprimer la colonne taux

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

```
> Clients_Immatriculations_traite_EA <- subset(Clients_Immatriculations_traite_EA , select = -taux)
> Clients_Immatriculations_traite_ET <- subset(Clients_Immatriculations_traite_ET , select = -taux)
```

Figure 53 Situation2_Séparation de l'échantillonnage d'apprentissage et test

6.2.1 Algorithme : C50

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

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

Figure 54 Situation2_C50 résultat de prédiction

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

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

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

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

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

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

Figure 55 Situation2_C50 : calcul de l'AUC

6.2.2 Algorithme : Naivebayes

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

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

Figure 56 Situation2_Naivebayes résultat de prédiction

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

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

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

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

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

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

Figure 57 Situation2_ Naivebayes: calcul de l'AUC

6.2.3 Algorithme : Random Forest

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

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

Figure 58 Formation de jeux d'apprentissage avec random forest

6.2.4 Algorithme : NNET

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

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

Figure 59 Formation de jeux d'apprentissage avec NNET

```
> nn_class <- predict(nnet, Clients_Immatriculations_traite_ET, type="class")
```

```
> table(nn_class)
nn_class
 berline compacte  minicar Routiere sportive
    4769      764     8483     1873     7731
```

Figure 60 Prédiction de de jeu du test avec NNET

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

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

Figure 61 Situation2_NNET : Dessin de la matrice de mélange

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

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

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

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

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

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

Figure 62 Situation2_NNET : calcul de l'AUC

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

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

```
> Clients_Immatriculations_traite_EA <- subset(Clients_Immatriculations_traite_EA , select = -taux)
> Clients_Immatriculations_traite_ET <- subset(Clients_Immatriculations_traite_ET , select = -taux)

> Clients_Immatriculations_traite_EA <- subset(Clients_Immatriculations_traite_EA , select = -age)
> Clients_Immatriculations_traite_ET <- subset(Clients_Immatriculations_traite_ET , select = -age)
```

Figure 63 Situation3_Séparation de l'échantillonnage d'apprentissage et test

6.3.1 Algorithme : C50

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

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

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

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

Tree size: 6

Non-standard options: attempt to group attributes

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

Figure 64 Situation3_C50 résultat de prédiction

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

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

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

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

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

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

Figure 65 Situation3_C50 : calcul de l'AUC

6.3.2 Algorithme : Naviebayes

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

```
> table(nb_class)
nb_class
 berline compacte  minicar Routiere sportive
    6562        0    10576        0     6482
> table( Clients_Immatriculations_traite_ET$categories, nb_class)
          nb_class
           berline compacte minicar Routiere sportive
  berline     4441        0     848        0     1993
  compacte       0        0    1695        0        1
  minicar     1465        0    6087        0        1
  Routiere     103        0     714        0     1673
  sportive     553        0    1232        0     2814
```

Figure 66 Situation3_Naivebayes résultat de prédiction

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

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

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

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

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

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

Figure 67 Situation3_Naivebayes : calcul de l'AUC

6.3.3 Algorithme : Random Forest

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

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

Figure 68 Situation3_Random Forest résultat de prédiction

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

Tableau 8 Situation3_Random Forest : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 69 Situation3_ Random Forest : calcul de l'AUC

6.3.4 Algorithme : NNET

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

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

Figure 70 Situation3_NNET résultat de prédiction

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

Tableau 9 Situation3_NNET : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 71 Situation3_ NNET : calcul de l'AUC

6.4 Situation 4 : remplacer la colonne taux à la colonne taux_classe

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

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

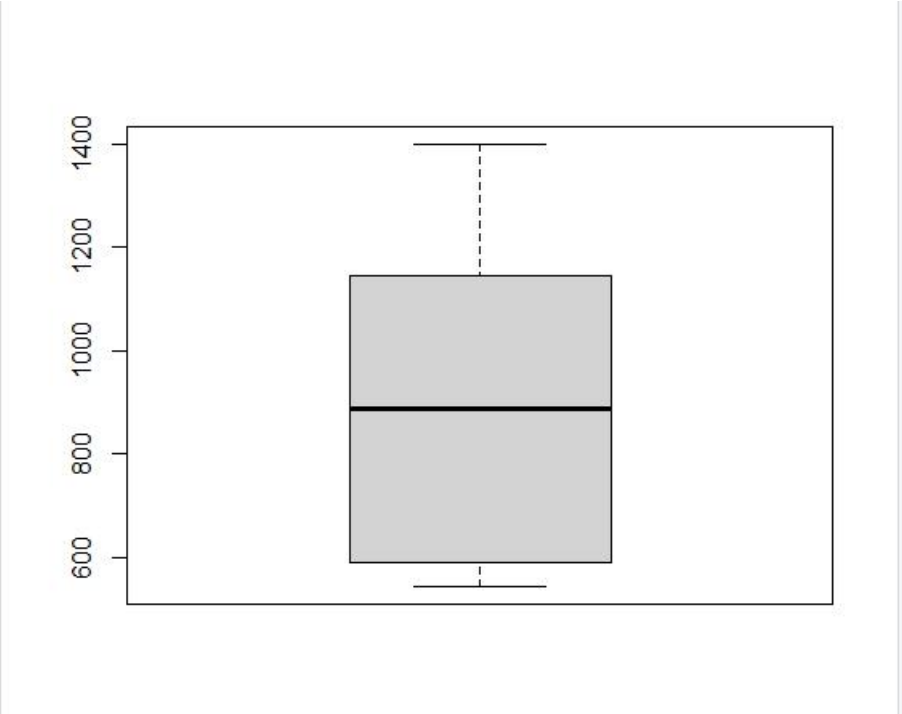

Figure 72 Le boxplot de taux

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

Figure 73 Récapitulatif du taux

Nous supprimons ensuite la colonne taux d'origine et renommons taux_classe en taux. Les données finales Clients_Immatriculations, sont présentées ci-dessous.

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

Figure 74 Récapitulatif du Clients_Immaticulations_traite

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

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

Figure 75 Situation4_Séparation de l'échantillonnage d'apprentissage et test

6.4.1 Algorithme : C50

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

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

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

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

Tree size: 50

Non-standard options: attempt to group attributes

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

Figure 76 Situation4_C50 résultat de prédiction

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

Tableau 10 Situation4_C50 : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 77 Situation4_C50 : calcul de l'AUC

6.4.2 Algorithme : Naviebayes

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

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

Figure 78 Situation4_Naivebayes : résultat de prediction

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

Tableau 11 Situation4_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 79 Situation4_ Naivebayes : calcul de l'AUC

6.4.3 Algorithme : Random Forest

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

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

Figure 80 Formation de jeux d'apprentissage avec random forest

6.4.4 Algorithme : NNET

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

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

Figure 81 Situation4_NNET résultat de prédiction

	berline	compacte	minicar	Routiere	sportive	**Rappel**
berline	4316	0	4	1074	1888	**0,592694315**
compacte	0	803	892	0	1	**0,473466981**
minicar	4	842	6705	1	1	**0,887726731**
Routiere	29	2	0	2237	222	**0,898393574**
sportive	386	0	1	539	3673	**0,798651881**
Précision	**0,91151**	**0,48755313**	**0,88200474**	**0,58088808**	**0,63491789**	
taux d'erreur	**0,2491956**					

Tableau 12 Situation4_NNET : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 82 Situation4_ NNET : calcul de l'AUC

6.5 Situation 5 : remplacer la colonne taux et la colonne âge à la colonne taux_classe et âge_classe

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

La classification du taux_classe reste inchangée par rapport à la situation4.

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

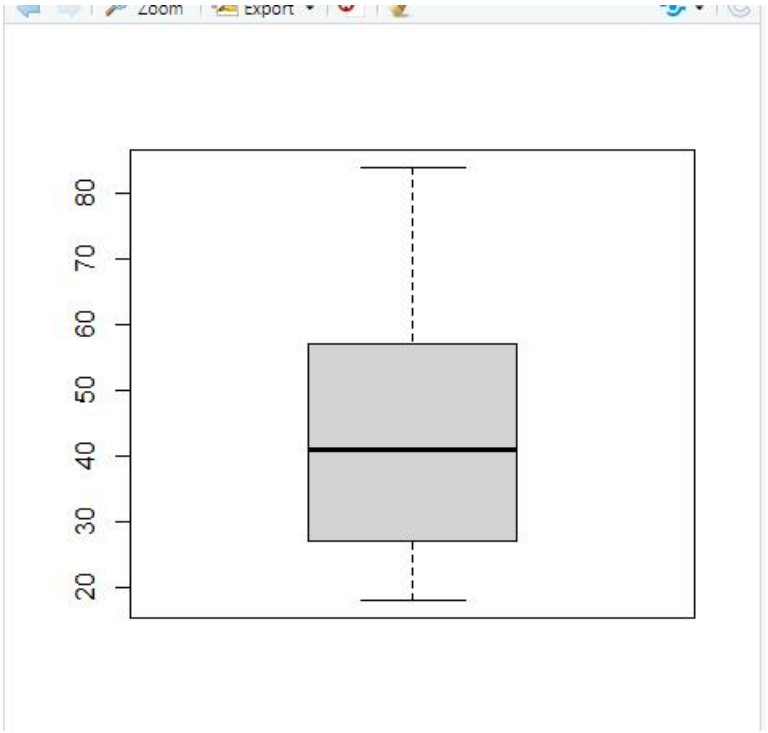

Figure 83 Le boxplot d'âge

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

Figure 84 Récapitulatif d'âge

Nous supprimons ensuite la colonne taux et âge d'origine, renommons taux_classe en taux et renommons age_classe en age. Les données finales Clients_Immatriculations, sont présentées ci-dessous.

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

Figure 85 Récapitulatif du Clients_Immatriculations_traite

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

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

Figure 86 Situation5_Séparation de l'échantillonnage d'apprentissage et test

6.5.1 Algorithme : C50

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

```
> treec <- C5.0(categories ~., Clients_Immatriculations_traite_EA)
c50 code called exit with value 1
> print(treec)

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

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

Tree size: 0

Non-standard options: attempt to group attributes

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

Figure 87 Situation5_C50 code

6.5.2 Algorithme : Naviebayes

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

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

Figure 88 Situation5_Naivebayes résultat de prédiction

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

Tableau 13 Situation5_Naivebayes : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 89 Situation5_ Naivebayes : calcul de l'AUC

6.5.3 Algorithme : Random Forest

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

```
> table(result.RF)
result.RF
 berline compacte  minicar Routiere sportive
    4704      507     8740     4176     5493
> table(Clients_Immatriculations_traite_ET3$categories, result.RF)
          result.RF
           berline compacte minicar Routiere sportive
  berline     4311        0       2     1177     1792
  compacte       0      250    1445        0        1
  minicar        4      257    7291        0        1
  Routiere       2        0       1     2280      207
  sportive     387        0       1      719     3492
```

Figure 90 Situation5_Random Forest résultat de prédiction

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

Tableau 14 Situation5_Random Forset : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 91 Situation5_ Random Forest: calcul de l'AUC

6.5.4 Algorithme : NNET

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

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

Figure 92 Situation5_NNET résultat de prédiction

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

Tableau 15 Situation5_NNET : Calcul du rappel, de la précision et des taux d'erreur

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

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

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

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

Figure 93 Situation5_ NNET : calcul de l'AUC

7. Conclusion des quatre algorithmes

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

Nous examinons les 5 classifications. La situation 1 ne supprime ni ne modifie les données ; la situation 2 supprime la colonne taux ; la situation 3 supprime les colonnes taux et âge ; la situation 4 reclasse le taux et la situation 5 reclasse le taux et l'âge. Nous combinons tous les cas pour produire un tableau, et les deux tableaux suivants montrent que l'algorithme NNET dans la situation 4 a la valeur AUC la plus élevée, c'est 0.9486. La différence entre le taux d'erreur de l'algorithme NNET et le taux d'erreur le plus bas dans ce cas est très faible. Par conséquent, nous choisissons l'algorithme NNET dans la situation 4 pour prédire les données de marketing.

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

Tableau 16 Comparaison de l'AUC

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

Tableau 17 Comparaison des taux d'erreur

8. Nettoyage de Marketing

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

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

Figure 94 Renommer des colonnes de Marketing

Ensuite, nous transformons les valeurs au format correct.

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

Figure 95 Convert to format demandé

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

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

```
> Marketing_traite$taux_classe <- ifelse(Marketing_traite$taux <= 588,"tauxbas",
+                                        ifelse(Marketing_traite$taux > 588& Marketing_traite$taux <=899,"tauxmoyen",
+                                               ifelse(Marketing_traite$taux > 899& Marketing_traite$taux <=1144,"tauxeleve",
+                                                      ifelse(Marketing_traite$taux > 1144,"tauxtreseleve ","Rien"))))
>
> Marketing_traite$taux_classe <- as.factor(Marketing_traite$taux_classe)
> Marketing_traite <- Marketing_traite[,-4]
> names(Marketing_traite)[7] <- "taux"
> Marketing_traite <- Marketing_traite[,-1]
> Marketing_traite$situationfamiliale <- gsub("C<U+FFFD>libataire", "Celibataire", Marketing_traite$situationfamiliale)
> Marketing_traite$age <- as.factor(Marketing_traite$age)
> Marketing_traite$sexe <- as.factor(Marketing_traite$sexe)
> Marketing_traite$nbenfantsacharge <- as.factor(Marketing_traite$nbenfantsacharge)
> Marketing_traite$taux <- as.factor(Marketing_traite$taux)
```

Figure 96 Création de class pour taux, corrige de valeurs, et convert to format demandé

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

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

Figure 97 Récapitulatif du tableau des Immatriculations nettoyés

9. Prédiction

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

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

Figure 98 Prédiction de Marketing avec NNET

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

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

Figure 99 Prédiction finale de Marketing