m5: Diabetes Mellitus Analysis and Classification

Hiloni Mehta, Ria Lulla, Kheman Garg

August 3, 2021

Description of the Problem:

Diabetes is one of the health concerns that affects millions of people throughout the world [1]. More than half of the people suffering from it are women [1]. Although diabetes cannot be cured, detecting diabetes at an early stage can help one to take appropriate measures. Hence, the objective of this study is to analyze and successfully predict diabetes in patients based on various medical characteristics of the Pima Indian population. In this research, we make a comprehensive exploration of various machine learning methods used to identify and classify whether the patients have diabetes or not. We compare the different algorithms to obtain the best accuracy.

The Function of the System:

Firstly, we will analyze the features that can be considered as indicators of diabetes and their correlation to diabetes. We also plan to create exploratory data visualizations to summarize the various characteristics of the data. The target variable in the dataset is a binary classifier that shows if the person is diabetic or not.

Based on the Pima Indian Diabetes dataset, we plan to efficiently perform classification and analyze the performance of various supervised learning algorithms such as Support Vector Machine (SVM), Decision Trees, Logistic Regression, K-Nearest Neighbors (KNN). The data will be split into a training set and testing set and stratification shall be performed to evaluate the performance of the model. Furthermore, the performance of the models will also be analyzed based on various metrics such as accuracy, specificity, precision, F-score, etc.

Motivation and Benefits of the Study:

Diabetes Mellitus is a condition that occurs when the blood glucose levels are too high. Glucose is what gives energy to our body [1]. It is the simplest form of sugar that our carbohydrates get broken down into during metabolism [1]. Our body cells need glucose as our source of energy and the process of maintaining a balance of glucose in cells and blood, is taken care of by a hormone called Insulin, which is released by the pancreas [2]. In diabetes, the accumulation of glucose in the blood can cause serious complications including heart diseases, skin diseases, or cardiovascular issues. Detecting diabetes at an early stage is quite challenging as it depends upon various factors. Analysis of the given data set will help us understand and develop various models that can help predict diabetes accurately. Using this predictive tool, one can make a fairly accurate prediction of whether the patient has diabetes or not.

Related Work

Study of the existing systems:

In the paper [20], the author performs comparison of machine learning techniques in order to classify the Pima Indian Diabetes Data set. The classification is done with various techniques like Decision Trees, Logistic Regression, Discriminant analysis, Support Vector Machines, KNN and ensemble learners [20] using the MATLAB software.They applied multiple machine learning techniques in order to classify it in 10-fold cross-validation fashion and achieved accuracy scores of around 70% for all models. However, this system has certain drawbacks. One of the drawback is that they have not considered any pre-processing methods like scaling, normalization and fine tuning methods which can help in improving the performance of the models.

Similarly, another paper [21], focuses on comparing the evaluation matrix of k-NN, RBF SVM, Linear SVM, Sigmoid SVM, Logictic Regression, Linear Discriminant Analysis, Classfication and Regression Trees, and Naive Bayes classification models. They try to achieve a system which provides best results using fine tuning of algorithms and 10 fold cross validation. However this approach also does not consider any methods for data cleaning and pre-processing because the absence of data reduces statistical power and the lost data can cause bias in the estimation of parameters [9].

These studied researches are informative in classifying the Pima indian diabetes dataset. This is because both these research papers consider cross validation as a very important step while evaluating various models. Hence, we take cross validation into consideration. Cross validation helps in assessing the effectiveness of the models [22]. Our project has a motive to compare various machine learning algorithms to help us understand and develop various models that predict diabetes accurately. We have considered data cleaning and replacing the missing values using data imputation methods as well. We even take into consideration various pre-processing techniques and cross validation and fine tuning method to improve the classification performance and avoid stochastic results.

Data and Core Algorithm:

Our dataset "Pima Indians Diabetes Database" is publicly available at the UCI Machine Learning repository, and is curated by the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset consists of records of the physical conditions of women who are Indian, native Americans in Arizona. All-female patients are at least 21 years old. The dataset contains 768 patient records and 9 features, out of which 8 of them are predictor (dependent) variables and one is a target (independent) variable. These features are as follows:

1. Pregnancies: Number of times the patient has been pregnant
2. Glucose(mg/dl) : Plasma Glucose Concentration levels
3. Blood Pressure(mm Hg): Diastolic Blood Pressure
4. Skin Thickness (mm): Triceps Skin Fold Thickness
5. Insulin(μ/ml) : 2-hour Serum Insulin
6. BMI (kg/m^2) : Body Mass Index (Weight/height)
7. Diabetes Pedigree Function: Measures diabetic genetic relationship and influence on patient's eventual diabetes risk
8. Age (years): Age of the patient
9. Outcome: Binary classifier which helps to determine whether the patient has diabetes or not

We plan on implementing Support Vector Machine as our core algorithm to build the classification model. In SVM, binary classification can be performed by finding the hyperplane and separating the training data into two classes. The ability of SVM in the case of a nonlinear function is that it can transform highly dimensional data into linearly separable data [3].

EXPLORATORY DATA ANALYSIS AND VISUALIZATION

Summary of the data:

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

Visualization of the Class Variable

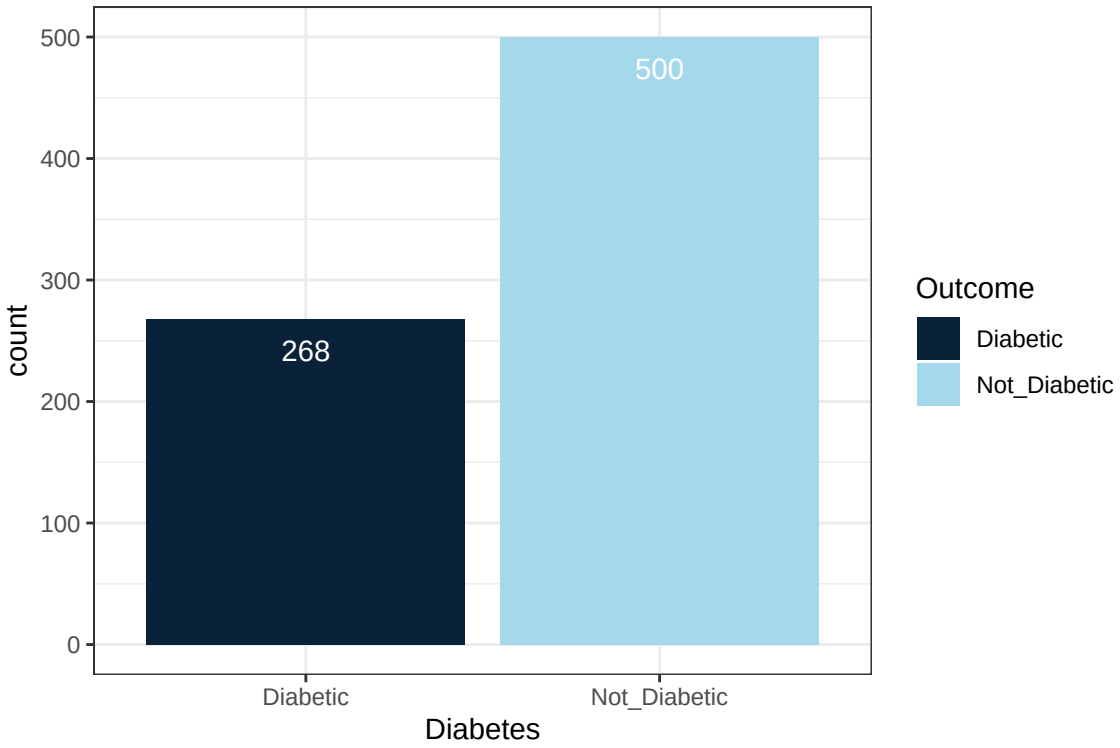


Figure 1: Distribution of Class variable

The dataset has 268 (34.8958333%) women who have diabetes and the remaining 500 (65.1041667%) who have not been diagnosed with the disease. The bar plot provides useful insights on the distribution of the target class i.e the value of the Outcome as Diabetic or Non-diabetic. We can see that there are more samples of women that are not diabetic than the ones that are diabetic.

Checking for Missing Values

```
##                 Glucose           BloodPressure           SkinThickness
##                       5                      35                     227
##                 Insulin                     BMI DiabetesPedigreeFunction
##                     374                      11                       0
##                     Age
##                       0
```

The above analysis indicates the number of zeroes present in the dataset. An analysis on zero values would not correctly classify the dataset. Removing these missing values is not appropriate in this case due to the small data size. Replacing the missing values with mean or median introduces bias in the data which causes a decrease in variance. [4] [5]

Visualization of the missing data

The plot shows that almost 51% of the samples are not missing any information, 25% of the data missing is from the Insulin and SkinThickness values, and the remaining ones have significantly less number of missing values($< 11\%$).

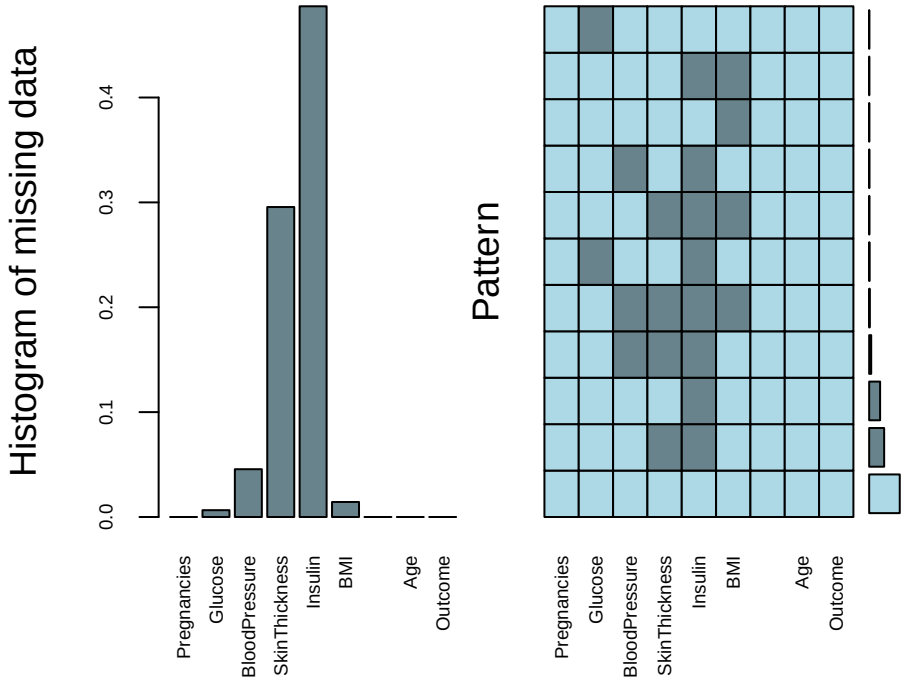


Figure 2: Missing Data Pattern

Imputing the Missing Values using Mice

```
set.seed(123)
# Multiple imputation Using MICE
mice_data <- mice(data, m = 5, method='pmm')
```

The MICE (Multivariate Imputation via Chained Equations) package in R, helps in creating multiple imputations as compared to a single imputation (such as mean). This package is used to replace missing values with plausible values to estimate more realistic regression coefficients that are not affected by missing values. It takes care of the uncertainty in missing values and obtains approximately unbiased estimates [4] [5].

The method of imputation in the MICE package that we used is Predictive Mean Matching (PMM) and the number of imputations is 5, which are default values. "Predictive Mean Matching (PMM) is similar to the regression method except that for each missing value, it fills in a value randomly from among the observed donor values from an observation whose regression-predicted values are closest to the regression-predicted value for the missing value from the simulated regression model." (Heitjan and Little 1991; Schenker and Taylor 1996) [8] [9]

Comparing the density plots after multiple imputation

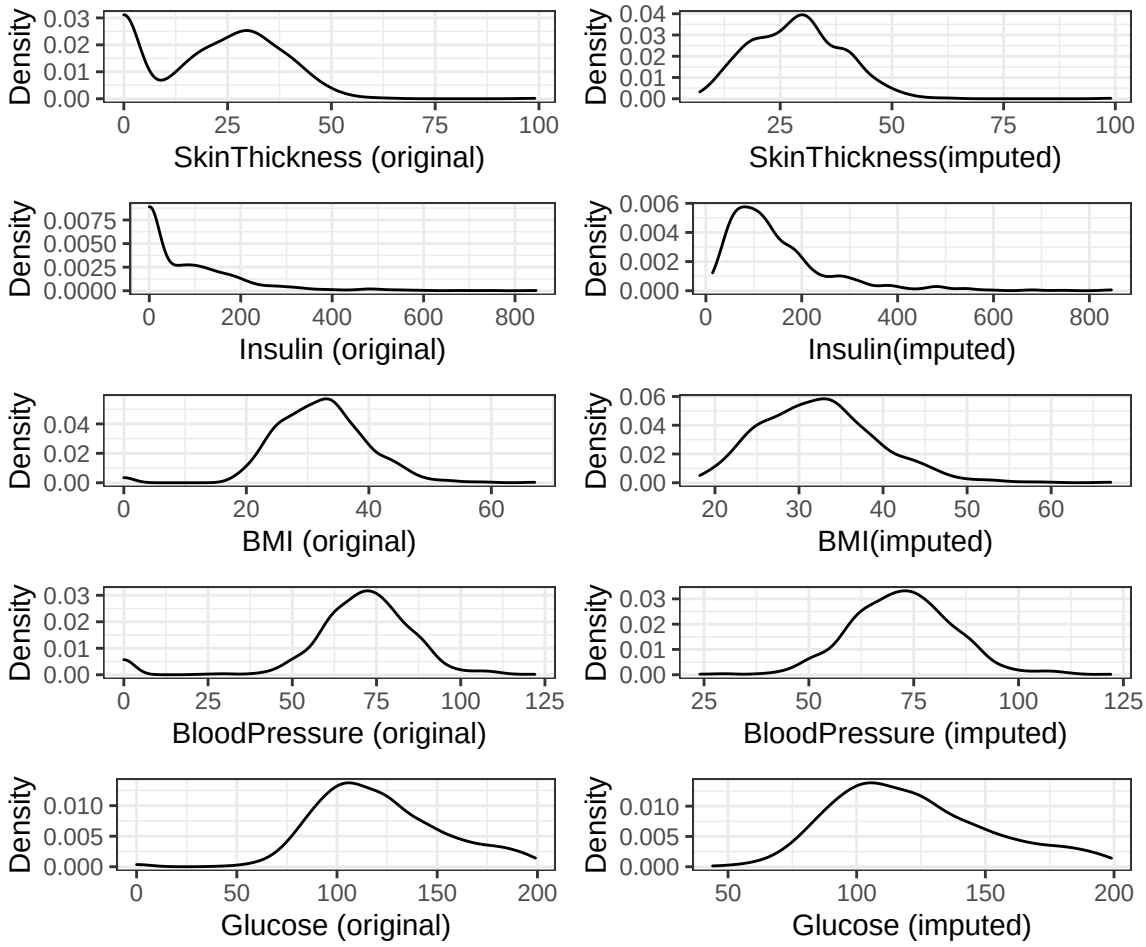


Figure 3: Density Plots comparison before and after multiple imputation

We can see that the density plots for original data distributions in comparison to imputed data distribution for the variables haven't changed significantly. Hence, the multiple imputed values using MICE package can be considered as plausible values.

UNIVARIATE ANALYSIS

Density Plots

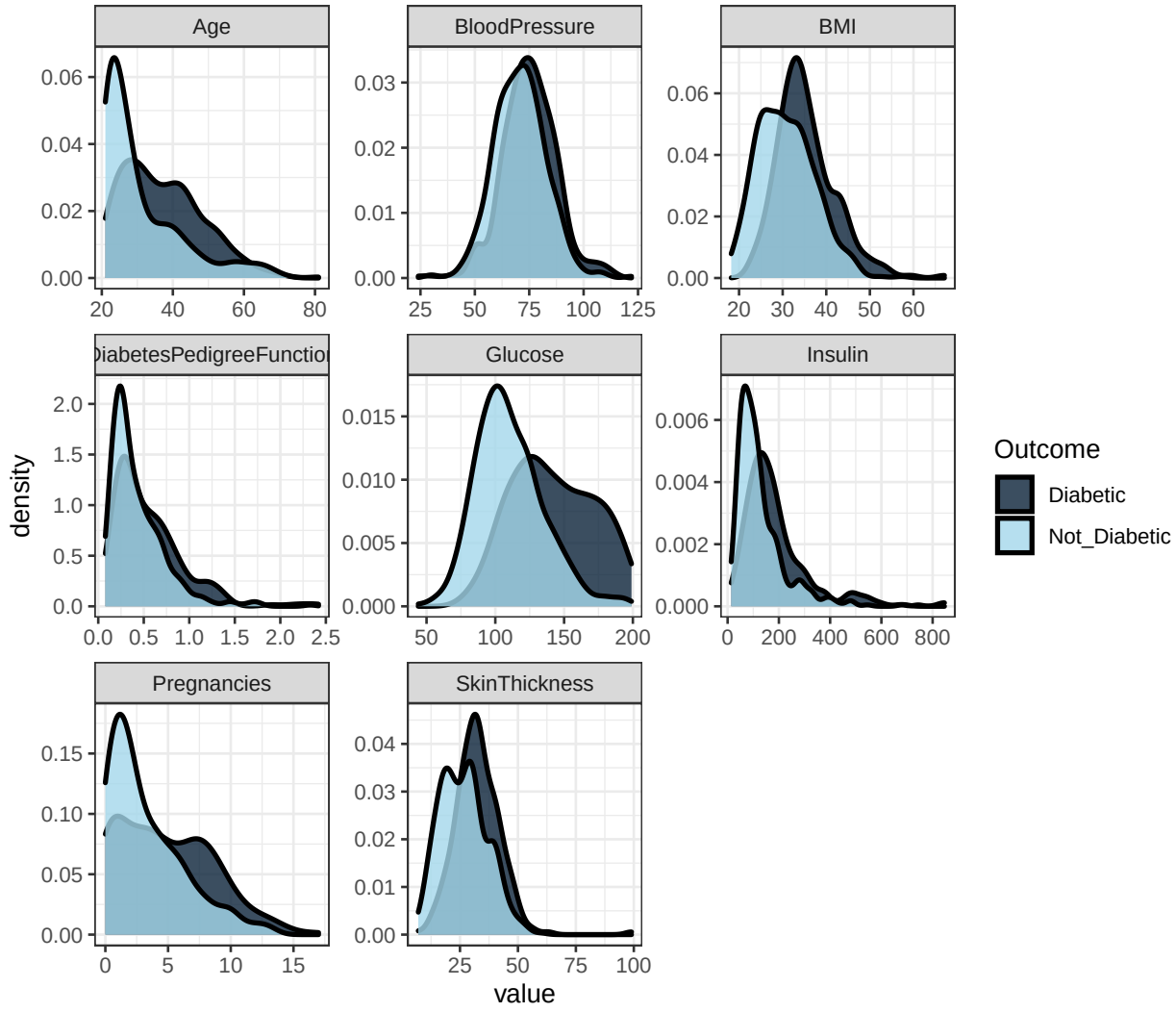
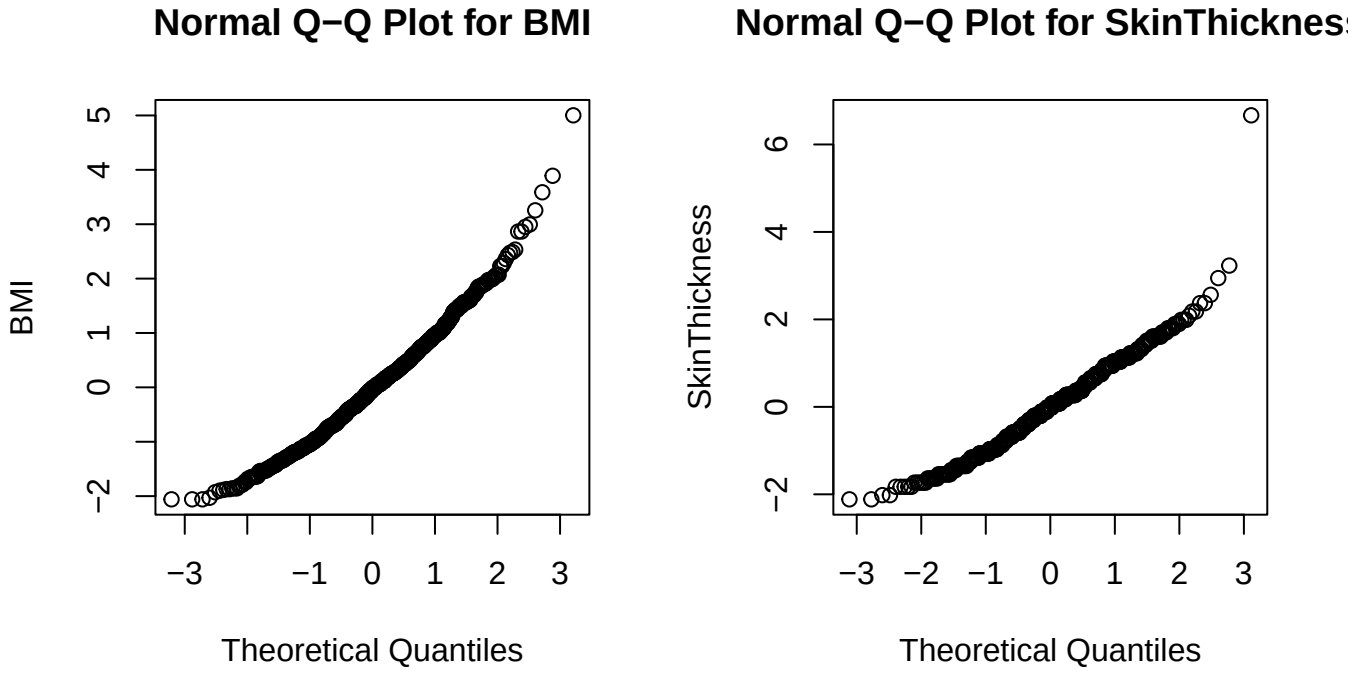


Figure 4: Density Plot Visualization

Q-Q Plot



The BMI and SkinThickness distribution of both non-diabetic and diabetic people are roughly close to normal distribution. According to the qqplot, we can see that the residuals are normally distributed. Thus, we can say that people with high BMI and more SkinThickness are more likely to be diabetic.

Box Plots

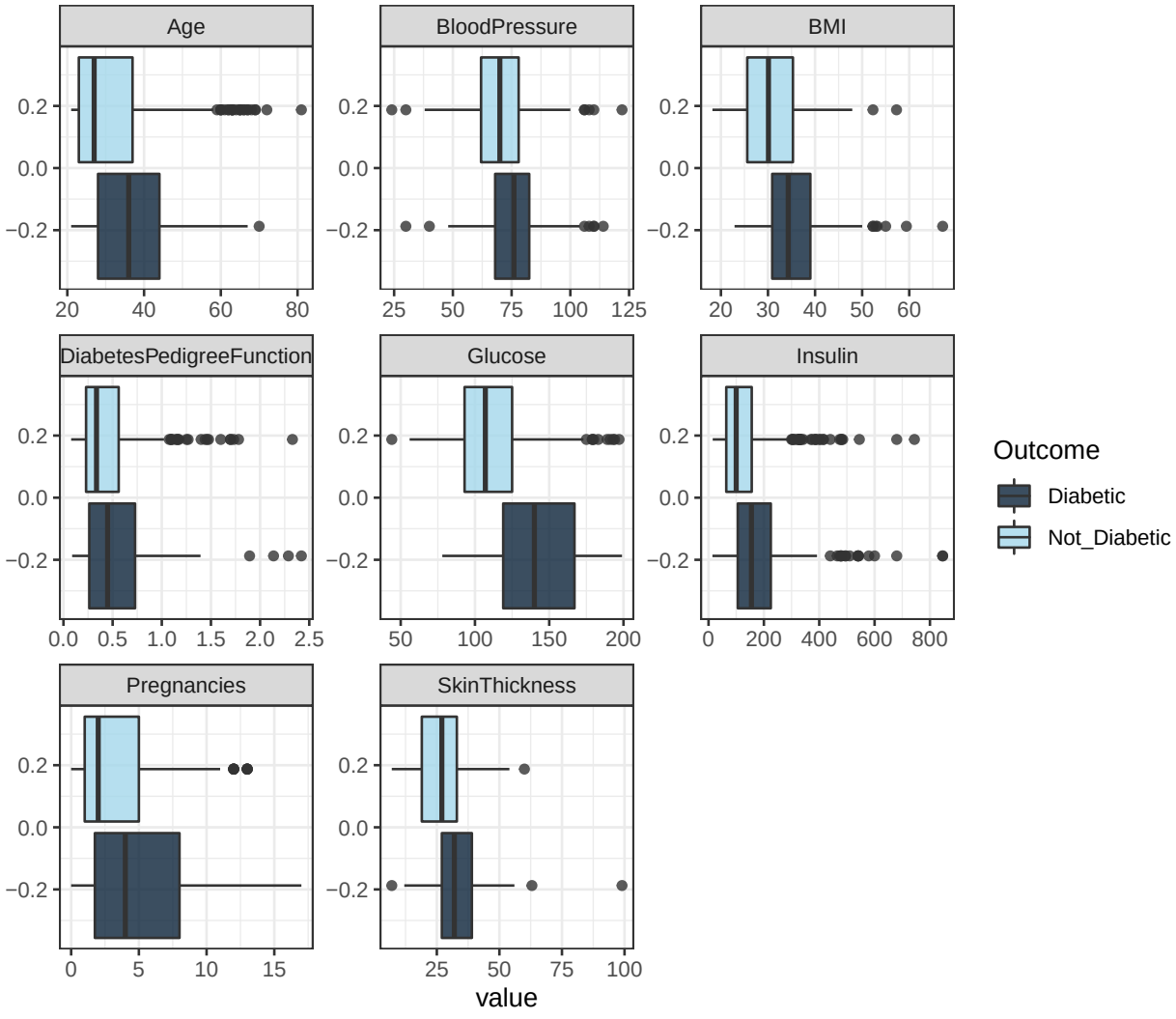


Figure 5: Box Plot Visualization

Box Plots play an important role in statistical analysis. The middle value is the median and the box represents the 25th and 75th percentile of the data. The dots are the outlier values and the whiskers give an idea about the data distribution. We can see that there are a few outliers in each plot. Box plots summarize the distribution of each attribute relating to the class variable.

From the figure, we can say that without controlling for age, women with more number of pregnancies maybe at a risk of contracting diabetes. Similarly, glucose concentration among diabetic women is seen to be more than non diabetic women. The median glucose value for a diabetic patient is around 140mg/dl. The age box plot for non diabetic women has some outliers but we see that age can be considered as an important factor in determining diabetes.

Correlation Plot

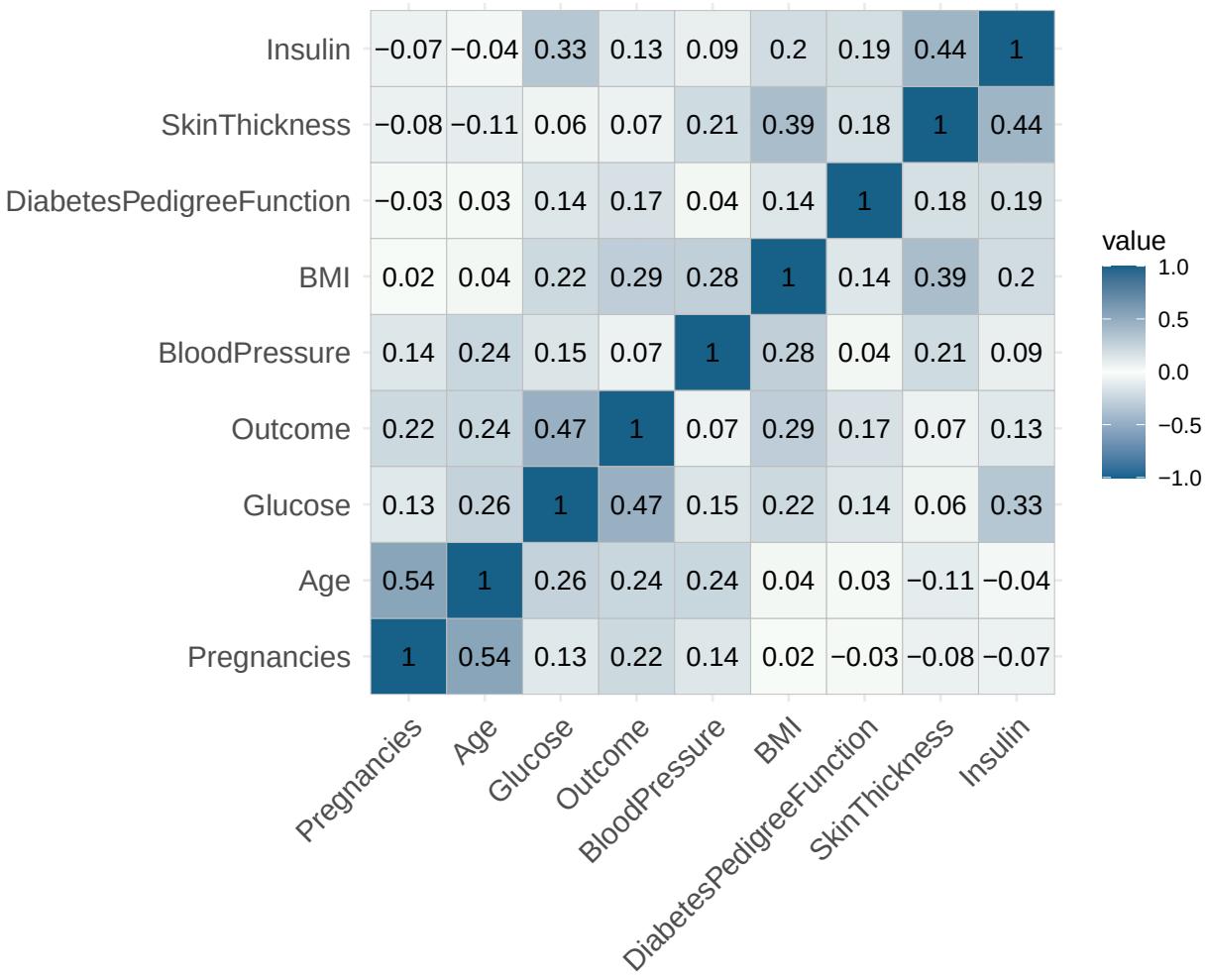


Figure 6: Correlation Plot for all Variables

According to the correlation plot, we can also see that the associations between target variable Outcome and independent variables, such as Insulin, BMI and Age are relatively high.

For the correlation between the predictor variables, we see that Skin Thickness - BMI, Glucose - Insulin, and Pregnancies - Age have a moderate linear correlation. Diabetes Pedigree Function appears to have little correlation with other predictor variables.

BIVARIATE ANALYSIS

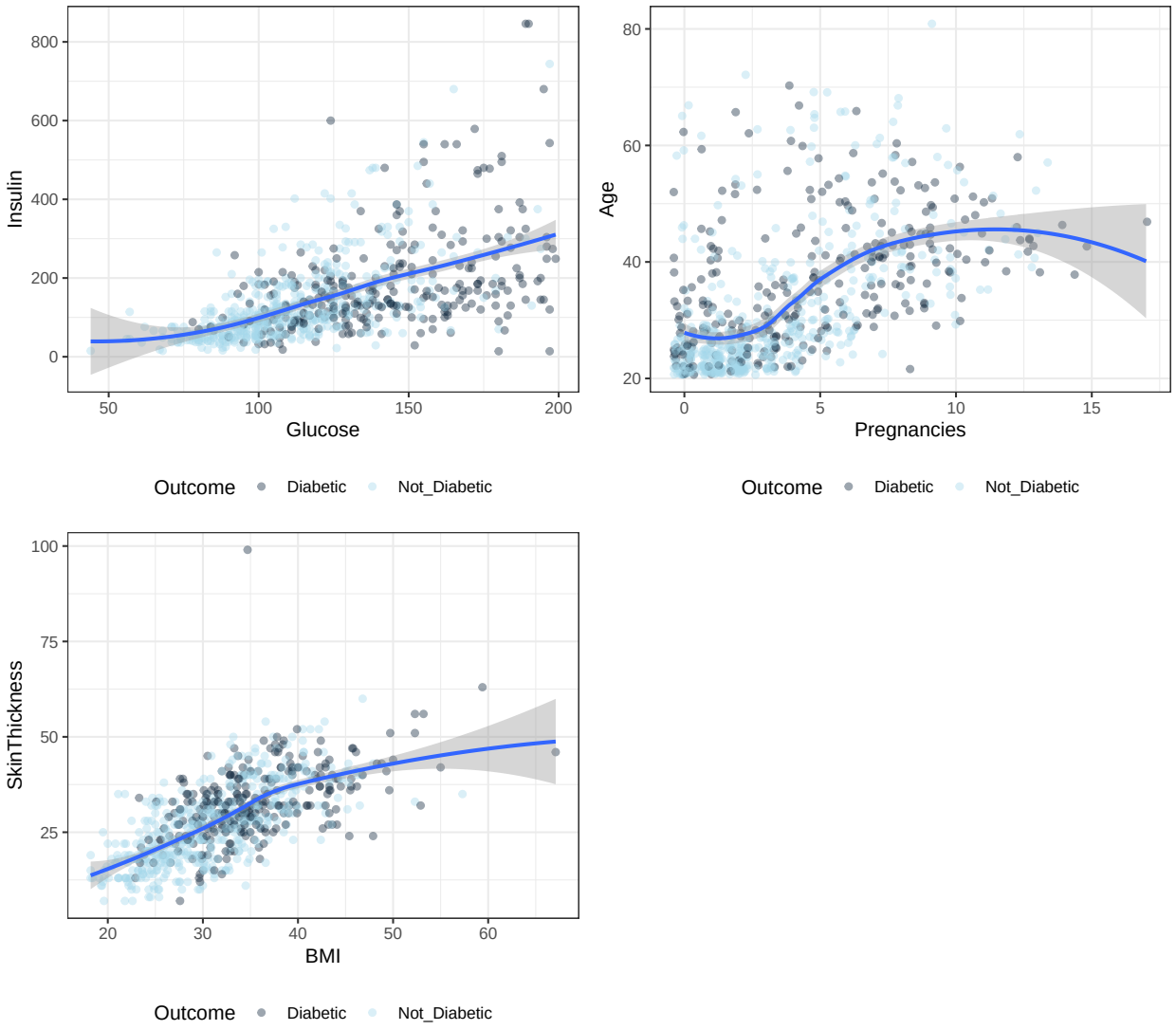


Figure 7: Using Scatterplot to study Bivariate Relationship

From scatter plots, BMI & Skin Thickness, Pregnancies & Age, and Glucose & Insulin seem to have positive linear relationships. It is evident from the above plots that as the glucose and insulin increase, there is a higher risk of the patient being diabetic. As the number of pregnancies and age in women increases, they are more prone to diabetes. Similarly, BMI over 30kg/m^2 with Skin Thickness above 20mm are seen as important factors in Diabetic patients.

Summary of the data visualizations

Overall, we can say that from the above visualizations Age, more number of pregnancies, increased BMI, and glucose levels along skin thickness play an important role in detecting Diabetes in patients. However, there might be some hidden relationships that will be uncovered upon further testing. Therefore, we will take all the predictor variables while classifying the dataset in order to correctly predict whether the patient has diabetes or not.

ALGORITHM TESTING

Currently we have implemented three candidate algorithms: Logistic Regression, Support Vector Machine(SVM), and K-Nearest Neighbors. These statistical learning algorithms will be examined based on the performance metrics.

Splitting the data into train and test sets.

We split our data into test and train data frames and the splitting criteria is 70%-30%.

```
#store rows for partition
partition <- caret::createDataPartition(y = mice_imputed$Outcome, times = 1, p = 0.7, list = FALSE)

# create training data set
train_set <- mice_imputed[partition,]

# create testing data set, subtracting the rows partition to get remaining 30% of the data
test_set <- mice_imputed[-partition,]
```

Cross-Validation

Instead of a single train-test dataset, we use cross validation to train our models. We use this method to reduce the variance introduced due to train-test splits. Cross validation is used to split the dataset into k-parts. Each split is called a fold. Typically, the value of k = 10. The algorithm is trained on a set of k folds and this is repeated multiple times so that every fold is given a chance to be held back as a part of the test dataset. After this execution, we have k different scores and we can summarize the average performance.

Algorithm 01: Logistic Regression

The first candidate algorithm we study is Logistic Regression. Logistic regression is used to predict the class (or category) of individuals based on one or multiple predictor variables (x). It is used to model a binary outcome, that is a variable, which can have only two possible values: 0 or 1, yes or no. Logistic regression belongs to a family, named Generalized Linear Model (GLM), developed for extending the linear regression model [10] [11].We use Logistic Regression for binary classification below. We make use of the CARET package to train our dataset with the GLM model. The logit model and confusion matrix are as follows:

```
## Generalized Linear Model
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## Pre-processing: centered (8), scaled (8), principal component signal
##  extraction (8)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 485, 484, 484, 484, 484, 484, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8212548  0.5476608  0.8697143
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           42           10
##   Not_Diabetic       38          140
##
##                Accuracy : 0.7913
##                  95% CI : (0.733, 0.8419)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 2.878e-06
##
##                   Kappa : 0.4991
##
##  Mcnemar's Test P-Value : 9.735e-05
##
##             Sensitivity : 0.5250
##             Specificity : 0.9333
##          Pos Pred Value : 0.8077
##          Neg Pred Value : 0.7865
##              Prevalence : 0.3478
##          Detection Rate : 0.1826
##    Detection Prevalence : 0.2261
##       Balanced Accuracy : 0.7292
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy in this model is 79.1304348 %.

Algorithm 02: Support Vector Machine(SVM)

Support Vector Machine algorithm is based on the idea of identifying a hyperplane that enables to classify the data points. A plane with a maximum margin is chosen in order to maximize the certainty with which data points are classified into respective classes.[12] We use SVM for binary classification below. We make use of the CARET package to train our dataset with the SVM model. Different kernel techniques can be used by modifying the method of the function. Below, we use the 'linearSVM' method with cost = 1.

```
## Support Vector Machines with Linear Kernel
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 484, 485, 484, 485, 484, 484, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.8222774  0.5326316  0.8805714
##
## Tuning parameter 'C' was held constant at a value of 1
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           39            8
##   Not_Diabetic       41          142
##
##                Accuracy : 0.787
##                  95% CI : (0.7283, 0.838)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 5.840e-06
##
##                   Kappa : 0.4804
##
##  Mcnemar's Test P-Value : 4.844e-06
##
##             Sensitivity : 0.4875
##             Specificity : 0.9467
##          Pos Pred Value : 0.8298
##          Neg Pred Value : 0.7760
##              Prevalence : 0.3478
##          Detection Rate : 0.1696
##    Detection Prevalence : 0.2043
##       Balanced Accuracy : 0.7171
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy in this model is 78.6956522 %.

Algorithm 03: K-Nearest Neighbor (KNN)

KNN is a lazy and simple algorithm that is good at producing results with small dataset. This algorithm considers that similar items exist in the same space or near each other. The K value plays a big role in calculating accuracy as it defines the number of data points that can be selected from that space[14]. In order to do the KNN analysis library "e1071" is selected which can help in calculating the optimal value of k.

```
## k-Nearest Neighbors
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 484, 485, 484, 484, 485, 484, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.7970794  0.5842105  0.8308571
##    7  0.8076809  0.5861404  0.8440000
##    9  0.8131236  0.5619298  0.8411429
```

```
##   11  0.8158488  0.5481871  0.8462857
##   13  0.8171972  0.5400585  0.8605714
##   15  0.8202665  0.5281871  0.8668571
##   17  0.8207485  0.5314035  0.8662857
##   19  0.8190125  0.5388889  0.8657143
##   21  0.8188212  0.5450877  0.8691429
##   23  0.8189632  0.5429240  0.8668571
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

In order to calculate the value of k, we have used 10 fold repeated cross-validation with repeats = 5 and calculated the ROC value where k at which the ROC is maximum is considered to be the optimal k value. After calculating k the accuracy is calculated with the given train data and test data and further a confusion matrix is drawn.

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           43           15
##   Not_Diabetic       37          135
##
##                Accuracy : 0.7739
##                  95% CI : (0.7143, 0.8263)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 4.208e-05
##
##                   Kappa : 0.4675
##
##  Mcnemar's Test P-Value : 0.003589
##
##             Sensitivity : 0.5375
##             Specificity : 0.9000
##          Pos Pred Value : 0.7414
##          Neg Pred Value : 0.7849
##              Prevalence : 0.3478
##          Detection Rate : 0.1870
##    Detection Prevalence : 0.2522
##       Balanced Accuracy : 0.7188
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy in this model is 77.3913043 %.

Algorithm 04: Multivariate Adaptive Regression Splines (MARS)

Linear models can be extended to capture non-linear relations. Multivariate Adaptive Regression Splines (MARS)(Friedman 1991) makes this possible using polynomial regression or step functions.Since most real-world data is non-linear in nature, this algorithm allows us to capture the non-linear relationship by assessing the non-linear functions. Thus, Multivariate adaptive regression splines(MARS) operates by assessing the cutpoints or knots that are similar to step functions. This process evaluates each data point for every predictor as a knot and creates a linear regression model with the candidate features.[15] For this model, we will be using the 'earth' library.

```
## Multivariate Adaptive Regression Spline
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 484, 484, 484, 484, 484, 484, ...
## Resampling results across tuning parameters:
##
##   nprune  ROC        Sens       Spec
##    2      0.7948480  0.5043860  0.8760000
##    3      0.7949916  0.5159649  0.8588571
##    4      0.8218972  0.5646784  0.8702857
##    6      0.8321554  0.5907602  0.8662857
##    7      0.8377861  0.5875439  0.8668571
##    8      0.8344595  0.5865497  0.8617143
##   10      0.8324327  0.5908187  0.8600000
##   11      0.8325915  0.5908187  0.8600000
##   12      0.8325915  0.5908187  0.8600000
##   14      0.8325915  0.5908187  0.8600000
##
## Tuning parameter 'degree' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were nprune = 7 and degree = 1.

## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           39           13
##   Not_Diabetic       41          137
##
##                Accuracy : 0.7652
##                  95% CI : (0.705, 0.8184)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 0.0001394
##
##                   Kappa : 0.4365
##
##  Mcnemar's Test P-Value : 0.0002386
##
##             Sensitivity : 0.4875
```

```
##             Specificity : 0.9133
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7697
##              Prevalence : 0.3478
##          Detection Rate : 0.1696
##    Detection Prevalence : 0.2261
##       Balanced Accuracy : 0.7004
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy for this model is 76.5217391 %.

Algorithm 05: Core Algorithm - Random Forest Algorithm

Random forest algorithm consists of a large number of individual decision trees which function as an ensemble. Every decision tree is made from randomly selecting data from the original data. This is done by randomly sampling a feature set. At the end, class prediction is made by aggregating the predictions from individual trees and a majority vote is calculated to determine the model's class[16]. Thus, the correlation between trees is reduced and this provides higher efficiency. We use the ranger method below which is a fast implementation of random forests [17].

```
## Random Forest
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 538, 538, 538, 538, 538, 538, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##   2     gini        0.8164715  0.5903483  0.8491611
##   2     extratrees  0.8282145  0.5681731  0.8688446
##   5     gini        0.8128033  0.5910542  0.8423914
##   5     extratrees  0.8261041  0.5936458  0.8511356
##   8     gini        0.8069405  0.5829030  0.8374397
##   8     extratrees  0.8247080  0.6112165  0.8463854
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = extratrees
##  and min.node.size = 1.
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           44           14
##   Not_Diabetic       36          136
```

```
##
##                Accuracy : 0.7826
##                  95% CI : (0.7236, 0.8341)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 1.156e-05
##
##                   Kappa : 0.488
##
##  Mcnemar's Test P-Value : 0.002979
##
##             Sensitivity : 0.5500
##             Specificity : 0.9067
##          Pos Pred Value : 0.7586
##          Neg Pred Value : 0.7907
##              Prevalence : 0.3478
##          Detection Rate : 0.1913
##    Detection Prevalence : 0.2522
##       Balanced Accuracy : 0.7283
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy in this model is 78.2608696 %.

Random Forest Algorithm (Fine Tuned)

Now, we try to improve the Random Forest algorithm in a way that it boosts the performance for our dataset.

Tuning algorithm is essential since it helps in building the model. In random forest, we cannot understand our results prior because the model is randomly processing, the error is minimized and an optimal tree is saved from the collection for the model.Tuning algorithm helps control the training process and get better accuracy results.

```
## Random Forest
##
## 538 samples
##   8 predictor
##   2 classes: 'Diabetic', 'Not_Diabetic'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 485, 484, 484, 484, 484, 484, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec
##    1    0.8223918  0.5546199  0.8628571
##    2    0.8260618  0.5905263  0.8571429
##    3    0.8251930  0.5883626  0.8514286
##    4    0.8249657  0.5958480  0.8520000
##    5    0.8225948  0.5990643  0.8508571
##    6    0.8239248  0.6012281  0.8502857
##    7    0.8225806  0.5980117  0.8451429
```

```
##     8    0.8227911  0.6000585  0.8440000
##     9    0.8221871  0.5970760  0.8440000
##    10    0.8216257  0.6014035  0.8440000
##    11    0.8229833  0.5947368  0.8440000
##    12    0.8219975  0.6021637  0.8468571
##    13    0.8216391  0.5958480  0.8480000
##    14    0.8227711  0.5971345  0.8491429
##    15    0.8227811  0.6023977  0.8428571
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

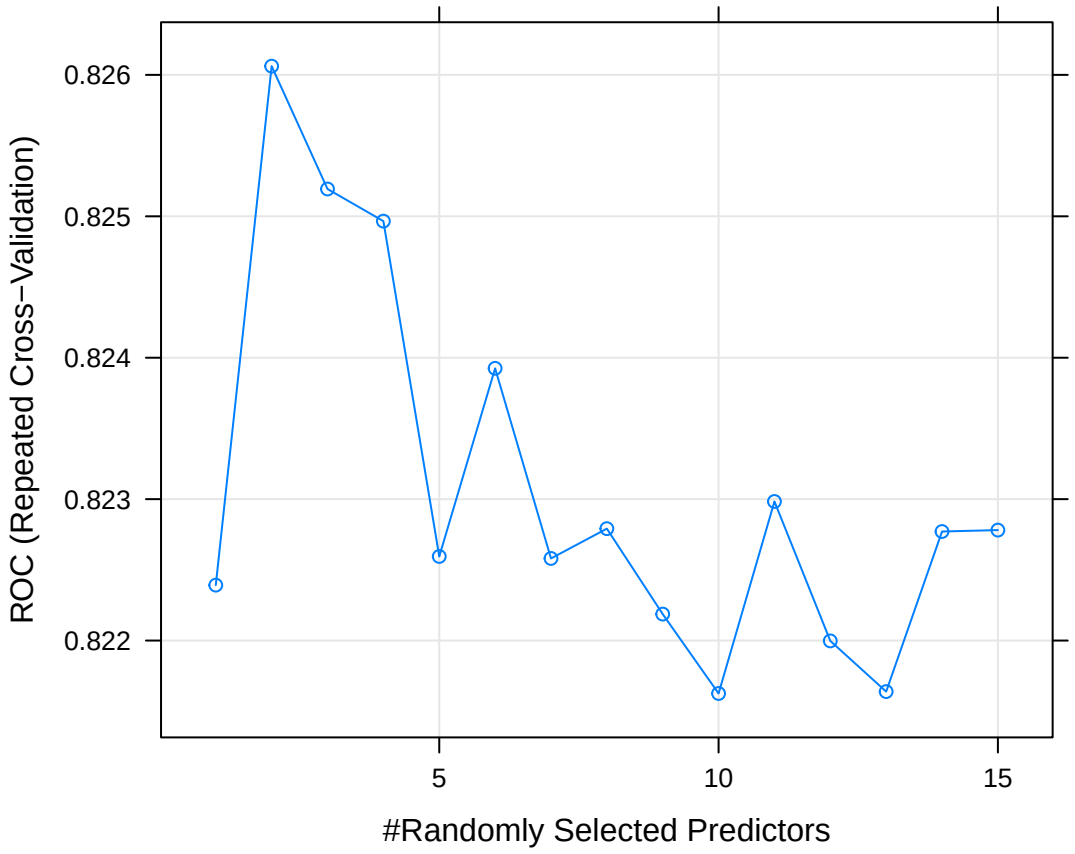


Figure 8: Random forest model using tuning parameter mtry

There are two parameters of the random forest model that are most likely to have the biggest effect on the accuracy of the model. Perhaps they are mtry and ntree. Either of them can be used to tune the function or model. mtry: Number of variables randomly collected to be sampled at every split. ntree: Number of branches to grow after every split time.

Grid search is an effective approach to hyperparameter tuning that will build and evaluate a model for each combination of algorithm parameters specified in a grid [18].In this we are only tuning on one parameter mtry, the grid search only has one dimension as vector.

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Diabetic Not_Diabetic
##   Diabetic           48           12
##   Not_Diabetic       32          138
##
##                Accuracy : 0.8087
##                  95% CI : (0.7518, 0.8574)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 1.312e-07
##
##                   Kappa : 0.5522
##
##  Mcnemar's Test P-Value : 0.004179
##
##             Sensitivity : 0.6000
##             Specificity : 0.9200
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.8118
##              Prevalence : 0.3478
##          Detection Rate : 0.2087
##    Detection Prevalence : 0.2609
##       Balanced Accuracy : 0.7600
##
##        'Positive' Class : Diabetic
##
```

The resulting accuracy in this model is 80.8695652 %.

This results recommends us the best optimal mtry value with highest roc value. Hence, we can see that the use of grid search is effectively seen and we successfully increased the accuracy for random forest algorithm. This is best tuned model that we receive from these methods.

Algorithms Comparison

Table 1: Performance Comparison of Algorithms

Algorithms	Accuracy
Logistic Regression(GLM)	79.1304348%
Support Vector Machine(SVM)	78.6956522%
K-Nearest Neighbours(KNN)	77.3913043%
MARS	76.5217391%
Random Forest	78.2608696%
Random Forest (Fine Tuned)	80.8695652%

Performance Metrics

We compare the models based on the accuracy and performance measures of Sensitivity, Specificity and AUC.

```
##                     Sensitivity/Recall Specificity        F1 Precision       AUC
## LogisticRegression              0.5250   0.9333333 0.6363636 0.8076923 0.8538333
## SVM                             0.4875   0.9466667 0.6141732 0.8297872 0.8598333
## KNN                             0.5375   0.9000000 0.6231884 0.7413793 0.8110000
## MARS                            0.4875   0.9133333 0.5909091 0.7500000 0.8323333
## RF                              0.5500   0.9066667 0.6376812 0.7586207 0.8537917
## RF_Tuned                        0.6000   0.9200000 0.6857143 0.8000000 0.8425833
```

Box Plot Comparison

We are going to compare the models over the training and resampling data with repeated cross validation. We have tried to compare using two metrics, Accuracy and Kappa value.

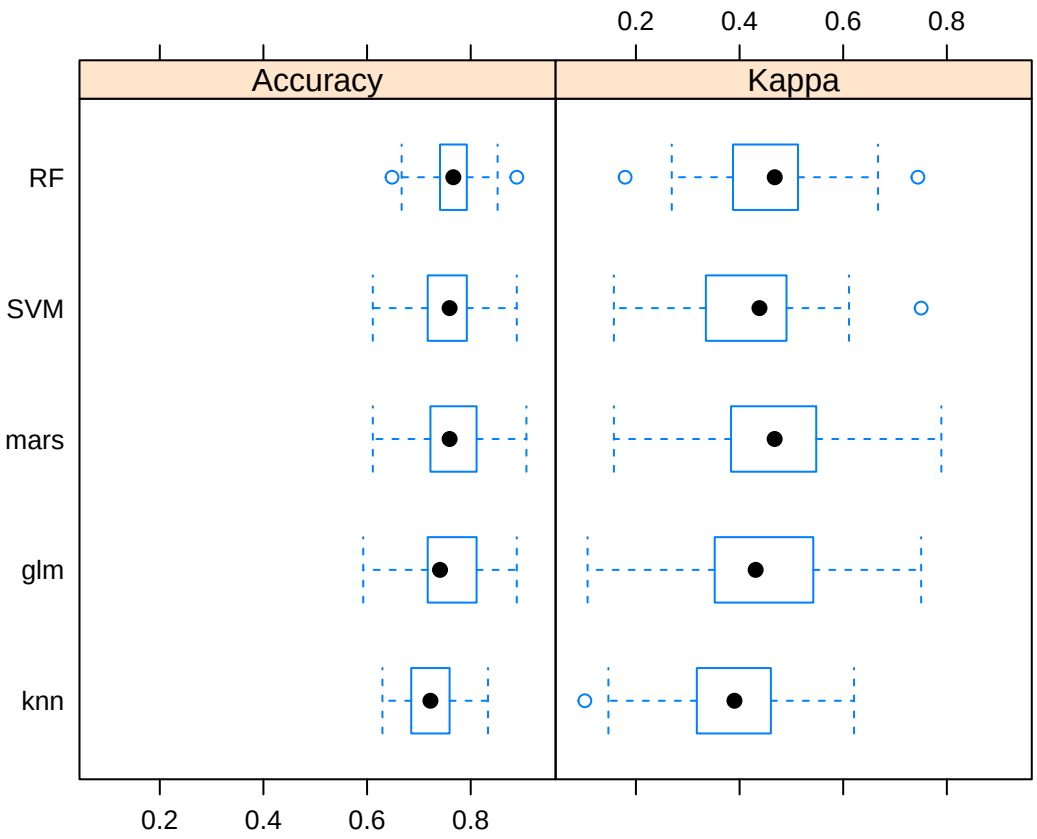


Figure 9: Comparison of Accuracies

Further, we also modify the comparisons according to ROC metrics.

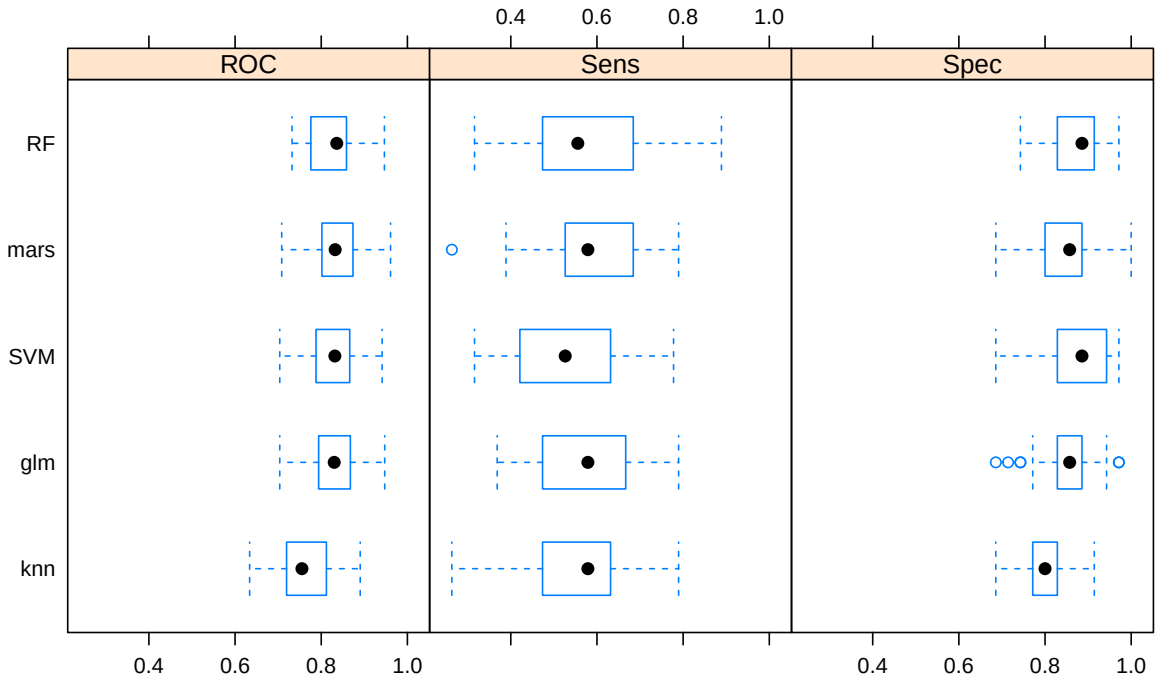


Figure 10: Comparison of ROC metrics

Summary

The above model comparison uses accuracy, ROC values, sensitivity, specificity, recall, precision and F1-score. For a clear visualization and to understand the distribution of each metric, we have illustrated box and whisker plots above. Random Forest algorithm when fine tuned with the help of grid search gives us the highest accuracy of 80.8695652 % as compared to the other models.However, model evaluation is based on several other performance metrics.

We can see from the table above that the Random Forest algorithm gives a higher sensitivity (recall) of 60% which provides a useful insight into identifying true positive rates. This means that the chances of providing correct information on a person having diabetes is higher. Another important metric called F-1 score which is the harmonic mean of the precision and recall [19] is also the highest 68.5714286% among all other models.

Conclusion

Thus, in this project, we have analyzed the Pima Indian Diabetes data set by using various visualizations to extrapolate the given information. We have implemented and compared the performance of five different algorithms namely Logistic Regression, Support Vector Machines, K-Nearest Neighbors, Multivariate Adaptive Regression Splines, and Random forest algorithm (core algorithm) which have further helped us in classifying whether the patient has diabetes or not.

The dataset consists of 768 records comprising the physical conditions of women who are Indian, native Americans in Arizona. We have considered all 9 attributes for our data analysis and classification. We have used the MICE package as a multiple imputation method to replace the missing data values that helps in resolving the zero values and also prevents bias in the data set. We split the data into 70% training data and 30% testing data respectively. Moreover, we considered repeated cross validation to reduce the variance introduced due to the train-test splits.

Further, we trained the above mentioned algorithms using the training data using pre-processing techniques and relevant methods using the CARET package. We then evaluated their performance on the testing data and evaluated the performance matrix for each algorithm. Fine tuning was performed on the core algorithm(Random forest) with the help of grid search that assists in increasing the overall accuracy of the model. The highest accuracy of 80.8695652 % was achieved for Random Forest algorithm among the other models.

At the end, we conclude that the fine tuned Random Forest algorithm gave us the best accuracy which may further help us to identify diabetic patients correctly.

Future Work

In the future to be able to achieve better results, we can utilize additional predictor variables of diabetes and a larger dataset to obtain better results for predicting diabetes among patients. We can also consider a detailed analysis of various performace metrics to evaluate the models.

References

[1] A. Agarwal and A. Saxena, "Analysis of Machine Learning Algorithms and Obtaining Highest Accuracy for Prediction of Diabetes in Women," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), 2019, pp. 686-690.

[2] Gowda Karegowda Ashs, A.S Manjunath and M.S. Jayaram, "Application of Genetic Algorithm Optimized Neural Network Connection Weights For Medical Diagnosis of Pima Indians Diabetes", International Journal on Soft Computing (IJSC), vol. 2, no. 2, May 2011.

[3] Utkin, L. V., Chekh, A. I., & Zhuk, Y. A. (2016). "Binary classification SVM-based algorithms with interval-valued training data using triangular and Epanechnikov kernels. Neural Networks, 80, 53–66. doi: 10.1016/j.neunet.2016.04.005

[4] Pearson, R.. "The problem of disguised missing data." SIGKDD Explor. 8 (2006): 83-92 https://dl.acm.org/doi/10.1145/1147234.1147247.

[5] Ramezani, Rohollah & Maadi, Mansoureh & Khatami, Seyedeh. (2017). A novel hybrid intelligent system with missing value imputation for diabetes diagnosis. Alexandria Engineering Journal. 2018, Pages 1883-1891, https://doi.org/10.1016/j.aej.2017.03.043.

[6] R. Katarya and S. Jain, "Comparison of Different Machine Learning Models for diabetes detection," 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE), 2020, pp. 1-5, doi: 10.1109/ICADEE51157.2020.9368899.

[7] R. Sehly and M. Mezher, "Comparative Analysis of Classification Models for Pima Dataset," 2020 International Conference on Computing and Information Technology (ICCIT-1441), 2020, pp. 1-5, doi: 10.1109/ICCIT-144147971.2020.9213821.

[8] Heitjan DF, Little RJA. Multiple imputation for the fatal accident reporting system. J R Stat Soc Series C (Appl Stat) 1991;40(1):13–29.

[9] Schenker N, Taylor JMG. Partially parametric techniques for multiple imputation. Comput Stat & Data Anal. 1996;22(4):425–446. doi: 10.1016/0167-9473(95)00057-7.

[10] Bruce, Peter, and Andrew Bruce. 2017. Practical Statistics for Data Scientists. O'Reilly Media.

[11] James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014.An Introduction to Statistical Learning: With Applications in R. Springer Publishing Company, Incorporated.

[12] Simon Tong, Daphne Koller Support Vector Machine Active Learning with applications to text classification. doi : 10.1162/153244302760185243

[13] S. C. Gupta and N. Goel, "Performance enhancement of diabetes prediction by finding optimum K for KNN classifier with feature selection method," 2020 Third

[14] Friedman, Jerome H. 1991. "Multivariate Adaptive Regression Splines." The Annals of Statistics. JSTOR, 1–67.

[15] http://uc-r.github.io/mars

[16] https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees

[17] https://www.rdocumentation.org/packages/ranger/versions/0.13.1/topics/ranger

[18] https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/

[19] https://deepai.org/machine-learning-glossary-and-terms/f-score

[20] A. Al-Zebari and A. Sengur, "Performance Comparison of Machine Learning Techniques on Diabetes Disease Detection," 2019 1st International Informatics and Software Engineering Conference (UBMYK), 2019, pp. 1-4, doi: 10.1109/UBMYK48245.2019.8965542.

[21] R. Sehly and M. Mezher, "Comparative Analysis of Classification Models for Pima Dataset," 2020 International Conference on Computing and Information Technology (ICCIT-1441), 2020, pp. 1-5, doi: 10.1109/ICCIT-144147971.2020.9213821.

[22] https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

Appendix–Code

```
knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE, fig.pos = 'H')
data <- read.csv("/Users/hilonim/Downloads/diabetes.csv") # reading data set
rows <- nrow(data)
cols <- ncol(data)
library(ggplot2) #data visualization
library(corrplot) #Correlation plot
library(ggcorrplot) #Correlation plot
library(mice) #multiple imputation
library(VIM) #Visualization and Imputation of Missing Values
library(tidyverse) #data analysis
library(ggfortify) #Data Visualization tool
library(datasets)  #pre-loaded dataset
library(GGally) #graphics plot - extension to ggplot2
library(caret) # Classification And Regression Training
library(gbm) # Generalized Boosted Regression Modeling
library(pROC) #ROC Curve
library(datasets)

f_data <- read.csv("/Users/hilonim/Downloads/diabetes.csv") # raw data set
summary(data)

data$Outcome <- ifelse(data$Outcome == 0, "Not_Diabetic", "Diabetic")
data$Outcome <- as.factor(data$Outcome)


# Setting up the theme for the visualizations
palette <- c('#0A2239' ,'#a4d8eb' ,'#1D84B5' ,'#132E32' ,'#176087')

ggplot <- function(...) ggplot2::ggplot(...) +
  scale_color_manual(values = palette) +
scale_fill_manual(values = palette) +
  theme_bw()

# Visualization of the Class Variable
ggplot(data, aes(Outcome, fill = Outcome)) +
  geom_bar() + geom_text(aes(label = ..count..),
  stat = "count", vjust =2.0,
colour = "white") + theme_bw() + labs( x = "Diabetes")

rows <- nrow(data)
cols <- ncol(data)

db <- nrow(f_data[f_data$Outcome==1 , ]) #Diabetic patients
db_percent <- db/nrow(f_data) * 100 #Percentage of Diabetic Patients

Non_db <- nrow(f_data[f_data$Outcome==0 , ]) # Non-Diabetic patients
db1_percent <- Non_db/nrow(f_data) * 100 #% of Non-Diabetic Patients
# Checking for NA values in the dataset
print(all(!is.na(data)))

# Replacing the Zero Values to NA except for Pregnancies and Outcome
```

```
columns <- colnames(data)[!colnames(data) %in% c("Pregnancies", "Outcome")]
Empty_data <- data[columns] == 0
data[columns][Empty_data] <- NA
# Printing the number of missing values in the dataset
Missing_values <- apply(Empty_data, 2, sum)
Missing_values

library(VIM)
aggr_plot <- aggr(data, col=c('LIGHTBLUE', 'LIGHTBLUE4'),
             numbers=TRUE, labels=names(data), cex.axis=.6,
             gap=2, ylab=c("Histogram of missing data","Pattern"))
set.seed(123)
# Multiple imputation Using MICE
mice_data <- mice(data, m = 5, method='pmm')


# Save the complete imputation output
mice_imputed <- complete(mice_data)

# Make sure there is no missing data
sum(is.na(mice_imputed))

library(ggplot2)
library(gridExtra)

p1 <- ggplot(f_data, aes(x = SkinThickness)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "SkinThickness (original)", y = "Density")

p2 <- ggplot(mice_imputed, aes(x = SkinThickness)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "SkinThickness(imputed)", y = "Density")

p3 <- ggplot(f_data, aes(x = Insulin)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "Insulin (original)", y = "Density")

p4 <- ggplot(mice_imputed, aes(x = Insulin)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "Insulin(imputed)", y = "Density")

p5 <- ggplot(f_data, aes(x = BMI)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "BMI (original)", y = "Density")

p6 <- ggplot(mice_imputed, aes(x = BMI)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
```

```
      labs(x = "BMI(imputed)", y = "Density")

p7 <- ggplot(f_data, aes(x = BloodPressure)) +
     geom_density(alpha = 0.5) +
     theme(legend.position = "bottom") +
     labs(x = "BloodPressure (original)", y = "Density")

p8 <- ggplot(mice_imputed, aes(x = BloodPressure)) +
    geom_density(alpha = 0.5) +
    theme(legend.position = "bottom") +
    labs(x = "BloodPressure (imputed)", y = "Density")

p9 <- ggplot(f_data, aes(x = Glucose)) +
      geom_density(alpha = 0.5) +
      theme(legend.position = "bottom") +
      labs(x = "Glucose (original)", y = "Density")

p10 <- ggplot(mice_imputed, aes(x = Glucose)) +
       geom_density(alpha = 0.5) +
       theme(legend.position = "bottom") +
       labs(x = "Glucose (imputed)", y = "Density")


grid.arrange(p1, p2, p3, p4,p5, p6,p7,p8,p9,p10, nrow =5, ncol=2)
mice_imputed %>%
  gather("key", "value", Pregnancies:Age) %>%
  ggplot(aes(x = value, fill = Outcome)) +
    facet_wrap(vars(key), ncol = 3, scales = "free") +
    geom_density(size = 1, alpha = 0.8)

library(nortest) #Tests for normality
qqnorm(scale(data$BMI), main = "Normal Q-Q Plot for BMI",
       xlab = "Theoretical Quantiles", ylab = "BMI")
qqnorm(scale(data$SkinThickness), main = "Normal Q-Q Plot for SkinThickness",
       xlab = "Theoretical Quantiles", ylab = "SkinThickness")
mice_imputed %>%
  gather("key", "value", Pregnancies:Age) %>%
  ggplot(aes(x = value, fill = Outcome)) +
    facet_wrap(vars(key), ncol = 3, scales = "free") +
    geom_boxplot(alpha = 0.8)
data_cor <- round(cor(f_data[1:9]),2)
# data_cor
library(ggcorrplot)
ggcorrplot(data_cor, hc.order = TRUE, lab = TRUE) +   scale_fill_gradient2(mid="#FBFEF9",
# BIVARIATE ANALYSIS
 a <- ggplot(data = mice_imputed, mapping = aes(x = Glucose,y=Insulin)) +
      geom_point(mapping = aes(color = Outcome),alpha = 0.4) +
      geom_smooth() +
      theme(legend.position = "bottom")
b <-  ggplot(data = mice_imputed, mapping = aes(x = Pregnancies,y=Age)) +
       geom_jitter(mapping = aes(color = Outcome),alpha = 0.4) +
       geom_smooth() +
       theme(legend.position = "bottom")
```

```r
c <- ggplot(data = mice_imputed, mapping = aes(x = BMI,y=SkinThickness)) +
       geom_point(mapping = aes(color = Outcome),alpha = 0.4) +
        geom_smooth() + theme(legend.position = "bottom")

grid.arrange(a,b,c, nrow = 2, ncol =2)

#store rows for partition
partition <- caret::createDataPartition(y = mice_imputed$Outcome, times = 1, p = 0.7, list = FALSE)

# create training data set
train_set <- mice_imputed[partition,]

# create testing data set, subtracting the rows partition to get remaining 30% of the data
test_set <- mice_imputed[-partition,]

library(cvTools)
#training the model
model_glm <- train(Outcome ~., data = train_set,
                           method = "glm",
                           metric = "ROC",
                           family = binomial(link="logit"),
                           tuneLength = 10,
                           trControl = trainControl(method = "repeatedcv",
                           number = 10, repeats =5,classProbs = T,
                           summaryFunction = twoClassSummary ),
                           preProcess = c("center","scale","pca"))

#printing the model
model_glm

# prediction on Test data set
predict_glm <- predict(model_glm, test_set[,-9])

acc_glm_fit <- confusionMatrix(predict_glm, test_set$Outcome )$overall['Accuracy']
acc_glm <- acc_glm_fit * 100

#Confusion Matrix and Statistic
cm_glm <- confusionMatrix(predict_glm, test_set$Outcome)
cm_glm

# Prediction Probabilities
pred_prob_glm <- predict(model_glm, test_set, type="prob")

# ROC value
roc_glm <- roc(test_set$Outcome, pred_prob_glm$Diabetic)

svm_Linear <- train(Outcome ~., data = train_set,
                      method = "svmLinear",
                      metric = "ROC",
                      tuneLength = 5,
                      trControl = trainControl(method = "repeatedcv", number = 10, classProbs = T,
                                  summaryFunction = twoClassSummary, repeats = 5 ),
                      preProcess = c("center", "scale"))
```

```r
#model
svm_Linear

# prediction on Test data set
predictSVM <- predict(svm_Linear, test_set[,-9])

#Confusion matrix
cm_svm <- confusionMatrix(predictSVM, test_set$Outcome)
cm_svm
#accuracy
acc_svm_fit <- confusionMatrix(predictSVM, test_set$Outcome )$overall['Accuracy']
acc_svm <- acc_svm_fit * 100

# Prediction Probabilities
pred_prob_svm <- predict(svm_Linear, test_set, type="prob")

# ROC value
roc_svm <- roc(test_set$Outcome, pred_prob_svm$Diabetic)

library(class)
library(e1071)

pima.train.lab <- train_set[,9]
pima.test.lab <- test_set[,9]

pima.train.data <- train_set[,1:8]
pima.test.data <- test_set[,1:8]


#train model
knn.pima <- train(Outcome ~ ., data = train_set,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  trControl = trainControl(method = "repeatedcv", number = 10, repeats = 5,
                   classProbs = T, summaryFunction = twoClassSummary),
                    metric = "ROC", tuneLength = 10)

#printing the model
knn.pima

learn_knn <- knn(train = pima.train.data,
                 test = pima.test.data,
                 cl = pima.train.lab,k= 17, prob = TRUE)



# prediction on Test data set
predict_knn <- predict(knn.pima, test_set[,-9])

#confusion matrix
cm_knn <- confusionMatrix(predict_knn, test_set$Outcome)
cm_knn
```

```r
#accuracy of the model
acc_knn_fit <- confusionMatrix(predict_knn, test_set$Outcome )$overall['Accuracy']
acc_knn <- acc_knn_fit * 100

# Prediction Probabilities
pred_prob_knn <- predict(knn.pima, test_set, type="prob")

# ROC value
roc_knn <- roc(test_set$Outcome, pred_prob_knn$Diabetic)
library(earth)
#training the model

model_mars <- train(Outcome ~., data = train_set,
                        method = "earth",
                        metric = "ROC",
                        tuneLength = 10,
                        trControl = trainControl(method = "repeatedcv",
                        number = 10,repeats = 5, classProbs = T,
                        summaryFunction = twoClassSummary ),
                        preProcess = c("center","scale"))


#printing the model
model_mars

# prediction on Test data set
predict_mars <- predict(model_mars, test_set[,-9])


#accuracy of the model
acc_mars_fit <- confusionMatrix(predict_mars, test_set$Outcome)$overall['Accuracy']
acc_mars <- acc_mars_fit * 100

#Confusion Matrix and Statistic
cm_mars <- confusionMatrix(predict_mars, test_set$Outcome)
cm_mars

# Prediction Probabilities
pred_prob_mars <- predict(model_mars, test_set, type="prob")

# ROC value
roc_mars <- roc(test_set$Outcome, pred_prob_mars$Diabetic)


library(randomForest) #random forests


learn_rforest <- train(Outcome ~., data = train_set,
                        method = "ranger",
                        metric = "ROC",
                        trControl = trainControl(classProbs = T,
                        summaryFunction = twoClassSummary ))
learn_rforest
```

```r
# prediction on Test data set
predict_rf <- predict(learn_rforest, test_set)

# Confusion Matrix
cm_rf <- confusionMatrix(predict_rf, test_set$Outcome, positive="Diabetic")

acc_rf_fit <- confusionMatrix(predict_rf, test_set$Outcome )$overall['Accuracy']
acc_rf <- acc_rf_fit * 100


# Prediction Probabilities
pred_prob_rf <- predict(learn_rforest, test_set, type="prob")

# ROC value
roc_rf <- roc(test_set$Outcome, pred_prob_rf$Diabetic)



# Confusion Matrix for Random Forest Model
cm_rf

set.seed(1)

#fine tuning the model
tunegrid <- expand.grid(.mtry=c(1:15))

#train model
rftune <- train(Outcome ~.,data = train_set,
                method = "rf",metric= "ROC", tuneGrid = tunegrid,
                trControl = trainControl(method = "repeatedcv",number = 10, repeats=5,
                             classProbs = T,summaryFunction = twoClassSummary),
                preProcess = c("center","scale"))

#printing the model
rftune

#model visualozation based on tuning parameters
plot(rftune)

# prediction on Test data set
predict_rfTune <- predict(rftune, test_set)


acc_rf_fit <- confusionMatrix(predict_rfTune, test_set$Outcome )$overall['Accuracy']
acc_rfTune <- acc_rf_fit * 100


# Prediction Probabilities
pred_prob_rfTune <- predict(rftune, test_set, type="prob")

# ROC value
roc_rfTune <- roc(test_set$Outcome, pred_prob_rfTune$Diabetic)
```

```r
# Confusion Matrix
cm_rfTune <- confusionMatrix(predict_rfTune, test_set$Outcome, positive="Diabetic")

# Confusion Matrix for Random Forest Model
cm_rfTune

LogisticRegression <-  c( cm_glm$byClass['Sensitivity'],
                     cm_glm$byClass['Specificity'], cm_glm$byClass['F1'],
          cm_glm$byClass['Precision'],roc_glm$auc)

SVM <- c( cm_svm$byClass['Sensitivity'], cm_svm$byClass['Specificity'],
          cm_svm$byClass['F1'],cm_svm$byClass['Precision'],
                                roc_svm$auc)

KNN <- c( cm_knn$byClass['Sensitivity'],cm_knn$byClass['Specificity'],
       cm_knn$byClass['F1'],cm_knn$byClass['Precision'],
       roc_knn$auc)

MARS <- c(cm_mars$byClass['Sensitivity'], cm_mars$byClass['Specificity'],
        cm_mars$byClass['F1'],cm_mars$byClass['Precision'],
        roc_mars$auc)

RF <- c(cm_rf$byClass['Sensitivity'], cm_rf$byClass['Specificity'],
        cm_rf$byClass['F1'], cm_rf$byClass['Precision'],
        roc_rf$auc)

RF_Tuned <- c(cm_rfTune$byClass['Sensitivity'],
            cm_rfTune$byClass['Specificity'],
            cm_rfTune$byClass['F1'],
            cm_rfTune$byClass['Precision'],
            roc_rfTune$auc)

all_results <- data.frame(rbind( LogisticRegression, SVM, KNN, MARS, RF, RF_Tuned))

names(all_results) <- c("Sensitivity/Recall", "Specificity","F1","Precision", "AUC")

all_results
#Accuracy Comparison
control <- trainControl(method="repeatedcv", number=10, repeats=3)

set.seed(7)
modelrf <- train(Outcome~., data=train_set, method="ranger", trControl=control)

set.seed(7)
modelglm <- train(Outcome~., data=train_set, method="glm", trControl=control)

set.seed(7)
modelsvm <- train(Outcome~., data=train_set, method="svmLinear", trControl=control,verbose=FALSE)

set.seed(7)
modelknn <- train(Outcome~., data=train_set, method="knn", trControl=control)

set.seed(7)
```

```
modelmars <- train(Outcome~., data=train_set, method="earth", trControl=control)

results <- resamples(list(RF=modelrf, glm =modelglm, SVM=modelsvm,mars=modelmars, knn=modelknn))

bwplot(results)
#ROC Metrics Comparison
control <- trainControl(method="repeatedcv", number=10, repeats=5,
            classProbs = TRUE,summaryFunction = twoClassSummary)

set.seed(7)
modelrf <- train(Outcome~., data=train_set, method="ranger", trControl=control, metric="ROC")

set.seed(7)
modelglm <- train(Outcome~., data=train_set, method="glm", trControl=control, metric="ROC")

set.seed(7)
modelsvm <- train(Outcome~., data=train_set, method="svmLinear",
            trControl=control,verbose=FALSE, metric="ROC")

set.seed(7)
modelknn <- train(Outcome~., data=train_set, method="knn", trControl=control, metric="ROC")

set.seed(7)
modelmars <- train(Outcome~., data=train_set, method="earth", trControl=control, metric="ROC")

results <- resamples(list(RF=modelrf, glm =modelglm, SVM=modelsvm,mars=modelmars, knn=modelknn))

bwplot(results)
```