

Data Mining

Rapport de projet



Loïc Mahier
Demetre Phalavandishvili

Table des matières

Table des matières	1
Introduction	2
Pré-traitements	2
Préparations des données	3
Le logiciel Knime	4
Modèle pour estimer la capacité d'emprunt	4
Construction du modèle	4
Evaluation des algorithmes	8
Modèle pour estimer le chiffre d'affaires prévisionnel annuel	9
Construction du modèle	9
Evaluation des algorithmes	11
Modèle de prédiction des variables catégoriques de secteur d'activité	12
Construction du modèle	12
Evaluation des algorithmes	15
Algorithmes pour le Secteur1	15
Matrice de confusion	15
Courbe Roc	16
Algorithmes pour le Secteur2	17
Matrice de confusion	17
Courbe Roc	18
Algorithmes pour le SecteurParticulier	19
Matrice de confusion	19
Courbe Roc	20
Conclusion	21

Introduction

L'objectif de ce projet est de permettre à Alphaprise, une entreprise B2B¹ qui vend des produits et des services, de traiter elle même ses données pour mieux comprendre les besoins de ses clients. Alphaprise souhaitant donc s'affranchir de son fournisseur de données, nous devons tout d'abord lui permettre d'estimer le chiffre d'affaires annuel prévisionnel et la capacité d'emprunt de chacun de ses clients. Ensuite, nous devons lui permettre de prendre en compte les changements de secteurs de ses clients.

Pour ce faire nous allons dans un premier temps préparer nos données, en les nettoyant. Puis nous allons construire plusieurs modèles pour répondre aux attentes d'Alphaprise : 3 en l'occurrence, un pour déterminer le chiffre d'affaires annuel prévisionnel, un pour déterminer la capacité d'emprunt et un dernier pour déterminer les secteurs d'activité. Nous tâcherons d'utiliser plusieurs algorithmes dans les différents modèles pour pouvoir les comparer et essayer d'avoir les meilleurs prédictions possibles.

Pré-traitements

Ainsi nous avons à notre disposition deux jeux de données. Un premier jeu destiné à l'apprentissage, qui contient 11566 lignes. Chaque lignes correspondant à un client. Le deuxième jeu est lui destiné aux tests, il contient 1000 lignes qui correspondent là encore à des clients. Le premier jeu de données contient 37 attributs que nous allons commencer par présenter.

Voici le jeu de données d'apprentissage :

- *Client* : l'identifiant de l'entreprise cliente
- *Secteur1* : indique si le client relève du secteur d'activité codé 1
- *Secteur2* : indique si le client relève du secteur d'activité codé 2
- *SecteurParticulier* : indique si le client est en fait un particulier
- *NBSalaries* : le nombre de salariés de l'entreprise cliente
- *CapaciteEmprunt* : le montant annuel maximum autorisé pour un prêt d'Alphaprise envers son client
- *PrévisionnelAnnuel* : le chiffre d'affaires annuel prévisionnel engendré par ce client pour Alphaprise
- *P1 à P30* : les chiffres d'affaires engendrés par ce client sur 30 familles de produits sur l'année écoulée

¹ "business to business"

Voici à présent le jeu de données test :

- *NBSalaries* : le nombre de salariés de l'entreprise cliente
- *P1 à P30* : les chiffres d'affaires engendrés par ce client sur 30 familles de produits sur l'année écoulée

Les variables ci-dessus sont fournis. Quant à celles ci-dessous, nous allons devoir les prédire :

- *PredictionCapaciteEmprunt* : le montant annuel maximum autorisé pour un prêt d'Alphaprise envers son client
- *PredictionPrevisionnelAnnuel* : le chiffre d'affaires annuel prévisionnel engendré par ce client pour Alphaprise
- *PredictionSecteur1* : indique si le client relève du secteur d'activité codé 1
- *PredictionSecteur2* : indique si le client relève du secteur d'activité codé 2
- *PredictionSecteurParticulier* : indique si le client est en fait un particulier

Préparations des données

Pour commencer, il est important de souligner que toutes les opérations de pré-traitements doivent être effectuées à la fois sur le jeu de données d'apprentissage et le jeu de données de test (pour les variables partagées). Commençons donc par nettoyer les données dans leur globalité avant de s'intéresser plus particulièrement aux variables utiles à la construction des différents modèles.

Dans un premier temps, nous convertissons le fichier .xlsx en .csv par commodité. Une fois cela fait nous avons donc des points-virgules “;” en guise de séparateur. De même nous remplaçons toutes les virgules “,” par des points “.”. Ensuite on remarque que la variable *PrevisionnelAnnuel* est entre crochet. Nous les supprimons donc pour avoir uniquement une valeur numérique et non plus une “string” (une chaîne de caractère) qui nuirait à son traitement. Cela se fait très simplement à l'aide d'un ctrl+f dans un éditeur de texte lambda. Ce que l'on peut constater ensuite, c'est que pour certaines valeurs de *PrevisionnelAnnuel* qui sont élevées, la notation est en puissance de 10 tel que 12,50e+006 par exemple. Nous devrions donc supprimer cette notation scientifique, cependant le logiciel que nous allons utiliser pour créer le modèle tolère plusieurs notations, nous ne modifierons donc pas cette variable davantage.

Dans un second temps, nous pensons que le fait qu'il manque des valeurs pour les secteurs posera problème dans notre modèle de classification. Nous remplacerons ainsi toutes les valeurs manquantes par des “0” pour le modèle de scoring. Ensuite, nous avons remarqué que la colonne *NbSalaries* contient des champs vides, pour régler ce problème nous remplacerons ces champs vides par la moyenne de cette colonne comme indiqué dans

le cours. Enfin nous pourrions supprimer la variable *Client* puisqu'il s'agit d'une variable artificielle (une clé informatique). Cependant là encore, le logiciel que nous allons utiliser pour créer les différents modèles nous permettra d'exclure des variables du jeu de données d'apprentissage fourni. Donc pas de raison de supprimer les variables pour l'instant.

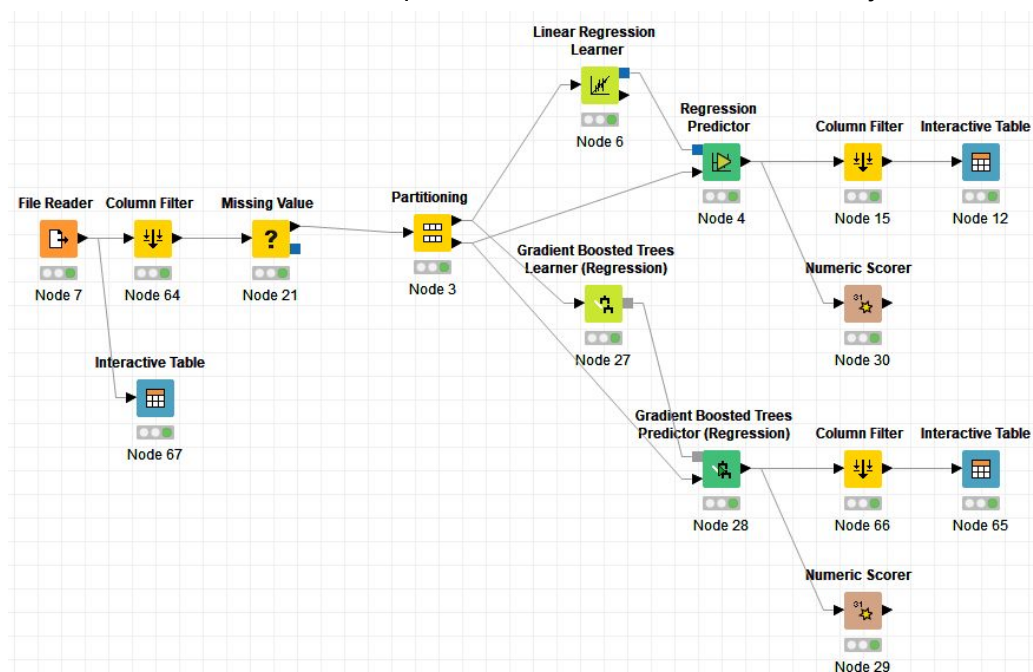
Le logiciel Knime

Nous avons choisi d'utiliser le logiciel KNIME pour construire nos modèles et les évaluer. KNIME est un logiciel libre et open-source que vous pourrez télécharger sur le site internet de KNIME (<https://www.knime.com/>).

Modèle pour estimer la capacité d'emprunt

Construction du modèle

Nous cherchons à présent à construire un premier modèle permettant de prédire la variable *CapaciteEmprunt*. Cette variable est une variable numérique continue, nous allons donc appliquer une régression linéaire pour ce modèle, mais aussi un gradient boosting en régression à titre de comparaison. Voici ci-dessous à quoi cela ressemble sous KNIME. Nous allons détailler les points un par un pour ce modèle précis, mais pour celui prédisant le chiffre d'affaires prévisionnel annuel nous ne rentrerons pas autant dans les détails, puisque ces deux modèles sont similaires et que la démarche sous Knime est toujours la même.



- File Reader

Sous KNIME, chaque opération est effectuée par un “noeud”. Celui-ci, le noeud File Reader permet de lire le fichier de données en entrée. Il n’y a rien de particulier à noter ici.

- Column Filter

Permet de sélectionner les colonnes que nous utilisons pour la prédiction. Nous excluons donc dans ce cas les variables *Client*, *PrevisionnelAnnuel*, *Secteur1*, *Secteur2*, *SecteurParticulier* qui ne permettent pas de prédire la variable *CapaciteEmprunt*.

- Interactive Table

Ce noeud permet d’avoir un rendu visuel de nos données. Dans le cas présent voici ce que nous obtenons :

Row ID	S Client	Secteur1	Secteur2	Secteur...	NbSalar...	Capacit...	Previsio...	D
Row0	C243986	?	1	?	0	8,000	0	0
Row1	C244014	?	?	?	0	10,000	0	0
Row2	C244032	?	1	?	0	20,000	0	0
Row3	C264563	?	?	?	1	15,000	17,500	1,93
Row4	C264587	1	?	?	1	10,000	19,200	123.
Row5	C264593	1	?	?	4	45,000	76,800	284.
Row6	C264616	?	?	?	0	5,000	0	10.5
Row7	C264619	?	?	?	0	8,000	0	56.3
Row8	C264632	?	?	?	1	8,000	17,500	8.22
Row9	C264666	?	?	?	6	5,000	159,600	102.
Row10	C264673	?	1	?	1	5,000	22,400	0
Row11	C264687	?	?	?	0	8,000	0	0
Row12	C264689	?	?	?	1	10,000	26,600	21.5
Row13	C264690	?	?	?	3	10,000	60,000	22.9
Row14	C264696	?	1	?	4	50,000	89,600	65.8
Row15	C264705	?	?	?	0	10,000	0	0
Row16	C264728	?	?	?	0	10,000	0	108.
Row17	C264738	?	?	?	20	30,000	570,000	36,6
Row18	C264741	1	?	?	2	8,000	38,400	27.4

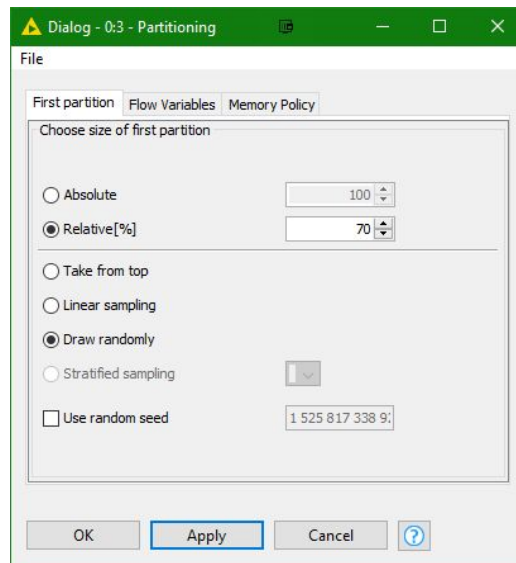
Nous pouvons donc voir dans cet extrait de la table, les variables du fichier de données en entrée. Knime les lis parfaitement. Sont représentée par un points d'interrogation rouge les données manquantes. Cependant, les variables de secteur étant exclus de ce modèle elles ne poseront aucun problème.

- Missing value

Ce noeud, comme son nom l’indique, permet de remplacer des valeurs manquantes. Dans le cas présent, nous remplaçons les valeurs manquantes de la colonnes *NbSalaries* par la moyenne de cette colonne.

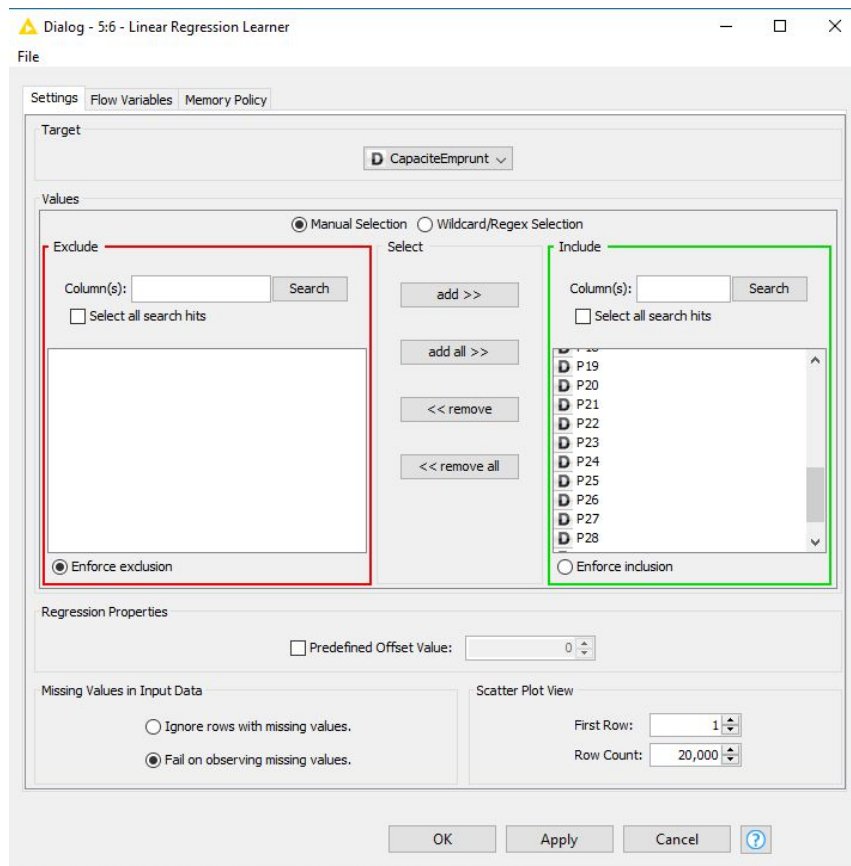
- Partitioning

Ce noeud permet de diviser l'ensemble des données. En l'occurrence, nous avons partitionné en deux nos données avec un ratio de 70% d'apprentissage et 30% de test. Comme vous pouvez le constater plus haut, le noeud Partitioning a deux sorties, une pour chaque jeu créé. Nous avons donc un jeu d'apprentissage de base, que nous avons divisé en deux jeux, un pour l'apprentissage (apprentissage) et un pour les tests (évaluation) ainsi qu'un jeu de test de base sur lequel nous ferons nos prédictions finales (généralisation).



- Linear Regression Learner

Ce noeud est celui où nous paramétrons la régression linéaire dans le but d'effectuer l'apprentissage à partir de notre partition d'apprentissage. Comme vous le constatez ci-dessous, nous conservons les variables *P1* à *P30* et *NbSalaries* pour ce modèle et nous précisons que nous cherchons à prédire la *CapacitéEmprunt*. Aussi, nous cochoons : ignorer les valeurs manquantes. A noter que nous aurions pu exclure ici les variables non-pertinentes (nous l'avons fait avec le noeud Column Filter).



- Regression Predictor

Ce noeud prédit la variable *CapaciteEmprunt* pour notre partition test. Il prend en entrée le modèle de régression obtenue précédemment, ainsi que la partition de test de notre jeu de données.

- Gradient Boosted Trees Learner

Dans un second temps, nous utilisons le gradient boosting en régression, employé à l'aide d'arbre de décision. L'idée est ainsi d'obtenir un ensemble d'arbres de décision pondérés. La configuration se fait de la même manière que pour la régression linéaire, à ceci près que nous n'ignorons pas les valeurs manquantes. Pour le reste, nous indiquons quelle variable nous cherchons à prédire (*CapaciteEmprunt* en l'occurrence), et nous incluons les variables *P1* à *P30* et *NbSalaries*.

- Gradient Boosted Trees Predictor

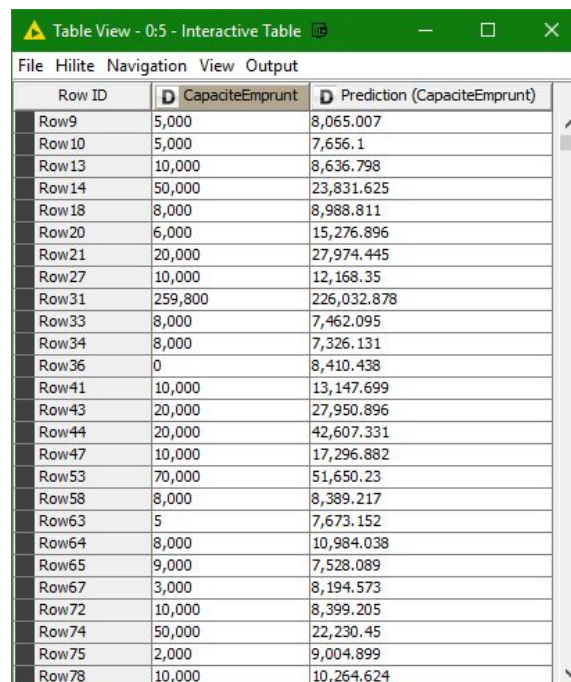
Ce noeud prédit la variable *CapaciteEmprunt* pour notre partition test. Il prend en entrée le modèle obtenu (à savoir un vecteur contenant la pondération de l'ensemble des arbre de décision), ainsi que la partition de test de notre jeu de données.

- Column Filter

Permet de sélectionner les colonnes que nous voulons en sortie, en l'occurrence *CapaciteEmprunt* et *PredictionCapaciteEmprunt*. Nous excluons donc toutes les autres variables.

- Interactive Table

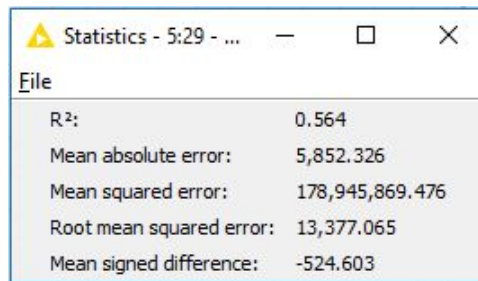
Ce noeud permet de visualiser les prédictions obtenues sous forme de table. Nous discuterons des résultats dans la section suivante : Evaluations du modèle.



Row ID	CapaciteEmprunt	Prediction (CapaciteEmprunt)
Row9	5,000	8,065.007
Row10	5,000	7,656.1
Row13	10,000	8,636.798
Row14	50,000	23,831.625
Row18	8,000	8,988.811
Row20	6,000	15,276.896
Row21	20,000	27,974.445
Row27	10,000	12,168.35
Row31	259,800	226,032.878
Row33	8,000	7,462.095
Row34	8,000	7,326.131
Row36	0	8,410.438
Row41	10,000	13,147.699
Row43	20,000	27,950.896
Row44	20,000	42,607.331
Row47	10,000	17,296.882
Row53	70,000	51,650.23
Row58	8,000	8,389.217
Row63	5	7,673.152
Row64	8,000	10,984.038
Row65	9,000	7,528.089
Row67	3,000	8,194.573
Row72	10,000	8,399.205
Row74	50,000	22,230.45
Row75	2,000	9,004.899
Row78	10,000	10,264.624

Evaluation des algorithmes

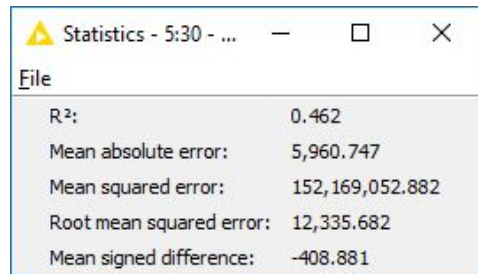
Dans le cas d'une régression linéaire, une mesures de précision est particulièrement intéressante pour évaluer l'algorithme : le mean squared error (MSE) ou erreur quadratique moyenne en français. De même que pour un gradient boosting en régression. Nous effectuons ces mesures sur notre partition de test et nous obtenons cette mesure ainsi que d'autres tel que l'erreur absolue moyenne grâce au noeud Scorer Numeric. Voyez ci-dessous ce que ce noeud produit en sortie :



Statistics - 5:29 - ...

File	
R ² :	0.564
Mean absolute error:	5,852.326
Mean squared error:	178,945,869.476
Root mean squared error:	13,377.065
Mean signed difference:	-524.603

Gradient Boosted Trees Regression



Statistics - 5:30 - ...

File	
R ² :	0.462
Mean absolute error:	5,960.747
Mean squared error:	152,169,052.882
Root mean squared error:	12,335.682
Mean signed difference:	-408.881

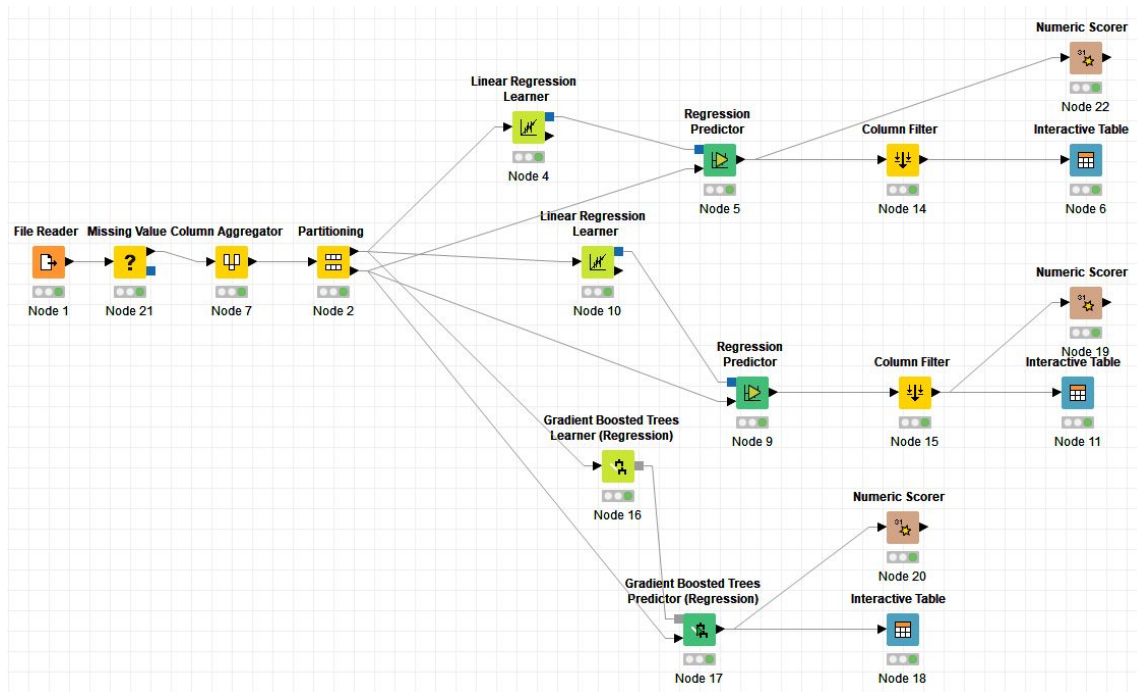
Régression Linéaire

Nous pouvons donc conclure que la régression linéaire produit une meilleur prédiction dans ce cas précis. En effet l'erreur quadratique moyenne est plus faible pour la régression linéaire que pour le gradient boosting, ce qui en fait un meilleur algorithme dans notre cas.

Modèle pour estimer le chiffre d'affaires prévisionnel annuel

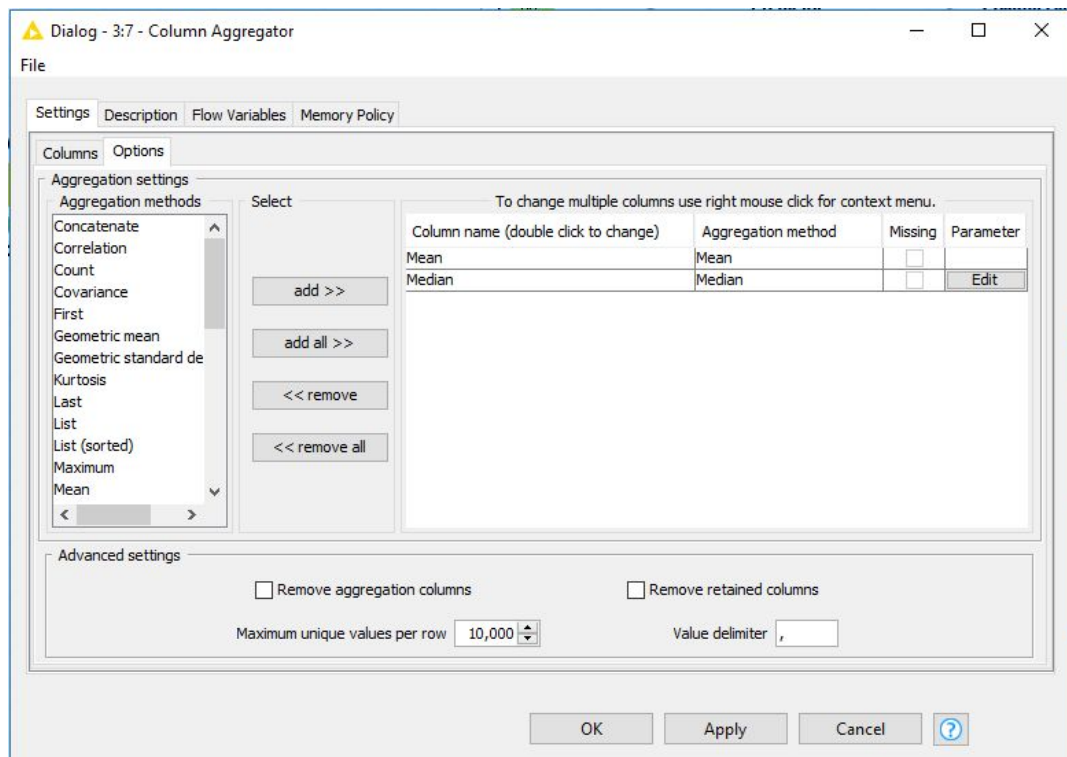
Construction du modèle

Nous cherchons donc à construire un second modèle permettant de prédire la variable *PrevisionnelAnnuel*. Cette variable est une variable numérique continue, nous allons donc encore appliquer une régression linéaire ainsi qu'un gradient boosting en régression dans ce modèle. Nous ne reviendrons pas sur les points déjà expliqués au préalable, seulement sur les nouveaux et ceux qui sont configurés différemment.



- Column Aggregator

Permet d'ajouter des variables à partir du fichier de donnée en entrée, tel que des sommes, des moyennes, etc. En l'occurrence, après plusieurs ajouts de variable, plusieurs tests, nous avons décidé d'ajouter deux variables : la moyenne et la médiane des $P1$ à $P30$. En effet ces deux variables peuvent améliorer les prédictions.



- Partitioning

Là encore, nous partitionnons nos données en entrée avec un ratio de 70% en apprentissage et 30% en test.

- Linear Regression Learner

Nous allons donc en faire deux. La première régression se fait uniquement sur les variables *P1* à *P30* et sur *NbSalaries*, nous excluons tout le reste. Aussi, nous ignorons les variables manquantes.

Pour la deuxième, nous conservons donc les variables *P1* à *P30* et *NbSalaries*. Mais nous ajoutons également les deux variables que nous avons créés, à savoir la moyenne et la médiane. Enfin nous indiquons encore que nous souhaitons ignorer les variables manquantes.

- Regression Predictor

Ce noeud prédit la variable *PrevisionnelAnnuel* pour notre partition test. Il prend en entrée le modèle de régression obtenue précédemment, ainsi que la partition de test de notre jeu de données. Il y a deux noeuds Linear Regression Learner, il faut donc deux noeuds Regression Predictor.

- Gradient Boosted Trees Learner

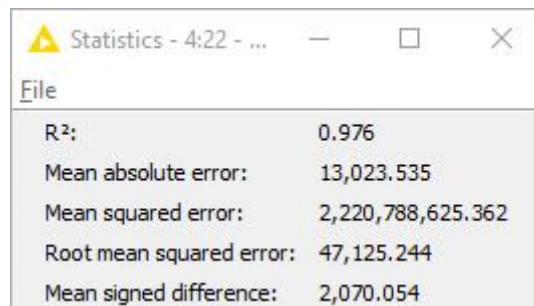
Ensuite, nous utilisons le gradient boosting en régression, employé à l'aide d'arbre de décision. La configuration ce fait de la même manière que pour le premier modèle, nous indiquons quelle variable nous cherchons à prédire (*PrevisionnelAnnuel* en l'occurrence), et nous incluons les variables *P1* à *P30* et *NbSalaries*.

- Gradient Boosted Trees Predictor

Ce noeud prédit la variable *PrevisionnelAnnuel* pour notre partition test. Il prend en entrée le modèle obtenu (à savoir un vecteur contenant la pondération de l'ensemble des arbre de décision), ainsi que la partition de test de notre jeu de données.

Evaluation des algorithmes

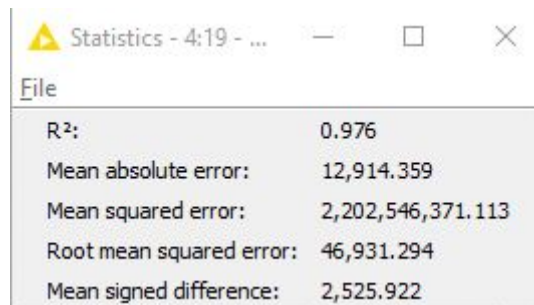
Comme expliqué pour le modèle précédent, la mesure MSE est très intéressante pour évaluer une régression linéaire. Nous l'utilisons aussi pour le gradient boosting. Toujours à l'aide du noeud Scorer Numeric, voici les mesures prises en sortie de notre modèle, sur la partition de test :



Statistics - 4:22 - ...

R ² :	0.976
Mean absolute error:	13,023.535
Mean squared error:	2,220,788,625.362
Root mean squared error:	47,125.244
Mean signed difference:	2,070.054

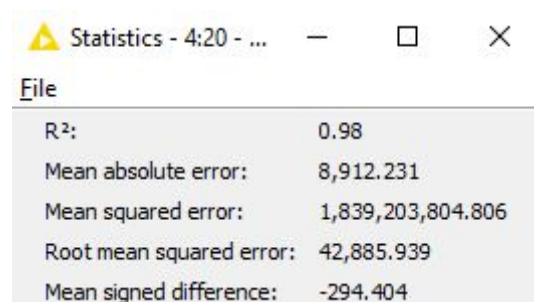
Régression Linéaire sans médiane et moyenne



Statistics - 4:19 - ...

R ² :	0.976
Mean absolute error:	12,914.359
Mean squared error:	2,202,546,371.113
Root mean squared error:	46,931.294
Mean signed difference:	2,525.922

Régression Linéaire avec médiane et moyenne



Statistics - 4:20 - ...

R ² :	0.98
Mean absolute error:	8,912.231
Mean squared error:	1,839,203,804.806
Root mean squared error:	42,885.939
Mean signed difference:	-294.404

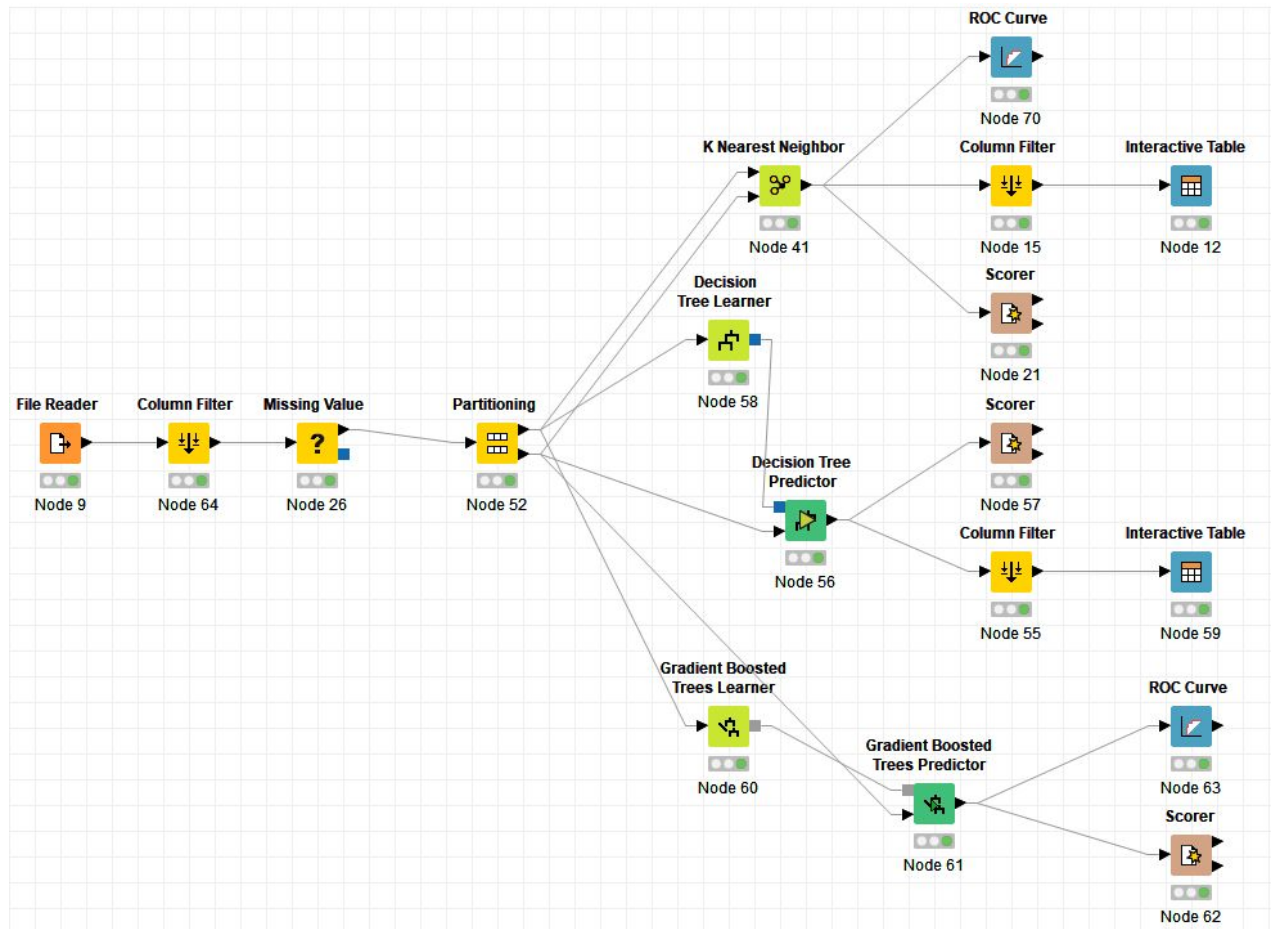
Gradient boosting en régression

Ce que nous constatons est donc que le gradient boosting en régression présente la mesure d'erreur quadratique moyenne la plus faible. En effet c'est ce modèle qui nous fournit la meilleur prédiction. On remarque également que pour la régression linéaire, l'ajout des variables moyenne et médiane apporte une amélioration à cette mesure.

Modèle de prédiction des variables catégoriques de secteur d'activité

Construction du modèle

Dans cette partie, nous cherchons à prédire à quel secteur d'activité appartient le client : *secteur1*, *secteur2* et *secteurParticulier*. Pour cela nous allons utiliser différents algorithmes d'apprentissage : les arbres des décisions, les k le plus proches voisins (k-PPV ou KNN en anglais) et le gradient boosting en classification.



- File Reader

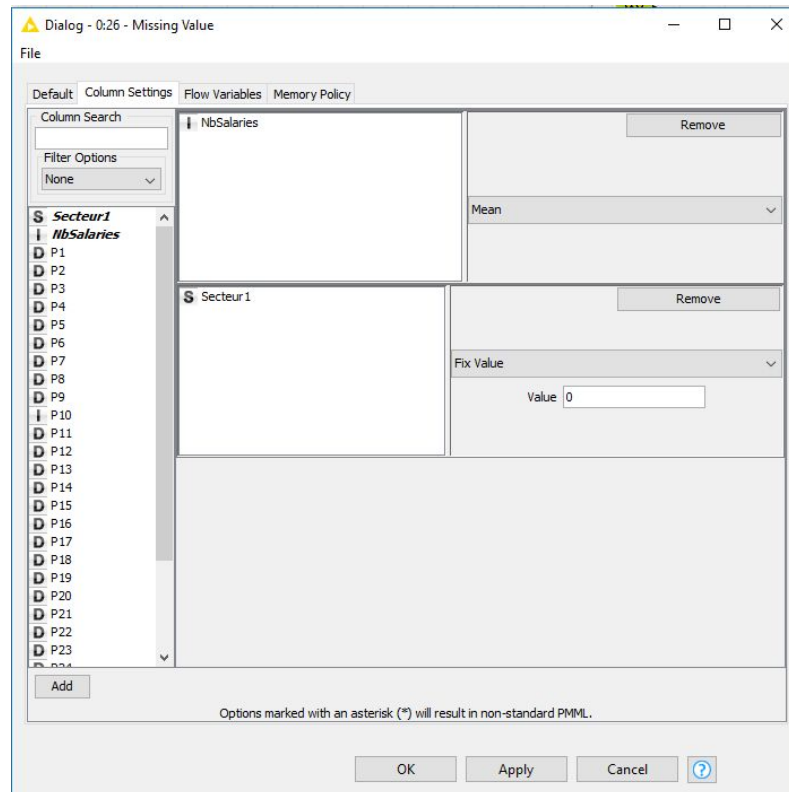
Tout d'abord, dans le fichier de jeu de données en entrée, nous avons besoin de convertir les colonnes correspondant aux secteurs d'activité en chaîne de caractère. De fait, les classifieurs que nous utilisons ont besoin d'avoir ce type de colonne pour la prédiction des classes.

- Column Filter

Ce noeud permet d'enlever les colonnes qui ne sont pas utilisées pour la prédiction des différents secteurs. Les colonnes inutiles sont dans le cas de la prédiction du *Secteur1* : *Secteur2*, *SecteurParticulier*, *CapaciteEmprunt* et *PrevisionAnnuel*. Même démarche pour le *Secteur2* et le *SecteurParticulier*.

- Missing Value

Puis pour éliminer les champs vides dans les colonnes des différents secteurs nous utilisons le noeud Missing Value qui remplit les champs vides de ces colonnes avec la valeur souhaitée, ici en l'occurrence par "0". Nous appliquons cette modification aux trois variables *Secteur1*, *Secteur2* et *SecteurParticulier*.



- Partitioning

Là encore, nous partitionnons nos données en entrée avec un ratio de 70% en apprentissage et 30% en test.

- K-Nearest Neighbor

Pour les k plus proche plus voisin, au premier apprentissage le nombre de voisinage considéré est de 3. De plus dans les options de K-Nearest-Neighbor, on indique qu'en sortie nous voulons la probabilité suivante : $P(\text{secteur1} = 0)$ et $P(\text{secteur1} = 1)$. Cela nous aide à dessiner la courbe ROC pour évaluer nos modèles.

- Decision Tree Learner

Pour l'arbre de décision, nous avons décidé d'utiliser l'indice de Gini comme mesure de qualité en utilisant la formule suivante : $G(S) = 1 - \sum_{i=1}^c p_i^2$ où P_i est la probabilité de choisir un élément de la classe i parmi les éléments de l'ensemble S .

- Decision Tree Predictor

Ce noeud permet de prédire si le client appartient au *Secteur1* pour notre partition test. Il prend en entrée le modèle obtenue à partir du noeud Decision Tree Learner, ainsi que la partition de test de notre jeu de données.

- Gradient Boosted Trees Learner

Comme nous l'avons expliqué, l'idée du Gradient Boosted Trees Learner est ainsi d'obtenir un ensemble d'arbres de décision pondérés. Dans le cas de la prédiction des secteurs nous utilisons Gradient Boosted Trees Learner en classification, ce qui nous permet de prédire les variables catégoriques. Avant de commencer la prédiction nous avons juste indiqué que la hauteur des arbres de décision ainsi obtenu ne doit pas dépasser 4.

- Gradient Boosted Trees Predictor

Ce noeud prédit le secteur pour notre partition test. Il prend en entrée le modèle obtenu (à savoir un vecteur contenant la pondération de l'ensemble des arbre de décision), ainsi que la partition de test de notre jeu de données. De plus dans le paramètre de ce noeud nous indiquons qu'il doit nous rajouter les probabilités suivantes : $P(\text{secteur1} = 0)$ et $P(\text{secteur1} = 1)$.

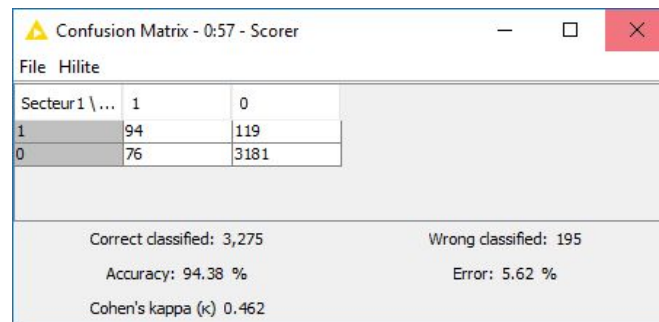
Evaluation des algorithmes

Algorithmes pour le Secteur1

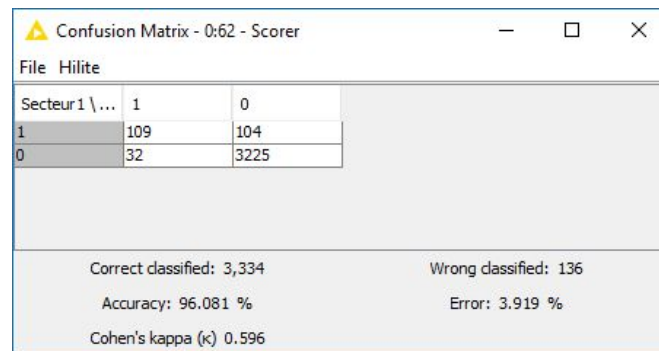
- Matrice de confusion

Confusion Matrix - 0:21 - Scorer		
File	Hilite	
Secteur1 \ ...	1	0
1	118	95
0	51	3206
<div> <div>Correct classified: 3,324</div> <div>Wrong classified: 146</div> <div>Accuracy: 95.793 %</div> <div>Error: 4.207 %</div> <div>Cohen's kappa (κ) 0.596</div> </div>		

KNN - Matrice de confusion



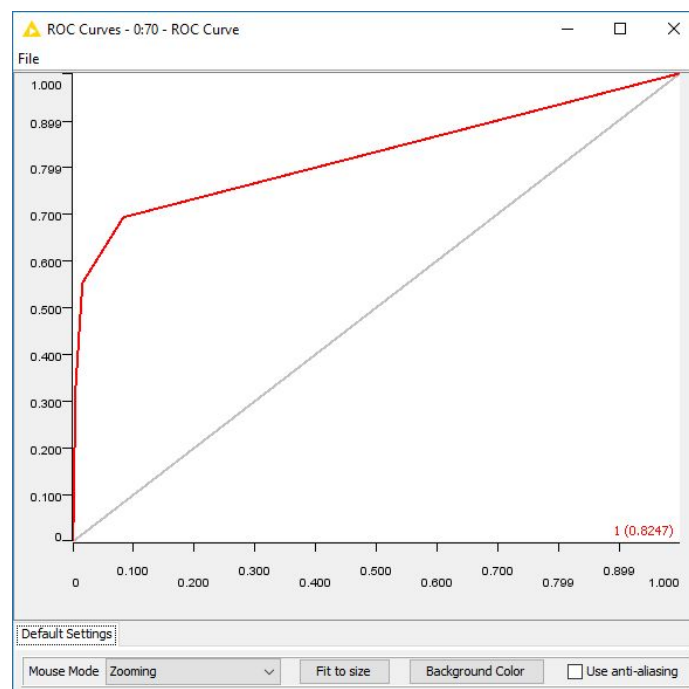
Arbre de décision - Matrice de confusion



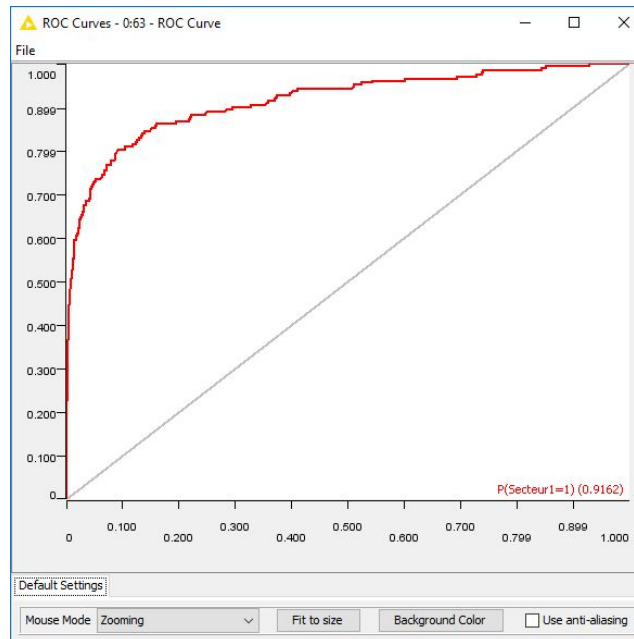
Gradient Boosted Trees - Matrice de confusion

On constate ainsi à partir des matrices de confusion que l'algorithme de gradient boosting en classification nous fournit le taux d'erreur le plus faible et par conséquent la meilleur précision.

- Courbe Roc



KNN - Courbe Roc



Gradient Boosted - Courbe Roc

Nous ne pouvons faire de courbe ROC pour l'algorithme d'arbre de décision. Cependant nous remarquons que pour la gradient boosting et KNN, la courbe ROC montre encore que le gradient boosting est meilleur pour prédire la variable *secteur1*. En effet nous sommes plus proche de la courbe ROC idéal dans le cas du gradient boosting.

Algorithmes pour le Secteur2

- Matrice de confusion

Les matrices de confusion ci-dessous montrent que l'on obtient encore les meilleurs mesures avec l'algorithme du gradient boosting en classification.

Confusion Matrix - 4:21 - Scorer		
File Hilite		
Secteur2 \ ...	1	0
1	387	317
0	284	2482
<div> <div>Correct classified: 2,869</div> <div>Wrong classified: 601</div> <div>Accuracy: 82.68 %</div> <div>Error: 17.32 %</div> <div>Cohen's kappa (κ) 0.455</div> </div>		

KNN - Matrice de Confusion

Confusion Matrix - 4:57 - Scorer

File Hilite

Secteur2 \ ...	1	0
1	372	332
0	303	2463

Correct classified: 2,835 Wrong classified: 635

Accuracy: 81.7 % Error: 18.3 %

Cohen's kappa (κ) 0.425

Arbre de décision - Matrice de Confusion

Confusion Matrix - 4:62 - Scorer

File Hilite

Secteur2 \ ...	1	0
1	399	305
0	162	2604

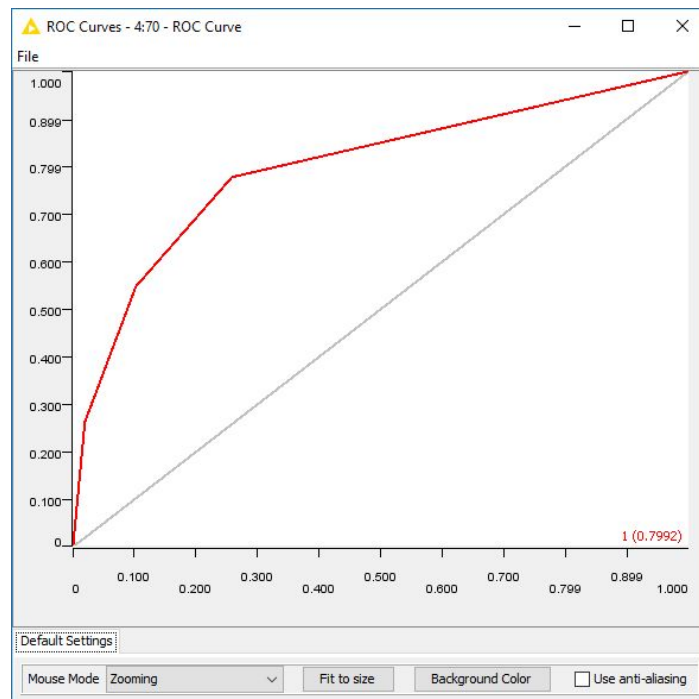
Correct classified: 3,003 Wrong classified: 467

Accuracy: 86.542 % Error: 13.458 %

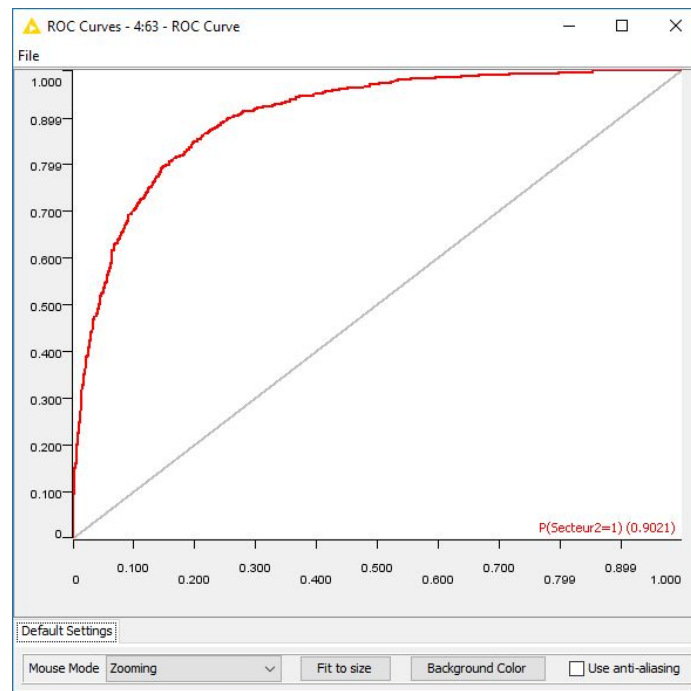
Cohen's kappa (κ) 0.55

Gradient Boosted Trees - Matrice de confusion

- Courbe Roc



KNN - Courbe ROC



Gradient Boosted Trees - Courbe ROC

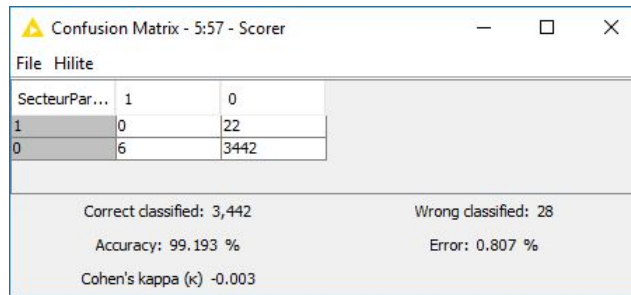
Les courbes ROC obtenues confirme l'observation faite à partir des matrices de confusion. En effet le gradient boosting est là encore bien plus proche de la courbe idéal, ce qui en fait donc le meilleur algorithme pour prédire le *secteur2*.

Algorithmes pour le SecteurParticulier

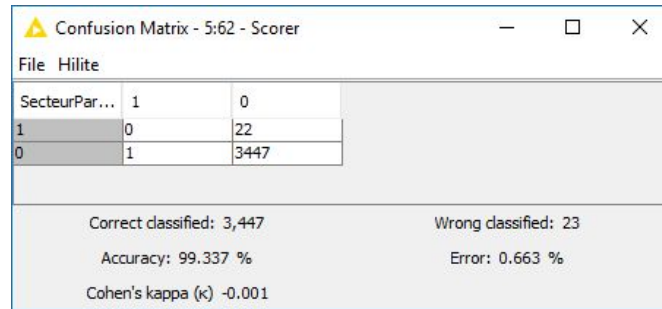
- Matrice de confusion

Confusion Matrix - 5:21 - Scorer		
File Hilite		
SecteurPar...	1	0
1	0	22
0	3	3445
Correct classified: 3,445		
Wrong classified: 25		
Accuracy: 99.28 %		
Error: 0.72 %		
Cohen's kappa (κ) -0.002		

KNN - Matrice de confusion



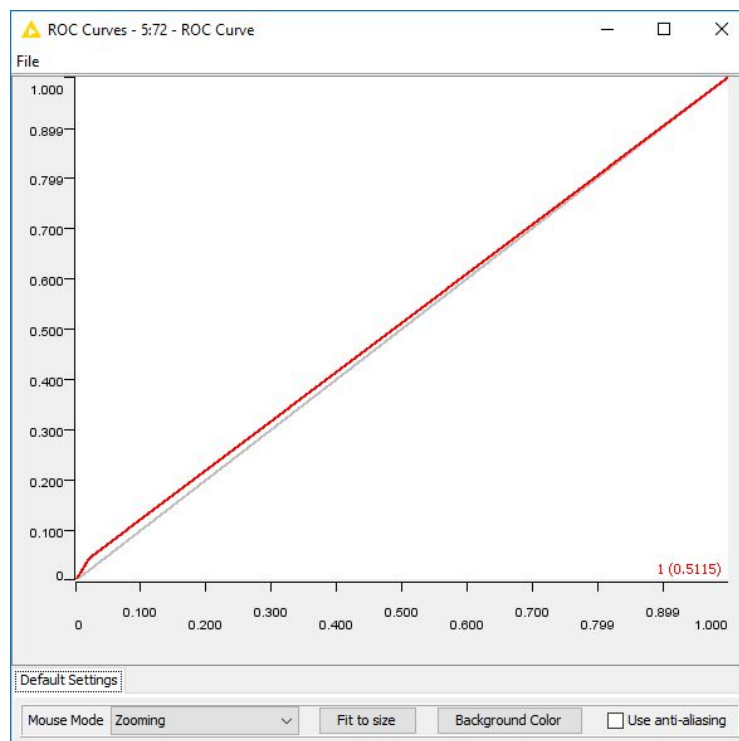
Arbre de décision - Matrice de confusion



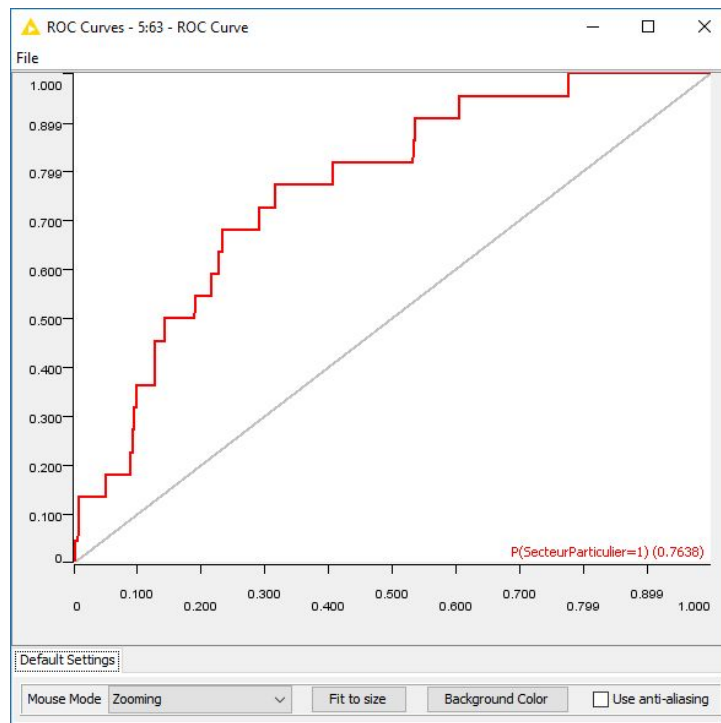
Gradient Boosted Trees - Matrice de confusion

Les matrices de confusion ci-dessus montrent que l'on obtient les meilleurs mesures avec l'algorithme du gradient boosting en classification. Cependant, on observe là que les mesures sont très proches. en effet l'écart entre les trois algorithmes se joue à moins de 0.2 %.

- Courbe Roc



KNN - Courbe ROC



Gradient Boosted Trees - Courbe ROC

Enfin, pour les courbes ROC du *SecteurParticulier* nous constatons que le gradient boosting est bien meilleur que KNN qui à une courbe proche de la courbe aléatoire. Ce qui peut s'expliquer par le fait qu'il y a dans nos données d'apprentissage très peu de "1" et donc beaucoup de "0". KNN ne sait donc pas comment se comporter.

Conclusion

Pour conclure suite aux évaluations des divers algorithmes, nous allons donc utiliser la régression linéaire pour prédire la variable *CapaciteEmprunt*, le gradient boosting en régression pour prédire la variable *PrevisionnelAnnuel* et le gradient boosting en classification pour prédire les différents secteurs d'activité.

Vous trouverez sur le dépôt Madoc l'intégralité de notre projet sous Knime, il est également disponible ainsi que tout le reste de nos travaux sur github à cette adresse (https://github.com/loic44650/Data_Mining_distanciel.git). Vous y trouverez également nos prédictions pour l'entreprise Alphaprise.