Rapport Projet industriel : Prédiction de durées de séjours aux Hospices Civils de Lyon (HCL)

Grégoire MASSOT

25 Janvier 2016

Table des matières

1 Présentation du projet

1.1 Contexte du projet

Ce projet industriel est orienté Data Science. Il s'agit de prédire le plus précisément possible des durées de séjour en milieu hospitalier à partir des données relatives aux patients à leur entrée à l'hôpital et des statistiques sur les patients précédents. Il s'agit donc d'un exercice d'apprentissage supervisé.

Ce projet s'inscrit dans une volonté de réduction des couts des Hopitaux Civils de Lyon (HCL). Une maximisation de l'occupation des lits hospitaliers permet d'accueillir plus de patients avec les moyens disponibles.

1.2 Outils utilisés pour le projet

On utilise exclusivement R et son interface Rstudio pour établir les prédictions. On utilisera les bibliothèques utiles pour le problème, en particulier **xgboost**

On utilisera LaTeX pour écrire les rapports (comme celui-ci). On utilise Antidote pour la correction orthographique.

1.3 Présentation de la base de données

On travaille sur la base de données hospitalière `base_ano.txt` ayant le squelette suivant :

Duree du séjour	Âge du patient	Diagnostic principal	Service hospitalier	...
duree1	age1	dp1	service1	...
duree2	age2	dp2	service2	...
duree3	age3	dp3	service3	...
...	...	...	...	...

La table contient les prédicteurs suivants :

Prédicteur	Nature	Decription	Modèle 1	Modèle 2	Modèle 3
moisent	Discret	Numéro du mois lors de l'entrée du patient	Non	Oui	Oui
moissor	Discret	Numéro du mois lors de la sortie du patient	Non	Non	Non
age	Continu	Age du patient, en années	Oui	Oui	Oui
codepos	Discret	Code postal du domicile du patient	Non	Oui	Oui
MI-DEPR2	Discrets	Comorbidités diagnostiquées	Non	Non	Oui
Elix	Discret	Nombre de comorbidités diagnostiquées	Non	Non	Oui
centre	Discret	Numéro du centre des HCL ou le patient a été pris en charge	Non	Oui	Oui
service	Discret	Numéro de l'unité médicale des HCL ou le patient a été pris en charge	Oui	Oui	Oui
entree	Discret	Mode d'entrée du patient	Non	Oui	Oui
sortie	Discret	Mode de sortie du patient	Non	Non	Non
diag	Discret	Diagnostic du patient selon la classification CIM-10	Oui	Oui	Oui
ID	Discret	ID du séjour	Non	Non	Oui

Dans le premier modèle, on garde seulement les prédicteurs de l'étude précédente. Dans le second modèle, on ajoute tous les prédicteurs qui sont connus de façon certaine lors de l'arrivée du patient.

Dans le dernier troisième modèle, on ajoute :

- Les comorbidités : elles ne sont pas connues de façon certaine lors de l'arrivée du patient
- l'ID : Il permet de situer temporellement le séjour par rapport aux autres dans la table. Ce n'est pas un prédicteur utilisable en pratique. Il permet cependant de mesurer l'importance de pouvoir situer les séjours dans le temps.

Le mois de sortie et le mode de sortie sont évidemment déterminés le jour de sortie du patient. On élimine donc ces données de la table pour tous les modèles.

1.4 Chargement de la BDD

On garde seulement les séjours dont la durée est supérieure ou égale à 1 jour. On découpe le jeu de données donnees_hcl en un TrainData sur lequel on construira le modèle et un TestData sur lequel on comparera les durées de séjours prédites et les durées de séjour réelles.

```
# Chargement de la BDD
donnees_hcl <- read.csv2(file = "base_ano.txt",header = TRUE, sep="
   \t")
# Sélection des variables (code à modifier en fonction du modèle)
donnees_hcl <- donnees_hcl[donnees_hcl$duree >= 1,]
donnees_hcl <- donnees_hcl[,-seq(8,57)]
donnees_hcl <- subset(donnees_hcl, select = -c(moissor, sortie))

# Transformation du facteur "age" en variable continue
donnees_hcl$age <- as.character(donnees_hcl$age)
donnees_hcl$age <- as.numeric(donnees_hcl$age)

# Création des jeux de Train et de Test
TrainData <- donnees_hcl[donnees_hcl$Selected == 1,]
TestData <- donnees_hcl[donnees_hcl$Selected == 0,]

predicteurs_Train <- subset(TrainData, select = -c(duree))
duree_Train <- subset(TrainData, select = c(duree))

predicteurs_Test <- subset(TestData, select = -c(duree))
duree_Test <- subset(TestData, select = c(duree))
```

1.5 Mesure de la performance des modèles

On utilisera le critère RMSE (*Root Mean Square Error*) pour pouvoir mesurer la performance des modèles proposés. C'est le critère qui a été utilisé dans l'étude précédente. Cela permettra de mesurer la performance de cette étude.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^{pred})^2}$$

```
RMSE <- function(y, pred)
{
  sqrt(mean((y - pred)^2))
}
```

On utilise aussi le critère MAE (*Mean Absolute Error*), plus parlant que le RMSE. C'est l'erreur moyenne sur les prévisions des durées de séjours (en jours)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i^{pred}|$$

```
MAE <- function(y, pred)
{
  mean(abs(y - pred))
}
```

2 Modélisation du problème

2.1 Présentation de la méthode

Le Gradient Boosting est une méthode de machine learning parmi les plus fréquemment utilisées pour les problèmes d'apprentissage supervisés. La bibliothèque xgboost est souvent utilisée dans les solutions gagnantes des challenges kaggle.

2.2 Implémentation de la méthode

2.2.1 Réglage des paramètres

Plusieurs paramètres sont à ajuster pour la méthode du gradient boosting. Les plus importants sont les suivants :

- nrounds : c'est le nombre d'arbres à construire. Plus il est grand, plus la prédiction est exacte et plus le temps de simulation est long
- maxdepth : c'est la profondeur maximale des arbres de décision.
- eta : permet de régler le compromis biais/variance lors de l'apprentissage.

On utilise la fonction train du package caret pour pouvoir choisir au mieux ces paramètres.

```
# Tuning des paramètres de xgboost
library(caret)
library(foreach)
library(doParallel)
cl <- makeCluster(8)

tuneGrid <- expand.grid(max_depth = c(2,4,6,8,10),
                        nrounds = 100,
                        eta = c(0.01,0.1,0.5,1))

fitControl <- trainControl(method = "repeatedcv",
                           number = 3,
                           repeats = 1)

xvalXGB <- train(TrainData$duree ~ .,
                 data = TrainData,
                 method = "xgbTree",
                 tuneGrid = tuneGrid,
                 trControl = fitControl)
```

2.2.2 Construction du modèle

Une fois ce réglage effectué, on peut lancer le gradient boosting sur le TrainData avec un nombre d'arbres le plus grand possible pour avoir le modèle le plus précis construit dans un temps raisonnable.

```
# Construction du modèle
library(xgboost)
param <- list("objective" = "reg:linear",
              "eta"=0.5,
              "max.depth"=4,
              "nthread" = 8)

modelXgboost <- xgboost(data = as.matrix(predicteurs_Train)
                        , label = as.matrix(duree_Train)
```

```
                            , params=param
                            , nrounds = 100)
predictionXGBoost <- predict(modelXgboost,
                             as.matrix(predicteurs_Test))

RMSE(duree_Test, predictionXGBoost)
```

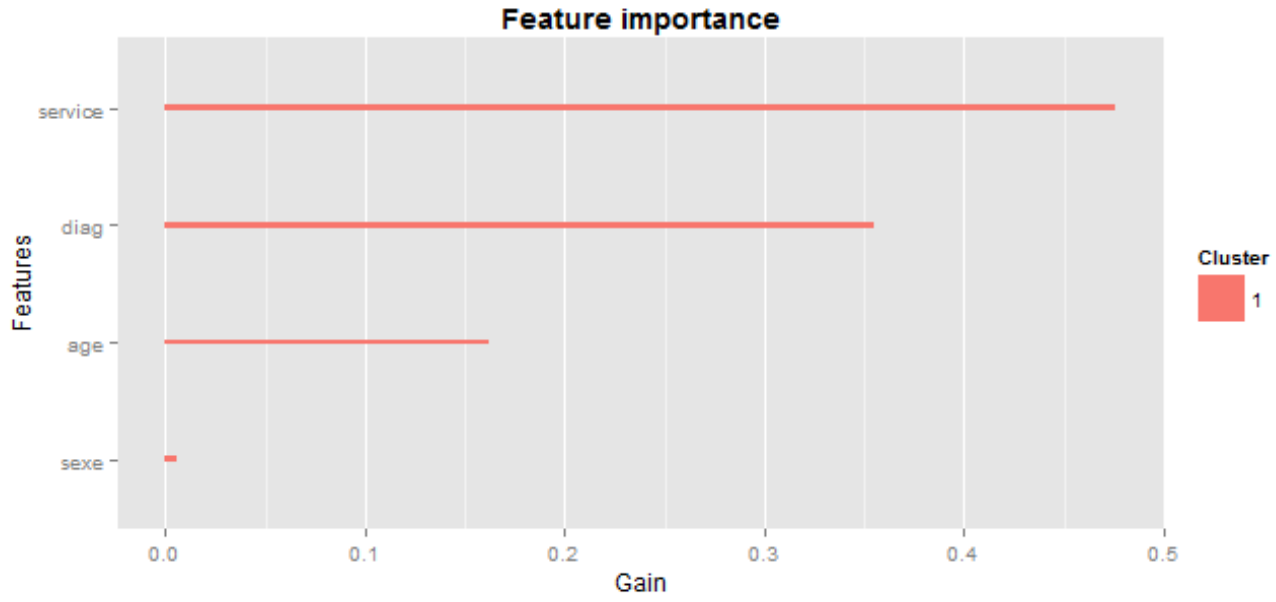
3 Analyse des résultats

3.1 Importance des différentes variables

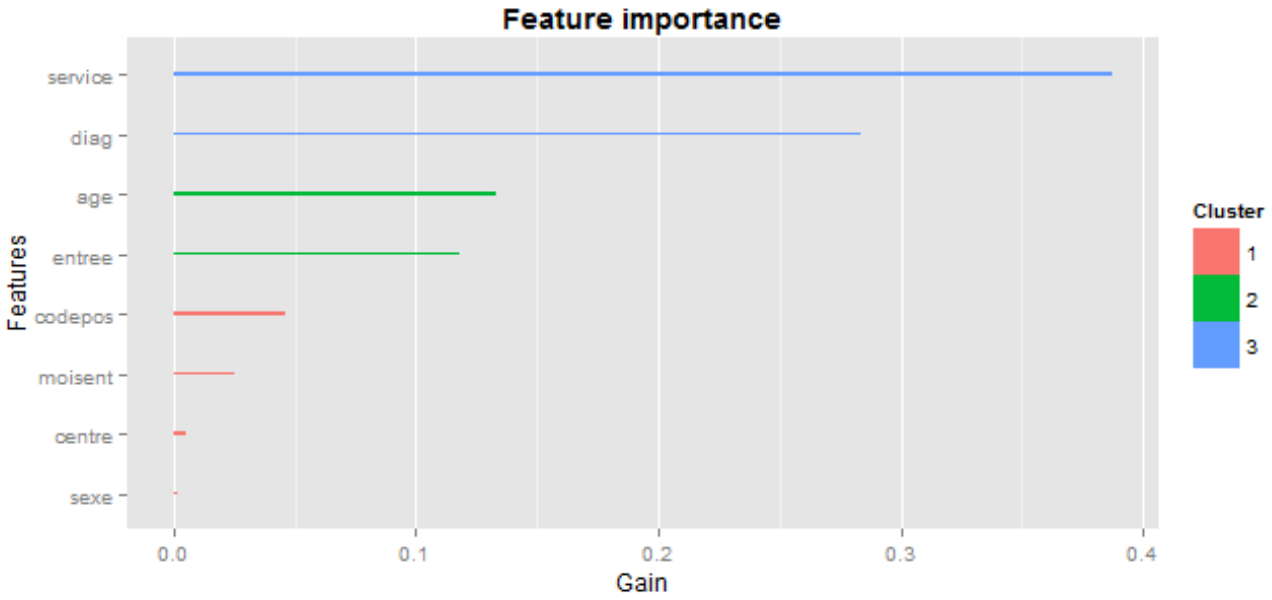
On utilise l'outil xgb.importance fourni avec xgboost pour déterminer les prédicteurs retenus par l'algorithme de machine learning.

```
names <- dimnames(predicteurs_Train)[[2]]
importance_matrix <- xgb.importance(names, model = modelXgboost)
xgb.plot.importance(importance_matrix)
xgb.plot.tree(feature_names = names, model = modelXgboost, n_first_
    tree = 2)
```

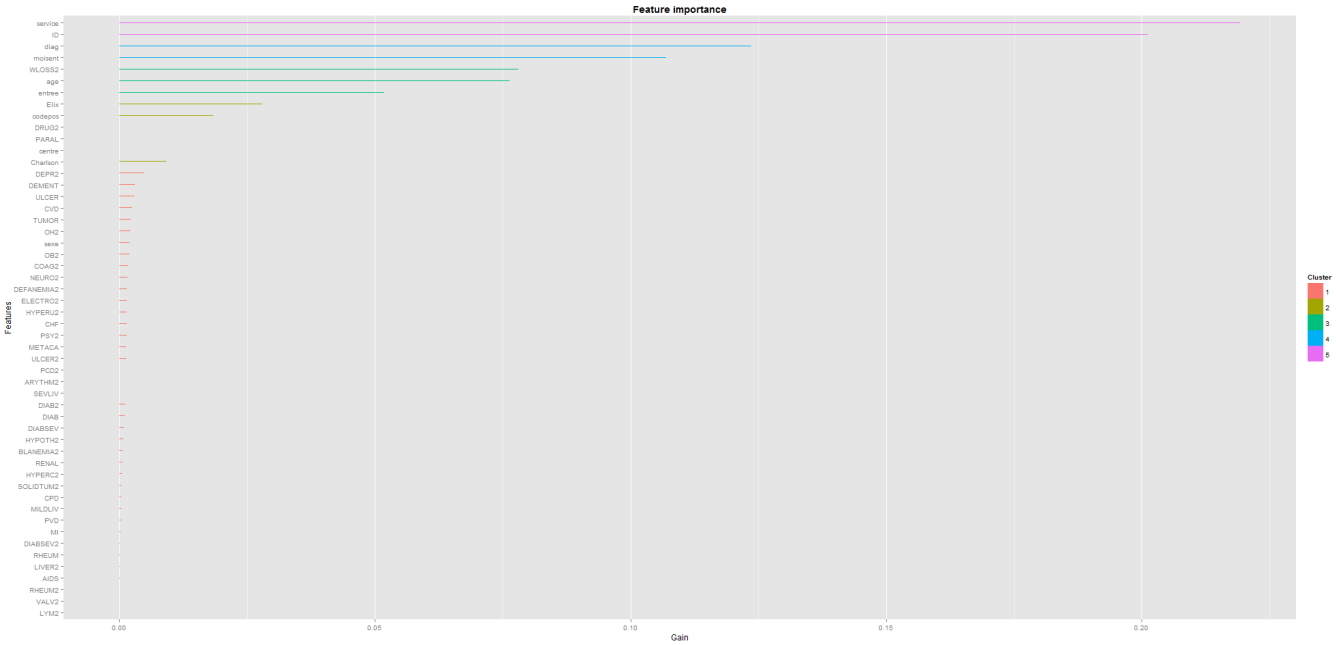
3.1.1 1er Modèle



3.1.2 2nd Modèle



3.1.3 3ème Modèle



3.2 Performance des modèles

	RMSE	MAE
Modèle 1	6.88	3.14 jours
Modèle 2	6.84	3.15 jours
Modèle 3	5.68	2.75 jours

On voit que l'ajout de l'identifiant (qui permet de situer le séjour temporellement par rapport aux autres) ainsi que certaines comorbidités permettent de gagner environ 10% de MAE

3.3 Incertitudes sur les résultats

Pour améliorer la précision de la prédiction, on peut demander à xgboost de donner prédire non plus une durée brute, mais la densité de probabilité de la variable aléatoire relative à la durée de séjour.

Pour cela, on ajoute le code suivant lors du chargement des données

```
donnees_hcl <- donnees_hcl[donnees_hcl$duree >= 1 & donnees_hcl$
    duree <= 30 ,]
duree_Test_RMSE <- subset(donnees_hcl[donnees_hcl$Selected == 0,],
    select = c(duree))
donnees_hcl$duree <- as.factor(donnees_hcl$duree)
```

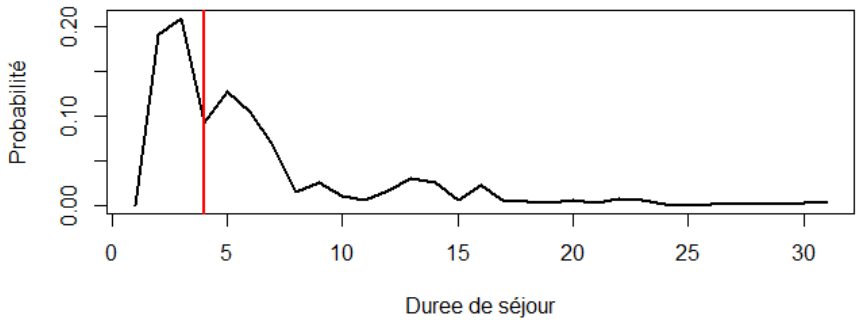
On remplace aussi les paramètres de xgboost.

```
param <- list("objective" = "multi:softprob",
              "eta"=0.3,
              "max.depth"=5,
              "nthread" = 8,
              "num_class" = 31)

modelXgboost <- xgboost(data = as.matrix(predicteurs_Train)
                        , label = as.matrix(duree_Train)
                        , params=param
                        , nrounds = 100)
xgbtest <- xgb.DMatrix(data.matrix(predicteurs_Test), missing=NA)

xgbpred <- predict(modelXgboost, xgbtest)
probs <- t(matrix(xgbpred, nrow=31, ncol=length(xgbpred)/31))
```

On affiche ensuite les résultats, par exemple, les 100 premières entrées du jeu de test

```
for(i in 1:100)
{
  plot(probs[i,], type='l')
  abline(v=as.numeric(duree_Test$duree[i])+1, col="red")
}
```

On obtient des figures de ce type :



En noir la densité de probabilité de la variable aléatoire associée à la durée de séjour. En rouge la durée de séjour effectivement observée.

RMSE : 3.62 MAE : 2.24 jours

Les erreurs sont plus petites car on s'est limité aux séjours allant de moins de 30 jours afin de visualiser facilement les résultats sur un graphique.