

# Rapport de projet Big data analytics

Huylenbroeck Florent  
Palgen Arnaud  
Delfosse Charly

15/05/19

## 1 Introduction

Dans le cadre du cours de Big data analytics, nous avons été amenés à faire un projet, ce projet consiste en une compétition. Celle-ci a pour but à déterminer, sur base d'un jeu de données décrivant certaines caractéristiques de plusieurs personnes, la probabilité qu'une personne place de l'argent dans une banque. Le but étant de minimiser l'erreur de prédiction sur l'ensemble des personnes. Le jeu de données mis à notre disposition nous donne plusieurs informations sur chaque personne : l'âge, le métier, le statut, la présence d'un crédit, prêt de maison, prêt personnel,... Il nous donne aussi plusieurs informations sur le dernier contact avec le client pour cette campagne : le type de communication, le mois, le jour, le nombre de contacts pendant la campagne, le nombre de jours depuis le dernier contact, le nombre de contacts pendant les campagnes précédentes, le résultat de la dernière campagne. Le jeu de données sur lequel on doit s'entraîner nous donne aussi le résultat c'est-à-dire si le client a mis de l'argent à terme dans la banque ou non. Il faut prédire cette probabilité sur le deuxième jeu de données qui, lui, nous fournit les mêmes informations mais pas le résultat.

## 2 Méthodologie

Plusieurs méthodes ont été utilisées pour arriver à déterminer au mieux la probabilité tout en minimisant l'erreur de prédiction.

### 2.1 Régression linéaire simple

La première technique utilisée a été la régression linéaire. En effet, cette méthode fait sens car le but est de prédire une probabilité dans  $[0, 1]$ , la régression linéaire est justement utile pour prédire une réponse sur un ensemble continu. La première approche dans la régression linéaire a été de tester les prédicteurs un à un, donc la droite de régression de chaque prédicteur par rapport à  $y$  (le résultat) a été déterminée. La fonction `lm()` de *R* nous permet justement de faire cette régression, la fonction `summary()` donne des informations importantes sur celle-ci, entre autre la *p-valeur* qui donne une bonne indication sur l'utilité de la droite de régression. En effet, cette valeur donne la probabilité qu'un point des données se trouve plus loin qu'une certaine distance  $t$  de la droite. Plus cette valeur est petite, plus la probabilité qu'un point soit loin de la droite est petite aussi. Ainsi, si on doit prédire le  $y$  d'une nouvelle donnée  $x$ , il y a grande chance pour que ce  $y$  soit situé proche de la droite de régression en  $x$ . Certains des prédicteurs,

comme par exemple le métier, ont été transformés en variables "dummy". Une variable "dummy" n'est autre qu'une variable qui vaut 1 si la personne est un étudiant (par exemple) et 0 si elle ne l'est pas. Cette transformation a été faite dans le but de pouvoir extraire, par exemple, un métier en particulier (exemple "student"). Ainsi, on se retrouve avec beaucoup plus de prédicteurs qu'auparavant. En effet, le prédicteur métier comprend 12 métiers différents et donc 12 variables "dummy" associées. On a ainsi pu dégager les meilleurs prédicteurs :

Prédicteur	p-valeur
age	0.053
campaign	$2.81 * 10^{-8}$
previous	$8.91 * 10^{-6}$
pdays	$< 2 * 10^{-16}$
contact	$< 2 * 10^{-16}$
default(no)	$6.44 * 10^{-12}$
marital(married)	$2.77 * 10^{-6}$
marital(single)	$2.01 * 10^{-8}$
job(student)	$8.68 * 10^{-14}$
job(retired)	$2.92 * 10^{-11}$
edu(university)	$3.77 * 10^{-7}$
poutcome(success)	$< 2 * 10^{-16}$
month(tous sauf dec)	$< 2 * 10^{-16}$
day_of_week(thu)	$2.94 * 10^{-5}$

## 2.2 Cross-validation

Pour tester l'efficacité de nos choix de modèle, une fonction de cross-validation a été faite. Elle découpe le jeu de données "Dtrain" en 10 parties, va s'entraîner sur 9 d'entre elles et va estimer l'erreur sur la dixième. Elle va répéter cette opération 10 fois (en prenant une partie qui n'a pas encore été prise pour tester l'erreur). Ensuite, la fonction va renvoyer la moyenne des erreurs. L'erreur qui est calculée à chaque fois est le *LogLoss* :

$$LogLoss = -\frac{1}{n} \sum_{i=1}^n [y_i \ln(\hat{p}_i) + (1 - y_i) \ln(1 - \hat{p}_i)],$$

où  $n$  est le nombre de données dans la partie de test,  $\hat{p}_i$  est la probabilité trouvée pour l'observation  $i$  de la partie de test et  $y_i$  est la valeur du résultat de l'observation  $i$  de la partie de test. Cette erreur est celle avec laquelle notre modèle sera évalué par Kaggle.

## 2.3 Régression linéaire multiple

Pour obtenir de meilleurs résultats que la régression linéaire, on a ensuite essayé d'associer plusieurs prédicteurs pour encore réduire l'erreur de prédiction. Pour ce faire, on a commencé par choisir 2 prédicteurs ayant une *p-valeur* inférieure à 0,05 et on a calculé la régression multiple de  $y$  en fonction de ces 2 valeurs. Encore une fois, R nous permet de voir les *p-valeurs* de chaque prédicteur dans la régression linéaire multiple. On a ainsi essayé d'associer 2,3,4,... prédicteurs en essayant d'obtenir une régression linéaire multiple avec des *p-valeurs* inférieures à 0,05. Si la *p-valeur* d'un prédicteur était supérieure à 0,05 dans la régression, celui-ci était enlevé et remplacé ou non par un autre. On a aussi utilisé notre sens logique et notre intuition pour trouver les meilleures combinaisons. En effet, certains prédicteurs ont plus de sens pour prédire la réponse que d'autres en connaissance du contexte. On a ainsi pu dégager les meilleures combinaisons de prédicteurs en régression linéaire :

Prédicteurs	Cross-valid.
<i>married + student</i>	0.2704
<i>married + student + contact</i>	0.2679
<i>married + student + contact + university + retired + month</i>	0.2671
<i>single + default(no) + pdays + contact + university + married + retired + student</i>	0.2659
<i>married + student + contact + university + retired + month + default(no)</i>	0.2658

Les 2 dernières combinaisons sont celles qui minimisent l'erreur trouvée par la cross-validation. Ce sont donc les meilleures combinaisons trouvées de régression linéaire.

## 2.4 Régression non-linéaire

Pour davantage améliorer les résultats, on a essayé d'appliquer la régression non-linéaire. La plupart des prédicteurs utilisés sont des variables *dummy*. Il est donc inutile d'essayer de mettre un exposant sur ces termes dans la régression. En effet  $1^n = 1$  et  $0^n = 0$ . Cependant, certains prédicteurs comme l'âge, ne sont pas des variables *dummy*. On a donc fait plusieurs essais pour voir s'il y avait un lien quadratique entre certains prédicteurs et  $y$ . On a ainsi trouvé de meilleures combinaisons de prédicteurs en utilisant la régression non-linéaire :

Prédicteurs	Cross-valid.
$age + age^2$	0.2690
$age + age^2 + age^3 + age^4 + age^5$	0.2685
$student + contact + age + age^2$	0.2668
$university + student + retired + contact + single + age + age^2$	0.2665
$university + student + retired + contact + campaign + pdays + age + age^2$	0.2662
$university + student + retired + contact + default(no) + pdays + month + poutcome + age + age^2$	0.2652
$university + student + retired + contact + default(no) + pdays + age + age^2$	0.2651
$university + student + retired + contact + default(no) + pdays + month + age + age^2 + age^3 + age^4 + age^5$	0.2646

<code>../age+age2.png</code>	<code>../age+age2+age3+age4+age5.png</code>
------------------------------	---

On a remarqué que l'âge était une variable qui donnait de meilleurs résultats dans une régression non-linéaire, en particulier avec un degré de 2 et de 5. Plusieurs essais ont été faits pour trouver d'autres prédicteurs qui avaient un lien quadratique avec  $y$ , mais sans succès. On remarque que la dernière combinaison semble être meilleure que les autres sur base de la *cross-validation*, mais nous verrons dans la section "résultats" qu'en pratique, ce n'est pas le cas.

### 3 Résultats et discussions

Au cours de la compétition, plusieurs essais ont été remis par notre groupe, voici les principaux :

Prédicteurs	Résultats
<i>married + student + contact</i>	0.65189
<i>student+contact+university+success+retired+age+age<sup>2</sup></i>	0.55179
<i>pdays + university + oct + student + campaign + mar + may + apr + thu + retired + contact + age + age<sup>2</sup></i>	0.56933
<i>student + contact + university + retired + age + age<sup>2</sup> + pdays + default(no)</i>	0.54196
<i>student + contact + university + retired + poutcome + month + pdays + age + age<sup>2</sup> + default(no)</i>	0.5612
<i>student+contact+university+retired+age+age<sup>2</sup>+age<sup>3</sup>+age<sup>4</sup>+age<sup>5</sup>+pdays+default(no)</i>	0.54848

On remarque que les meilleures combinaisons trouvées avec la régression non-linéaire sont celles qui donnent le meilleur score. La fonction de *cross-validation* est donc un bon outil pour évaluer l'efficacité d'une prédiction. Cependant, on voit que le prédicteur "mois" semblait efficace dans la combinaison des prédicteurs. Mais en pratique, cette variable ne donne pas de si bons résultats. Celle-ci ne fait pas partie de la meilleure combinaison.

### 4 Conclusion

En conclusion, ce projet a été instructif. Il nous a permis d'appliquer la théorie vue au cours dans le cadre d'une compétition. Au final, nous avons trouvé, grâce à la régression non-linéaire, une combinaison permettant de prédire la probabilité qu'un client dépose de l'argent à long terme dans la banque avec une erreur de prédiction de 0,54196(LogLoss).