UMONS

Faculté des Sciences

Big Data Analytics I

Rapport de projet

Sam Boosko
Rémy Decocq
Dimitri Waelkens

Année Académique 2018-2019
Master en Sciences Informatiques
Faculté des Sciences, Université de Mons

1 Introduction

La jeu de données fourni a été construit lors de campagnes marketing menées par un organisme bancaire, sous la forme d'appels téléphoniques vers de potentiels clients. Pour chaque personne sondée, il est renseigné si oui ou non, à la suite de cet appel, elle a souscrit à un dépôt bancaire à long terme dans ladite banque. Le but de la compétition est de prédire si ce sera le cas pour de nouveau client en se basant sur des variables mesurées identiques. Le jeu d'entrainement contient 30436 observations (dont le résultat "a souscrit" est connu et est repris par la variable y), tandis ce que le jeu de test (dont le y nous est sciemment pas communiqué) est séparé en deux et contient au total $10182 \times 2 = 20364$ mesures.

Les données sont des mesures de variables de deux types : celles relatives au client lui-même (données personnelles) et celles relatives aux potentiels sondages sur le client durant les campagnes. On a donc :

Variables explicatives

`Persos :`

- *age*, (type de) *job*, statut civil *marital*, niveau du milieu d'éducation *edu*
- *default* a une défaillance de crédit, *housing* sous prêt immobilier, *loan* a un prêt personnel

`Campagnes :`

- *contact*, *month*, *day_of_week* : type de communication, mois et jour de la semaine du dernier contact
- *campaign* : nombre de contacts établis durant la campagne correspondant au jeu de données
- *pdays* : nombre de jours passés depuis le dernier contact d'une campagne précédente
- *previous* : nombre de contacts déjà établis avant cette campagne
- *poutcome* : résultat pour ce client suite à la campagne précédente

Variable expliquée

- *y* : le client a ouvert un dépôt à long terme dans l'organisme bancaire

2 Méthodologie

2.1 Analyse des variables et observations

Une variable que l'on peut remettre en question serait *default*. Effectivement, si on regarde la proportion des valeurs pour cette variable catégorique, on a 75.24% de 'no', 24.74% de 'unknown' et moins de 0.01% de 'yes' (3 observations). Une variable qui est presque binaire de la sorte avec une valeur à la sémantique inconnue apporte de la confusion et ne permet pas d'expliciter rationnellement une information. Elle sera donc omise.

Un autre problème peut être mis en avant pour la variable *pdays*. Bien qu'elle soit numérique, une valeur de 999 renseigne 'le client n'a jamais été contacté dans une campagne précédente' (et donc le nombre de jours passés depuis le contact lors de la dernière campagne est indéterminé). On utilise donc une valeur numérique pour signifier quelque chose qui n'est pas quantifiable (au contraire des valeurs pour les clients déjà contactés qui s'étendent de 0 à 11 jours). Un compromis pour avoir une meilleure sémantique serait une variable catégorique, considérant 'recent' si le nombre de jours est < 6, 'late' s'il est ≥ 6 et 'never' pour les valeurs 999. Notons que cela se fait au prix d'une perte d'information, mais le nombre conséquent d'observations pour lesquelles *pdays* vaut 999 (30329) nous y motive.

On constate également des incohérences en considérant la variable $previous$ avec $pdays$. Théoriquement, si $pdays = 999$, alors le client n'a jamais été contacté donc $previous$ (indiquant le nombre de contacts ultérieurs toutes campagnes confondues) doit valoir 0. Or, 1316 observations ne satisfont pas ce prédicat (soit 4% du jeu de données) parmi lesquelles 108 ont un réponse positive pour la variable expliquée y. Ces observations sont importantes, effectivement seulement 2342 parmi les 28094 observations affichent une réponse positive pour y. Pour que notre modèle soit efficace, il faut pouvoir tenir compte de toutes ces précieuses observations.

2.2 Sélection du type de modèle

Comme les prédictions à faire sont sur la variable *y* qui est binaire, on s'intéresse aux méthodes de classification. Trois approches ont été considérées : LDA, la Régression Logistique et les Arbres de décision. La première semble assez peu adaptée : nous ne pouvons pas affirmer rentrer dans ses hypothèses de distribution gaussienne sur les prédicteurs et de séparation claire des classes, qui ne seraient qu'au nombre de 2 (ce qui n'exploite pas la force de LDA). La seconde a été sélectionnée dans un premier temps, dans l'optique d'en améliorer les résultats avec les méthodes de Bagging en implémentant les arbres par la suite si le temps le permettais. Effectivement, la **Régression Logistique** semble toute indiquée pour notre problème : variable expliquée binaire, correspondance avec la fonction d'erreur *LogLoss* utilisée pour évaluer nos résultats, dummification des variables catégoriques aisée.

2.3 Sélection des prédicteurs pertinents

Une fois les premiers modèles construits, il est apparu évident que certaines variables n'y étaient vraiment pas significatives. Afin de sélectionner un ensemble optimal de variables apportant de l'information au modèle parmi les 14 disponibles, les technique de **Stepwise Selection** est employée. En l'opérant dans les deux sens, cela permet de déduire un ensemble fort de variables importantes sur lesquelles reconstruire le modèle de régression logistique.

2.4 Séparation du jeu de données

Comme énoncé à la section 2.1, seulement 2342 observations présentent une réponse positive, ce qui ne constitue que 7% du jeu total. Notre modèle risque de ne pas les prendre suffisamment en compte car noyés dans la masse (ie. le modèle overfit sur des observations majoritairement telles que $y = 0$). Construire le modèle sur un jeu plus balancé pourrait offrir de meilleurs résultats. Une méthode est donc mise en œuvre pour séparer le jeu de données en *training/validation sets* tout en tenant compte de cela, dont la procédure est :

`SepDataset`, paramètres numériques *prop_ok* et *balance*

1. Diviser le set en *obs_no* et *obs_ok* respectivement les observations telles que y=0 et y=1
2. Constituer le **validation set** : prendre $\frac{size(obsok)}{prop}$ observations de *obs_ok* et un même nombre dans *obs_no*
3. Considérer *obs_remaining* l'ensemble des observations n'étant pas dans le **validation set**, prendre parmi elles toutes les observations y=1 qu'on désigne par *obs_remaining_ok*
4. Constituer le **training set** : l'union de *obs_remaining_ok* et *size(obs_remaining_ok)* $\times$ *balance* observations de l'ensemble *obs_remaining* telles que y=0

On obtient alors un validation set de taille limitée contenant $\frac{size(obsok)}{prop} \times 2$ observations, dont la moitié sont telles que y=1 et l'autre y=0. Le training set contient d'une part toutes les observations restantes du set telles que y=1, et d'autre part un nombre d'observations telles que y=0 égal à *balance* $\times$ la taille de cet ensemble. Donc plus *balance* est faible, plus il y aura un nombre équivalent de y=0 et y=1 dans le training set. Le training et validation set sont bien distincts.

2.5 Validation du modèle

Afin de vérifier la cohérence de notre modèle et estimer l'erreur de test, ainsi que d'autres statistiques en découlant, la **Cross-Validation** est la méthode qui est employée. La procédure se présente de cette façon :

`CrossValidate`, paramètres entiers *nbrCV* et *nbrFolds*

1. itérer de 1 à *nbrCV* en générant *nbrFolds* partitions du dataset aléatoirement à chaque fois
2. Dans chaque itération, itérer sur chacun des *nbrFolds* en le considérant comme l'ensemble de test, et appliquant `SepDataSet` sur l'ensemble formé par les *nbrFolds-1* autres
3. Lancer le modèle avec les prédicteurs fixés sur l'ensemble de training retourné par `sepDataSet`, évaluer l'erreur sur l'ensemble de test (le fold courant)
4. Calculer une moyenne des erreurs sur les *nbrFolds* itérations, à partir des *nbrCV* moyennes ainsi calculées, afficher leur distribution et en calculer les moments

3 Résultats et discussion

Ci-dessous le résultat du fitting du modèle de régression logistique en utilisant la combinaison de variables déduites comme les plus explicatives par les procédures de sélection stepwise :

```
[1] "Dataset separation, obs for validation :  234 with y=0 and 234  y=1
[1] "Nbr ok remaining 2108"
[1] "Nbr no remaining 27860"
[1] "Nbr no selected in remaining (remaining ok x 5 ) : 10540"
[1] "TOTAL training set size = ( 2108 + 10540 ) = 12648"
[1] "Real repartition of y in TRAINING selected data"

    0     1
10540  2108
[1] "Real repartition of y in VALIDATING selected data"

  0   1
234 234
```

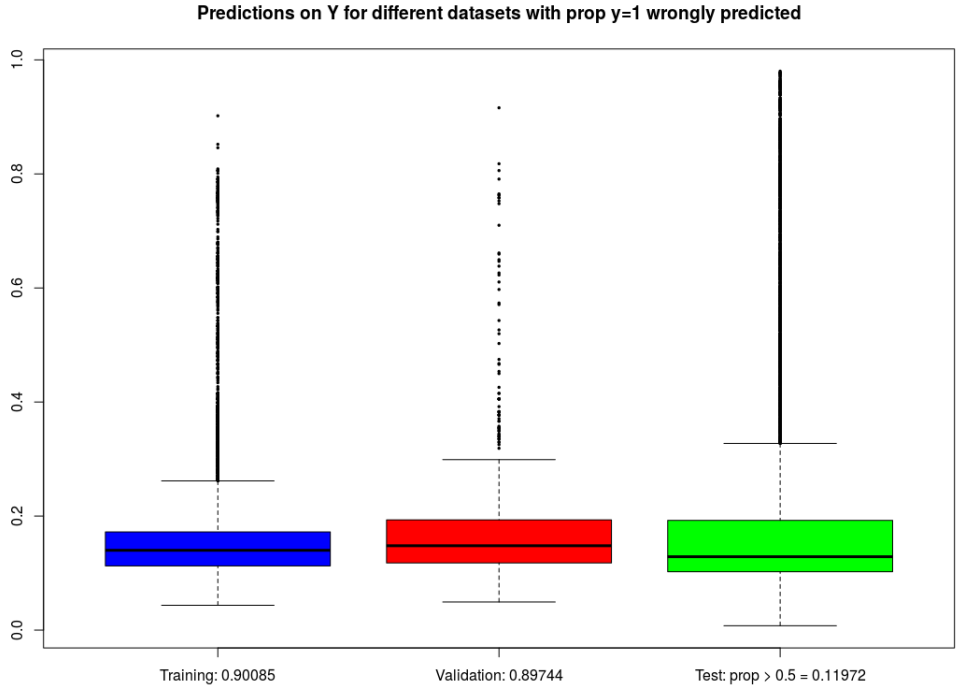
```
Call:
glm(formula = y ~ ., family = "binomial", data = train_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9340  -0.5887  -0.5179  -0.4514   2.3654

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       1.551011   0.456677   3.396 0.000683 ***
jobmiddle        -0.514759   0.105318  -4.888 1.02e-06 ***
jobpoor          -0.651029   0.104614  -6.223 4.87e-10 ***
maritalmarried    0.023336   0.080915   0.288 0.773041
maritalsingle     0.116365   0.087597   1.328 0.184040
maritalunknown    0.582807   0.554398   1.051 0.293147
contacttelephone -0.187712   0.111256  -1.687 0.091562 .
monthaug         -1.183384   0.093546 -12.650  < 2e-16 ***
monthdec         -0.398266   1.127808  -0.353 0.723989
monthjul         -0.929720   0.085931 -10.819  < 2e-16 ***
monthjun         -1.108754   0.143163  -7.745 9.58e-15 ***
monthmar          1.124213   0.170766   6.583 4.60e-11 ***
monthmay         -1.318953   0.137164  -9.616  < 2e-16 ***
monthnov         -1.071419   0.097088 -11.036  < 2e-16 ***
monthoct          1.923758   0.354634   5.425 5.81e-08 ***
day_of_weekmon   -0.167787   0.081381  -2.062 0.039231 *
day_of_weekthu    0.071827   0.077139   0.931 0.351783
day_of_weektue   -0.045471   0.082392  -0.552 0.581025
day_of_weekwed    0.112413   0.079096   1.421 0.155253
campaign         -0.027971   0.009863  -2.836 0.004568 **
pdaysnever       -1.579703   0.432468  -3.653 0.000259 ***
pdaysrecent       0.717116   0.615211   1.166 0.243760
previous         -0.489936   0.115861  -4.229 2.35e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 11397  on 12647  degrees of freedom
Residual deviance: 10639  on 12625  degrees of freedom
AIC: 10685
```

```
             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         12647      11397
job           2    83.49     12645      11314 < 2.2e-16 ***
marital       3    18.01     12642      11296 0.0004371 ***
contact       1   108.98     12641      11187 < 2.2e-16 ***
month         8   475.58     12633      10711 < 2.2e-16 ***
day_of_week   4    17.93     12629      10693 0.0012730 **
campaign      1     9.31     12628      10684 0.0022823 **
pdays         2    25.95     12626      10658 2.314e-06 ***
previous      1    19.32     12625      10639 1.107e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

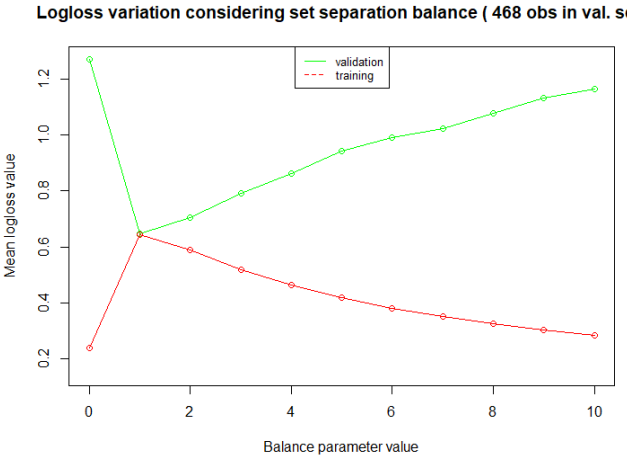
Ce modèle fait l'hypothèse que la relation avec la variable expliquée est linéaire. Pour se faire une idée de ses performances, on peut calculer les prédictions qu'il fournit sur d'une part les données du training set (dont le résultat sera donc sujet à de l'overfitting) et d'autre part sur le validation set (qui contient donc un nombre équivalent d'observations telles que y=0 et y=1). On en sort les mesures suivantes, avec matrices de confusion (les colonnes étant les vraies valeurs et les lignes les valeurs prédites) :

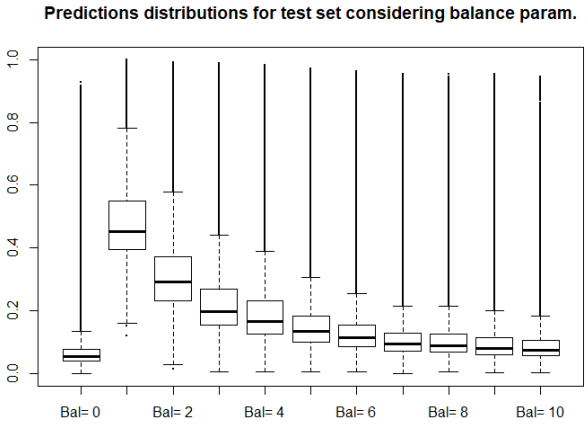
```
[1] "Predictions on TRAINING give logloss : 0.42057297823954 and prop y wrongly predicted as 0 : 0.900853889943074"
      0    1
0 10442 1899
1    98  209
[1] "Predictions on VALIDATION give logloss : 0.909976484720567 and prop y wrongly predicted as 0 : 0.897435897435897"
    0   1
0 231 210
1   3  24
```

Cette classification a été effectuée considérant un seuil de 0.5 (au dessus de cette probabilité prédite, y=1). Afin d'expliciter les distributions des probabilités qui sont les prédictions sur différents sets (y compris l'ensemble réel de test hors validation), des boxplots sont utilisés. Comme les faux négatifs semblent visiblement être le point faible du modèle, on en renseigne également la proportion pour chaque set dont on connaît les vraies valeurs de y. Pour le test set, on se contente d'afficher la proportion des prédictions qui mènent à considérer une réponse positive y=1.

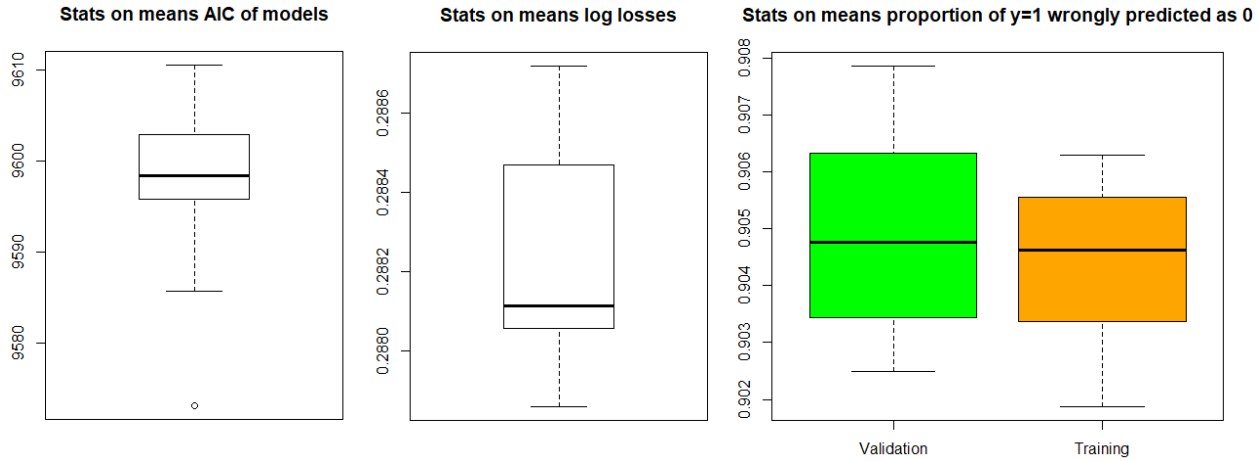


À noter que ce modèle a été fit et validé sur des ensembles tels que obtenus par un appel à la procédure `SepDataSet` avec les paramètres *prop=10* et *balance=5*. Il y a donc dans le training set toutes les observations telles que y=1 n'étant pas dans le validation set et une proportion de $5\times$ d'observations y=0. Afin de mieux visualiser le lien entre le paramètre *balance* et l'efficacité du modèle entrainé vis-à-vis de la LogLoss, des moyennes sont calculées sur plusieurs utilisations de `SepDataSet` par valeur de balance (avec un nombre d'observations du validation set fixé par le paramètre *prop* à 10). On visualise également les distributions des prédictions sur le test set par de tels modèles :





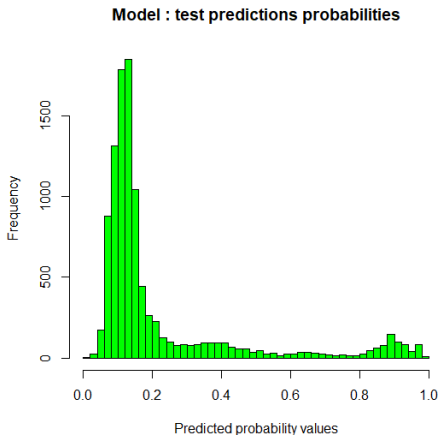
La procédure de cross-validation va permettre d'estimer l'erreur de test et de mettre en évidence la stabilité du modèle. À chaque fold, on retient 3 mesures : l'indicateur AIC du modèle construit, la valeur de la fonction d'erreur LogLoss par rapport à son application sur le test set du fold et la proportion de faux négatifs sur les réponses qui en sortent. Il s'agit de moyenne sur 10 runs où le jeu de données était partitionné en 10 folds (appel à `CrossValidate` avec *nbrCV*=10 et *nbrFolds*=10, donc *size(trainingset)* $\approx$ 11300 et *size(testset)* $\approx$ 3040), l'aspect aléatoire résidant dans la sélection des observations de `SepDataSet`.



4 Conclusion

Après avoir appliqué la *stepwise selection* et déterminé quelles variables semblaient significative sous une hypothèse linéaire, nous avons évalué le modèle sur le training et validation set, et prédit les probabilités sur l'ensemble de test. Le fait que la logloss obtenue sur le validation set soit si élevée est dû à la proportion équivalente d'observations telles que y vaut 1 et 0, et la faible taille de cet ensemble (on se met donc en quelque sorte "dans le pire cas"). On a illustré le fait qu'en augmentant la taille du training set avec des observations telles que y=0, la logloss augmente mais la variance de la sortie *y* diminue, un bon compromis étant la valeur 5 pour *balance* (si *prop* est fixé à 10). Selon les résultats des prédictions mises en ligne, procéder de la sorte à une séparation balancée du dataset nous a permis de faire descendre la logloss de 0.55 à 0.52.

La validation croisée apporte des informations sur la façon dont le modèle se comporte quand il est construit et testé sur différentes parties du jeu de données dont on dispose. Globalement, les résultats sont assez constants et la proportion de faux négatifs reste importante, de l'ordre de 90%. Finalement, en réappliquant le modèle de régression logistique avec les variables sélectionnées sur un ensemble de training tel que retourné par `SepDataSet` avec *balance*=5 et *prop*=10, les prédictions ont la distribution suivante :




```
> print_infos(results$infos)
[1] "Model fitting on variables :"
[1] "job"          "marital"      "contact"      "month"
 "day_of_week" "campaign"     "pdays"        "previous"

             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                         12647      11397
job          11    83.99      12636      11313 2.482e-13 ***
marital       3    15.63      12633      11298 0.0013487 **
contact       1   110.79      12632      11187 < 2.2e-16 ***
month         8   519.87      12624      10667 < 2.2e-16 ***
day_of_week   4    22.16      12620      10645 0.0001863 ***
campaign      1    12.94      12619      10632 0.0003208 ***
pdays         1    26.36      12618      10606 2.839e-07 ***
previous      1    19.23      12617      10586 1.158e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '
 ' 1
[1] "Predictions on TRAINING give logloss : 0.41850165715151
1 and prop y wrongly predicted as 0 : 0.903225806451613"
      0    1
0 10447 1904
1    93  204
[1] "Predictions on VALIDATION give logloss : 0.92390692446
2802 and prop y wrongly predicted as 0 : 0.918803418803419"
     0   1
0 231 215
1   3  19

> summary(results$test_preds)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.007766 0.101484 0.129185 0.213813 0.187415 0.983534
```