Course: HarvardX-Data Science: Capstone
Project: Diabetes Risk Prediction - Medical Lab Data Analysis

Submitted by: Ayu Tiwari

Date: 12/28/2020

Abstract

Diabetes is a chronic disease caused by high levels of blood sugar (or glucose level), diabetes can lead to serious damages like the heart, blood pressure, eyes, kidneys. There are two types of diabetes are Type 1 and Type 2. Type 2 diabetes is a chronic condition also called "diabetes mellitus". This project uses Kaggle's UCI diabetes data set to detect the early stage of diabetes cases. This project will classify the people who are at the risk of getting diabetes by applying various Machine Learning models and predictions.

Introduction

This project will use the data set from UCI early-stage Diseases detection which is available on Kaggle.com and UCI websites, data set file name is "**`diabetes_data_upload.csv`**" (https://github.com/ayutiwari/Data_Science_Project) which is a freely available data source and contains 520 observations on 17 variables. This data has been collected from the UCI Repository of Machine Learning Databases. [1, 2]

This project will explore UCI diabetes data set and calculate the prediction of the risk of early-stage diabetes by using training, test and a validation set. The validation data set is used only for the predictions from the final model which will produce the higher accuracy, sensitivity and specificity. In this project, various Machine Learning models will be built and simulated on the training data set and the final model will be selected based on higher accuracies in prediction.

This project is categorized in following topics:
Data description, data exploration, data preparation and training and evaluating of the Machine Learning Models - Logistic Regression, K-Nearest Neighbors, Decision Tree, and Random Forest Model. Final Model recommendation followed by conclusion.

Data Description and Data Exploration

In this section of the project, we will explore the available data set, do the required transformations and perform the detailed analysis.

```
#read data from csv file and replace spaces from column name
data <- read_csv("diabetes_data_upload.csv", col_types = "dfffffffffffffff")
colnames(data) <- make.names(colnames(data))

#rearrange column values
col <- which(data[1,]=="No")
for (i in col)
{
      data[,i] <- factor(data[[i]], levels = c("Yes","No"))
}
```

1) Data set structure

```
str(data)

 $ Age           : num [1:520] 40 58 41 45 60 55 57 66 67 70 ...
 $ Gender        : Factor w/ 2 levels "Male","Female": 1 1 1 1 1 1 1 1 1 1 .
 $ Polyuria          : Factor w/ 2 levels "Yes","No": 2 2 1 2 1 1 1 1 1 2 ...
 $ Polydipsia        : Factor w/ 2 levels "Yes","No": 1 2 2 2 1 1 1 1 1 1 ...
 $ sudden.weight.loss: Factor w/ 2 levels "Yes","No": 2 2 2 1 1 2 2 1 2 1 ...
 $ weakness          : Factor w/ 2 levels "Yes","No": 1 1 1 1 1 1 1 1 1 1 ...
 $ Polyphagia        : Factor w/ 2 levels "Yes","No": 2 2 1 1 1 1 1 2 1 1 ...
 $ Genital.thrush    : Factor w/ 2 levels "Yes","No": 2 2 2 1 2 2 1 2 1 2 ...
 $ visual.blurring   : Factor w/ 2 levels "Yes","No": 2 1 2 2 1 1 2 1 2 1 ...
 $ Itching           : Factor w/ 2 levels "Yes","No": 1 2 1 1 1 1 2 1 1 1 ...
 $ Irritability      : Factor w/ 2 levels "Yes","No": 2 2 2 2 1 2 2 1 1 1 ...
 $ delayed.healing   : Factor w/ 2 levels "Yes","No": 1 2 1 1 1 1 1 2 2 2 ...
 $ partial.paresis   : Factor w/ 2 levels "Yes","No": 2 1 2 2 1 2 1 1 1 2 ...
 $ muscle.stiffness  : Factor w/ 2 levels "Yes","No": 1 2 1 2 1 1 2 1 1 2 ...
 $ Alopecia          : Factor w/ 2 levels "Yes","No": 1 1 1 2 1 1 2 2 2 1 ...
 $ Obesity           : Factor w/ 2 levels "Yes","No": 1 2 2 2 1 1 2 2 1 2 ...
 $ class             : Factor w/ 2 levels "Positive","Negative": 1 1 1 1 1 ..
```

2) Data Dimensions

```
#display rows and columns count
dim(data)

[1] 520  17
```

A data set has 520 observations on 17 variables:
Features set = { 1.Age - 1.20-65
2.Sex -1. Male, 2.Female
3.Polyuria 1.Yes, 2.No.
4.Polydipsia 1.Yes, 2.No.
5.sudden weight loss 1.Yes, 2.No.
6.weakness 1.Yes, 2.No.
7.Polyphagia 1.Yes, 2.No.
8.Genital thrush 1.Yes, 2.No.
9.visual blurring 1.Yes, 2.No.
10.Itching 1.Yes, 2.No.
11.Irritability 1.Yes, 2.No.
12.delayed healing 1.Yes, 2.No.
13.partial paresis 1.Yes, 2.No.
14.muscle stiffness 1.Yes, 2.No.
15.Alopecia 1.Yes, 2.No.
16.Obesity 1.Yes, 2.No.
17.Class 1.Positive, 2.Negative.
}

3) Sample Data

```
#display first 6 records
head(data)
```

```
        Age Gender Polyuria Polydipsia sudden.weight.l… weakness Polyphagia
      Genital.thrush visual.blurring Itching Irritability delayed.healing
      partial.paresis
        <dbl> <fct>  <fct>    <fct>      <fct>            <fct>    <fct>
      <fct>          <fct>           <fct>   <fct>        <fct>
      <fct>
      1    40 Male   No       Yes        No               Yes      No
      No             No              Yes     No           Yes             No
      2    58 Male   No       No         No               Yes      No
      No             Yes             No      No           No              Yes
      3    41 Male   Yes      No         No               Yes      Yes
      No             No              Yes     No           Yes             No
      4    45 Male   No       No         Yes              Yes      Yes
      Yes            No              Yes     No           Yes             No
      5    60 Male   Yes      Yes        Yes              Yes      Yes
      No             Yes             Yes     Yes          Yes             Yes
      6    55 Male   Yes      Yes        No               Yes      Yes
      No             Yes             Yes     No           Yes             No
      # … with 4 more variables: muscle.stiffness <fct>, Alopecia <fct>,
      Obesity <fct>, class <fct>
```

Normalize data set in R for further exploration

From the above sample, we have observed that the last column "class" is a factor variable with string values – "Positive and "Negative" to represent the diabetes result outcome. Gender is also a factor variable with two possible values "Male" and "Female", and Age is a numeric variable with age numbers. All other columns are factor variables to represent the various symptoms and conditions populated with "Yes" or "No" values. We will perform some transformations on these column values for easy analysis and plot purposes.

- Age - Age column has numeric values, we will normalize this column between scale of 0 and 1 to make a fair comparison with everything else.
- Gender – Gender column has "Male" and "Female" factor values, convert Male =1 and Female =0.
- All other columns - "Yes" will be changed to 1 and "No" will be changed to 0.

This transformation will make the data set more normalized to make correct analysis.

```
#Call preprocess function to convert age column values
temp_df <- preProcess(data, method="range")
dia_df <- predict(temp_df, newdata = data)

#convert class values to Positive = 1 and Negative = 0
levels(dia_df$class) <-  c(1,0)

#convert Gender Male=1 and Female=0
dia_df$Gender <- ifelse(dia_df$Gender== "Male", 1, 0)
```

```
#Convert other columns Yes=1 and No=0
dia_df$Polyuria <- ifelse(dia_df$Polyuria == "Yes", 1, 0)
dia_df$Polydipsia <- ifelse(dia_df$Polydipsia == "Yes", 1, 0)
dia_df$sudden.weight.loss <- ifelse(dia_df$sudden.weight.loss == "Yes", 1, 0)
dia_df$weakness <- ifelse(dia_df$weakness == "Yes", 1, 0)
dia_df$Polyphagia <- ifelse(dia_df$Polyphagia == "Yes", 1, 0)
dia_df$Genital.thrush <- ifelse(dia_df$Genital.thrush == "Yes", 1, 0)
dia_df$visual.blurring <- ifelse(dia_df$visual.blurring == "Yes", 1, 0)
dia_df$Itching <- ifelse(dia_df$Itching == "Yes", 1, 0)
dia_df$Irritability <- ifelse(dia_df$Irritability == "Yes", 1, 0)
dia_df$delayed.healing <- ifelse(dia_df$delayed.healing == "Yes", 1, 0)
dia_df$partial.paresis <- ifelse(dia_df$partial.paresis == "Yes", 1, 0)
dia_df$muscle.stiffness <- ifelse(dia_df$muscle.stiffness == "Yes", 1, 0)
dia_df$Alopecia <- ifelse(dia_df$Alopecia == "Yes", 1, 0)
dia_df$Obesity <- ifelse(dia_df$Obesity == "Yes", 1, 0)
```

4) Correlation of every pair of features

```
set.seed(1, sample.kind="Rounding")

 #unfactor class variable to numeric
 temp_df <- dia_df
 temp_df$class<- as.numeric(as.character(temp_df$class))

 #generate correlation of all features
 corr_df<- temp_df %>% cor()
 print(corr_df)
```

Correlation plot:

```
heatmap(as.matrix(corr_df), Colv = NA, Rowv = NA, scale="row")
```

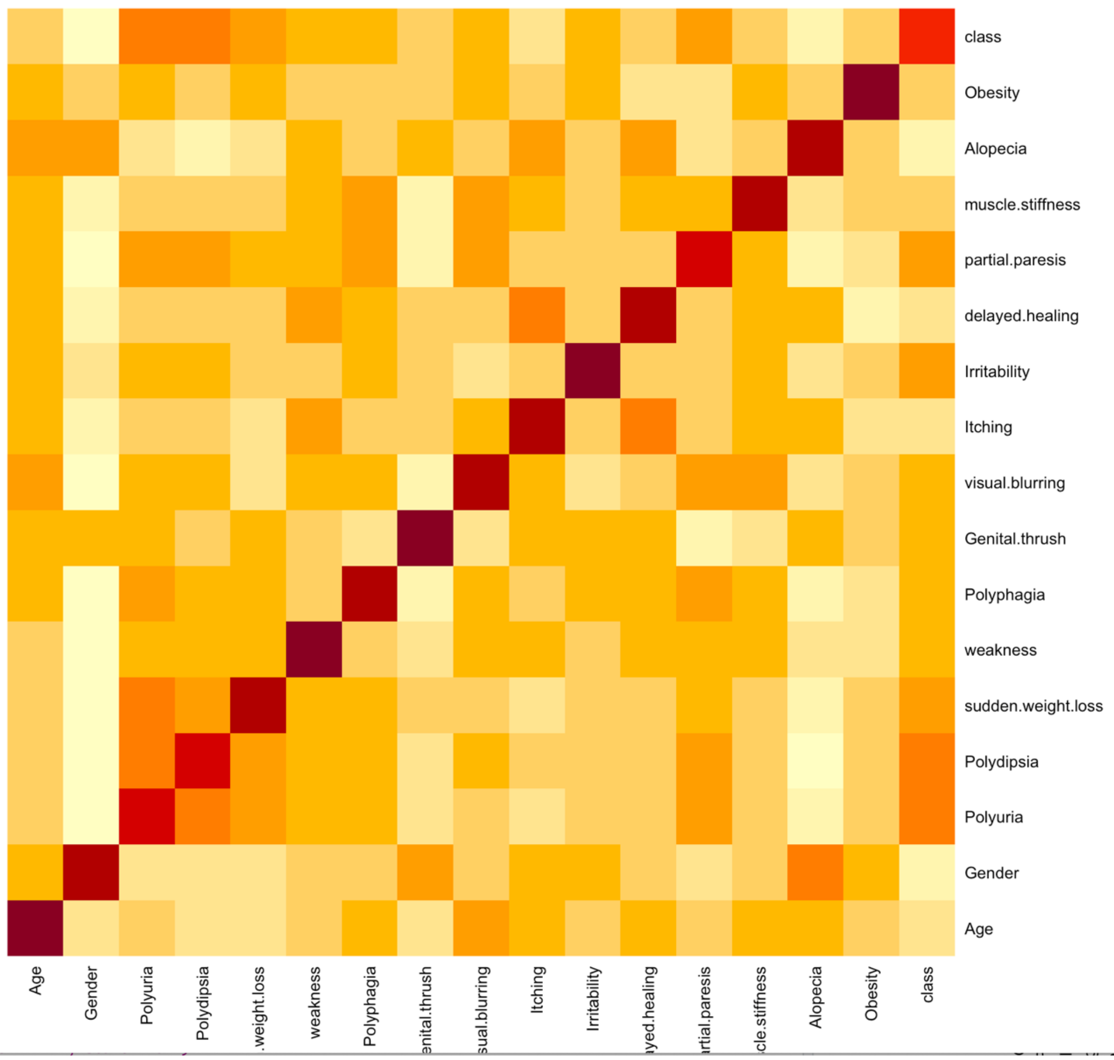

Figure 1: Heatmap of feature correlation

In the above plot dark colors show more correlation with features. Polyuria and Polydipsia have significant correlation with diabetes (outcome) variable. Also notice the correlation between other pairs of features like Age, Gender and other dire symptom conditions.

5) Count of Positive and Negative results of diabetes

```
#find out how many people have diabetes and how many don’t
temp_df %>% group_by(class) %>% summarize(count=n())

class count
  <dbl> <int>
1     0   200
2     1   320

#Plot
temp_df %>% ggplot(aes(class)) + geom_bar(fill = c("darkgreen", "red")) +
ggtitle("Diabetes Positive and Negative outcome")
```

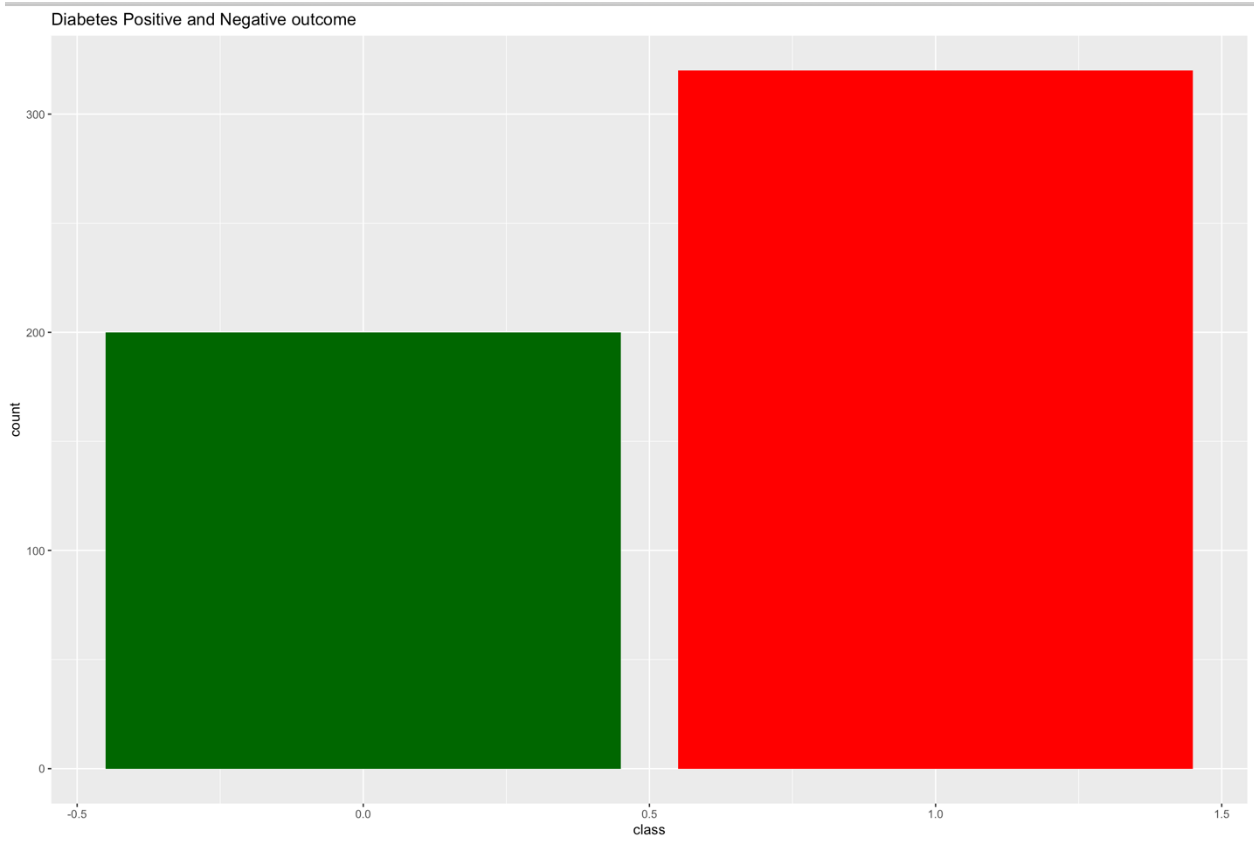

Figure 2: Positive and Negative results of diabetes

Here, “1” red color indicates diabetic Positive cases and “0” green indicates Negative cases of diabetes.

6) Check on Diabetes cases by average age group

```
#calculate average age for diabetic and non-diabetic cases
x<- temp_df %>% group_by(class) %>% summarise(avg_age=mean(Age))

  #plot
  x %>% ggplot(aes(x=class, y=avg_age)) + geom_bar(stat="identity", fill
  = c("red", "darkgreen")) + ggtitle("Average age group and Diabetes
  outcome")
```

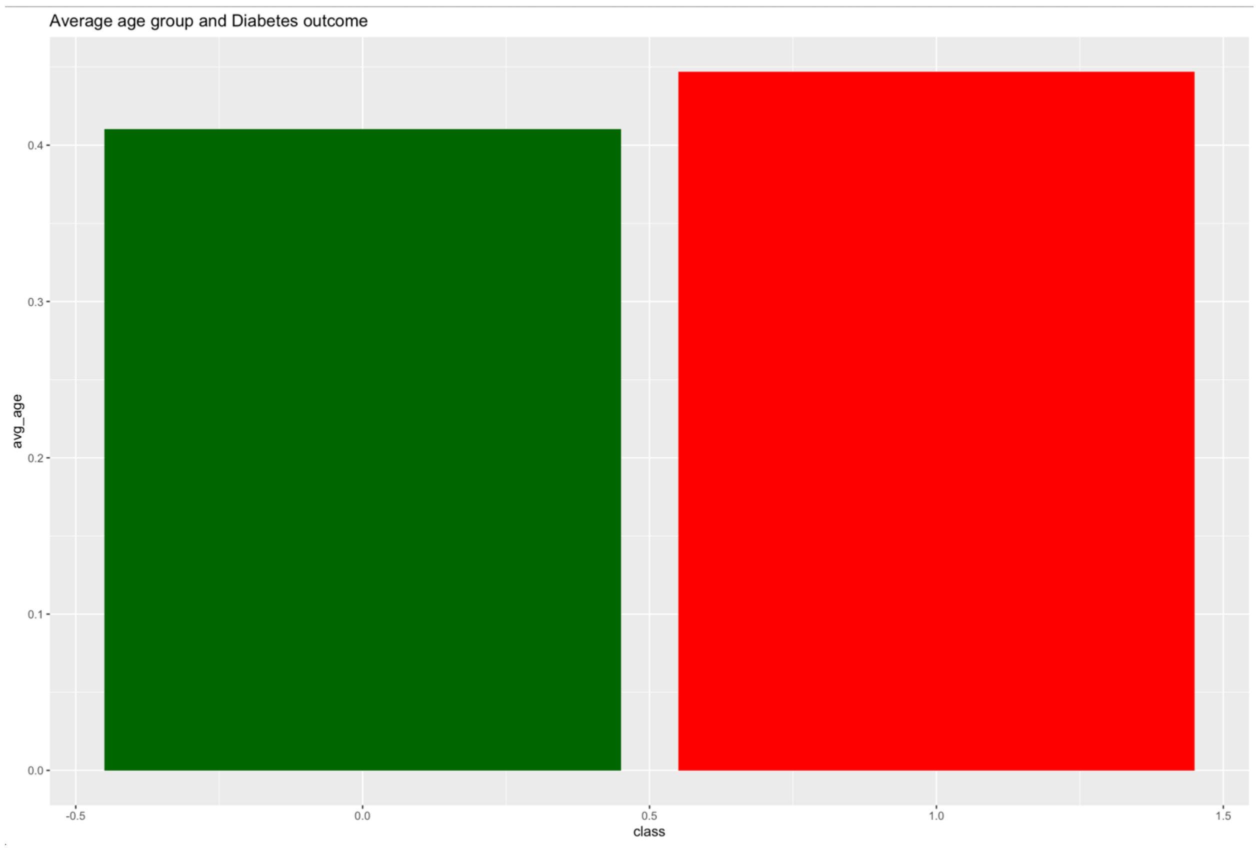

Figure 3: Average age for diabetic and non-diabetic cases

Figure 3 shows relations between age and diabetes outcome, average age of people having diabetes is higher. Here, “1” red color indicates diabetic Positive cases and “0” green indicates Negative cases of diabetes.

7) Scatterplots of this data set.

```
pairs(temp_df, col=data$class)
```

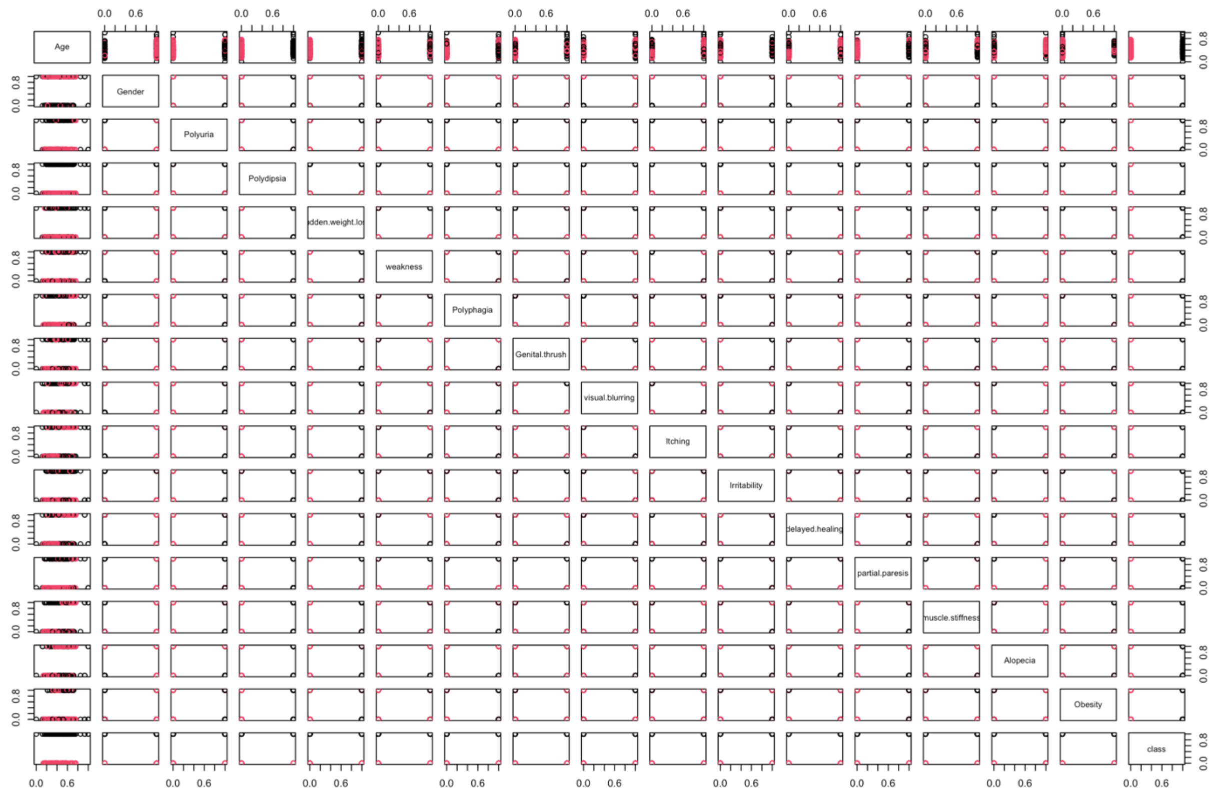

Figure 4: Relationship of all attributes

A relationship of features Polyuria and Polydipsia

From the observations we can say that Polyuria and Polydipsia are two major variables in diabetes detection in this data set. If any patient has both conditions, then probability of Positive outcome diabetic detection is higher. The correlation between polyuria and polydipsia is described below.

```
#relationship of features Polyuria and Polydipsia
temp_df %>% ggplot(aes(Polyuria, Polydipsia, colour = class)) +
geom_jitter(height = 0.3, width = 0.2) + ggtitle("Polydipsia and Polyuria
Prevalence in diabetes")
```

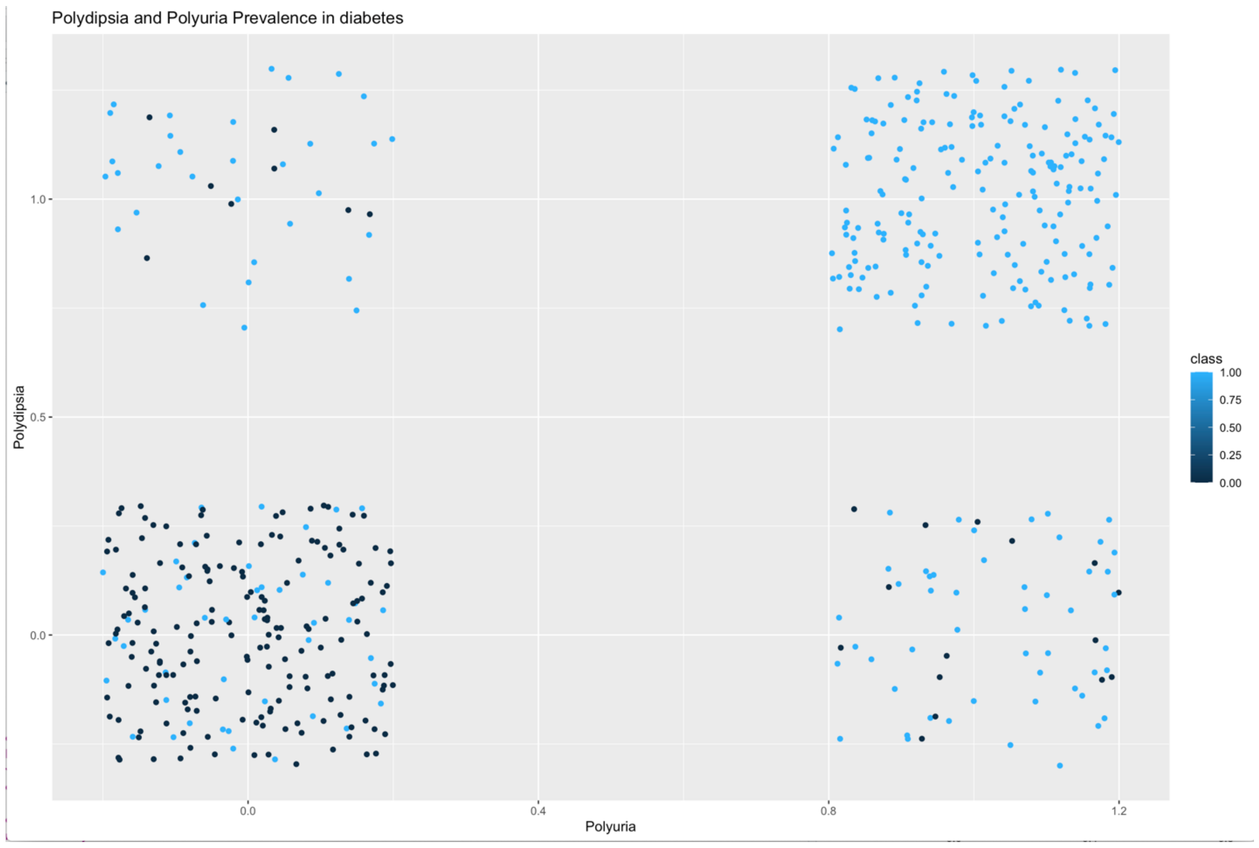

Figure 5: Polyuria and Polydipsia symptoms relation in diabetes outcome

Here, on diabetic scale “1” represents diabetic outcome and “0” represents non-diabetic outcome

A relationship of features Sudden weight Loss and Weakness

```
#relationship of features Sudden weight Loss and Weakness
temp_df %>% ggplot(aes(sudden.weight.loss, weakness, colour = class)) +
geom_jitter(height = 0.3, width = 0.2) + xlab("Sudden Weight Loss") +
ylab("Weakness") + ggtitle("Sudden Weight Loss and Weakness Prevalence of
Diabetes")
```

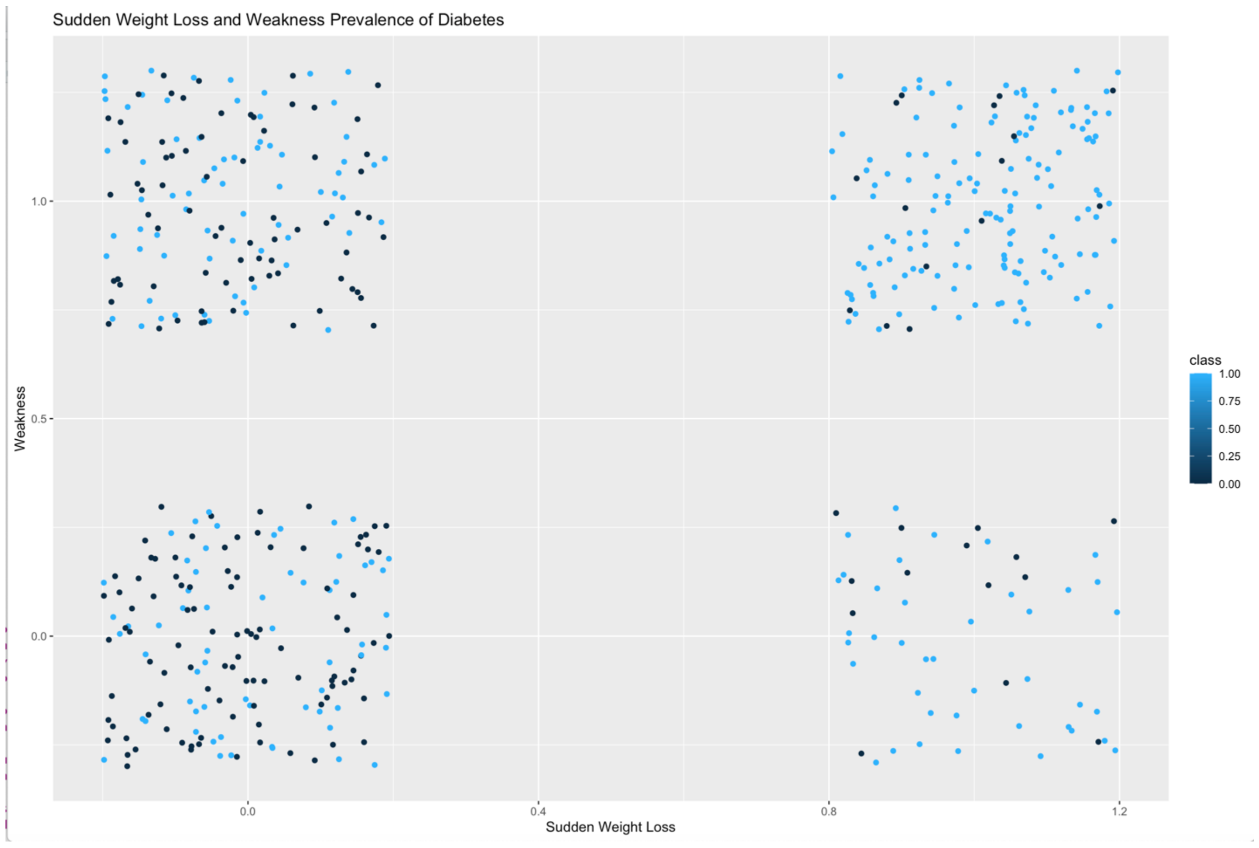

Figure 6: Sudden weight Loss and Weakness symptoms relation in diabetes outcome

Here, on diabetic scale "1" represents diabetic outcome and "0" represents non-diabetic outcome

A relationship of features Age and Obesity

```
#Age and Obesity in diabetes relation
temp_df %>% ggplot(aes(Obesity, Age, colour = class)) + geom_jitter(height =
0.3, width = 0.2) + xlab("Obesity") + ylab("Age") + ggtitle("Age and Obesity
Prevalence of Diabetes")
```

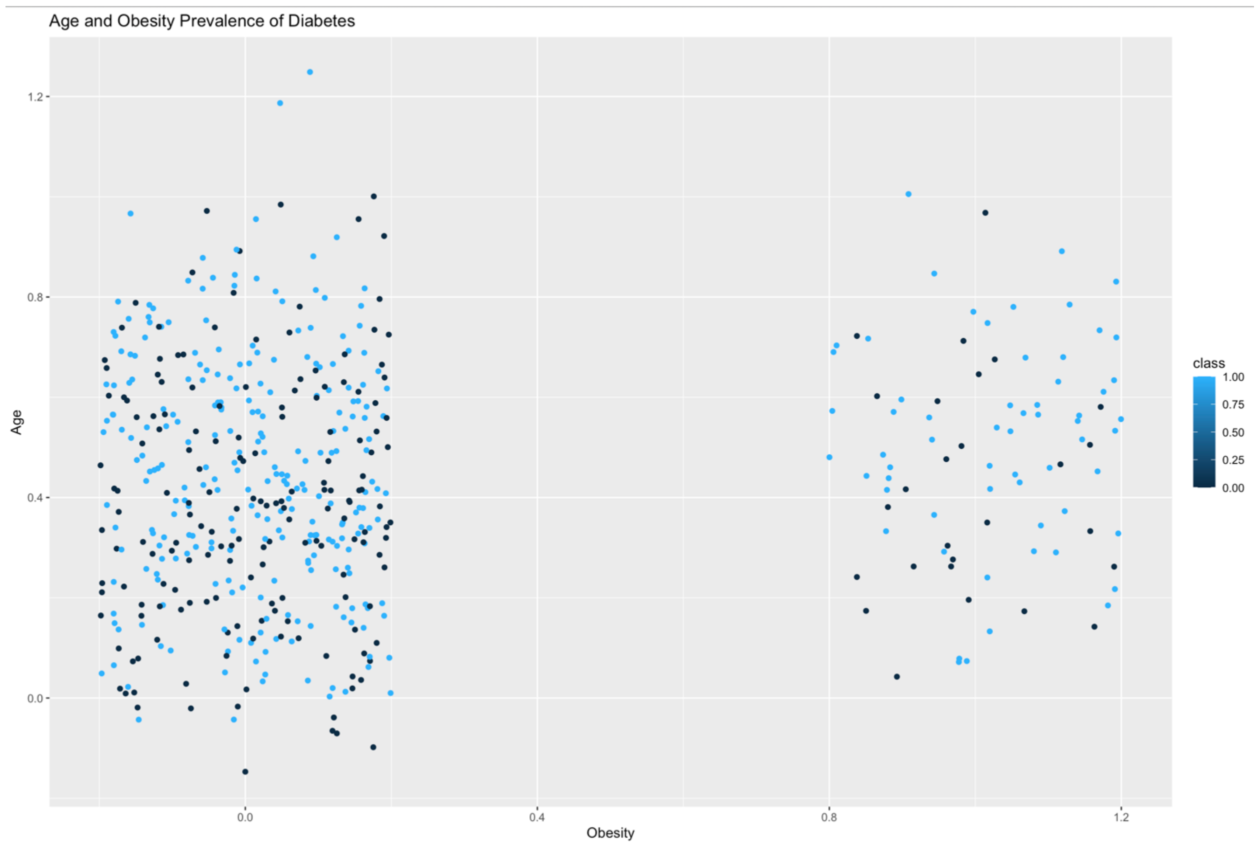

Figure 7: Age and Obesity factor in diabetes outcome

Here, on diabetic scale “1” represents diabetic outcome and “0” represents non-diabetic outcome.

A relation between Genital Thrush and Visual Blurring

```
#Genital Thrush and Visual Blurring symptoms relationship with diabetes
outcome
temp_df %>%
  ggplot(aes(Genital.thrush, visual.blurring, colour = class)) +
  geom_jitter(height = 0.3, width = 0.2) +
  xlab("Genital Thrush") +
  ylab("Visual Blurring") +
  ggtitle("Genital Thrush and Visual Blurring Prevalence of Diabetes")
```

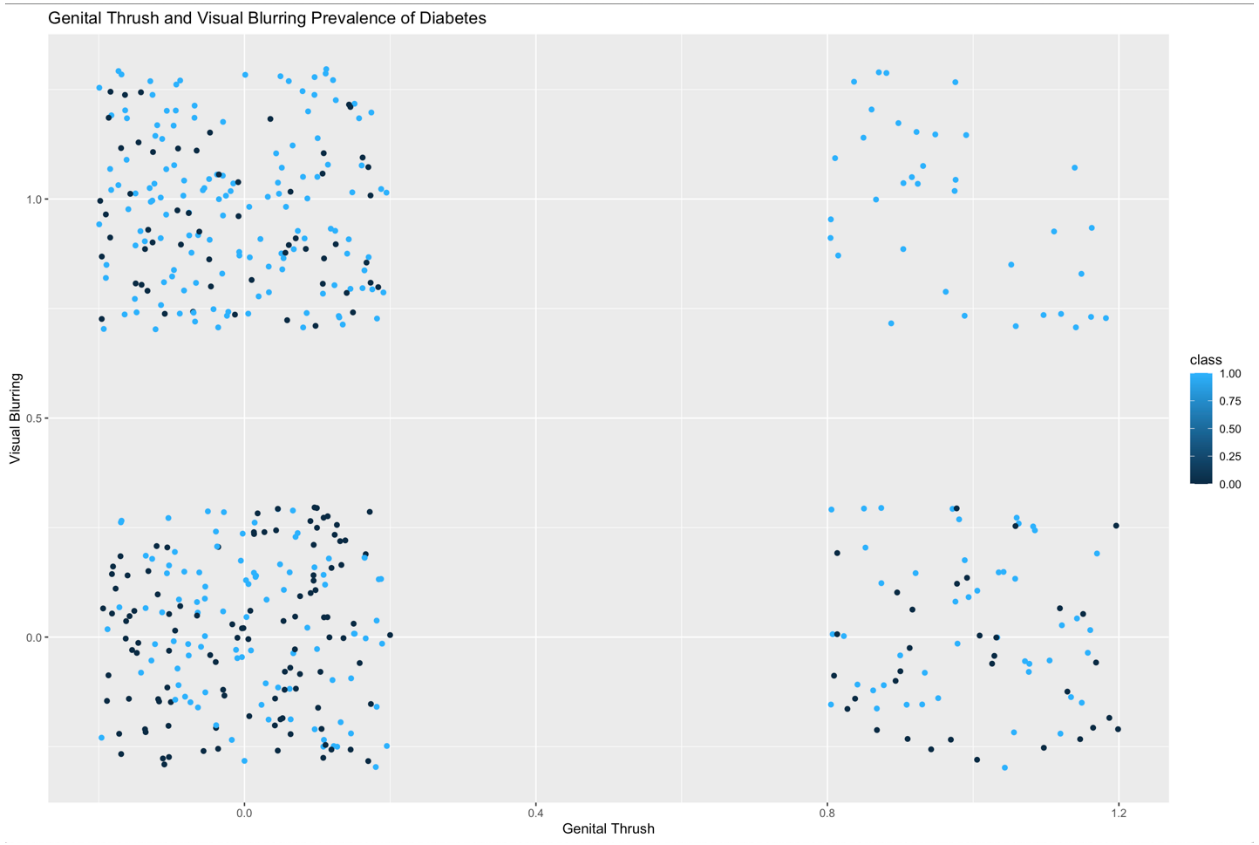

Figure 8: Genital Thrush and Visual Blurring symptoms relationship in diabetes outcome

Here, on diabetic scale “1” represents diabetic outcome and “0” represents non-diabetic outcome.

Preparing Data Set for Training, Test and Validation

Split data set into train set and test set. Also create a validation set for the final model validation.

```
#create a validation data set
set.seed(88, sample.kind="Rounding")

validation_index <- createDataPartition(data$class, times = 1, p = 0.20, list
= FALSE)
validation <- data %>% slice(validation_index)
diabetes <- data %>% slice(-validation_index)

#create train and test sets from diabetes data set
set.seed(16, sample.kind="Rounding")

test_index <- createDataPartition(diabetes$class, times = 1, p = 0.20, list =
FALSE)
train <- diabetes %>% slice(-test_index)
test <- diabetes %>% slice(test_index)
```

Machine Learning Model Design and Simulation:

In this section we will implement four Machine Learning models and measure their performance.

1. Logistic regression

The Logistic regression is used to classify if the given patient will get diabetes outcome or not. Logistic regression model will be used with family = “binomial”. All attributes will be used to generate the summary of the model.

We apply a logic of regression model for 5 folds to estimate the optimal value of the probability p (a tuning parameter).

```
#range of tuning parameters
tune_p <- seq(0.2, 0.5, by = 0.01)

#repeat experiment 5 times
folds <- 5

#Matrix to store the mean of accuracy
lr_acc_p <- matrix(nrow = folds, ncol = length(tune_p))

#create data partitions
part_data <- createFolds(1:nrow(train), k = folds)

for (i in 1:folds)
{

  #create train and test sets
  temp_train <- train %>% slice(-part_data[[i]])
  temp_test <- train %>% slice(part_data[[i]])

  #generate a matrix with mean of accuracy and sensitivity
  lr_acc_p[i,] <- sapply(tune_p, function(p){

  #apply logistic regression model
   train_lr_model <- glm(as.numeric(class=="Positive")~., family =
"binomial", data = temp_train)

  #obtain the predictions (these are probabilities)
  predict_res <- predict(train_lr_model, temp_test, type = "response")

  #if prediction > p then classify diabetes as Positive outcome otherwise
   negative outcome.

  t_lr_cm <- confusionMatrix(ifelse(predict_res > p, "Positive","Negative")
%>% factor(levels = c("Positive","Negative")), temp_test$class)

  #return the mean of the accuracy and sensitivity
   return(mean(c(t_lr_cm$overall["Accuracy"],
                 t_lr_cm$byClass["Sensitivity"])))
})

}

#Calculate Mean
x<-colMeans(lr_acc_p)

max(x)
```

0.9264023

```
#find optimal value of P
p<- tune_p[which.max(x)]
print(p)
```

0.39

```
#plot
hist(x, main="Linear Regression average accuracies")
```

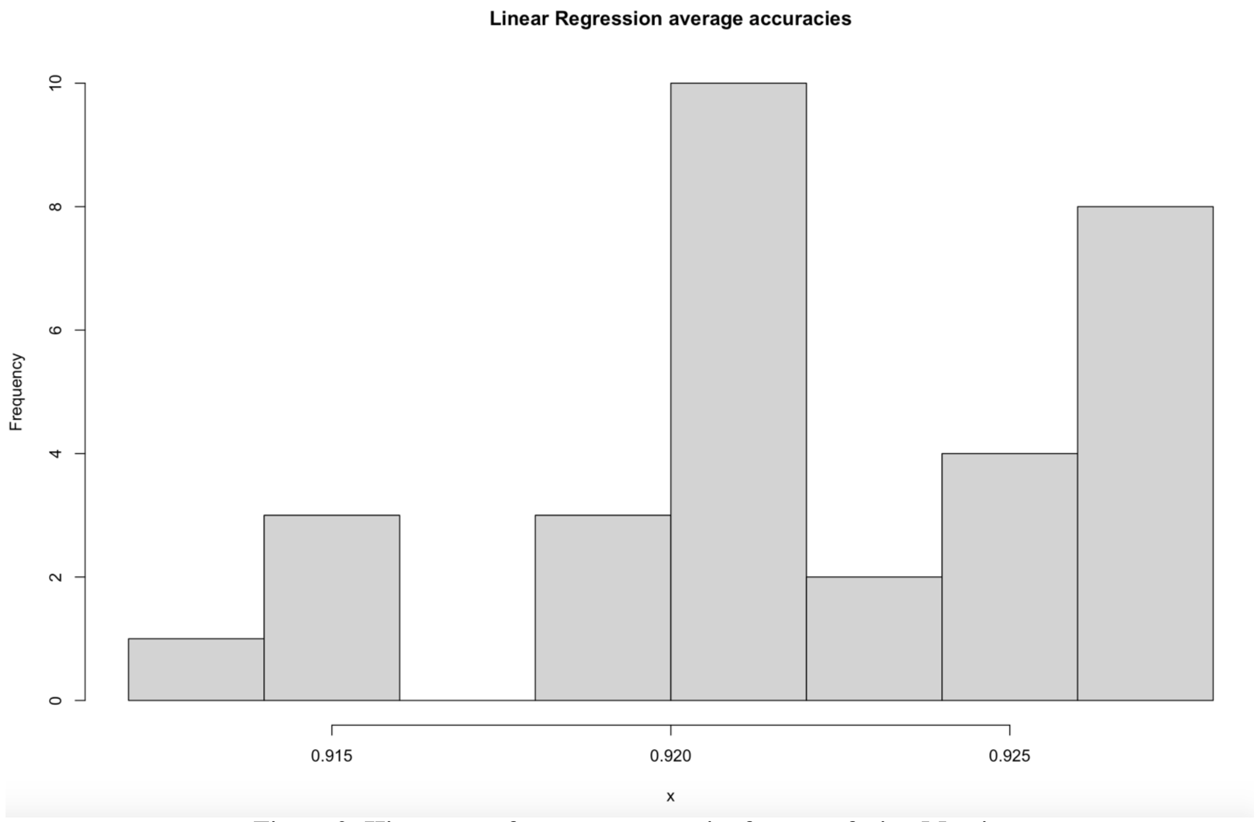

Figure 9: Histogram of average accuracies from confusion Matrix

Above simulation gives results close to 0.3, so we can estimate p=0.3 to accurately predict diabetes outcome in linear regression. Now, calculate overall accuracy of the regression model using all factors.

```
#Logistic Regression model Overall Accuracy
LR_model  <- glm(as.numeric(class=="Positive")~., family = "binomial", data =
train)

summary(LR_model)

Call:
glm(formula = as.numeric(class == "Positive") ~ ., family = "binomial", data
= train)

Deviance Residuals:
    Min        1Q    Median        3Q       Max

-2.89372  -0.22035   0.00148   0.04088   2.35046

Coefficients:

                      Estimate Std. Error z value Pr(>|z|)

(Intercept)          13.069182   2.869034   4.555 5.23e-06 ***
Age                  -0.042913   0.028973  -1.481 0.138577
GenderFemale          4.421507   0.787725   5.613 1.99e-08 ***
PolyuriaNo           -5.295269   1.043595  -5.074 3.89e-07 ***
PolydipsiaNo         -5.921264   1.190787  -4.973 6.61e-07 ***
sudden.weight.lossNo  0.198991   0.707726   0.281 0.778581
weaknessNo           -1.564935   0.700871  -2.233 0.025559 *
PolyphagiaNo         -1.338105   0.685009  -1.953 0.050771 .
Genital.thrushNo     -2.717071   0.743077  -3.657 0.000256 ***
visual.blurringNo    -0.886445   0.837054  -1.059 0.289597
ItchingNo             3.025370   0.843574   3.586 0.000335 ***
IrritabilityNo       -1.177414   0.757366  -1.555 0.120037
delayed.healingNo     0.195336   0.702942   0.278 0.781101
partial.paresisNo    -1.688774   0.705249  -2.395 0.016640 *
muscle.stiffnessNo    1.627605   0.751474   2.166 0.030320 *
AlopeciaNo            0.403597   0.768260   0.525 0.599348
ObesityNo            -0.006143   0.779239  -0.008 0.993710
--
-Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)
    Null deviance: 442.70  on 331  degrees of freedom
Residual deviance: 110.07  on 315  degrees of freedom

AIC: 144.07
Number of Fisher Scoring iterations: 8
```

Prediction

The above trained model is used to predict the diabetes outcome from the test data set and generate the ROC curve plot to represent the accuracy of the model:

```
#Prediction from Linear regression model
predict_test <- predict(LR_model, test, type = 'response')
```

```
#ROC curve calculation
library(ROCR)

roc_predict_train <- predict(LR_model, type = 'response')
roc_prediction <- prediction(roc_predict_train, train$class)
roc_performance <- performance(roc_prediction, 'tpr','fpr')
plot(roc_performance, colorize = TRUE, text.adj = c(-0.2,1.7))
```

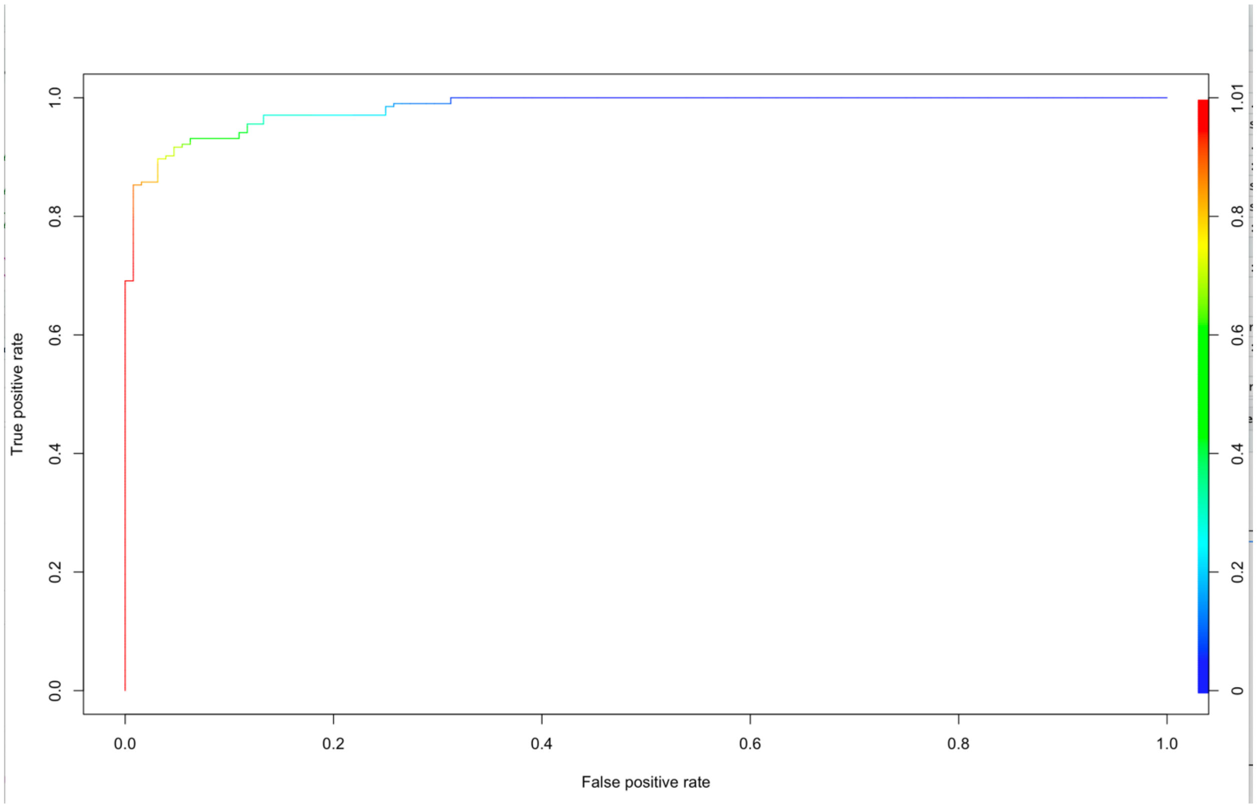

Figure 10: ROC curve to validate the result of prediction

```
#consider p = 0.3 as the measuring probability threshold and if prediction
result is greater than p will be considered patient is diabetic, if
prediction is less than 0.3 will be considered non-diabetic.

x <- ifelse(predict_test > p, "Positive","Negative") %>% factor(levels =
c("Positive","Negative"))

#confusion matrix
cm_lr <- confusionMatrix(x, test$class)

#save accuracy and sensitivity
lr_acc <- cm_lr$overall["Accuracy"]
lr_sen <- cm_lr$byClass["Sensitivity"]

print(lr_acc)
```

```
Accuracy
 0.8928571
```

```
print(lr_sen)

Sensitivity
 0.9615385
```

Logistic regression model produced ~89% accuracy and sensitivity ~96%.

2. K-Nearest Neighbors

The k-nearest neighbor method is used to classify patients by their similarity with other patients. In this approach, we will use the train dataset to build a KNN model and use a test dataset to measure the accuracy of the model.
Here k=3, 5, 7 or 9, number of neighbors will be considered in the KNN model for comparison.

```
#Data Transformation for KNN - Remove Age, Gender and class columns and
convert other column values to boolean (TRUE or FALSE)
knn_mdl_train <- {train[-c(1, 2, 17)]=="Yes"} %>% as_tibble
#Add Gender column with boolean values
knn_mdl_train <- cbind(train[2] == "Male", knn_mdl_train)
#add class column back
knn_mdl_train <- cbind(knn_mdl_train, train[17])

#transform test data set
knn_mdl_test <- {test[-c(1, 2, 17)]=="Yes"} %>% as_tibble
knn_mdl_test <- cbind(test[2] == "Male", knn_mdl_test)

#define k = 3,5,7 and 9
tune_k <- seq(3, 9, by = 2)
folds <- 5
knn_acc_k <- matrix(nrow = folds, ncol = length(tune_k))
set.seed(4, sample.kind="Rounding")
part_data <- createFolds(1:nrow(train), k = folds)

for (i in 1:folds)
{

#create train and test sets and remove the prediction column class
temp_train <- knn_mdl_train %>% slice(-part_data[[i]])
temp_test <- knn_mdl_train %>% slice(part_data[[i]]) %>% select(-c("class"))

#apply KNN algo to estimate the cutoff value of k
knn_acc_k[i,] <- sapply(tune_k, function(k){
  #create the knn model using jaccard distance metric
  temp_knn <- knn(train_set = temp_train,
                  test_set = temp_test,
                  k = k,
                  categorical_target = "class",
                  comparison_measure="jaccard")

                  #predictions
    temp_preds_knn <- temp_knn$test_set_scores$categorical_target %>%
factor(levels = c("Positive","Negative"))

    #create the confusion matrix
    temp_knn_cm <- confusionMatrix(temp_preds_knn, knn_mdl_train %>%
slice(part_data[[i]]) %>% .$class)

    #return the mean of the accuracy and sensitivity
    return(mean(c(temp_knn_cm$overall["Accuracy"],
                  temp_knn_cm$byClass["Sensitivity"])))
})
}

#maximum value of KNN 5 folds accuracy
x<-colMeans(knn_acc_k)
max(x)
```

```
0.9322823
```

```
#find the optimal value of k
opt_k <- tune_k[which.max(colMeans(knn_acc_k))]
print(opt_k)
```

```
3
```

```
#plot
hist(x, main="KNN average accuracies for k = 3,5,7 and 9")
```

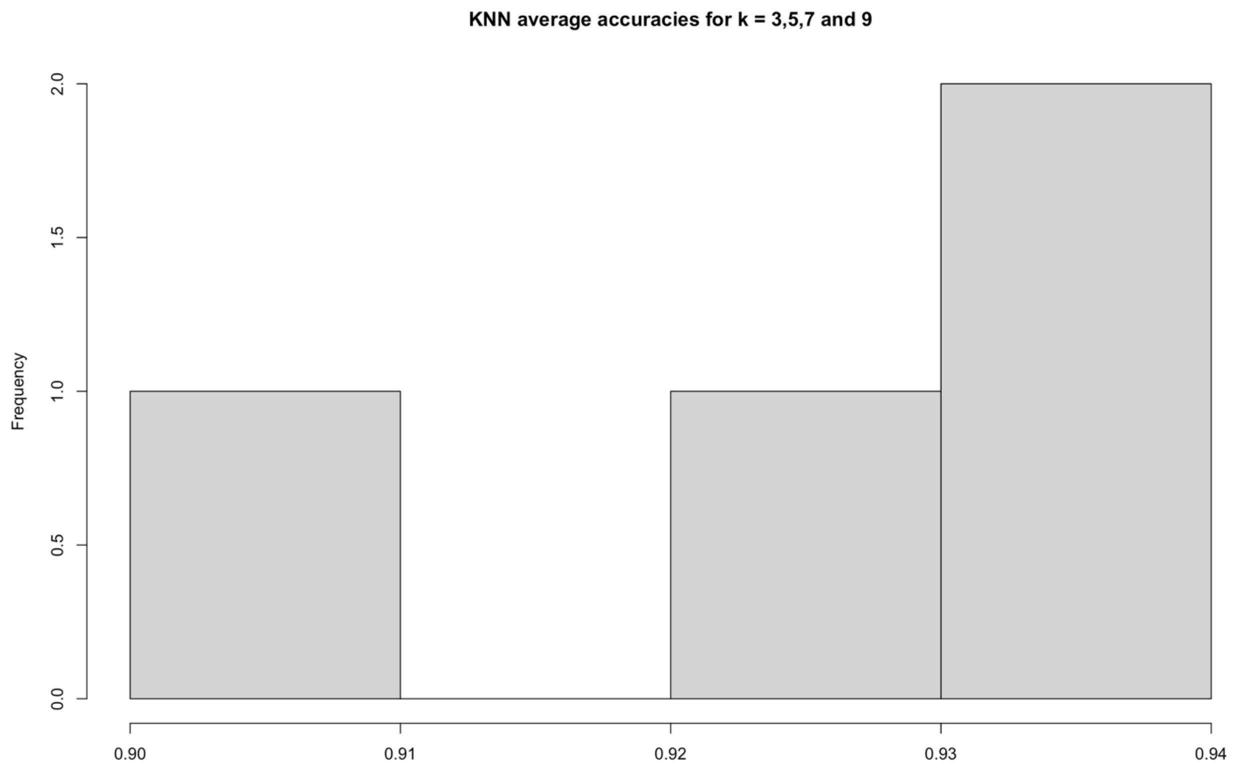

Figure 11: KNN average accuracies for k=3,5,7 and 9

Generate KNN model with optimal value of K and measure the accuracy and sensitivity.

```
#generate a model with optimal value k=3
knn_model <- knn(train_set = knn_mdl_train,
                 test_set = knn_mdl_test,
                 k = opt_k,
                 categorical_target = "class",
                 comparison_measure="jaccard")

#Prediction using test data set
predict_knn <- knn_model$test_set_scores$categorical_target %>% factor(levels
= c("Positive","Negative"))

#Generate confusion matrix
knn_cm <- confusionMatrix(predict_knn, test$class)

#Accuracy and Sensitivity
knn_acc <- knn_cm$overall["Accuracy"]
knn_sen <- knn_cm$byClass["Sensitivity"]

print(knn_acc)
```

```
Accuracy
0.952381
```

```
print(knn_sen)
```

```
Sensitivity
0.9807692
```

Here, KNN model gives accuracy up to ~95% and Sensitivity up to ~98% with the optimal value k=3.

3. Decision Tree

Next, this project will define the classification decision tree which has "nodes" at which data is separated, eventually terminating in "leaves nodes", which define the model's assigned class. The rpart R package is used to create the decision tree. Firstly, the model is constructed to decide the optimal complexity parameter from the simulation. This complexity parameter (cp) will be the deciding factor by which the model needs the performance improvement.

```
#create decision tree model
set.seed(64, sample.kind="Rounding")

#generate the range of cutoff values to decide the optimal cutoff value
dec_tree_model <- train(class~.,
                        method = "rpart",
                        tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)),
                        data = diabetes)

#plot from samples results from decision tree. Optimal cp cutoff value
ggplot(dec_tree_model, highlight = TRUE)
```

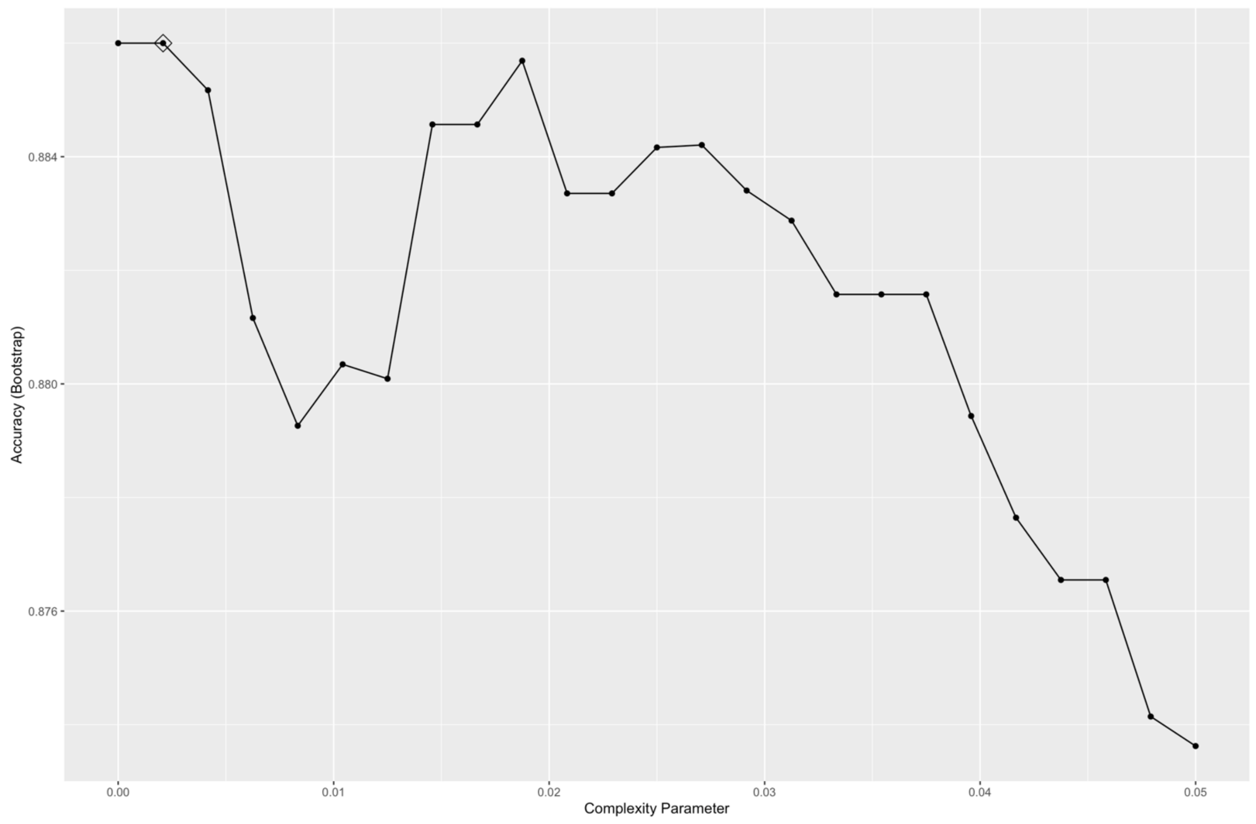

Figure 12: 25 samples results from decision tree accuracy with random 25 predictors

```
#cp error estimation projection plot
dec_tree_model$results %>%
  ggplot(aes(x = cp, y = Accuracy)) +
  geom_line() +
  geom_point() +
  geom_errorbar(aes(x = cp,
                    ymin = Accuracy - AccuracySD,
                    ymax = Accuracy + AccuracySD))
```

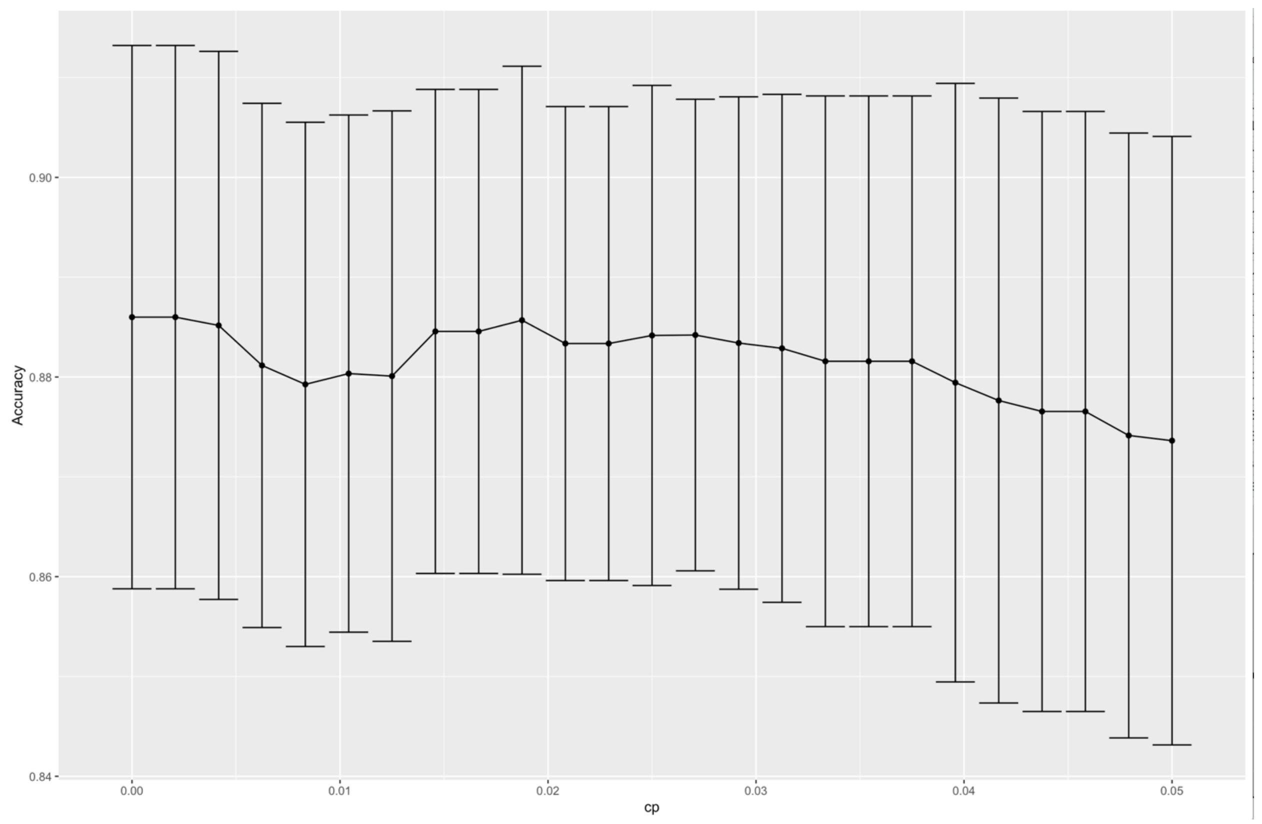

Figure 13: cp error projection plot

This model is further improved to generate the decision tree on train data set with the chosen optimal value of cp from the above projected model.

```
#optimal cp value
cp_cuttoff <- dec_tree_model$bestTune

print(cp_cuttoff)
```

```
0.002083333
```

```
#generate a new model using train data with optimal cp value
dtree <- rpart(class~., cp = cp_cuttoff, data = train)


#plot tree structure
rpart.plot(dtree, type = 5)
title("Decision Tree")
```

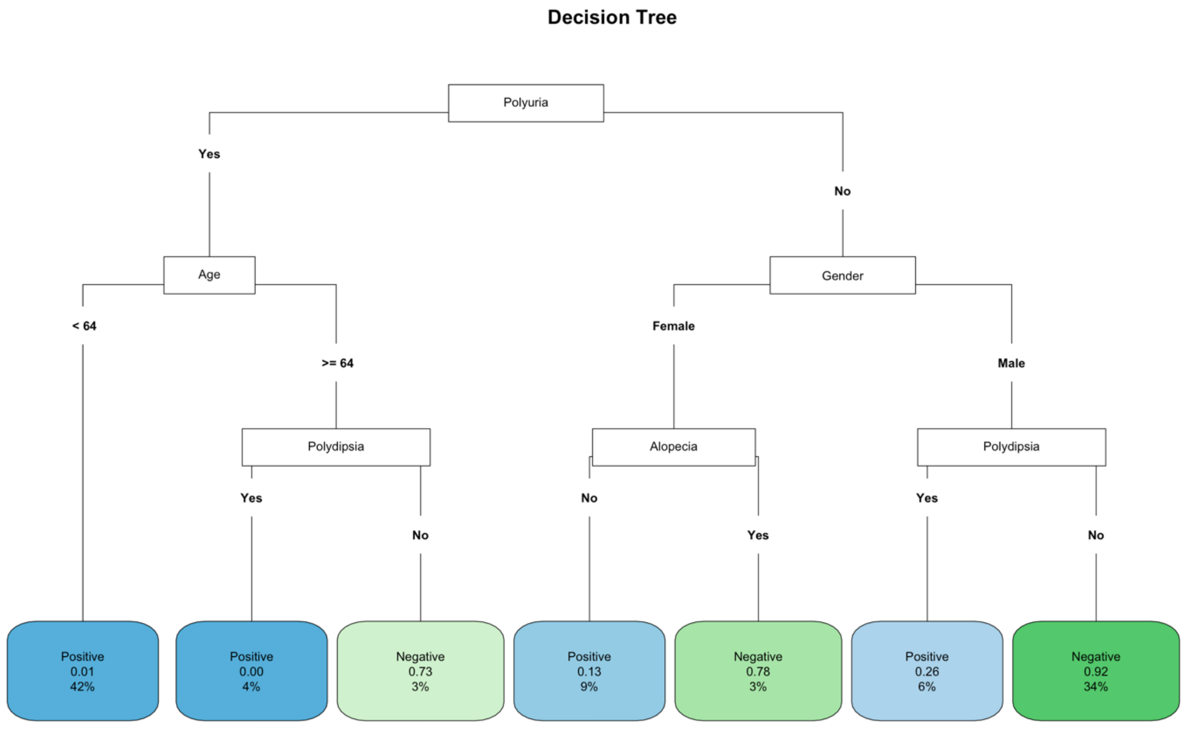

Figure 14: Decision tree classification of diabetes

```
#Measure the importance of feature variables
importance <- t(dtree$variable.importance)
print(importance)
```

Importance precedence order	Variable name	Value
1	Polyuria	67.93315
2	Polydipsia	58.46981
3	Gender	36.71019
4	sudden.weight.loss	28.69226
5	partial.paresis	23.20698
6	Polyphagia	22.78503
7	Age	9.56284
8	Alopecia	5.871147
9	delayed.healing	2.20769
10	muscle.stiffness	2.20769
11	weakness	2.20769

Importance matrix of the decision trees suggests that “Polyuria” symptom is the most important feature variable in the decision tree, then “Polydipsia” is the second most important parameter and so on.
To calculate the model accuracy, this experiment will be repeated to find the optimal tune value by running an algorithm for 5 folds, note that this method is similar to Logistic Regression model technique.

```
#run experiment 5 folds to find the optimal value of tuning parameter
#tune cutoff parameter
tune_p <- seq(0.1, 0.8, by = 0.025)
folds <- 5
temp_dtree_acc <- matrix(nrow = folds, ncol = length(tune_p))
set.seed(4, sample.kind="Rounding")
part_data <- createFolds(1:nrow(train), k = folds)

#The cross validation returns the mean of the accuracy and sensitivity
for(i in 1:folds) {

  #prepare train and test sets
  temp_train <- train %>% slice(-part_data[[i]])
  temp_test <- train %>% slice(part_data[[i]])

  #Accuracy results dtree model with cutoff p
  temp_dtree_acc[i,] <- sapply(tune_p, function(p){

    #decision tree model
    temp_dtree_mod <- rpart(class~., cp = cp_cuttoff, data = temp_train)

    #Prediction calls
    temp_dtree_predict <- predict(temp_dtree_mod, temp_test) %>%
      as_tibble %$%
      ifelse(Positive > p, "Positive", "Negative") %>%
      factor(levels = c("Positive", "Negative"))

    #Generate Confusion Matrix
    temp_dtree_cm <- confusionMatrix(temp_dtree_predict, temp_test$class)

    # Mean of Accuracy and Sensitivity
    return(mean(c(temp_dtree_cm$overall["Accuracy"],
temp_dtree_cm$byClass["Sensitivity"])))

  })
}


#Median is used to define the tune parameter cutoff, this will reduce the
error because of many repetitions of values which are matching the same opt_p

opt_p <- median(tune_p[min_rank(desc(colMeans(temp_dtree_acc)))==1])

print(opt_p)
```

```
0.1
```

```
#apply model on validation dataset
dtree_predict <- predict(dtree, validation) %>%
  as_tibble %$%
  ifelse(Positive > opt_p, "Positive", "Negative") %>%
  factor(levels = c("Positive", "Negative"))

#confusion matrix
dtree_cofmat <- confusionMatrix(dtree_predict, validation$class)

dtree_acc <- dtree_cofmat $overall["Accuracy"]
```

```
dtree_sen <- dtree_cofmat $byClass["Sensitivity"]
print(dtree_acc)
```

```
Accuracy
0.9038462
```

```
print(dtree_sen)
```

```
Sensitivity
  0.984375
```

Overall, the accuracy of the decision tree model is around ~90%. And sensitivity is 98%, which is the worse from KNN model.

4. Random Forest

Random forest is similar to a decision model but this method aggregates multiple decorrelated decision trees to form a forest and perform the prediction.

```
set.seed(11, sample.kind="Rounding")

#create train model
rf<- randomForest(class~., data = train, ntree=100)

#importance of variables
print(rf$importance)
```

```
                      MeanDecreaseGini
Age                          15.958679
Gender                       12.850453
Polyuria                     34.119299
Polydipsia                   31.177819
sudden.weight.loss            7.645953
weakness                      2.876751
Polyphagia                    4.346428
Genital.thrush                3.354113
visual.blurring               4.854486
Itching                       4.749277
Irritability                  4.057589
delayed.healing               4.626114
partial.paresis               8.236052
muscle.stiffness              4.691222
Alopecia                      5.581028
Obesity                       2.732245
```

From above matrix, “Polyuria” is the most important feature variable and “**Polydipsia**” is second most important variable, and so on.

```
#Random Forest tree structure all terminal nodes will be shown as -1
getTree(rf, 1, labelVar=TRUE)
```

```
left daughter right daughter          split var split point status prediction
1              2              3         Polydipsia           1      1
<NA>
2              4              5           Polyuria           1      1
<NA>
3              6              7     Genital.thrush           1      1
<NA>
4              0              0               <NA>           0     -1
Positive
5              8              9            Obesity           1      1
<NA>
6             10             11    partial.paresis           1      1
<NA>
7             12             13             Gender           1      1
<NA>
8              0              0               <NA>           0     -1
Positive
9             14             15     Genital.thrush           1      1
<NA>
10             0              0               <NA>           0     -1
Positive
11            16             17           Polyuria           1      1
<NA>
12            18             19           Polyuria           1      1
<NA>
13            20             21           weakness           1      1
<NA>
14             0              0               <NA>           0     -1
Positive
15            22             23    visual.blurring           1      1
<NA>
16            24             25                Age          42      1
<NA>
17            26             27           weakness           1      1
<NA>
18            28             29           Alopecia           1      1
<NA>
19            30             31           weakness           1      1
<NA>
20            32             33 sudden.weight.loss           1      1
<NA>
21            34             35 sudden.weight.loss           1      1
<NA>
22            36             37                Age          63      1
<NA>
23             0              0               <NA>           0     -1
Positive
24             0              0               <NA>           0     -1
Positive
25             0              0               <NA>           0     -1
Positive
26             0              0               <NA>           0     -1
Negative
27            38             39       Irritability           1      1
<NA>
28            40             41         Polyphagia           1      1
<NA>
```

```
29              0               0                <NA>           0     -1
Positive
30             42              43          Polyphagia           1      1
<NA>
31             44              45            Alopecia           1      1
<NA>
32              0               0                <NA>           0     -1
Positive
33              0               0                <NA>           0     -1
Negative
34              0               0                <NA>           0     -1
Negative
35             46              47            Alopecia           1      1
<NA>
36              0               0                <NA>           0     -1
Negative
37              0               0                <NA>           0     -1
Positive
38              0               0                <NA>           0     -1
Positive
39             48              49                 Age          58      1
<NA>
40             50              51            weakness           1      1
<NA>
41              0               0                <NA>           0     -1
Positive
42              0               0                <NA>           0     -1
Negative
43             52              53     visual.blurring           1      1
<NA>
44             54              55          Polyphagia           1      1
<NA>
45              0               0                <NA>           0     -1
Negative
46              0               0                <NA>           0     -1
Positive
47              0               0                <NA>           0     -1
Positive
48              0               0                <NA>           0     -1
Negative
49              0               0                <NA>           0     -1
Positive
50             56              57     visual.blurring           1      1
<NA>
51             58              59    muscle.stiffness           1      1
<NA>
52             60              61             Itching           1      1
<NA>
53             62              63             Itching           1      1
<NA>
54             64              65             Obesity           1      1
<NA>
55              0               0                <NA>           0     -1
Negative
56              0               0                <NA>           0     -1
Negative
```

```
57              0               0                <NA>           0     -1
Positive
58              0               0                <NA>           0     -1
Negative
59              0               0                <NA>           0     -1
Positive
60              0               0                <NA>           0     -1
Negative
61              0               0                <NA>           0     -1
Positive
62              0               0                <NA>           0     -1
Negative
63             66              67     delayed.healing           1      1
<NA>
64              0               0                <NA>           0     -1
Negative
65              0               0                <NA>           0     -1
Positive
66              0               0                <NA>           0     -1
Positive
67              0               0                <NA>           0     -1
Negative

#do the prediction based on 100 trees random forest
rf_prdt <- predict(rf, test)

#generate confusion matrix
rf_cm <- confusionMatrix(rf_prdt, test$class)

#Accuracy and Sensitivity
rf_100_acc <- rf_cm$overall["Accuracy"]
rf_100_sen <- rf_cm$byClass["Sensitivity"]

print(rf_100_acc)

Accuracy
0.9761905

print(rf_100_sen)

Sensitivity
  0.9807692
```

From above model simulations we can notice that accuracy is improved and close to ~98%. And sensitivity is also ~98%.

Final Model Selection

From all above models, Logistic Regression performs worse and is not well suited for medical conditions like diabetes analysis. KNN and Decision Tree models give better outcomes than Logistic Regression, but they are also not best suited models. Random forest gives better prediction results and seems more accurate. Now, we will do the validation on the Random forest as our final model.

Random Forest - Final Result Validation

```
set.seed(1, sample.kind="Rounding")
#create train model using diabetes data set
rf_final <- randomForest(class~., data = diabetes, ntree=100)

#do the prediction based on 100 trees random forest
rf_predict_final <- predict(rf_final, validation)

#generate confusion matrix
rf_cm_final <- confusionMatrix(rf_predict_final, validation$class)

#Accuracy and Sensitivity
rf_acc_final <- rf_cm_final$overall["Accuracy"]
rf_sen_final <- rf_cm_final$byClass["Sensitivity"]

print(rf_acc_final)
```

```
Accuracy
 0.9903846
```

```
print(rf_sen_final)
```

```
Sensitivity
 1.0000000
```

```
print(rf_cm_final)
```

```
Confusion Matrix and Statistics
          Reference
Prediction Positive Negative
  Positive       64        1
  Negative        0       39

               Accuracy : 0.9904
                 95% CI : (0.9476, 0.9998)
    No Information Rate : 0.6154
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9796

 Mcnemar's Test P-Value : 1

            Sensitivity : 1.0000
            Specificity : 0.9750
         Pos Pred Value : 0.9846
         Neg Pred Value : 1.0000
             Prevalence : 0.6154
         Detection Rate : 0.6154
   Detection Prevalence : 0.6250
      Balanced Accuracy : 0.9875

       'Positive' Class : Positive
```

From the Random Forest model, this project is able to achieve the accuracy up to ~99% and higher Sensitivity of 100% success rate, which implies that this model is likely to correctly classify diabetic patients.

Conclusion

Diabetes risk prediction project has used Kaggle's UCI diabetes data set to detect the early stage of diabetes on the given sample of patient's data. This data set is further split into a training set and validation set and examines various features effects to estimate the best suited model for diabetes disease prediction. Logistic regression and K-Nearest Neighbors, Decision Tree and Random Forest models have been analyzed. From all models result observations, Random Forest performs the best and produces the improved accuracy and sensitivity.

Future Scope

There are various possibilities to improve the ML models by introducing the new feature variables like Fasting blood glucose, insulin level, high-density lipoprotein, triglycerides, demographic information with family history and hereditary, smoking and alcohol intake percentage, ect. Also, the shape of the data generated by a large cohort plays an important role while building a better performing model. There is also a possibility to build a model to classify Type 1 and Type 2 diabetes separately based on demographics, lifestyle and family history parameters.

References

1. UCI Data set Information: https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.#
2. UCI Data download link : https://archive.ics.uci.edu/ml/machine-learning-databases/00529/
3. Data Science Book: https://rafalab.github.io/dsbook/
4. R tutorial Help from R-Studio