Prédire le Diabète

Joao BABADOUDOU

2023-08-03

1-Contexte

Le diabète est un problème de santé publique majeur dans le monde entier, touchant des millions de personnes et ayant un impact significatif sur leur qualité de vie. Dans cette étude nous proposons une approche basée sur les machines a vecteurs de support (SVM) pour détecter la maladie. Pour cela nous allons utiliser des données récupérées sur **Kaggle**, présentant certaines caractéristiques tels que la pression sanguine ou la quantité d'insuline dans le sang de certaines personnes.

Commençons par charger toutes les bibliothèques dont on aura besoin et les données. Les données sont disponibles sur mon github.

```
library(readr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(GGally)
library(gt)
library(e1071)
library(caret)
library(skimr)
library(DALEX)
# Importons nos données
df <- read_delim("diabetes.csv", delim = ";",
                        escape_double = FALSE, col_types = cols(Outcome = col_factor(levels = c("0",
                                                                                                "1"))),
```

2-Analyse exploratoire des données

2-1) Présentation du jeu de données

Pour les informations sur les données, je vous laisse le soin de visiter la page de Kaggle:

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

2-2) Analyse univariée

```
my_skim <- skim_with(numeric = sfl( p50 = NULL,hist=NULL, n_missing=NULL, complete_rate=NULL,p25=NULL
diabetes_df <- my_skim(df[,-9])

diabetes_df %>%
  select(-skim_type)   %>%
  gt() %>%
  cols_label(
             numeric.mean = "Moyenne", numeric.sd = "Ecart-type",
             numeric.p0 = "Min" ,
             numeric.p100 = "Max") %>%
  opt_stylize(style = 6, color = "cyan", add_row_striping = TRUE) %>%
  tab_header(title = "Summary of Variables in the diabetes data")
```

Statistiques descriptives

Summary of Variables in the diabetes data

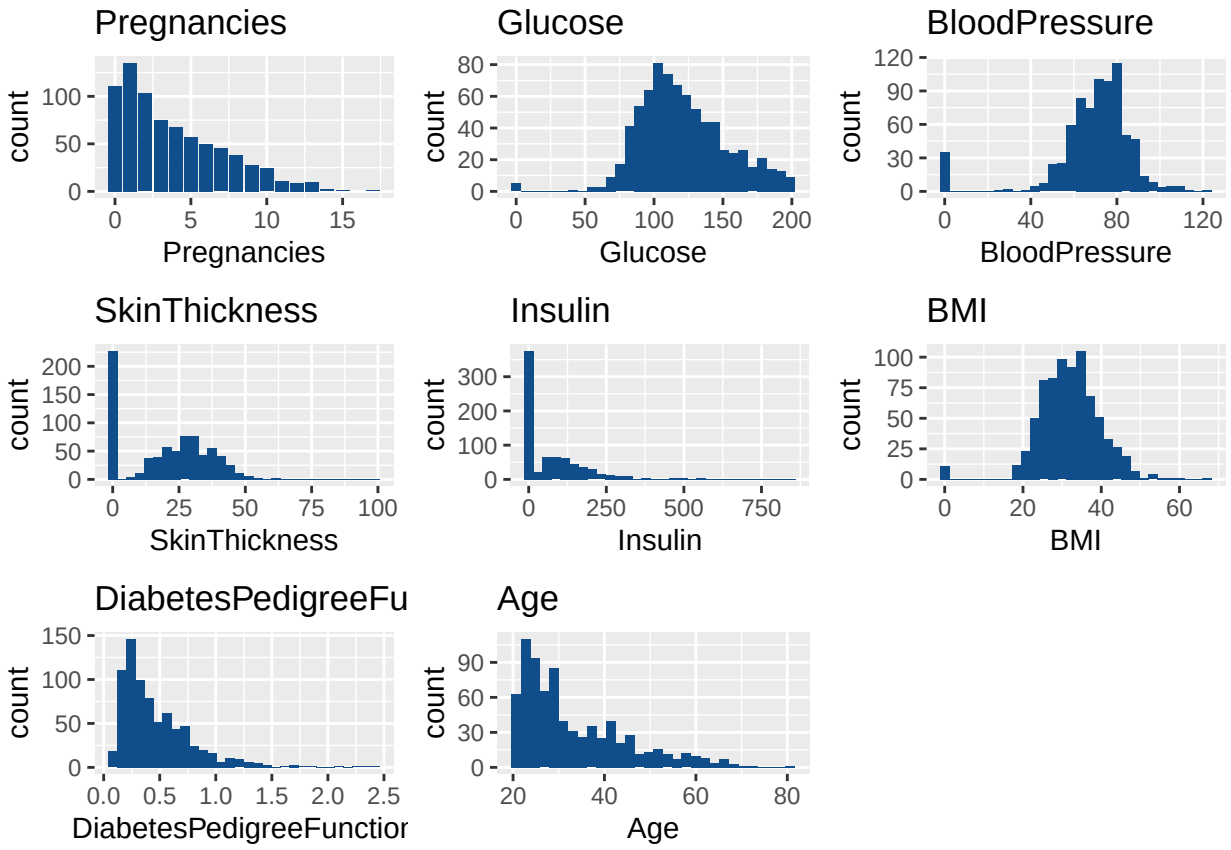
skim_variable	n_missing	complete_rate	Moyenne	Ecart-type	Min	Max
Pregnancies	0	1	3.8450521	3.3695781	0.000	17.00
Glucose	0	1	120.8945312	31.9726182	0.000	199.00
BloodPressure	0	1	69.1054688	19.3558072	0.000	122.00
SkinThickness	0	1	20.5364583	15.9522176	0.000	99.00
Insulin	0	1	79.7994792	115.2440024	0.000	846.00
BMI	0	1	31.9925781	7.8841603	0.000	67.10
DiabetesPedigreeFunction	0	1	0.4718763	0.3313286	0.078	2.42
Age	0	1	33.2408854	11.7602315	21.000	81.00

L'analyse du tableau nous permet de constater l'absence de valeurs manquantes et des écarts-types généralement élevés, surtout pour la quantité d'Insuline dans le sang.

Distribution des variables Affichons maintenant la distribution de chacune des variables exceptés la variable explicative.

```
c2    <-    ggplot(df,    aes(Pregnancies))    +
  geom_bar(fill    =    "#104E8B")   +
  ggtitle(names(df[1]))

d2    <-    ggplot(df,    aes(Glucose))    +
  geom_histogram( fill    =    "#104E8B")   +
```

```
  ggtitle(names(df[,2]))

e2    <-    ggplot(df,    aes(BloodPressure) )    +
  geom_histogram( fill    =    "#104E8B" )      +    ggtitle(names(df[,3]))

f2    <-    ggplot(df,    aes(SkinThickness))    +
  geom_histogram( fill    =    "#104E8B")    +
  ggtitle(names(df[,4]))

g2    <-    ggplot(df,    aes(Insulin))    +
  geom_histogram( fill    =    "#104E8B")    +    ggtitle(names(df[,5]))

h2    <-    ggplot(df,    aes(BMI))    +
  geom_histogram(fill    =    "#104E8B")    +
  ggtitle(names(df[,6]))

i2    <-    ggplot(df,    aes(DiabetesPedigreeFunction))    +
  geom_histogram(fill    =    "#104E8B")    +    ggtitle(names(df[,7]))

j2    <-    ggplot(df,    aes(Age))    +
  geom_histogram(fill    =    "#104E8B")    +
  ggtitle(names(df[,8]))


gridExtra::grid.arrange(c2,d2,e2,f2,g2,h2,i2,j2)
```

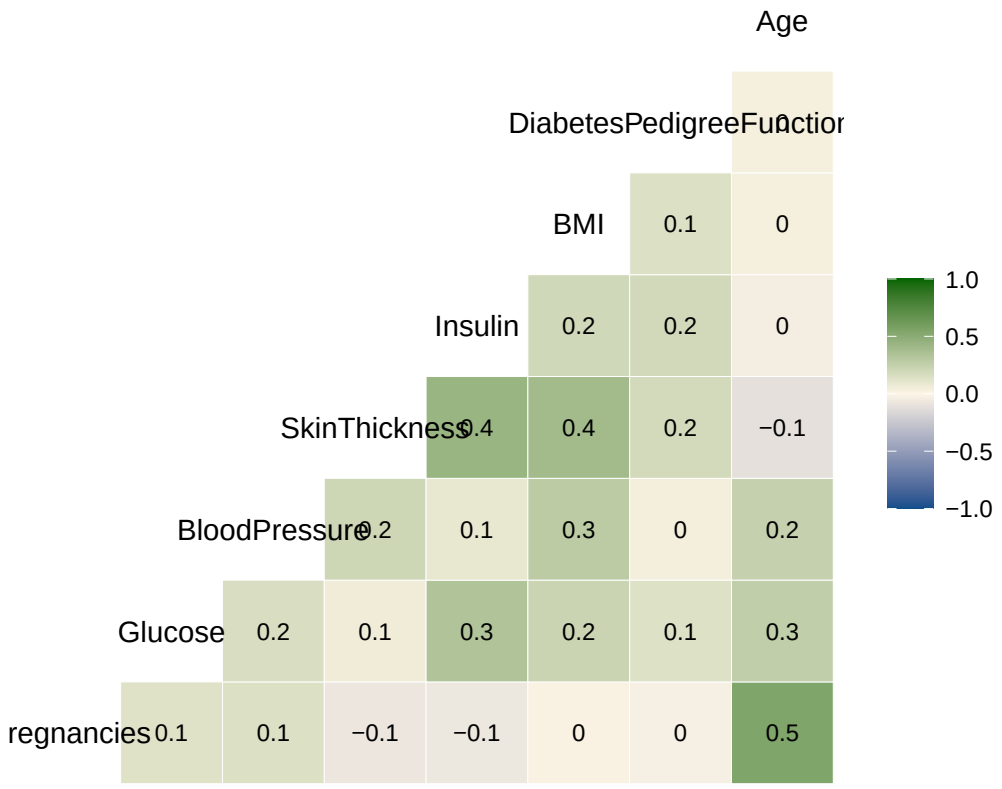


On peut constater que les variables ont pour la plupart une distribution asymétrique à gauche. C'est à dire que la majorité des valeurs de la variable est supérieure à la valeur moyenne. Il conviendra pour nous de procéder a une standardisation des données pour la suite des analyses

2-3) Analyse multivariée

Sortons dans un premier temps une matrice de corrélation pour vérifier s'il y a des relations linéaires significatives entre les variables explicatives.

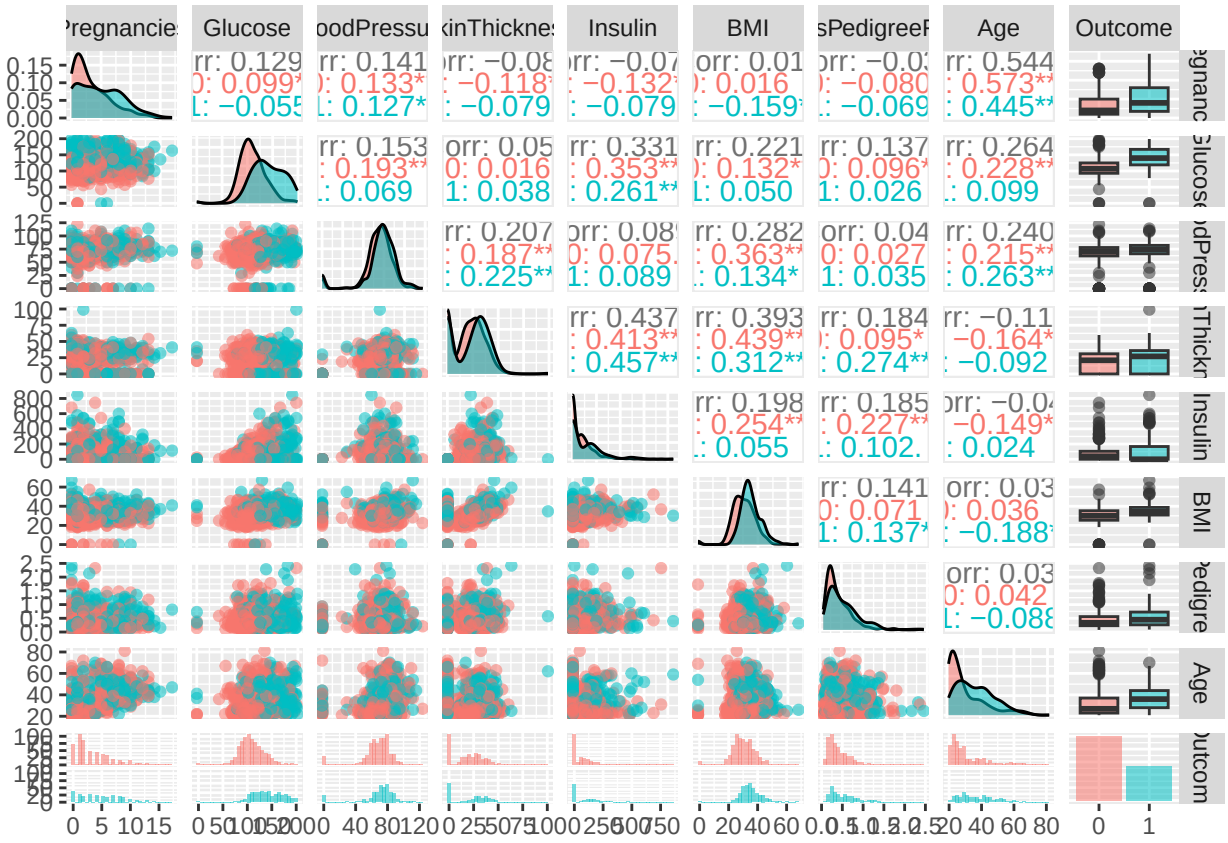
```
ggcorr(df[,-9],low = "#104E8B",mid= "#FDF5E6", high="darkgreen", label = TRUE,label_size = 3)
```



Constatons que les variables **Age** et **Pregnancies** ont une relation linéaire plus ou moins forte. Concernant les autres variables la linéarité de leurs relations n'est pas significative au vu de la valeur des coefficients.

Avec le fonction `ggpairs` de (Schloerke et al. 2021) nous allons largement résumer notre analyse multivariée et tirer les informations nécessaires.

```
ggpairs(df, aes(colour = Outcome, alpha = 0.4))
```



Quand on analyse la densité de chacune des variables repartit suivant **Outcome** , on constate un déphasage entre les courbes de **1** et celles de **0**. En effet ses figures, posées en diagonale, montrent que toutes les variables choisies sont importante dans la détection du diabète. On peut néanmoins remarquer que la quantité de **glucose** dans le sang et l'**age** pourraient être des facteurs très déterminant du diabète.

3-Modélisation

3-1) Preprocessing

Ici, nous allons préparer nos données a être utiliser dans le modèle. La première étape sera la normalisation des données.

Normalisation des données

Une étape très importante consiste à normaliser d'abord les valeurs des variables quantitatives. Toutes les mesures seront alors placées sur un pied d'égalité. Du coup, si certaines d'entre elles sont très petites, elles ne seront pas « oubliées » parmi d'autres mesures bien plus grandes . Nous allons utiliser la normalisation min-max. Ce processus transforme les variables de sorte que toutes les valeurs se situent dans la plage comprise entre 0 et 1.

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}

df[,1:8] <- apply(df[,1:8],MARGIN = 2 , FUN = normalize)
```

Séparation des donnes en données d'apprentissage et donnes de test

Nous allons séparer nos données en deux partie. Réservons 80% pour l'entraînement et donc 20% pour le test du modèle.

```
split_size  =   0.80
sample_size =   floor(split_size    *   nrow(df))
set.seed(563)
train_indices   <-  sample(seq_len(nrow(df)),   size    =
                        sample_size)
train   <-  df[train_indices,   ]
test    <-  df[-train_indices,  ]
```

3-2) Entrainement du modèle

Nous allons faire recourt à la fonction `train` pour trouver notre modèle. A l'aide de la recherche aléatoire et de la cross validation nous allons déterminer les paramètres de notre modèle. En effet, la recherche aléatoire est souvent efficace, car elle explore l'espace des hyperparamètres de manière aléatoire plutôt que de manière systématique comme la recherche simple. Cela permet de couvrir une plus grande variété de combinaisons d'hyperparamètres en moins d'itérations.

```
set.seed(023)
grid <- expand.grid( sigma=runif(25,0.1,30) ,  C = runif(25, 0.1, 30))


tuned_model <- train(Outcome ~ ., data = train, method = 'svmRadial', tuneGrid = grid, trControl =
trainControl(method = 'cv', number = 5))
```

Affichons les meilleurs paramètres après l'entraînement

```
print(tuned_model$bestTune)
```

```
##       sigma        C
## 51 6.769878 2.575663
```

```r
confusionMatrix(tuned_model)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    0    1
##          0 64.7 34.9
##          1  0.2  0.3
##
##  Accuracy (average) : 0.6498
```

On peut voir l'**accuracy** moyen au cours de la recherche de nos paramètres. Il est de 0.75

Maintenant, on va construire le modèle final en utilisant les paramètres précédemment déterminés.

```r
final_model <- svm(Outcome ~ ., data = train, kernel = 'radial', cost = tuned_model$bestTune$C,probabili
```

On peut observer la performance du modèle sur les données d'entraînement

```r
Predictions <- predict(final_model, newdata = train)
conf_matrix <- table(Predictions, train$Outcome)
Accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste('Le modèle reussit sa classification à', Accuracy*100,'%'))
```

```
## [1] "Le modèle reussit sa classification à 85.1791530944625 %"
```

Une fois que le modèle est crée, passons à la prédiction sur les données de test.

3-3) Prédiction et évaluation du modèle

```r
predictions <- predict(final_model, newdata = test)
matrixx <- table(predictions, test$Outcome)
confusionMatrix(matrixx,mode = "everything")
```

```
## Confusion Matrix and Statistics
##
##
## predictions  0  1
##           0 84 22
##           1 18 30
##
##                Accuracy : 0.7403
##                  95% CI : (0.6635, 0.8075)
##     No Information Rate : 0.6623
##     P-Value [Acc > NIR] : 0.02326
```

```
##
##                   Kappa : 0.4081
##
##  Mcnemar's Test P-Value : 0.63526
##
##             Sensitivity : 0.8235
##             Specificity : 0.5769
##          Pos Pred Value : 0.7925
##          Neg Pred Value : 0.6250
##               Precision : 0.7925
##                  Recall : 0.8235
##                      F1 : 0.8077
##              Prevalence : 0.6623
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.6883
##       Balanced Accuracy : 0.7002
##
##        'Positive' Class : 0
##
```

Pour l'évaluation du modèle, on ne va pas utiliser uniquement l'exactitude, communément appelé **Accuracy**, à cause du déséquilibre au niveau des classes dans notre jeu de données: **65% des données sont celles de personnes non diabétiques**.

Le **recall** de notre modèle est de **82.35%**. Cela signifie que le modèle arrive à mieux prédire les cas de personnes sans diabète. Autrement, il y a de faible chance que le modèle se trompe sur le cas d'une personne qui n'a pas le diabète. Cependant le pourcentage de personne classé diabétique à tort est assez élevé. Soit

$$a = 1 - Specificité$$

$$a = 53.41$$

Globalement le modèle estimé permet d'avoir de bonne prédictions sur l'ensemble de données en prenant comme métrique le **Recall** et la **F1**.

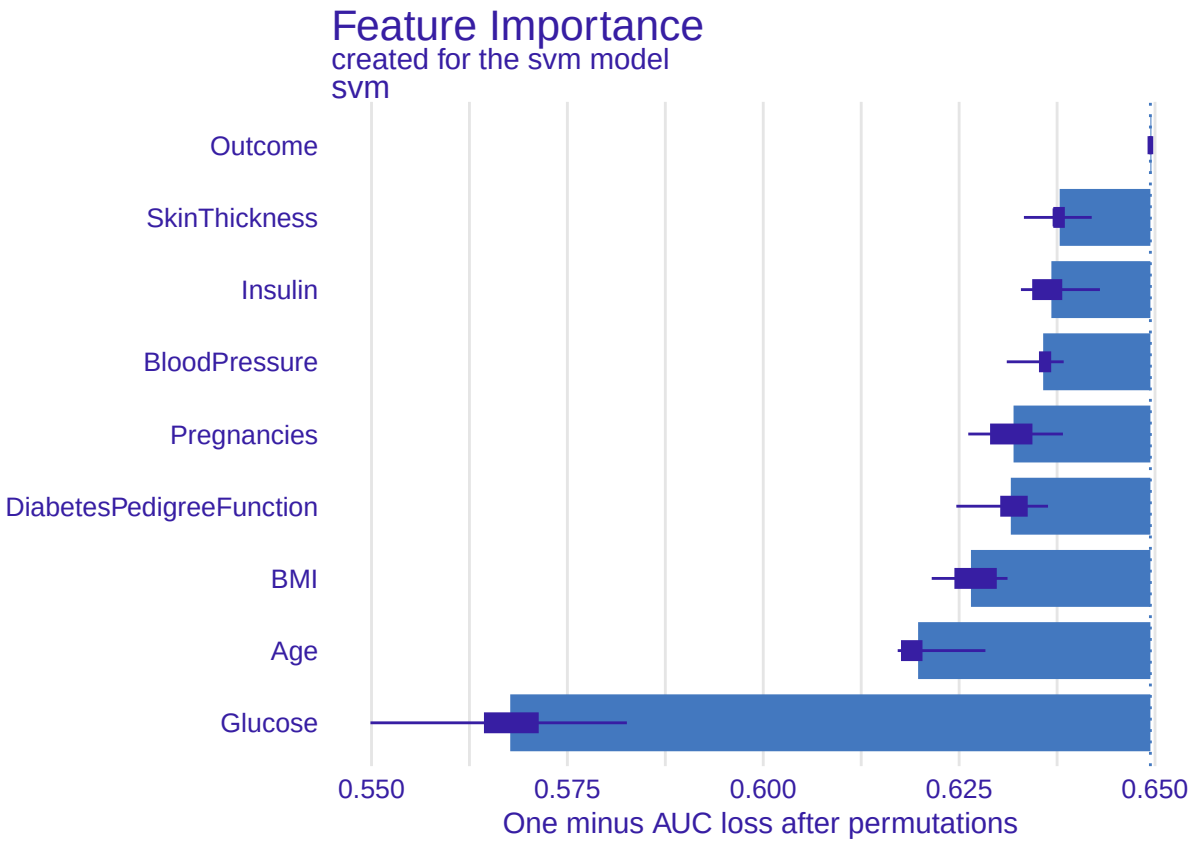
4- Les meilleurs variables prédictives

Grâce aux fonctions `explain` et `model_parts` de (Biecek 2018) nous allons ressortir les variables ayant le plus d'importance dans la conception de notre modèle.

```
ex <- explain(final_model,data=train, y=as.numeric( train$Outcome),predict_function = predict,predict_fu
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  svm  (  default  )
##   -> data              :  614  rows  9  cols
##   -> data              :  tibble converted into a data.frame
##   -> target variable   :  614  values
##   -> predict function  :  predict
##   -> predicted values  :  Predict function column set to:  2 1 1 1 2 2 1 1 1 1 2 2 2 1 1 2 1 1 2 1 1
##   -> model_info        :  package e1071 , ver. 1.7.13 , task classification (  default  )
##   -> predicted values  :  factor (  WARNING  ) with levels:  0, 1
##   A new explainer has been created!
```

```
plot(model_parts(ex))
```



Les variables **Glucose**,**Age** et **BMI** sont les plus importantes dans la constitution de notre modèle. Ce résultat vient confirmer les hypothèses posées au niveau de l'analyse descriptive avec `ggpairs`

5-Références

Biecek, Przemyslaw. 2018. “DALEX: Explainers for Complex Predictive Models in r” 19: 1–5. https://jmlr.org/papers/v19/18-416.html.

Schloerke, Barret, Di Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg, and Jason Crowley. 2021. “GGally: Extension to ’Ggplot2’.” https://CRAN.R-project.org/package=GGally.