Diabetes Diagnosis Project

Daping Zhang

8 June 2022

1. Introduction

This project is to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not? We will use the Pima Indians Diabetes Database, obtained from Kaggle. The dataset consists of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. We will do some analysis and necessary pro-processing on the dataset, then try to build some models using some methods and compare them with the accuracy of predictions. Finally, a model with the highest prediction accuracy is selected.

2. Importing the libraries and dataset

```
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(matrixStats)
library(corrplot)
library(factoextra)
library(randomForest)
library(GGally)
library(gam)
```

Pima Indians Diabetes Database:

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/download/

```
# importing the dataset
urlfile = "https://raw.githubusercontent.com/dapingz/Diabetes-Project/main/diabetes.csv"

# loading the dataset as data frame
diab_dat <- as.data.frame(read_csv(url(urlfile)))
```

We now display the first 10 rows of the dataset in the table below:

```
knitr::kable(head(diab_dat, 10))
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31.0	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0.0	0.232	54	1

```
str(diab_dat)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : num  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : num  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : num  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : num  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : num  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                     : num  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : num  1 0 1 0 1 0 1 0 1 1 ...
```

We also look at how many people have diabetes and how many do not. and the proportion of “no” and “yes”

```
table(diab_dat$Outcome)
```

```
##
##   0   1
## 500 268
```

```
prop.table(table(diab_dat$Outcome))
```

```
##
##         0         1
## 0.6510417 0.3489583
```

3. Data Pre-processing & Data Explorations

3.1 Convert the target variable type

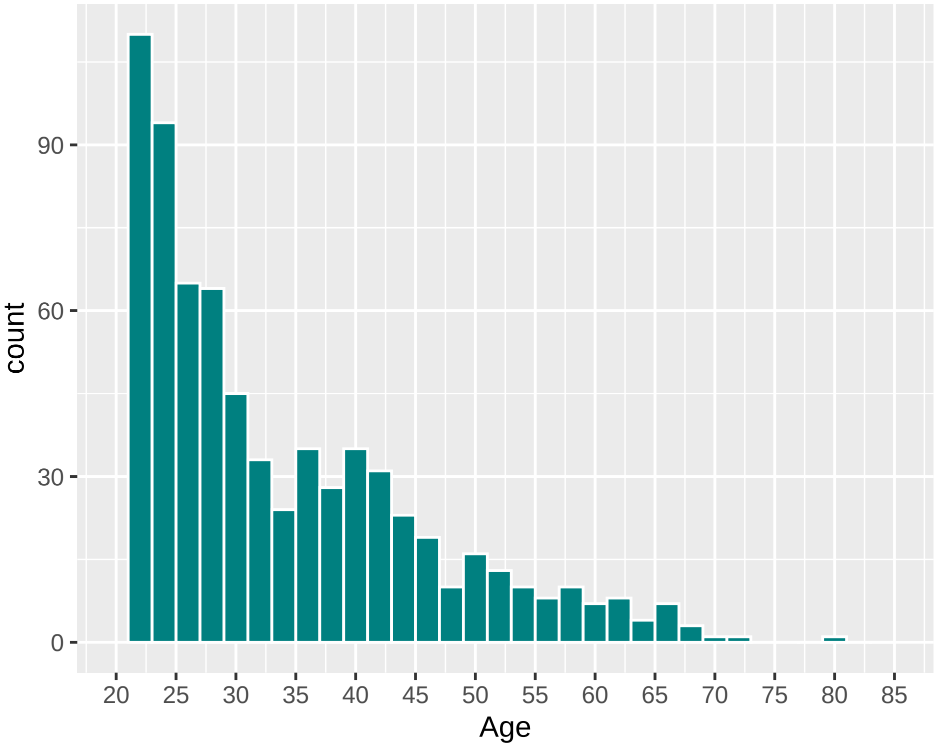
We convert the Outcome to factor, while changing ‘1’ and ‘0’ to ‘yes’ and ‘no’ to be more meaningful.

```
diab_dat$Outcome <- as.factor(ifelse(diab_dat$Outcome == "1", "Yes", "No"))
```

3.2 Age Analysis and grouping ages

Have a look at the Age variable. As can be seen from the distribution in the figure below, the age distribution is uneven, and there are too many young people in the data set.

```
ggplot(aes(x = Age), data = diab_dat) +
  geom_histogram(binwidth = 2 , color = "white", fill = "#008080") +
  scale_x_continuous(limits = c(20,85), breaks = seq(20,85,5))
```

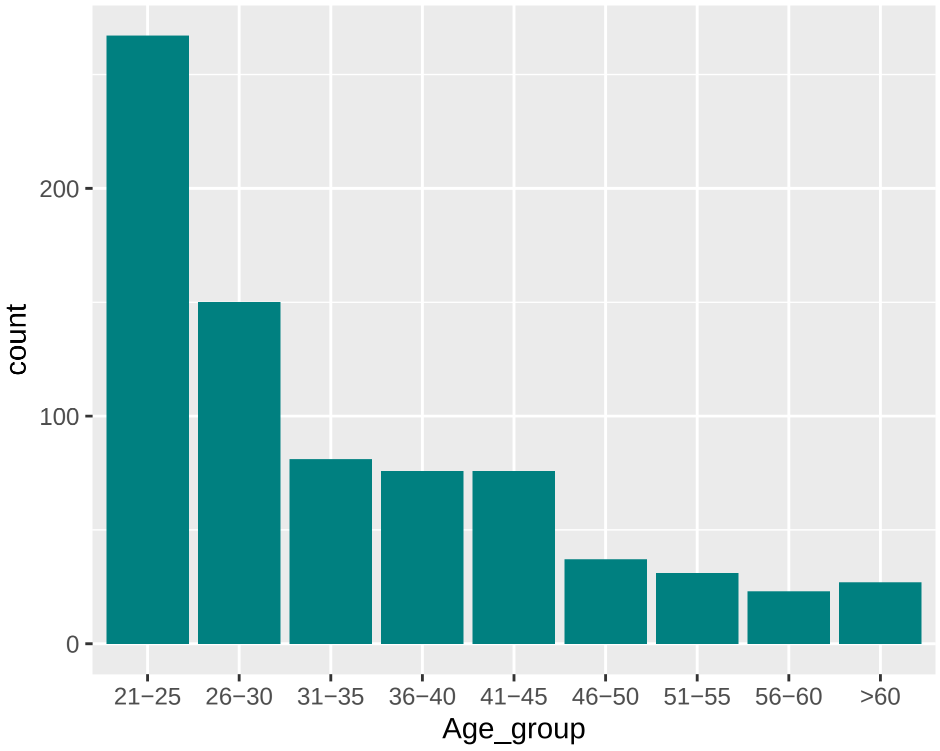


Let's divide age into different age groups

```
# create a temp dataset "diab_dat2", which is only used for analyzing the age_group
diab_dat2 <- diab_dat
# Create Age group column from "Yes" and "No"
diab_dat2$Age_group <- ifelse(diab_dat2$Age < 21, "<21",
                         ifelse((diab_dat2$Age>=21) & (diab_dat2$Age<=25), "21-25",
                           ifelse((diab_dat2$Age>25) & (diab_dat2$Age<=30), "26-30",
                             ifelse((diab_dat2$Age>30) & (diab_dat2$Age<=35), "31-35",
                               ifelse((diab_dat2$Age>35) & (diab_dat2$Age<=40), "36-40",
                                ifelse((diab_dat2$Age>40) & (diab_dat2$Age<=45), "41-45",
                                  ifelse((diab_dat2$Age>45) & (diab_dat2$Age<=50), "46-50",
                                   ifelse((diab_dat2$Age>50) & (diab_dat2$Age<=55), "51-55",
                                    ifelse((diab_dat2$Age>55) & (diab_dat2$Age<=60), "56-60",
                                      ">60")))))))))
```

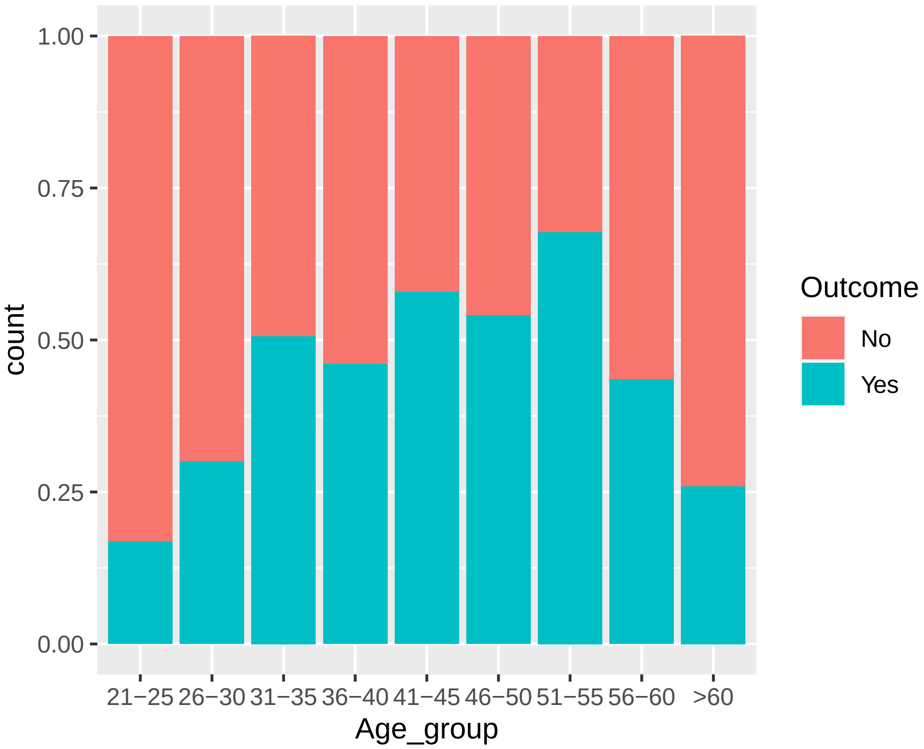
```r
# Convert Age_group to factor
diab_dat2$Age_group <- factor(diab_dat2$Age_group,
  levels=c('<21','21-25','26-30','31-35','36-40','41-45','46-50','51-55','56-60','>60'))
```

```r
# Barplot by Age_group
ggplot(aes(x = Age_group), data = diab_dat2) +
  geom_bar(fill='#008080')
```



From the plot above we can see that people aged 21-25 are the most in the dataset. Next, we have a look at the portion of the people who have been diagnosed diabetes by age group

```r
ggplot(diab_dat2, aes(x = Age_group, fill = Outcome)) +
  geom_bar(position = "fill")
```



The plot above shows that the 40-50 age group has the highest rate of diabetes diagnoses, followed by the 30-35 and 35-40 age groups.

3.3 Checking missing values

First check if there is any null values in the dataset

```
sum(is.na(diab_dat))
```

```
## [1] 0
```

There are no null values in the dataset, but we see that there are some zero values in the dataset.

```
knitr::kable(head(diab_dat))
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	Yes
1	85	66	29	0	26.6	0.351	31	No
8	183	64	0	0	23.3	0.672	32	Yes
1	89	66	23	94	28.1	0.167	21	No
0	137	40	35	168	43.1	2.288	33	Yes
5	116	74	0	0	25.6	0.201	30	No

A value of zero does not make sense and thus indicates missing value. We will replace these zeros with the median of each column.

```
# step1: replace all zeros with NAs
diab_dat[diab_dat == 0] <- NA

# step2: replace all "NA"s with the median of each column
diab_dat <- diab_dat %>%
  mutate_if(is.numeric, function(x) ifelse(is.na(x), median(x, na.rm = T), x))
```

Now showing the dataset again, we can see that all columns containing “0” are now replaced with median.

```
knitr::kable(head(diab_dat))
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	125	33.6	0.627	50	Yes
1	85	66	29	125	26.6	0.351	31	No
8	183	64	29	125	23.3	0.672	32	Yes
1	89	66	23	94	28.1	0.167	21	No
4	137	40	35	168	43.1	2.288	33	Yes
5	116	74	29	125	25.6	0.201	30	No

3.4 Splitting the dataset into independent and dependent datasets

We use row 1 to 8 as independent dataset and covert it to a matrix and use the Outcome as dependent dataset

```
diab_x <- as.matrix(diab_dat[,1:8])
diab_y <- diab_dat$Outcome
```

For independent dataset: there are 768 samples and 8 predictors in the dataset (features matrix).

```
dim(diab_x)[1]
```

```
## [1] 768
```

```
dim(diab_x)[2]
```

```
## [1] 8
```

For dependent dataset: there 768 outcomes in the dataset (target variable).

```
length(diab_y)
```

```
## [1] 768
```

Check the proportions of No and Yes samples in the dataset.

A. See how many patients are in each category (No or Yes).

```
summary(diab_y)
```

```
##  No Yes
## 500 268
```

B. See what the proportion of No and Yes in a table.

```
table(diab_y)
```

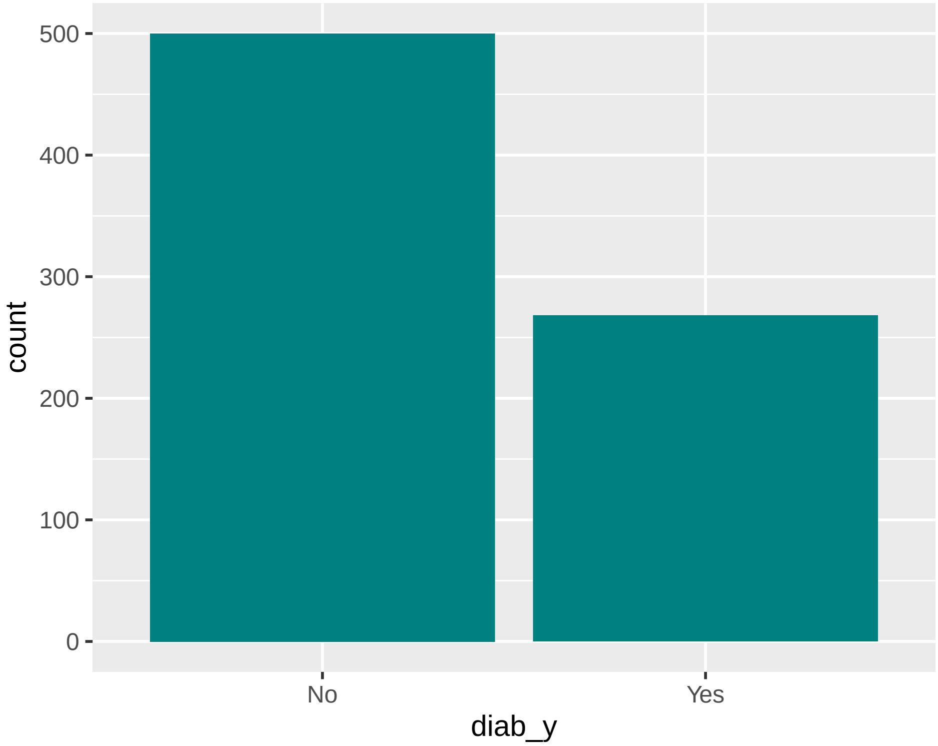
```
## diab_y
##  No Yes
## 500 268
```

```
prop.table(table(diab_y))
```

```
## diab_y
##        No       Yes
## 0.6510417 0.3489583
```

C. See the proportion of No and Yes in bar plot

```
data.frame(diab_y) %>%
  ggplot(aes(diab_y)) +
  geom_bar(fill='#008080')
```



3.5 Feature Scaling and Splitting Datasets

```
# Scaling all predictor variables
x_centered <- sweep(diab_x, 2, colMeans(diab_x))
x_scaled <- sweep(x_centered, 2, colSds(diab_x), FUN = "/")

# After scaling, the standard deviation is 1 for all columns
colSds(x_scaled)
```

```
## [1] 1 1 1 1 1 1 1 1
```

```
summary(x_scaled)
```

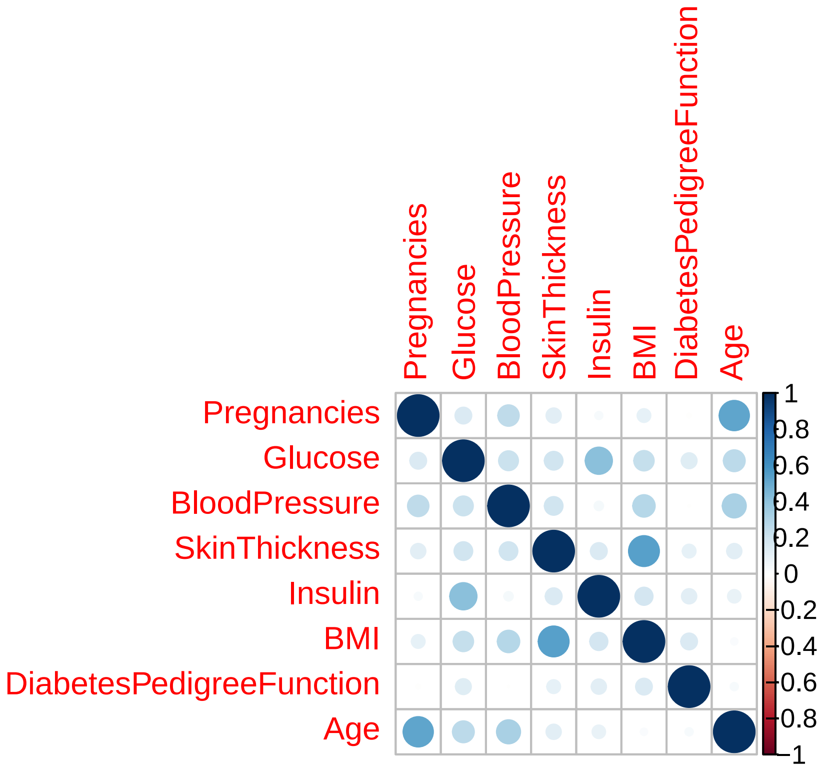
```
##   Pregnancies         Glucose        BloodPressure      SkinThickness
##  Min.   :-1.1485   Min.   :-2.5513   Min.   :-4.00001   Min.   :-2.51479
##  1st Qu.:-0.8130   1st Qu.:-0.7197   1st Qu.:-0.69331   1st Qu.:-0.46729
##  Median :-0.1420   Median :-0.1530   Median :-0.03197   Median :-0.01229
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.: 0.5291   3rd Qu.: 0.6109   3rd Qu.: 0.62937   3rd Qu.: 0.32896
##  Max.   : 4.2197   Max.   : 2.5410   Max.   : 4.10141   Max.   : 7.95020
##     Insulin              BMI            DiabetesPedigreeFunction
##  Min.   :-1.4664   Min.   :-2.07343   Min.   :-1.1888
##  1st Qu.:-0.2219   1st Qu.:-0.72074   1st Qu.:-0.6885
##  Median :-0.1814   Median :-0.02258   Median :-0.2999
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.:-0.1554   3rd Qu.: 0.60286   3rd Qu.: 0.4659
##  Max.   : 8.1651   Max.   : 5.03911   Max.   : 5.8797
##       Age
##  Min.   :-1.0409
##  1st Qu.:-0.7858
##  Median :-0.3606
##  Mean   : 0.0000
##  3rd Qu.: 0.6598
##  Max.   : 4.0611
```

3.6 Correlation Analysis

We now perform correlation analysis between independent variables. We observe the correlations through two plots.

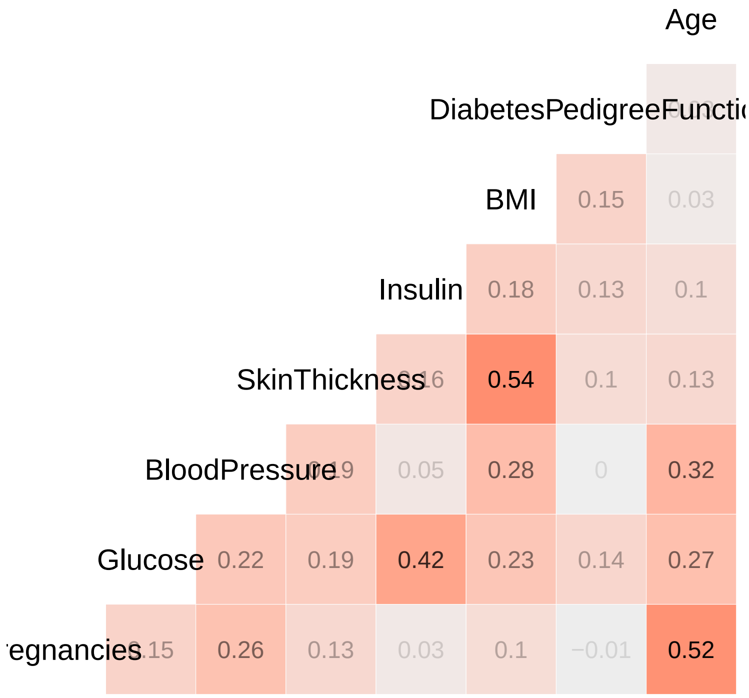
3.6.1 corrplot

```
corr_mat <- cor(diab_x)
corrplot(corr_mat)
```



3.6.2 plot correlation matrix

```
ggcorr(diab_x, label = TRUE, label_size = 3, label_round = 2, label_alpha = TRUE) +
        theme(legend.position = "none")
```



The output of ggcorr shows that pregnancy and age have the highest correlation coefficient at 0.6, followed by SkinThickness and BMI at 0.5, and glucose and insulin at 0.4. The remaining variables do not have strong correlations. Since the highest correlation coefficient among these variables is only 0.6, and we only have 8 predictors, we will keep all predictors.

4. Splitting Datasets into Training Set and Test Set

Split the dataset into a training set (80%) and test set (20%)

```
set.seed(1, sample.kind = "Rounding")    # if using R 3.6 or later
test_index <- createDataPartition(diab_y, times = 1, p = 0.2, list = FALSE)
test_x <- x_scaled[test_index,]
test_y <- diab_y[test_index]
train_x <- x_scaled[-test_index,]
train_y <- diab_y[-test_index]
```

By checking the proportion of No and Yes samples in both training set and test set, we see the proportions are very close.

```
# check the proportion in training set
prop.table(table(train_y))
```

```
## train_y
##        No       Yes
## 0.6514658 0.3485342
```

```
# check the proportion in test set
prop.table(table(test_y))
```

```
## test_y
##        No       Yes 
## 0.6493506 0.3506494
```

5. Building the Models

Model 1: Logistic Regression Model

```
# train a glm model on the training set
train_glm <- train(train_x, train_y, method = "glm")

# generate predictions on the test set
glm_preds <- predict(train_glm, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_glm <- confusionMatrix(data = glm_preds, reference = test_y)
result_glm <- c(cm_glm$byClass[c("Sensitivity","Specificity",
                                 "Prevalence")],cm_glm$overall["Accuracy"])
result_glm
```

```
## Sensitivity Specificity  Prevalence    Accuracy 
##   0.8900000   0.5185185   0.6493506   0.7597403
```

Model 2: rpart Model

```
# train a rpart model on the training set
train_rpart <- train(train_x, train_y, method = "rpart")

# generate predictions on the test set
rpart_preds <- predict(train_rpart, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_rpart <- confusionMatrix(data = rpart_preds, reference = test_y)
result_rpart <- c(cm_rpart$byClass[c("Sensitivity","Specificity",
                                     "Prevalence")], cm_rpart$overall["Accuracy"])
result_rpart
```

```
## Sensitivity Specificity  Prevalence    Accuracy 
##   0.9200000   0.5185185   0.6493506   0.7792208
```

Model 3: LDA Model

```
# train a LDA model on the training set
train_lda <- train(train_x, train_y, method = "lda")

# generate predictions on the test set
lda_preds <- predict(train_lda, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_lda <- confusionMatrix(data = lda_preds, reference = test_y)
result_lda <- c(cm_lda$byClass[c("Sensitivity","Specificity",
                                 "Prevalence")], cm_lda$overall["Accuracy"])
result_lda
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.8700000   0.5185185   0.6493506   0.7467532
```

Model 4: QDA Model

```
# train a QDA model on the training set
train_qda <- train(train_x, train_y, method = "qda")

# generate predictions on the test set
qda_preds <- predict(train_qda, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_qda <- confusionMatrix(data = qda_preds, reference = test_y)
result_qda <- c(cm_qda$byClass[c("Sensitivity","Specificity",
                                 "Prevalence")], cm_qda$overall["Accuracy"])
result_qda
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.8600000   0.5000000   0.6493506   0.7337662
```

Model 5: Loess Model

```
# set.seed(5)
set.seed(5, sample.kind = "Rounding") # simulate R 3.5

# train a loess model on the training set
train_loess <- train(train_x, train_y, method = "gamLoess")

# generate predictions on the test set
loess_preds <- predict(train_loess, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_loess <- confusionMatrix(data = loess_preds, reference = test_y)
```

```r
result_loess <- c(cm_loess$byClass[c("Sensitivity","Specificity",
                                    "Prevalence")], cm_loess$overall["Accuracy"])
result_loess
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.8900000   0.5555556   0.6493506   0.7727273
```

Model 6: K-Nearest Neighbours Model

```r
# train a KNN model on the training set, and find the best value in the model
set.seed(7,sample.kind = "Rounding")
train_knn <- train(train_x, train_y,
                   method = "knn",
                   tuneGrid = data.frame(k = seq(3, 21, 2)))

# determine the value of k
train_knn$bestTune # the best value is 10 or 21
```

```
##     k
## 10 21
```

```r
# use the final value in the model to generate predictions on the test set
knn_preds <- predict(train_knn, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_knn <- confusionMatrix(data = knn_preds, reference = test_y)
result_knn <- c(cm_knn$byClass[c("Sensitivity","Specificity",
                                "Prevalence")], cm_knn$overall["Accuracy"])
result_knn
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.9300000   0.5740741   0.6493506   0.8051948
```

Model 7: Random Forest model

```r
set.seed(9,sample.kind = "Rounding")

# train a random forest model on the training set
train_rf <- train(train_x, train_y,
                    method = "rf",
                    importance = TRUE,
                    tuneGrid = data.frame(mtry = c(3,5,7,9)))

# find the value of mtry gives the highest accuracy
train_rf$bestTune  # the best value of mtry is 3
```

```
##   mtry
## 1    3
```

```
# find the most important variable in the random forest model
varImp(train_rf)
```

```
## rf variable importance
##
##                          Importance
## Glucose                     100.000
## BMI                          34.970
## Age                          28.771
## Pregnancies                  22.484
## Insulin                      11.927
## DiabetesPedigreeFunction     10.178
## SkinThickness                 2.177
## BloodPressure                 0.000
```

```
# use the best value of mtry in the model to generate predictions on the test set
rf_preds <- predict(train_rf, test_x)

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_rf <- confusionMatrix(data = rf_preds, reference = test_y)
result_rf <- c(cm_rf$byClass[c("Sensitivity","Specificity",
                               "Prevalence")], cm_rf$overall["Accuracy"])
result_rf
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.8600000   0.6481481   0.6493506   0.7857143
```

Model 8: Ensemble Model

```
# Creating an ensemble model that combine all the models trained before
ensemble <- cbind(glm = glm_preds=="No", lda = lda_preds=="No",
                  qda = qda_preds=="No", loess = loess_preds=="No",
                  rf = rf_preds=="No", knn = knn_preds=="No",
                  rpart = rpart_preds=="No")

# generate predictions
ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "No", "Yes")

# calculate the accuracy, sensitivity, specificity, and prevalence of the model on test set
cm_ensemble <- confusionMatrix(data = factor(ensemble_preds), reference = test_y)
result_ensmble <- c(cm_ensemble$byClass[c("Sensitivity","Specificity", "Prevalence")],
                    cm_ensemble$overall["Accuracy"])
result_ensmble
```

```
## Sensitivity Specificity  Prevalence    Accuracy
##   0.8800000   0.5555556   0.6493506   0.7662338
```

Creating an ensemble list that shows the accuracy of these models.

```
# create a list of accuracy by model names
model_names <- c("Logistic regression","rpart","LDA","QDA","Loess",
                 "K nearest neighbors","Random forest","Ensemble")

models_accuracy <- c(cm_glm$overall["Accuracy"], cm_rpart$overall["Accuracy"],
                     cm_lda$overall["Accuracy"], cm_qda$overall["Accuracy"],
                     cm_loess$overall["Accuracy"], cm_knn$overall["Accuracy"],
                     cm_rf$overall["Accuracy"], cm_ensemble$overall["Accuracy"])
data.frame(Model = model_names, Accuracy = models_accuracy)
```

```
##                 Model  Accuracy
## 1 Logistic regression 0.7597403
## 2               rpart 0.7792208
## 3                 LDA 0.7467532
## 4                 QDA 0.7337662
## 5               Loess 0.7727273
## 6 K nearest neighbors 0.8051948
## 7       Random forest 0.7857143
## 8            Ensemble 0.7662338
```

```
# select a best prediction model according to highest accuracy
best_model <- subset(models_accuracy, models_accuracy==max(models_accuracy))
best_model
```

```
##  Accuracy
## 0.8051948
```

Model Comparisons

Compare the accuracy of the models, and how they perform in terms of sensitivity, specificity.

```
result_all <- data.frame(rbind(result_glm,result_rpart,result_lda,result_qda,
                               result_loess,result_knn, result_rf,result_ensmble))
names(result_all) <- c("Sensitivity","Specificity", "Prevalence", "Accuracy")
result_all
```

```
##                Sensitivity Specificity Prevalence  Accuracy
## result_glm            0.89   0.5185185  0.6493506 0.7597403
## result_rpart          0.92   0.5185185  0.6493506 0.7792208
## result_lda            0.87   0.5185185  0.6493506 0.7467532
## result_qda            0.86   0.5000000  0.6493506 0.7337662
## result_loess          0.89   0.5555556  0.6493506 0.7727273
## result_knn            0.93   0.5740741  0.6493506 0.8051948
## result_rf             0.86   0.6481481  0.6493506 0.7857143
## result_ensmble        0.88   0.5555556  0.6493506 0.7662338
```

We can see that the best model is KNN model because it has highest accuracy, it also has better performance in Sensitivity, Specificity on the same Prevalence.

6. Conclusion

We used a few different methods to build some models and calculated and compared their accuracy. We recommend the KNN model as the best model because it has the highest prediction accuracy. In the future we can try to use other methods to train the models to get better accuracy; We may also divide the Age variable into different "Age Group" and find some different methods to improve the accuracy. I actually tried to add "Age Group" as one of the predictor, and trained the model used LDA and QDA methods and better accuracy by creating a "Age Group" variable.

References

1. HarvardX PH125.8x Data Science: Machine Learning: 7.1 Final Assessment: Breast Cancer Prediction Project
2. Kaggle: Pima Indians Diabetes Database, Dataset Notebooks
3. Kaggle: MIRI CHO,[Classification] Diabetes or Not (with basic 12ML)
4. Kaggle: LEARNING-MONK, PIMA Indian Diabetes - Logistic Regression with R
5. ggcorr: https://briatte.github.io/ggcorr