Insurance

Raphael MERCIER and Alexis SAVATON

Table of contents

Problématique / Objectif **2**

Partie 1 : Notre Base de Données **2**
- Type de variable . . . 2
- Les ajustements . . . 3
- Statistiques descriptives . . . 3
  - Nos variables quantitatives . . . 3

Partie 2 : Construction des modèles **4**
- La préparation . . . 4
- Nos modèles . . . 4
  - LDA . . . 4
  - QDA . . . 7
  - KNN . . . 9
  - LOGISTIC REGRESSION . . . 11
  - DECISION TREE . . . 13
  - RANDOM FOREST . . . 17
  - BOOSTING . . . 19

Partie 3 : C'est lequel le best ? **21**
- Comparaison Courbe ROC . . . 21
- Comparaison des F1-score . . . 21
- Conclusion : . . . 22
- Source . . . 22

Problématique / Objectif

Nos clients, une compagnie d'assurance santé, nous demande de l'aide pour savoir quels sont les clients qui pourraient être intéressés de souscrire une assurance pour leur voiture. Pour cela nous disposons d'une large base de donnée que nous utiliseront pour construire nos modèles.

⇒ On veut finalement déterminer notre meilleur modèle de prédiction pour ne pas rater de clients potentiels.

Partie 1 : Notre Base de Données

Type de variable

La variable que l'on cherche à prédire est ***Response***, elle vaut 0 si le client n'est pas intéressé ou 1 s'il est intéressé.

Pour cela on dispose de 10 autres variables :

- ***Gender*** : *Factor*, le genre de l'individu
- ***Age*** : *Integer*, l'âge de l'individu
- ***Driving_License*** : *Factor*, Variable codée 0 si l'individu n'a pas son permis, et 1 s'il l'a
- ***Region_Code*** : *Factor*, code unique correspondant à la région de l'individu
- ***Previously_Insured*** : *Factor*, Variable codée 0 si l'individu n'a pas d'assurance pour son véhicule, et 1 s'il en a une
- ***Vehicule_Age*** : *Factor*, l'age du véhicule
- ***Vehicule_Damage*** : *Factor*, Variable codée 0 si le véhicule de l'individu n'a jamais été endommagé, et 1 s'il l'a déjà été
- ***Annual_Premium*** : *Integer*, la somme que l'individu doit payer comme prime d'assurance chaque année
- ***Policy_Sales_Channel*** : *Integer*, code anonymisé pour le canal de communication avec l'individu (mail, téléphone etc..)
- ***Vintage*** : *Integer*, nombre de jours depuis lequel l'individu est client de l'assurance

Les ajustements

Notre base de données contient 381109 clients et est fortement déséquilibrée. Le taux d'individus avec *Response = 1* est de 12.26% alors que pour *Response = 0* il est de 87.74%. Ce problème d'équilibrage pose des problèmes dans l'estimation des modèles, ainsi que dans leur qualité pour prédire *Response = 1*.

Nous avons donc dû faire un rééquilibrage de la base :

- Nous avons créé un nouveau data frame, contenant dans un premier temps tous les individus prenant *Response = 1*.
- Ensuite, on a tiré aléatoirement des individus prenant *Response = 0* pour compléter.
- Ce qui nous donne finalement le data frame suivant contenant 100 000 individus :
  - 46.62% pour *Response = 1*
  - 53.38% pour *Response = 0*

Notre nouvelle base de donnée se porte maintenant sur 100 000 individus, et la variable *Response* contient désormais 2 classes pratiquement équilibrées. Ces modifications nous permettront de construire par la suite des modèles de meilleure qualité, tout en gardant un temps de calcul raisonnable.

On avait également des valeurs aberrantes dans la variable *Annual_premium*, que l'on a décidé de supprimer pour éviter de diminuer la performance de nos modèles prédictifs.

Statistiques descriptives

Nos variables quantitatives

Quelques statistiques de nos données sur nos variables numériques :

Age	Annual_Premium	Vintage
Min. :20.00	Min. : 2630	Min. : 10.0
1st Qu.:27.00	1st Qu.:24329	1st Qu.: 81.0
Median :40.00	Median :31797	Median :154.0
Mean :40.49	Mean :29713	Mean :154.3
3rd Qu.:50.00	3rd Qu.:39361	3rd Qu.:227.0
Max. :81.00	Max. :63432	Max. :299.0

Dans notre base de données, la moyenne d'âge de nos individus est de 40 ans, ils paient en moyenne 29 713$ d'assurance par an et sont en moyenne clients chez l'assureur depuis 154 jours.

Partie 2 : Construction des modèles

La préparation

1. Découpage

Nous avons décidé de faire 2 splits, un pour les modèles peu gourmands en puissance de calcul, et un second avec moins d'individus pour les modèles plus gourmands. La proportion est la même dans les deux splits, 2/3 des individus pour train, et 1/3 pour test.

2. Les Recettes

Nous avons plusieurs recettes adaptées à certains modèles :
- **df_rec_1** : recette sans la variable Policy_Sales_Channel, pour éviter un soucis de colinéarité parfaite - **df_rec_num** : recette avec seulement les variables numériques
- **df_rec** : recette avec toutes les variables
- **df_rec_boosting** : recette pour le boosting avec moins d'individus

Nos modèles

LDA

Pour notre tout premier modèle nous avons une LDA (Linear Discriminant Analysis), qui utilise la recette : **df_rec_1**.

- La matrice de confusion :

	Prédiction		
Réalité	**N**	**Y**	**Total**
N	7996	5052	**13048**
Y	441	10875	**11316**
Total	**8437**	**15927**	**24364**

- Les mesures de performance du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	77.45
Erreur globale de classement	22.55
Vrai négatif	94.77
Vrai positif	68.28
Précision	61.28

Petit point sur les mesures de performance :

- ***Accuracy*** : C'est le pourcentage total des prédictions correctes par rapport au nombre total d'échantillons. En d'autres termes, c'est le nombre d'échantillons correctement classés divisé par le nombre total d'échantillons.

$$Accuracy = \frac{Nombre\ d'echantillons\ correctement\ classes}{Nombre\ total\ d'echantillons} \tag{1}$$

- ***Erreur globale de classement*** : C'est le pourcentage total des prédictions incorrectes par rapport au nombre total d'échantillons. C'est essentiellement le complément de l'accuracy.

$$Erreur\ globale\ de\ classement = \frac{Nombre\ d'echantillons\ incorrectement\ classes}{Nombre\ total\ d'echantillons} \tag{2}$$

- ***Vrai négatif*** : C'est le nombre d'échantillons de la classe négative (classe 0) qui ont été correctement prédits comme négatifs.

$$Vrai\ negatif\ (TN) = \frac{Nombre\ d'echantillons\ classes\ correctement\ comme\ negatifs}{Nombre\ total\ d'echantillons\ reels\ negatifs} \tag{3}$$

- ***Vrai positif*** : C'est le nombre d'échantillons de la classe positive (classe 1) qui ont été correctement prédits comme positifs.

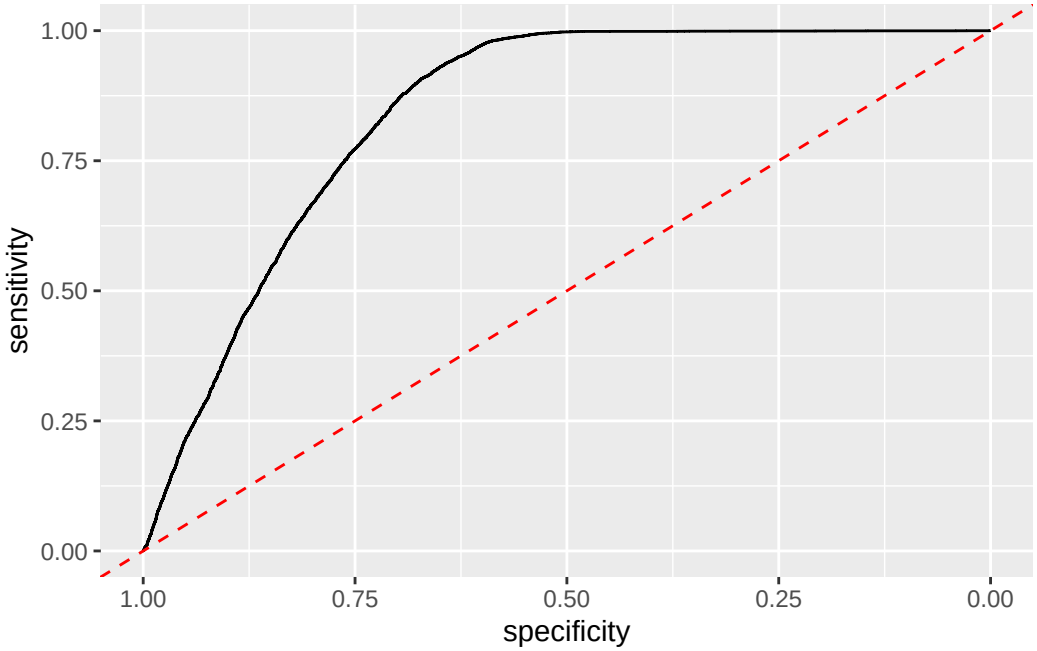
$$Vrai\ positif\ (TP) = \frac{Nombre\ d'echantillons\ classes\ correctement\ comme\ positifs}{Nombre\ total\ d'echantillons\ reels\ positifs} \tag{4}$$

- ***Précision*** : C'est le pourcentage d'échantillons de la classe positive (classe 1) qui ont été correctement prédits comme positifs parmi tous les échantillons prédits comme positifs. C'est une mesure de la qualité des prédictions positives du modèle.

$$Prcision = \frac{Nombre\ de\ vrais\ positifs}{Nombre\ de\ vrais\ positifs + Nombre\ de\ faux\ positifs} \tag{5}$$

Revenons à notre modèle LDA. Bien qu'il soit moyen avec une accuracy de 77,45 %, son point positif est qu'il permet de minimiser considérablement le nombre d'individus qui sont en réalité intéressés et que le modèle avait prédit comme ne l'étant pas. Ce qui peut en effet être important pour une compagnie d'assurance, prédire qu'un individu est intéressé alors qu'il ne l'est pas aura probablement moins d'incidence que de rater un client qui était intéressé.

- La Courbe ROC :




```
Area under the curve: 0.8419
```

On a une aire sous la courbe de 0.84196, le modèle a une capacité de discrimination correcte pour une LDA.

QDA

Notre modèle QDA (Quadratic Discriminant Analysis) utilise la recette : **df_rec_1**, tout comme la LDA.

- La matrice de confusion :

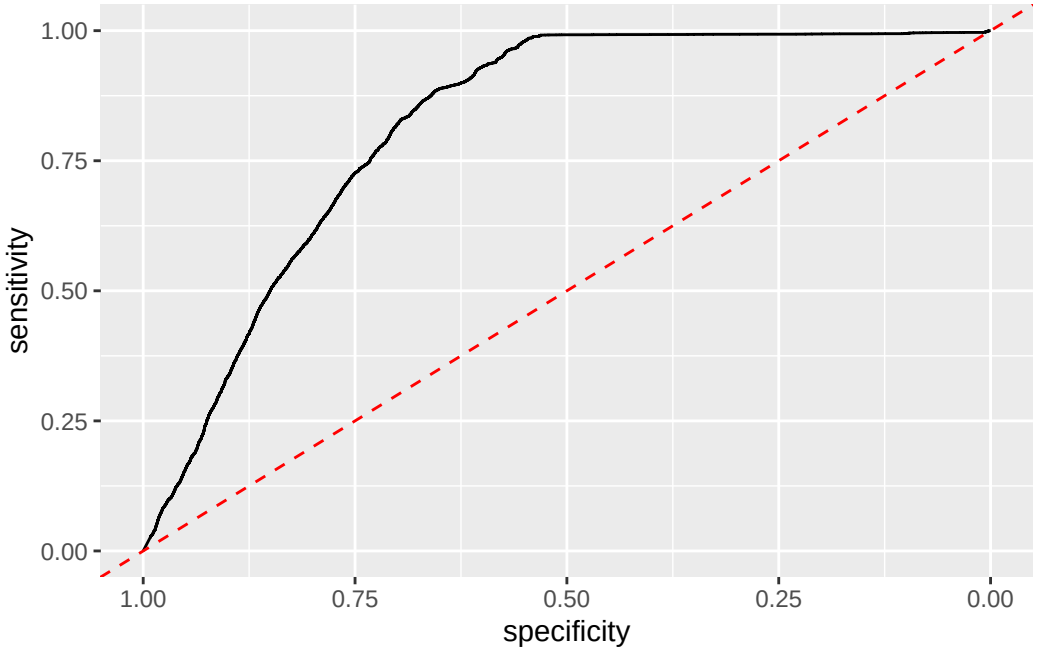
	Prédiction		
Réalité	**N**	**Y**	**Total**
N	9149	3899	**13048**
Y	2060	9256	**11316**
Total	**11209**	**13155**	**24364**

- Mesure des performances du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	75.54
Erreur globale de classement	24.46
Vrai négatif	81.62
Vrai positif	70.36
Précision	70.12

Le modèle QDA est assez similaire au modèle LDA, avec une accuracy de 75.54%, mais il rate un nombre plus élevé de clients potentiels, ce qui en fait un modèle moins intéressant que la LDA.

- La courbe ROC :




```
Area under the curve: 0.8189
```

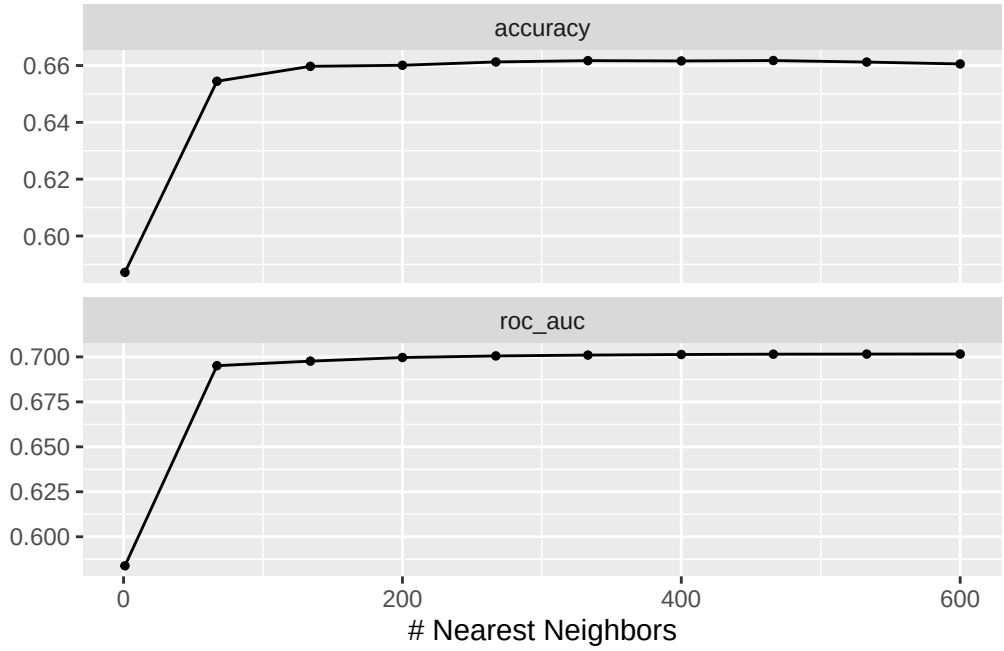
Une aire sous la courbe un peu inférieure à celle de la LDA.

KNN

Nous avons maintenant notre modèle des KNN (k-Nearest Neighbours), ce modèle utilise la recette : **df_rec_num**.

- Optimisation des paramètres :

Tout d'abord pour le modèle KNN nous devons optimiser le nombre de voisin k :



Le nombre de voisin optimal est de : **466**.

- La matrice de confusion :

	Prédiction		
Réalité	**N**	**Y**	**Total**
N	7693	5355	**13048**
Y	2850	8466	**11316**
Total	**10543**	**13821**	**24364**

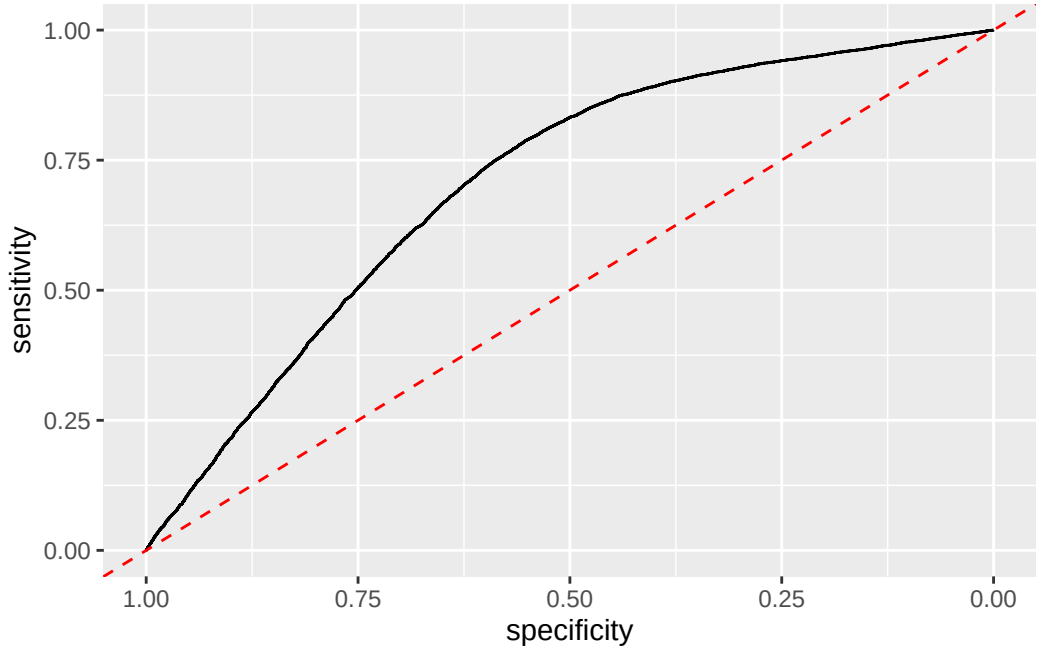
- Mesure des performances du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	66.32
Erreur globale de classement	33.68
Vrai négatif	72.97

Métrique	Valeur en %
Vrai positif	61.25
Précision	58.96

Pour ce modèle, l'accuracy est très faible, il sera très peu fiable pour prédire si nos individus sont intéréssés ou non.

- La courbe ROC :




```
Area under the curve: 0.705
```

La courbe ROC a une aire également très faible. Le nombre optimal de voisins est élevé, ce qui demande des ressources conséquentes pour calculer ce modèle qui est au final très mauvais, on ne le gardera pas.

LOGISTIC REGRESSION

Pour essayer de prédire une variable binaire nous pouvons aussi utiliser le modèle Logit, plus couramment utilisé en économétrie.
Ce modèle utilise la recette : **df_rec_1**.

- La matrice de confusion :

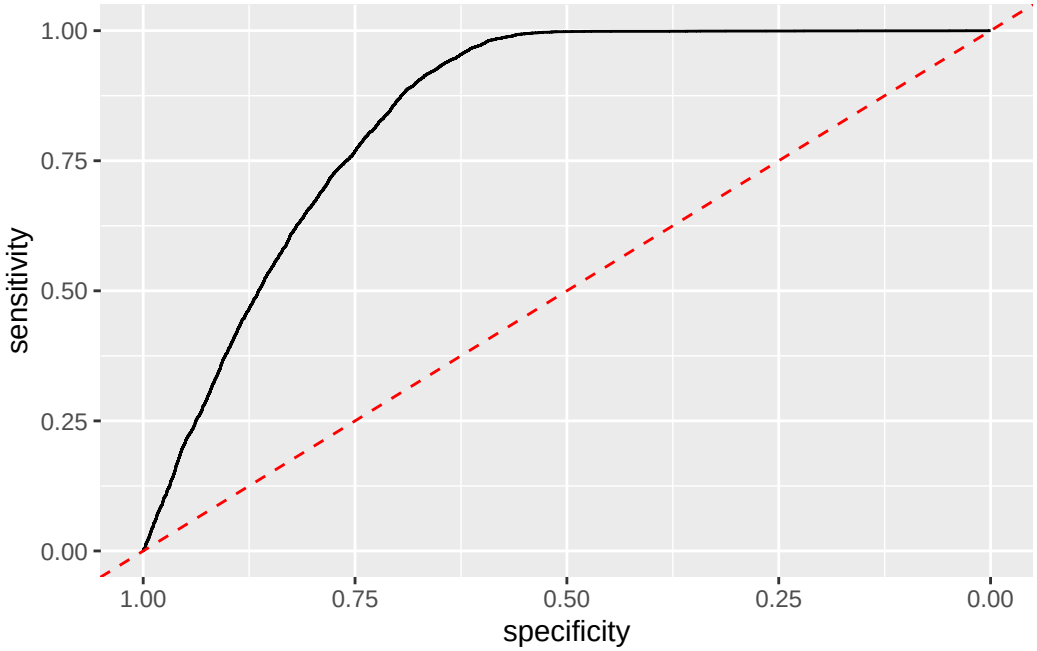
	Prédiction		
Réalité	**N**	**Y**	**Total**
N	8526	4522	**13048**
Y	830	10486	**11316**
Total	**9356**	**15008**	**24364**

- Mesure de performance du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	78.03
Erreur globale de classement	21.97
Vrai négatif	91.13
Vrai positif	69.87
Précision	65.34

Nous obtenons des résultats assez similaires à la LDA, avec une accuracy un peu plus élevée pour ce modèle et un peu plus de vrai positif pour la LDA. Ce modèle, tout comme la LDA est donc assez moyen.

- La courbe ROC :




```
Area under the curve: 0.842
```

Une courbe ROC aussi assez similaire à celle de LDA, son aire est de 0.842. Le modèle a une capacité de discrimination correcte.

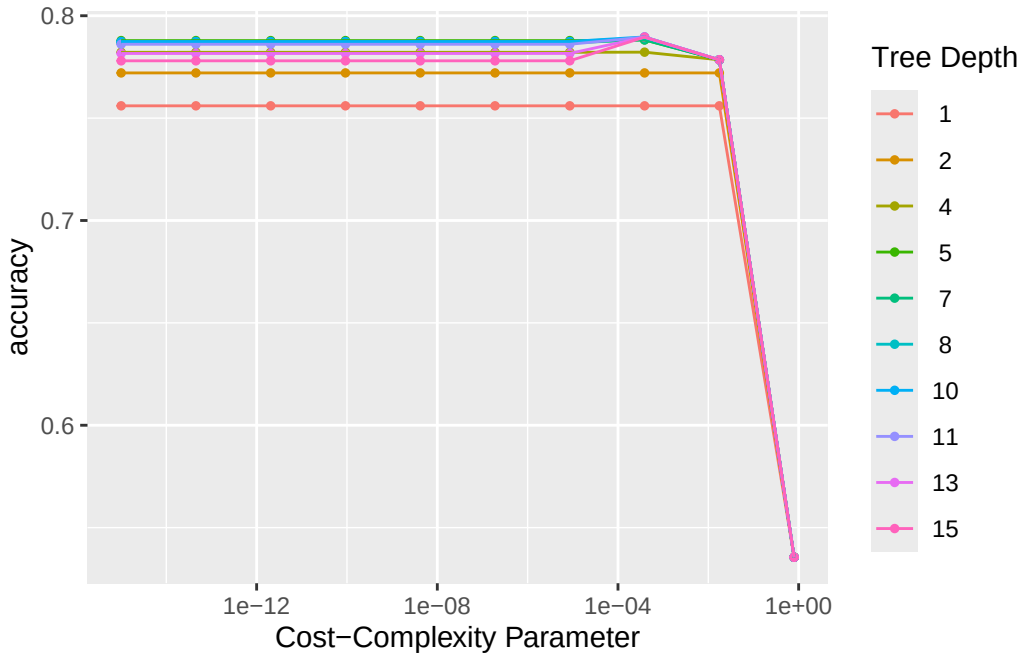
DECISION TREE

Nous avons maintenant le modèle d'arbre de décision, qui utilise la recette : **df_rec**.

- Optimisation des paramètres :

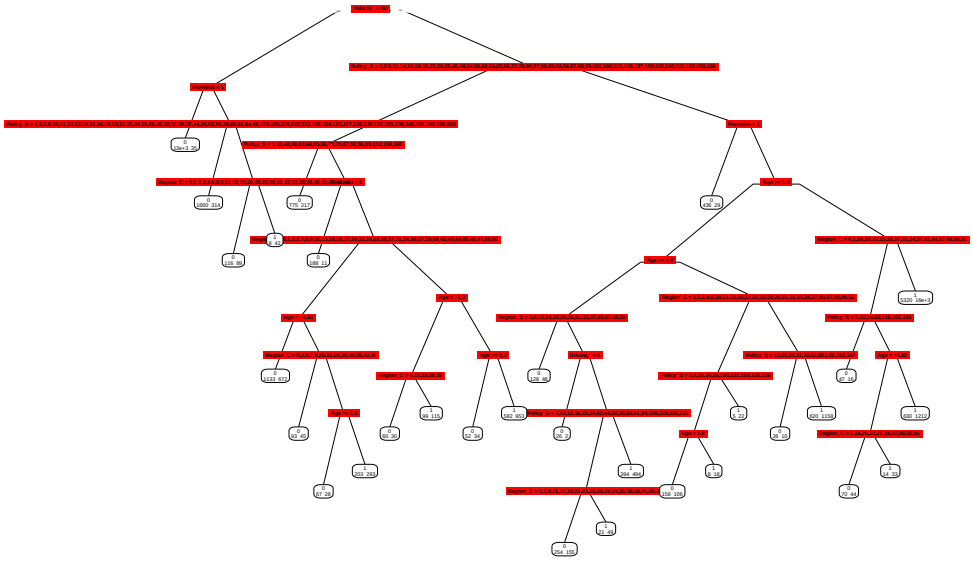
Tout d'abord, pour ce modèle, nous devons optimiser le paramètre de cout de complexité (cost complexity), qui régularise la croissance de l'arbre en pénalisant les structures plus complexes, favorisant ainsi des modèles plus simples et généralisables. Par ailleurs, nous devons également optimiser la profondeur de l'arbre.

Nous choisiront le couple de cost complexity et de tree depth qui maximisent l'accuracy du modèle.

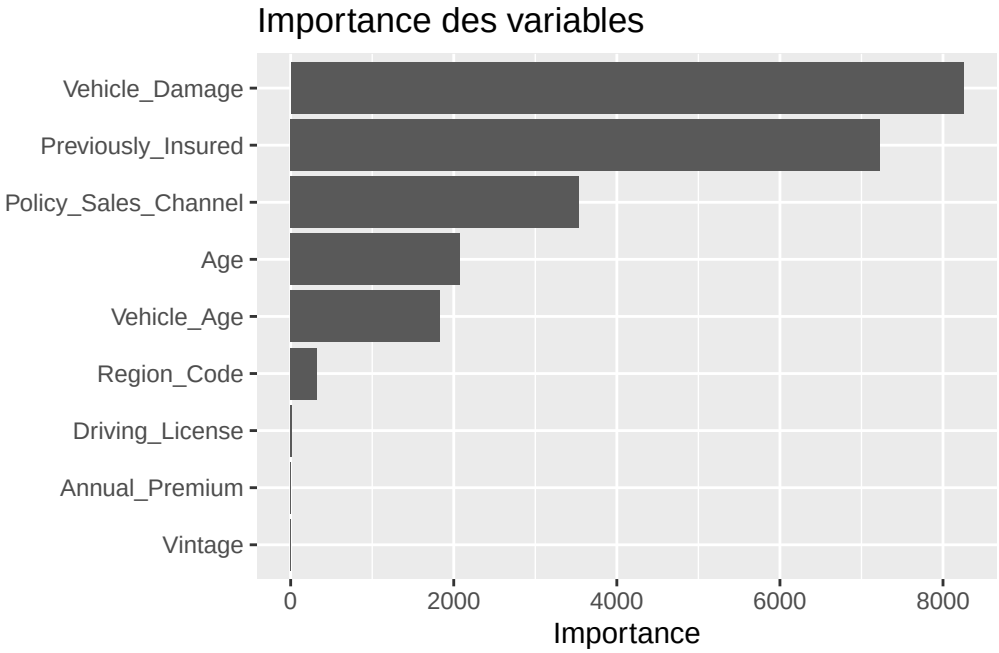


On obtient un paramètre γ optimal de : **0.0003880511** et une profondeur de **10**.

- Visualistaion de l'arbre obtenu :



- Les variables importantes :



Les variables les plus importantes pour réailser les split dans l'abre sont donc : ***Vehicle_Damage*** et ***Previously_Insured***.

- La matrice de confusion :

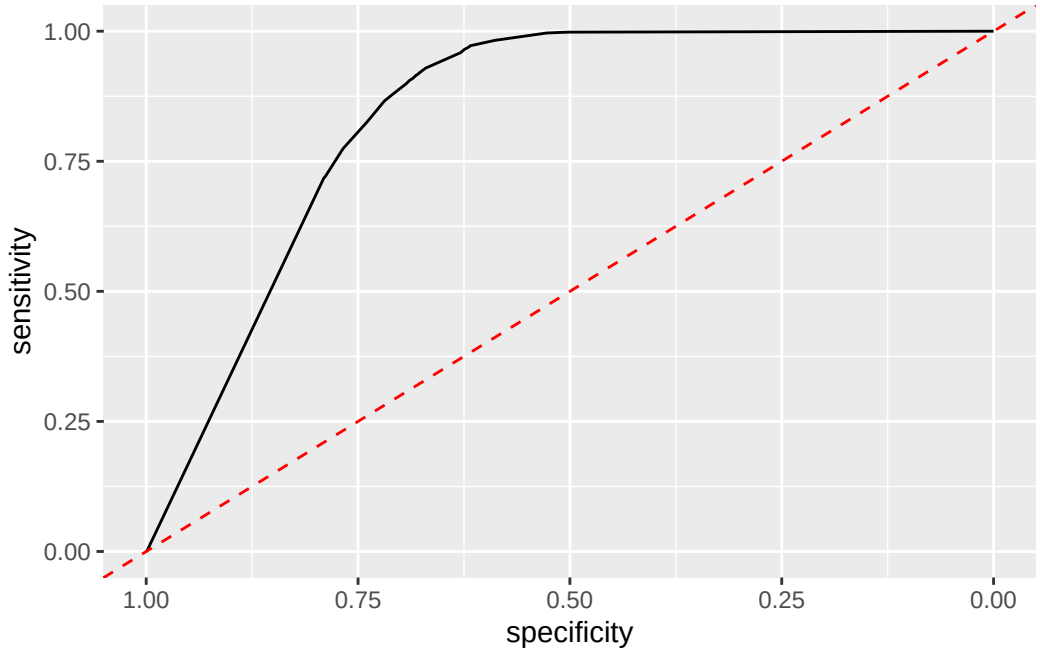
	Prédiction		
Réalité	**N**	**Y**	**Total**
N	8998	4050	**13048**
Y	1076	10240	**11316**
Total	**10074**	**14290**	**24364**

- Les mesures de performance du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	78.96
Erreur globale de classement	21.04
Vrai négatif	89.32
Vrai positif	71.66
Précision	68.96

Les performances de ce modèle sont comparables à celles de la LDA et de la régression logistique, avec une accuracy légèrement supérieure, mais un taux de vrai positif toujours inférieur à celui de la LDA. Dans l'ensemble, ce modèle est donc plutôt moyen aussi.

- La Courbe ROC :



```
Area under the curve: 0.8413
```

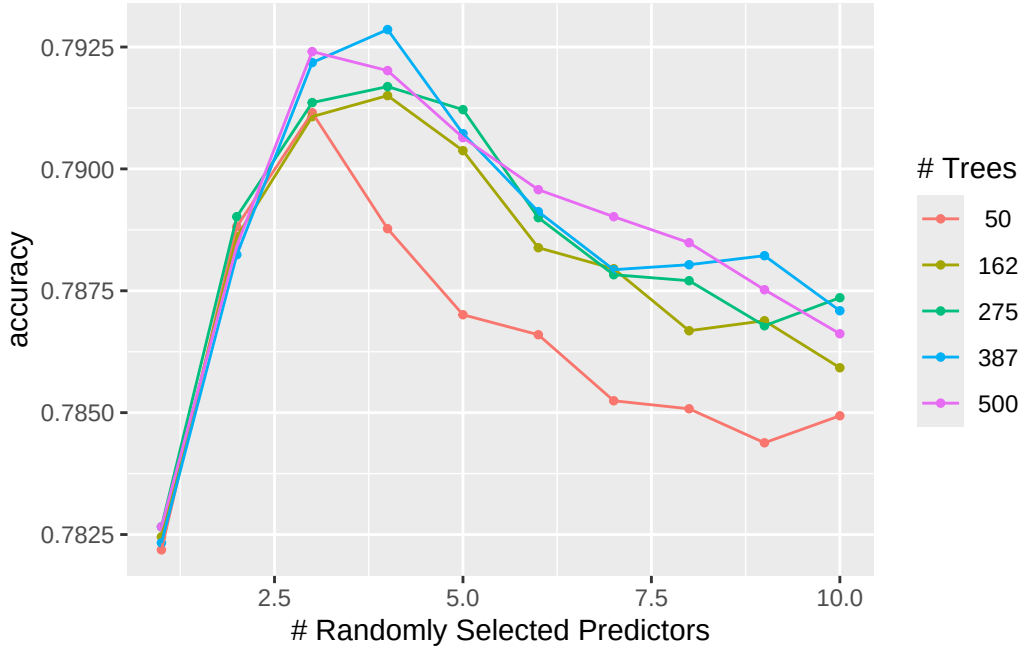
L'aire de la courbe ROC est de 0.8413, le modèle a une capacité de discrimination correcte.

RANDOM FOREST

Notre modèle de random forest, qui se base sur des arbres de décisions, utilise la recette : **df_rec**.

- Optimisation des paramètres :

Pour ce modèle nous avons décider d'optimiser 2 paramètres : le nombre d'arbres (ntrees) et le nombre de variables sélectionnées au hasard à chaque division d'arbre (mtry).



Les meilleurs hyperparamètres sont : **ntrees = 500** & **mtry = 3**.

- La matrice de confusion :

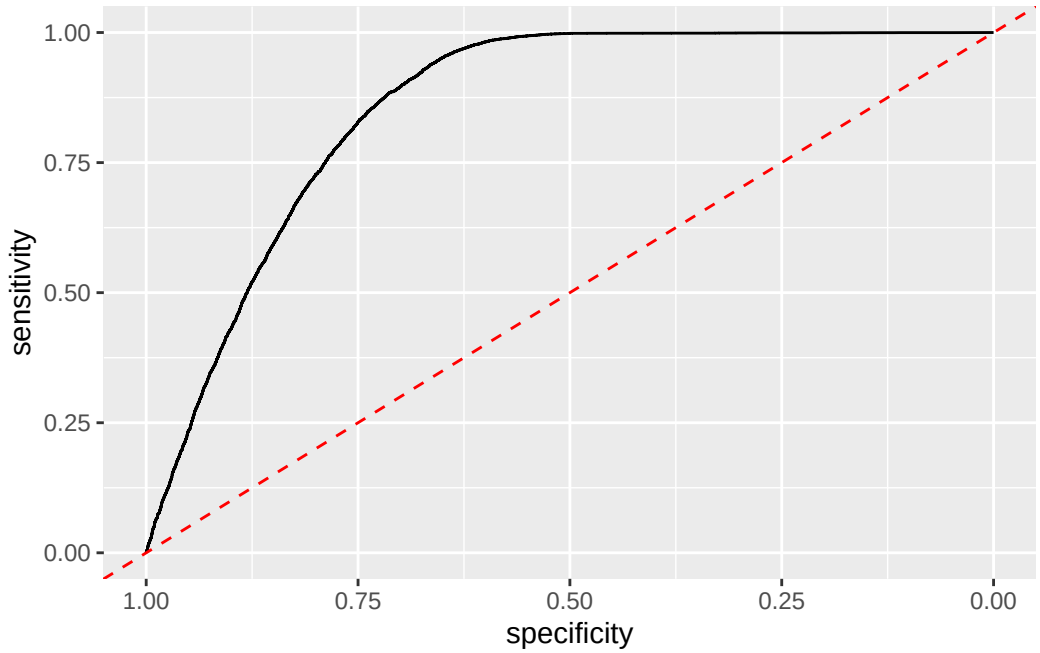
	Prédiction		
Réalité	**N**	**Y**	**Total**
N	9293	3755	**13048**
Y	1299	10017	**11316**
Total	**10592**	**13772**	**24364**

- Les mesures de performance du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	79.26
Erreur globale de classement	20.74
Vrai négatif	87.74
Vrai positif	72.73
Précision	71.22

Notre Random Forest est le modèle avec la meilleure accuracy : 79.26%, et est globalement légèrement meilleur que l'arbre de décision.

- La Courbe ROC :




```
Area under the curve: 0.8577
```

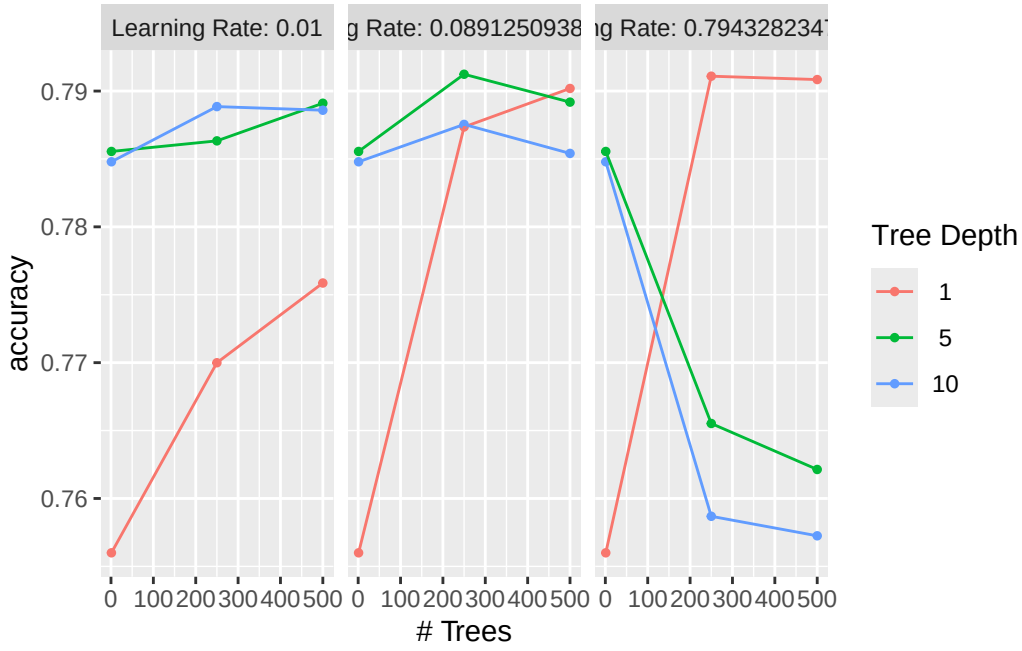
L'aire de la courbe ROC est de 0.8577, le modèle a une capacité de discrimination assez bonne.

BOOSTING

Notre dernier modèle, le BOOSTING, qui se base lui aussi sur des arbres de décisions, utilise la recette : **df_rec_boosting**.

- Optimisation des paramètres :

Pour le Boosting, nous avons décidé d'optimiser 3 paramètres : le nombre d'arbre (trees), la profondeur des arbres (tree depth) et le taux d'apprentissage (learning rate - $ $).



Les meilleurs hyperparamètres sont : **ntrees = 250** , **depth = 5** & λ = **0.1**.

- La matrice de confusion :

	Prédiction		
Réalité	**N**	**Y**	**Total**
N	9113	3935	**13048**
Y	1161	10155	**11316**
Total	**10274**	**14090**	**24364**

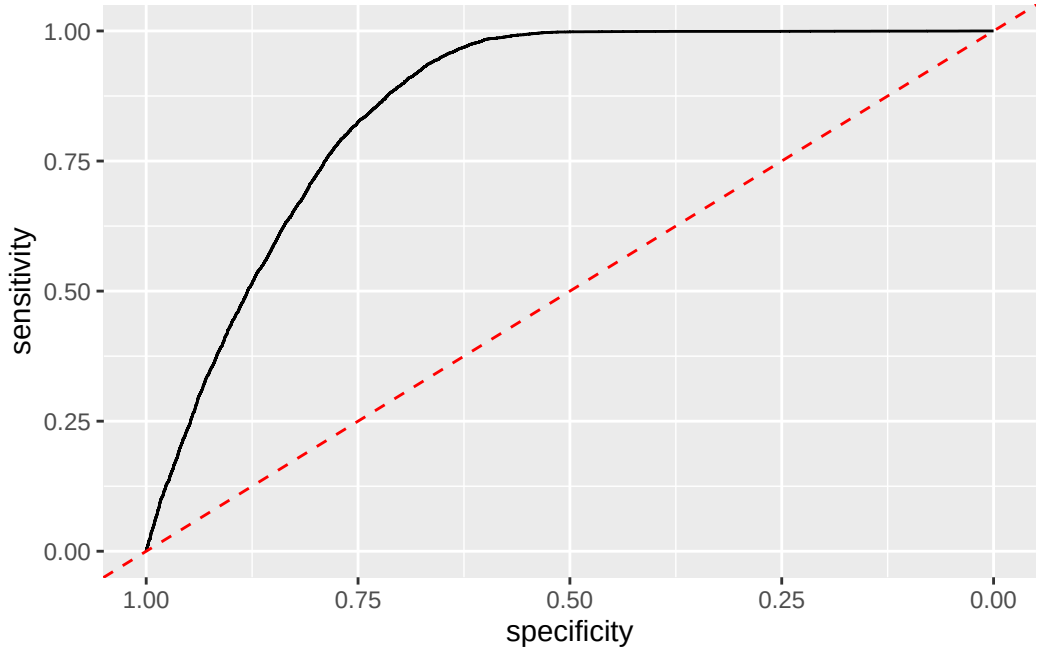
- Les mesures de performance du modèle :

Métrique	Valeur en %
Mesure des performances	
Accuracy	79.08
Erreur globale de classement	20.92

Métrique	Valeur en %
Vrai négatif	88.7
Vrai positif	72.07
Précision	69.84

Le Boosting est un de nos meilleurs modèles, avec une accuracy légèrement inférieure à celle de la Random Forest.

- La Courbe ROC :

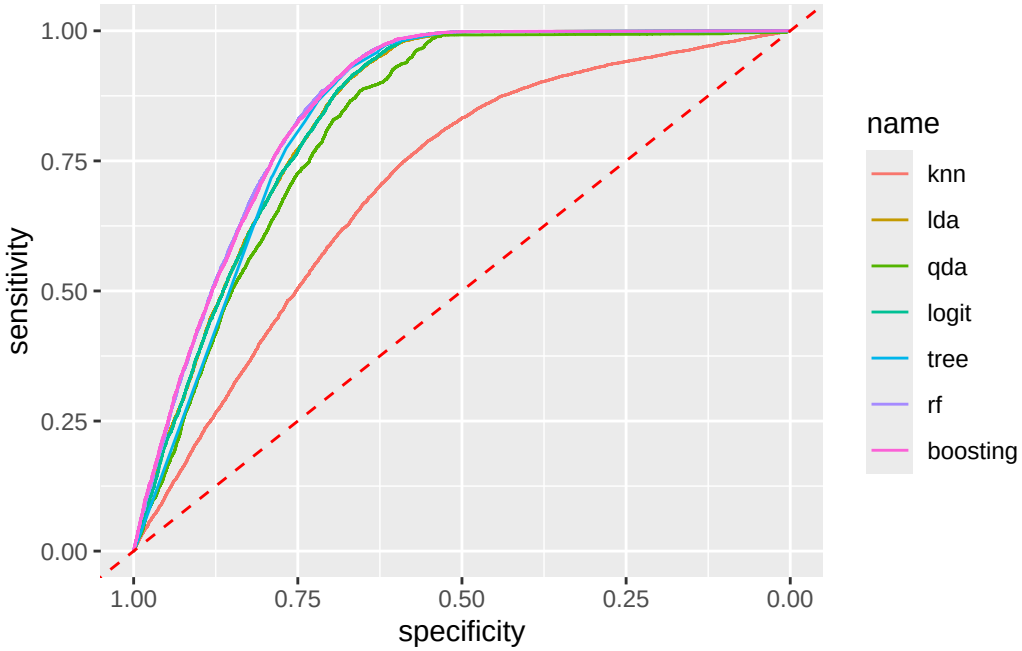



```
Area under the curve: 0.8576
```

L'aire de la courbe roc est de 0.8576, le modèle a une capacité de discrimination assez bonne, quasiment identique à la random forest.

Partie 3 : C'est lequel le best ?

Comparaison Courbe ROC



Les courbes ROC de tous nos modèles restent assez proches les unes des autres, mais deux sont au-dessus, celle du Boosting et de la Random Forest. A l'opposé, celle des knn est nettement moins bonne.

Comparaison des F1-score

Le F1-score est une mesure de précision d'un modèle de classification, qui tient compte à la fois de la précision (capacité du modèle à identifier correctement les exemples positifs) et du rappel (capacité du modèle à identifier tous les exemples positifs).

$F1 = 2 \times \frac{\text{précision} \times \text{rappel}}{\text{précision} + \text{rappel}}$

Modèle	F1-Score
LDA	0.744
QDA	0.754
LOGIT	0.761
KNN	0.652
ARBRE	0.778
RF	0.786
BOOSTING	0.781

Les meilleurs F1-Score sont ceux de : l'Arbre de Décision, de la Random Forest et du Boosting.

Conclusion :

Après avoir construit tous nos modèles, on remarque qu'il y en aucun qui soit particulièrement bon. L'accuracy maximale atteinte par un de nos modèles est de 79.26% pour la random forest, ce qui reste globalement moyen pour un modèle de prédiction.

Pour choisir notre meilleur modèle, nous allons nous baser sur le temps de calcul, les courbes ROC et le F1-Score. Nos meilleurs modèles sont l'**Arbre de Décision**, la **Random Forest** et le **Boosting**.
Le Boosting est un modèle avec une bonne aire sous la courbe ROC, un bon F1-Score, mais il est extrêmement gourmand en ressources de calcul, il prend autant de temps que la random forest, alors qu'il est construit à partir d'une base de données contenant 10 fois moins d'individus.
La Random Forest a aussi un bon F1-Score, une bonne aire sous la courbe ROC, mais le temps de calcul est long.
Ce qui nous mène à l'Arbre de Décision, qui a certes de moins bons résultats que les deux modèles précédents, mais un temps de calcul très rapide, ce qui compense la petite différence de performances.

L'Arbre de Décision est notre meilleur modèle !

Source

- Kobia.fr
- Logo MECEN