# Load required libraries

library(e1071) # for Naive Bayes classifier

library(randomForest) # for Random Forest classifier

library(ggplot2) # for plotting

# Read the diabetes dataset

data <- read.csv('/Users/DataScienceClass/Desktop/R/diabetes\_prediction\_dataset.csv')

# Check column names

colnames(data)

# Explore the structure of the dataset

str(data)

# Split the data into training and testing sets

set.seed(123) # for reproducibility

train\_index <- sample(1:nrow(data), 0.8\*nrow(data)) # 80% for training

train\_data <- data[train\_index, ]

test\_data <- data[-train\_index, ]

# Define features and target variable

features <- names(data)[-9] # excluding the target variable 'diabetes'

target <- "diabetes"

# Train Naive Bayes classifier

nb\_model <- naiveBayes(as.factor(diabetes) ~ ., data = train\_data)

# Train another model (Random Forest for example)

rf\_model <- randomForest(as.factor(diabetes) ~ ., data = train\_data)

# Predictions on test data

nb\_pred <- predict(nb\_model, newdata = test\_data, type = "class")

rf\_pred <- predict(rf\_model, newdata = test\_data)

# Confusion matrix for Naive Bayes

nb\_conf\_matrix <- table(Actual = test\_data$diabetes, Predicted = nb\_pred)

# Confusion matrix for Random Forest

rf\_conf\_matrix <- table(Actual = test\_data$diabetes, Predicted = rf\_pred)

# Accuracy calculation

nb\_accuracy <- sum(diag(nb\_conf\_matrix)) / sum(nb\_conf\_matrix) \* 100

rf\_accuracy <- sum(diag(rf\_conf\_matrix)) / sum(rf\_conf\_matrix) \* 100

# Combine predictions and actual diabetess for plotting

predictions <- data.frame(Actual = test\_data$diabetes,

Naive\_Bayes = nb\_pred,

Random\_Forest = rf\_pred)

# Plotting

ggplot(predictions, aes(x = Actual, fill = Naive\_Bayes)) +

geom\_bar(position = "dodge") +

labs(title = "Comparison of Actual vs Predicted (Naive Bayes)",

x = "diabetes",

y = "Count") +

theme\_minimal()

ggplot(predictions, aes(x = Actual, fill = Random\_Forest)) +

geom\_bar(position = "dodge") +

labs(title = "Comparison of Actual vs Predicted (Random Forest)",

x = "diabetes",

y = "Count") +

theme\_minimal()

# Final dataset with predictions

final\_dataset <- cbind(test\_data, Naive\_Bayes\_Prediction = nb\_pred, Random\_Forest\_Prediction = rf\_pred)

# Load necessary libraries

library(caret)

# Load data

data <- read.csv('/Users/DataScienceClass/Desktop/R/diabetes.csv')

str(models)

# Define columns names

colnames(data) <- c("Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome")

# Convert 'Outcome' to factor

data$Outcome <- factor(data$Outcome, levels = c(0, 1), labels = c("No", "Yes"))

# Perform stratified sampling

set.seed(123)

train\_index <- createDataPartition(data$Outcome, p = 0.75, list = FALSE)

train <- data[train\_index, ]

test <- data[-train\_index, ]

# Define the control parameters

ctrl <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = twoClassSummary)

# Define models

models <- list(

C50 = list(

model = "C5.0",

train\_function = train

),

C50\_PCA = list(

model = "C5.0",

preProcess = "pca",

train\_function = train

),

NaiveBayes = list(

model = "nb",

train\_function = train

),

NeuralNetwork = list(

model = "nnet",

train\_function = train

),

SVM\_SMO = list(

model = "svmLinear",

train\_function = train

),

RandomForest = list(

model = "rf",

train\_function = train

)

)

# Train models

results <- lapply(models, function(model) {

train\_function <- model$train\_function

if (length(train\_function) == 1) {

train\_function <- as.character(train\_function)

if (exists(train\_function)) {

train\_function <- get(train\_function)

if (exists("model$preProcess", model)) {

train\_function(Outcome ~ ., data = train, method = model$model, trControl = ctrl, preProcess = model$preProcess)

} else {

train\_function(Outcome ~ ., data = train, method = model$model, trControl = ctrl)

}

} else {

stop(paste("Train function", train\_function, "not found"))

}

} else {

stop("train\_function should be a single function name")

}

})

# Compare models

model\_results <- resamples(results)

# Summarize results

summary(model\_results)

# Plot results

bwplot(model\_results, metric = "Accuracy")

# Check the best model

best\_model <- getModelInfo()$C50$label

# Print the best model

print(paste("The best model is:", best\_model))

## Loading the required libraries

library(ggplot2)

library(ggvis)

library(corrplot)

library(caTools)

library(ROCR)

## Data Loading

The observations of the people are stored in a CSV format, named diabetes.csv.The data is loaded in the environment.Let's check how the data is structured.

data = read.csv("C:/Users/crsri/Documents/Diabetes\_Prediction/Data/diabetes.csv")

head(data)

summary(data)

str(data)

## Correlations

The proportionalities of the attributes of the data can be identified by the correlation coefficient,either numerically or visually.They helps to know which attributes are highy dependent on the prediction variable:Outcome.

correlations <- cor(data)

correlations

corrplot(correlations, method="color")

## Visualization

Visualizations are used to grasp the structure of data and its relations,like how they vary and their

relationships with the otehr data.They are said to be EDA.

A matrix of scatterplots is produce for this dataset.

pairs(data, col=data$Outcome)

### Glucose and Insulin

The glucose and the insulin are the major factors of the diabetes...which in turn have direct proportionality in the future during the diabetes.They are the major cause of the occurence.They are strong correlated on each other.

data %>% ggvis(~Glucose,~Insulin,fill =~Outcome) %>% layer\_points()

### BMI ad DiabetesPedigreeFunction

The BMI and DiabetesPedigreeFunction is plotted here.

data %>% ggvis(~BMI,~DiabetesPedigreeFunction,fill =~Outcome) %>% layer\_points()

### Age and Pregnancies

The males have 0 for the pregnancy attribute, which is why we find a lot of values plottinh zero in this grpah.

data %>% ggvis(~Age,~Pregnancies,fill =~Outcome) %>% layer\_points()

## Preparing the data

The dataset is divided as two parts, training data and testing data, with a Splitratio of 0.75. It means that 2/3rds of the data is labelled by training set and the rest 1/3rd of data is the testing set.The division of the dataset is by means of a random order generated by the seed.

set.seed(88)

split <- sample.split(data$Outcome, SplitRatio = 0.75)

data\_train <- subset(data, split == TRUE)

data\_test <- subset(data, split == FALSE)

## Logistic regression

The Logistic regression helps to classify the concern person will get diabetes or not.Since we are using the logistic regression we have to mention that, family = binomial.We are using all the attributes we have in the dataset.Let us take a look at the summary.

model <- glm (Outcome ~ .-Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age, data = data\_train, family = binomial)

summary(model)

## Prediction

The trained model is used to predict the data for the testing data and for the training data(For checking accuracy purposes and for ROC curve)

predict\_train <- predict(model, type = 'response')

predict\_test <- predict(model, newdata = data\_test, type = 'response')

## ROC Curve

ROCRpred <- prediction(predict\_train, data\_train$Outcome)

ROCRperf <- performance(ROCRpred, 'tpr','fpr')

plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))

## Comparison

By comparing the real values with the real data, we can see the how our machine learning algorithm performs.

predict\_test\_c = predict\_test

i = 1

while(i <= length(predict\_test))

{

if(predict\_test[i] < 0.5)

predict\_test\_c[i] = 0

else

predict\_test\_c[i] = 1

i = i + 1;

}

compare <- data.frame(data\_test$Outcome,predict\_test\_c)

colnames(compare) <- c("Observed Values","Predicted values")

ggplot(data = compare,aes(x = "Observed Values", y = "Predicted values")) + geom\_abline() +

xlab("Observed Values") + ylab("Predicted values") + theme\_classic()

compare

TEST 1

## Loading the required libraries

library(ggplot2)

library(ggvis)

library(corrplot)

library(caTools)

library(ROCR)

## Data Loading

data <- read.csv('/Users/DataScienceClass/Desktop/R/diabetes.csv')

head(data)

summary(data)

str(data)

## Correlations

correlations <- cor(data)

correlations

corrplot(correlations, method="color")

## Visualization

pairs(data, col=data$Outcome)

### Glucose and Insulin

ggvis(data, ~Glucose, ~Insulin, fill =~Outcome) %>%

layer\_points()

### BMI ad DiabetesPedigreeFunction

ggvis(data, ~BMI, ~DiabetesPedigreeFunction, fill =~Outcome) %>%

layer\_points()

### Age and Pregnancies

ggvis(data, ~Age, ~Pregnancies, fill =~Outcome) %>%

layer\_points()

## Preparing the data

set.seed(88)

split <- sample.split(data$Outcome, SplitRatio = 0.75)

data\_train <- subset(data, split == TRUE)

data\_test <- subset(data, split == FALSE)

## Logistic regression

model <- glm (Outcome ~ .-Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age, data = data\_train, family = binomial)

summary(model)

## Prediction

predict\_train <- predict(model, type = 'response')

predict\_test <- predict(model, newdata = data\_test, type = 'response')

## ROC Curve

ROCRpred <- prediction(predict\_train, data\_train$Outcome)

ROCRperf <- performance(ROCRpred, 'tpr','fpr')

plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))

## Comparison

predict\_test\_c = predict\_test

i = 1

while(i <= length(predict\_test))

{

if(predict\_test[i] < 0.5)

predict\_test\_c[i] = 0

else

predict\_test\_c[i] = 1

i = i + 1;

}

compare <- data.frame(data\_test$Outcome,predict\_test\_c)

colnames(compare) <- c("Observed Values","Predicted values")

ggplot(data = compare,aes(x = "Observed Values", y = "Predicted values")) + geom\_abline() +

xlab("Observed Values") + ylab("Predicted values") + theme\_classic()

compare

TEST 2

## Loading the required libraries

library(ggplot2)

library(ggvis)

library(corrplot)

library(caTools)

library(ROCR)

## Data Loading

data <- read.csv('/Users/DataScienceClass/Desktop/R/diabetes.csv')

head(data)

summary(data)

str(data)

## Data Cleaning

# Assuming you want to handle missing values by removing rows with missing values

data <- na.omit(data)

## Feature Scaling

# You can scale the numeric features if needed

numeric\_cols <- sapply(data, is.numeric)

data[numeric\_cols] <- lapply(data[numeric\_cols], scale)

## Correlations

correlations <- cor(data)

correlations

corrplot(correlations, method="color")

## Visualization

pairs(data, col=data$Outcome)

### Glucose and Insulin

ggvis(data, ~Glucose, ~Insulin, fill =~Outcome) %>%

layer\_points()

### BMI ad DiabetesPedigreeFunction

ggvis(data, ~BMI, ~DiabetesPedigreeFunction, fill =~Outcome) %>%

layer\_points()

### Age and Pregnancies

ggvis(data, ~Age, ~Pregnancies, fill =~Outcome) %>%

layer\_points()

## Preparing the data

set.seed(88)

split <- sample.split(data$Outcome, SplitRatio = 0.75)

data\_train <- subset(data, split == TRUE)

data\_test <- subset(data, split == FALSE)

## Logistic regression

# Assuming the threshold is 0, you can use a threshold of 0.5 to classify the values

data\_train$Outcome <- ifelse(data\_train$Outcome > 0.5, 1, 0)

# Verify the unique values again

unique(data\_train$Outcome)

model <- glm (Outcome ~ .-Pregnancies + Glucose + BloodPressure + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age, data = data\_train, family = binomial)

summary(model)

## Prediction

predict\_train <- predict(model, type = 'response')

predict\_test <- predict(model, newdata = data\_test, type = 'response')

## ROC Curve

ROCRpred <- prediction(predict\_train, data\_train$Outcome)

ROCRperf <- performance(ROCRpred, 'tpr','fpr')

plot(ROCRperf, colorize = TRUE, text.adj = c(-0.2,1.7))

## Comparison

predict\_test\_c = predict\_test

i = 1

while(i <= length(predict\_test))

{

if(predict\_test[i] < 0.5)

predict\_test\_c[i] = 0

else

predict\_test\_c[i] = 1

i = i + 1;

}

compare <- data.frame(data\_test$Outcome,predict\_test\_c)

colnames(compare) <- c("Observed Values","Predicted values")

ggplot(data = compare,aes(x = "Observed Values", y = "Predicted values")) + geom\_abline() +

xlab("Observed Values") + ylab("Predicted values") + theme\_classic()

compare