Report_PimaDiabetes

Overview

The project intends to find a machine learning model which performs better than linear regression (the most basic and simple model) in diagnosing diabetes as well as find important variables relevant to diabetes. Here, two more advanced models are introduced and tested: Decsion Tree (DT) and eXreme Gradient Boosting (XGBoost).

About Dataset

The dataset used in the project is called ''PimaDiabetes". It is introduced in Kaggle and originally from the National Institute of Diabetes and Digestive and Kidney Diseases. In particular, all patients here are females at least 21 years old of Pima Indian heritage. It can be downloaded at my Github repository https://github.com/Ding-KI/HarvardX-PH125.9x-capstone or via my R script.

The dataset consists of 8 medical predictor variables and 1 target variable, `Outcome`. `Outcome` is a class variable (1 or 0), which means the patient has diabetes or not. Here are descriptions about 8 predictor variables:

- `Pregnancies` : Number of times pregnant
- `Glucose` : Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- `BloodPressure` : Diastolic blood pressure (mm Hg)
- `SkinThickness` : Triceps skin fold thickness (mm)
- `Insulin` : 2-Hour serum insulin (mu U/ml)
- `BMI` : Body mass index (weight in kg/(height in m)^2)
- `DiabetesPedigree` : Diabetes pedigree function
- `Age` : Age (years)

Key Steps

First, import, inspect and preprocess the data. Second, conduct the correlation analysis on variables. Third, set up the machine learning models and prune them. Final, visualize the evaluation of models to identify the best model to predict diagnose outcome.

Preprocessing Data

Load Revelant Packages

```
#library all needed packages
library(tidyverse)
library(ggplot2)
library(corrplot)
library(caret)
library(rpart)
library(treeheatr)
library(xgboost)
library(shapviz)
library(cowplot)
library(pROC)
```

Install and Import Dataset

```
#install dataset from my github repository
file_name <-"diabetes.csv"

#import dataset
```

```
PimaDiabetes <- read.csv(file_name, header=TRUE)
rm(file_name)
```

Inspect and Wrangle Dataset

```
#inspect dataset
str(PimaDiabetes)
```

```
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

There are some values of 0 in Glucose, BloodPressure, SkinThickness, Insulin and BMI, which indicates the missing value (NA). Convert them to NA and remove them.

```
PimaDiabetes <- PimaDiabetes %>% mutate_at(vars(2:8), ~na_if(.,0))
PimaDiabetes <- na.omit(PimaDiabetes)
```

Encode the 0 and 1 value in Outcome to “NonDiabetic” and “Diabetic”.

```
PimaDiabetes$Outcome <- as.factor(ifelse(PimaDiabetes$Outcome == 0,
                                         "NonDiabetic", "Diabetic"))
```

Check the preproccessed data again:

```
glimpse(PimaDiabetes)

Rows: 392
Columns: 9
$ Pregnancies             <int> 1, 0, 3, 2, 1, 5, 0, 1, 1, 3, 11, 10, 1, 1…
$ Glucose                 <int> 89, 137, 78, 197, 189, 166, 118, 103, 115,…
$ BloodPressure           <int> 66, 40, 50, 70, 60, 72, 84, 30, 70, 88, 94…
$ SkinThickness           <int> 23, 35, 32, 45, 23, 19, 47, 38, 30, 41, 33…
$ Insulin                 <int> 94, 168, 88, 543, 846, 175, 230, 83, 96, 2…
$ BMI                     <dbl> 28.1, 43.1, 31.0, 30.5, 30.1, 25.8, 45.8, …
$ DiabetesPedigreeFunction <dbl> 0.167, 2.288, 0.248, 0.158, 0.398, 0.587, …
$ Age                     <int> 21, 33, 26, 53, 59, 51, 31, 33, 32, 27, 51…
$ Outcome                 <fct> NonDiabetic, Diabetic, Diabetic, Diabetic,…
```

Correlation Analysis

Analysis on Age, BMI, Glucose and Diabetes

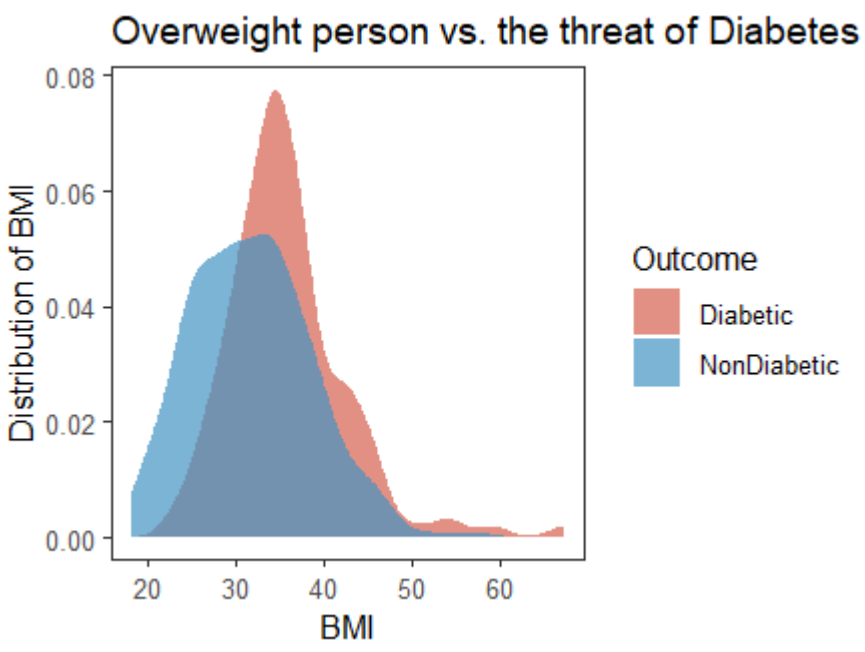
According to National Insititutes of Health (NIH), risk factors for developing diabetes include family history, race, age, weight and so on. We select two key features: Age and BMI, as combined conditions in the glucose level and diagnosis outcome to observe patterns and trends. Here are the description given by NIH that age and weight are risk factor:

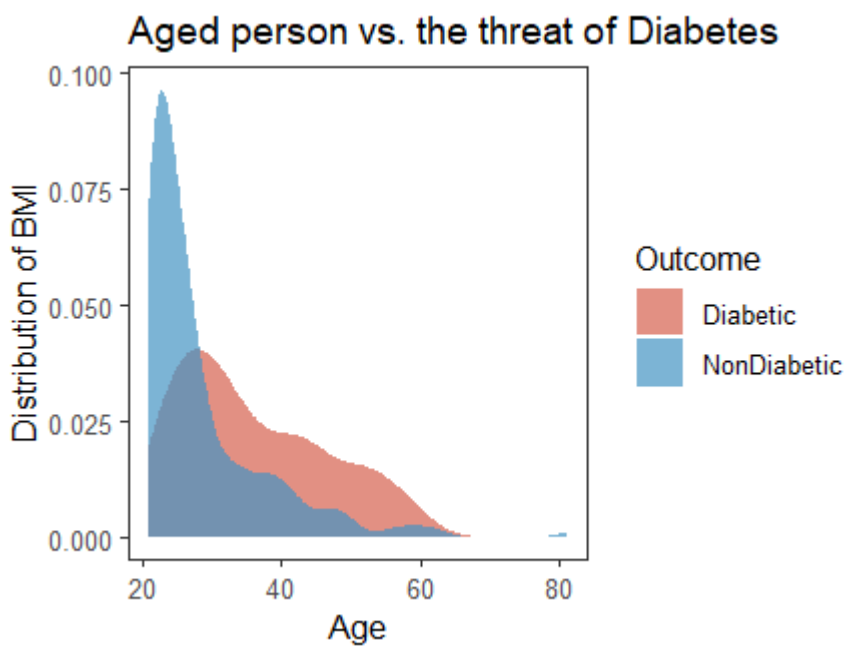
- Weight: have overweight or obesity
- Age: age 45 or older

Distribution of BMI and Age

First, make a density plot of BMI and age respectively in nondiabetes and diabetes:

```
ggplot(PimaDiabetes, aes(BMI, fill = Outcome)) +
  geom_density(color = NA, alpha = 0.7) + ylab("Distribution of BMI") +
  ggtitle("Overweight person vs. the threat of Diabetes")+
  scale_fill_manual(values = c("Diabetic" = "#D6604D", "NonDiabetic" = "#4393C3")) +
  theme_bw()+
  theme(panel.grid.major = element_blank(),  #remove all the grid lines
        panel.grid.minor = element_blank())

ggplot(PimaDiabetes, aes(Age, fill = Outcome)) +
  geom_density(color = NA, alpha = 0.7) + ylab("Distribution of BMI") +
  ggtitle("Aged person vs. the threat of Diabetes")+
  scale_fill_manual(values = c("Diabetic" = "#D6604D", "NonDiabetic" = "#4393C3")) +
  theme_bw()+
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())
```





According two density plots, The BMI distribution for non-diabetic individuals (blue) is broader, covering a wider range of BMI values, with the peak at a lower BMI (around 30). In contrast, the BMI distribution for diabetic individuals (red) is narrower, with the peak at a higher BMI (around 35). Overall, diabetic individuals have higher BMI values compared to non-diabetic individuals.

Similarly, non-diabetic individuals (blue) have a higher and sharper peak at a younger age, around 20 years old, and their distribution decreases sharply as age increases. Diabetic individuals (red) have a broader distribution, and their presence starts to increase from the mid-20s to 30s, continuing steadily across older age groups. It suggests that the proportion of diabetic individuals increases with age.

Ballon Plot: BMI and Age for Glucose Level

According to WHO, the criteria of weight status based on BMI is: Underweight(<18.5), Healthy Weight(18.5-24.9), Overweight(25.0-29.9), Obese Class 1(30.0-34.9), Obese Class 2(35.0-39.9) and Obese Class 3(≥40). Convert the BMI level to weight status, as well as group by age, for later bubble plot:

```
PimaDiabetes <- PimaDiabetes %>% mutate(
  BMI_category = case_when(
    BMI <25 ~ "Healthy/Underweight",
    BMI>=25 & BMI <30 ~ "Overweight",
    BMI>=30 & BMI <35 ~ "Obesity I",
    BMI>=35 & BMI <40 ~ "Obesity II",
    BMI>=40 ~ "Obesity III"
  ),
  Age_category = case_when(
```

```
    Age < 30 ~ "<30",
    Age >= 30 & Age < 40 ~ "30-40",
    Age >= 40 & Age < 50 ~ "40-50",
    Age >= 50 & Age < 60 ~ "50-60",
    Age >= 60  ~ ">60",
  )
)%>%
  mutate(Age_category = factor(Age_category, levels = c
                               ("<30", "30-40", "40-50", "50-60", ">60")),
         BMI_category = factor(BMI_category, levels = c
                               ("Healthy/Underweight","Overweight",
                                 "Obesity I","Obesity II","Obesity III"))
         ) %>%
  arrange(Age_category,BMI_category)
```

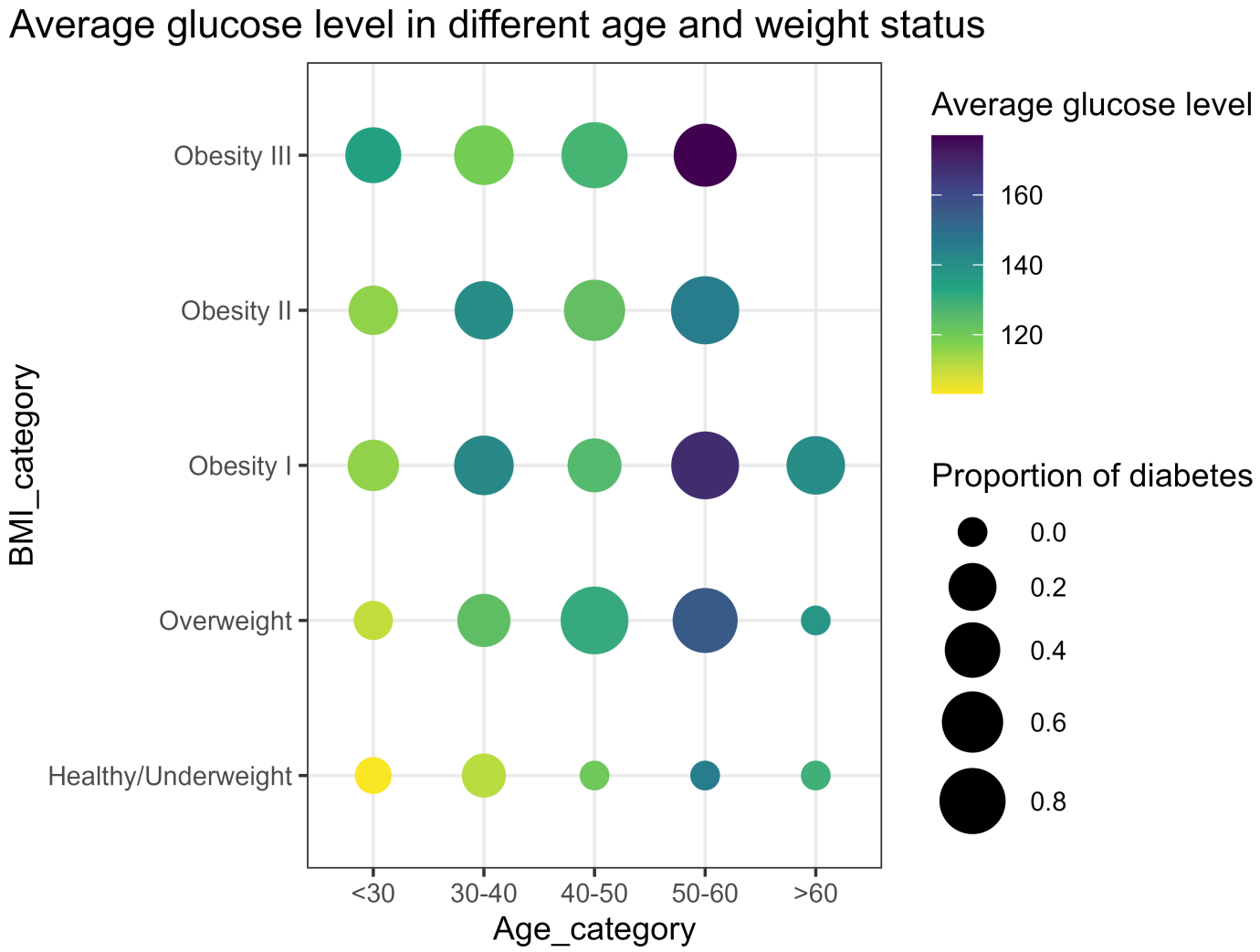
Hence the dataset is divided by age and BMI into 23 groups (There is no person with age>60 meanwhile obesity II or III, so that the amount of groups is 5*5-2=23) .Calculate the average glucose level and the proportion of diabetes in each groups:

```
Pima_in_Groups <- PimaDiabetes %>%
  group_by(BMI_category,Age_category) %>%
  summarise(n=n(),
            avgGlucose = mean(Glucose),
            PropDiabetes = sum(Outcome=="Diabetic")/n(),
            .groups = "drop")

print(Pima_in_Groups,n=23)
```

```
# A tibble: 23 × 5
   BMI_category        Age_category     n avgGlucose PropDiabetes
   <fct>               <fct>        <int>      <dbl>        <dbl>
 1 Healthy/Underweight <30             34       103.       0.0294
 2 Healthy/Underweight 30-40            8       111.       0.125
 3 Healthy/Underweight 40-50            1       120        0
 4 Healthy/Underweight 50-60            1       145        0
 5 Healthy/Underweight >60              1       129        0
 6 Overweight          <30             56       110.       0.0536
 7 Overweight          30-40           15       123.       0.333
 8 Overweight          40-50            8       131.       0.875
 9 Overweight          50-60            4       155.       0.75
10 Overweight          >60              2       138        0
11 Obesity I           <30             67       116.       0.284
12 Obesity I           30-40           26       142.       0.538
13 Obesity I           40-50           17       126.       0.353
14 Obesity I           50-60            8       168.       0.875
15 Obesity I           >60              2       141        0.5
16 Obesity II          <30             47       116.       0.234
17 Obesity II          30-40           16       140.       0.5
18 Obesity II          40-50           15       123.       0.6
19 Obesity II          50-60            9       146.       0.889
20 Obesity III         <30             34       134.       0.412
21 Obesity III         30-40           13       120.       0.538
22 Obesity III         40-50            5       128.       0.8
23 Obesity III         50-60            3       177        0.667
```

Make the bubble plot based this grouped data:

```
Bubble <- Pima_in_Groups %>% ggplot(aes(x=Age_category,y=BMI_category))+
  geom_point(aes(size = PropDiabetes, color=avgGlucose))+
  scale_size_continuous(name = "Proportion of diabetes",
                        range = c(4,10))+
  scale_color_viridis_c(name = "Average glucose level",
                        direction = -1,)+
  theme_bw()+
  labs(title="Average glucose level in different age and weight status")+
  theme(plot.title.position = "plot")

ggsave("Bubble.png", Bubble, dpi = 600, width = 6, height = 4.5)
```

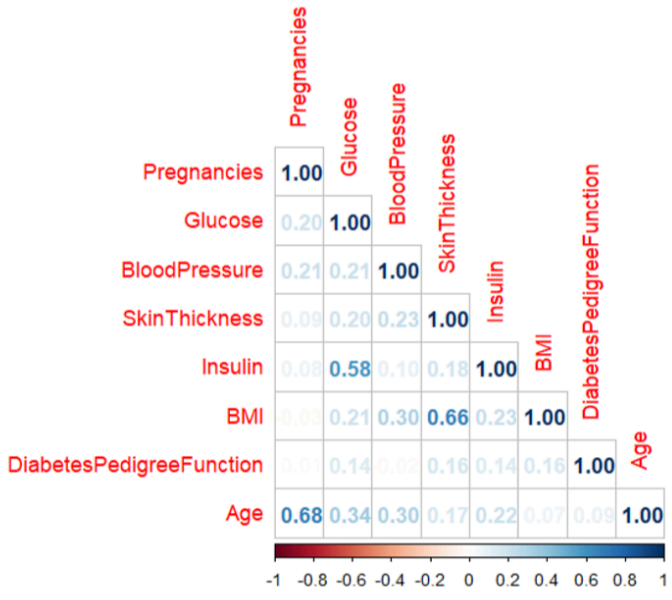


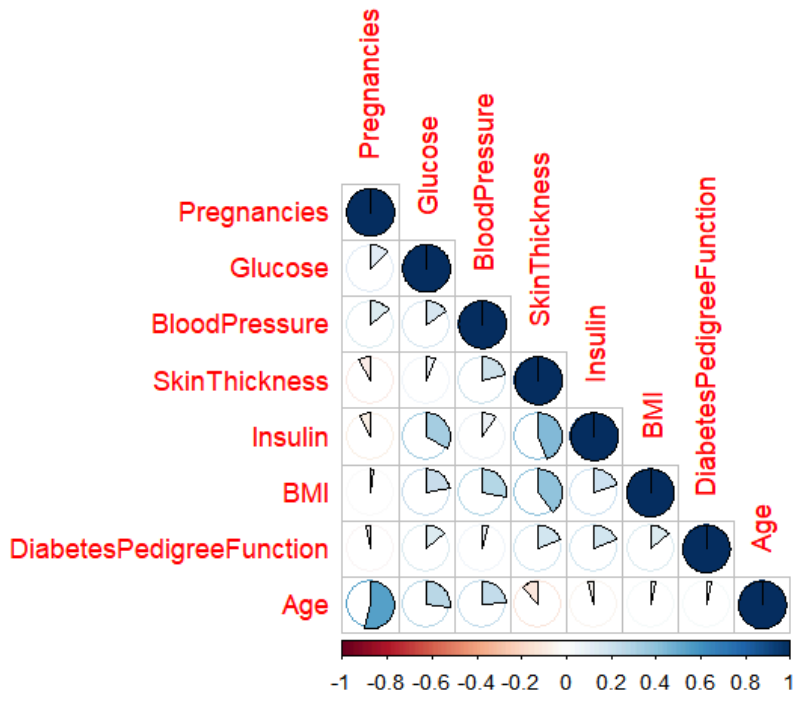
The bubble plot suggests that individuals aged 50-60, with Obesity I or III, is most likely to have diabetes, while they also exhibit high glucose level excess exceeding 160 mmol/L in glucose tolerance test. By contrast, healthy or underweighted individuals, are less likely to be diabetic.

Correlogram for each predicator variables

Make correlograms to find the correlation coefficient between each pair predicator variables. As correlograms shown, none of features has a coefficient of 0.7 or higher, so all features should be considered in machine learning model.

```
corrplot(cor(PimaDiabetes[,-9]),
         type = "lower",
         method = "number")

corrplot(cor(PimaDiabetes[,-9]),
         type = "lower",
         method = "pie")
```





Machine Learning

Create training set and test set

Remove the col 'BMI_category' and 'Age_category', and divide the dataset into 80% training and 20% test subset with Caret package.

```
PimaDiabetes <- PimaDiabetes[-c(10,11)]

set.seed(123)
index <- createDataPartition(PimaDiabetes$Outcome, p=0.8, list=FALSE)
pimaTrain <- PimaDiabetes[index,]
pimaTest <- PimaDiabetes[-index,]
```

Inspect the training set and test set

Check the dimension of subsets and whether the diabetes rate is equal between subsets. The results suggests that all is right.

```
 dim(pimaTest)
[1] 78  9
 dim(pimaTrain)
[1] 314   9

mean(pimaTrain$Outcome == "Diabetic")
[1] 0.3312102
mean(pimaTest$Outcome == "Diabetic")
[1] 0.3333333
```

Cross-validation

Selected 'cross-validation' as the method of cross-validation. The dataset will be randomly divived into 10 subset, one of which will be used as the validation set each time, and the remaining 9 subsets will be used as the training set. Also, specifies the class probabilities are calculated in classfication problems, as well as twoClassSummary as the summary function for model perfomance in this binary classification problems. It's reconmmanded to use "repeatedcv" method but it requires a lot of time to run.

Assign all mentioned parametres to 'Folds':

```
Folds <- trainControl(method = "cv",
                      number = 10,
```

```
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary,
                        verboseIter = TRUE,
                        allowParallel = TRUE)
```

Decision Tree

Train a primary DT model

Build the primary decision tree model, then evaluate the trained model on the test set as follow:

```
DT <- rpart(Outcome ~., data = pimaTrain, method = "class")
DT_pred <- predict(DT,newdata = pimaTest, type = "class")
confusionMatrix(DT_pred,pimaTest$Outcome)
```

Given the confusion matrix, out of 78 observations, it wrongly predicts 18 observations. The primary DT model has achived 76.92% accuracy.

```
Confusion Matrix and Statistics

             Reference
Prediction    Diabetic NonDiabetic
  Diabetic          15           7
  NonDiabetic       11          45

               Accuracy : 0.7692
                 95% CI : (0.66, 0.8571)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 0.03295
```

Splitting Criteria Comparsion

There are two common splitting criteria used in decision tree model: one is called “gini” which is used in rpart package by default, another is “entropy”. The splitting criteria is supplied using parms argument as a list. Here, “gini” and “entropy” are used in model and tested for accuracy.

```
DT_gini <- rpart(Outcome ~.,
                 data = pimaTrain,
                 method = "class",
                 parms = list(split = "gini"))

DT_entropy <- rpart(Outcome ~.,
                    data = pimaTrain,
                    method = "class",
                    parms = list(split = "entropy"))
```

The result suggests that there is no difference in accuracy between two spiliting criteria. Therefore, “gini” will be used by default in next step.

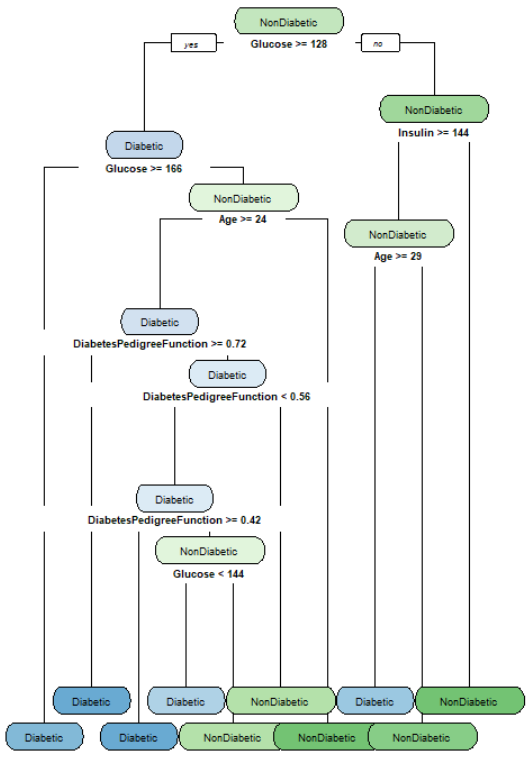
```
DT_gini_pred <- predict(DT_gini, newdata = pimaTest, type = "class")
mean(DT_gini_pred == pimaTest$Outcome)

[1] 0.7692308

DT_entropy_pred <- predict(DT_entropy, newdata = pimaTest, type = "class")
mean(DT_entropy_pred == pimaTest$Outcome)
```

```
[1] 0.7692308
```

Prune the DT model

Plot the DT model, and it is obvious that primary DT tree model, the tree structure is deep and fragile:

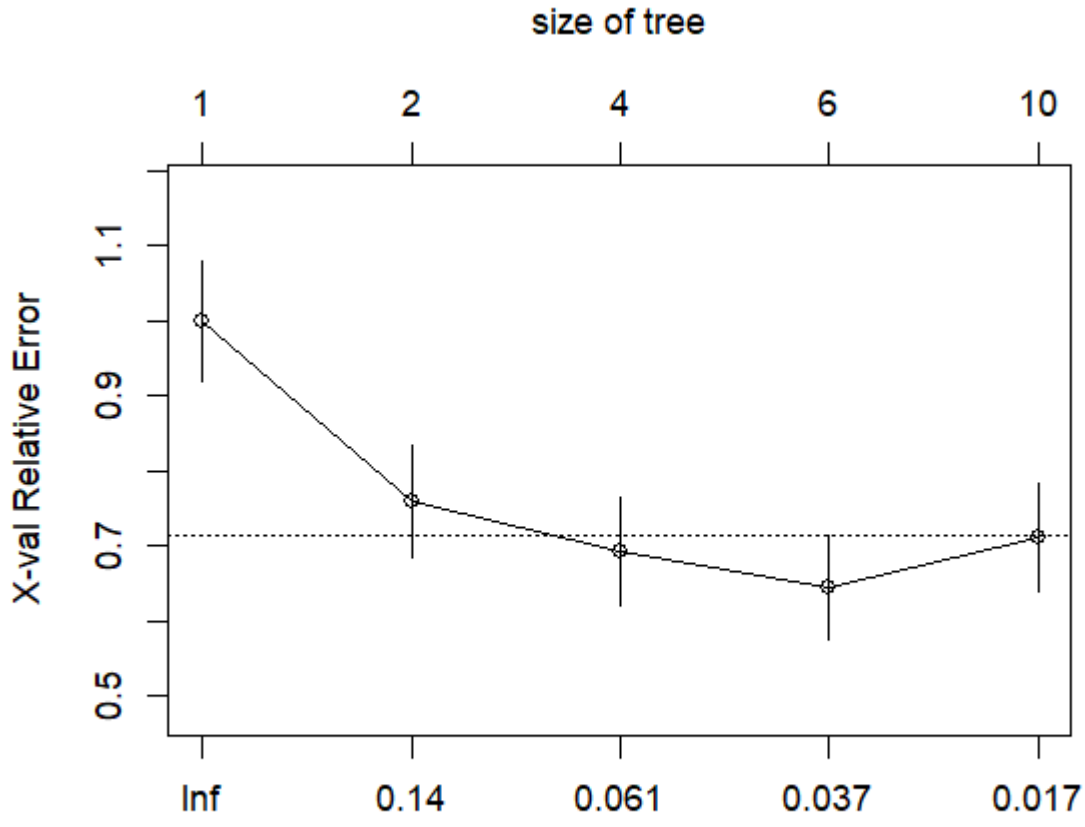
```
rpart.plot(DT,extra=0, type=2,cex = 0.5)
```



It needs to be more interpretation while keep performance. One popular way is adjust complexity parameter (CP) to pruning the tree, by reducing branches. To see the error vs CP:

```
DT_gini$cptable

          CP nsplit rel error    xerror       xstd
1 0.25961538      0 1.0000000 1.0000000 0.08019147
2 0.07692308      1 0.7403846 0.7884615 0.07484327
3 0.04807692      3 0.5865385 0.7403846 0.07330294
4 0.02884615      5 0.4903846 0.6538462 0.07018169
5 0.01000000      9 0.3750000 0.6442308 0.06980519
```

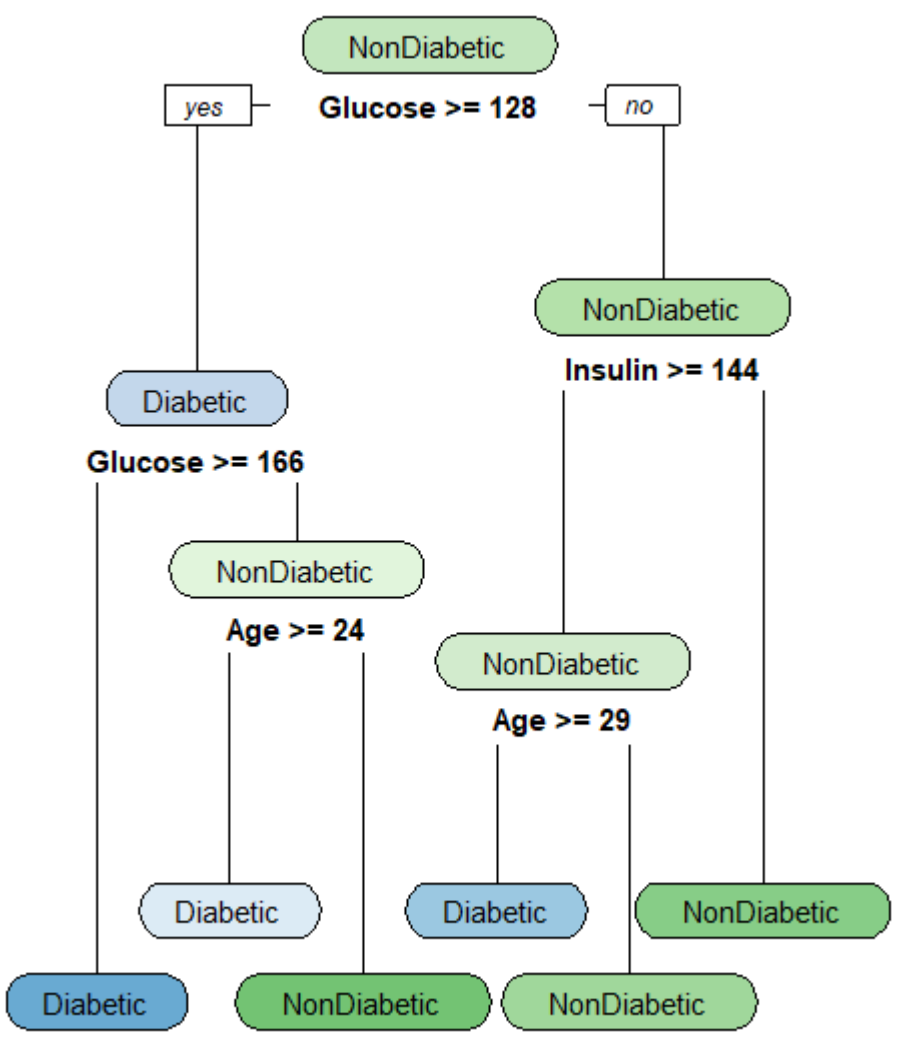
```
plotcp(DT)
```



What's the best CP? Someone will select the CP value that induced the least X error 0.01, while someone use 1SE rule: determine the threshold as Xerror+1*SE, then select the simplest model with Xerror under threshold. In this case, the minmum Xerror is 0.6442308 (for CP=0.01, nsplit=9), so that the threshold is 0.714036. The simplest model will be with cp=0.02884615.

Pass the cp parameter to the model training and calculate the optimal accuracy:

```
DT_gini_optimal <- rpart(Outcome ~., data = pimaTrain,
                                                  method = "class",cp=0.02884615)
DTplot <- rpart.plot(x=DT_gini_optimal,extra=0, type=2)
ggsave("DTplot.png", dpi=600,width=3,height=4.5)
```



```
DT_gini_best_pred <- predict(DT_gini_best,newdata = pimaTest, type = "class")
mean(DT_gini_best_pred == pimaTest$Outcome)

[1] 0.7820513
```

Tune the DT model

Grid search is a process that systematically explores a predefined subset of the hyperparameter space for the targeted algorithm, evaluating the cost function for each generated hyperparameter set. Here the minsplit and maxdepth will be ajusted:

- minsplit: the minimum number of observations that must exist in the node in order for a split to be attempted.
- maxdepth: The maximum depth of any node of the final tree.

Generate a sequence 1 to 20 for both minsplit and maxdepth, then combine them to be a grid. Use the first loop to generate models using various hyperparameter combination, and the second loop to calculate their test accuracy.

```
DT_hyper_grid <- expand.grid(minsplit=seq(1,20,1),
                             maxdepth=seq(1,20,1))
DT_hyper_tune <- list()
DT_hyper_tune_accuracy <- c()

for (i in 1:400) {

  minsplit <- DT_hyper_grid$minsplit[i]
  maxdepth <- DT_hyper_grid$maxdepth[i]

  DT_hyper_tune[[i]] <- rpart(formula = Outcome ~ .,
                              data = pimaTrain,
                              method = "class",
                              minsplit = minsplit,
                              maxdepth = maxdepth)}

for (i in 1:400) {
  model <- DT_hyper_tune[[i]]
  pred <- predict(object = model,
                  newdata = pimaTest[,1:8],
                  type = "class")
  DT_hyper_tune_accuracy[i] <- signif(mean(pimaTest$Outcome==pred),10)}
```

To find which model achieves the highest accuracy:

```
which.max(DT_hyper_tune_accuracy)
[1] 1
```

The hyperparameter of the best model:

```
DT_hyper_tune[[1]]$control

$minsplit
[1] 1

$minbucket
[1] 0

$cp
[1] 0.03

$maxcompete
```

```
[1] 4

$maxsurrogate
[1] 5

$usesurrogate
[1] 2

$surrogatestyle
[1] 0

$maxdepth
[1] 1

$xval
[1] 10
```

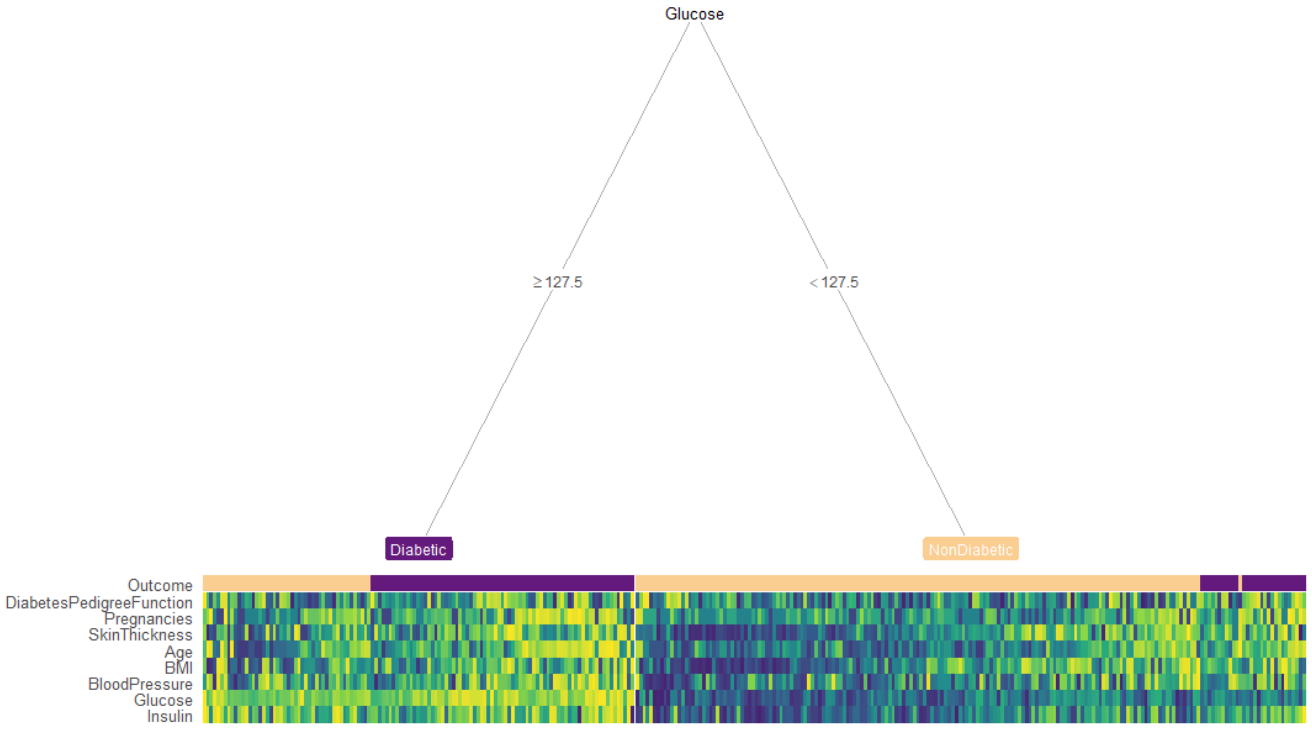
Use the hyperparameter combination to train DT model again, assign it directly to “DT”.

```
DT <- rpart(Outcome ~., data = pimaTrain, method = "class",
            minsplit=1,maxdepth=1,cp=0.02884615)
DT_pred <- predict(DT,newdata = pimaTest, type = "class")
confusionMatrix(DT_pred,pimaTest$Outcome)
```

```
Confusion Matrix and Statistics

             Reference
Prediction    Diabetic NonDiabetic
  Diabetic          19           9
  NonDiabetic        7          43

               Accuracy : 0.7949
                 95% CI : (0.6884, 0.878)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 0.009224

                  Kappa : 0.5472

 Mcnemar's Test P-Value : 0.802587

            Sensitivity : 0.7308
            Specificity : 0.8269
         Pos Pred Value : 0.6786
         Neg Pred Value : 0.8600
             Prevalence : 0.3333
         Detection Rate : 0.2436
   Detection Prevalence : 0.3590
      Balanced Accuracy : 0.7788

       'Positive' Class : Diabetic
```

Now the accuracy is 79.49%, higher than the primary one by 2.57%. It is a slight promotion. To plot the final DT model more vividly, use R package treeheatr:

```
heat_tree(partykit::as.party(DT))
```



XGBoost model

Train a primary XGBoost model

Build the primary decision tree model, it needs eight predicator variables to combined as a matrix, and target variable 'Outcome' to be 0 and 1. Revert the Outcome of 'Diabetic' and 'NonDiabetic' to 1 and 0. Before calculate the accuracy, translate the prediction into label: The prediction is the probability of positive/diabetic. If the value of prediction>0.5 (which means the outcome are more likely to be Diabetic), labeled it as '1'; otherwise, labelled it as '0'.

```
XGB <- xgboost(data = as.matrix(pimaTrain[,1:8]),
               label = as.numeric(pimaTrain$Outcome)-1,
               booster = "gbtree",
               objective = "binary:logistic",
               nrounds = 100,
               verbose = 0)
```

```
XGB_pred <-predict(XGB, as.matrix(pimaTest[,1:8]),type="response")
XGB_pred_label <- ifelse(XGB_pred > 0.5, 1, 0)
mean(XGB_pred_label == (as.numeric(pimaTest$Outcome) - 1))

[1] 0.7564103
```

Tune the XGBoost model

Conduct a grid research to find the best set of hyperparameters. 6 hyperparameter wil be used in grid search: nrounds, eta, max_depth, gamma, colsample_tree, min_child_weight. It takes about 25 minutes to evaluate 3*4*3*3*3*3=972 combinations with 16GB Memory.

```
XGB_hyper_grid <- expand.grid(
  nrounds = seq(100,300,100),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(4, 5, 6),
  gamma = c(0, 1, 2),
  colsample_bytree = c(0.5, 0.75, 1.0),
  min_child_weight = c(1, 3, 5),
  subsample = 1
)

XGB_findtune <- train(
  x = as.matrix(pimaTrain[, 1:8]),
  y = as.factor(pimaTrain$Outcome),
```

```
  method = "xgbTree",
  trControl = Folds,
  tuneGrid = XGB_hyper_grid,
  metric = "ROC",
  verbose = TRUE
)
```

This prints out the best values found in the grid search:

```
XGB_findtune$bestTune

    nrounds max_depth   eta gamma colsample_bytree min_child_weight subsample
136     100         5 0.025     2              0.5                1         1
```

Use those values to train XGBoost model again:

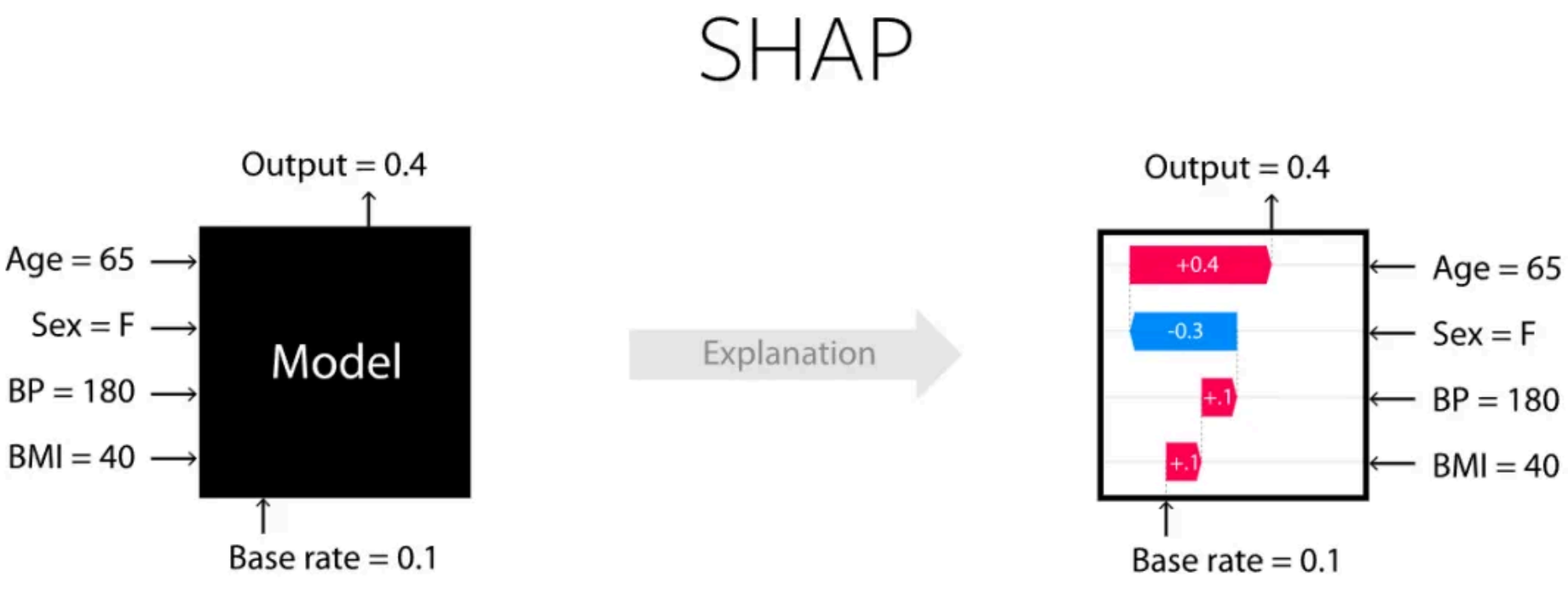
```
XGB <- xgboost(data = as.matrix(pimaTrain[,1:8]),
               label = as.numeric(pimaTrain$Outcome)-1,
               booster = "gbtree",
               objective = "binary:logistic",
               nrounds = 100,
               max_depth = 5,
               eta = 0.025,
               gamma = 2,
               colsample_bytree = 0.5,
               min_child_weight = 1,
               subsample = 1,
               scale_pos_weight = 0.5,
               verbose = 0)
```

Make the prediction and calculate the accuracy. It achieves 82.05%, a remarkble optimization.

```
XGB_pred <-predict(XGB, as.matrix(pimaTest[,1:8]),type="response")
XGB_pred <- ifelse(XGB_pred > 0.5, 1, 0)
mean(XGB_pred == (as.numeric(pimaTest$Outcome) - 1))

[1] 0.8205128
```

Interpret the model with SHAP

XGboost had been a “black-box” model until “shapley value” was introduced. Without SHAP value, it is almost impossible to understand how variables affect on the final model. That is, SHAP provides solid and interpretable explaination for researchers.



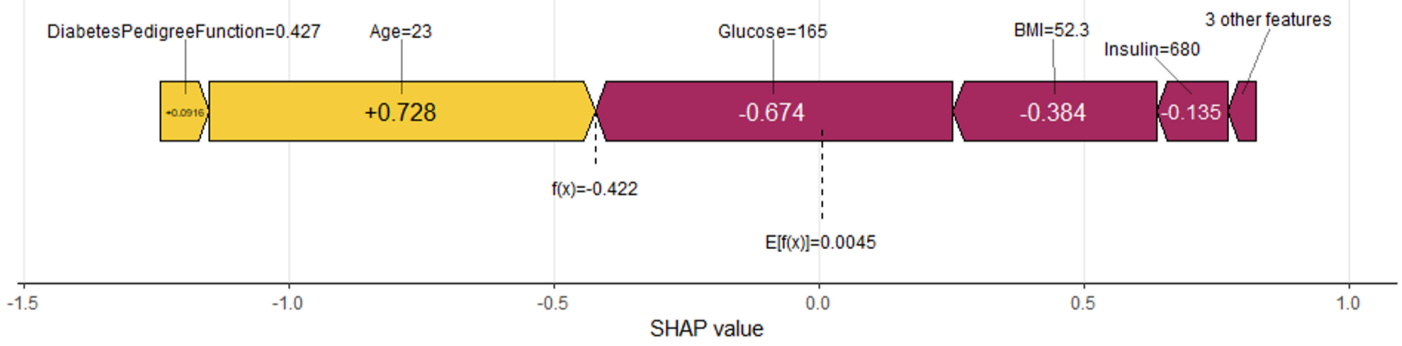
The interpretation of the SHAP is: Given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value. Here are four key features of SHAP: Efficiency, Symmetry, Dummy and Addivity.
At the current stage, most implementations of SHAP are based on Python. With the growing popularity of the algorithm, R has also introduced SHAP-related explanations. However, in R, the primary package for SHAP explanations is `shapviz`. This package is limited to interpreting models from XGBoost, LightGBM, and H2O, and does not support other machine learning models.

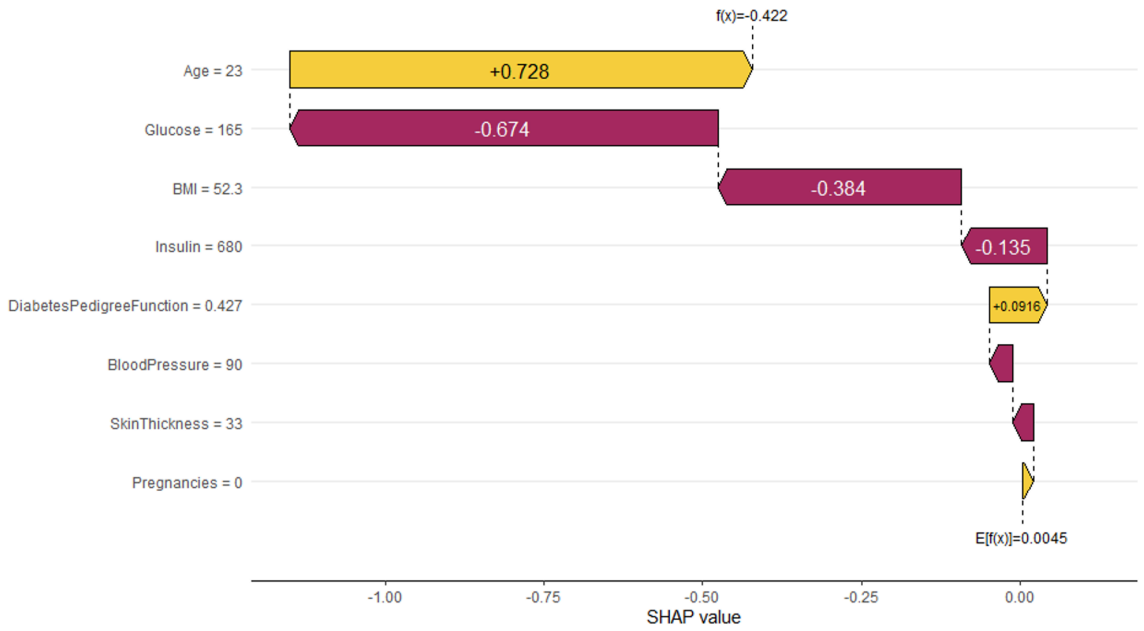
Calculate SHAP with shapviz:

```
shap_result <- shapviz(XGB, X_pred = as.matrix(pimaTrain[,1:8]))
```

Use the SHAP to see how variables affect on the XGB model to predict outcome, take the 100th individuals in train set:

```
sv_force(shap_result, row_id = 100)
sv_waterfall(shap_result,row_id = 100)
```



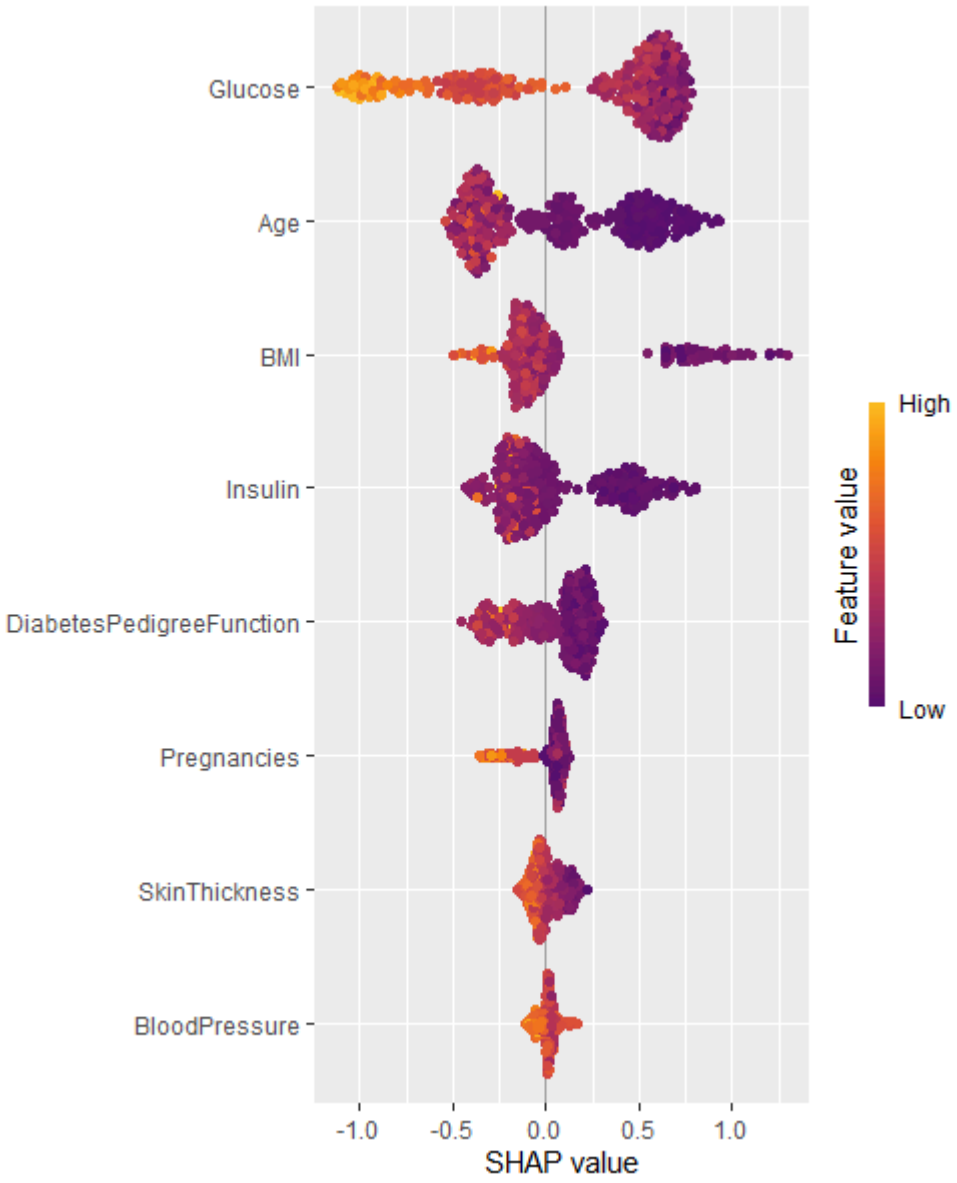


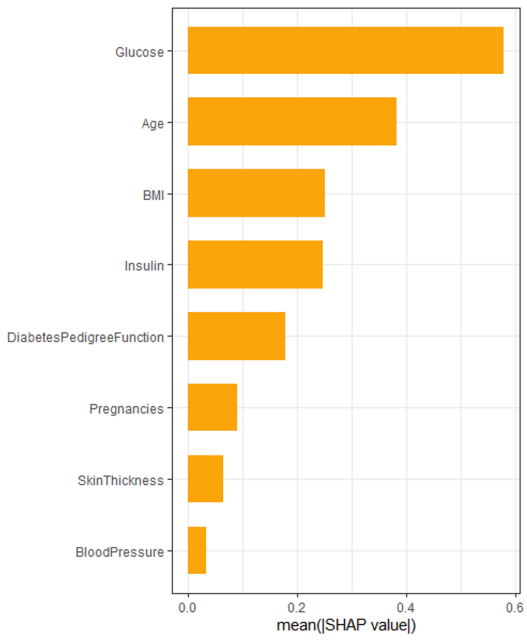
Sum up SHAP value of each variables of the individual, and get the final SHAP value (E(f(x)) of 0.0045- less than 50%, which means the individual is unlikely to be diabetic.

To find the contribution of variables (importance):

```
sv_importance(shap_result, kind = "beeswarm")
sv_importance(shap_result)+theme_bw()
```

Similarly with DT model, the glucose is the most important variable, followed by Age and BMI. It is also accordant with the result of Correlation Analysis.





Logistic Linear Regression

Logistic linear regression model can be used directly in package caret:

```
GLM <- train(Outcome~., data = pimaTrain, method = "glm",
             family =binomial,trControl=Folds)
```

Evaluate the model based on accuracy:

```
GLM_pred <- predict(GLM, newdata = pimaTest[,1:8])
mean(GLM_pred==pimaTest$Outcome)

[1] 0.8076923
```

The accuracy is lower than one of XGBoost, but higher the one of Decsion Tree. Next, look into the coefficient factors table:

```
summary(GLM)$coef

                            Estimate  Std. Error     z value
(Intercept)              10.4939233986 1.402843733  7.48046497
Pregnancies              -0.1096102741 0.063112794 -1.73673620
Glucose                  -0.0384236297 0.006578605 -5.84069522
BloodPressure             0.0005848601 0.013285102  0.04402376
SkinThickness            -0.0111204033 0.019150356 -0.58068912
Insulin                   0.0011012606 0.001446409  0.76137585
BMI                      -0.0773854212 0.030079061 -2.57273393
DiabetesPedigreeFunction -1.1476686992 0.487644516 -2.35349453
Age                      -0.0352189281 0.021030712 -1.67464272
                            Pr(>|z|)
(Intercept)              7.406011e-14
```

```
Pregnancies              8.243375e-02
Glucose                  5.198343e-09
BloodPressure            9.648855e-01
SkinThickness            5.614500e-01
Insulin                  4.464326e-01
BMI                      1.008987e-02
DiabetesPedigreeFunction 1.859788e-02
Age                      9.400435e-02
```

How to understand the coef table:

- Estimate: the estimation of β0 - the contribution of indicator variables to target variable.
- Std. Error: the standard error of estimate, the less Std. Error, the more reliable estimate is
- z.value: to test the estimate whther equal to 0 or not
- Pr(>|z|): the p value, if p <0.05, the correspoding variable is significant.

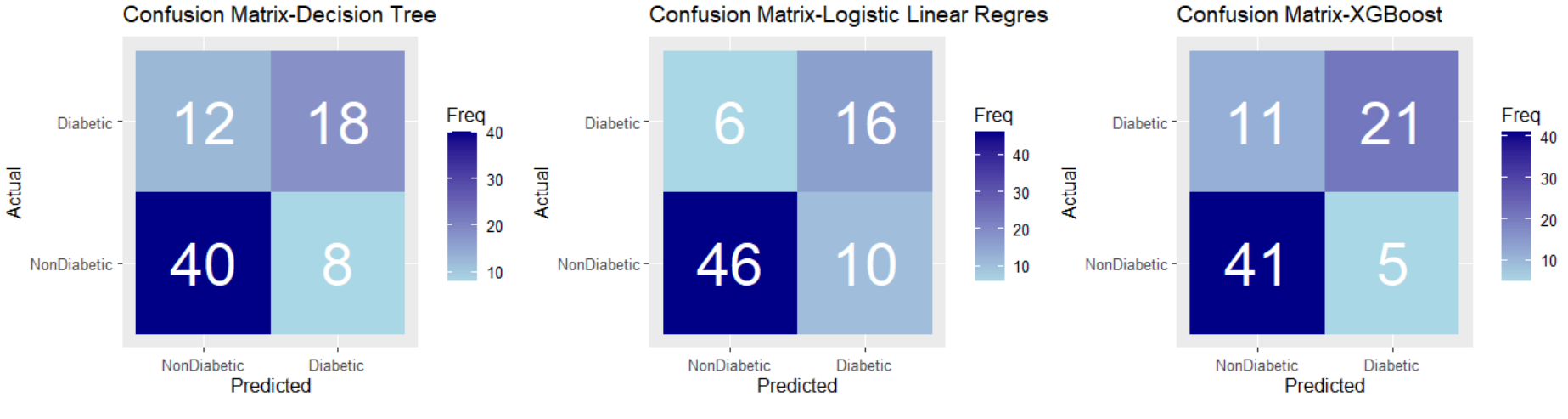
Therefore, glucose, BMI and diabetes pedigree function are significant variables with p.value <0.05. Meanwhile, blood pressure, skin thickness and insulin are insignificant variable with p.value >0.05.

Model Comparasion

plot confusion matrix of three models:

```
cf_DT <- as.data.frame(confusionMatrix(DT_pred, pimaTest$Outcome)$table)
colnames(cf_DT) <- c("Actual", "Predicted", "Freq")
cf_DT$Actual <- factor(cf_DT$Actual, levels = c("NonDiabetic","Diabetic" ))
cf_DT$Predicted <- factor(cf_DT$Predicted, levels = c("NonDiabetic","Diabetic" ))

cfp_DT <- ggplot(cf_DT, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 12) +
  labs(title = "Confusion Matrix-Decision Tree") +
  xlab("Predicted") +
  ylab("Actual") +
  scale_fill_gradient(low = "lightblue", high = "darkblue")

XGB_pred <- factor(XGB_pred, levels=c(1,0), labels=c("NonDiabetic", "Diabetic"))
cf_XGB <- as.data.frame(confusionMatrix(XGB_pred,pimaTest$Outcome)$table)
colnames(cf_XGB) <- c("Actual", "Predicted", "Freq")
cf_XGB$Actual <- factor(cf_XGB$Actual, levels = c("NonDiabetic","Diabetic" ))
cf_XGB$Predicted <- factor(cf_XGB$Predicted, levels = c("NonDiabetic","Diabetic" ))

cfp_XGB <- ggplot(cf_XGB, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 12) +
  labs(title = "Confusion Matrix-XGBoost") +
  xlab("Predicted") +
  ylab("Actual") +
  scale_fill_gradient(low = "lightblue", high = "darkblue")

cf_GLM <- as.data.frame(confusionMatrix(GLM_pred, pimaTest$Outcome)$table)
colnames(cf_GLM) <- c("Actual", "Predicted", "Freq")
cf_GLM$Actual <- factor(cf_DT$Actual, levels = c("NonDiabetic","Diabetic" ))
cf_GLM$Predicted <- factor(cf_DT$Predicted, levels = c("NonDiabetic","Diabetic" ))

cfp_GLM <- ggplot(cf_GLM, aes(x = Predicted, y = Actual, fill = Freq)) +
  geom_tile() +
  geom_text(aes(label = Freq), color = "white", size = 12) +
```

```
  labs(title = "Confusion Matrix-Logistic Linear Regression") +
  xlab("Predicted") +
  ylab("Actual") +
  scale_fill_gradient(low = "lightblue", high = "darkblue")

plot_grid(cfp_DT,cfp_GLM,cfp_XGB,nrow = 1)
```



plot the barplot of accuracies:

```
accuracies <- data.frame(Model=c("DT","XGB","GLM"),
                         Accuracy=c(0.7949,0.8205128,0.8076923))
ggplot(accuracies,aes(x=Model,y=Accuracy,fill=Model))+
  geom_bar(stat='identity',width = 0.6)+
  geom_text(aes(label=round(Accuracy,3)),hjust=1,vjust=-6,size=3.5)+
  scale_fill_manual(values = c("lightblue", "cornflowerblue", "midnightblue")) +
  xlab("Model")+
  theme_bw()+
  theme(legend.position = "none")
```

Given the plot, XGBoost model has the highest accuracy, Logistic Regression owns the second one, while DT performs worst on predicting diagnosis outcome.

Insight into XGBoost, SHAP value suggests that glucose, age and BMI are top three most important variables, which is allgned with the information provided by WHO and NIH: age and BMI are risk factors in developing diabetes, while the glucose level indicates one's pancreatic β-cell function and ability to regulate blood sugar.

To seek for better method to predict, LightGBM, Catboost and so on are recommanded. However, compared to Python, R is less flexible and versatile in machine learning. In fact, Tensorflow and Pytorch are mainstream platform now.