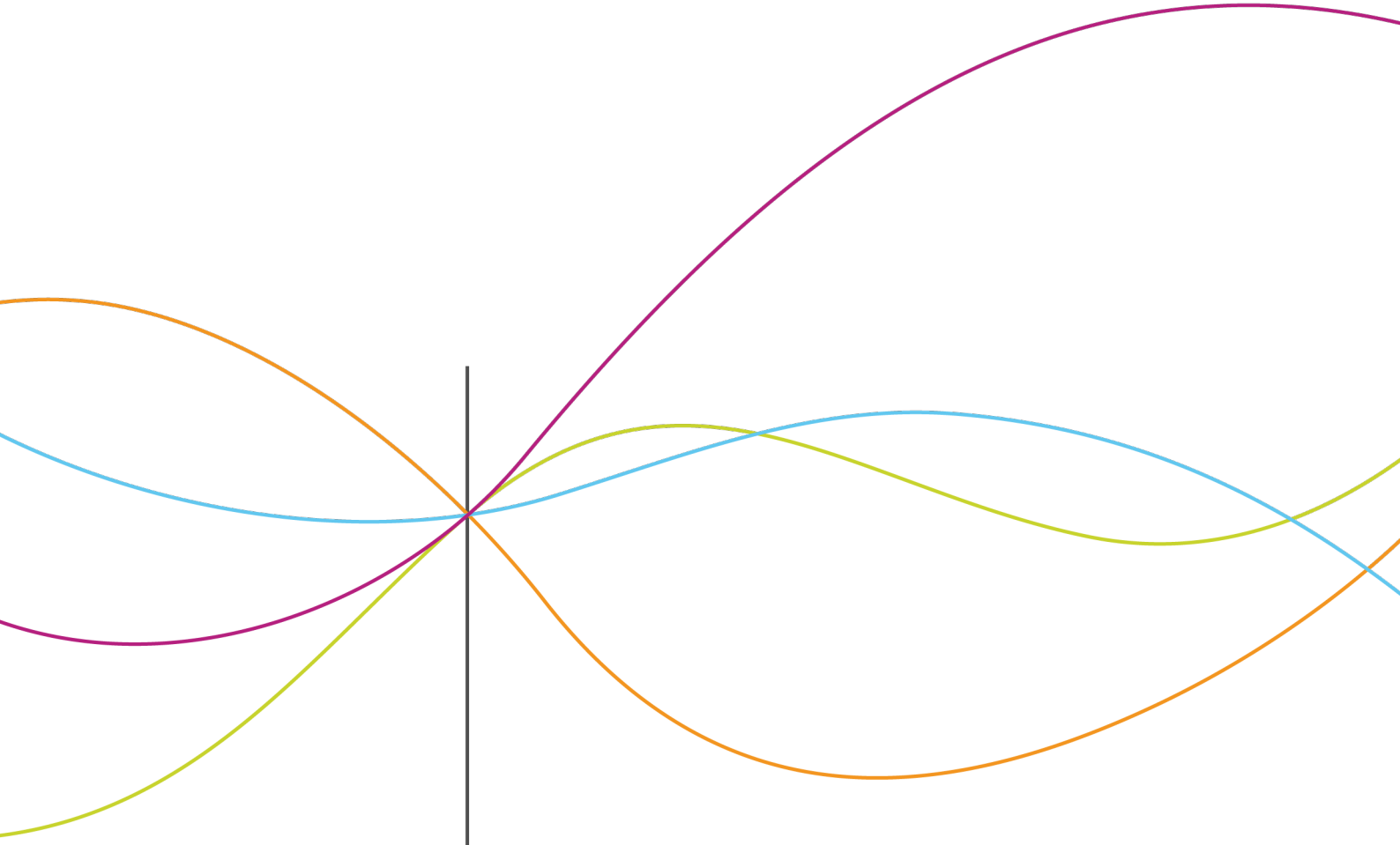
École nationale
de la statistique
et de l'administration
économique

ENSAE ParisTech

université PARIS-SACLAY



Gouysse Margaux
Grelety Antoine
Watrigant Timothée

TP3 : Prédiction du diabète

2017/2018

Introduction

Notre TP n°3 traite un problème de prédiction de l'état de santé d'un patient. En particulier, nous nous intéressons à un problème de classification binaire pour prédire si un patient est diabétique ou non à partir de ses relevés médicaux et caractéristiques individuelles. Nous allons tout d'abord décrire les données utilisées, spécifier les modèles d'apprentissage puis présenter nos résultats.

Données/Statistiques descriptives

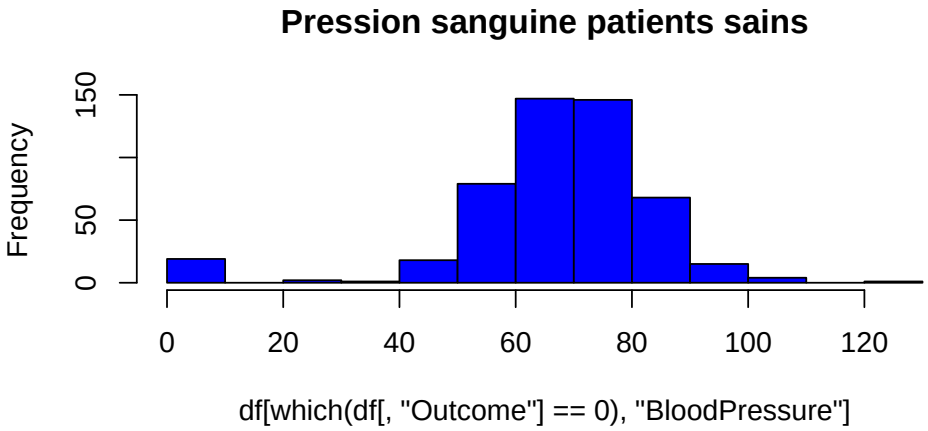
Notre étude repose sur les données suivantes : 1.Le nombre de grossesses du patient 2.La concentration de glucose du patient 3.La pression sanguine 4.L'épaisseur de la peau 5.La concentration d'insuline 6.L'indice de masse corporel 7.Une valeur sur la part d'hérédité du diabète 8.L'âge du patient 9.Et si le patient est diabétique

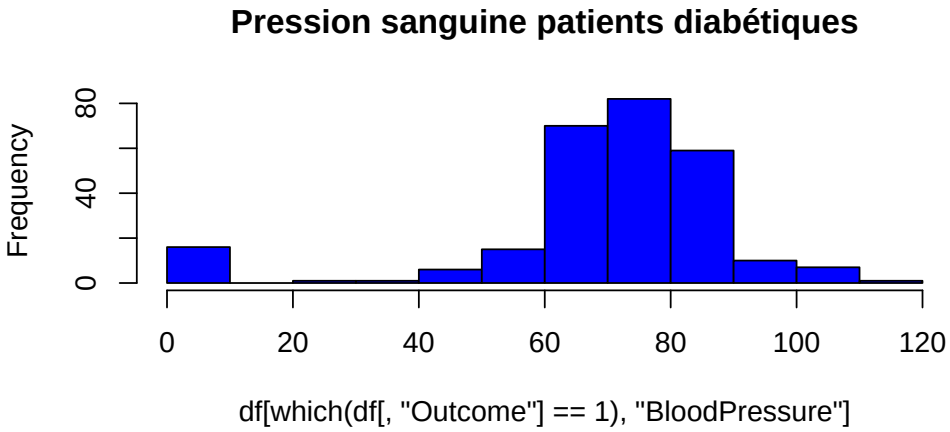
```
  Pregnancies        Glucose      BloodPressure    SkinThickness
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
    Insulin           BMI        DiabetesPedigreeFunction      Age
 Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
 1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
 Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
 Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
 3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
 Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
 Outcome
 0:500
 1:268
```

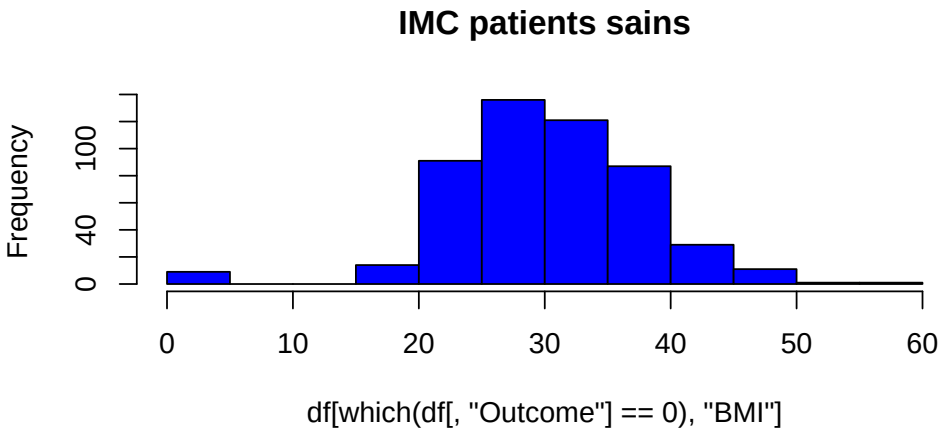
Notre étude ne présente pas de données manquantes, cependant les mesures des variables 2. à 6. comportent des valeurs nulles, ce qui est physiquement impossible pour un être vivant. Ces valeurs nulles sont certainement dues à une absence de mesure. Nous allons remplacer ces valeurs nulles par la médiane de l'échantillon de la variable considérée.

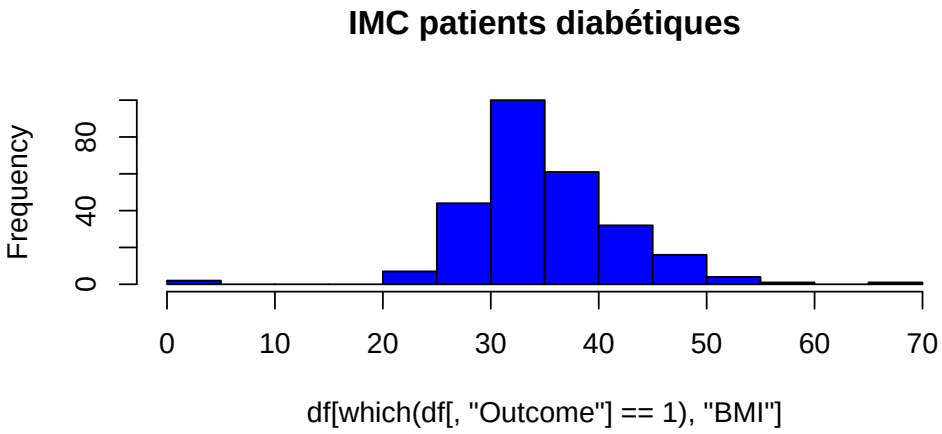
Ici notre échantillon est composé de 768 patients avec 268 patients diabétiques soit 35% de l'échantillon de départ. Il faudra donc que nos modèles aient des performances supérieures à 65% puisque si les résultats sont inférieurs à 65%, on aura plus de chance de prédire toujours que notre patient est non diabétique (65% de l'échantillon) que d'utiliser notre modèle.

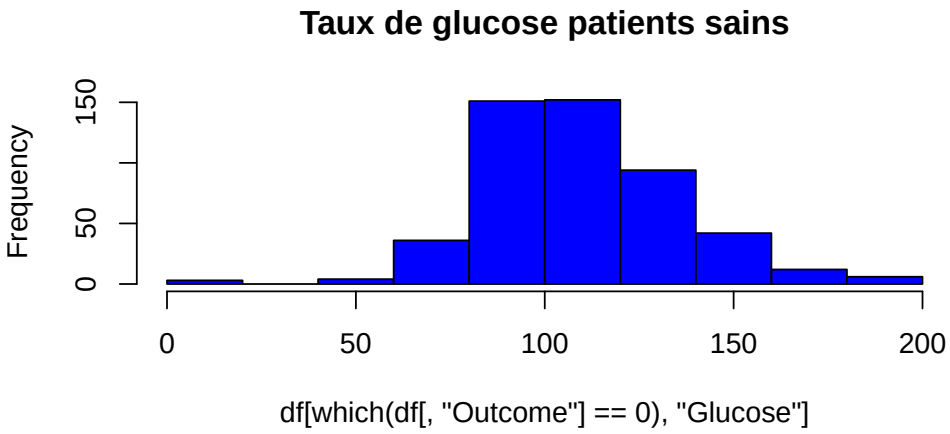
On trace des histogrammes afin de montrer les potentielles différences entre diabétiques :

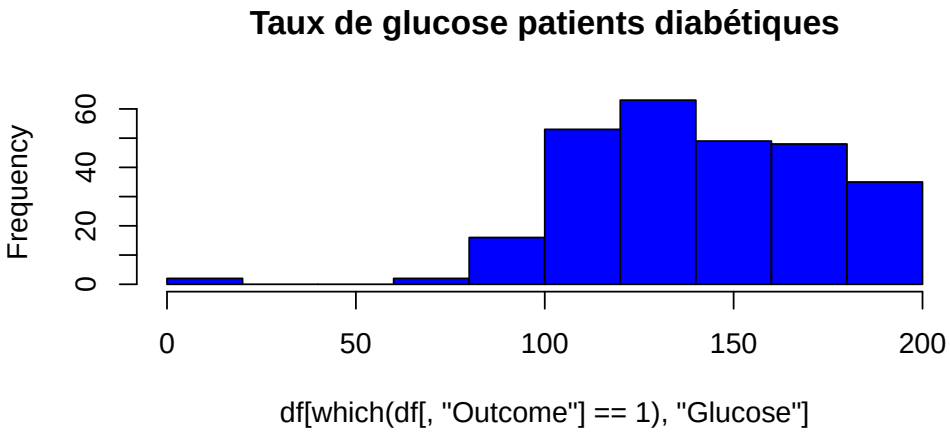





IMC patients sains
Frequency
0
40
100
0
10
20
30
40
50
60
df[which(df[, "Outcome"] == 0), "BMI"]


IMC patients diabétiques
Frequency
0
40
80
0
10
20
30
40
50
60
70
df[which(df[, "Outcome"] == 1), "BMI"]





Concernant, le taux de glucose et l'IMC, les observations semblent proches pour les deux catégories de patients. On observe en revanche que le taux de glucose est plus important pour les patients diabétique. Cette variable semble donc significative et risque d'avoir un rôle important dans la prédiction du diabète.
Le but de notre étude sera donc de déterminer si un patient est diabétique ou pas à partir des autres données du patient. Pour cela, on divisera notre échantillon en 2 parties avec une partie utilisée pour l'apprentissage de nos modèles et une partie pour tester les performances de notre modèle puisqu'on est dans un cas d'apprentissage supervisé.

Modèles

Arbre de décision

Notre premier modèle est un arbre de décision. Nous effectuons une validation croisée avec plusieurs valeurs du paramètre CP afin d'extraire celui ayant les meilleures performances. CP permet de contrôler la taille de l'arbre en élaguant les branches. Il mesure en quelque sorte la complexité de l'arbre.

```
> l = sample(1:nrow(df),nrow(df)/8)
> data_valid <- df[l,]
> ind.data_cross = setdiff(1:nrow(df),l)
> data_cross <- df[ind.data_cross,]
> #cross validation 7 fold
> fold=7
> nbdata = nrow(data_cross)
> prediction_eval = list()
> accuracy_eval = c()
> list_cp = c(10^c(-(1:3)))
> for (k in 1:fold)
```

```
+   {
+ for (c in list_cp)
+   {
+   ind.test= ((k-1)*(nbdata/fold)+1):(k*(nbdata/fold))
+   ind.train = setdiff(1:nrow(data_cross),ind.test)
+   fit <- rpart(Outcome ~ .,data= data_cross[ind.train,],
+                control = rpart.control(cp = c, minsplit = 2, minbucket = 1))
+   prediction_eval[[k]] <- predict(fit, data_cross[ind.test,])
+   prediction_eval[[k]]<-apply(prediction_eval[[k]],1,which.max)-1
+   prediction_eval[[k]] <- as.factor(prediction_eval[[k]])
+   accuracy_eval<-rbind(accuracy_eval,c(c,k,1-sum(ifelse(prediction_eval [[k]]!= data_cross[ind.test,"
+   }
+ }
> colnames(accuracy_eval) <- c("cp","k","acc")
> boxplot(acc~cp,data = accuracy_eval)
```

Support Vector Machine

Pour le second modèle, nous avons testé un SVM. La taille relativement réduite de l'échantillon nous permet de tester ce type de modèle au coût de calcul élevé. Nous avons testé deux types de Kernels : un sigmoid et un Radial Basis Function (RBF). Pour chacun de ces kernels, nous avons effectué une validation croisée avec différentes valeurs de gamma avec un paramètre de coût c constant et égal à c=1000. Gamma permet de définir la sphère d'influence de l'échantillon d'entrainement, ou bien complexité du modèle. Le modèle est donc très sensible à la valeur de gamma. Il est donc important de choisir cette valeur de manière rigoureuse, ce qui justifie notre validation croisée.

```
> fold=3
> nbdata = nrow(df)
> list_gamma = c(10^c(-(1:3)))
> print("----------------KERNEL=SIGMOID--------------")
> for (k in 1:fold)
+ {
+   for (c in list_gamma)
+   {
+     ind.test= ((k-1)*(nbdata/fold)+1):(k*(nbdata/fold))
+     ind.train = setdiff(1:nrow(df),ind.test)
+     fit_svm<-svm(Outcome ~.,data= df[ind.train,],cost=1000,gamma=c,kernel="sigmoid")
+     pred_eval_svm<-predict(fit_svm,df[ind.test,])
+     print(paste("precision:",1-sum(ifelse(pred_eval_svm != df[ind.test,]$Outcome,1,0))/nrow(df[ind.te
+           "gamma=",c,"k=",k))
+   }
+ }
> print("---------------KERNEL=RBF--------------")
> for (k in 1:fold)
+ {
+   for (c in list_gamma)
+   {
+     ind.test= ((k-1)*(nbdata/fold)+1):(k*(nbdata/fold))
+     ind.train = setdiff(1:nrow(df),ind.test)
+     fit_svm<-svm(Outcome ~.,data= df[ind.train,],cost=1000,gamma=c) #rbf par dÃ©faut
+     pred_eval_svm<-predict(fit_svm,df[ind.test,])
+     print(paste("precision:",1-sum(ifelse(pred_eval_svm != df[ind.test,]$Outcome,1,0))/nrow(df[ind.te
+                 "gamma=",c,"k=",k))
+   }
+ }
```

K Plus Proches Voisins (KNN)

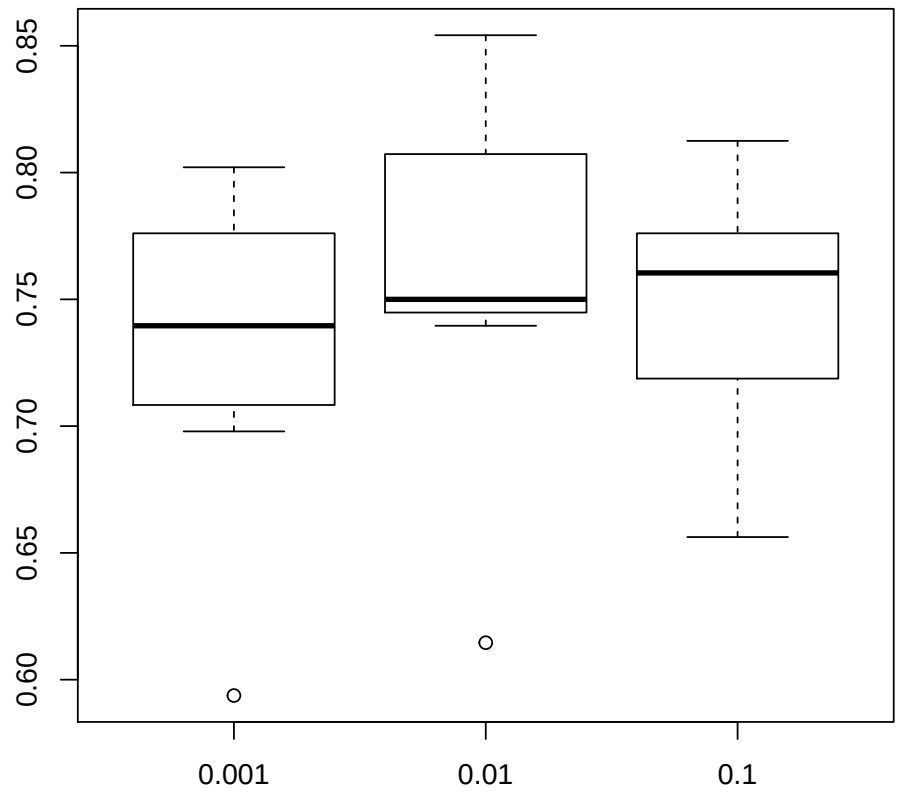
Notre dernier modèle est un algorithme KNN. Nous testerons celui-ci avec trois valeurs différentes de proximité : K=3,5 et 7. Ces trois valeurs seront également testées avec une validation croisée :

```
> list_kpp=c(3,5,7)
> #cross validation
> fold=5
> for (k in 1:fold)
+ {
+   for (c in list_kpp)
+   {
+     ind.test= ((k-1)*(nbdata/fold)+1):(k*(nbdata/fold))
+     ind.train = setdiff(1:nrow(df),ind.test)
+     fit.knn<-kNN(Outcome~.,df[ind.train,],df[ind.test,],k=c,norm=TRUE)
+     print(paste("precision:",1-sum(ifelse(fit.knn != df[ind.test,]$Outcome,1,0))/nrow(df[ind.test,]),
+             "kpp=",c,"k_fold=",k))
+   }
+ }
```

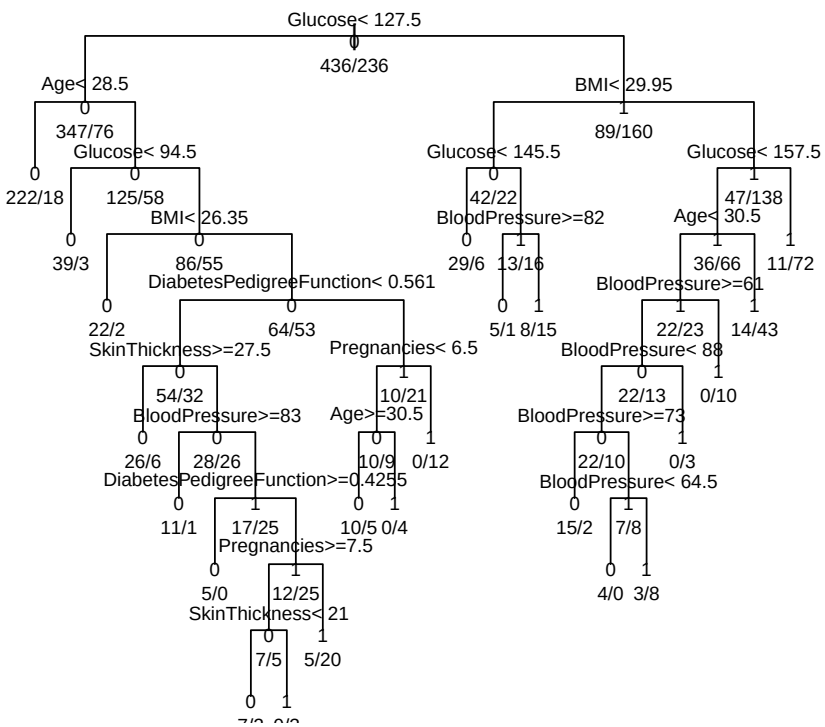
Résultats

Arbre de décision

Traçons le diagramme représentant les valeurs de l'accuracy en fonction des paramètres cp de l'arbre :



D'après ce diagramme, on va donc fixer le paramètre cp à 0.01 et on va obtenir l'arbre de décision suivant :



On calcule alors l'accuracy associé à cet arbre :

```
[1] 0.6979167
```

On arrive donc à 69.8% de précision de test sur le modèle avec un paramètre de 0.01. On a donc des performances supérieures à notre baseline qui serait de prédire toujours qu'un patient soit non diabétique (65% de l'échantillon)

Support Vector Machine

La cross validation nous fournit alors les valeurs suivantes pour le kernel sigmoid :

```
[1] "----------------KERNEL=SIGMOID--------------"

[1] "precision: 0.68359375 gamma= 0.1 k= 1"
[1] "precision: 0.7109375 gamma= 0.01 k= 1"
[1] "precision: 0.75390625 gamma= 0.001 k= 1"
[1] "precision: 0.70703125 gamma= 0.1 k= 2"
[1] "precision: 0.70703125 gamma= 0.01 k= 2"
[1] "precision: 0.73828125 gamma= 0.001 k= 2"
[1] "precision: 0.6953125 gamma= 0.1 k= 3"
[1] "precision: 0.75 gamma= 0.01 k= 3"
[1] "precision: 0.80078125 gamma= 0.001 k= 3"
```

Et pour le kernel de type Radial Basis :

```
[1] "precision: 0.70703125 gamma= 0.1 k= 1"
[1] "precision: 0.75390625 gamma= 0.01 k= 1"
[1] "precision: 0.765625 gamma= 0.001 k= 1"
```

```
[1] "precision: 0.63671875 gamma= 0.1 k= 2"
[1] "precision: 0.70703125 gamma= 0.01 k= 2"
[1] "precision: 0.73828125 gamma= 0.001 k= 2"
[1] "precision: 0.73828125 gamma= 0.1 k= 3"
[1] "precision: 0.8203125 gamma= 0.01 k= 3"
[1] "precision: 0.8046875 gamma= 0.001 k= 3"
```

On peut alors voir qu'un kernel de type Radial Basis avec Gamma=0.001 est la composition la plus performante avec 79% de précision. Modélisons alors la matrice de confusion associée à ce modèle :

```
$table
          Reference
Prediction   0   1
         0 147  32
         1  21  56

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
  7.929688e-01   5.270496e-01   7.381117e-01   8.409032e-01   6.562500e-01
AccuracyPValue  McnemarPValue
  1.158856e-06   1.695641e-01
```

K Plus Proches Voisins (KNN)

```
[1] "precision: 0.718954248366013 kpp= 3 k_fold= 1"
[1] "precision: 0.738562091503268 kpp= 5 k_fold= 1"
[1] "precision: 0.745098039215686 kpp= 7 k_fold= 1"
[1] "precision: 0.836601307189543 kpp= 3 k_fold= 2"
[1] "precision: 0.803921568627451 kpp= 5 k_fold= 2"
[1] "precision: 0.77124183006536 kpp= 7 k_fold= 2"
[1] "precision: 0.862745098039216 kpp= 3 k_fold= 3"
[1] "precision: 0.836601307189543 kpp= 5 k_fold= 3"
[1] "precision: 0.830065359477124 kpp= 7 k_fold= 3"
[1] "precision: 0.901960784313726 kpp= 3 k_fold= 4"
[1] "precision: 0.882352941176471 kpp= 5 k_fold= 4"
[1] "precision: 0.849673202614379 kpp= 7 k_fold= 4"
[1] "precision: 0.823529411764706 kpp= 3 k_fold= 5"
[1] "precision: 0.810457516339869 kpp= 5 k_fold= 5"
[1] "precision: 0.758169934640523 kpp= 7 k_fold= 5"
```

Les meilleurs résultats pour ce modèle sont obtenus avec k =3 (en moyenne). On a alors la matrice de confusion suivante :

```
$table
          Reference
Prediction   0   1
         0 139  40
         1  29  48

$overall
      Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
    0.73046875     0.38427217     0.67169099     0.78382305     0.65625000
AccuracyPValue  McnemarPValue
    0.00667533     0.22864426
```

Cet algorithme permet d'obtenir 73% de précision sur l'échantillon test. Observons cependant que le modèle a plus de difficultés à prédire la classe 1. Sur l'ensemble de l'échantillon test, il prédit $29 + 48 = 77$ individus diabétiques, alors que seulement 48 le sont réellement. Le modèle prédit alors 37.6% de faux-positifs diabétiques. De l'autre côté, le taux de faux-négatifs, c'est à dire les individus prédits non diabétiques alors qu'ils le sont, représente une part plus faible avec 28.7%.
Nous observons des résultats similaires pour les autres modèles.

Conclusion

Pour ce TP, nous nous sommes donc intéressés à la prédiction du diabète. Pour cela, nous avons donc mis au point 3 modèles (arbre de décision, SVM, Knn).Une validation croisée nous a permis de calibrer nos modèles de manière rigoureuse en choisissant les bons paramètres, tout en vérifiant que nos modèles généralisent correctement au delà de l'échantillon d'entrainement. Ces trois modèles peuvent être considérés comme performants car ces trois algorithmes fournissent des résultats compris entre 66% et 73% de précision qui sont supérieurs à la baseline (65%). L'algorithme SVM est le modèle le plus peformant avec 79% de précision.