PROJECT REPORT ON

"Predictive Modeling and analysis of PIMA Indian Diabetes Database using Python"

From

DEPARTMENT OF

Statistics

Submitted under the category of

Individual

By

Soham Suresh Halbandge

Under the Guidance of

Mrs. Shailaja Rane

Submitted in partial fulfillment of the requirement

For qualifying the

'Jigyaasa'- SHP Certificate Course

June 2020

JIGYAASA- SCIENCE HONORS PROGRAM

K.C. COLLEGE

CHURCHGATE, MUMBAI 400028

KISHINCHAND CHELLARAM COLLEGE

'JIGYAASA'

SCIENCE HONORS PROGRAM

Batch XVI-2019-2021

This is to certify that ***<u>Soham Halbandge</u>*** *have/has completed the project work as a part fulfillment of 'Jigyaasa'-****Science Honors Program*** *of K.C. College in the subject of* ***<u>Statistics.</u>*** *The work was carried out under the category of* ***Individual*** *research and entitled, "****Predictive Modeling and analysis of PIMA Indian Diabetes Database using Python****"*

It is the outcome of the original experiments performed by him/her/them, during the year 2019 – 2021. The data information presented over here is true to the best of our knowledge.

Date: ___________

Teacher In charge	***Coordinator***	***Convener***
(Mrs. Shailaja Rane)	***(Dr. Sheela Valecha)***	***(Dr.Sagarika Damle)***

Statement of the Candidate

As a part of ‘Jigyaasa’- Science Honors Program or as required by ‘Jigyaasa’- Science Honors Program, I wish to state that the work entered in this research project entitled **“Predictive Modeling and analysis of PIMA Indian Diabetes Database using Python”** forms my own contribution to Research work carried out under the guidance of **Mrs. Shailaja Rane**, Assistant Professor, in the Department of Statistics at K.C. College, Mumbai. This work has not been submitted for any other degree.

Certified by

Mrs. Shailaja Rane	Soham Halbandge
(Research Guide)	(Research Student)
Date	Date

Acknowledgment:

I would like to thank our SHP Convener - **Dr. Sagarika Damle** and SHP Coordinator- **Dr. Sheela Valecha** for allowing me to be part of the Science Honors Program (SHP). This research paper would not have been possible without the guidance of my mentor, **Mrs. Shailaja Rane**. I sincerely extend my gratitude to our principal- **Dr. Hemlata K. Bagla**, Vice Principals, **Dr. Shalini Sinha, Prof. Samarjit Padhi**, and the **Management Board Members**. I would like to thank our Head of Department **Dr. Asha Jindal** and other staff members, teaching and non-teaching staff of the Statistics Department for their support and encouragement. I am also thankful to my parents, friends, and well-wishers.

For the dataset - Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

"Predictive Modeling and analysis of PIMA Indian Diabetes Database using Python"

Soham Halbandge, Mrs. Shailaja Rane*

Department of Statistics, K.C. College, Vidyasagar Principal K.M. Kundnani Chowk, D.W. Road, Churchgate, Mumbai 400020, India

*E-mail address of Author for correspondence (logzii123@gmail.com)

Abstract

Diabetes is a chronic disease that occurs when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Considering how chronic this disease is, the number of diabetic patients has been increasing day by day which is becoming a huge concern for society. Understanding how critical this disease is this is the inspiration for the selection of the topic for the research paper.

This study aims to select the best machine learning classification algorithm which accurately predicts whether or not the patients in the dataset have diabetics or not.

The dataset which is being dealt with is secondary data. This secondary dataset is being collected from the website which is an online community of data scientists and machine learning practitioners (Kaggle). Several machine learning classification algorithms were fitted like Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Naïve Bayes Classifier, Decision Tree and Random Forest. Python is the software used for the analysis. This study is based on 768 observations under 9 variables. In this study it has been found out that the average Blood Pressure for a diabetic patient is in between 68.25 to 73.39 mmHg. The average blood pressure of non-diabetic patient is between 66.6 and 69.76 mmHg. The population average glucose level for diabetic patient is estimated to be in between 138 to 145 mg/dL. The population average glucose level for non-diabetic patients is estimated to be in between 108 to 112 mg/dL. The best fitted machine learning algorithms which accurately predicts whether the person is diabetic or not are Logistic Regression and Random Forest.

This paper includes the inferential, univariate, and multivariate statistical analysis and develops the predictive models for the classification of the PIMA Indian Diabetes Database.

Keywords: Inferential, Univariate, Multivariate, Predictive Modeling

Introduction

Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Insulin is a hormone that regulates blood sugar. Hyperglycemia, or raised blood sugar, is a common effect of uncontrolled diabetes and over time leads to serious damage to many of the body's systems, especially the nerves and blood vessels.

WHO estimates the total death due to diabetes will rise to 50% in the next decade? According to WHO and ADA, there are four types of diabetes: Type-I, Type-II diabetes, Gestational Diabetes (GDM) and rare specific diabetes. Type-I diabetes is responsible for 5% to 10% of total diabetes, it occurred due to lack of insulin production. Destruction of the pancreas organ is the source of insulin production loss in human body which leads to insulin-dependent diabetes. Type-II diabetes, however, is more common (90% of the diabetic population) is caused by “insulin resistance” and metabolic disorders, the result is an increase in sugar levels in the blood.

Between 2000 and 2016, there was a 5% increase in premature mortality from diabetes. In high-income countries, the premature mortality rate due to diabetes decreased from 2000 to 2010 but then increased in 2010-2016. In lower-middle-income countries, the premature mortality rate due to diabetes increased across both periods.

In 2014, 8.5% of adults aged 18 years and older had diabetes. In 2019, diabetes was the direct cause of 1.5 million deaths. To present a more accurate picture of the deaths causes by diabetes, however, deaths due to higher-than-optimal blood glucose through cardiovascular disease, chronic kidney disease and tuberculosis should be added. In 2012 (year of the latest available data), there were another 2.2 million deaths due to high blood glucose.

As of 2015, 30.3 million people in the United States, or 9.4 per cent of the population, had diabetes. More than 1 in 4 of them didn’t know they had the disease. Diabetes affects 1 in 4 people over the age of 65. About 90-95 per cent of cases in adults are type 2 diabetes.

By contrast, the probability of dying from any one of the four main non-communicable diseases (cardiovascular diseases, cancer, chronic respiratory diseases or diabetes) between the ages of 30 and 70 decreased by 18% globally between 2000 and 2016.

Objective:

The objective of this study aims to select the best machine learning classification algorithm which accurately predicts whether or not the patients in the dataset have diabetics or not.

Methodology

The methodology of this study goes through following stages: -

i. About the Data
ii. Explanatory Data Analysis
iii. Inferential Statistics
iv. Data preprocessing
   - Missing Values
   - Outliers
   - Standardization
v. Train-Test Split
vi. Working Procedure

i. About the Data

The dataset which is being dealt with is secondary data. This secondary dataset is being collected from the website which is an online community of data scientists and machine learning practitioners (Kaggle - https://www.kaggle.com/). This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset has been uploaded by UCI Machine learning and was updated 5 years ago (as of July 2021). The given dataset has 768 observations under 9 variables naming -

- Pregnancies - Number of times pregnant
- Glucose - Plasma glucose concentration - 2 hours in an oral glucose tolerance test (mg/dL)
- BloodPressure - Diastolic blood pressure (mmHg)
- SkinThickness - Triceps skin fold thickness (mm)
- Insulin - 2-Hour serum insulin (mu U/ml)
- BMI - Body mass index
- DiabetesPedigreeFunction - Diabetes pedigree function

- Age - Age (years)
- Outcome - Class variable (Target variable- 0 and 1)

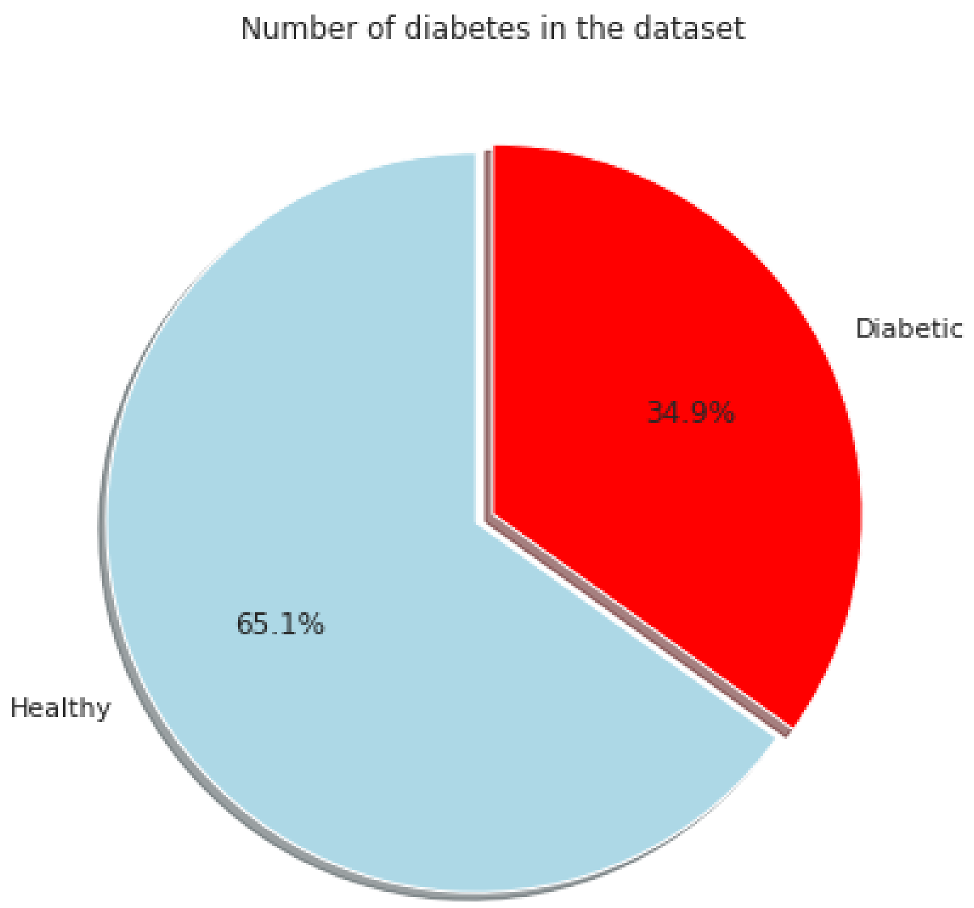
Where 0 stands for non-Diabetic patient and 1 stand for Diabetic Patient.

In this dataset there are 8 dependent variables (Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age) and 1 independent variable (Outcome).

The classification algorithm will be **binary** classification algorithm (i.e., whether person is diabetic patient or not)

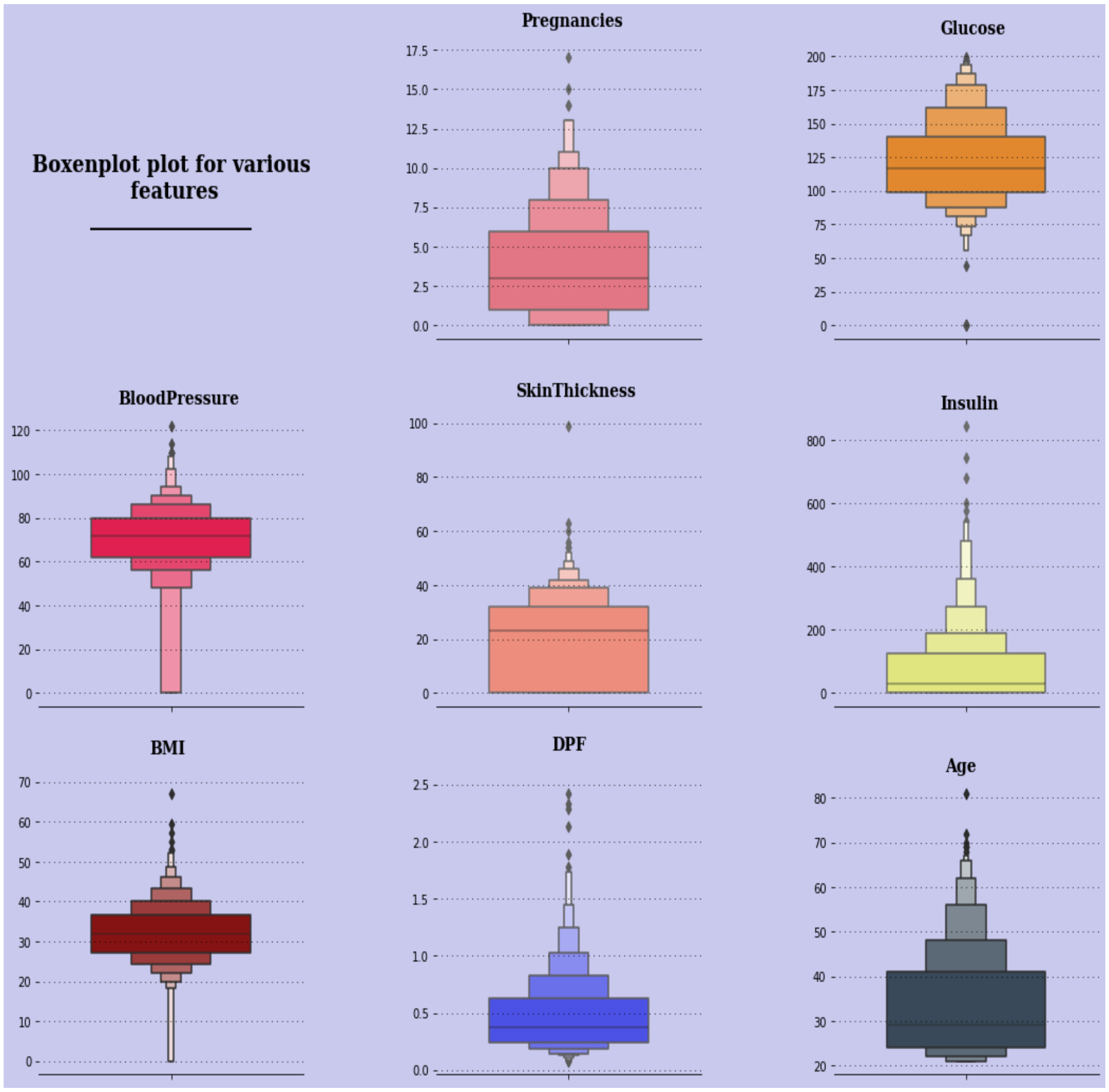
ii. <u>**EDA (Explanatory Data Analysis)**</u>

Explanatory Data Analysis is approach about how data analysis should be carried out. It gives proper insights about the data and it helps to understand the dataset very well. This research carries out both univariate and multivariate data analysis.



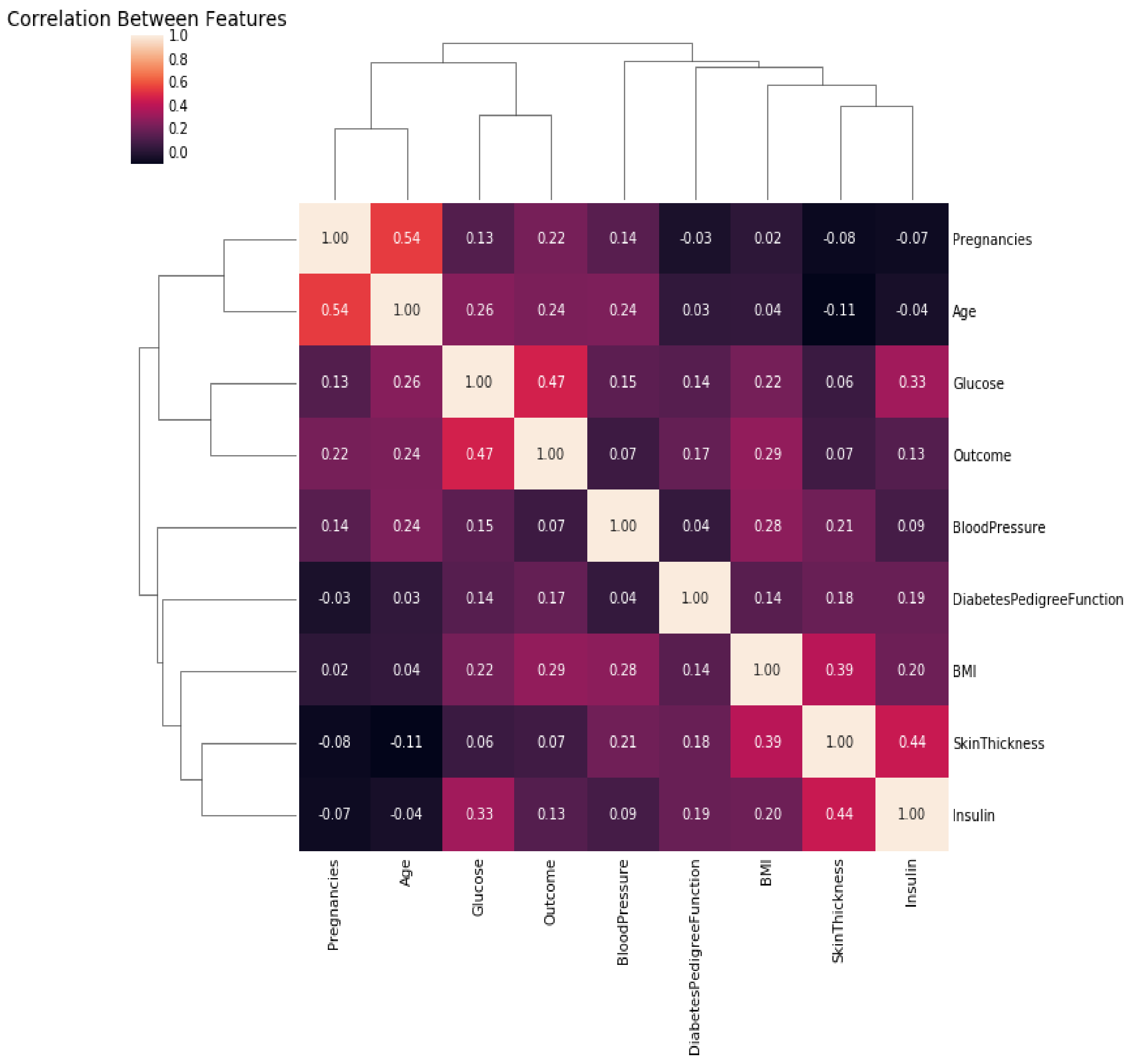
Graph (i)

Interpretation – In the given dataset 65.1 % of the population are healthy (non-diabetic) whereas the remaining 34.9 % are diabetic.



Graph (ii)

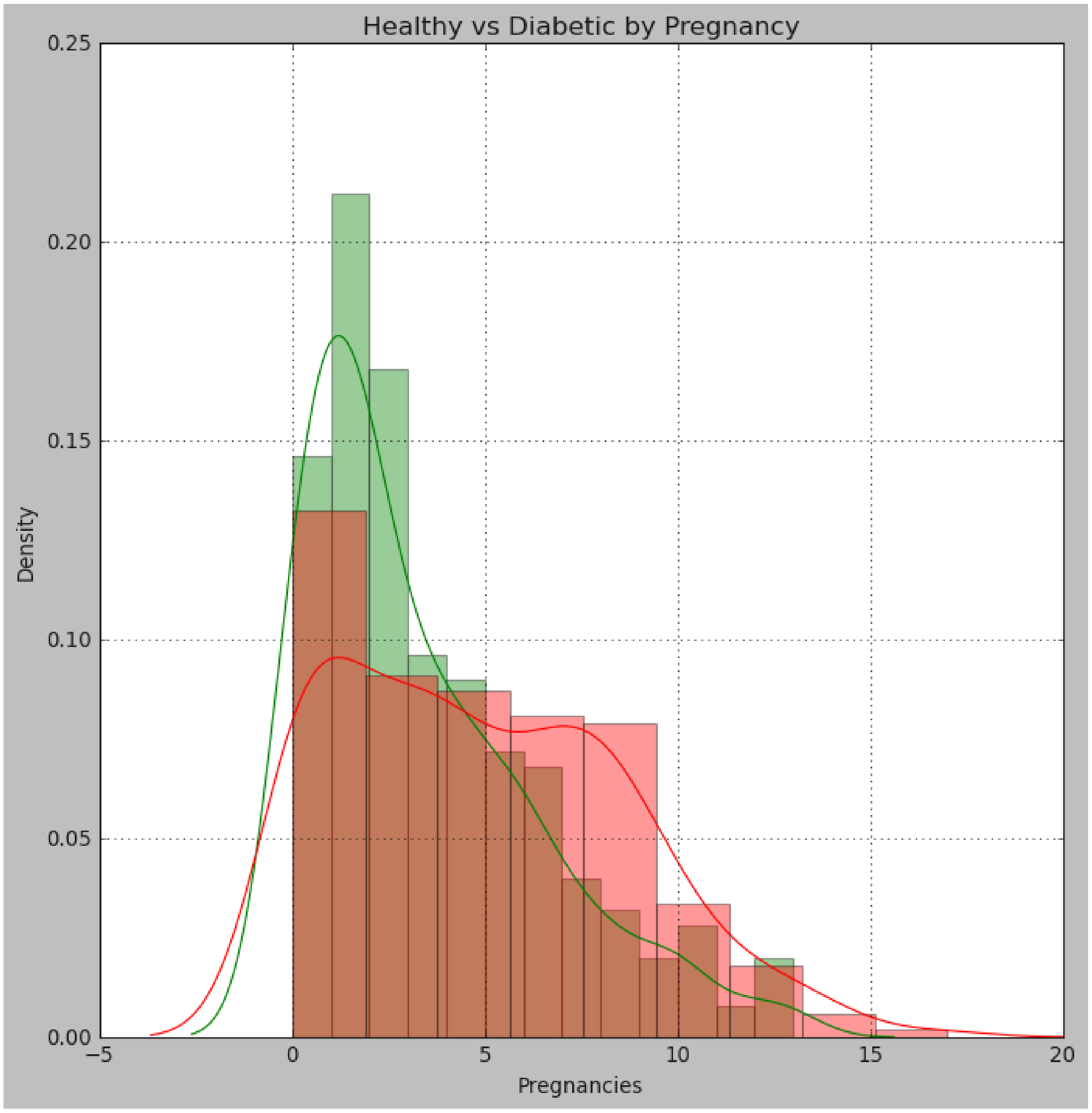
Interpretation – In the above boxplots, it is found that there are outliers in the variables which will be treated eventually before fitting the machine learning classification model.



	Pregnancies	Age	Glucose	Outcome	BloodPressure	DiabetesPedigreeFunction	BMI	SkinThickness	Insulin
Pregnancies	1.00	0.54	0.13	0.22	0.14	-0.03	0.02	-0.08	-0.07
Age	0.54	1.00	0.26	0.24	0.24	0.03	0.04	-0.11	-0.04
Glucose	0.13	0.26	1.00	0.47	0.15	0.14	0.22	0.06	0.33
Outcome	0.22	0.24	0.47	1.00	0.07	0.17	0.29	0.07	0.13
BloodPressure	0.14	0.24	0.15	0.07	1.00	0.04	0.28	0.21	0.09
DiabetesPedigreeFunction	-0.03	0.03	0.14	0.17	0.04	1.00	0.14	0.18	0.19
BMI	0.02	0.04	0.22	0.29	0.28	0.14	1.00	0.39	0.20
SkinThickness	-0.08	-0.11	0.06	0.07	0.21	0.18	0.39	1.00	0.44
Insulin	-0.07	-0.04	0.33	0.13	0.09	0.19	0.20	0.44	1.00

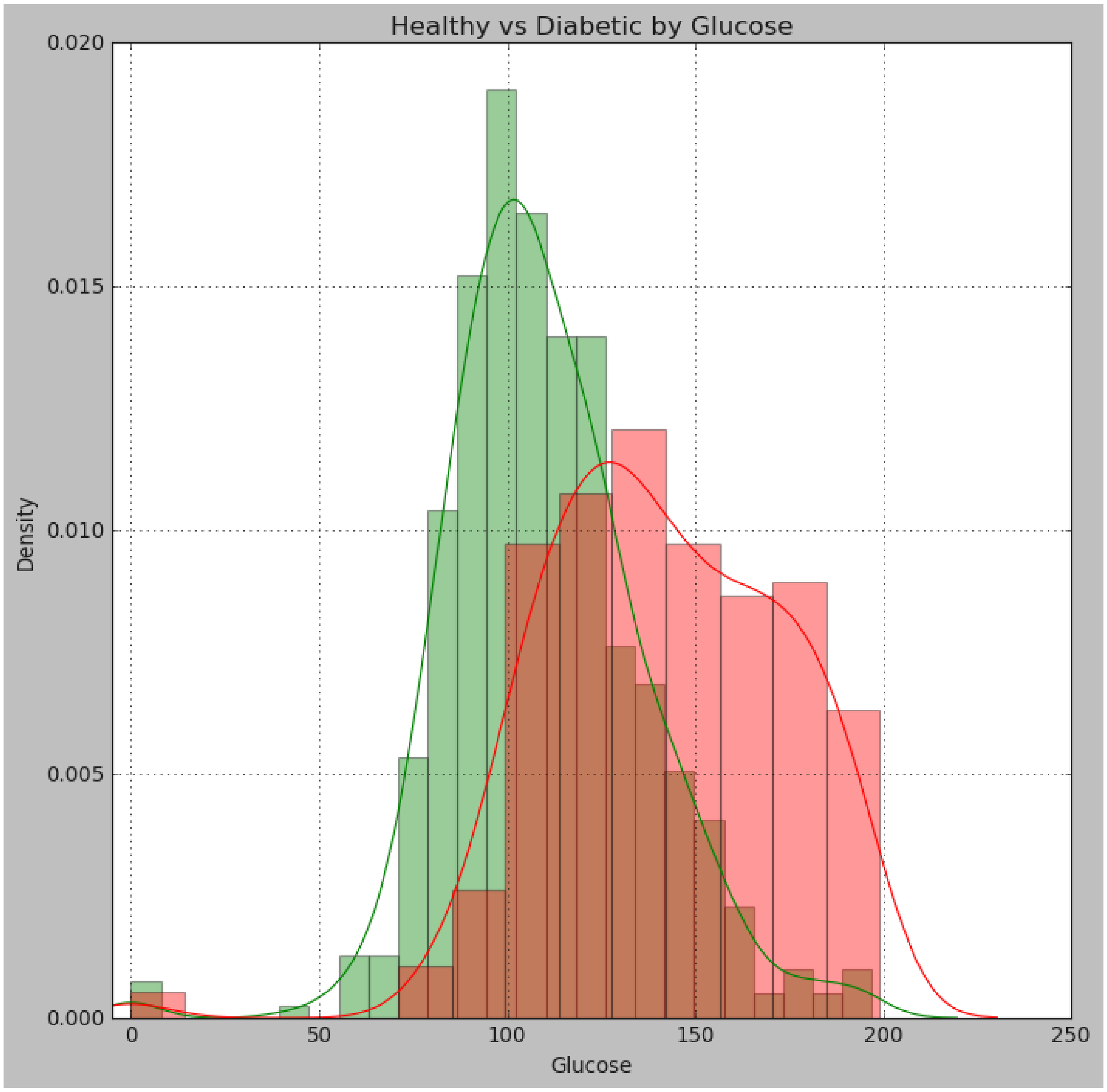
Graph (iii)

Interpretation – The above heatmap shows the correlation between the variables. It is found that there is no strong correlation between the dependent variables and independent variables as well there is no strong correlation among the dependent variables.



Graph (iv)

Interpretation - From above graph, we can say that the Pregnancy isn't likely cause for diabetes as the distribution between the Healthy and Diabetic is almost same.

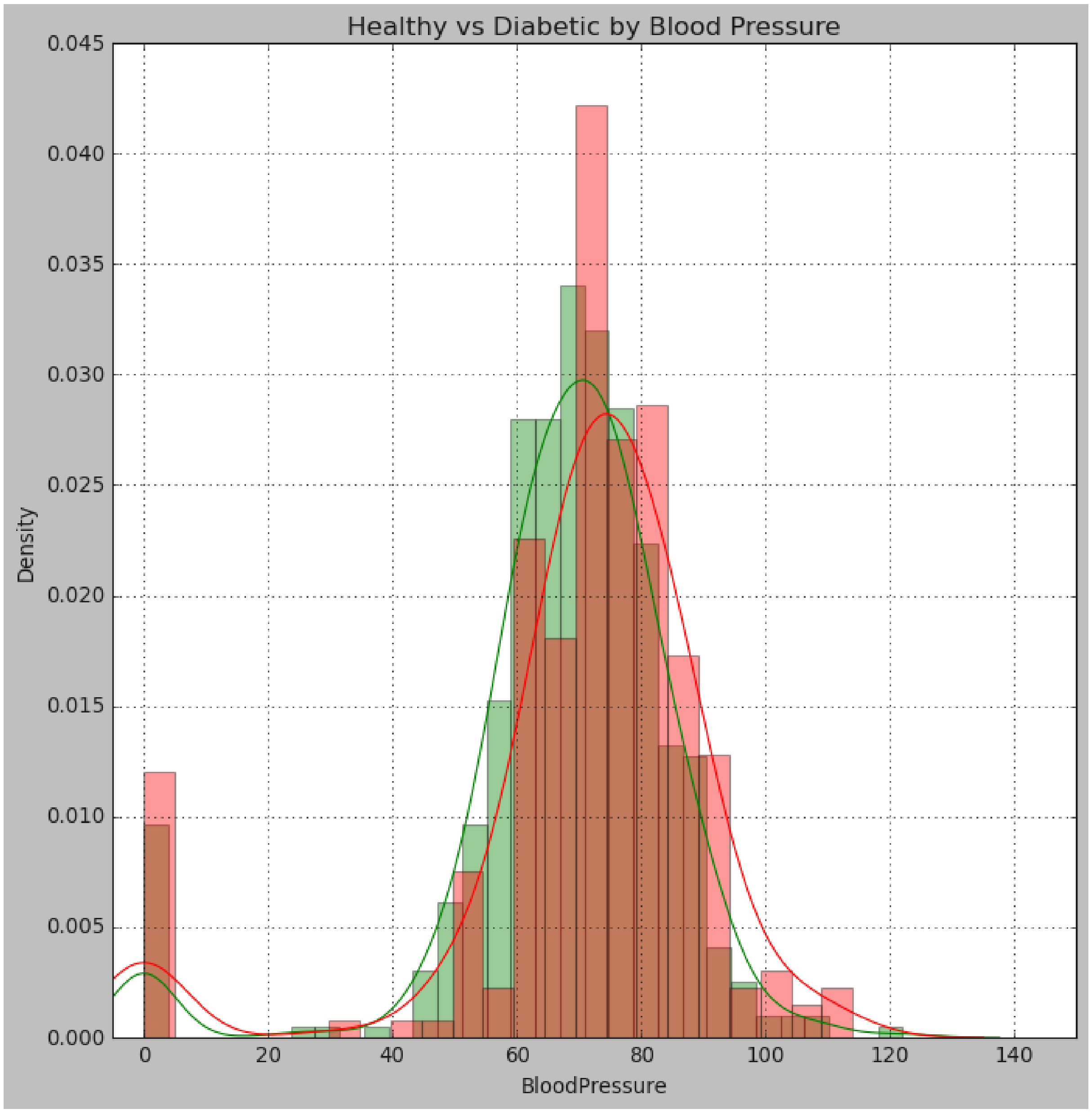


Graph (v)

Interpretation –

The Glucose level for a Normal Adult is around 120-130mg/dl anything above it means that the person is likely suffering from pre-diabetes and diabetes.

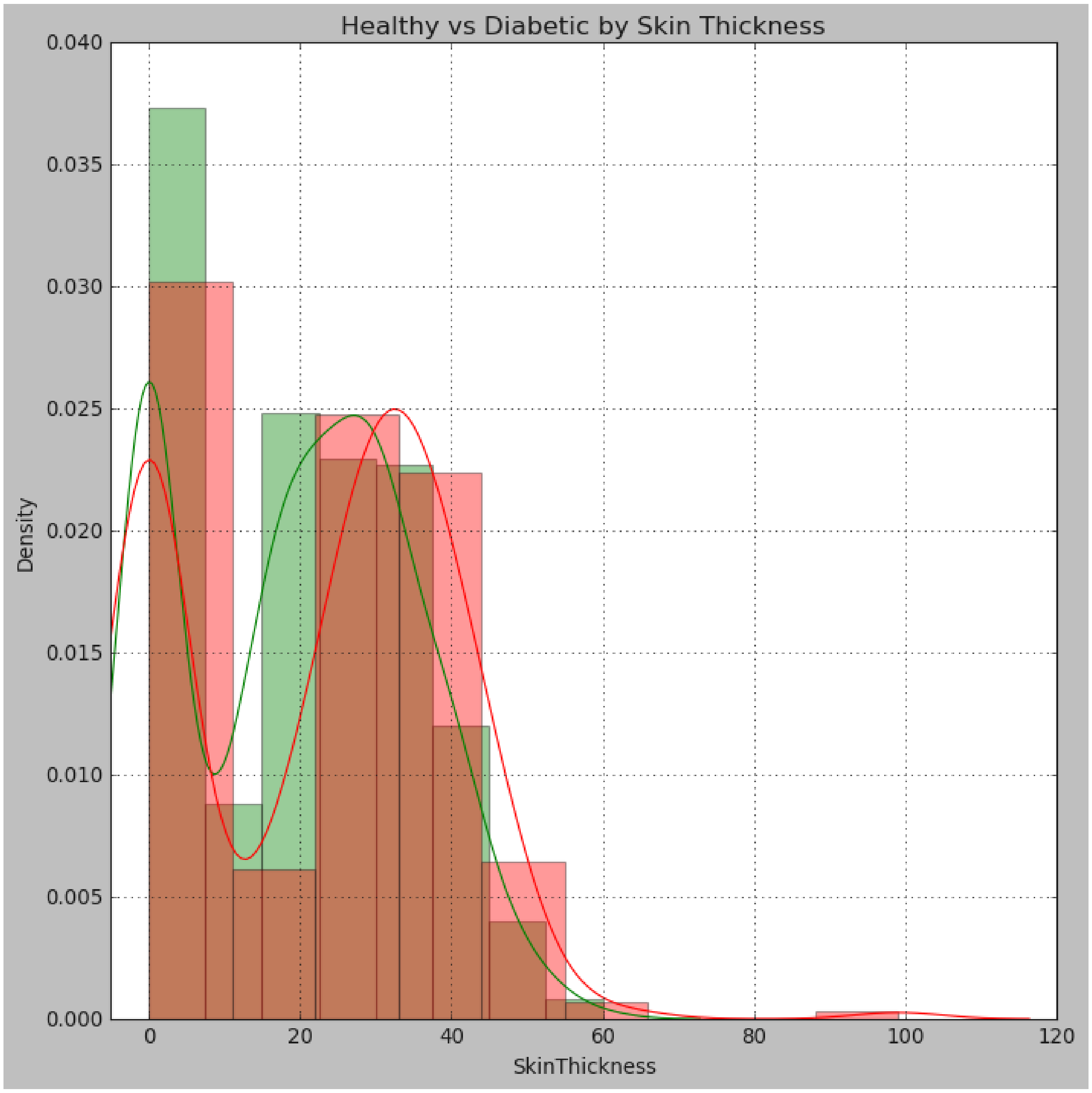
From above graph, we can see the Healthy person are more around 120mg/dl but it then gradually drops, and for diabetic person it is vice versa.



Graph (vi)

Interpretation –

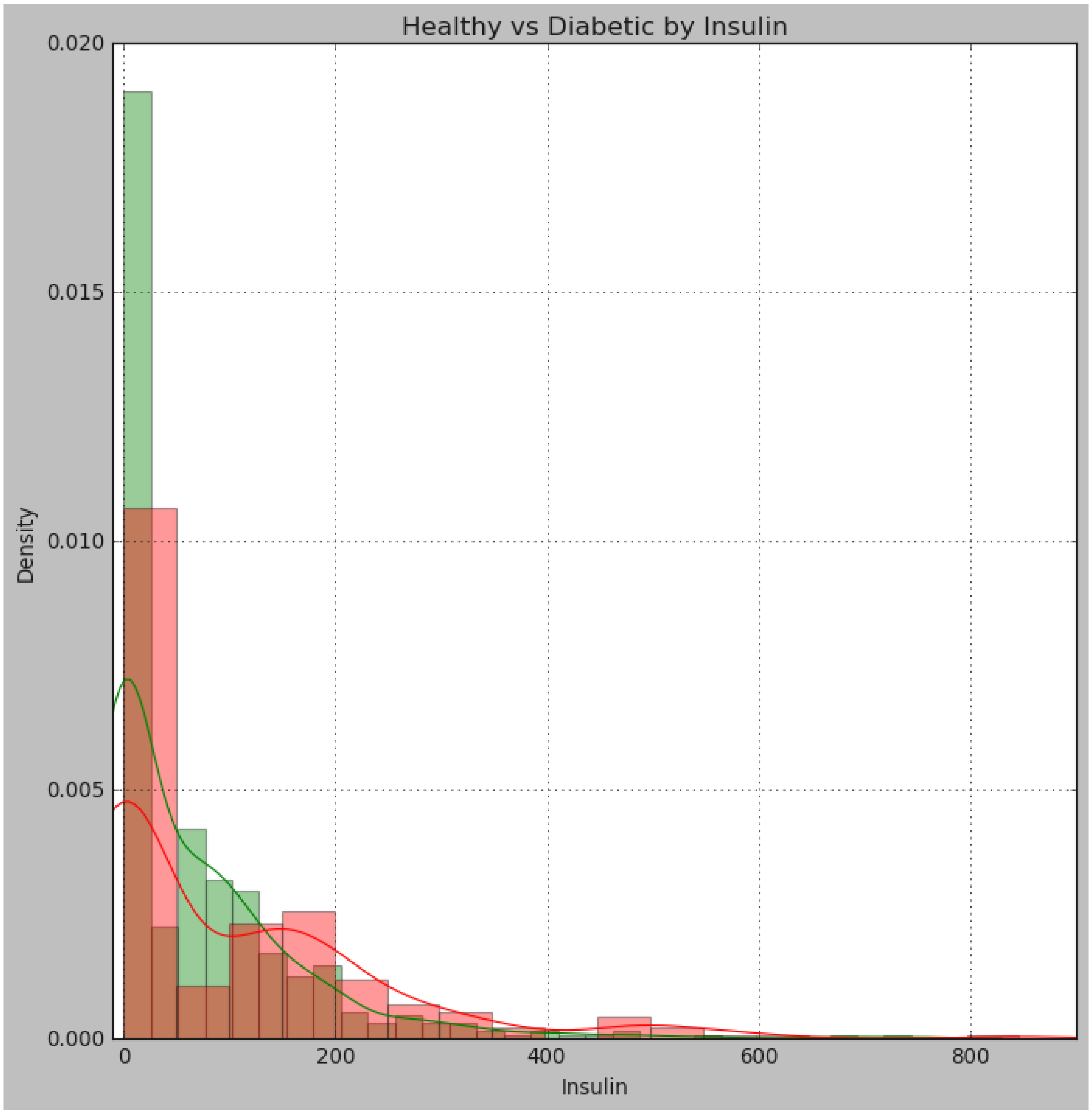
From above graph, it can be concluded that diabetic and healthy people are evenly distributed with low and normal BP but, there are less healthy people who have high BP.



Graph (vii)

Interpretation –

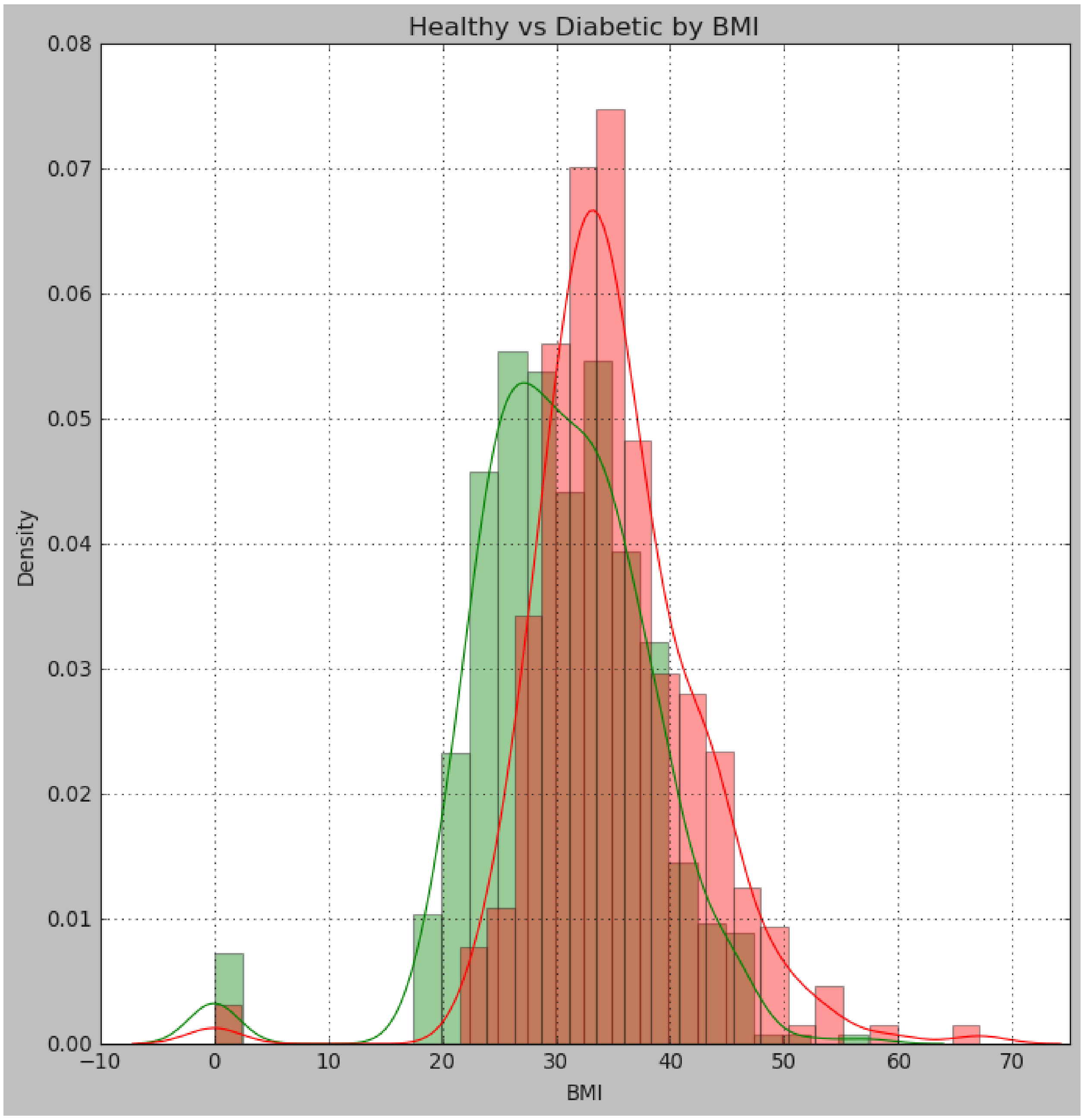
From above graph, the distribution between healthy and diabetic people are around same for skin thickness.



Graph (viii)

Interpretation –

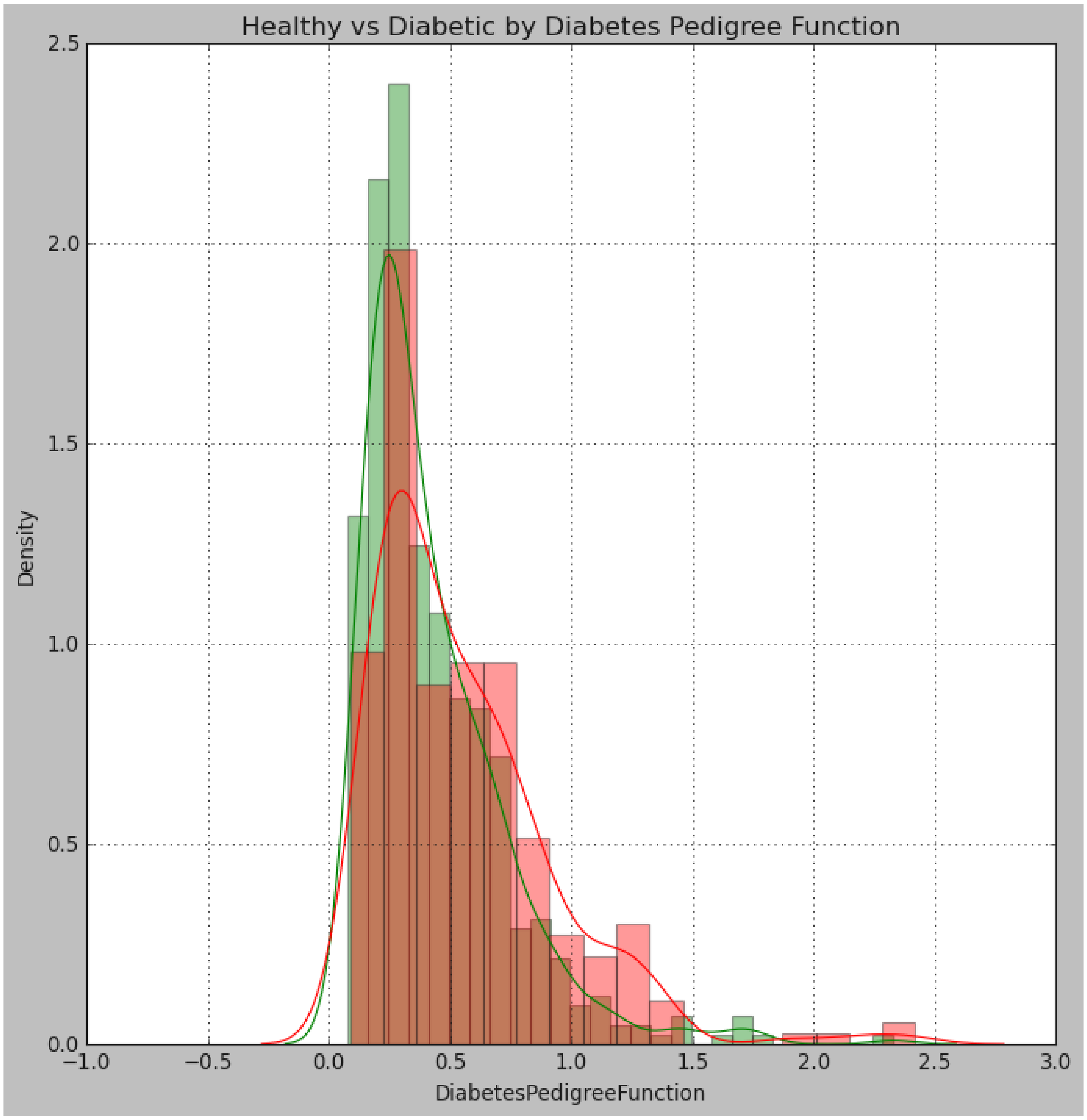
From above graph, it can be concluded that as the level of insulin increases there are high chances of being a diabetic patient. There are more healthy people around insulin levels 0-100.



Graph (ix)

Interpretation –

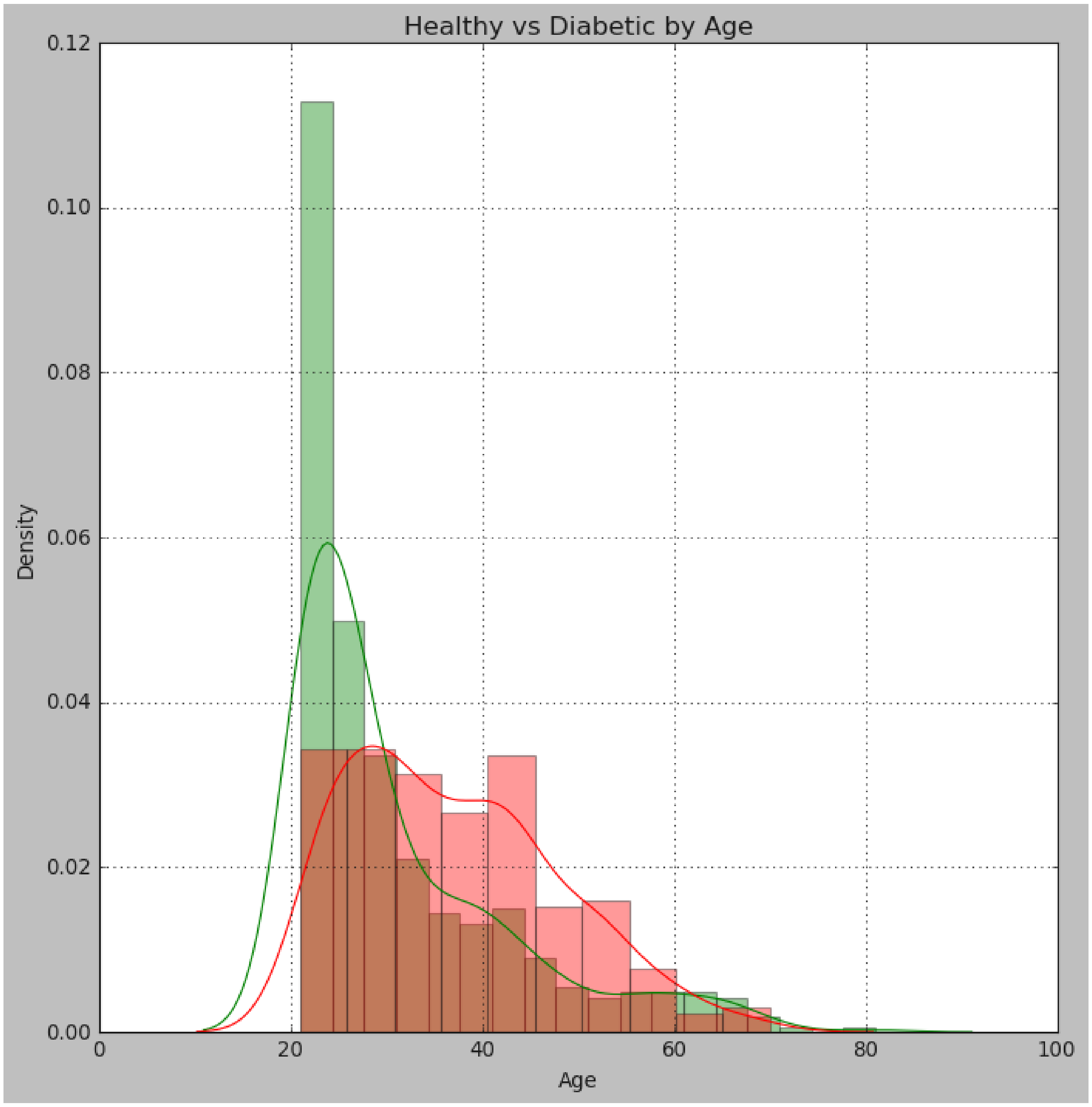
From above graph it can be concluded that, as the BMI increases the person likely being healthy decreases and being diabetic patient increases.



Graph (x)

Interpretation –

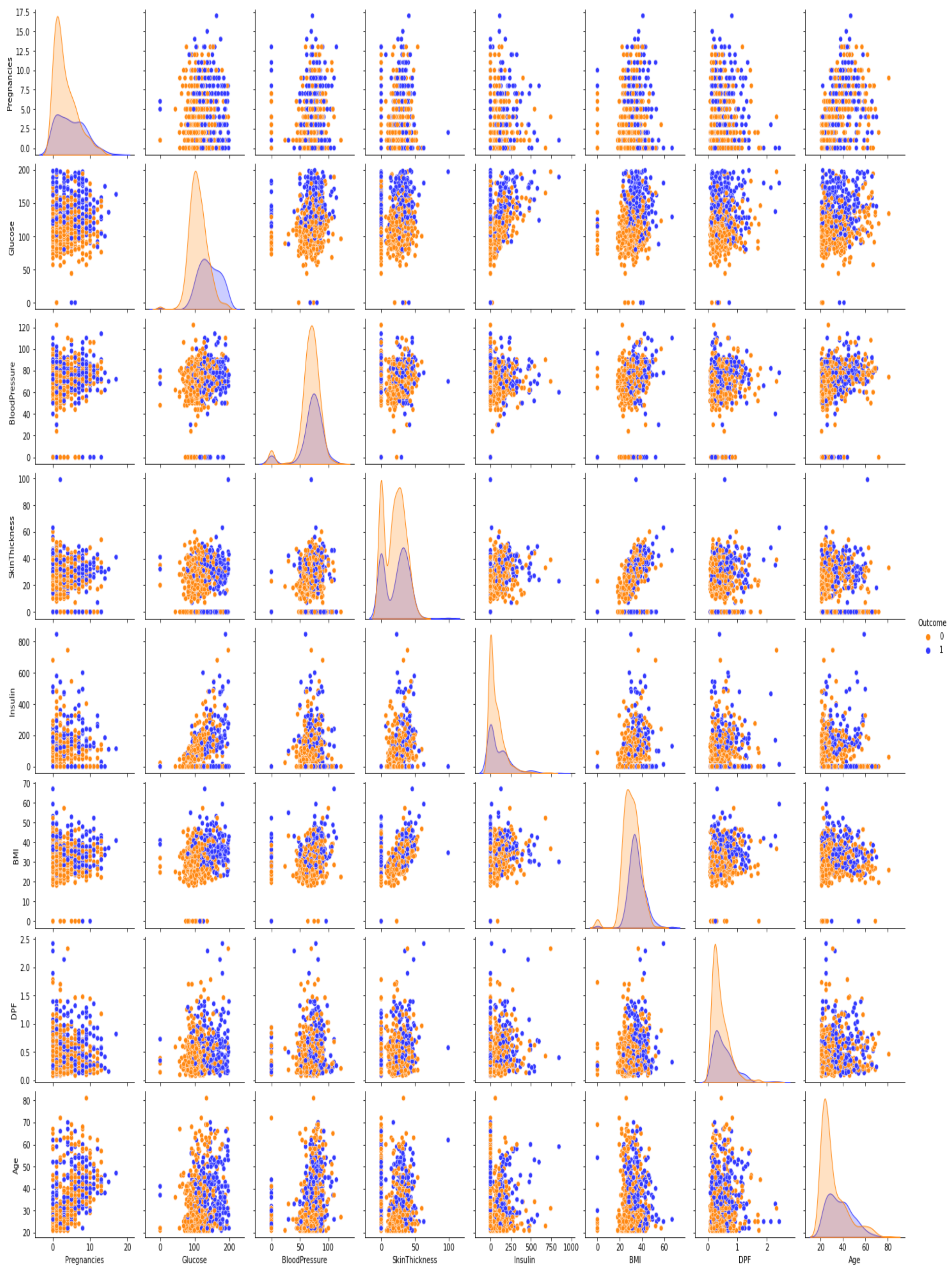
From above graph, as the function increase the diabetic people increases, showing that the diabetes could be hereditary for that individual.



Graph (xi)

Interpretation –

From above graph, it can be said that there are more healthy people around 20-25 age but as the age gradually increases so does the people being diabetic, this shows that age and diabetes go hand in hand.


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DPF
Age
Outcome
0
1

Graph (xii)

Conclusions from Explanatory Data Analysis are as follows: -

- There are no missing (NaN) values in the data.
- The number of outliers in all the variables is very less.
- According to the heatmap, there is no apparent linear correlation between feature variables according to the heatmap.
- The distribution curve of insulin and DPF is right-skewed in nature.
- The distribution curve of Glucose with respect to Outcome shows that there is a smaller number of people with high Glucose level but they have higher chances of diabetes.
- The BloodPressure lies between 40 and 100, and there are fewer people with diabetes in this range.

Graph (xii)/Pair plot tells the following -

- As Insulin and Glucose increase, there are higher chances of Diabetes.
- As BMI and Glucose increase, there are higher chances of Diabetes.
- Age alone isn't really an indicator of Diabetes.
- Middle-aged people with high Glucose levels and high BloodPressure levels have higher chances of Diabetes.

iii. Inferential Statistics

Performing Statistical inference using data analysis to infer properties of distribution of probability. Inferential statistical analysis infers properties of our current population.

Finding the average Blood Pressure of diabetic patient.
Population: Diabetic Patients

Parameter of interest: Average Blood Pressure

```
unbiased_point_estimate = dataset[dataset.Outcome == 1]['BloodPressure'].mean()
std = dataset[dataset.Outcome == 1]['BloodPressure'].std()
(unbiased_point_estimate,std)
```

```
(70.82462686567165, 21.49181165060413)
```

```
Margin_of_error = 1.96 * std/np.sqrt(dataset[dataset.Outcome == 1]['BloodPressure'].count
())
Margin_of_error
```

```
2.57312983381338
```

```
lcb = unbiased_point_estimate - Margin_of_error
ucb = unbiased_point_estimate + Margin_of_error
(lcb,ucb)
```

```
(68.25149703185826, 73.39775669948503)
```

```
sm.stats.DescrStatsW(dataset[dataset.Outcome == 1]['BloodPressure']).zconfint_mean()
```

```
(68.25154431372279, 73.3977094176205)
```

With 95% confidence interval, the average Blood Pressure for a diabetic patient is in between 68.25 to 73.39 mmHg.

Finding the average Blood Pressure of non-diabetic patient.

Population: Diabetic Patients

Parameter of interest: Average Blood Pressure

```
unbiased_point_estimate = dataset[dataset.Outcome == 0]['BloodPressure'].mean()
std = dataset[dataset.Outcome == 0]['BloodPressure'].std()
(unbiased_point_estimate,std)
```

```
(68.184, 18.063075413305828)
```

```
Margin_of_error = 1.96 * std/np.sqrt(dataset[dataset.Outcome == 0]['BloodPressure'].count())
Margin_of_error
```

```
1.5832983686687918
```

```
lcb = unbiased_point_estimate - Margin_of_error
ucb = unbiased_point_estimate + Margin_of_error
(lcb,ucb)
```

```
(66.6007016313312, 69.76729836866879)
```

```
sm.stats.DescrStatsW(dataset[dataset.Outcome == 0]['BloodPressure']).zconfint_mean()
```

```
(66.60073072481028, 69.76726927518972)
```

With 95 % confidence interval, the average blood pressure of non-diabetic patient is between 66.6 and 69.76 mmHg.

Finding the average glucose level for diabetic patient

Population: Diabetic Patient

Parameter of Interest: Average Glucose level

```
unbiased_point_estimate = dataset[dataset.Outcome == 1]['Glucose'].mean()
unbiased_point_estimate
```

```
141.25746268656715
```

```
std = dataset[dataset.Outcome == 1]['Glucose'].std()
std
```

```
31.939622058007195
```

```
std_error = std/np.sqrt(dataset[dataset.Outcome == 1]['Glucose'].count())
std_error
```

```
1.9510229398885643
```

```
lcb = unbiased_point_estimate - 1.96 *std_error
ucb = unbiased_point_estimate + 1.96 *std_error
(lcb , ucb)
```

```
(137.43345772438556, 145.08146764874874)
```

```
sm.stats.DescrStatsW(dataset[dataset.Outcome == 1]['Glucose']).zconfint_mean()
```

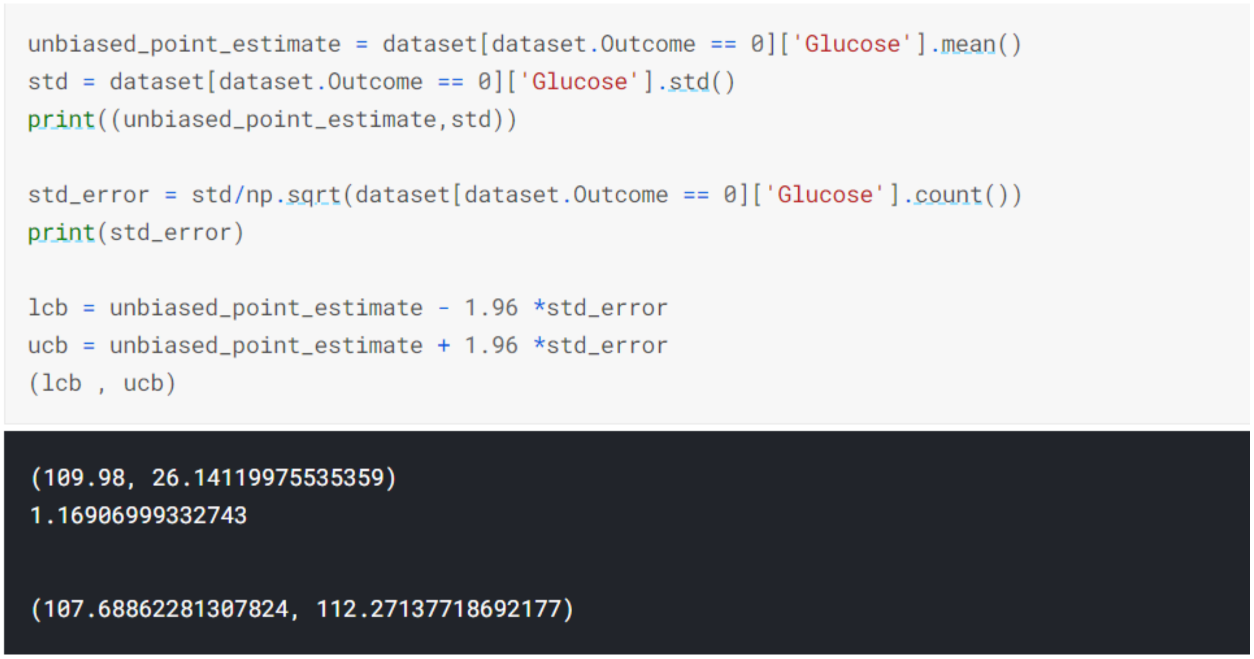
```
(137.43352799137412, 145.08139738176018)
```

With 95% confidence interval, the population average glucose level for diabetic patient is estimated to be in between 138 to 145 mg/dL.

Finding average glucose level for non-diabetic patient

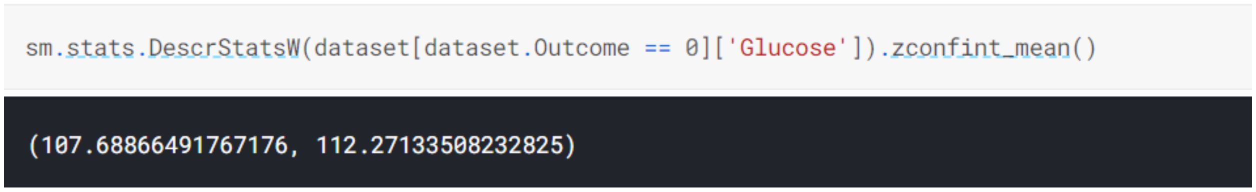
Population: Non-Diabetic Patient

Parameter of Interest: Average Glucose level



```
unbiased_point_estimate = dataset[dataset.Outcome == 0]['Glucose'].mean()
std = dataset[dataset.Outcome == 0]['Glucose'].std()
print((unbiased_point_estimate,std))

std_error = std/np.sqrt(dataset[dataset.Outcome == 0]['Glucose'].count())
print(std_error)

lcb = unbiased_point_estimate - 1.96 *std_error
ucb = unbiased_point_estimate + 1.96 *std_error
(lcb , ucb)
```

```
(109.98, 26.14119975535359)
1.1690699332743


(107.68862281307824, 112.27137718692177)
```



```
sm.stats.DescrStatsW(dataset[dataset.Outcome == 0]['Glucose']).zconfint_mean()
```

```
(107.68866491767176, 112.27133508232825)
```

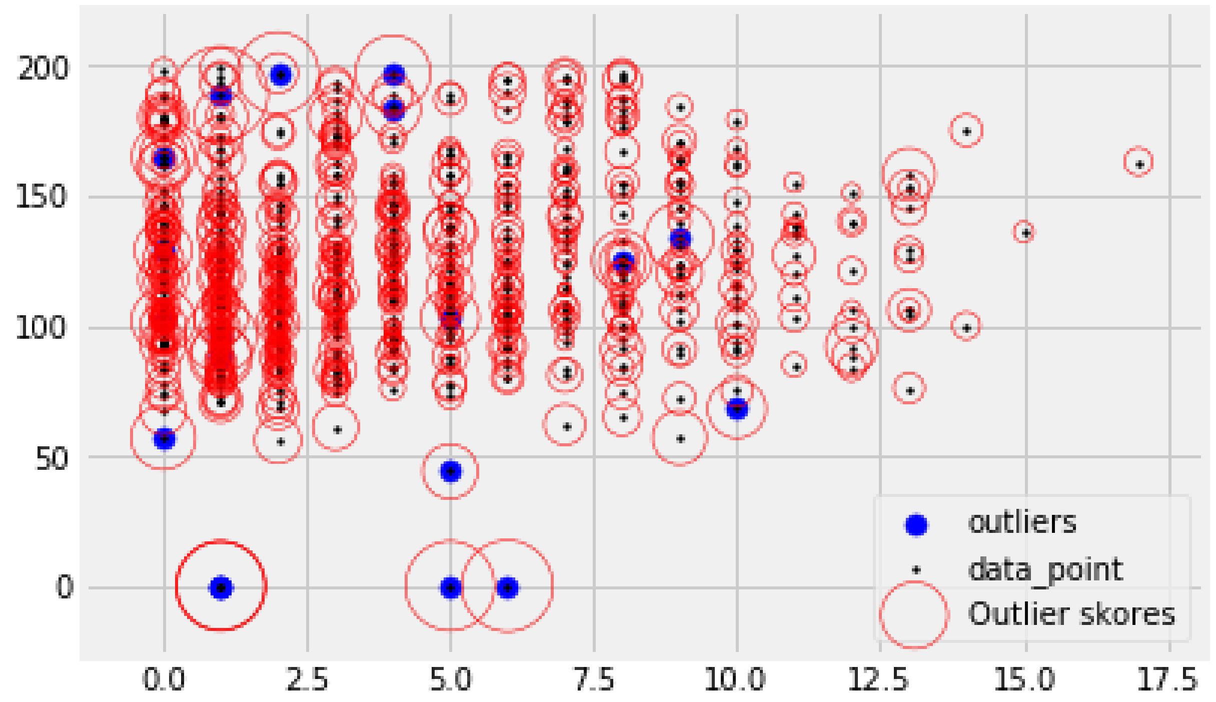
There is a clear distinction of glucose level between the diabetic and non-diabetic patient. With 95% confidence the population average glucose level for non-diabetic patients is estimated to be in between 108 to 112 mg/dL. (Rounded off to nearest decimal).

iv. Data Preprocessing

```
df.isnull().sum()
```

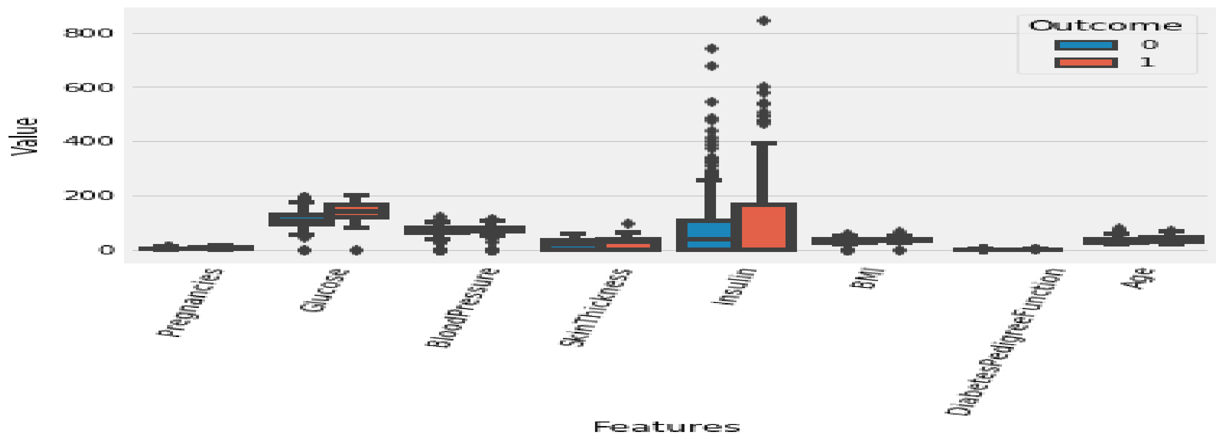
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DPF              0
Age              0
Outcome          0
dtype: int64
```

The given dataset has no missing values. There is no need for the imputation in the dataset.

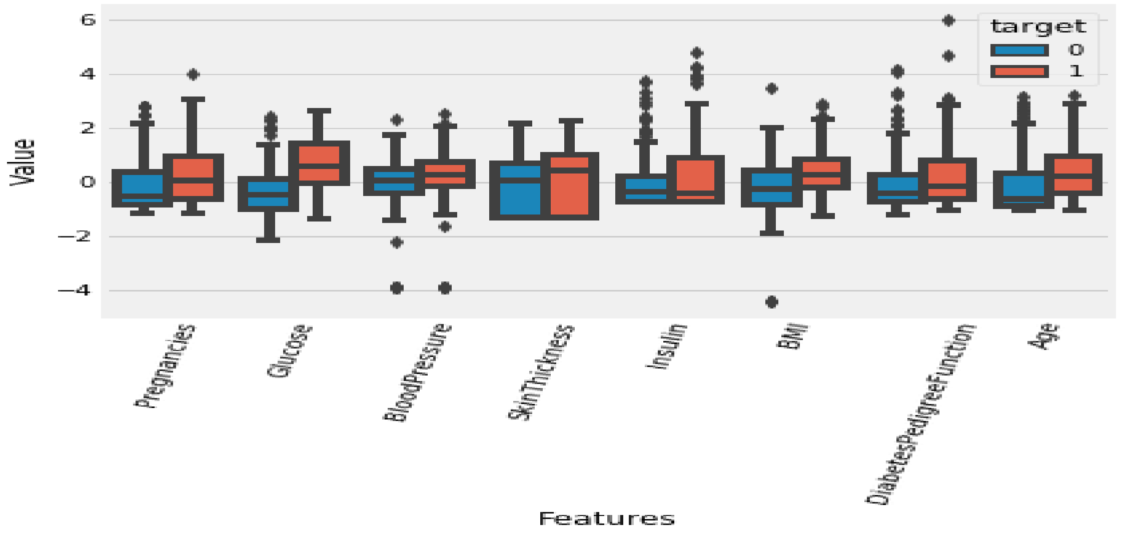


The above plot is been created with help of the LocalOutlierFactor() function in python. The names of the variables were gathered during a single-column list. With the assistance of the Local Outlier Factor (LOF) algorithm which is a highly unsupervised anomaly detection method that computes the local density deviation of a given datum with regard to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors [4]. Then outlier values with threshold were applied to a listing and later normalized the values to avoid bias. To drop the outliers the function that has been used is drop(outlier_index) where outlier_index is the variable where outlier values with threshold were applied to a list.

Dataset may contain attributes with a mix of scales for various quantities. It is said that the more effective the data would be if the attributes have the same scale. To avoid this problem, the data has been standardized. In python, StandardScaler() removes the mean and scales variable to the unit variance and this is been operated independently by features. It can be influenced by outliers so before standardizing it has been made sure that data has outliers has been removed.



Before Standardization



After Standardization

v. Train-Test split

Using the train-test split technique to evaluate the performance for the classification models.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values

The given dataset has divided 70 percent of the dataset as Train Dataset by using Simple Random Sampling technique and the remaining 30 percent of the dataset as Test Dataset by using Simple Random Sampling Technique.

vi. <u>Working Procedure</u>

To check the adequacy of classifiers performance measures need to be taken into account. True positives (TP) refer to the positive cases that were correctly labelled by the classifier, while true negatives (TN) are the negative cases that were correctly labeled the classifier. False positives (FP) are the negative cases that were incorrectly labelled, while false negatives (FN) are the positive cases that were incorrectly labelled. Some evaluation measures of classifiers are as follows

(1) Accuracy: Overall, how often is the classifier correct?
Accuracy = (TN)/(TP+TN+FP+FN).
(2) Specificity When it's actually no, how often does it predict no?
Specificity = (TN)/(TN+FP).
(3) Sensitivity: When it's actually yes, how often does it predict yes?
Sensitivity = (TP)/(TP+FN).
(4) F1-Score: This is a weighted average of the true positive rate (recall) and precision.
F1-score = 2TP/(2TP+FP+FN)
(5) Precision: When it predicts yes, how often is it correct?
Precision = TP/(TP+FP)

8. Results

After doing all the data pre-processing and Train-Test split. It’s time to train and test the machine learning algorithms and check its accuracy.

1. **Logistic Regression**

   Assumptions: -

- Observations should come from independent sample
- The dependent variable should be binary
- Logistic Regression requires little or no multicollinearity
- Large sample size
- Logistic regression assumes linearity of independent variables and log odds
- No outliers

During the data pre-processing it has been made sure that dataset has no multicollinearity and outliers.

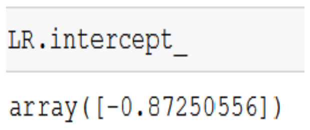
Checking the significance of the model

H_0: All the variables in the model are statistically insignificant i.e.

$$\beta_1 = \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_7 = \beta_8 = 0$$

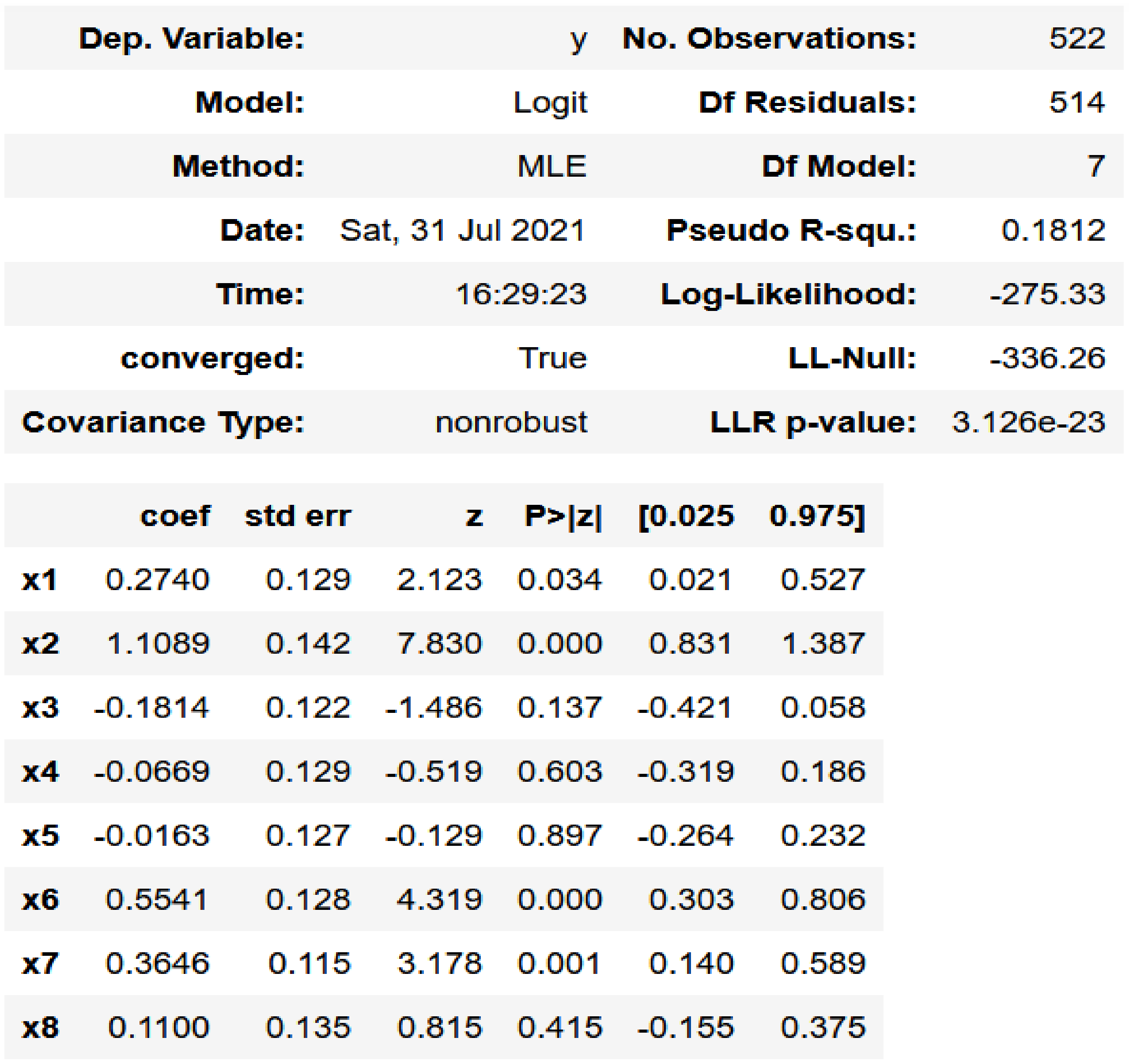
H_1: At least one of the independent variables is statistically significant i.e.

H_1: at least one $\beta i \neq 0$ (i=1,2,3,4,5,6,7,8)



```
LR.intercept_
```

```
array([-0.87250556])
```

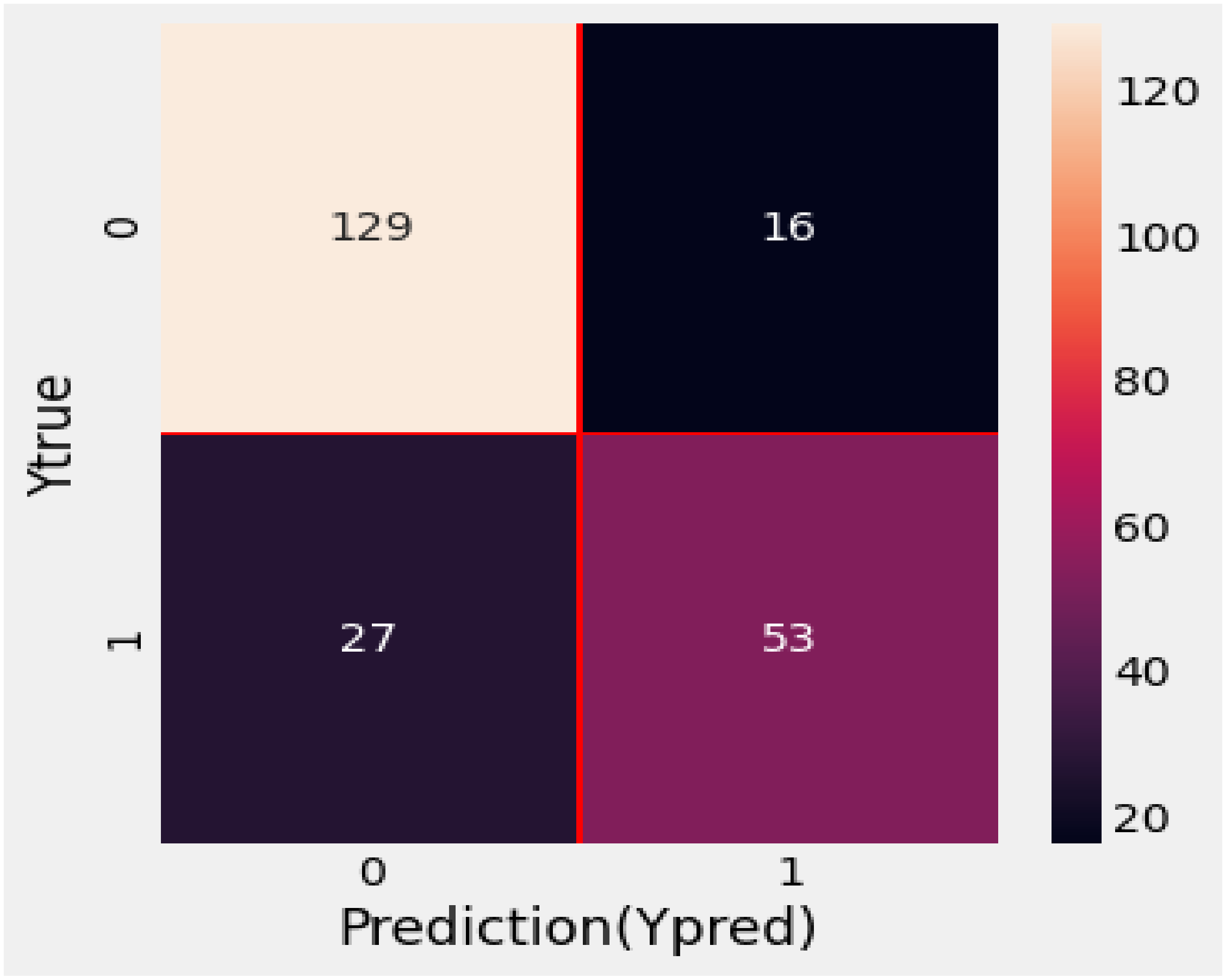


Dep. Variable:	y	**No. Observations:**	522
Model:	Logit	**Df Residuals:**	514
Method:	MLE	**Df Model:**	7
Date:	Sat, 31 Jul 2021	**Pseudo R-squ.:**	0.1812
Time:	16:29:23	**Log-Likelihood:**	-275.33
converged:	True	**LL-Null:**	-336.26
Covariance Type:	nonrobust	**LLR p-value:**	3.126e-23

	coef	std err	z	P>|z|	[0.025	0.975]
x1	0.2740	0.129	2.123	0.034	0.021	0.527
x2	1.1089	0.142	7.830	0.000	0.831	1.387
x3	-0.1814	0.122	-1.486	0.137	-0.421	0.058
x4	-0.0669	0.129	-0.519	0.603	-0.319	0.186
x5	-0.0163	0.127	-0.129	0.897	-0.264	0.232
x6	0.5541	0.128	4.319	0.000	0.303	0.806
x7	0.3646	0.115	3.178	0.001	0.140	0.589
x8	0.1100	0.135	0.815	0.415	-0.155	0.375

Since p-value 3.126e-23 < 0.05, we reject our null hypothesis (H_0) and conclude that the model Logistic Regression is significantly fitting the data better than a null model with no predictors.

The model has been fitted and it gives the confusion matrix as shown below.



The results obtained from the confusion matrix are as follows:

1. Accuracy = 80.89 %
2. Specificity = 76.81 %
3. Sensitivity = 82.69 %
4. F1-score = 85.71 %
5. Precision = 88.97 %

The fitted logistic regression model is given by

f(x)= -0.8720556(constant)+2.74(x1) +1.1089(x2)-0.1814(x3)-0.0669(x4)-0.0163(x5) +0.5541(x6) +0.3645(x7) +0.11(x8)

where,

x1= Pregnancies

x2= Glucose

x3= BloodPressure

x4= SkinThickness

x5= Insulin

x6= BMI

x7= DiabetesPedigreeFunction

x8= Age

Interpretation –

The Logistic Regression model accurately predicts whether the person is diabetic or not with an accuracy of 80.89 %

2. **K-Nearest Neighbor –**

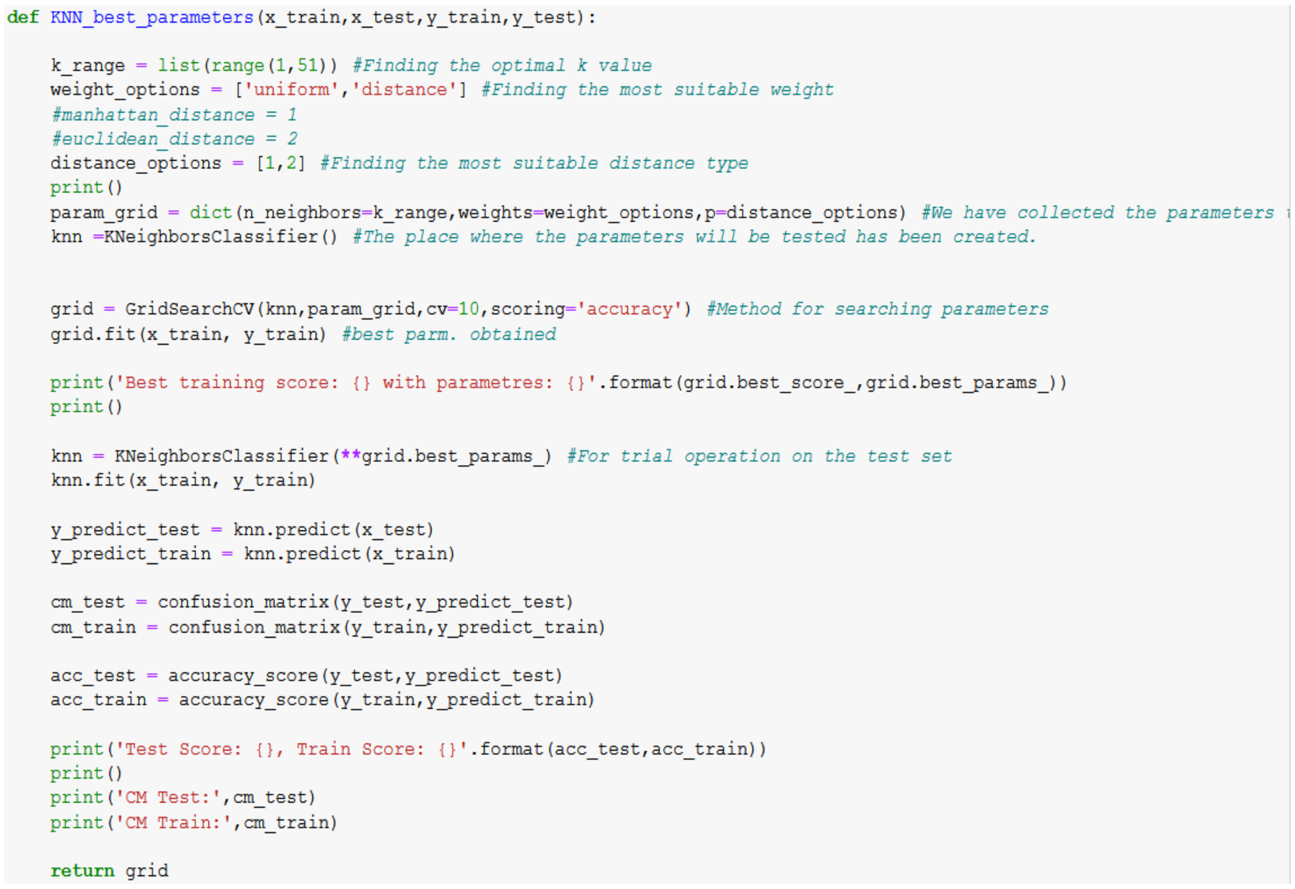
Assumptions

•KNN assumes that the data is in a feature space. More exactly, the data points are in a metric space. The data can be scalars or possibly even multidimensional vectors. Since the points are in feature space, they have a notion of distance – This need not necessarily be Euclidean distance although it is the one commonly used.

•Each of the training data consists of a set of vectors and class labels associated with each vector. In the simplest case, it will be either + or – (for positive or negative classes). But KNN, can work equally well with an arbitrary number of classes.

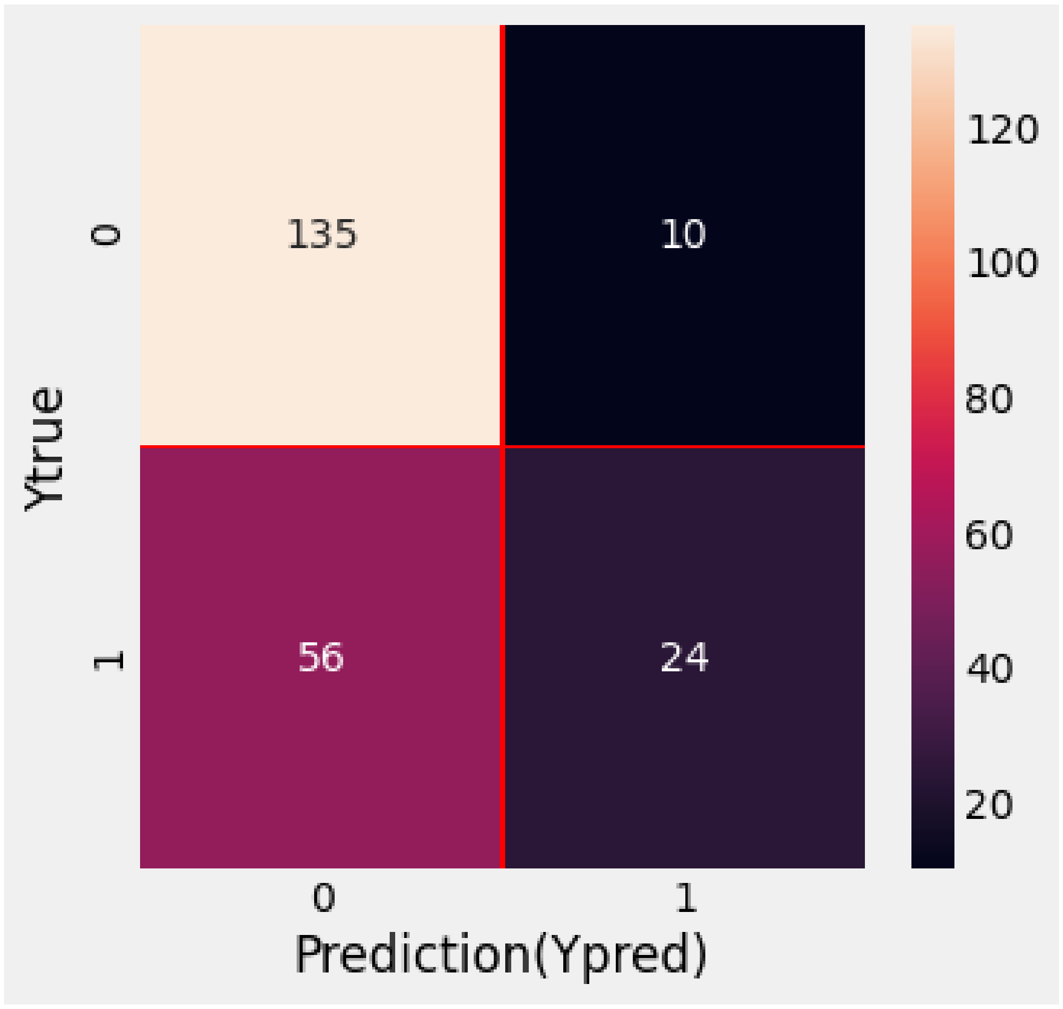
•We are also given a single number "k”. This number decides how many neighbors (where neighbors are defined based on the distance metric) influence the classification. This is usually an odd number if the number of classes is 2. If k=1, then the algorithm is simply called the nearest neighbor algorithm.

Fitting the model with parameter tuning (finding the best parameters for the model).



```
def KNN_best_parameters(x_train,x_test,y_train,y_test):

    k_range = list(range(1,51)) #Finding the optimal k value
    weight_options = ['uniform','distance'] #Finding the most suitable weight
    #manhattan_distance = 1
    #euclidean_distance = 2
    distance_options = [1,2] #Finding the most suitable distance type
    print()
    param_grid = dict(n_neighbors=k_range,weights=weight_options,p=distance_options) #We have collected the parameters
    knn =KNeighborsClassifier() #The place where the parameters will be tested has been created.


    grid = GridSearchCV(knn,param_grid,cv=10,scoring='accuracy') #Method for searching parameters
    grid.fit(x_train, y_train) #best parm. obtained

    print('Best training score: {} with parametres: {}'.format(grid.best_score_,grid.best_params_))
    print()

    knn = KNeighborsClassifier(**grid.best_params_) #For trial operation on the test set
    knn.fit(x_train, y_train)

    y_predict_test = knn.predict(x_test)
    y_predict_train = knn.predict(x_train)

    cm_test = confusion_matrix(y_test,y_predict_test)
    cm_train = confusion_matrix(y_train,y_predict_train)

    acc_test = accuracy_score(y_test,y_predict_test)
    acc_train = accuracy_score(y_train,y_predict_train)

    print('Test Score: {}, Train Score: {}'.format(acc_test,acc_train))
    print()
    print('CM Test:',cm_test)
    print('CM Train:',cm_train)

    return grid
```

The metric used in the KNN classification algorithm is “**Euclidean**”. After tuning the model, the “**n-neighbor**” which is best for the model is 36. The confusion matrix after tuning the parameters is as shown below:



The results obtained from the confusion matrix are as follows:

1. Accuracy = 70.67 %
2. Specificity = 70.59 %
3. Sensitivity = 70.68 %
4. F1-score = 80.36 %
5. Precision = 93.10 %

Interpretation –

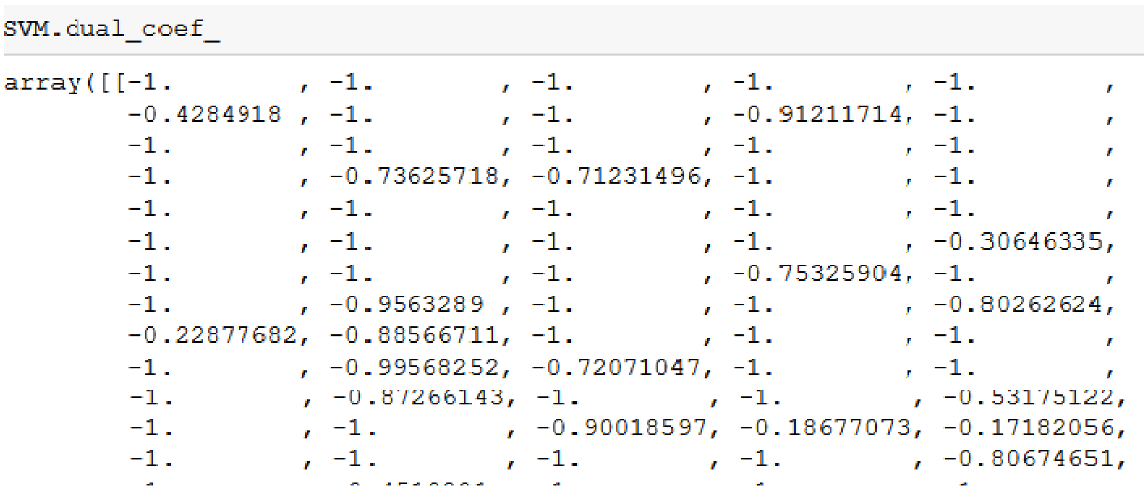
The K-Nearest Neighbors model accurately predicts whether the person is diabetic or not with an accuracy of 70.67 %

3. Support Vector Machine

Assumption: It assumes data is independent and identically distributed.

The kernel used in the Support Vector Machine is “rbf (Radial Basis Function).”

Dual coefficients of the support vector in the decision function multiplied by their targets.

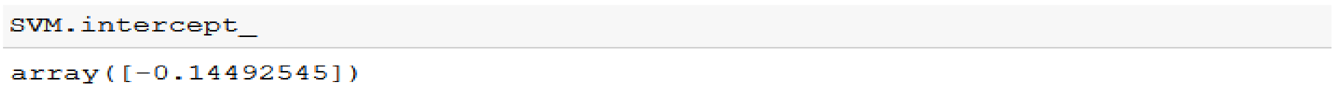


```
SVM.dual_coef_
```

```
array([[-1.        , -1.        , -1.        , -1.        , -1.        ,
        -0.4284918 , -1.        , -1.        , -0.91211714, -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -0.73625718, -0.71231496, -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -1.        ,
        -1.        , -1.        , -1.        , -1.        , -0.30646335,
        -1.        , -1.        , -1.        , -0.75325904, -1.        ,
        -1.        , -0.9563289 , -1.        , -1.        , -0.80262624,
        -0.22877682, -0.88566711, -1.        , -1.        , -1.        ,
        -1.        , -0.99568252, -0.72071047, -1.        , -1.        ,
        -1.        , -0.87266143, -1.        , -1.        , -0.53175122,
        -1.        , -1.        , -0.90018597, -0.18677073, -0.17182056,
        -1.        , -1.        , -1.        , -1.        , -0.80674651,
```

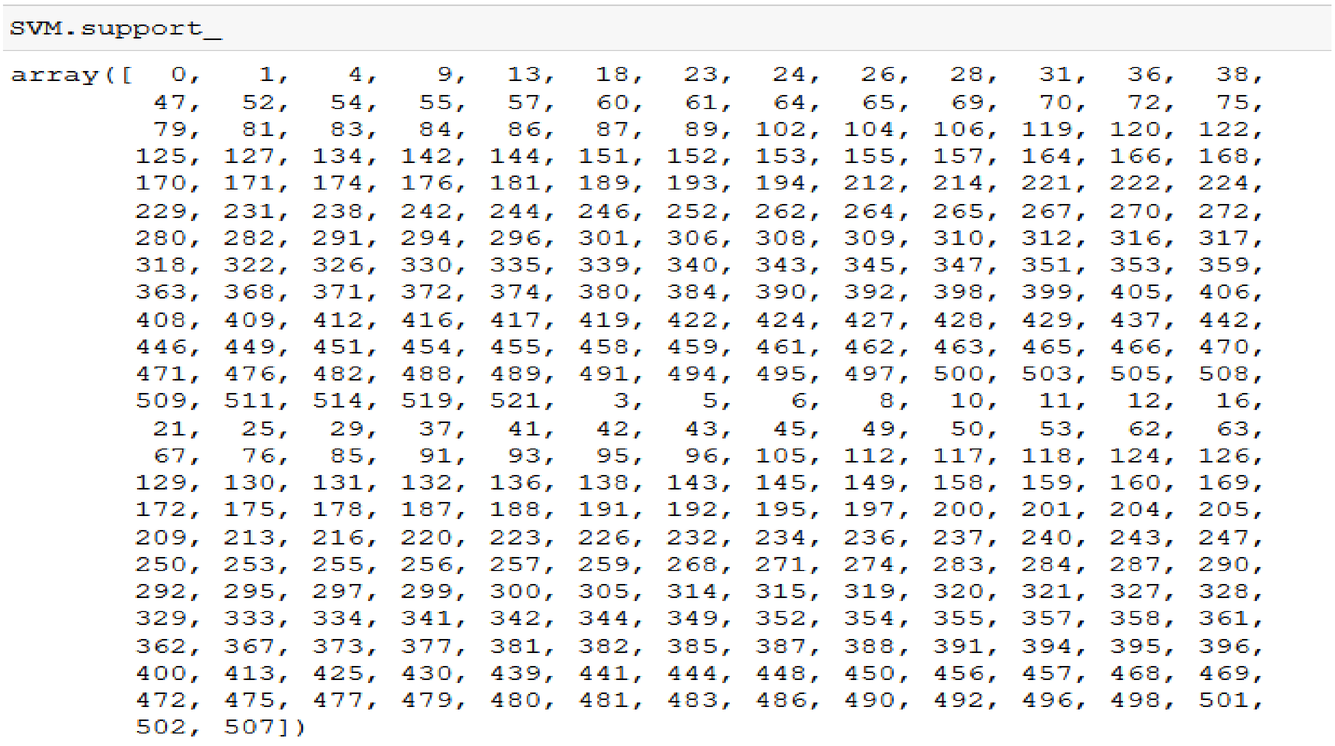
Note: - Since the dual coefficient are too big this is not complete dual coefficients of the SVM model

The intercept ndarray of shape (n_classes * (n_classes - 1) / 2,) which shows the Constants in decision.



```
SVM.intercept_
```

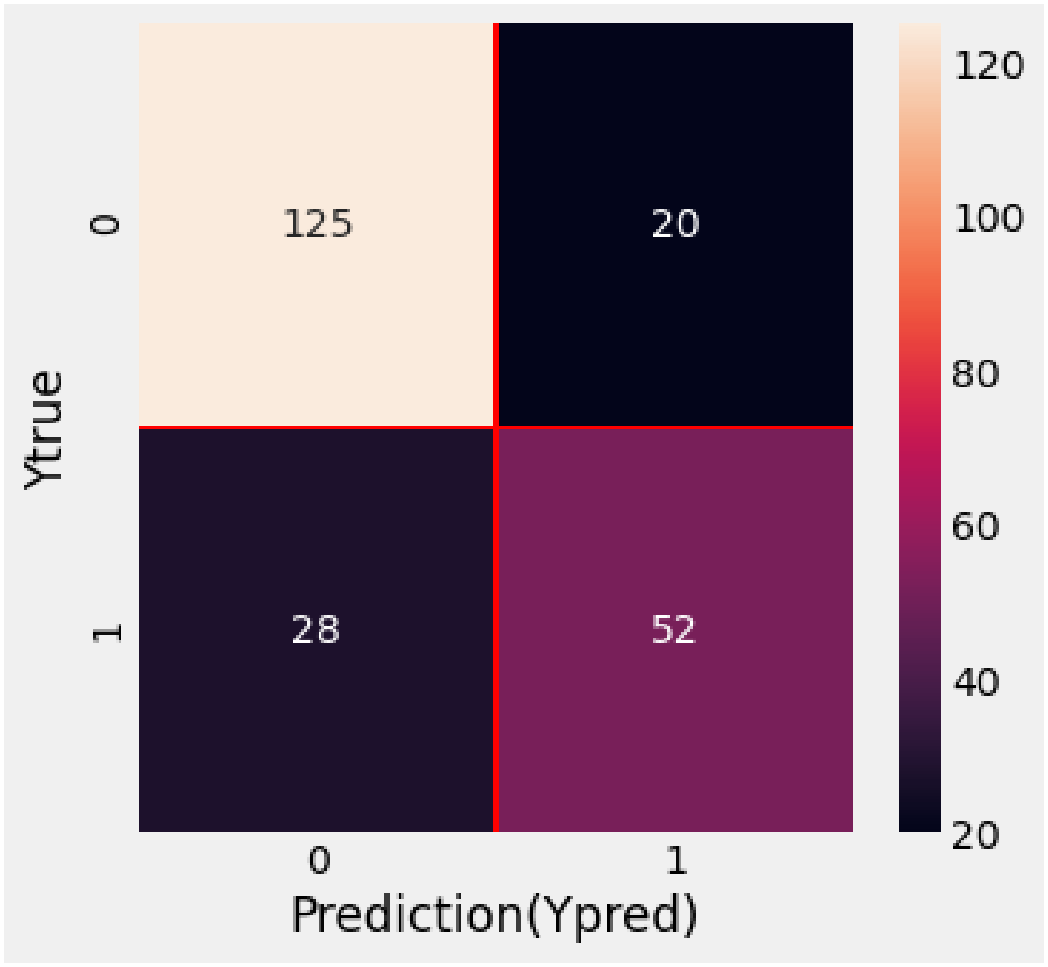
```
array([-0.14492545])
```



```
SVM.support_
```

```
array([  0,   1,   4,   9,  13,  18,  23,  24,  26,  28,  31,  36,  38,
        47,  52,  54,  55,  57,  60,  61,  64,  65,  69,  70,  72,  75,
        79,  81,  83,  84,  86,  87,  89, 102, 104, 106, 119, 120, 122,
       125, 127, 134, 142, 144, 151, 152, 153, 155, 157, 164, 166, 168,
       170, 171, 174, 176, 181, 189, 193, 194, 212, 214, 221, 222, 224,
       229, 231, 238, 242, 244, 246, 252, 262, 264, 265, 267, 270, 272,
       280, 282, 291, 294, 296, 301, 306, 308, 309, 310, 312, 316, 317,
       318, 322, 326, 330, 335, 339, 340, 343, 345, 347, 351, 353, 359,
       363, 368, 371, 372, 374, 380, 384, 390, 392, 398, 399, 405, 406,
       408, 409, 412, 416, 417, 419, 422, 424, 427, 428, 429, 437, 442,
       446, 449, 451, 454, 455, 458, 459, 461, 462, 463, 465, 466, 470,
       471, 476, 482, 488, 489, 491, 494, 495, 497, 500, 503, 505, 508,
       509, 511, 514, 519, 521,   3,   5,   6,   8,  10,  11,  12,  16,
        21,  25,  29,  37,  41,  42,  43,  45,  49,  50,  53,  62,  63,
        67,  76,  85,  91,  93,  95,  96, 105, 112, 117, 118, 124, 126,
       129, 130, 131, 132, 136, 138, 143, 145, 149, 158, 159, 160, 169,
       172, 175, 178, 187, 188, 191, 192, 195, 197, 200, 201, 204, 205,
       209, 213, 216, 220, 223, 226, 232, 234, 236, 237, 240, 243, 247,
       250, 253, 255, 256, 257, 259, 268, 271, 274, 283, 284, 287, 290,
       292, 295, 297, 299, 300, 305, 314, 315, 319, 320, 321, 327, 328,
       329, 333, 334, 341, 342, 344, 349, 352, 354, 355, 357, 358, 361,
       362, 367, 373, 377, 381, 382, 385, 387, 388, 391, 394, 395, 396,
       400, 413, 425, 430, 439, 441, 444, 448, 450, 456, 457, 468, 469,
       472, 475, 477, 479, 480, 481, 483, 486, 490, 492, 496, 498, 501,
       502, 507])
```

After tuning the parameter, the results of the confusion matrix are shown below:



The results obtained from the confusion matrix are as follows:

1. Accuracy = 78.67 %
2. Specificity = 72.22 %
3. Sensitivity = 81.7 %
4. F1-score = 83.89 %
5. Precision = 86.21 %

Interpretation –

The Support Vector Machine model accurately predicts whether the person is diabetic or not with an accuracy of 78.67 %

4. Naïve Bayes Classifier

1. Assumption: Presence of conditional independence.
   The parameter Naïve Bayes Classifier have is **priors**. It calculates Prior probabilities of the classes. The priors are found according to the data.

```
NB.class_prior_
```

```
array([0.65517241, 0.34482759])
```

```
NB.epsilon_
```

```
1e-09
```

```
NB.sigma_
```

```
array([[0.83982754, 0.66217357, 0.82657412, 0.88559213, 0.68291884,
        1.04432948, 0.80022583, 0.97947108],
       [1.19702336, 0.92034094, 1.31827463, 1.20459438, 1.52060003,
        0.68153905, 1.29530618, 0.88259469]])
```

```
NB.theta_
```

```
array([[-0.13955096, -0.36186883, -0.04515459, -0.04816145, -0.12188342,
        -0.20618166, -0.1236651 , -0.16848329],
       [ 0.26514682,  0.68755078,  0.08579372,  0.09150675,  0.2315785 ,
         0.39174516,  0.23496369,  0.32011825]])
```

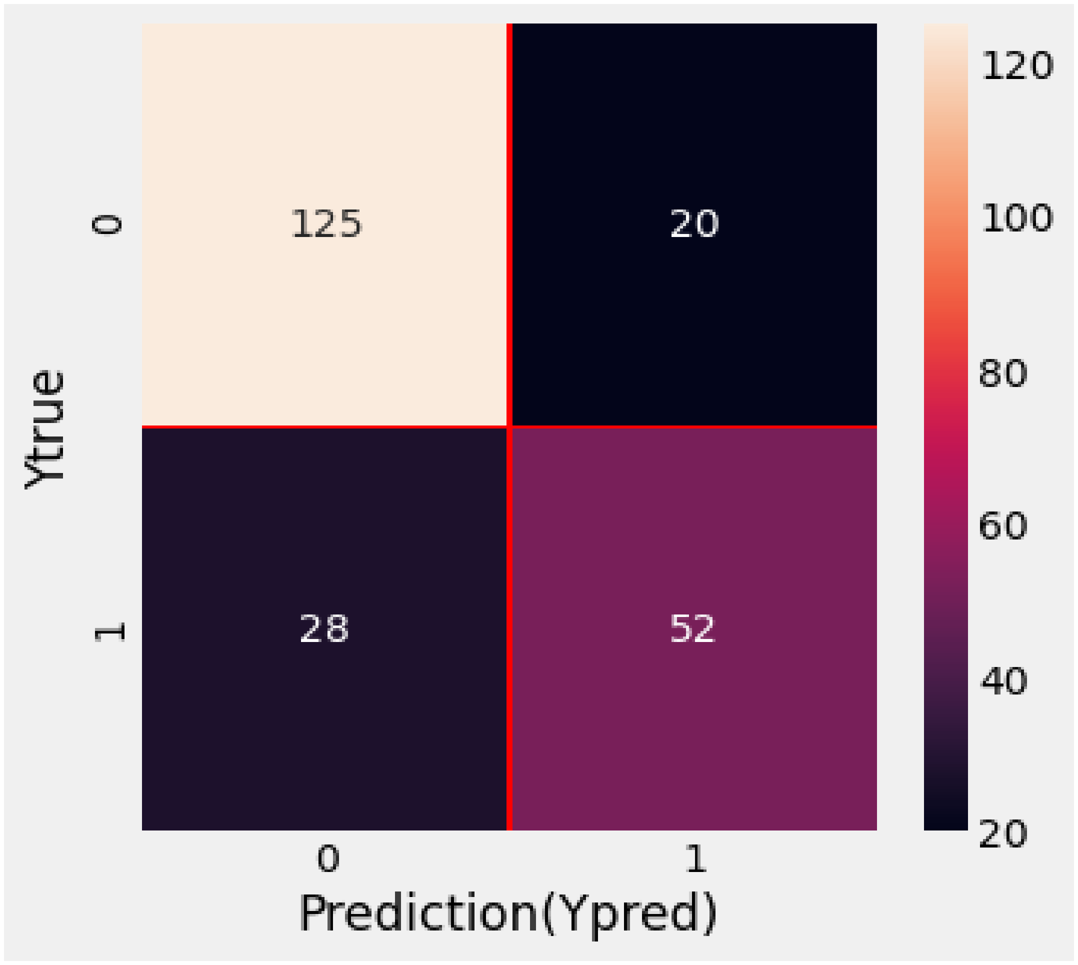
The class prior (0.65517241,0.34482759) indicates the class priors of each class ie. 0.65517241 indicates the class prior probability of non-diabetic person whereas 0.34482759 indicates the class prior probability of diabetic patient.

Epsilon denotes the absolute additive value of variances.

Sigma denotes the variance of each feature per class.

Theta denotes mean of each feature per class.

After tuning the parameter, the results of the confusion matrix are shown below:



The results obtained from the confusion matrix are as follows:

1. Accuracy = 74.67 %
2. Specificity = 63.86 %
3. Sensitivity = 80.99 %
4. F1-score = 80.14 %
5. Precision = 79.31 %

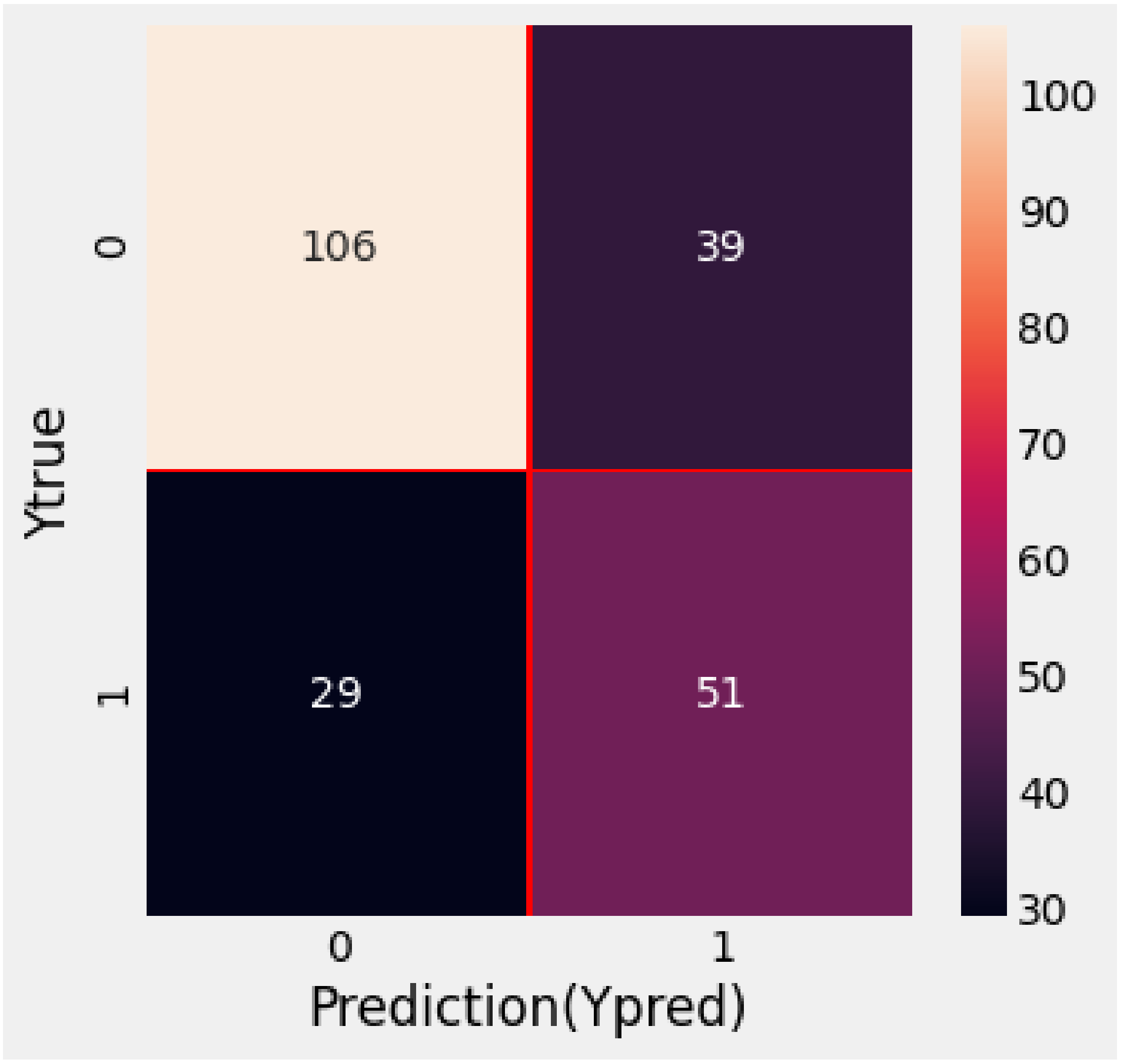
   The Naïve Bayes Classifier model accurately predicts whether the person is diabetic or not with an accuracy of 74.67 %

5. **Decision Tree**

Assumptions:

- Initially, whole training data is considered as root.
- Records are distributed recursively on the basis of the attribute value.

No parameters were tuned for this model. The results of the confusion matrix are given below:



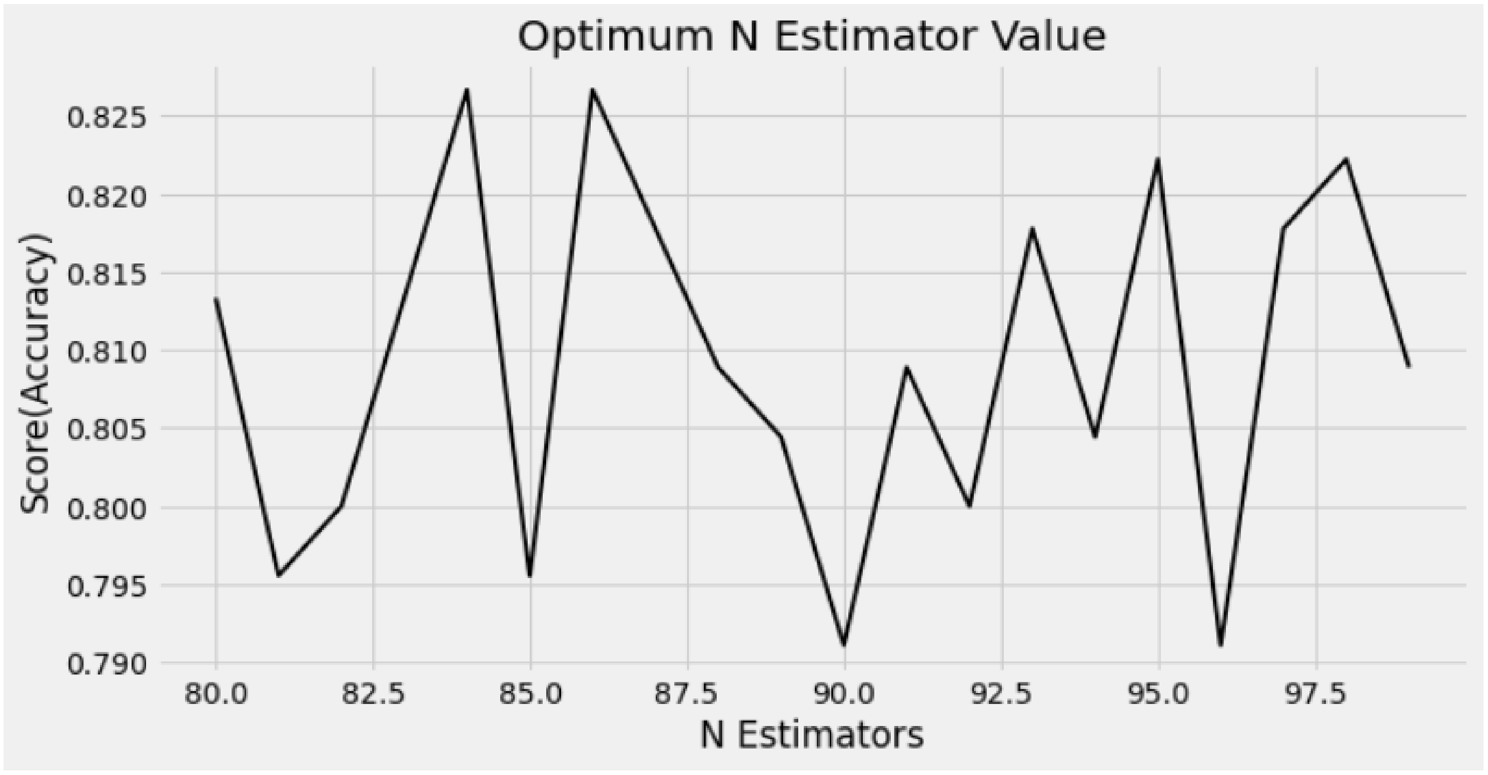
The results obtained from the confusion matrix are as follows:

1. Accuracy = 68 %
2. Specificity = 54.55 %
3. Sensitivity = 76.64 %
4. F1-score = 74.47 %
5. Precision = 72.41 %

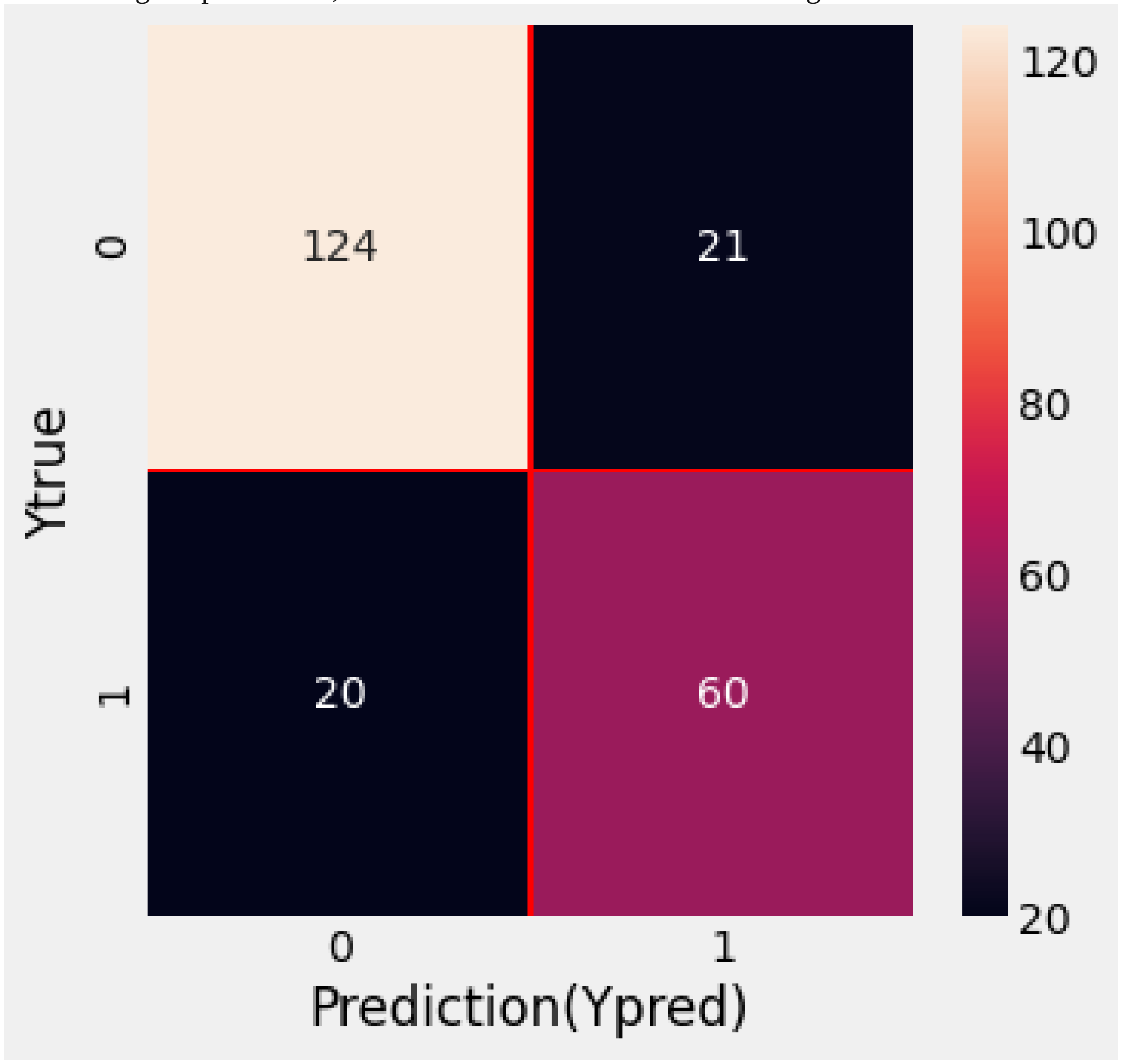
The Decision Tree model accurately predicts whether the person is diabetic or not with an accuracy of 68 %

Random Forest
Assumption of no formal distributions. Being a non-parametric model, it can handle skewed and multi-modal data.
The criterion used in the Random Forest is "**Gini Index**"
The parameters of the model have been tuned to find the optimum n estimator value. This gives the number of trees to build before taking the maximum voting or averages of predictions. Higher number of trees gives better performance but makes code slower.

```
#Find Optimum K value
scores = []
for each in range(80,100):
    RFfind = RandomForestClassifier(n_estimators = each)
    RFfind.fit(x_train,y_train)
    scores.append(RFfind.score(x_test,y_test))

plt.figure(1, figsize=(10, 5))
plt.plot(range(80,100),scores,color="black",linewidth=2)
plt.title("Optimum N Estimator Value")
plt.xlabel("N Estimators")
plt.ylabel("Score(Accuracy)")
plt.grid(True)
plt.show()
```



After tuning the parameters, the results of the confusion matrix are given below:



The results obtained from the confusion matrix are as follows:

1. Accuracy = 81.78 %
2. Specificity = 74.07 %
3. Sensitivity = 86.11 %
4. F1-score = 85.81 %
5. Precision = 85.52 %

   The Random Forest model accurately predicts whether the person is diabetic or not with an accuracy of 81.78 %

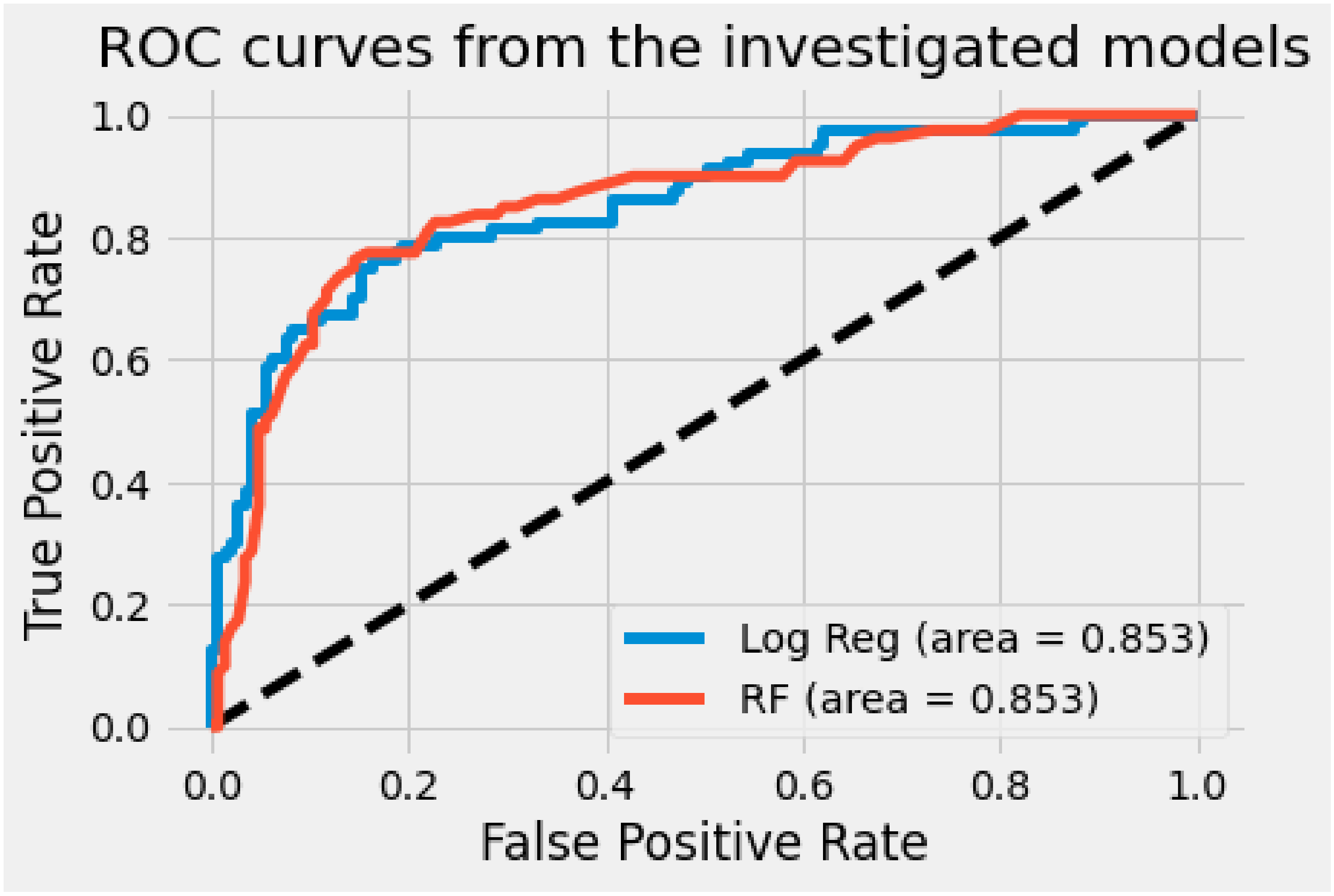
NOTE: **All the machine learning classification algorithms fitted have been cross validates with K-folds cross validation.**

Conclusion –

In this research paper the classifiers, Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Naïve Bayes, Decision Tree, Random Forest were used for classification of the imputed PIMA Indian Diabetes database. In all the classifiers, the parameters have been tuned, which insists that these models are not baseline models. Further the data is divided into training and testing datasets using the 70-30 ratio. The performance of the classification model depends upon the quality of the dataset and also the amount of data pre-processing is done behind it which helps to increase the accuracy of the data. After running all the classifiers, the results are given below: -

	Accuracy (in %)	Specificity (in %)	Sensitivity (in %)	F1-score (in %)	Precision (in %)
Logistic Regression	80.89	76.81	82.69	85.71	88.97
K-Nearest Neighbors	70.67	70.59	70.68	80.36	93.1
Support Vector Machine	78.67	72.22	81.70	83.89	86.21
Naïve Bayes Classifier	74.67	63.86	80.99	80.14	79.31
Decision Tree	68	54.55	76.64	74.47	72.41
Random Forest	81.78	74.07	86.11	85.81	85.52

After referring to the above table, the highest accuracy is of Random Forest as compared to all the other models. The other classification model which is close to random forest is logistic regression the difference of accuracies between the models is very less, also the Precision and Specificity is high for logistic regression as compared to Random Forest. So, random forest can't be just declared the best classification algorithm just on the basis of accuracy. Let's try to compare both the models on basis of AUC/ROC curve.



The area under curve for both the models is still the same (0.853). It can be concluded that as of now both the models are best for the prediction whether the person will be diabetic or not. Therefore, Random Forest and Logistic Regression are the best machine learning classification models which can predict whether person is diabetic or non-diabetic as compared to other classification models.

The performance, accuracy can be increased with Ensemble methods, and which model is better between Logistic Regression and Random Forest can be taken as future.

11.References

https://www.who.int/news-room/fact-sheets/detail/diabetes#:~:text=Overview,hormone%20that%20regulates%20blood%20sugar.
http://www.orsense.com/application.php?ID=6

https://www.academia.edu/43751015/Classification_of_Pima_Indian_Diabetes_Dataset_using_Ensemble_of_Decision_Tree_Logistic_Regression_and_Neural_Network

https://www.who.int/news/item/27-05-2021-new-wha-resolution-to-bring-much-needed-boost-to-diabetes-prevention-and-control-efforts

https://bncdf.org/diabetes/

https://www.who.int/news-room/fact-sheets/detail/diabetes

https://www.diabetesfreelife.org/diabetes

https://ikejabird.com/death-by-diabetes/

https://www.who.int/news-room/facts-in-pictures/detail/diabetes

https://www.who.int/news-room/fact-sheets/detail/diabetes#:~:text=In%202012%20(year%20of%20the,in%20premature%20mortality%20from%20diabetes.

https://www.ayurspace.com/blogs/articles/what-is-diabetes-ayurvedic-medicine-for-diabetes

https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes

https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes#:~:text=As%20of%202015%2C%2030.3%20million,adults%20are%20type%202%20diabetes.

https://ghanahealthnews.com/all-about-diabetes/

https://digiinewsnetwork.com/2020/08/16/diabetes-obesity-a-modern-day-health-problem/

https://www.kaggle.com/siddheshera/pima-diabetes-with-eda-12-models-beginner

https://www.lexjansen.com/wuss/2018/130_Final_Paper_PDF.pdf

https://medium.com/@hashinclude/knn-kth-nearest-neighbour-algorit-4b83c6fa31bf#:~:text=data%20is%20needed%20during%20the%20testing%20phase.

https://saravananthirumuruganathan.wordpress.com/tag/knn/

https://www.sciencedirect.com/science/article/pii/S0026265X1630666X

https://www.cnblogs.com/wintor12/p/3606666.html

https://saravananthirumuruganathan.wordpress.com/tag/bioinformatics/

https://www.ijirmps.org/papers/2019/6/615.pdf

https://plagiarismdetector.net/-https://ijssst.info/Vol-15/No-3/data/3857a513.pdf

https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

https://github.com/DivyaThakur24/DataAnalysisDiabetesFactor#:~:text=Fork%204-,This%20dataset%20is%20originally%20from%20the%20National%20Institute%20of%20Diabetes,measurements%20included%20in%20the%20dataset

https://scikit-learn.org/stable/auto_examples/neighbors/plot_lof_outlier_detection.html

https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/