Projet DATA MINING

TERRIER Jules
BOGLIETTO Lucas

UNIVERSITÉ
CÔTE D'AZUR

Université Cote d'Azur
Année 2019/2020

Objectif du projet :

L'objectif est la création d'un modèle de prédiction du risque de défaut de paiement pour les clients et son application aux instances à prédire. On souhaite donc utiliser les techniques de classification afin de générer un modèle de prédiction de la classe des clients :

- defaut = Oui (positif)

- defaut = Non (négatif)

Plusieurs classifieurs seront générés et testés en appliquant les différentes méthodes de classification et en ajustant les paramètres afin d'optimiser les résultats. Seul le classifieur le plus performant sera conservé sachant que l'on souhaite avant tout minimiser les risques financiers en évitant d'accorder un emprunt à tort.

Le classifieur sélectionné sera ensuite appliqué à l'ensemble de données à prédire afin de prédire pour chaque client s'il est susceptible d'avoir un défaut de paiement (classe defaut = Oui) ou non (classe defaut = Non). Afin d'évaluer les classifieurs générés, on définira un ou des critère(s) (basés sur les taux de succès/échecs, la matrice de confusion ou les mesures d'évaluation par exemple) en fonction des objectifs de l'application décrits ci-dessus. On comparera les résultats des classifieurs générés selon ces critères afin d'identifier le plus pertinent.

Exploration et visualisation des données :

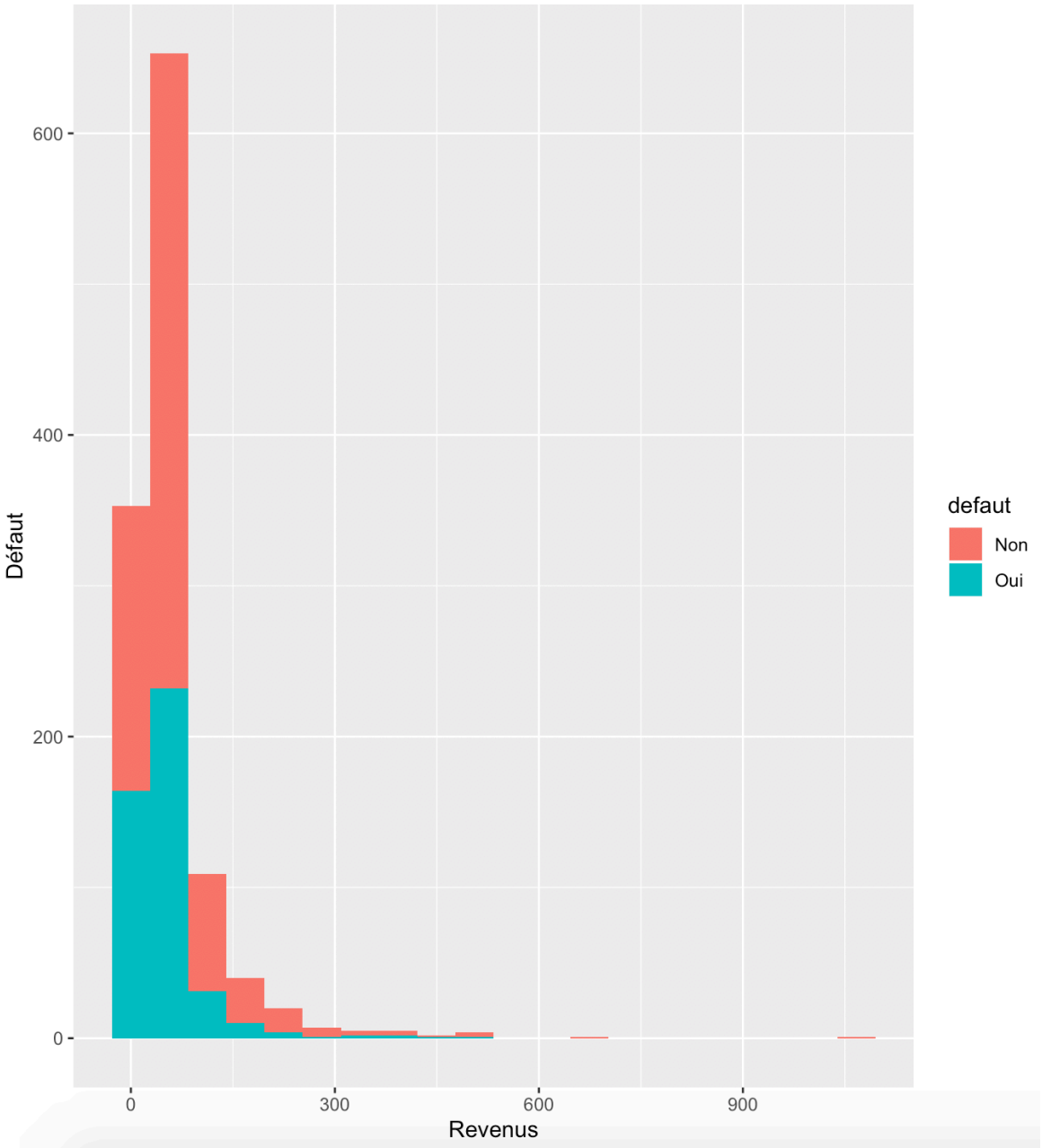
Pour commencer notre analyse, nous avons importer les données du projet dans une variable data_projet sur le logiciel R studio. Ces données se composent de 1200 observations pour 12 variables. Ci-dessous nous avons un aperçu de ces données pour les 18 premières explorations.

	branch	ncust	customer	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
1	3	3017	10012	28	Bac+2	7	2	44	17.7	2.990592	4.797408	Non
2	3	3017	10017	64	Bac+5 et plus	34	17	116	14.7	5.047392	12.004608	Non
3	3	3017	10030	40	Niveau bac	20	12	61	4.8	1.042368	1.885632	Non
4	3	3017	10039	30	Niveau bac	11	3	27	34.5	1.751220	7.563780	Non
5	3	3017	10071	35	Niveau bac	2	9	38	10.9	1.462126	2.679874	Oui
6	3	3017	10096	26	Bac+3	2	4	38	11.9	0.954142	3.567858	Oui
7	3	3017	10128	25	Niveau bac	4	2	30	14.4	1.045440	3.274560	Non
8	3	3017	10129	65	Bac+4	29	14	189	5.0	3.364200	6.085800	Non
9	3	3017	10140	21	Bac+3	0	0	23	3.9	0.305877	0.591123	Non
10	3	3017	10140	142	62	Niveau bac	33	13	96	17.5	6.686400	10.113600	Non
11	3	3017	10169	30	Bac+4	4	3	39	10.6	2.389452	1.744548	Oui
12	3	3017	10197	38	Bac+5 et plus	7	7	124	19.1	9.520968	14.163032	Non
13	3	3017	10200	18	Bac+2	0	0	35	3.9	0.174720	1.190280	Non
14	3	3017	10218	53	Niveau bac	9	13	41	13.3	2.333884	3.119116	Non
15	3	3017	10234	18	Bac+2	0	0	15	7.4	0.828060	0.281940	Oui
16	3	3017	10351	18	Bac+2	0	0	56	15.7	1.837528	6.954472	Oui
17	3	3017	10423	19	Niveau bac	2	0	31	0.3	0.024552	0.068448	Oui
18	3	3017	10428	19	Bac+2	0	0	26	5.8	0.586612	0.921388	Oui

Recherche de relations entre les variables :

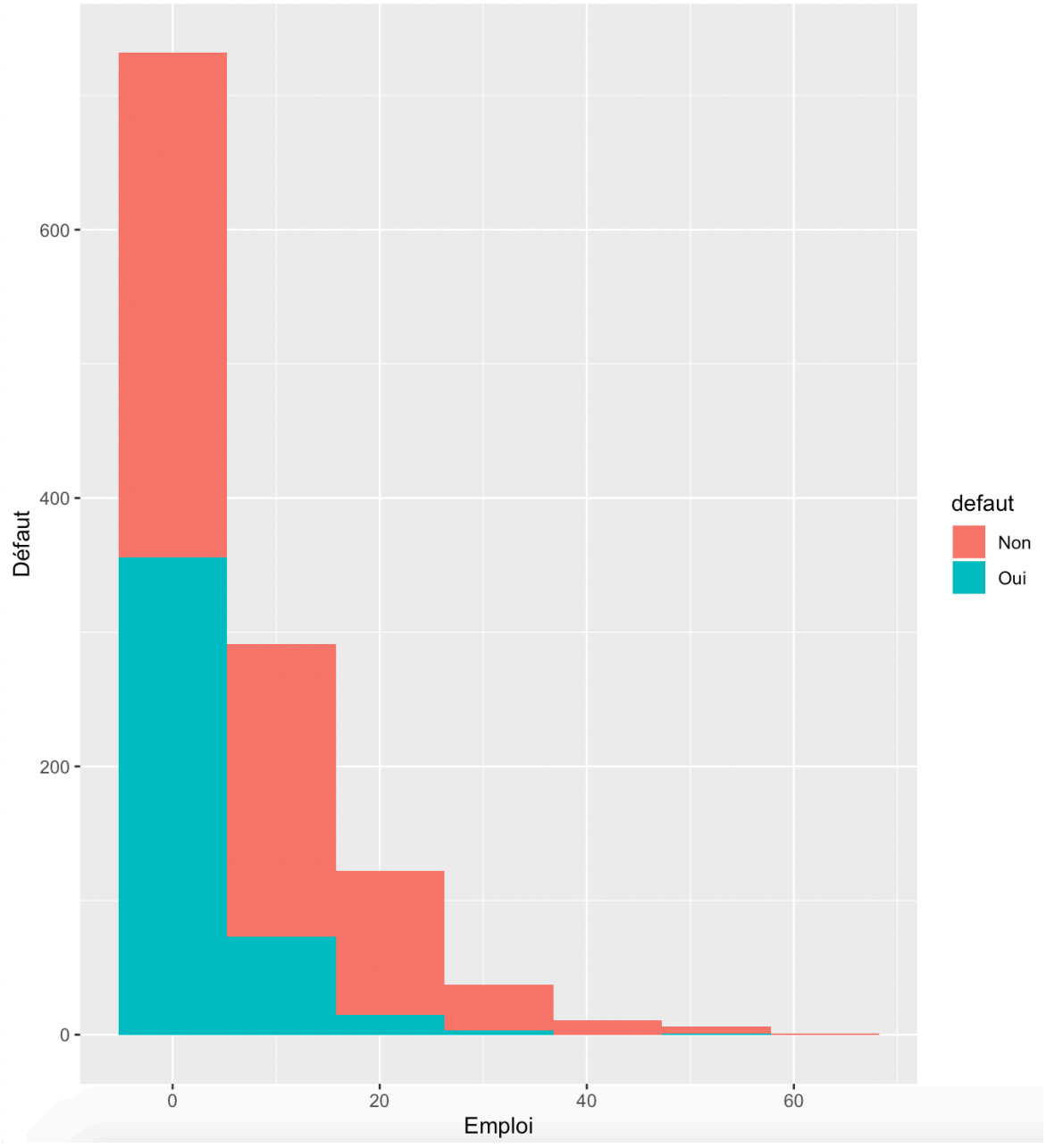
- A l'aide d'histogramme :

Nous nous somme tout d'abord intéressé à la relation entre les revenus des clients et la présence d'un défaut bancaire ou non.
Nous avons obtenus l'histogramme suivant :



D’après cet histogramme, on peut supposer que les revenus et le défaut bancaire sont liés. En effet, on peut remarquer que plus le revenus est élevé plus la part de défaut bancaire est faible. On peut donc penser que ce seront les client avec de faibles revenus qui seront les plus susceptible d’être en défaut bancaire.

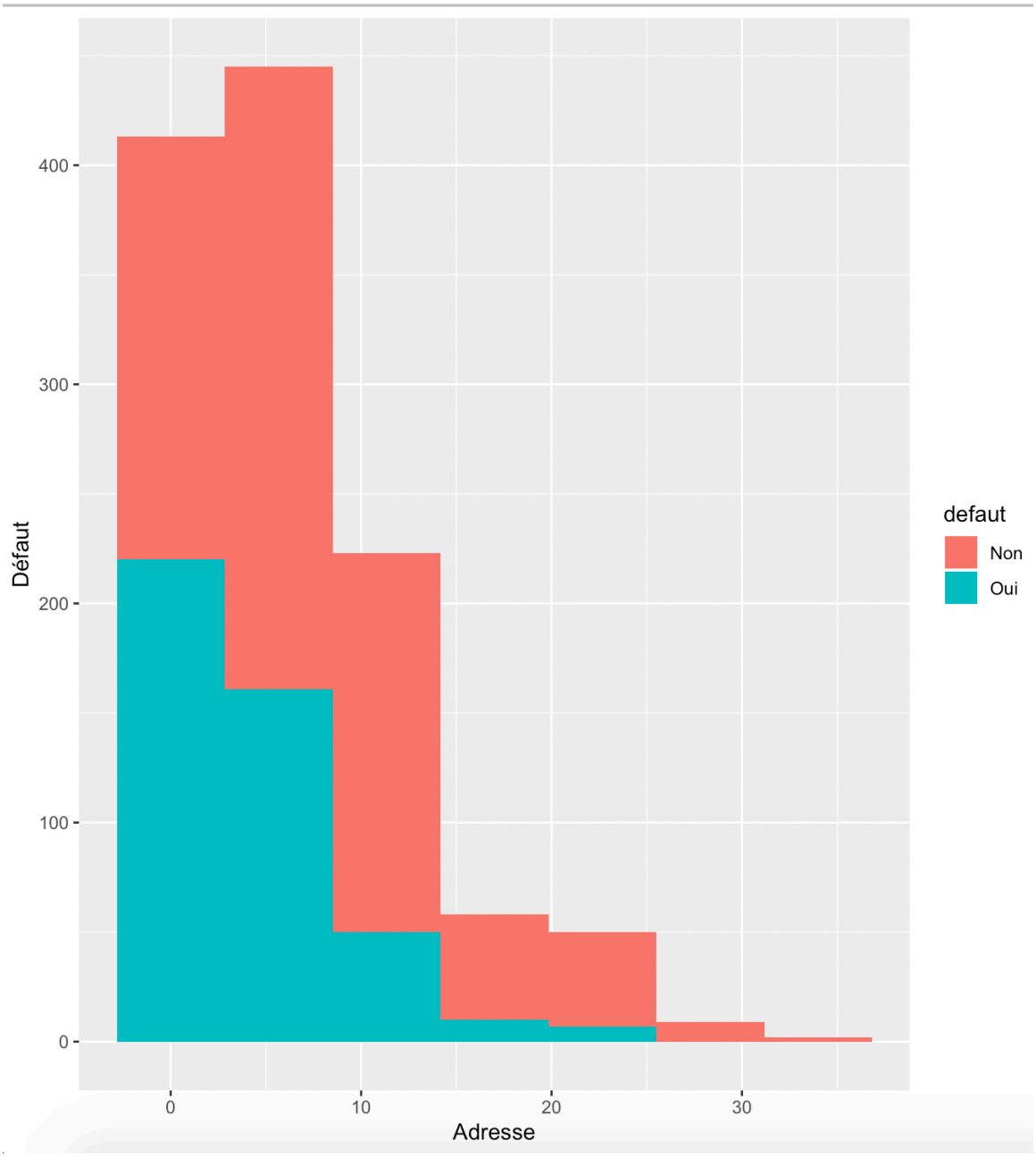
Comme les revenus sont liés à l'emploi, nous nous sommes ensuite intéressé à la relation entre l'emploi et le défaut. Nous avons ainsi obtenus l'histogramme suivant :



La variable emploi est défini comme le nombre d'années en fonction avec l'employeur actuelle. On peut donc considérer que plus la valeur de la variable emploi est élevée, plus cela démontre que le client a un emploi stable, et donc un salaire régulier, qui généralement a tendance à croitre avec le temps.

C’est pour cela que nous pouvons constater que plus la valeur de la variable emploi est élevée, plus le risque de défaut sera faible.

Enfin, nous nous sommes intéressés à la relation entre les variables défaut et adresse. La variable adresse est défini quant à elle, comme le nombre d’années de résidence du client au même domicile. Nous avons obtenu l’histogramme suivant :

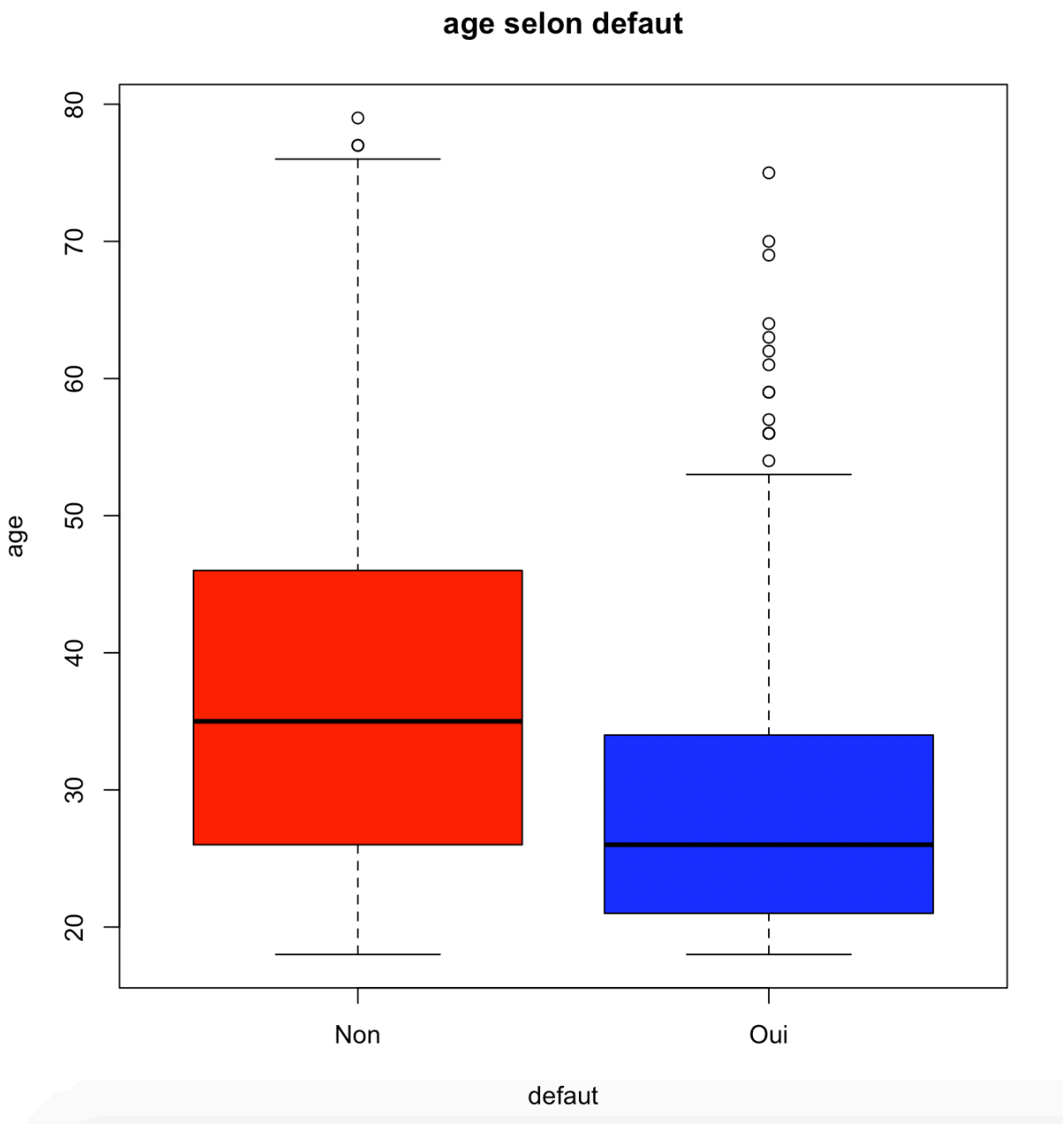


On remarque donc que plus le client habite depuis longtemps dans son logement, moins il est susceptible de faire un défaut de paiement. On peut donc penser que la stabilité du domicile apporte une sécurité bancaire.

Nous avons également réalisé d'autres histogrammes, pour les relations entre la variable défaut d'un coté et successivement les variables âge, et le niveau d'étude de l'autre coté. Cependant, nous n'avons pas pu dégager de réelle relations entre ces variables.

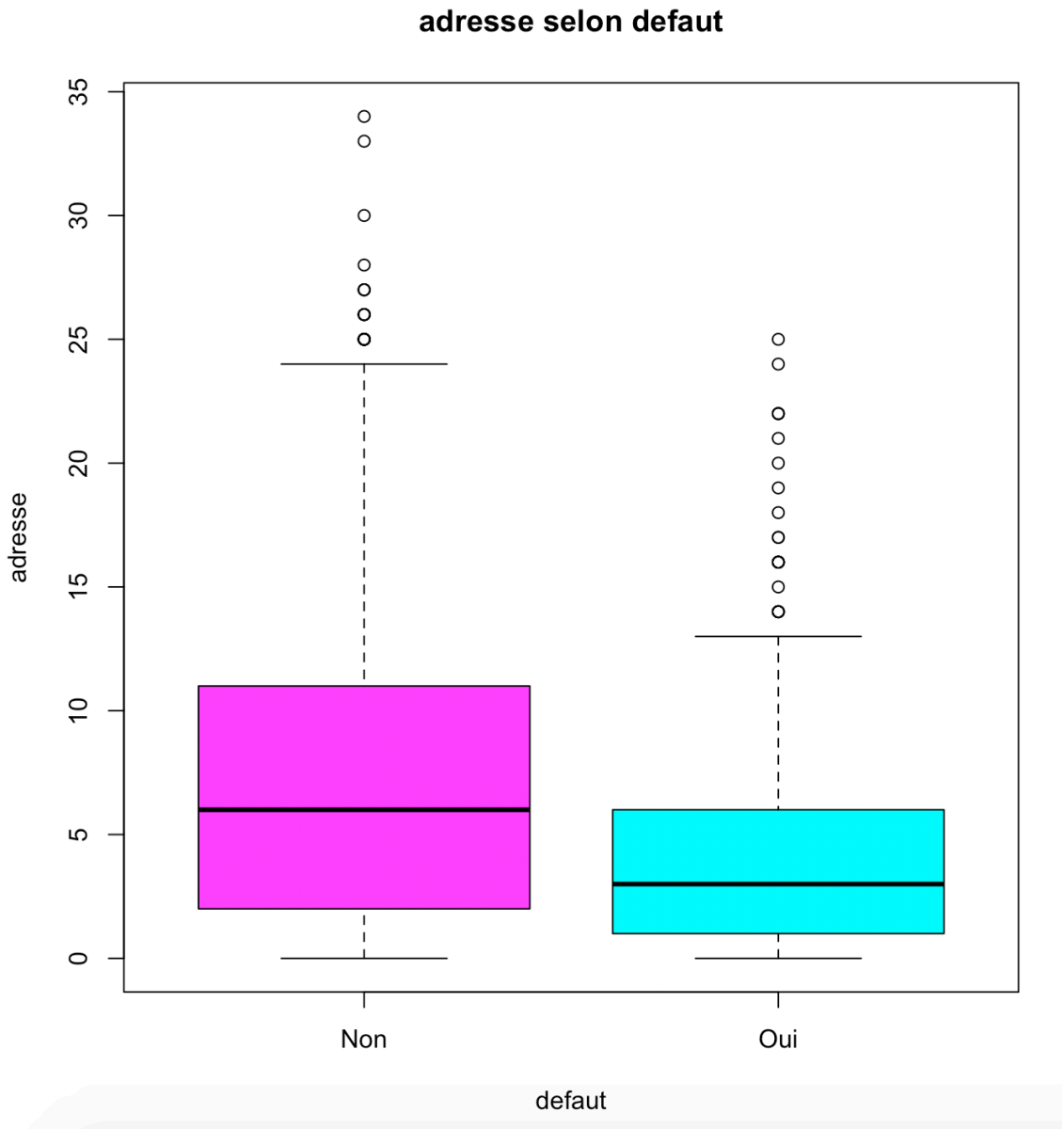
- A l'aide de boite à moustache :

La boite à moustache permet de comparer deux populations différentes en fonction d'une variable.



Nous avons donc prit comme population, les client ayant fait un défaut contre les client n'en ayant pas fait, en fonction de leurs âge. Nous avons obtenu le graphe ci-dessus.
On remarque bien que sur l'ensemble de nos données les personnes ayant fait un défaut bancaire, ont tendance à être plus jeune que ceux qui n'en ont pas fait.

Ensuite, nous avons effectué la boite à moustache de la variable adresse en fonction de la population ayant fait défaut ou non.



On peut également constater ici que dans nos données les clients ayant fait défaut ont tendance à habiter dans leur logement depuis une plus courte durée que les clients n'ayant pas fait de défauts.

Pré-traitement des données :

Nous avons décidé de supprimer les variables ncust et customer, car d'après nous elles n'étaient pas nécessaire pour la prédiction du défaut bancaire. Nous avons également modifié la variable branch pour la transformer en variable catégorielle afin de ne pas effectuer des calculs inutile dessus.
Par la suite, nous avons renommé les variables en Français pour des facilités de compréhension logique.

Définition de la méthode d'évaluation des classifieurs :

On peut évaluer la qualité d'un classifieur de plusieurs manières.
On peut s'intéresser aux mesures d'évaluations des succès et échecs. Un succès est une prédiction correcte sur l'ensemble d'apprentissage, tandis qu'un échec est une prédiction fausse sur ce même ensemble. (classe réelle = 'True'; prédictions = 'False' désigne une mauvaise prédiction du classifieur).
« Vraies positifs » (VP) et « Vraies Négatifs » (VN), sont définies comme étant les succès. « Faux négatifs » (FN) et « Faux positifs » (FP) sont les échecs.

Pour évaluer la qualité de notre classifieur, nous avons 4 mesures d'évaluations de succès et d'échec, calculé grâce à VP, FP, VN et FN.

On a : la Sensibilité (proportion d'exemple de test positifs correctement prédits), la Spécificité (proportion d'exemples de test négatifs correctement prédits), la Précision (proportion de prédictions positives correctes), et le Taux de vrais négatifs (proportion de prédictions négatives correctes).
Ces mesures permettent de sélectionner le classifieur qui maximise les succès.

Ensuite, on peut s'intéresser aux matrices de confusion des classifieurs. La matrice de confusion contient toutes les valeurs calculées précédemment et permet donc de juger de la qualité du classifieur.

Dans notre cas nous avons plutôt choisi d'utiliser les indicateurs numériques AUC, qui représente l'aire sous les courbes ROC. Plus cette valeur est élevée, meilleur sera le classifieur.
Nous avons choisi cette méthode d'évaluations de nos classifieurs car elle est indépendante des matrices de coûts, ce qui n'est pas le cas des matrices de confusion dans lesquelles ont cherche à minimiser les coûts.

Également, la représentation graphique des courbes ROC permet de visualiser et comparer simultanément la performance de nos classifieurs. Ces indicateurs sont aussi opérationnels dans le cas de distributions très déséquilibrés. Chaque point de coordonnées de la courbe ROC est calculé à partir d'une matrice de confusion, donc au vu de toutes ces caractéristiques, nous avons pensé que l'AUC était plus pertinent que les matrices de confusion.

Création des données d'apprentissages et de tests :

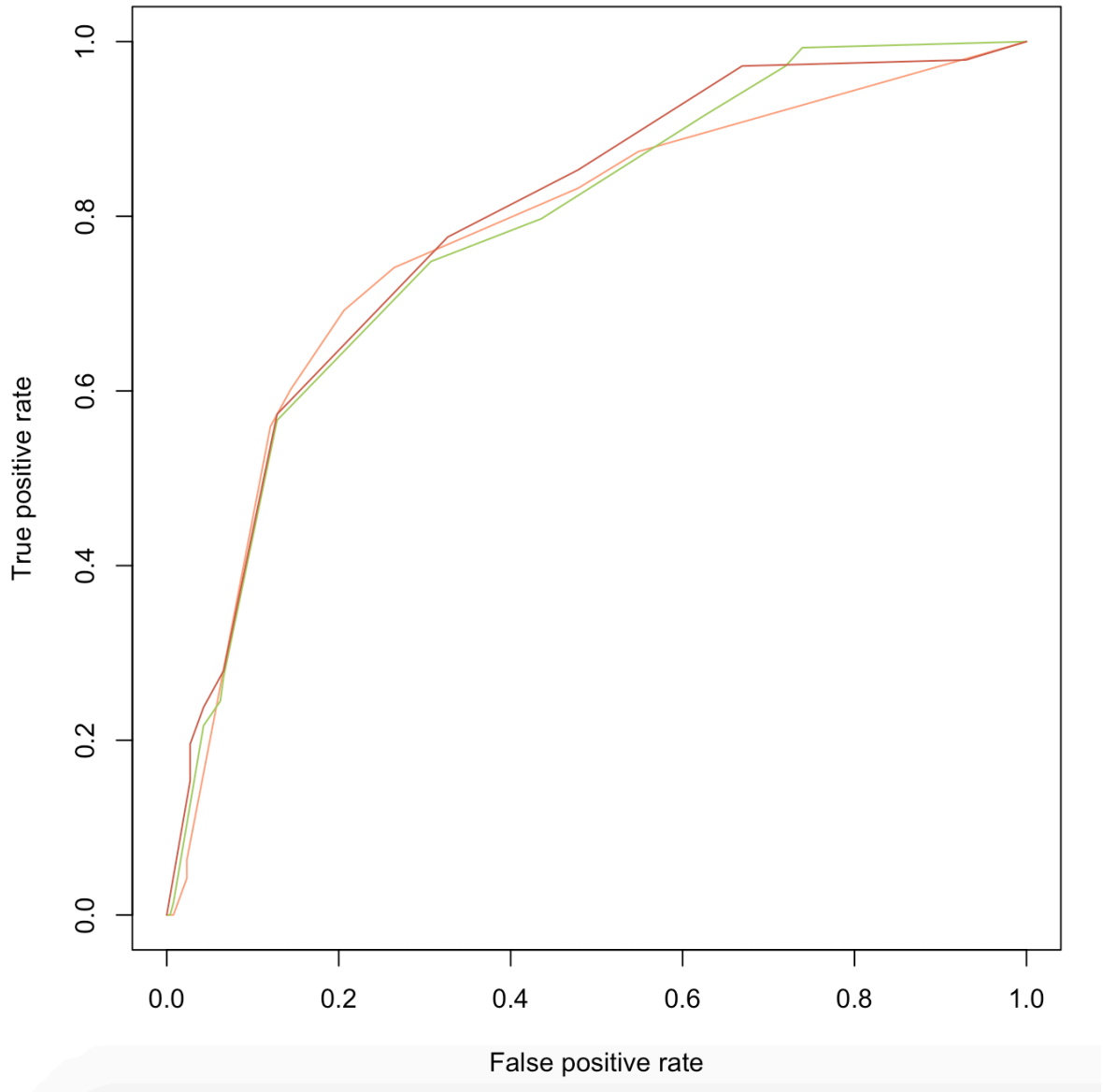
Nous avons séparé nos 1200 observations en deux jeux de données par partitionnement (percentage split). Nous avons sélectionné les 800 (2/3) premières observations pour notre ensemble d'apprentissage, et les 400 (1/3) autres pour notre ensemble de test.

Description des configurations des classifieurs générés :

Dans un premier nous avons utilisé les classifieurs des librairies Rpart, C5.0, et Tree que nous avons appliqué à l'ensemble d'apprentissage. Par la suite nous avons dessiné les arbres généré par les trois classifieurs. Ensuite nous avons appliqué ces classifieurs à l'ensemble de test, pour en obtenir les matrices de confusions, et ainsi en tirer toutes les informations disponible. (sensibilité, spécificité, précision, et le taux de vrais négatifs).
Malgré les calculs effectué nous n'avons pas tenu compte des résultats, car nous avons décidé d'utiliser comme méthode d'évaluation les indicateurs numériques AUC.

Préalablement au calcul et à l'affichage de nos courbes ROC nous avons calculer les probabilités de prédictions de OUI et de NON, pour chaque classifieurs par rapport à la variable défaut.

Nous avons obtenus les courbes ROC suivantes pour Rpart C5.0 et Tree:



Sur ce graphique nous avons la courbe orange qui est la courbe ROC du classifieur Rpart, la courbe verte pour C5.0, et la courbe rouge pour Tree.

En observant ce graphique on peut supposer que c'est la courbe rouge qui a l'AUC le plus élevé.

En calculant l'air sous les courbes ROC, nous avons obtenus les AUC suivant :
Pour Rpart : 0.7773258
Pour C5.0 : 0.7829583
Pour Tree : 0.7932301

Ces résultats confirme notre hypothèse qui était que le meilleur classifieur selon notre méthode d'évaluation est Tree, car sont AUC est le plus élevé.

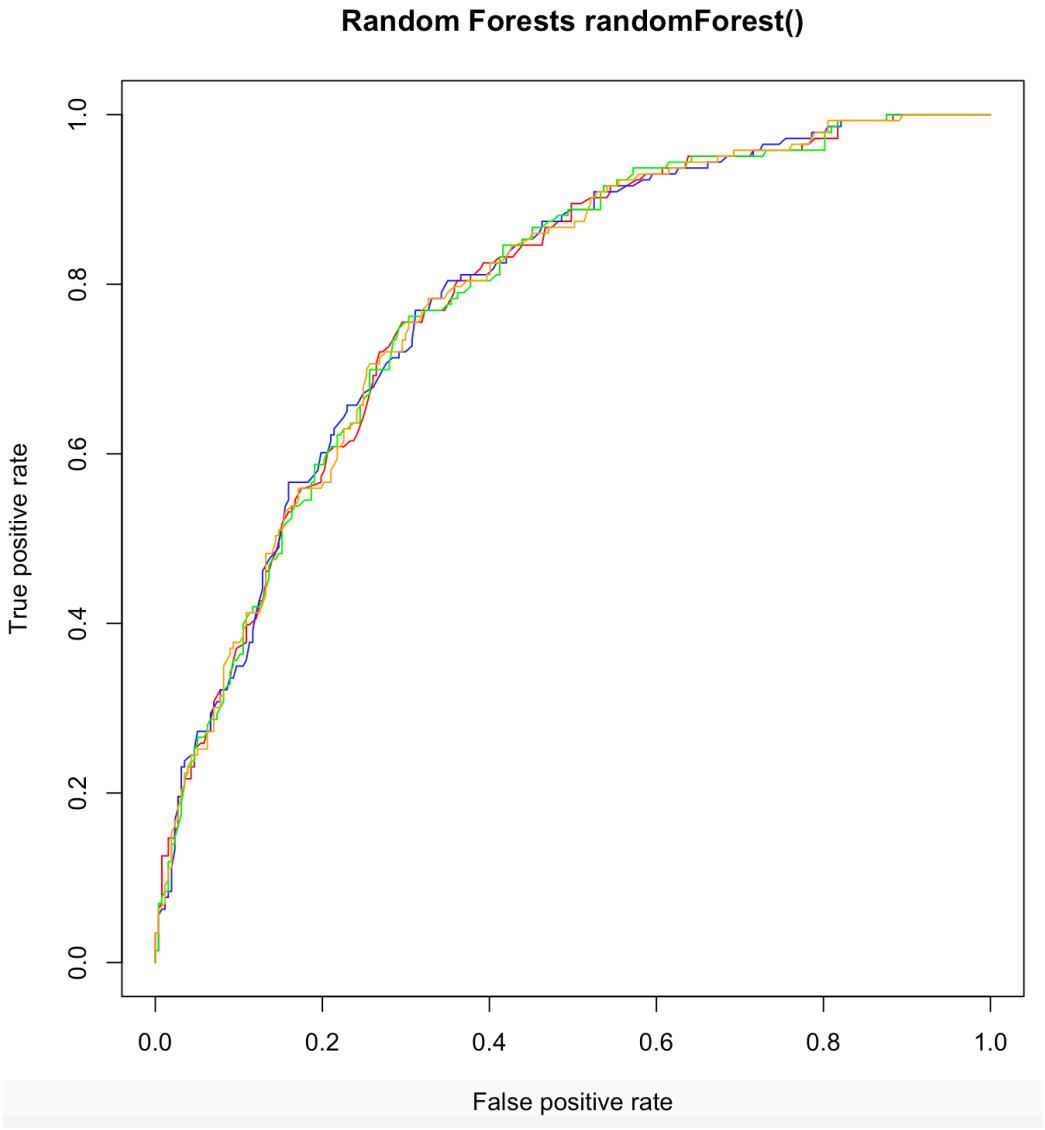
Par la suite, nous avons utilisé les fonctions vue en cours qui résument toutes les étapes que nous avons effectué précédemment pour d'autres types de classifieurs ainsi que différents paramètres, puis déterminé leurs qualités grâce à l'indicateur AUC.

Nous avons donc voulu tester le classifieur Random Forest, qui se base sur les méthodes de Bagging et de sélection aléatoire de dimensions, qui consistent au fait combiner différents modèles prédictif.
Le principe étant de créer de multiples arbres de décision appris sur des sous-ensembles de données légèrement différents (d'où l'aléa). Cette forêt d'arbres va faire des prédictions sur ces sous-ensembles de données, et la prédiction finale sera basée sur la classe majoritairement prédite. Plus le nombre d'arbres est important plus la prédiction sera pertinente, car on va réduire les risques d'erreurs du classifieur.

Ses avantages sont qu'il est un classifieur plutôt efficace si on a un grand nombre de variables à traiter dans notre jeu de données, et son principe de fonction est adéquate à des jeux de données où la répartition des classes est déséquilibrée.
Ses inconvénients viennent du fait qu'il se base sur pleins d'arbres eux-mêmes traitant de sous-ensembles différents à chaque fois et donc la présence d'exceptions dans les données, va rendre difficile l'interprétation du résultat.

Comme paramètres dans notre fonction, nous avons le nombre d'arbres ntree, et le nombre de variables sélectionnées par arbres mtry. (mtry << M le nombre total de variables)
Nous avons testé sur 300 arbres pour 3 puis 5 variables par arbres. Puis de même avec 500 arbres.

Un de nos tirages nous à donner ces courbes ROC ci dessous :



Rouge : ntree = 300, mtry =3, AUC = 0.783148757856929
Bleu : ntree = 300, mtry = 5, AUC = 0.784522870125983
Vert : ntree = 500, mtry = 3, AUC = 0.782835841201601
Orange : ntree = 500, mtry = 5, AUC = 0.783638540447879

Le problème étant que pour chaque tirages différents nos valeurs AUC sont différentes à cause de l'aléa et nous ne pouvons donc pas tirer une réelle conclusion sur nos prédictions avec Random Forest. Nous retiendrons donc de façon après plusieurs observations que Random Forest donne une valeur pour l'AUC entre 0,783 et 0,785 environ.

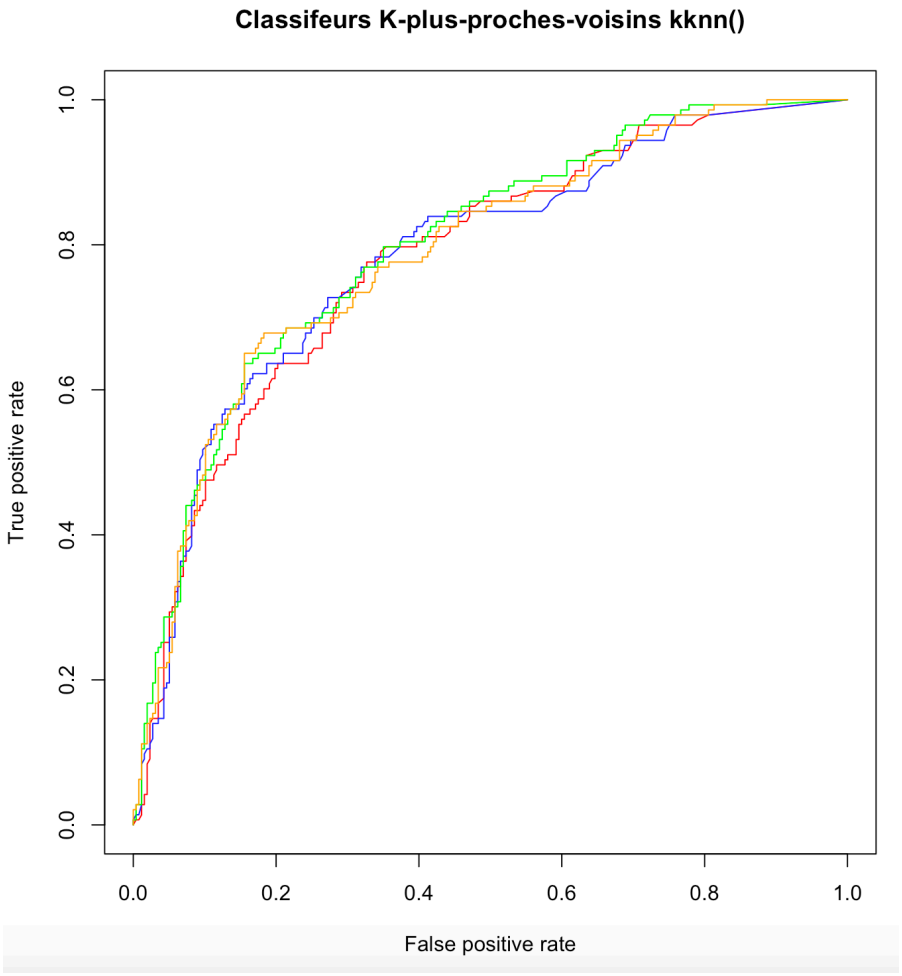
Nous avons, ensuite, voulu tester et juger de la qualité du classifieur basé sur l'approche des « K-Plus Proches Voisins », ou « K-Nearest Neighbors » (KNN). Pour cela, de la même façon que précédemment nous avons utilisé les fonctions créent sur R lors des cours, pour effectuer chaque étapes menant à l'affichage des courbes ROC et le calcul des valeurs AUC.

Le principe de ce classifieur est de projeter l'espace des données, où chaque exemple de l'EA est représenté comme point dans cette espace des données, ce point étant placé à une certaine position selon les valeurs de ces variables.
On choisi ensuite K, un nombre de « plus proches voisins » considérés pour prédire une instance, si cette instance est entourée par une classe majoritairement par rapport à une autre c'est celle-ci qui sera prédite.

Bien entendu, les calculs liés à l'application de ce classifieur tiennent compte du type des données que l'on va traiter et tout cela est configuré dans la librairie R nommé « kknn ».

Comme paramètres dans notre fonction nous avons :
- k : le nombre de « plus proches voisins » considérés
- Distance : la valeur de la distance Minkowski considéré

Nous avons obtenu les courbes ROC suivantes pour différents paramètres :



Rouge : k=10, distance=1, AUC = 0.778795134826263
Bleu : k=10, distance=2, AUC = 0.783107942641017
Vert : k=20, distance=1, AUC = 0.797202797202797
Orange : k=20, distance=2, AUC = 0.788101004054311

On remarque bien que pour un plus grand nombre de « plus proches voisins » et une distance de Minkowski plus faible, le classifieur est plus pertinent au vue de l'AUC.

C'est pour cela que par la suite, nous avons voulu tester, pour un nombre de « k plus proches voisins » bien plus élevé, par exemple 100 puis 200, pour voir si la différence était flagrante au niveau des AUC.

- k=100, distance=1, AUC = 0.808603847514352
- k=100, distance=2, AUC = 0.798590514543821
- k=200, distance=1, AUC = 0.805637941824712
- k=200, distance=2, AUC = 0.796059971157247

On peut voir que l'écart n'est pas énorme, mais si on prends un nombre k trop élevé on perds en efficacité dans notre prédiction. En tâtonnant, nous avons constaté que la valeur AUC la plus élevée est atteinte lorsqu'on prends k=90 et distance=1, AUC = 0.808631057658294.

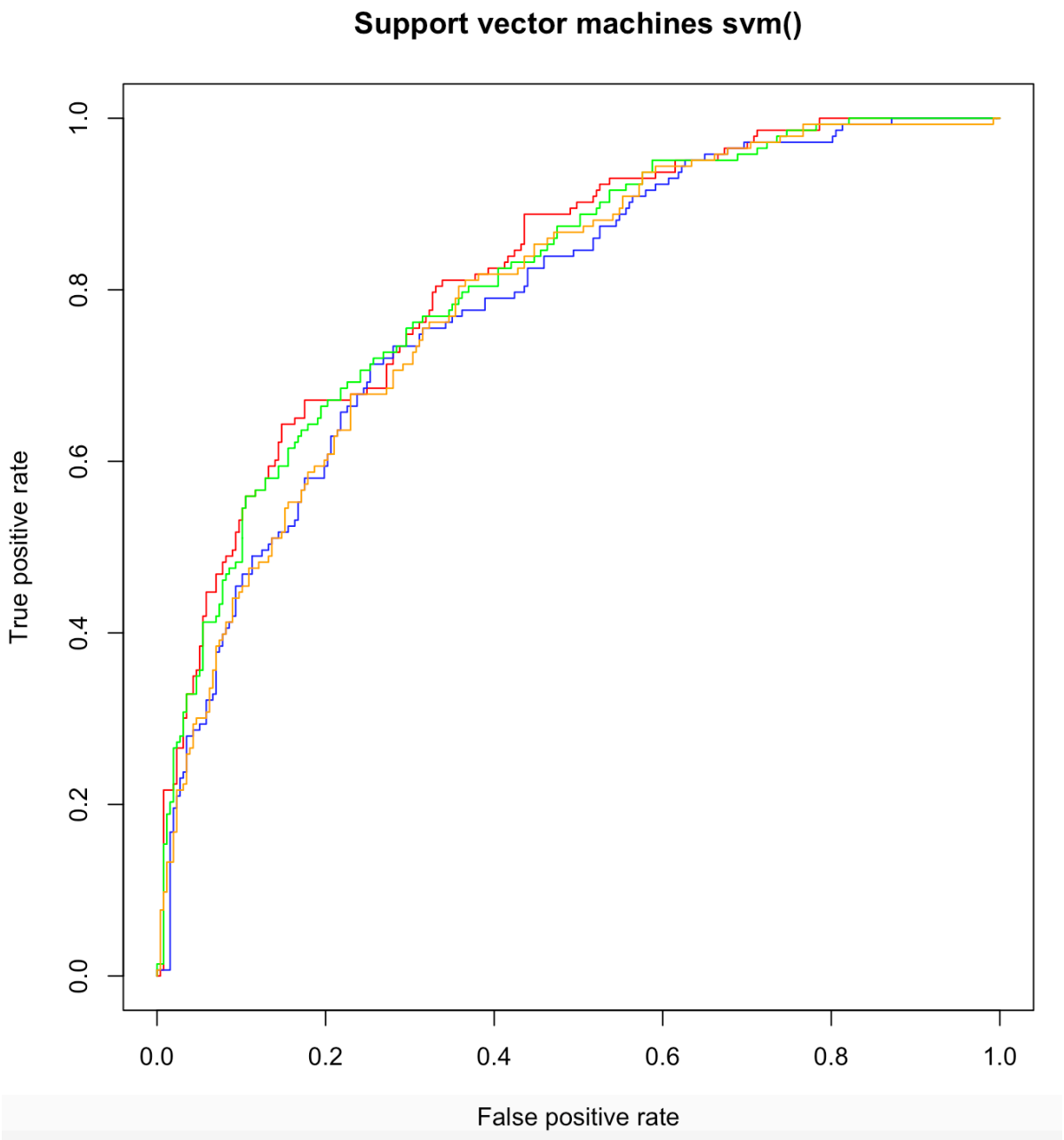
Après cela, nous avons décidé d'utiliser encore une fonction vue en cours, pour appliquer le classifieur « Support Vector Machine ».
Ce classifieur est basé sur la notion d'hyperplan, frontière décisionnelle entre les classes. On projette les données contenant M variables dans un espace à M-dimension. Puis on cherche à définir une frontière entre les sous espaces de points de ces classes. C'est en quelque sortes l'épaisseur de cette frontière qui va donner l'efficacité de notre classifieur, on va donc chercher à maximiser cette distance qui sépare les deux points les plus proches de deux classes, qui sont les vecteurs supports. Cependant, un problème subsiste, les produits scalaires de vecteurs dans des dimensions élevées sont très couteux, et il nous faut donc utiliser les différents types de fonctions noyaux vues en cours (linéaire, polynomial, radial, sigmoïde). Cette méthode est implémenter par la librairie « e1071 » de R, et nous permet de créer notre classifieur.

Ses avantages sont que c'est une approche efficace pour des données a dimension élevée, et qu'elle ne nécessite pas forcément beaucoup d'exemples par rapport au dimension pour être pertinente.

Il existe aussi des inconvénients, comme le choix du type de la fonction noyau (c'est pourquoi nous avons testé tous les types), et le risque de sur-apprentissage du classifieur lorsqu'il ne dispose de peu de données.

Comme paramètres dans notre fonction nous avons donc seulement :
- Type de fonction noyau : linéaire, polynomial, radial, sigmoïde.

Nous avons obtenu les courbes ROC suivantes pour différents paramètres :



Rouge : kernel=linear, AUC = 0.819351854371311
Bleu : kernel=polynomial, AUC = 0.786808522217082
Vert : kernel=radial, AUC = 0.811433702484287
Orange : kernel=sigmoid, AUC = 0.792196130717529

On voit bien ici, que le type de noyau le plus pertinent est le type « linéaire », c'est celui qui nous donne la valeur AUC la plus élevée. On peut remarquer aussi que sur tous nos classifieurs, c'est celui qui renvoi la valeur des AUC la plus élevée.

Enfin, nous avons souhaité, appliquer le classifieur utilisant les « Méthodes Bayésiennes », basées sur les Probabilités (Probabilités conditionnelles, Indépendances des variables, etc…).

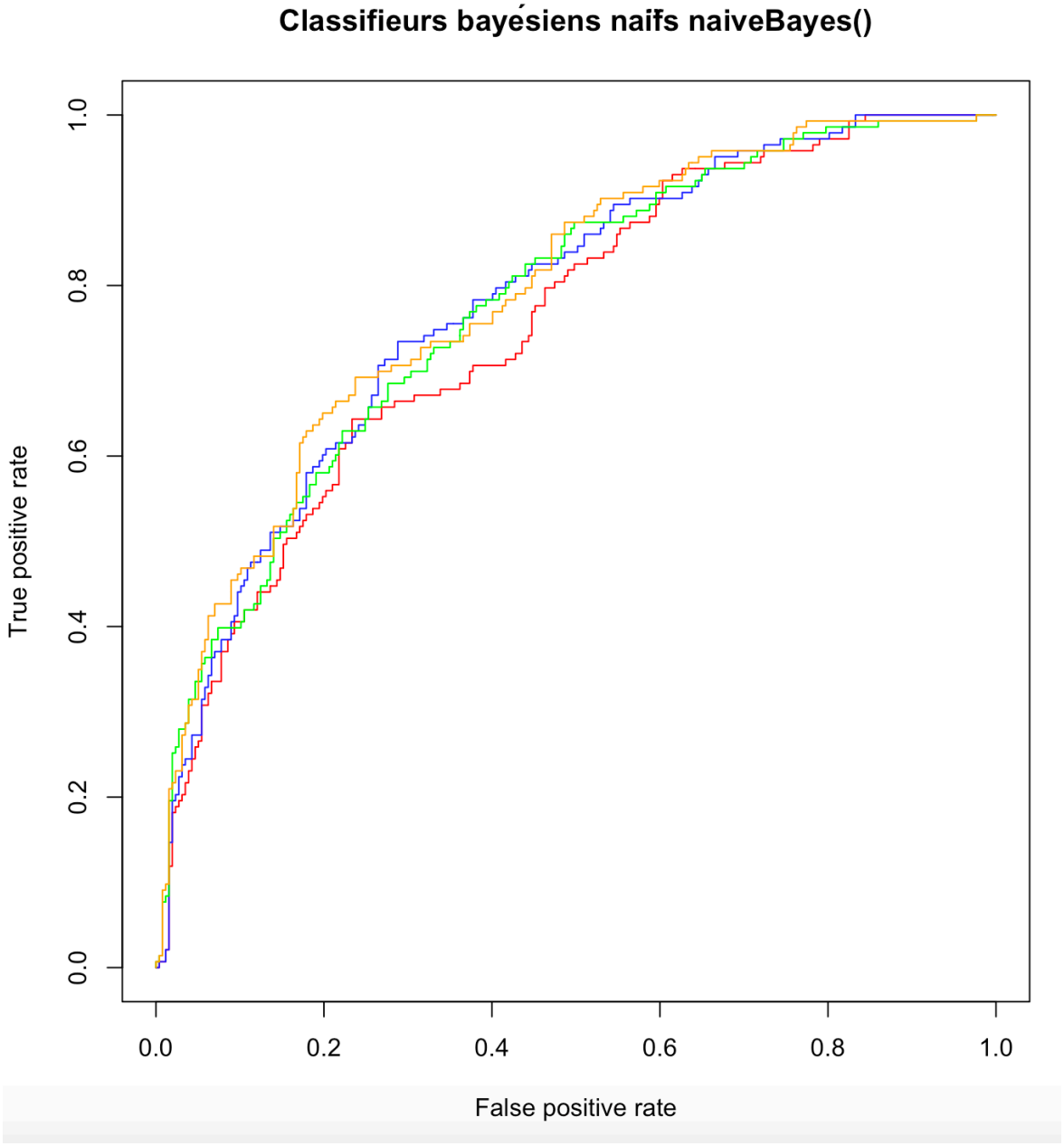
Les probabilités d'occurrences des valeurs des variables dans une classe sont considérées indépedantes, et on combine les probabilités du nombre d'exemple de l'ensemble EA appartenant à une classe C, au nombre d'exemple de l'ensemble EA appartenant à la classe C et possédant la valeur x.

Encore une fois, nous avons utilisé la fonction vu en cours, ayant pour paramètres :
- Laplace = intensité du lissage de Laplace, lissage des frontières décisionnelles entre les classes
- Usekernel = utilisation d'un noyau d'estimation de densité ou non

Les classifieurs bayésiens ont montré leur efficacité concernant la classification de document et le filtrage de spams.

Évidemment, nous avons utilisé les fonctions de la librairie R « naivebayes » pour créer notre propre fonction et en tirer les AUC.

Nous avons obtenu les courbes ROC suivantes pour différents paramètres :



Rouge : laplace=0, usekernel=FALSE, AUC = 0.753149574161247
Bleu : laplace=20, usekernel=FALSE, AUC = 0.778427797883051
Vert : laplace=20, usekernel=TRUE, AUC = 0.775189790753994
Orange : laplace=0, usekernel=TRUE, AUC = 0.788386710565698

C'est donc lorsqu'on n'utilise pas le lissage de Laplace et avec un noyau d'estimation de de densité que l'on a le plus de pertinence dans nos prédictions avec les AUC étant plus élevés AUC = 0.788386710565698.

Nous n'avons pas dénier utiliser le classifieur « ANN = Artificial Neural Networks» qui se veut être une reproduction du fonctionnement des neurones biologiques et qui utilisent un réseau de neurones interconnectés. Car il contient une part d'aléa lui aussi, et nous donnes donc différentes valeurs des AUC, ce qui ne nous permet pas de tirer une réelle conclusion sur la qualité de nos prédictions.

Comme expliqué précédemment, nous avons décidé de choisir le meilleur classifieur en fonction de l'indice AUC. Après avoir comparé tous les indices de de ces classifieurs pour différent paramètres, nous décidé de conserver le classifieur SVM pour notre prédiction. Nous pouvons justifier ce choix par le fait que pour un noyau dit de type « linear », nous obtenons la valeur la plus élevé de l'indice AUC de tous nos classifieur. On peut donc considérer que dans notre cas, et avec notre critère de sélection, le classifieur SVM sera le plus performant.

Résumé des résultats de l'application du classifieur sélectionné à l'ensemble de données à prédire :

L'application de notre classifieur SVM à l'ensemble des données à prédire nous a donné les résultats suivant:
Nous avons obtenus 207 prédictions de non défauts de paiements, et donc 93 prédictions prédictions de défauts de paiements.
Pour les probabilités de prédire un non défaut de paiement nous avons obtenus des valeurs entre 0.005909 pour la plus faible et 0.999391 pour la plus élevé. Ensuite nous avons obtenus une moyenne de 0.637270 ainsi qu'une médiane s'élevant à 0.660682. On a également pu constater que les 50% des valeurs compris entre le premier et le 3ème quartiles sont compris entre 0.462345 et 0.850404.
Pour les probabilités de prédire un défaut de paiement nous avons obtenus des valeurs entre 0.0006086 pour la plus faible et 0.9940910 pour la plus élevé. Ensuite nous avons obtenus une moyenne de 0.3627295 ainsi qu'une médiane s'élevant à 0.3393176. On a également pu constater que les 50% des valeurs compris entre le premier et le 3ème quartiles sont compris entre 0.1495960 et 0.5376546. On constate que la probabilité d'obtenir un défaut de paiement, est, en moyenne, beaucoup plus faible. Nous pouvons également constaté que ce soit pour la classe « oui » ou la classe « non » la probabilité maximale est extrêmement proche de 1 ce qui veut dire que dans certains cas nous pouvons quasiment être sur que notre prédictions est correcte.

Conclusion :

Pour conclure, nous avons été amenés à penser durant la réalisation de cette application, que plus l'ensemble d'apprentissage serait conséquent, et donc comportant un plus grand notre de profils de clients différents, plus notre prédiction serait fiable. On peut penser qu'avec des données de départ de taille 3 ou 4 fois supérieure, la confiance en notre prédiction aurait été beaucoup plus forte.
On a pu également constaté une cohérence dans nos résultats, car nous avons obtenus environ 67% de non défaut, alors que nos données de départ comprenaient 63% de non défaut, ce qui n'est pas très éloigné. Plus notre ensemble de départ sera grand plus ces pourcentages devraient être proche.

Enfin, nous n'avons pas rencontré de réelles difficultés, nous avons suivi le déroulement des TP fait en cours pour les différents aspects de notre analyse, ce qui a été d'une grande aide. Nous avons également utilisé le cour détaillé pour tenter de faire la meilleure analyse de ces résultats possible, ainsi que choisir le meilleur classifieur.