Détection de fraude dans le domaine de l'assurance

Arbres de décision et prédiction

Léa Darne
Hugo Munive

Janvier 2024

Introduction

Contexte et objectifs

La construction des arbres de prédiction représente un élément essentiel dans le but de prédire au mieux les événements possible de se produire. Le présent rapport s'inscrit dans le cadre du projet de fin d'année du cours d'Analyse de Données, visant à développer un modèle de prédiction du risque de déclarations frauduleuses d'accidents par les assurés. L'objectif principal est d'explorer les ensembles de données, composés de données financières, démographiques et relatives aux accidents, afin de concevoir des classifieurs performants.

Justification des outils utilisés

La justification des outils choisis repose sur la nécessité d'exploiter des méthodes robustes de classification et de clustering. Ces outils sont cruciaux pour atteindre l'objectif de minimiser les risques financiers en identifiant de manière précise les déclarations frauduleuses. L'utilisation de méthodes de rééquilibrage des classes, de tests de corrélation et de clustering permettra d'optimiser la prédiction du risque.

Aperçu du projet

L'ensemble du projet se décompose en plusieurs étapes clés. Dans le chapitre 1, nous explorerons les caractéristiques fondamentales des ensembles de données. Le chapitre 2 se concentrera sur la correction des déséquilibres dans les données. Le chapitre 3, dédié aux *Tests de corrélation*, examinera les relations entre les différentes variables. Par la suite, le chapitre 4, *Clustering*, se penchera sur le regroupement des données similaires. Enfin, dans le chapitre 5, *Méthode de rééquilibrage et clustering*, nous combinerons ces approches pour espérer de maximiser la performance du modèle de prédiction.

Chaque sous-partie suivante détaillera les choix, les méthodes et les résultats spécifiques à sa phase correspondante du projet.

Toutes les fonctions auxiliaires, ainsi que les complémentaires du code R seront mis en annexe.

Sommaire

Chapitre 1

Première approche et visualisation des données

Toutes les fonctions auxiliaires utilisées dans ce projet seront documentées dans le code en annexe.

1.1 Chargement et exploration des données

Dans cette première partie, nous abordons le chargement des données à partir du fichier CSV fourni, "Data_Projet1.csv". Nous utilisons des bibliothèques telles que ggplot2 pour la visualisation initiale et inspectons les premières lignes du jeu de données avec la fonction View(). De plus, nous explorons le résumé statistique et les noms des variables pour obtenir une compréhension initiale de la structure des données.

1.1.1 Chargement et préparation des données

Dans toute la suite, les données utilisées dans ce projet seront stockées dans une variable appelée fraud_data. Tout d'abord, à l'aide de la commande str(fraud_data), on peut observer qu'il s'agit d'un data frame contenant 1100 observations et 12 variables. Parmi ces 12 variables, qui seront appelées les variables explicatives, il y a la variable fraudulent, qui est notre variable cible à prédire.

```
fraud_data <- read.csv("Donnees/Data_Projet_1.csv",
            header = TRUE, sep = ",", dec = ".", stringsAsFactors = T)
```

En revanche, on est contraint d'enlever les variables du type id, vu que leur but est seulement d'organiser les variables pour une meilleure lecture des données, mais ne participent pas à la prédiction de la variable cible. Pour ce faire, on utilise la fonction subset

```
fraud_data <- subset(fraud_data, select=-c(customer_id, claim_id))
```

1.1.2 Première approche de visualisation des caractéristiques des données

Afin de cerner notre problème de prédiction de la classe cible *fraudulent*, une première idée c'est de regarder, dans un premier temps, la répartition de la classe cible, i.e. le nombre des cas réels *oui* et *non* dans la classe parmi les 1100 observations. Dans un second temps, nous utilisons des graphiques à barres pour examiner la distribution des déclarations frauduleuses en fonction de diverses variables explicatives. On obtient donc les graphiques suivants :

FIGURE 1.1 – Plot des toutes les variables

D'après les graphiques, on peut constater qu'effectuer une corrélation entre les variables explicatives et la variable cible n'est pas immédiat. Dans certains cas, on est incapable de savoir, ou même, de visualiser les données tels qu'ils sont représentés. Alors, étant donné que pour l'instant l'on est incapable d'effectuer des modifications supplémentaires aux données, on peut commencer la section d'apprentissage supervisé et la création des arbres de décision.

1.2 Arbres de décision

1.2.1 Pré-traitement des données

Afin de pouvoir créer des arbres de décision, on utilisera la démarche suivante : Créer un ensemble d'apprentissage (EA) et un ensemble de tests (ET) à partir des 1100 observations dans fraud_data. Par convention, on choisira une taille pour l'EA de deux tiers de la taille totale de l'échantillon, soit les observations 1 jusqu'à 734. Le reste sera utilisé comme l'ET, afin d'évaluer la performance des arbres de décision.

```
fraud_data_EA <- fraud_data[1:734,]
fraud_data_ET <- fraud_data[735:1100,]
```

1.2.2 Création des arbres

Nous utilisons différentes méthodes d'apprentissage d'arbres de décision (rpart, C5.0, tree, ctree) pour modéliser la relation entre les variables et la déclaration de fraude. Les arbres ainsi générés sont ensuite visualisés pour une meilleure compréhension.

```
tree1 <- rpart(fraudulent~., fraud_data_EA, parms = list(split = "gini"))
tree1.1 <- rpart(fraudulent~., fraud_data_EA, parms = list(split = "information"),
                 control = rpart.control(maxdepth = 9))
tree2 <- C5.0(fraudulent~., fraud_data_EA, param = list(split = "gini"))
tree2.1 <- C5.0(fraudulent~., fraud_data_EA, param = list(split = "information"))
tree3 <- tree(fraudulent~., fraud_data_EA)
```

Comme on peut l'observer, on utilise plusieurs méthodes de création d'arbres de décision. Bien que les 3 méthodes principales sont rpart, C5.0 et tree, on utilise différent paramètres pour leur création.

Tout d'abord, on test l'efficacité de changer le paramètre de création de l'arbre "gini" et "information". Il s'agit de deux méthodes pour interpréter l'information contenue dans le data frame et en déduire des méthodes de sélection et d'apprentissage afin de prédire la classe cible.

1.2.3 Visualisation des arbres

Dans toute la suite, on considérera la même méthode pour afficher les arbres de prédiction.

```
plot(tree2, type="simple", main="Arbre C5.0")
plot(tree2.1, type="simple", main="Arbre C5.0")
plot(tree3, main="Arbre tree", col="blue", fill="blue")
text(tree3, pretty = 0)
```

Ainsi, on obtient les arbres suivants :

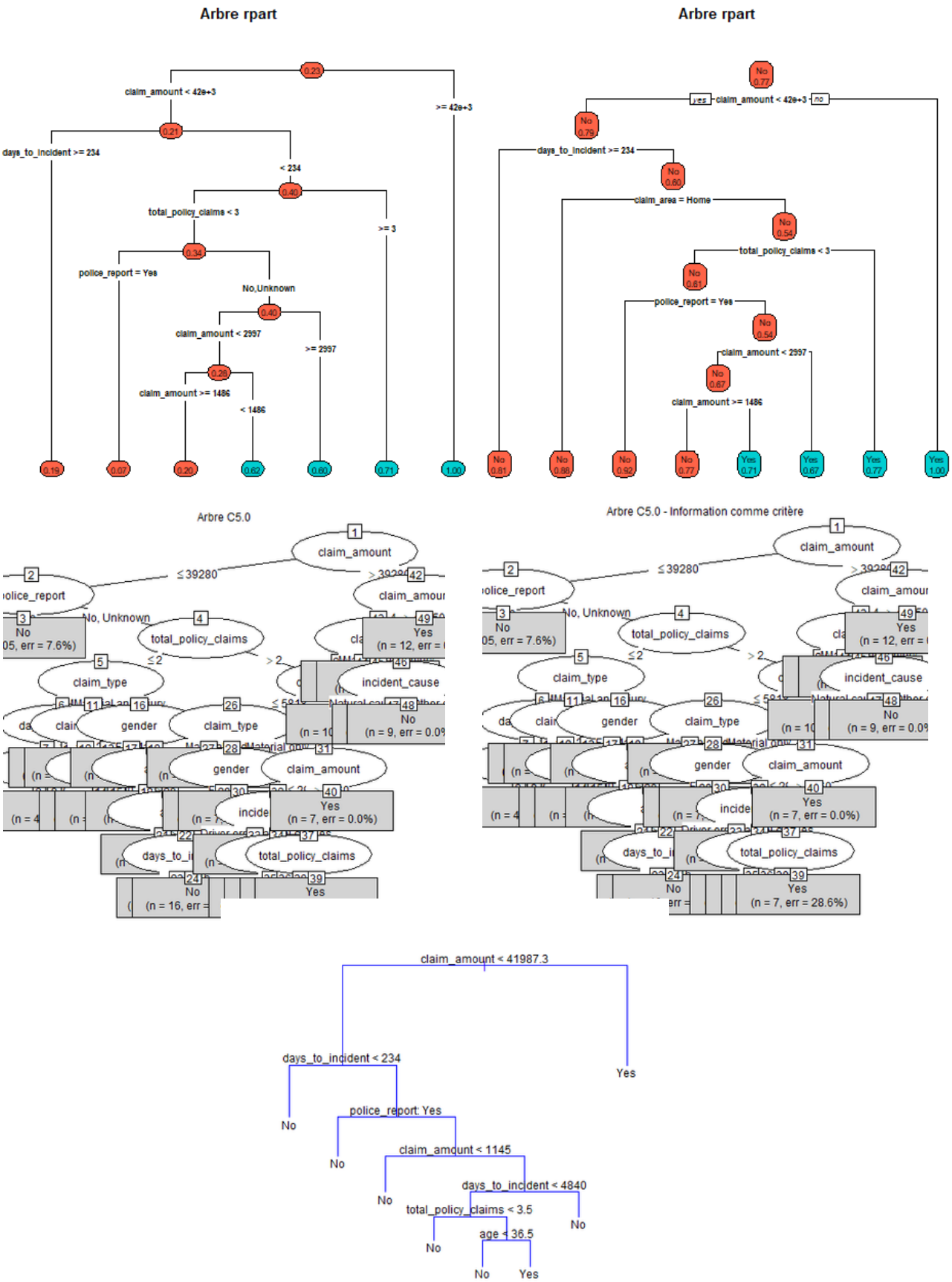

FIGURE 1.2 – Arbres de prédiction

1.3 Courbes ROC

Une fois que les arbres de prédiction sont créés, on peut les comparer grâce à leurs courbes ROC (DEF).

1.3.1 Fonction auxiliaire

Dans toute la suite, on affiche les courbes ROC à l'aide de cette fonction auxiliaire :

```
plot(ROC("rpart"), col = "green")
plot(ROC("rpart1"), col = "black", add = TRUE)
plot(ROC("C5.0"), col = "red", add = TRUE)
plot(ROC("C5.01"), col = "yellow", add = TRUE)
plot(ROC("tree"), col = "blue", add = TRUE)
legend(0.5, 0.5, legend=c("rpart","rpart(info)", "C5.0","C5.0(info)", "tree"),
col=c("green","black", "red","yellow","blue"), lty=1:3, cex=0.8)
title(main="Courbes ROC")
```

1.3.2 Graphique des courbes ROC

Ainsi, on obtient les courbes ROC suivantes :

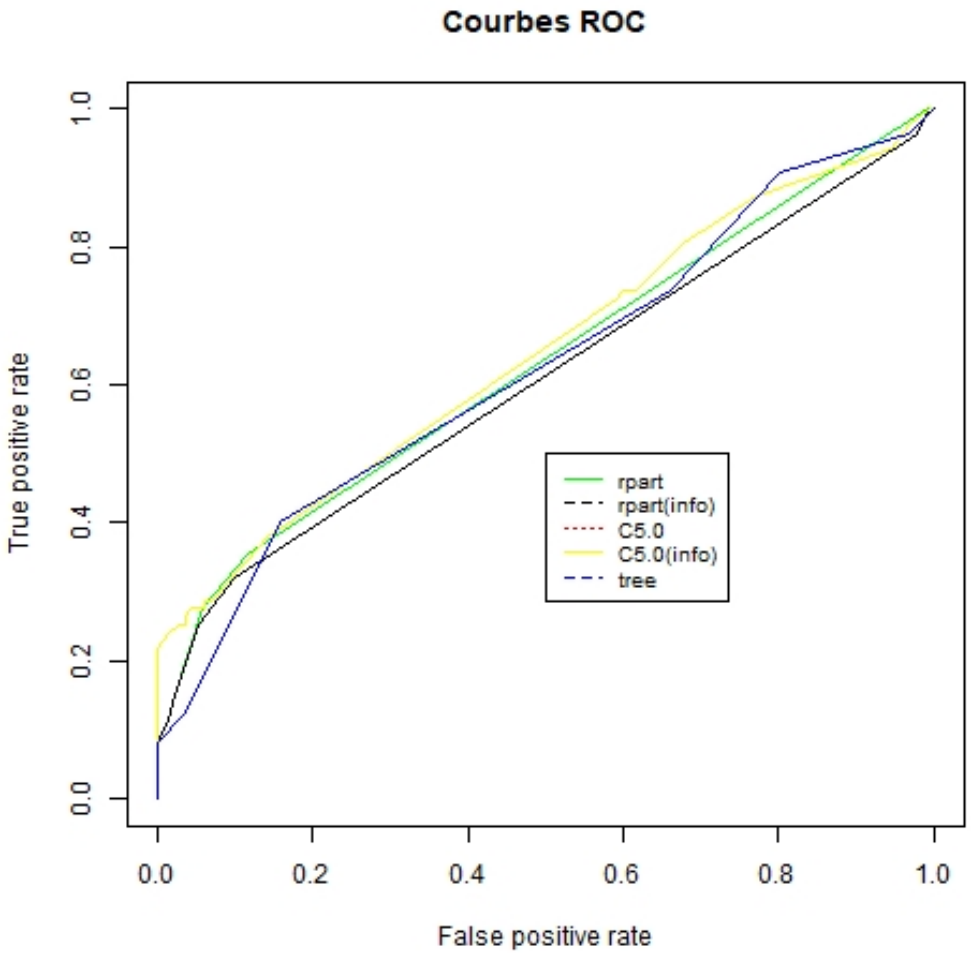

FIGURE 1.3 – Courbes ROC

On peut constater que toutes les courbes restent relativement inchangées. On s'attend donc à des arbres de prédiction peu performants et ce avec n'importe quel arbre choisi dans cette section. Pour déterminer la performance des arbres, on continue encore notre étude d'évaluation avec des tests d'air sous la courbe et des matrices de confusion.

1.4 Aire sous la courbe (AUC)

1.4.1 Fonction auxiliaire

Dans toute la suite du projet, on utilisera cette fonction qui nous aidera à calculer

```
AUC("rpart") #AUC : 0.630783174720883
AUC("rpart1") #AUC : 0.608659827792197
AUC("C5.0") #AUC : 0.646150043257941
AUC("C5.1") # AUC : 0.646150043257941
AUC("tree") #AUC : 0.625406830634862
```

1.4.2 Interprétation de l'AUC

On constate que toutes les AUC restent en moyenne de 0.631. Ceci peut s'interpréter comme des arbres de prédiction peu performants.

1.5 Mesure d'évaluation

1.5.1 Fonction auxiliaire et résultats

Dans toute la suite du projet, on utilisera cette fonction qui sert à afficher la matrice de transition associée à l'arbre de prédiction, ainsi que les différentes caractéristiques et propriétés de la matrice.

```
MC("rpart")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  263  63
       Yes  16  24

               Accuracy : 0.7842
                 95% CI : (0.7384, 0.8252)
    No Information Rate : 0.7623
    P-Value [Acc > NIR] : 0.1789

                  Kappa : 0.2684

 Mcnemar's Test P-Value : 2.274e-07

            Sensitivity : 0.27586
            Specificity : 0.94265
         Pos Pred Value : 0.60000
         Neg Pred Value : 0.80675
             Prevalence : 0.23770
         Detection Rate : 0.06557
   Detection Prevalence : 0.10929
      Balanced Accuracy : 0.60926

       'Positive' Class : Yes"
MC("C5.0")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  262  62
       Yes  17  25

               Accuracy : 0.7842
                 95% CI : (0.7384, 0.8252)
    No Information Rate : 0.7623
    P-Value [Acc > NIR] : 0.1789

```

```
                  Kappa : 0.2754

 Mcnemar's Test P-Value : 7.407e-07

            Sensitivity : 0.28736
            Specificity : 0.93907
         Pos Pred Value : 0.59524
         Neg Pred Value : 0.80864
             Prevalence : 0.23770
         Detection Rate : 0.06831
   Detection Prevalence : 0.11475
      Balanced Accuracy : 0.61321

       'Positive' Class : Yes"
MC("tree")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  269  76
       Yes  10  11

               Accuracy : 0.765
                 95% CI : (0.7182, 0.8075)
    No Information Rate : 0.7623
    P-Value [Acc > NIR] : 0.4798

                  Kappa : 0.1226

 Mcnemar's Test P-Value : 2.398e-12

            Sensitivity : 0.12644
            Specificity : 0.96416
         Pos Pred Value : 0.52381
         Neg Pred Value : 0.77971
             Prevalence : 0.23770
         Detection Rate : 0.03005
   Detection Prevalence : 0.05738
      Balanced Accuracy : 0.54530

       'Positive' Class : Yes"
MC("rpart1")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  264  65
       Yes  15  22

               Accuracy : 0.7814
                 95% CI : (0.7355, 0.8227)
    No Information Rate : 0.7623
    P-Value [Acc > NIR] : 0.2135

                  Kappa : 0.2482

 Mcnemar's Test P-Value : 4.293e-08

            Sensitivity : 0.25287
            Specificity : 0.94624
         Pos Pred Value : 0.59459
         Neg Pred Value : 0.80243
             Prevalence : 0.23770
         Detection Rate : 0.06011
   Detection Prevalence : 0.10109
      Balanced Accuracy : 0.59956
```

```
107
108         'Positive' Class : Yes"
109 MC("C5.1")
110 "Confusion Matrix and Statistics
111
112           Reference
113 Prediction  No Yes
114        No  262  62
115        Yes  17  25
116
117                Accuracy : 0.7842
118                  95% CI : (0.7384, 0.8252)
119     No Information Rate : 0.7623
120     P-Value [Acc > NIR] : 0.1789
121
122                   Kappa : 0.2754
123
124  Mcnemar's Test P-Value : 7.407e-07
125
126             Sensitivity : 0.28736
127             Specificity : 0.93907
128          Pos Pred Value : 0.59524
129          Neg Pred Value : 0.80864
130              Prevalence : 0.23770
131          Detection Rate : 0.06831
132    Detection Prevalence : 0.11475
133       Balanced Accuracy : 0.61321
134
135         'Positive' Class : Yes"
```

1.5.2 Interprétation

Dans nos résultats, on se situe dans un cadre de détection de fraude. Ainsi, on cherche à maximiser la sensibilité (pour détecter le plus de fraudes possible) et aussi la spécificité (pour éviter de faussement détecter de la fraude). Ainsi, on peut voir que l'arbre C5.0 est le meilleur car il a la meilleure sensibilité et spécificité. De plus, il a la meilleure AUC. On peut aussi voir que les courbes ROC des arbres C5.0 et C5.0(info) sont les plus proches de l'angle supérieur gauche, ce qui est un bon indicateur de la qualité de l'arbre.

De ce fait, on s'intéresse donc à des méthodes et stratégies pour améliorer les performances de nos arbres de prédiction. Dans cette première section, on a pu observer un déséquilibre des classes. En effet, on dispose de 1100 observations avec 846 cas non frauduleux et le reste qui l'est. Ce déséquilibre peut être à l'origine de la mauvaise performance. On va donc procéder avec une stratégie de rééquilibrage des classes.

Chapitre 2

Méthode de rééquilibrage des classes

La méthode utilisée dans cette section consiste à rééquilibrer la classe fraudulent par une méthode de sur-échantillonnage. En effet, on cherche à augmenter le nombre de la casse minoritaire en simulant des observations.

2.1 Chargement de données

On se place dans le même cadre que dans la première section.

2.2 Principe du rééquilibrage

Ainsi, on utilise la fonction ovun.sample afin d'utiliser la méthode de sur-échantillonage

```
fraud_data <- resample(fraud_data)
qplot(fraudulent, data=fraud_data, fill=fraudulent, geom="bar", main="Fraudulent",
    xlab="Fraudulent", ylab="Nombre de cas")
```

Une fois la librairie activée, on crée la fonction resample afin d'automatiser la méthode. De plus, avec la fonction ovun.sample, les observations obtenues sont ordonnées. Donc on utilise la fonction sample pour avoir nos données mélangées. Une fois nos données prêtes à être étudiées, on observe avec un plot[2.1] que, effectivement, nos données ont été rééquilibrées.

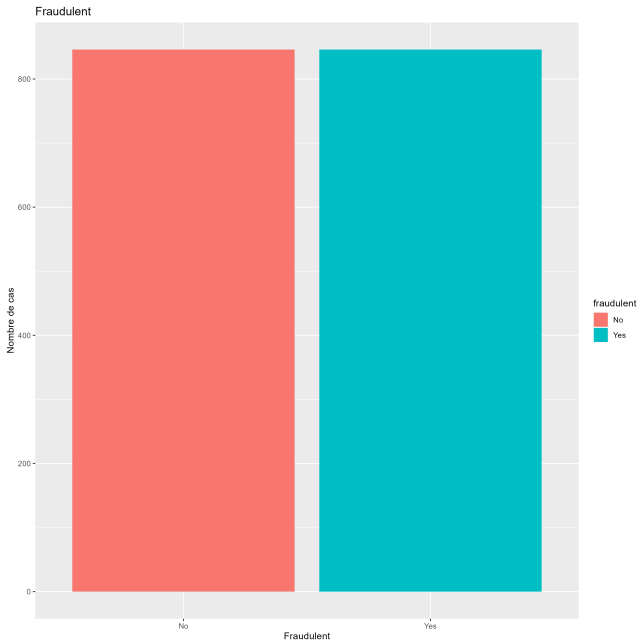

FIGURE 2.1 – Répartition de la classe rééquilibrée

Avec les paramètres correctes, on arrive à équilibrer parfaitement nos classes. Ainsi, on continue de la même manière que dans la section 1. On crée nos arbres de prédiction avec seulement rpart, C5.0 et tree. Bien évidemment, on enlève systématiquement les variables ID.

```
fraud_data <- subset(fraud_data, select=-c(customer_id, claim_id))
```

2.3 Application

Création de l'ensemble de tests et l'ensemble d'apprentissage, on remarque qu'on dispose maintenant de 1692 observations.

```
fraud_data_EA <- fraud_data[1:1128,]
fraud_data_ET <- fraud_data[1129:1692,]
```

2.4 Arbres prédicatifs et étude du modèle

On procède de la même façon que dans la section 1

2.4.1 Création des arbres et visualisation

```
tree1 <- rpart(fraudulent~., fraud_data_EA, parms = list(split = "gini"))
tree2 <- C5.0(fraudulent~., fraud_data_EA, param = list(split = "gini"))
tree3 <- tree(fraudulent~., fraud_data_EA)
```

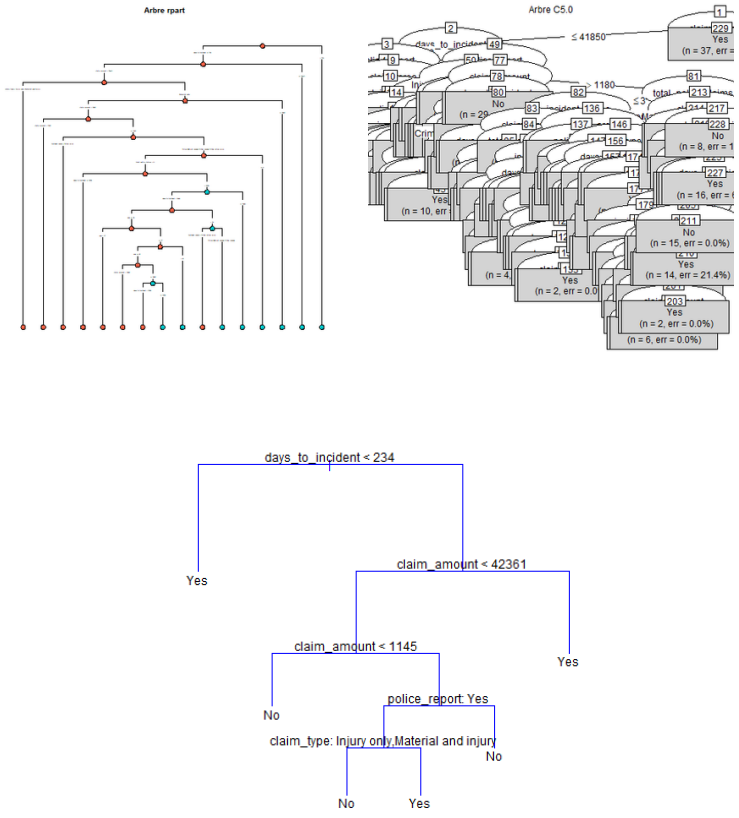

FIGURE 2.2 – Arbres de prédiction

On peut observer que la lecture des arbres est plus compliquée, étant donné que l'on dispose de plus d'observations.

2.4.2 Courbes ROC

On utilise la même fonction que dans la section 1, adapté pour seulement 3 arbres.

```
plot(ROC("rpart"), col = "green")
plot(ROC("C5.0"), col = "red", add = TRUE)
plot(ROC("tree"), col = "blue", add = TRUE)
```

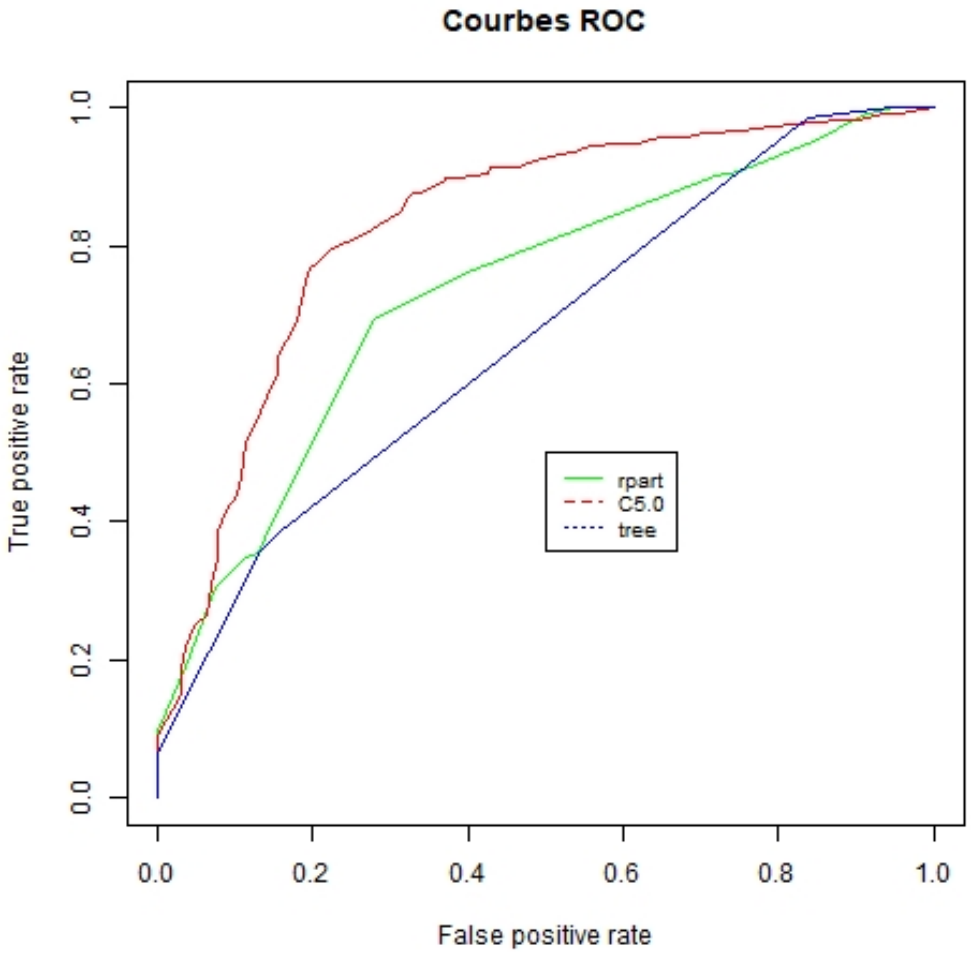

FIGURE 2.3 – Courbes ROC

On constate que l'arbre C5.0 est le plus performant parce que la courbe ROC est la plus proche du coin supérieur gauche, ce qui veut dire que le taux de vrais positifs est le plus élevé et le taux de faux positifs le plus faible. On peut aussi calculer l'aire sous la courbe des arbres pour confirmer cette hypothèse.

2.4.3 AUC

On utilise la même fonction que dans la section 1, modifiée pour seulement 3 arbres.

```
AUC("rpart") #AUC : 0.733895444756495
AUC("C5.0") #AUC : 0.829943453018148
AUC("tree") #AUC : 0.665100814830674
```

En effet, l'arbre C5.0 est le plus performant avec un AUC de 0.829943453018148. Si on le compare à l'AUC obtenu dans la section 1 (qui était de 0.646150043257941), on peut remarquer une amélioration, en pourcentage, de 28.4%.

2.4.4 Mesure d'évaluation

De même, cette fonction a été modifiée pour seulement 3 arbres

```
MC("rpart")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  195  90
       Yes  76 203

               Accuracy : 0.7057
                 95% CI : (0.6662, 0.743)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.4116

 Mcnemar's Test P-Value : 0.313

            Sensitivity : 0.6928
            Specificity : 0.7196
         Pos Pred Value : 0.7276
         Neg Pred Value : 0.6842
             Prevalence : 0.5195
         Detection Rate : 0.3599
   Detection Prevalence : 0.4947
      Balanced Accuracy : 0.7062

       'Positive' Class : Yes"
MC("C5.0")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  183  38
       Yes  88 255

               Accuracy : 0.7766
                 95% CI : (0.7399, 0.8103)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5494

 Mcnemar's Test P-Value : 1.27e-05

            Sensitivity : 0.8703
            Specificity : 0.6753
         Pos Pred Value : 0.7434
         Neg Pred Value : 0.8281
             Prevalence : 0.5195
         Detection Rate : 0.4521
   Detection Prevalence : 0.6082
      Balanced Accuracy : 0.7728

       'Positive' Class : Yes"
MC("tree")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  228 180
       Yes  43 113

               Accuracy : 0.6046
```

```
                  95% CI : (0.5629, 0.6452)
    No Information Rate : 0.5195
    P-Value [Acc > NIR] : 2.902e-05

                   Kappa : 0.2228

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

            Sensitivity : 0.3857
            Specificity : 0.8413
         Pos Pred Value : 0.7244
         Neg Pred Value : 0.5588
             Prevalence : 0.5195
         Detection Rate : 0.2004
   Detection Prevalence : 0.2766
      Balanced Accuracy : 0.6135

       'Positive' Class : Yes"
```

D'après les matrices de transition obtenues, on remarque que l'arbre C5.0 est le plus performant avec un taux de bonne classification de 77.66De plus, on remarque que l'arbre C5.0 est le plus performant en terme de sensibilité avec un taux de 87.03On peut critiquer que le nombre de faux négatifs est assez élevé (88) parmi les 338 cas positifs mais le nombre de faux positifs est assez faible (38) parmi les 271 cas négatifs. On peut donc conclure que l'arbre C5.0 est le plus performant pour la détection de fraude. Il nous reste étudier la performance des arbres en questions de variables explicatives. En effet, on remarque dans le plot des arbres que la lecture est assez compliquée, étant donné le nombre d'observations et de variables explicatives. Ainsi, on peut se demander si toutes les variables explicatives ont un rôle pertinent dans la détection de fraude.

Chapitre 3

Tests de corrélation

Dans cette section, on vise à déterminer la significativité des variables explicatives par rapport à la variable de classe. Pour ce faire, on utilisera des tests de corrélation, ainsi que les graphiques en mosaïques. Le résultat attendu sera en fonction du test effectué et de la p-value obtenue.

3.1 Test de significativité

Pour ces premiers tests, on charge les données dans une variable, on rééquilibre les données et on enlève les deux variables ID.

3.1.1 Test de Chi2 pour variables catégorielles

On utilise la fonction chisq.test qui nous permet de visualiser la p-value pour un test de corrélation de chi2. Si le test est réalisé pour niveau de 91%, on en conclue que si la p-value est inférieure à 0.05, alors cette variable est significative et donc corrélée avec la variable de classe. Ici, on a créer des fonctions pour afficher le résultat directement.

```
"pvchi2_table()
 police_report      claim_type incident_cause          gender      claim_area
  2.474865e-10   1.775015e-08   9.940190e-02   1.585209e-01   2.931374e-01
> pvfish_table()
 police_report      claim_type incident_cause          gender      claim_area
  1.441902e-10   1.358592e-08   9.969912e-02   1.585007e-01   2.931189e-01"
```

Ainsi, on peut en déduire que les variables qui ne sont pas corrélées sont gender, et claim area.

3.1.2 Test de Pearson, Spearman et Kendall pour variables numériques

De même, on fait un test de corrélation pour les variables catégorielles. On crée une fonction pour afficher le résultat directement. La seule différence c'est que pour pouvoir évaluer les variables numériques entre-elles, la variable de classe doit elle-même être numérique. On va donc changer les valeurs de la variable fraudulent et poser "Yes" = 1 et "Non" = 0. Afin de ne pas changer la variable où sont stockées nos données initiales, on crée une variable temporaire fraud data copie.

```
psk("age")
'[1] "p-value Pearson =  4.61337284270551e-07"
[1] "p-value Spearman =  2.48938561558996e-07"
[1] "p-value Kendall =  2.73961583160717e-07"'
psk("days_to_incident")
'"[1] "p-value Pearson =  1.63154239370855e-06"
[1] "p-value Spearman =  4.29831904665197e-11"
[1] "p-value Kendall =  5.59674069164966e-11"'
psk("claim_amount")
'[1] "p-value Pearson =  0.455960214966064"
[1] "p-value Spearman =  0.0703230077916243"
[1] "p-value Kendall =  0.0703407348935491"'
psk("total_policy_claims")
'[1] "p-value Pearson =  0.00980647494827477"
[1] "p-value Spearman =  0.0372912761703214"
[1] "p-value Kendall =  0.0373492251937505"'
```

Avec une p-value < 0.05, on rejette l'hypothèse nulle, il y a donc une corrélation entre la variable et la variable cible

3.1.3 Test en mosaïque

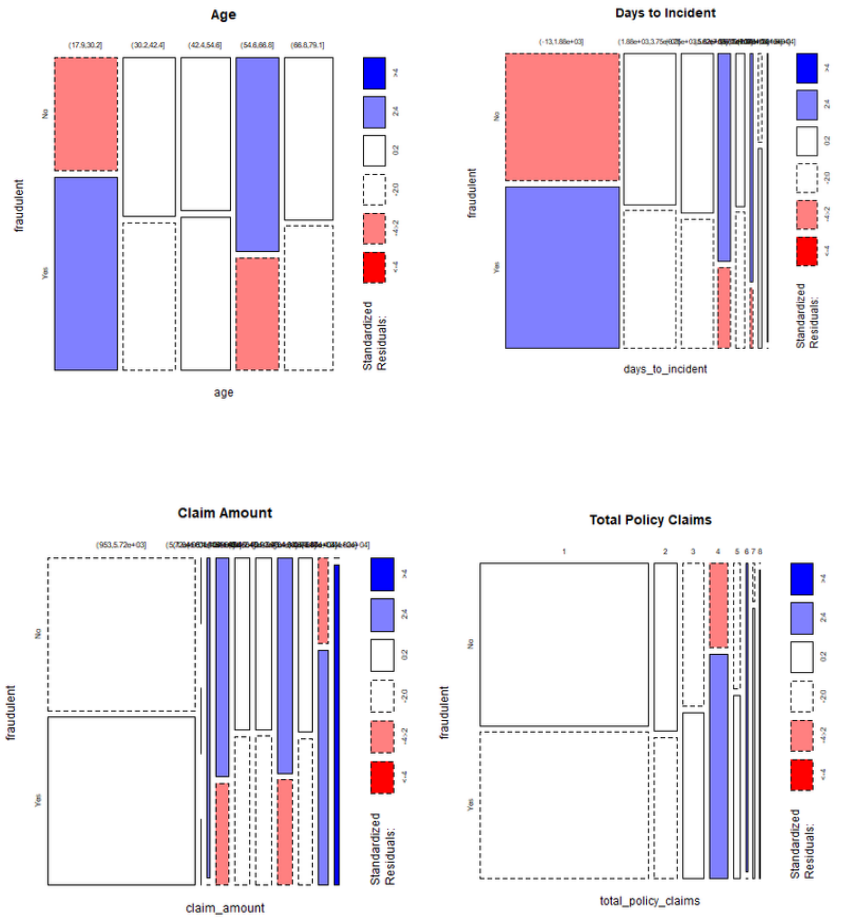

FIGURE 3.1 – Graphique en mosaïque

Ainsi, on a pu étudier la correlation entre les variables et la variable classe étudiée. On peut remarquer certaines incohérences au niveau de la p-value calculée. De ce fait, notre critère se basera dans le graphique en mosaïque. On remarquera que si on al'apparition des rectangles rouges ou bleu, on gardera alors cette variable car ceci veut dire que, en plus d'être significative, elle joue un rôle important dans la caractérisation de la variable de classe.

3.2 Fonction AttrEval

On a une fonction prédefinie qui nous indique déjà le niveau de significativité par rapport à l'ensemble global de l'information dans nos données. Avec la fonction AttrEval on peut déterminer ce niveau.

```
Eval("GainRatio")
"total_policy_claims       police_report    days_to_incident                 age
        0.104655533         0.019078534         0.009190180         0.008606188
         claim_type          claim_area      incident_cause              gender
        0.007791333         0.003102022         0.001044547         0.000363893
       claim_amount
       -1.000000000"
Eval("Gini")
"claim_amount       police_report                age    days_to_incident
       0.0303624371        0.0171140163        0.0136374392        0.0120551733
         claim_type total_policy_claims      incident_cause          claim_area
       0.0070925484        0.0055454957        0.0016318665        0.0009763576
             gender
       0.0002522038"
Eval("Accuracy")
"claim_amount                 age       police_report    days_to_incident
         0.07210402          0.06560284          0.05910165          0.05614657
         claim_type total_policy_claims      incident_cause          claim_area
         0.04314421          0.03546099          0.02127660          0.01300236
             gender
         0.01122931"
Eval("Relief")
"      age    days_to_incident      incident_cause       police_report
         0.12529551          0.10283688          0.10224586          0.09929078
total_policy_claims        claim_amount          claim_type              gender
         0.09574468          0.07978723          0.03250591          0.02836879
         claim_area
         0.01536643"
Eval("MDL")
"       claim_amount       police_report                age    days_to_incident
       0.0382797154        0.0211697834        0.0113500490        0.0067407198
total_policy_claims          claim_type          claim_area              gender
       0.0060765507        0.0059883394       -0.0005329839       -0.0020233076
     incident_cause
      -0.0060326374"
```

3.3 Application

3.3.1 Chargement et retraitement de données

Les données utilisées dans toute la suite seront rééquilibrées.

```
fraud_data <- read.csv("Donnees/Data_Projet_1.csv",
                    header = TRUE, sep = ",", dec = ".", stringsAsFactors = T)
resample <- function(fraud_data){
  resample <- ovun.sample(fraudulent ~ ., data = fraud_data, method = "over", N =
      1.539 * length(fraud_data$fraudulent))
  indices <- sample(nrow(resample$data))
  resample <- resample$data[indices,]
  return(resample)
}
fraud_data_sample <- resample(fraud_data)
"2/3 de 1692 pour fraud_data_EA et 1/3 pour fraud_data_ET"
fraud_data_EA <- fraud_data_sample[1:1128,]
fraud_data_ET <- fraud_data_sample[1129:1692,]
"On enl ve les variables non significatives"
fraud_data_EA <- subset(fraud_data_EA, select=-c(customer_id, claim_id, claim_area,
    gender, incident_cause))
```

3.3.2 Création des arbres et visualisation

```
tree1 <- rpart(fraudulent~., fraud_data_EA, parms = list(split = "gini"))
tree2 <- C5.0(fraudulent~., fraud_data_EA, param = list(split = "gini"))
tree3 <- tree(fraudulent~., fraud_data_EA)
rpart.plot(tree1, type=4, extra=7, box.col=c("tomato", "darkturquoise")[
    tree1$frame$yval], main="Arbre rpart")
plot(tree2, type="simple", main="Arbre C5.0")
plot(tree3, main="Arbre tree", col="blue")
text(tree3, pretty = 0)
```

Ainsi, on peut visualiser les arbres

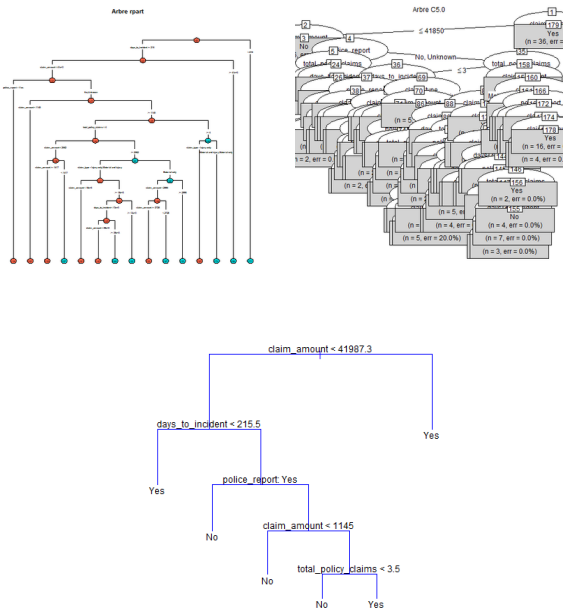

FIGURE 3.2 – Arbres de prédiction

3.3.3 Courbes ROC

On utilise la même fonction que dans la section 2.

```
plot(ROC("rpart"), col = "green")
plot(ROC("C5.0"), col = "red", add = TRUE)
plot(ROC("tree"), col = "blue", add = TRUE)
legend(0.5, 0.5, legend=c("rpart", "C5.0", "tree"), col=c("green", "red", "blue"),
    lty=1:3, cex=0.8)
title(main="Courbes ROC")
```

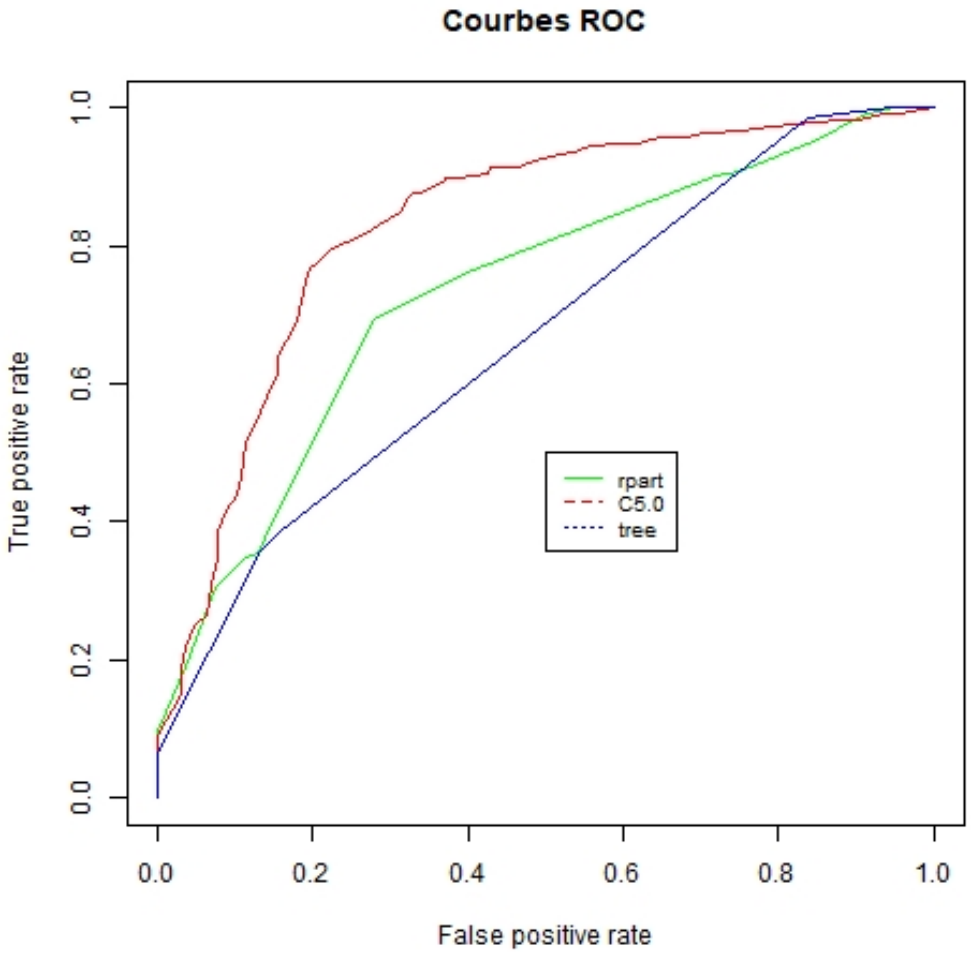

FIGURE 3.3 – Courbes ROC

3.3.4 AUC

On utilise la même fonction que dans la section 2.

```
AUC("rpart") #AUC : 0.746698943661972
AUC("C5.0") #AUC : 0.818228118712274
AUC("tree") #AUC : 0.679564889336016
```

3.3.5 Matrice de confusion

On utilise la même fonction que dans la section 2.

```
MC("rpart")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  209 104
       Yes  71 180

               Accuracy : 0.6897
                 95% CI : (0.6497, 0.7277)
    No Information Rate : 0.5035
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.3799

```

```
 Mcnemar's Test P-Value : 0.01556

            Sensitivity : 0.6338
            Specificity : 0.7464
         Pos Pred Value : 0.7171
         Neg Pred Value : 0.6677
             Prevalence : 0.5035
         Detection Rate : 0.3191
   Detection Prevalence : 0.4450
      Balanced Accuracy : 0.6901

       'Positive' Class : Yes"
MC("C5.0")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  223  95
       Yes  57 189

               Accuracy : 0.7305
                 95% CI : (0.6918, 0.7667)
    No Information Rate : 0.5035
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4615

 Mcnemar's Test P-Value : 0.00269

            Sensitivity : 0.6655
            Specificity : 0.7964
         Pos Pred Value : 0.7683
         Neg Pred Value : 0.7013
             Prevalence : 0.5035
         Detection Rate : 0.3351
   Detection Prevalence : 0.4362
      Balanced Accuracy : 0.7310

       'Positive' Class : Yes"
MC("tree")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  235 167
       Yes  45 117

               Accuracy : 0.6241
                 95% CI : (0.5827, 0.6642)
    No Information Rate : 0.5035
    P-Value [Acc > NIR] : 5.493e-09

                  Kappa : 0.2505

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

            Sensitivity : 0.4120
            Specificity : 0.8393
         Pos Pred Value : 0.7222
         Neg Pred Value : 0.5846
             Prevalence : 0.5035
         Detection Rate : 0.2074
   Detection Prevalence : 0.2872
      Balanced Accuracy : 0.6256

       'Positive' Class : Yes"
```

Ainsi, en enlevant certaines variables non significatives, on arrive avec l'arbre C5.0 à obtenir moins de résultats faux négatifs, améliorant le fait de n'arrive pas à prédire les cas qui sont réellement positifs. C'est donc le résultat cherché. On pourrait donc se demander, en plus de la significativité individuelle des variables explicatives, leur comportement en tant que variables par groupes. Et donc, on procédera par une étude de clustering des variables.

Chapitre 4

Clustering

4.1 Chargement et prétraitement de données

On charge les données et on modifie la variable total policy claims pour la transformer en factor.

4.2 Matrice de distance et clustering par partitionnement

```
dmatrix <- daisy(fraud_data)
km4 <- kmeans(dmatrix, 5)
table(km4$cluster, fraud_data$fraudulent)
"     No Yes
  1 391  38
  2  91   0
  3 135  52
  4 124  39
  5 105 125"
```

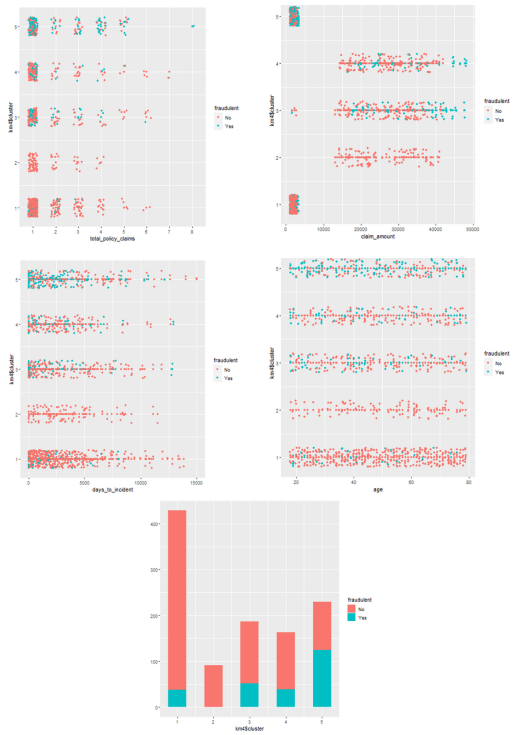

FIGURE 4.1 – Clustering

4.3 Application

L'idée dans cette section c'est de créer une variable de cluster. C'est-à-dire, en utilisant la méthode de k-means, avec un paramètre de 6, on peut créer 6 groupes avec lesquels on va associer la variable de classe. de Cette façon, on va pouvoir ajouter une variable qui est encore plus représentative que les autres variables explicatives. De plus, à l'aide de cette nouvelle variable, les arbres de prédiction gagneront en performance dans tous les aspects.

4.3.1 Arbres

```
tree1 <- rpart(fraudulent~., fraud_data_EA)
tree2 <- C5.0(fraudulent~., fraud_data_EA)
tree3 <- tree(fraudulent~., fraud_data_EA)
```

4.3.2 ROC

On utilise la même fonction que dans la section 2.

```
plot(ROC("rpart"), col = "green")
plot(ROC("C5.0"), col = "red", add = TRUE)
plot(ROC("tree"), col = "blue", add = TRUE)
```

4.3.3 AUC

On utilise la même fonction que dans la section 2.

```
AUC("rpart")
AUC("C5.0")
AUC("tree")
```

4.3.4 Matrice de confusion

```
MC("rpart")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  244  57
       Yes  43 220

               Accuracy : 0.8227
                 95% CI : (0.7886, 0.8533)
    No Information Rate : 0.5089
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.645

 Mcnemar's Test P-Value : 0.1936

            Sensitivity : 0.7942
            Specificity : 0.8502
         Pos Pred Value : 0.8365
         Neg Pred Value : 0.8106
             Prevalence : 0.4911
         Detection Rate : 0.3901
   Detection Prevalence : 0.4663
      Balanced Accuracy : 0.8222

       'Positive' Class : Yes"
MC("C5.0")
"Confusion Matrix and Statistics

```

```
          Reference
Prediction  No Yes
       No  257  28
       Yes  30 249

               Accuracy : 0.8972
                 95% CI : (0.8691, 0.921)
    No Information Rate : 0.5089
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7943

 Mcnemar's Test P-Value : 0.8955

            Sensitivity : 0.8989
            Specificity : 0.8955
         Pos Pred Value : 0.8925
         Neg Pred Value : 0.9018
             Prevalence : 0.4911
         Detection Rate : 0.4415
   Detection Prevalence : 0.4947
      Balanced Accuracy : 0.8972

       'Positive' Class : Yes"
MC("tree")
"Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  242  55
       Yes  45 222

               Accuracy : 0.8227
                 95% CI : (0.7886, 0.8533)
    No Information Rate : 0.5089
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6451

 Mcnemar's Test P-Value : 0.3681

            Sensitivity : 0.8014
            Specificity : 0.8432
         Pos Pred Value : 0.8315
         Neg Pred Value : 0.8148
             Prevalence : 0.4911
         Detection Rate : 0.3936
   Detection Prevalence : 0.4734
      Balanced Accuracy : 0.8223

       'Positive' Class : Yes"
```

Dans tout notre projet, on a cherché à minimiser le taux de faux négatifs, aussi appelé sensibilité (TP/(TP+FN)). Ainsi, on applique ce modèle aux données où on doit prédire la classe fraudulent. La méthode de clustering semble être la plus efficace pour donner à l'arbre C5.0 la capacité de prédire la classe.

Chapitre 5

Méthode de variables significatives et clustering

5.1 Test expérimental

Dans cette dernière partie, on utilise en même temps la méthode de clustering et la méthode de variables significatives. On charge les données, on les rééquilibre, on enlève les variables non significatives et on en extrait des clusters. Une fois les clusters formés, on ajoute cette nouvelle variable comme dans le chapitre 3 et on en déduit les arbres de prédition.

```
dmatrix <- daisy(fraud_data)
kmi <- kmeans(dmatrix, 6)
fraud_data$cluster <- kmi$cluster
tree1 <- rpart(fraudulent~., fraud_data_EA)
tree2 <- C5.0(fraudulent~., fraud_data_EA)
tree3 <- tree(fraudulent~., fraud_data_EA)
# ROC
plot(ROC("rpart"), col = "green")
plot(ROC("C5.0"), col = "red", add = TRUE)
plot(ROC("tree"), col = "blue", add = TRUE)
AUC("rpart")
"Aire de l'arbre rpart est = 0.941854907682246"
AUC("C5.0")
"Aire de l'arbre C5.0 est = 0.965714141973135"
AUC("tree")
"Aire de l'arbre tree est = 0.908732454595764"
```

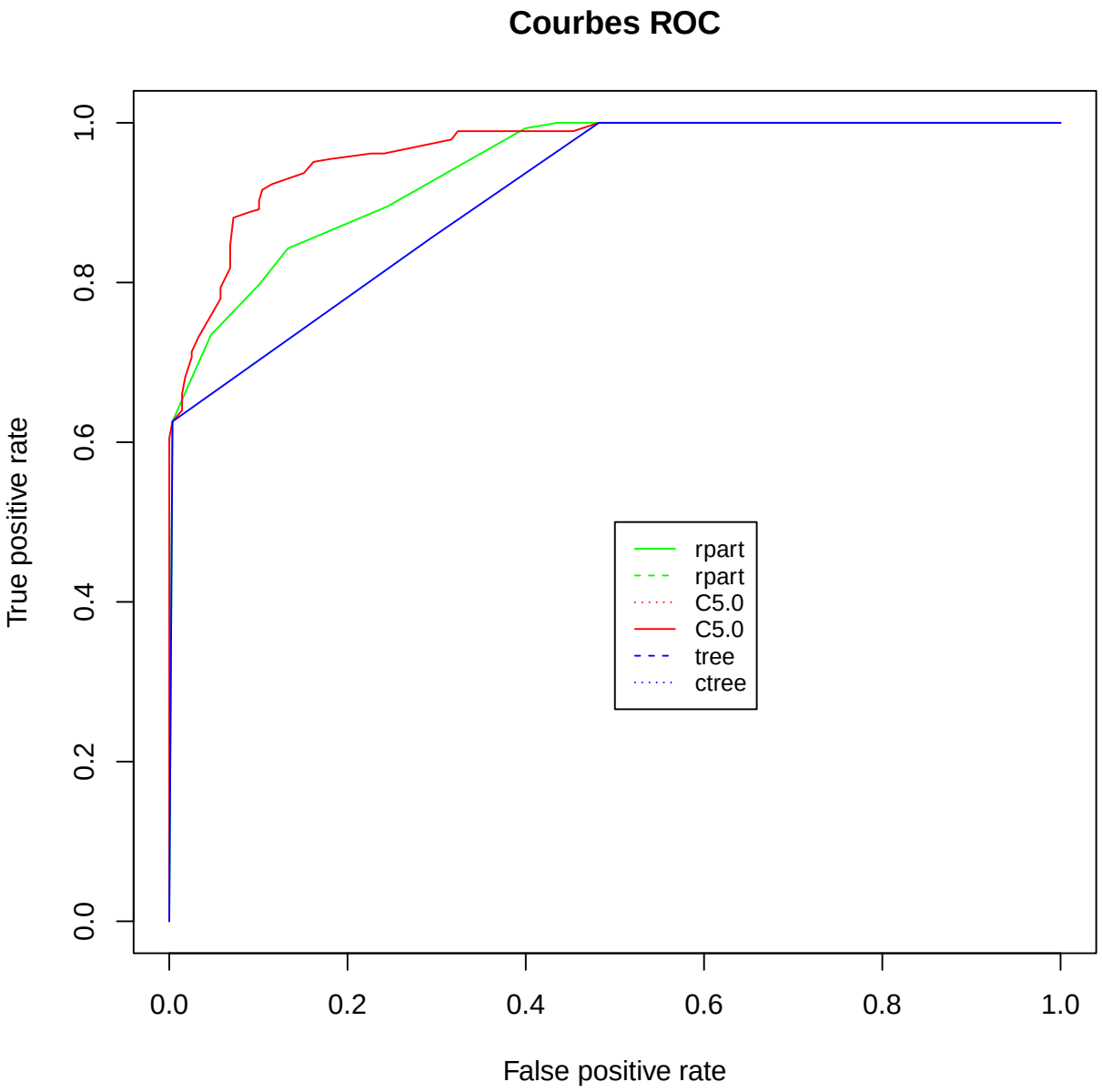

FIGURE 5.1 – Courbes ROC

```
"RPART
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  241  45
       Yes  37 241

               Accuracy : 0.8546
                 95% CI : (0.8228, 0.8827)
    No Information Rate : 0.5071
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7093

 Mcnemar's Test P-Value : 0.4395

            Sensitivity : 0.8427
            Specificity : 0.8669
```

```
         Pos Pred Value : 0.8669
         Neg Pred Value : 0.8427
             Prevalence : 0.5071
         Detection Rate : 0.4273
   Detection Prevalence : 0.4929
      Balanced Accuracy : 0.8548

       'Positive' Class : Yes

C5.0
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  246  22
       Yes  32 264

               Accuracy : 0.9043
                 95% CI : (0.8769, 0.9273)
    No Information Rate : 0.5071
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8084

 Mcnemar's Test P-Value : 0.2207

            Sensitivity : 0.9231
            Specificity : 0.8849
         Pos Pred Value : 0.8919
         Neg Pred Value : 0.9179
             Prevalence : 0.5071
         Detection Rate : 0.4681
   Detection Prevalence : 0.5248
      Balanced Accuracy : 0.9040

       'Positive' Class : Yes

TREE
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  277 107
       Yes   1 179

               Accuracy : 0.8085
                 95% CI : (0.7736, 0.8402)
    No Information Rate : 0.5071
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.619

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

            Sensitivity : 0.6259
            Specificity : 0.9964
         Pos Pred Value : 0.9944
         Neg Pred Value : 0.7214
             Prevalence : 0.5071
         Detection Rate : 0.3174
   Detection Prevalence : 0.3191
      Balanced Accuracy : 0.8111

       'Positive' Class : Yes"
```

5.2 Interprétation

Dans un premier, on peut observer des changements au niveau des matrices de confusion. En effet, bien que le côté aléatoire de la fonction ovun.sample joue un rôle, on remarque que le nombre de faux négatifs est presque nul. Par contre, le nombre de faux positifs a augmenté, et la précision a diminuée. On remarque aussi pour l'arbre C5.0, qui était supposé être le plus performant, une légère augmentation dans le nombre des faux positifs et négatifs. Ainsi, on peut émettre l'hypothèse de que, même si on a fait un test de corrélation entre les variables explicatives et la variable de classe, on n'a pas étudié l'impact que les couples ou les groupes qui forment ces variables ont envers la variable de classe. On préférera donc garder l'arbre de prédiction avec la méthode de clustering sans enlever les variables non significatives.

Chapitre 6

Conclusion

En conclusion, on peut remarquer que la construction des arbres de prédiction a été faite de façon intuitive. Peu à peu on se rapprochait d'un modèle qui augmentait les performances avec des nouvelles méthodes telles que le rééquilibrages de classes, la suppression des variables non significatives et la construction des clusters. Pour pouvoir choisir un bon modèle, on a dû se concentrer sur les méthodes de classification telle que la matrice de confusion. On regarde le taux de succès et en particulier, le taux de faux négatifs. Dans le cadre de la détection de fraude, on a comme priorité la détection des possibles cas frauduleux et minimiser ainsi les coûts éventuels d'une fraude.

Dans le fichier annexe Data result.csv, on a utilisé notre arbre de prédiction décrit dans la section 4, l'arbre C5.0 afin d'utiliser les données disponibles dans le fichier Data projet 1 New et prédire la classe.

Cependant il faut remarquer que l'utilisation de l'arbre de prédiction C5.0 construite par notre méthode reste aléatoire à chaque génération de rééquilibrage de la classe. Ainsi, si on veut améliorer encore la prédiction, il est donc recommandable d'essayer en boucle jusqu'à trouver un arbre de prédiction, qui semble être toujours C5.0, pour lequel nos attentes de réduction de faux négatifs et une bonne précision sont accomplies.