



Datamining en finance et assurance

Prévision de la côte de risque en assurance

ZIADI Flora

Mastère Spécialisé

Année 2017-2018

Table des matières

1. Introduction	3
2. Analyse exploratoire et retraitement de la base de données	4
2.1. Présentation de la base de données	4
2.2. Retraitement de la base de données	6
3. Modélisation	8
3.1. Régression linéaire + régularisation L2 (ridge)	8
3.2. K-NN (k-Nearest Neighbors)	9
3.3. SVM (Support Vector Machine)	10
3.4. Arbre de décision	12
3.5. Forêt aléatoire	13
3.6. Réseau de neurones	14
4. Conclusion	16

1. Introduction

L'objet de ce projet est d'appliquer des méthodes de Machine Learning sur une base assurantielle. La principale difficulté rencontrée a été de trouver une base de données intéressante dans le domaine des assurances.

Au final notre choix s'est porté sur des données fournissant les caractéristiques d'automobiles et leur risque assurantiel. Vous trouverez les données aux liens suivants : <https://archive.ics.uci.edu/ml/datasets/automobile>

Cette base de données comprend trois types de données :

- Les caractéristiques d'une voiture ;
- Sa côte de risque d'assurance ;
- Ses pertes d'utilisation normalisées en comparaison des autres voitures.

La côte de de risque d'assurance représente le surplus de risque par rapport au risque associé au prix. Les voitures reçoivent initialement un symbole de facteur de risque associé à leur prix. Ensuite, si le risque est jugé plus grand (ou moins), ce symbole est ajusté en le déplaçant vers le haut (ou le bas) de l'échelle. Les actuaires appellent ce processus «symboling». Une valeur de +3 indique que l'auto est risquée, -3 qu'elle est probablement assez sûre. Ainsi, dans cette étude nous allons essayer de prédire cette côte de risque d'assurance en fonction des caractéristiques de la voiture.

Dans une première partie, nous allons analyser et retraiter la base données. Puis dans une seconde partie nous modélisons cette côte de risque d'assurance à l'aide des algorithmes suivants : une régression linéaire, un KNN, un SVM, un arbre de décision, une forêt aléatoire et des réseaux de neurones.

2. Analyse exploratoire et retraitement de la base de données

2.1. Présentation de la base de données

Notre base de données est constituée de 205 lignes et 26 variables.

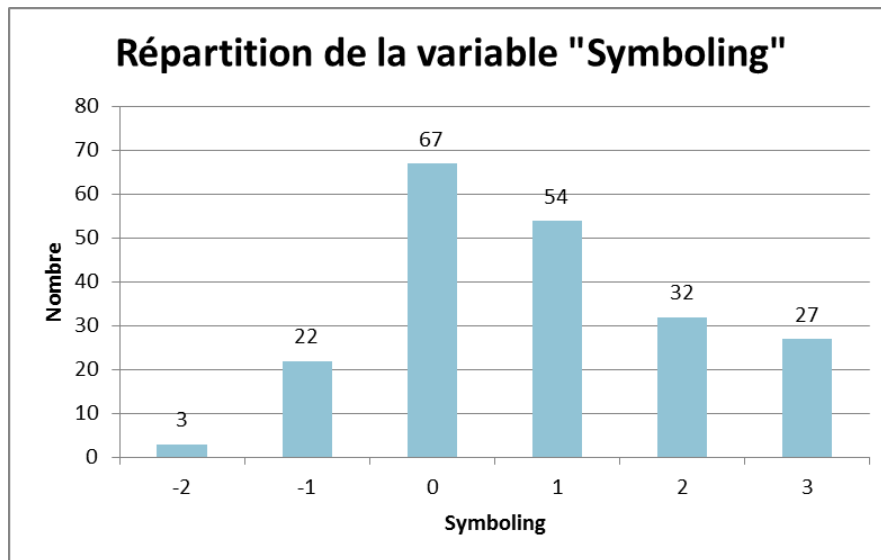
La liste des variables est la suivante :

1. symboling: -3, -2, -1, 0, 1, 2, 3.
2. normalized-losses: continue allant de 65 à 256.
3. make: alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
4. fuel-type: diesel, gas.
5. aspiration: std, turbo.
6. num-of-doors: four, two.
7. body-style: hardtop, wagon, sedan, hatchback, convertible.
8. drive-wheels: 4wd, fwd, rwd.
9. engine-location: front, rear.
10. wheel-base: continue allant 86,6 à 120,9.
11. length: continue allant 141,1 à 208,1.
12. width: continue allant 60,3 à 72,3.
13. height: continue allant de 47,8 à 59,8.
14. curb-weight: continue allant 1488 à 4066.
15. engine-type: dohc, dohcvt, l, ohc, ohcf, ohcv, rotor.
16. num-of-cylinders: eight, five, four, six, three, twelve, two.
17. engine-size: continue allant de 61 à 326.
18. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore: continue allant de 2,54 à 3,94.
20. stroke: continue allant de 2,07 à 4,17.
21. compression-ratio: continue allant de 7 à 23.
22. horsepower: continue allant de 48 à 288.
23. peak-rpm: continue allant de 4150 à 6600.
24. city-mpg: continue allant de 13 à 49.
25. highway-mpg: continue allant de 16 à 54.
26. price: continue allant de 5118 à 45400.

Les 2 premiers champs sont des cibles pertinentes de régression et les 24 autres sont les variables explicatives éligibles. Nous allons tâcher de prédire la première valeur (une note de risque) en fonction des 24 dernières.

Ci-dessous vous trouverez quelques statistiques descriptives de la base de données.

60% des voitures ont une côte de risque d'assurance valant soit 0 soit 1. Seul 12% des voitures sont classées assez sûre (-2 ou -1).



Statistiques descriptives :

```
##      symboling      normalized.losses      make      fuel_type
## Min.   :-2.0000   Min.   : 5118   toyota    : 32   diesel: 20
## 1st Qu.: 0.0000   1st Qu.: 7775   nissan    : 18   gas   :185
## Median : 1.0000   Median :10295   mazda    : 17
## Mean   : 0.8341   Mean   :13185   honda    : 13
## 3rd Qu.: 2.0000   3rd Qu.:16500   mitsubishi: 13
## Max.   : 3.0000   Max.   :45400   subaru    : 12
##                                     (Other) :100
## aspiration num_of_doors      body.style drive.wheels engine.location
## std :168    ?      : 0      convertible: 6 4wd: 9      front:202
## turbo: 37   four:116      hardtop    : 8 fwd:120      rear : 3
##                                     two : 89      hatchback :70 rwd: 76
##                                     sedan      :96
##                                     wagon      :25
##
##
##      wheel.base      length      width      height
## Min.   : 86.60   Min.   :141.1   Min.   :60.30   Min.   :47.80
## 1st Qu.: 94.50   1st Qu.:166.3   1st Qu.:64.10   1st Qu.:52.00
## Median : 97.00   Median :173.2   Median :65.50   Median :54.10
## Mean   : 98.76   Mean   :174.0   Mean   :65.91   Mean   :53.72
## 3rd Qu.:102.40   3rd Qu.:183.1   3rd Qu.:66.90   3rd Qu.:55.50
## Max.   :120.90   Max.   :208.1   Max.   :72.30   Max.   :59.80
##
##
##      curb.weight engine.type num.of.cylinders engine.size fuel.system
## Min.   :1488     dohc : 12   eight : 5      Min.   : 61.0   mpfi :94
## 1st Qu.:2145     dohcv: 1   five  : 11     1st Qu.: 97.0   2bbl :66
## Median :2414     l    : 12   four  :159     Median :120.0   idi  :20
## Mean   :2556     ohc  :148   six   : 24     Mean   :126.9   1bbl :11
## 3rd Qu.:2935     ohcf : 15   three : 1      3rd Qu.:141.0   spdi : 9
## Max.   :4066     ohcv : 13   twelve: 1      Max.   :326.0   4bbl : 3
##                                     rotor: 4   two   : 4      (Other): 2
```

```
##      bore      stroke  compression.ratio  horsepower
##  Min.   :2.540   Min.   :2.070   Min.    : 7.00   Min.    : 48.0
##  1st Qu.:3.150   1st Qu.:3.110   1st Qu. : 8.60   1st Qu. : 70.0
##  Median :3.310   Median :3.290   Median  : 9.00   Median  : 95.0
##  Mean   :3.329   Mean   :3.257   Mean    :10.14   Mean    :104.3
##  3rd Qu.:3.580   3rd Qu.:3.410   3rd Qu. : 9.40   3rd Qu. :116.0
##  Max.   :3.940   Max.   :4.170   Max.    :23.00   Max.    :288.0
##
##      peak.rpm      city.mpg      highway.mpg      price
##  Min.   :4150   Min.   :13.00   Min.   :16.00   Min.   : 5118
##  1st Qu.:4800   1st Qu.:19.00   1st Qu.:25.00   1st Qu. : 7775
##  Median :5200   Median :24.00   Median :30.00   Median :10295
##  Mean   :5124   Mean   :25.22   Mean   :30.75   Mean   :13185
##  3rd Qu.:5500   3rd Qu.:30.00   3rd Qu.:34.00   3rd Qu. :16500
##  Max.   :6600   Max.   :49.00   Max.   :54.00   Max.   :45400
##
```

2.2. Retraitement de la base de données

Nous avons effectué le retraitement suivant sur la base de données :

1. Nettoyer les données en affectant le bon type aux variables qui ont été mal typées ;
2. Remplacer les valeurs manquantes ;
3. Remplacer les variables catégorielles par des variables booléennes "dummies"
4. Normaliser les données pour les algorithmes sensibles aux différences d'amplitude dans les variables
5. Création d'un jeu de données de test qui sera utilisé pour évaluer tous les algorithmes.

Remplacement des valeurs manquantes

Nous avons des valeurs manquantes sur les variables suivantes (notées "?" dans le CSV) :

- 2. normalized-losses: 41
- 6. num-of-doors : 2
- 19. bore : 4
- 20. stroke : 4
- 22. horsepower : 2
- 23. peak-rpm : 2
- 26. price : 4

Au total, la base de données comporte 12 lignes avec des données manquantes.

Une solution serait de supprimer les lignes qui comportent des données manquantes. Néanmoins, comme nous avons une petite base de données, nous avons préféré remplacer les valeurs manquantes.

La fonction `knnImputation()`, fournie par le package `DMwR`, remplace les valeurs manquantes d'une variable quantitative par la médiane des 10 valeurs de cette variable correspondantes aux observations les plus proches de celle qui contient la valeur manquante. Pour une variable

catégorielle, la médiane est remplacée par sa valeur la plus fréquente parmi les 10 observations les plus proches.

Transformation des variables catégorielles

Nous avons remplacé les variables catégorielles par des variables booléennes "dummies" pour les algorithmes traitant des valeurs numériques. C'est par exemple le cas pour le SVM ou encore les Réseaux de neurones.

Normalisation des données

Nous centrons et réduisons chacune des variables numériques, de façon à ce qu'elles aient toutes une moyenne nulle et un écart type de 1. Cette normalisation est utile à certains algorithmes (e.g. les réseaux de neurones) et ne dégrade pas les performances des algorithmes pour lesquels la normalisation n'est pas nécessaire (e.g. les forêts aléatoires).

Séparation apprentissage/test

Pour comparer les différents algorithmes, nous isolons un jeu de test pour évaluer chacun des algorithmes. Nous utilisons ce jeu pour tous les algorithmes afin de ne pas biaiser la comparaison (nous observons de forts écarts de performance selon le tirage de ces jeux pour un algorithme donné). Notre échantillon test est constitué de 55 lignes d'observations.

Les autres données vont être utilisées pour réaliser une validation croisée de type k-fold (isolation de k sous-jeux de taille égale, ces jeux seront tour à tour utilisés comme jeu de validation) visant à identifier les meilleurs paramètres.

3. Modélisation

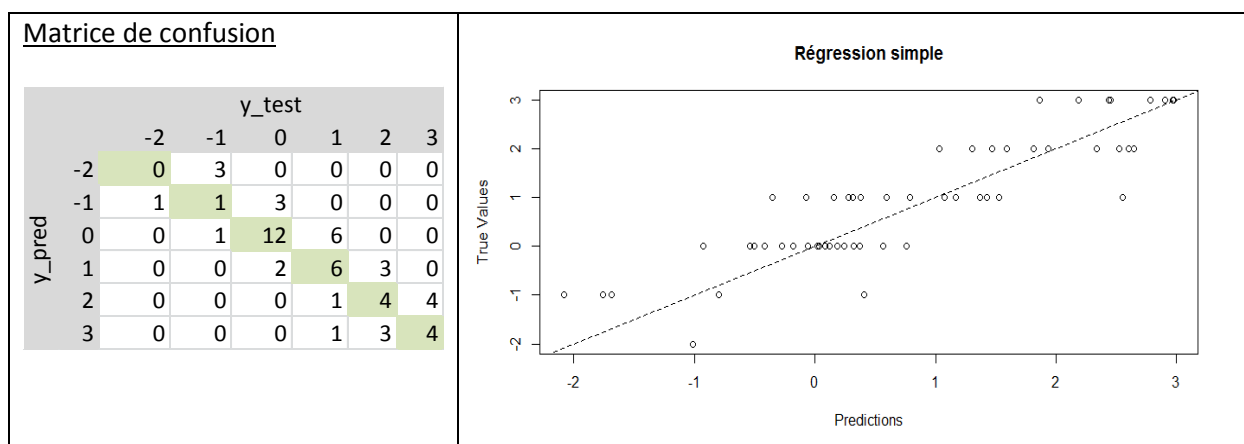
Dans cette étude, nous allons tâcher de prédire la côte de risque d'assurance en fonction des caractéristiques de la voiture.

Pour ce faire nous avons préféré modéliser avec un modèle de régression et non de classification étant donné qu'il existe une notion d'ordre entre les valeurs de la variable à prédire.

Nous évaluerons la performance de nos prédictions par l'erreur quadratique moyenne (MSE) sur un jeu de test similaire pour tous les algorithmes.

3.1. Régression linéaire + régularisation L2 (ridge)

Tout d'abord, nous testons une simple régression. Nous obtenons un **MSE = 0,403**.



Nous observons 27 valeurs prédites bien classées parmi 55. Néanmoins, les autres valeurs prédites ne sont pas loin et se trouvent à + 1 ou - 1 de la valeur réelle.

Par la suite, nous avons voulu comparer cette régression simple avec une régression linéaire régularisée. Nous avons choisi de tester une régularisation de type L2 (Ridge).

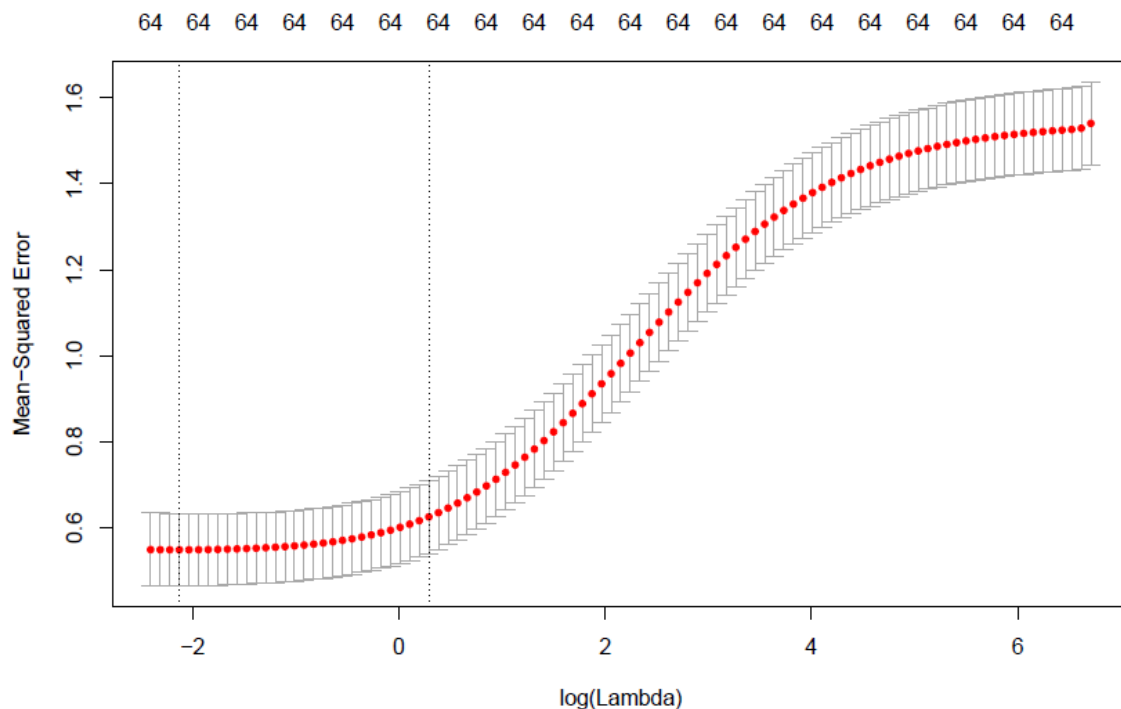
Ridge estimation

In a linear regression model, the Ridge estimator β_R of β_0 is the solution of the following optimization problem,

$$\beta_R = \arg \min_{\beta} \sum_{i=1}^n (Y_i - \beta^T X_i)^2 + \lambda \sum_{j=1}^d \beta_j^2,$$

where β_j are the components of β .

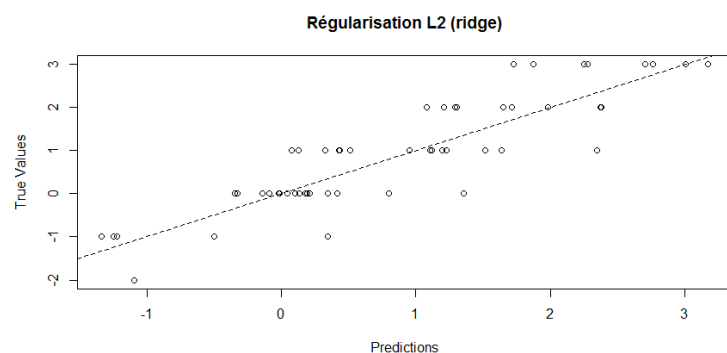
Nous identifions le meilleur paramètre lambda avec une validation croisée (k-fold).



Le meilleur paramètre lambda obtenu est égal à 0,1190638. Avec ce lambda, nous obtenons un **MSE = 0,355**.

Matrice de confusion

y_pred \ y_test						
	-2	-1	0	1	2	3
-2	0	0	0	0	0	0
-1	1	3	0	0	0	0
0	0	2	15	5	0	0
1	0	0	2	6	5	0
2	0	0	0	3	5	4
3	0	0	0	0	0	4



Nous obtenons avec la régression Ridge un meilleur MSE. De plus, on observe 33 valeurs bien classées parmi 55.

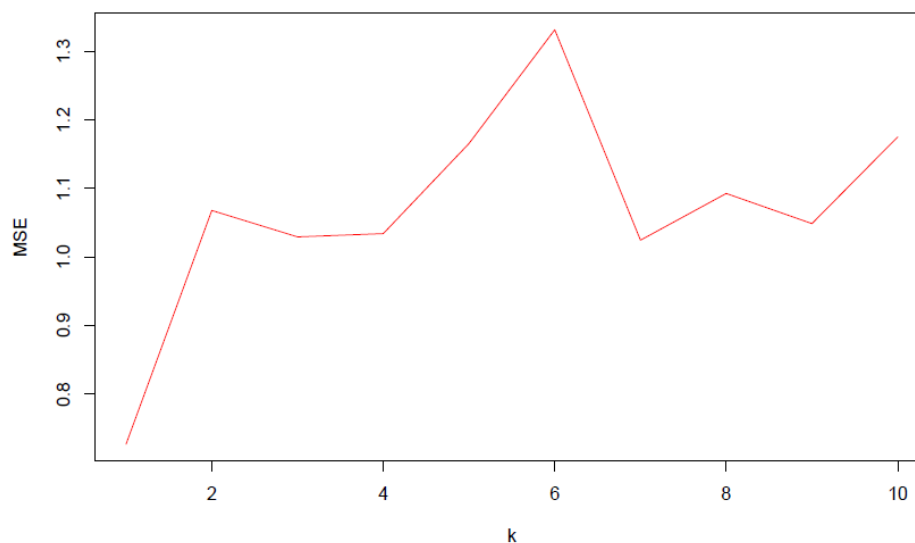
3.2. K-NN (k-Nearest Neighbors)

Nous testons ensuite la méthode des k plus proches voisins (k-NN). Dans la reconnaissance de formes, l'algorithme k-NN est une méthode non-paramétrique utilisée pour la classification et la régression. Dans les deux cas, les données d'entrées sont constituées de k exemples d'entraînement les plus proches dans l'espace des variables explicatives. Dans une classification k-NN, la sortie correspond à l'appartenance à une classe. Un objet est classé par un vote majoritaire de ses voisins, l'objet étant assigné à la classe la plus commune parmi ses k plus proches voisins (k est un entier

positif, typiquement petit). Si $k = 1$, alors l'objet est simplement assigné à la classe de ce voisin le plus proche. Dans la régression k-NN, la sortie correspond à la valeur de propriété de l'objet. Cette valeur est la moyenne des valeurs de ses k plus proches voisins.

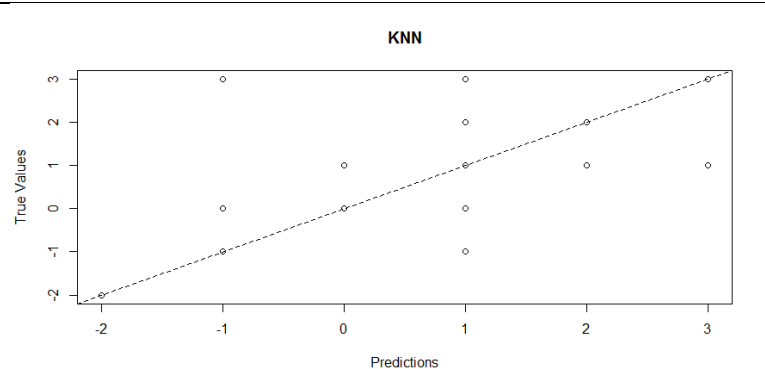
Pour commencer, nous testons un modèle **KNN avec les paramètres par défaut**, nous obtenons un **MSE = 1.218**.

Puis nous essayons de trouver le k optimal en utilisant une validation croisée. Nous testons k entre 1 et 10. Le graphe ci-dessous correspond au MSE des modèles testés en fonction des différents k . Au final, le **MSE minimal vaut 0.727** et est atteint pour $k=1$.



Matrice de confusion

	y_test						
	-2	-1	0	1	2	3	
-2	0	0	0	0	0	0	
-1	1	3	0	0	0	0	
0	0	2	15	5	0	0	
1	0	0	2	6	5	0	
2	0	0	0	3	5	4	
3	0	0	0	0	0	4	

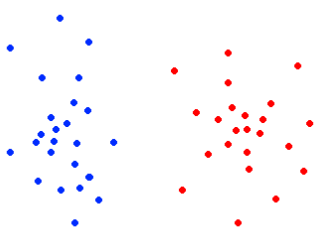
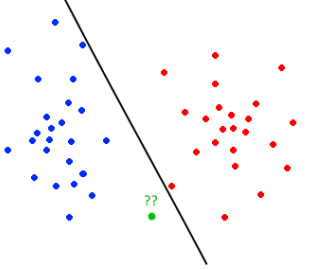
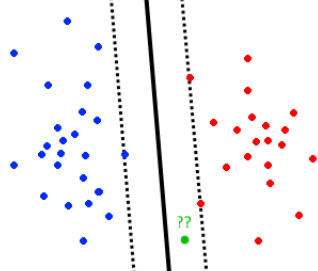


Sur la matrice de confusion, 42 valeurs sont bien classées. On voit que les points sont assez dispersés.

3.3. SVM (Support Vector Machine)

Les SVM (machines à vecteurs support) sont des techniques d'apprentissage supervisées permettant de résoudre des problèmes de classification. Leur utilisation peut aussi être étendue aux problèmes de régression.

Supposons que nous avons le problème de classification suivant :

	<p>Nous cherchons une droite séparatrice, permettant de séparer le plan en deux : nous voulons que tous les points bleus soient situés d'un côté et que tous les points rouges soient situés de l'autre côté.</p>
	<p>Cependant, toutes les droites séparant les points ne conviennent pas forcément. Nous souhaitons que la droite apprise ait de bonnes propriétés de généralisation. Dans l'exemple ci-contre, une nouvelle donnée, le point vert, sera attribué à la classe bleue, alors qu'il est plus proche du nuage de points rouge. La droite de séparation n'est pas la plus adaptée.</p>
	<p>Nous allons donc chercher la droite entourée par la plus grande marge possible. C'est-à-dire, la droite telle que le point le plus proche parmi les données à classer soit le plus loin possible. Avec cette droite (unique), nous voyons que le point vert sera attribué correctement à la classe rouge.</p>

Les SVM sont la généralisation de ce principe en dimension supérieure ou égale à 2 et dans le cas où tous les points ne sont pas forcément linéairement séparables. Dans le cas où les données ne sont pas forcément linéaires (e.g. les données sont séparables par une parabole), on utilise des noyaux (kernels). Les noyaux vont permettre de projeter les données dans un plan en dimension supérieur où les données sont linéairement séparables.

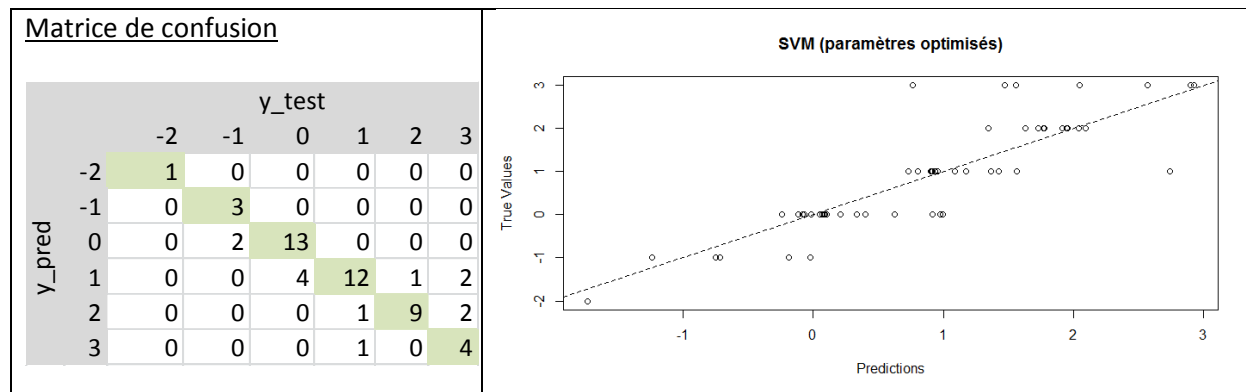
En utilisant les **paramètres par défaut du SVM**, nous obtenons un **MSE = 0.433**.

Puis nous essayons d'optimiser les paramètres du SVM. Nous explorons ici plusieurs noyaux (kernels) ainsi que plusieurs constantes de régularisation (la régularisation d'un SVM n'est pas notée comme celle d'une régression L1 ou L2 : plus le paramètre est grand, moins on pénalise la complexité du modèle).

Nous faisons varier les paramètres suivants :

- les constantes de régularisation suivantes : 1, 16, 16^2 et 16^3 .
- Les noyaux suivants : « polynomial », « radial » et « sigmoid ».

Le modèle qui obtient les meilleurs résultats a pour constante de régularisation 16 et le noyau radial. Son MSE vaut 0.375.



On obtient 42 valeurs prédites bien classées sur 55 pour l'échantillon test.

3.4. Arbre de décision

Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches (les « feuilles » de l'arbre), et sont atteints en fonction de décisions prises à chaque étape.

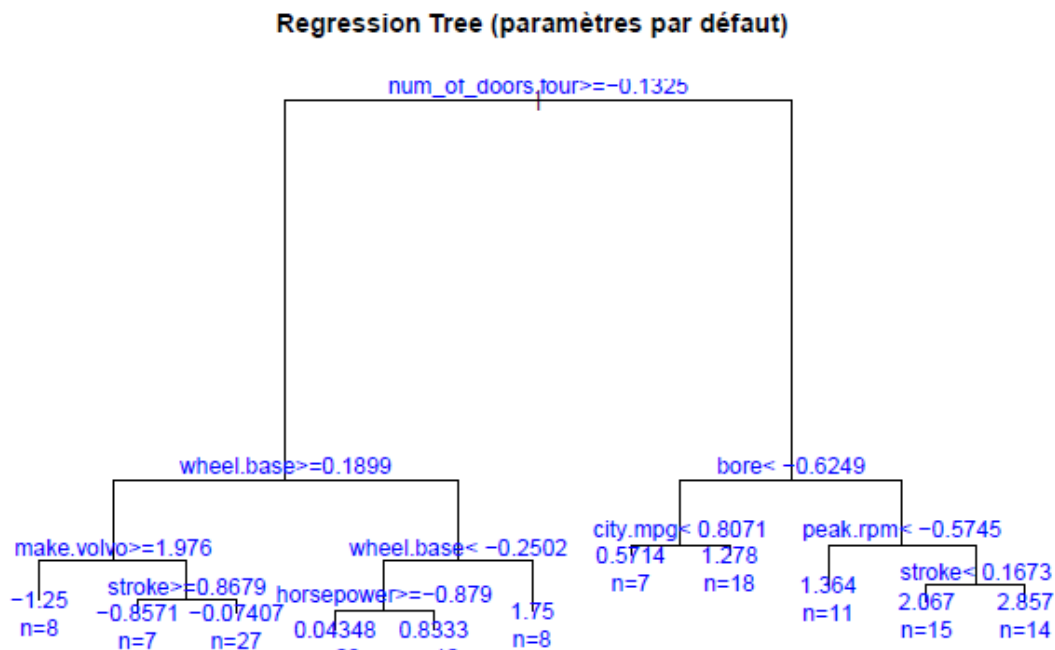
Tout d'abord, nous calculons l'arbre de décision avec les paramètres par défaut. Nous obtenons un **MSE = 0.85**.

Puis nous cherchons à optimiser les paramètres de l'arbre afin d'obtenir un meilleur MSE. Nous faisons varier les paramètres suivants :

- Cp : ce paramètre intervient en pré-élagage lors de la construction de l'arbre, une segmentation est acceptée uniquement si la réduction relative de l'indice de Gini est supérieure à « cp ».
Nous faisons varier ce paramètre entre 0.01 et 0.20 par pas de 0.02.
- Minsplit (par défaut = 20) : le nombre minimal d'observations qui doivent exister dans un nœud pour qu'une tentative de division soit effectuée.
Nous faisons varier ce paramètre entre 10 et 30 par pas de 1.
- Maxdepth (par défaut = 30) : La profondeur maximale de l'arbre.
Nous faisons varier ce paramètre entre 10 et 30 par pas de 1.

Le modèle qui obtient les meilleurs résultats à les paramètres suivants : minsplit=15, maxdepth=10, cp=0.01. Son MSE est le suivant : 0.718.

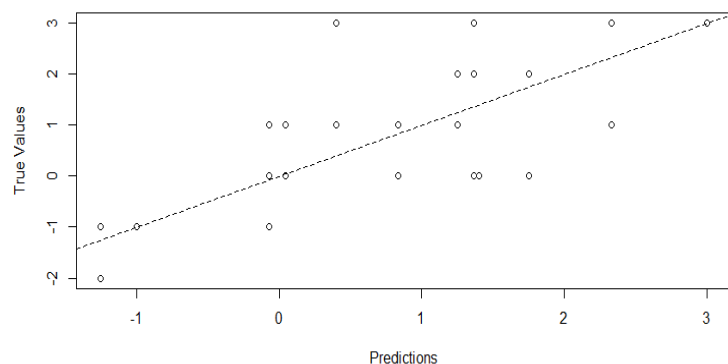
L'arbre est représenté ci-dessous :



Matrice de confusion

y_pred	y_test					
	-2	-1	0	1	2	3
-2	0	0	0	0	0	0
-1	1	3	0	0	0	0
0	0	2	9	7	0	1
1	0	0	5	6	7	1
2	0	0	3	1	3	4
3	0	0	0	0	0	2

Regression Tree (paramètres optimisés)



On observe seulement 23 points bien classés parmi 55 sur la base de test. Les points sont assez dispersés.

3.5. Forêt aléatoire

La forêt aléatoire combine les concepts de sous-espaces aléatoires et de bagging. L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents. L'objectif de la forêt aléatoire est de réduire le risque de sur-apprentissage (en comparaison de l'arbre de décision).

Tout d'abord, nous calculons la forêt aléatoire avec les paramètres par défaut. Nous obtenons un **MSE = 0.29**.

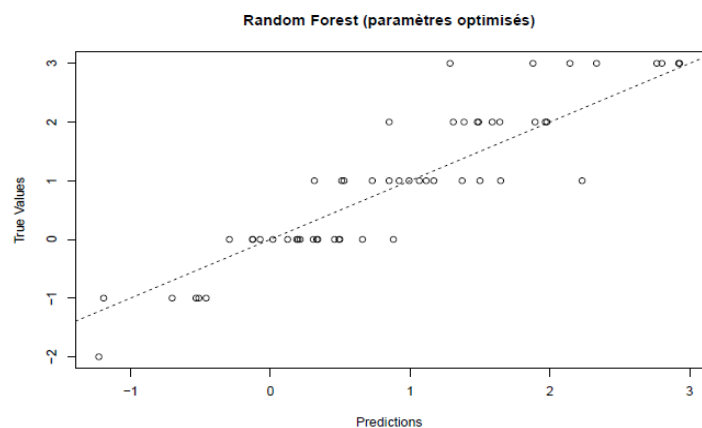
Nous essayons ensuite d'améliorer ce résultat en optimisant les paramètres. Nous souhaitons faire varier les paramètres suivants :

- **ntrree** : Le nombre d'arbres dans la forêt. (par défaut = 500)
Nous faisons varier ce paramètre entre 200 et 800 par pas de 50.
- **mtry** : le nombre de variables testées à chaque division (par défaut = racine carré du nombre de colonne, donc ici 8)
Nous faisons varier ce paramètre entre 5 et 20 par pas de 1.

Le modèle qui obtient les meilleurs résultats à les paramètres suivants : ntrree = 500, mtry = 13.
Son MSE est le suivant : 0.288. Nous n'avons pas réellement amélioré le MSE.

Matrice de confusion

y_pred \ y_test	y_test					
	-2	-1	0	1	2	3
-2	0	0	0	0	0	0
-1	1	4	0	0	0	0
0	0	1	14	1	0	0
1	0	0	2	11	1	1
2	0	0	0	3	9	3
3	0	0	0	0	0	4



On observe seulement 42 valeurs prédites bien classées parmi 55 sur la base de test.

3.6. Réseau de neurones

Un réseau de neurones est l'association d'objets élémentaires, les neurones formels, en un graphe plus ou moins complexe. Il est caractérisé par son type d'architecture et son mode d'apprentissage.

Nous allons ici tester les performances d'un réseau de neurones fully connected avec différentes architectures. Nous utiliserons des réseaux allant jusqu'à 3 couches, avec une fonction logistique (sigmoïde) pour l'activation des couches cachées.

Nous testons la différente architecture présentées dans le tableau ci-dessous :

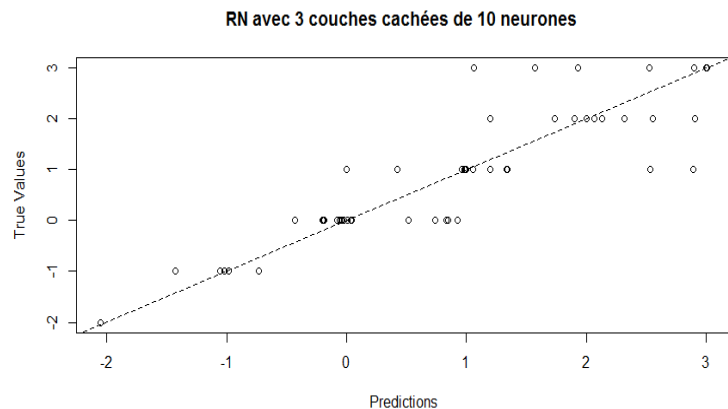
Architecture du Réseaux de Neurones (RN)	MSE
RN avec 1 couche cachée de 10 neurones	1.2
RN avec 1 couche cachée de 20 neurones	0.964
RN avec 1 couche cachée de 40 neurones	1.291
RN avec 2 couches cachées de 10 neurones	0.982
RN avec 2 couches cachées de 20 et 10 neurones	0.509
RN avec 3 couches cachées de 10 neurones	0.381

En l'absence de régularisation sur notre réseau (des techniques existent, comme la régularisation L2 sur chacun des nœuds ou le dropout pour des réseaux profonds), on observe clairement qu'avec le réseau à 1 couche de 40 neurones cachés notre réseau surapprend les données du jeu d'apprentissage et échoue à généraliser au jeu de test. On constate que de meilleurs résultats sont obtenus en augmentant la profondeur qu'en augmentant la taille des couches.

Donc le réseau qui obtient les meilleurs résultats possède 3 couches cachées de 10 neurones. Son MSE est égal 0,381.

Matrice de confusion

	y_test						
	-2	-1	0	1	2	3	
y_pred	-2	1	0	0	0	0	0
	-1	0	5	0	0	0	0
	0	0	0	12	2	0	0
	1	0	0	5	10	2	1
	2	0	0	0	0	6	2
	3	0	0	0	2	2	5



On observe seulement 39 valeurs prédites bien classé parmi 55 sur la base de test.

4. Conclusion

Nous avons essayé de prédire la côte de risque d'assurance d'une voiture. Cette information peut s'avérer très intéressante pour la tarification d'un contrat automobile.

Nous avons choisi de modéliser la variable à expliquer comme une régression plutôt qu'une classification, car il y a une notion d'ordre entre les valeurs de la variable à prédire.

Pour comparer les modèles, nous avons préféré nous baser sur le MSE au lieu de la matrice de classification. En effet, nous avons arrondi les valeurs prédites pour calculer cette matrice et donc nous perdons de l'information. Nous avons simplement affiché la matrice de confusion pour avoir une idée de la répartition entre les valeurs observées et les valeurs prédites.

Nous récapitulons pour chaque modèle les meilleurs résultats obtenus sur la base de test dans le tableau suivant :

Modèle	MSE
Régression linéaire	0.403
Régularisation L2 (ridge)	0.355
KNN	0.727
SVM	0.375
Arbre de décision	0.728
Forêt aléatoire	0.288
Réseau de neurones	0.381

Dans l'ensemble, nous obtenons des résultats assez bons. Lorsqu'on regarde la matrice de confusion, la valeur prédite est en général égale à la valeur observée ou avec un décalage de plus ou moins 1.

Le modèle qui obtient les meilleurs résultats sur notre jeu de données est la forêt aléatoire avec un MSE = 0.288.

Les modèles qui obtiennent les moins bons résultats sont les deux méthodes à base de partition : knn et arbre de décisions. Leur MSE est proche de 0.7.

Nous obtenons des résultats assez similaires pour la régression linéaire avec une régularisation de type L2 (ridge), le SVM et les réseaux de neurones : leur MSE est proche de 0.4.

Les réseaux de neurones donnent généralement de bons résultats, mais les paramètres optimaux sont souvent plus difficiles à déterminer car l'espace des architectures à explorer est considérable.