Artificial Intelligence and Machine Learning

Project Report
Semester-IV (Batch-2022)

Title of the Project

Diabetes Prediction

Supervised By:

Faculty Name : Dr Jatin Arora

Submitted By:

Diksha → 2210990282

Divij → 2210990292

Diya → 2210990300

Yashika → 2210990977

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

Diabetes is one of the chronic diseases that causes blood sugar levels to rise. If diabetes is left untreated and undiagnosed, it can lead to complications. The time-consuming identification process leads to a patient's referral to a diagnostic Centre and consultation with a doctor. Predictive analytics in healthcare is a difficult challenge, but it can eventually assist physicians in making timely decisions about a patient's health and condition based on data. The emergence of machine learning methods solves this crucial issue.

The aim of this project is to create a model that can reliably predict the accuracy of diabetes in patients. To detect diabetes at an early stage, this project employs machine learning classification algorithms: Logistic Regression, SVM, Decision tree, Random Forest and Neural Networks are implemented. The dataset's purpose is to diagnose whether a patient has diabetes using diagnostic measures included in the dataset. Various measures like Precision, Accuracy, Specificity, and Recall are measured over classified instances using Confusion Matrix.

The accuracy of the algorithms used are compared and discussed. The study's comparison of the various machine learning techniques shows which algorithm is better suited for diabetes prediction. Using machine learning methods, this project aims to assist doctors and physicians in the early detection of diabete

1.INTRODUCTION

1.1 INTRODUCTION

Various classification strategies are used in the medical field for classifying data into different classes. Diabetes is a condition that affects the body's ability to produce the hormone insulin, which causes carbohydrate metabolism to become irregular and blood glucose levels to increase. High blood sugar is a common symptom of diabetes. If diabetes is not treated, it can lead to a variety of complications. Diabetic ketoacidosis and nonketotic hyperosmolar coma are two significant complications. Diabetes is considered a severe health problem in which the amount of sugar in the blood cannot be regulated. Diabetes is influenced by a variety of factors such as height, weight, genetic factors, and insulin, but the most important factor to remember is sugar concentration. The only way to avoid problems is to identify the problem early. This dataset comes from the Kaggle.

The dataset is divided into three sections, after which classification techniques are used. The training dataset is a sample of the dataset that is used to match the model. Validation Dataset, a dataset sample used for fine-tuning parameters and comparing model output accuracy and error rates between the training and validation datasets. Testing Dataset is a sample of a dataset that is used to assess the model's output.

Various machine learning techniques are implemented. Confusion matrix is obtained and is compared with all classification algorithms. This comparison of the various machine learning techniques shows which algorithm is better suited for diabetes prediction. Correlation between parameters and the best accuracy score using various supervised machine learning algorithms is obtained.

1.2 OBJECTIVES

- Since a decade, the number of people diagnosed with diabetes has risen significantly. The current human lifestyle is the primary cause of diabetes rise.
- Main objective of this project is to analyze the data, and see if it is possible to gleam any further information from the data to determine correlation between parameters and diabetes.
- The second is to attempt to get the best accuracy score using various supervised learning machine learning algorithms. To find out which algorithm is able to best predict whether a person has diabetes or not based on this dataset.
- The accuracy of the algorithms used are compared and discussed. The study's comparison of the various machine learning techniques shows which algorithm is better suited for diabetes prediction. Using machine learning methods, this project aims to assist doctors and physicians for predicting whether a person has diabetes or not.

1.3 MOTIVATION

The current human lifestyle is the primary cause of increasing diabetes. The three types of errors that may occur in today's medical diagnosis method:

1. The false-negative form, in which a patient is diabetic in fact but test results show that he or she does not have diabetes.

2. The false-positive type. In this type, a patient in reality is not a diabetic patient but test reports say that he/she is a diabetic patient.

3. The third type is an unclassifiable type in which a system cannot diagnose a given case. This happens because of insufficient knowledge extraction from past data, a given patient may get predicted in an unclassified type.

However, in fact, the patient must predict whether he or she will be diabetic or non-diabetic. Such diagnostic errors can result in unnecessary treatments or no treatments at all when they are needed. To prevent or mitigate the magnitude of such an effect, a machine learning algorithm must be used to build a framework that provides reliable results while reducing human effort.

1.4 OVERVIEW OF PROJECT

Machine learning has the great ability to revolutionize the diabetes risk prediction with the help of advanced computational methods and availability of a large amount of epidemiological and genetic diabetes risk dataset. Detection of diabetes in its early stages is the key for treatment. This work has described a machine learning approach to predicting diabetes or not. The technique may also help researchers to develop an accurate and effective tool that will reach at the table of clinicians to help them make better decisions about disease status.

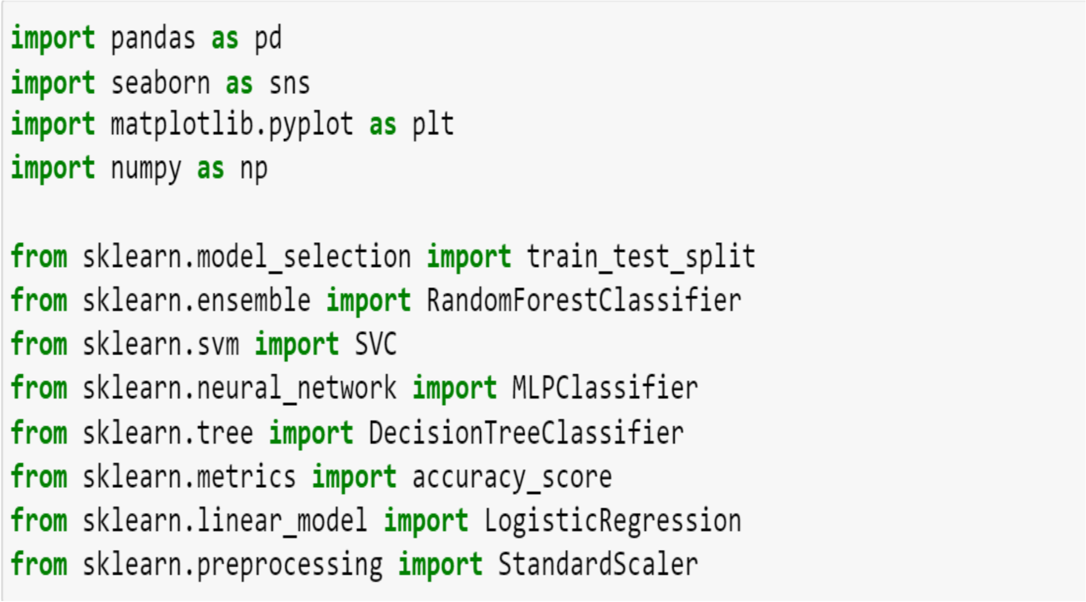
1.5 CHAPTERWISE SUMMARY

The first chapter is an introductory chapter, which gives an overview of the project. It includes four divisions - introduction, objectives, motivation, overview and chapter wise summary. The second chapter is data analysis, where the dataset is analyzed and studied for further classifications. Third chapter deals with the different machine learning models used. To detect diabetes at an early stage, this project employs machine learning classification algorithms: Logistic Regression, SVM, Decision tree, Random Forest and Neural networks are implemented. The last chapter gives an elaborate idea about the results of different models.

2.DATA ANALYSIS

2.1 STRUCTURE OF DATA

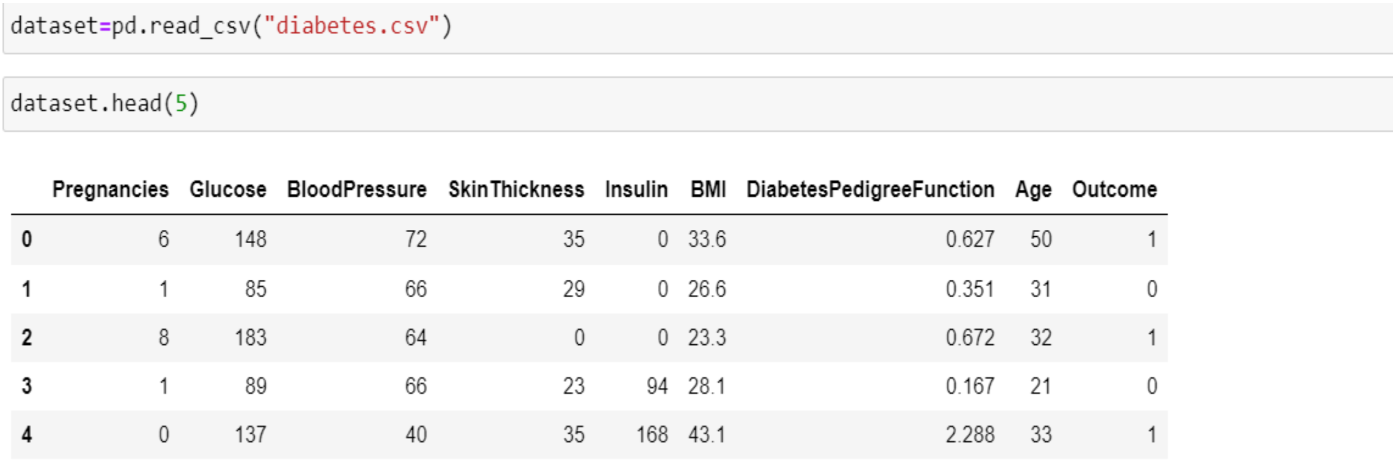
The dataset is originally from the Kaggle. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females. The datasets consist of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age etc.



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

Fig2.1.1 Importing Libraries

Fig2.1.1, Importing libraries to implement various machine learning for classification techniques.



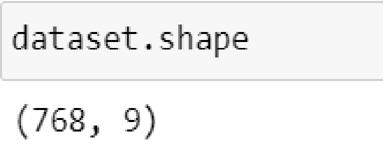
```
dataset=pd.read_csv("diabetes.csv")
```

```
dataset.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Fig2.1.2 Loading Dataset

Fig2.1.2, Loading the dataset to understand data structure.



```
dataset.shape
```

```
(768, 9)
```

Fig 2.1.3 Shape of dataset

Fig 2.1.3, represent total number of rows and columns in Dataset

2.2 PARAMETERS IMPLEMENTED

Pregnancies: No. of times pregnant

Glucose: Plasma glucose concentration for 2 hours in an oral glucose tolerance test.

Blood Pressure: Diastolic blood pressure (mm Hg). It is the bottom number in blood pressure tests, and is the pressure in the arteries when the heart rests between beats. A normal diastolic blood pressure is < 80 mm HG.

Skin Thickness: Triceps skin fold thickness (mm). Studies have been conducted, with conclusions that there are associations between people with thicker skin and diabetes.

Insulin: 2-Hour serum insulin (mu U/ml). Insulin is a hormone made by the pancreas that allows your body to use sugar (glucose) from carbohydrates in the food that you eat for energy or to store glucose for future use. A high insulin level is associated with diabetes.

BMI: Body mass index (weight in kg/ (height in m) ^2)

Range of BMI:

BMI < 18.5 - underweight

18.5 < BMI < 24.9 - ideal weight

25 < BMI < 29.9 - overweight

29.9 < BMI - obese

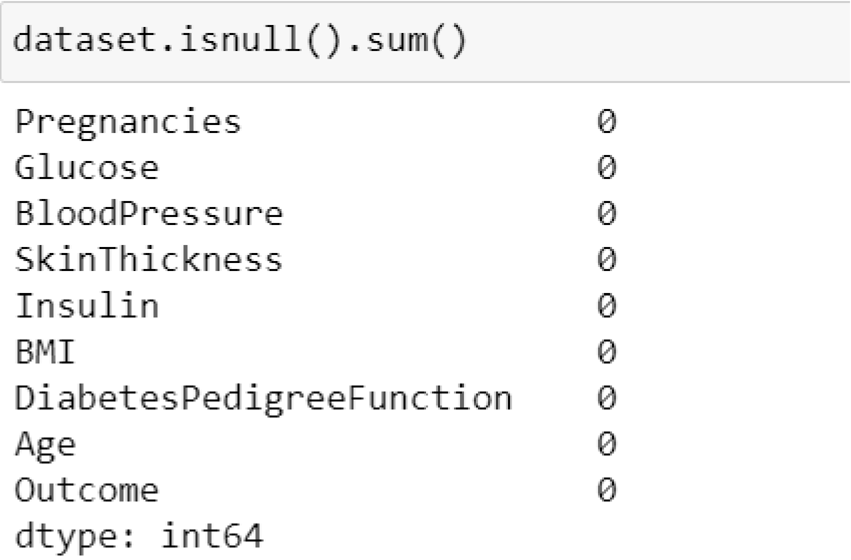
Diabetes Pedigree Function: It is a synthesis of the diabetes mellitus history in relatives and the genetic relationship of those relatives to the subject.

Results show that a person with a higher pedigree function tested positive and those who had a lower pedigree function tested negative.

Age: Age of the patient in years

Outcome: The target column which we are interested in finding out. 1 - diabetic, 0 - non-diabetic

2.3 EXPLORATORY DATA ANALYSIS

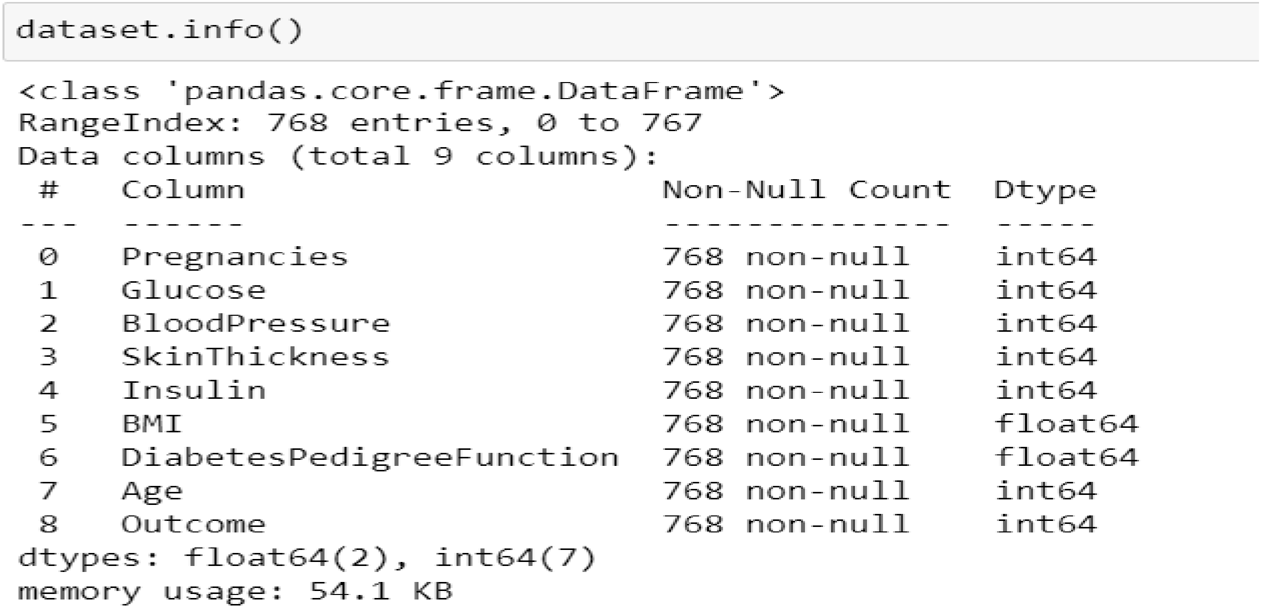


```
dataset.isnull().sum()
```

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

Fig 2.3.1 Exploratory Data Analysis

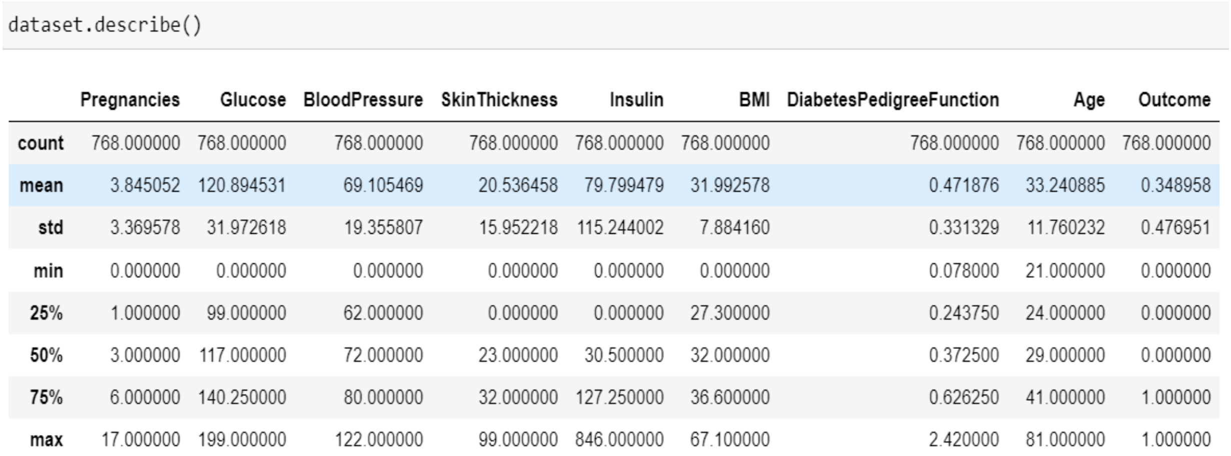
Fig 2.3.1, is analyzing the dataset and checking any missing values.



```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

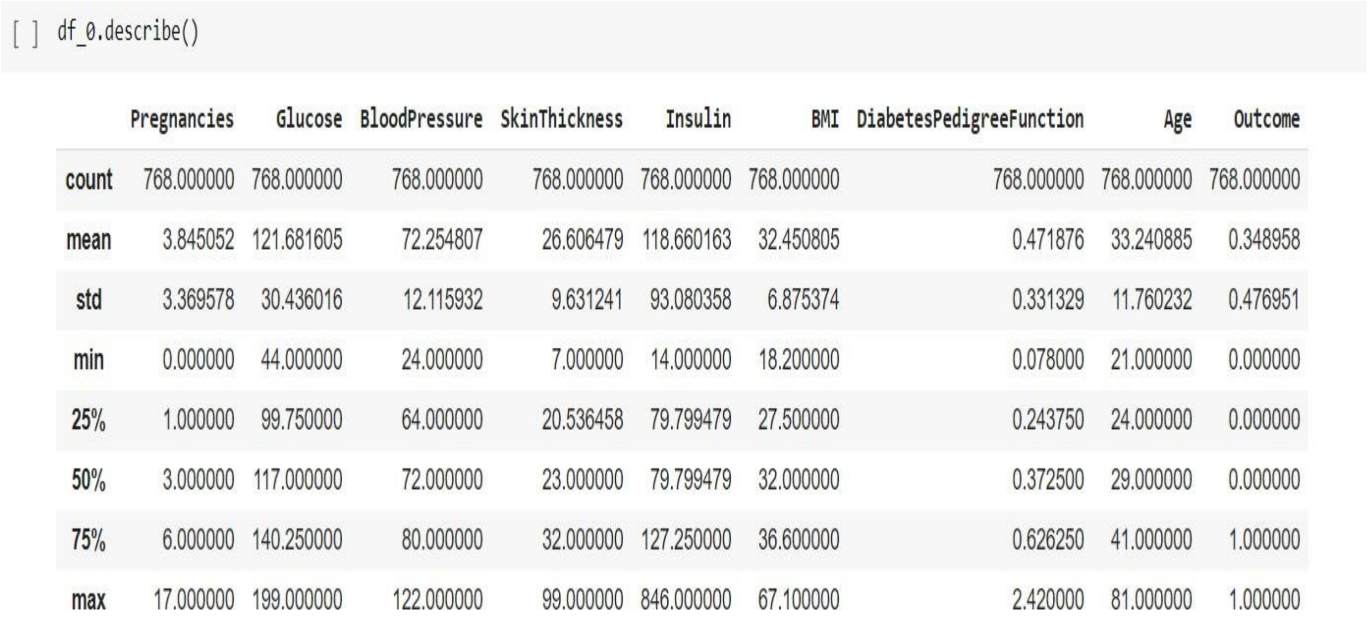
Fig 2.3.2 Dataset Information Fig 2.3.2 Dataset information's are checked.



```
dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 2.3.3 Calculating Mean, Count, Min, Max and Standard Deviation.



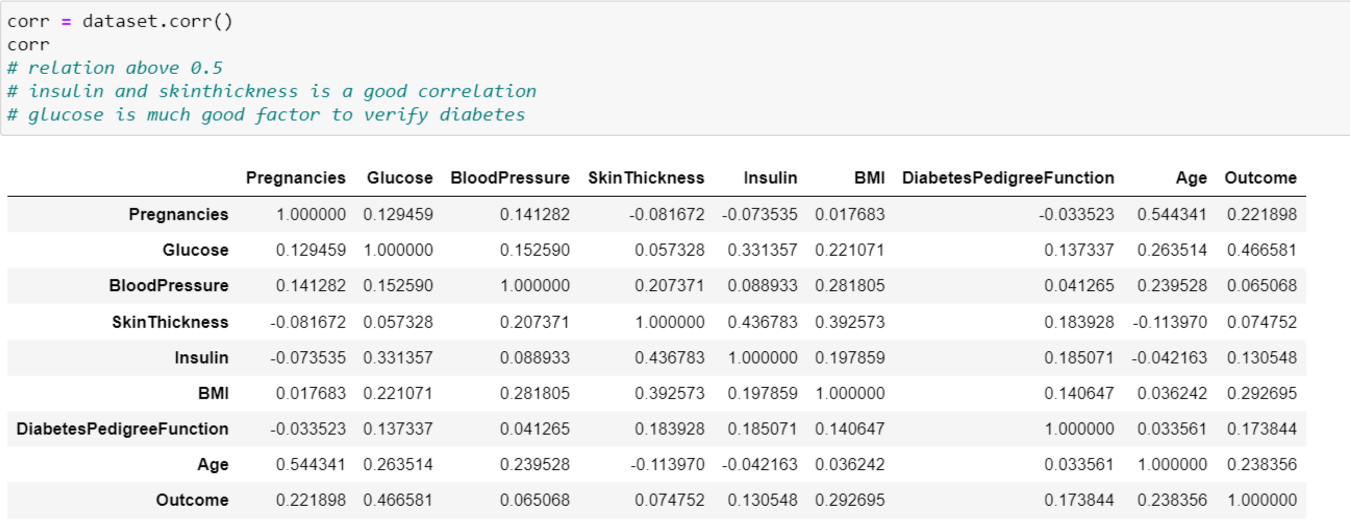
```
[ ]  df_0.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876	33.240885	0.348958
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329	11.760232	0.476951
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.000000	0.000000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Fig 2.3.5 Creating a copy of the original dataset and replace the 0 values of the impacted columns with the mean values

Now that the 0 values are accounted for, we can proceed with the rest of the Exploratory Data Analysis.

2.4 CORRELATION OF DATA



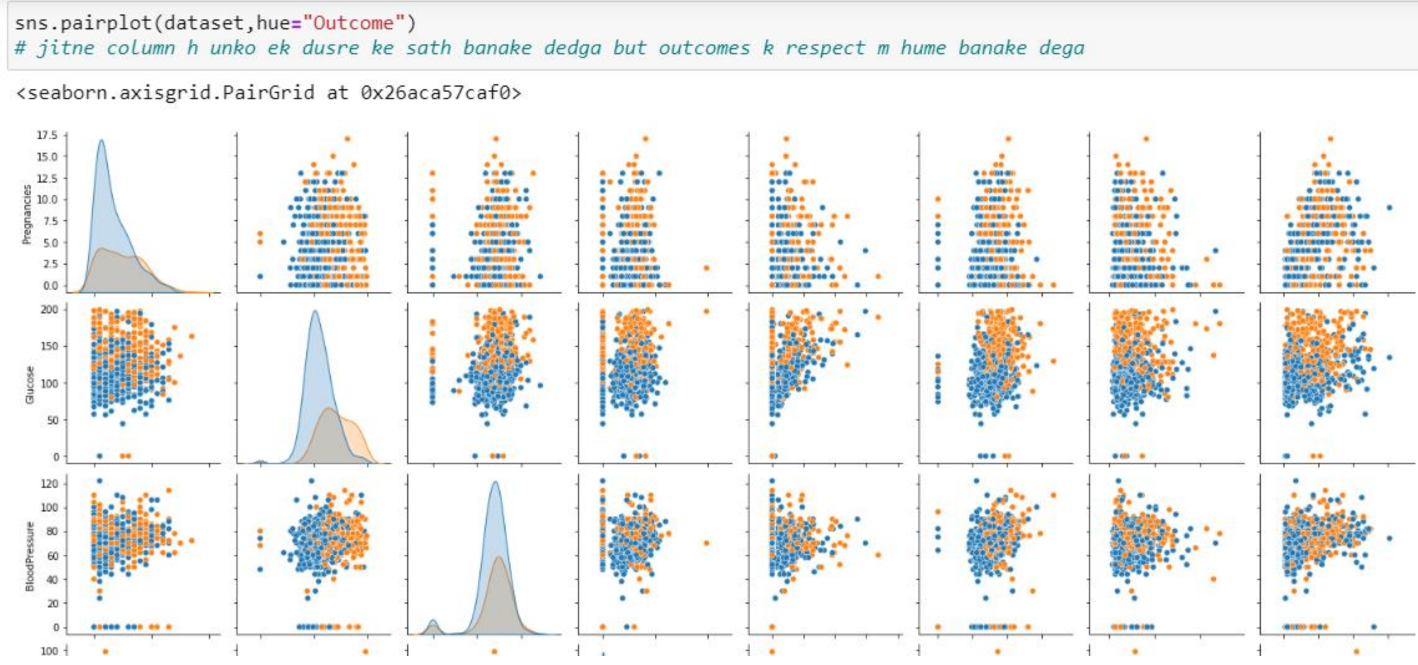
```
corr = dataset.corr()
corr
# relation above 0.5
# insulin and skinthickness is a good correlation
# glucose is much good factor to verify diabetes
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

Fig 2.5.1 Correlation of Data

The parameter with the highest positive correlation to each other is BMI and Skin Thickness. This is further confirmed by the SNS pair plot. The rest do not have strong multi-collinearity to each other.

2.5 SNS PAIR PLOT

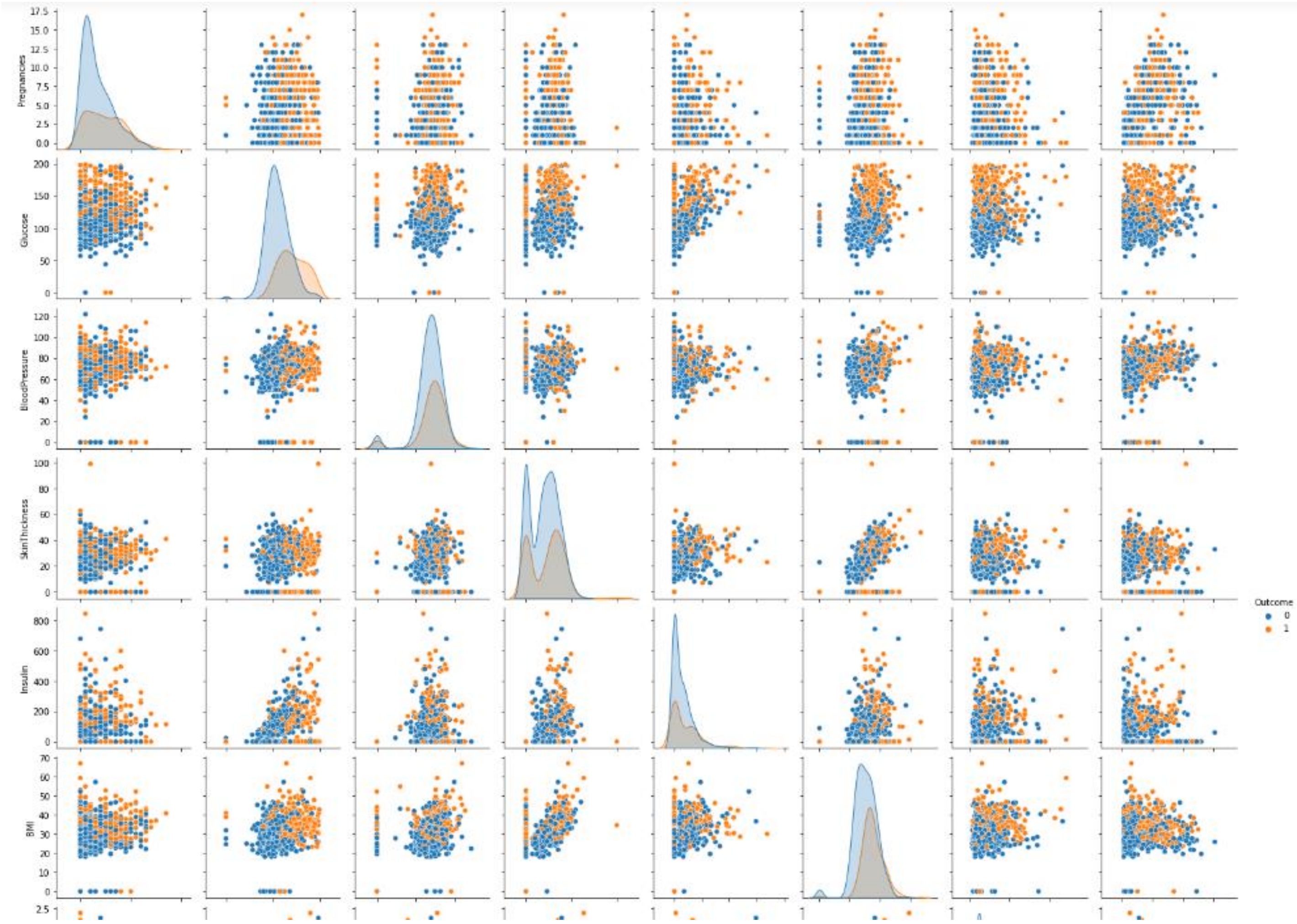


```
sns.pairplot(dataset,hue="Outcome")
# jitne column h unko ek dusre ke sath banake dedga but outcomes k respect m hume banake dega
```

```
<seaborn.axisgrid.PairGrid at 0x26aca57caf0>
```

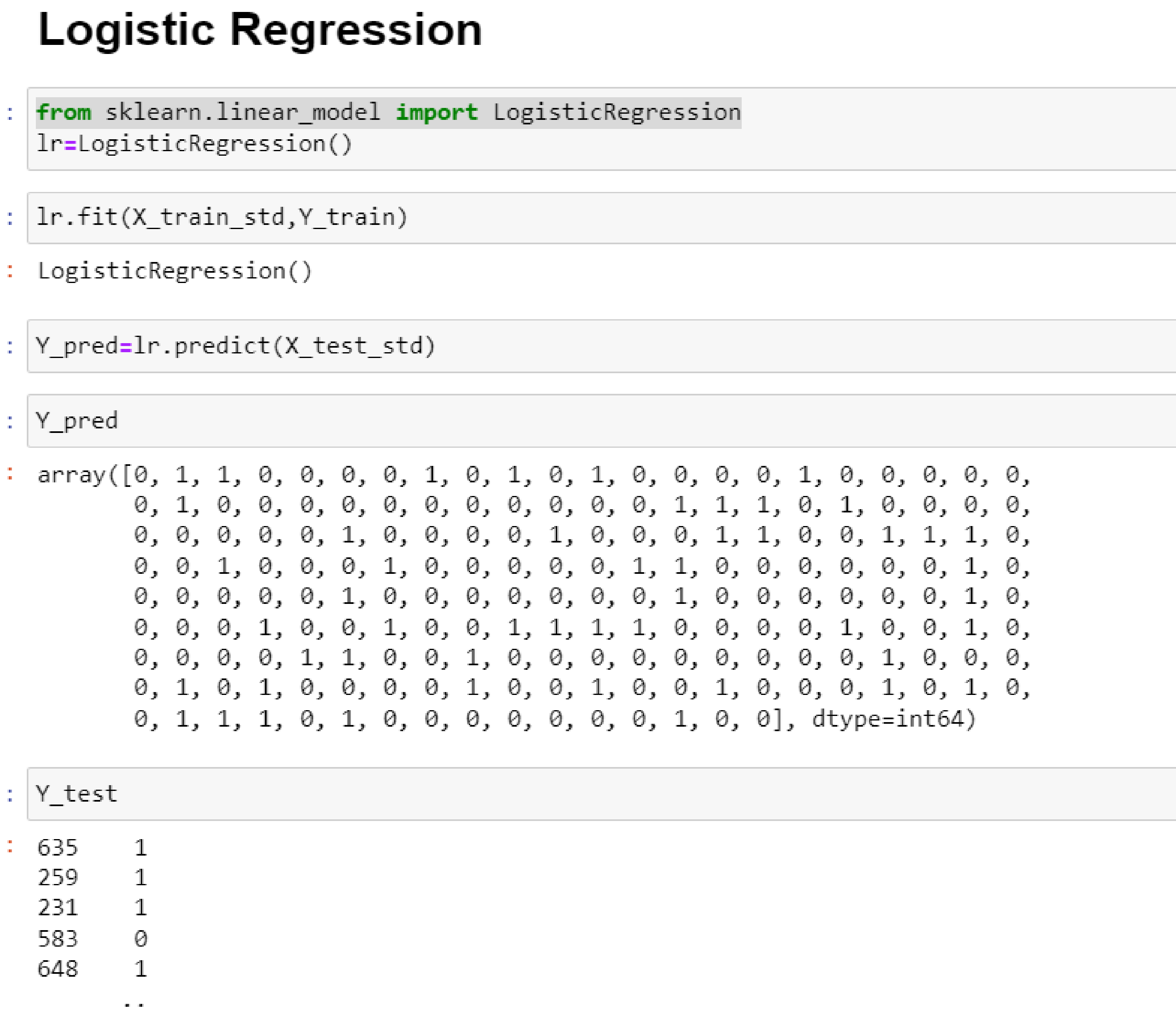
SNS Pair plot

From the plots, we can see that in histogram plot distribution, that most of the parameters are positively skewed, with outcome having a bimodal distribution, which is to be expected. Glucose and Blood Pressure are the only parameters which most resemble a normal distribution. Plot a pair plot to see which parameters might have a stronger correlation with either outcomes of diabetic patient and non-diabetic patient



Building a function for performing ML algos testing

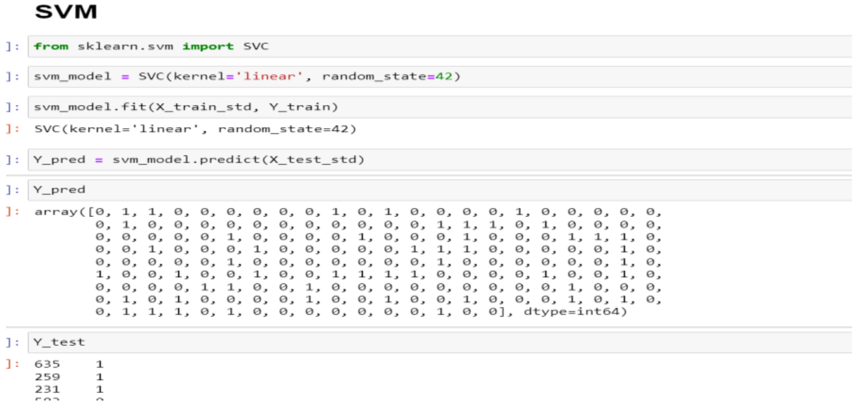
2.5.1 Logistic Regression Model



Logistic Regression

```
: from sklearn.linear_model import LogisticRegression
  lr=LogisticRegression()

: lr.fit(X_train_std,Y_train)

: LogisticRegression()

: Y_pred=lr.predict(X_test_std)

: Y_pred

: array([0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
         0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0,
         0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
         0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
         0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
         0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
         0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
         0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)

: Y_test

: 635    1
  259    1
  231    1
  583    0
  648    1
        ..
```

Support Vector Machine Model

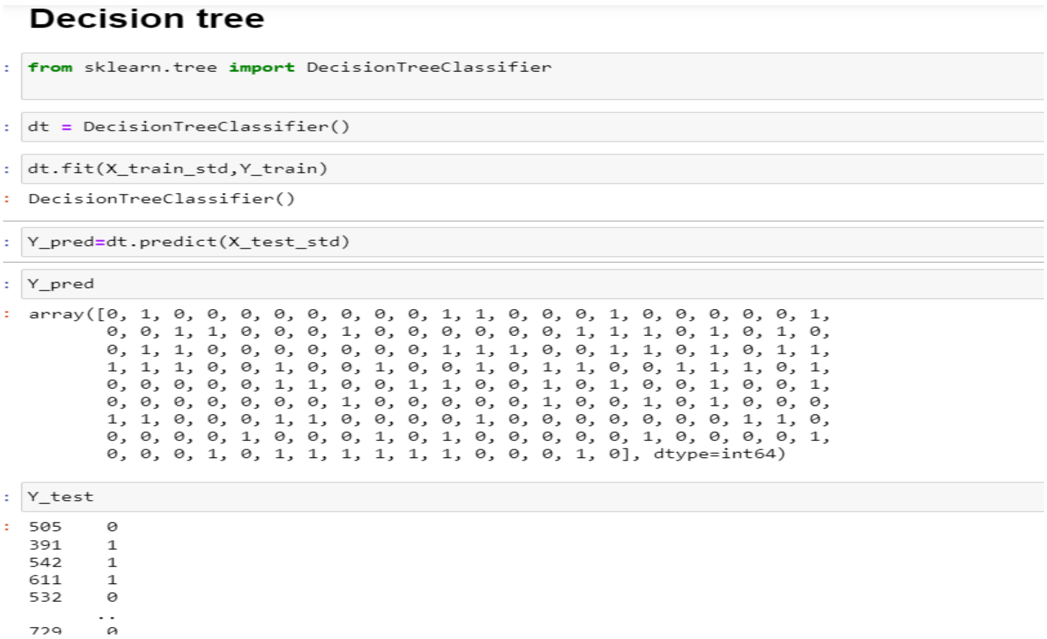


SVM

```
]: from sklearn.svm import SVC

]: svm_model = SVC(kernel='linear', random_state=42)

]: svm_model.fit(X_train_std, Y_train)
]: SVC(kernel='linear', random_state=42)

]: Y_pred = svm_model.predict(X_test_std)

]: Y_pred
]: array([0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
          0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
          0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
          0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
          1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
          0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
          0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
          0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)

]: Y_test
]: 635    1
   259    1
   231    1
```

Fig 3.4.4.1 Support Vector Machine Code

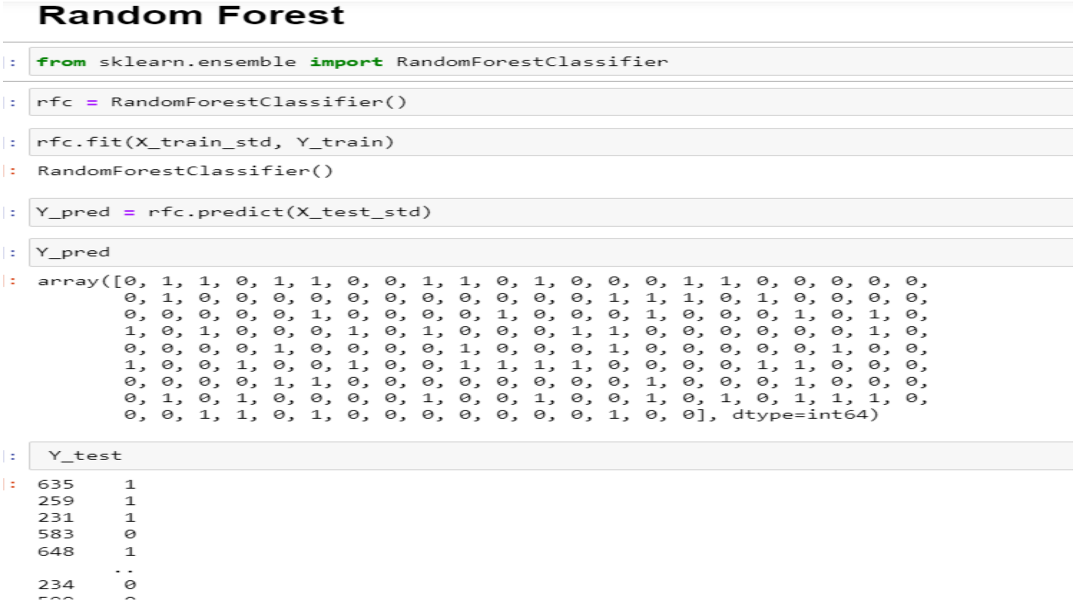
2.5.2 Decision Tree Model



Decision tree

```
: from sklearn.tree import DecisionTreeClassifier

: dt = DecisionTreeClassifier()

: dt.fit(X_train_std,Y_train)
: DecisionTreeClassifier()

: Y_pred=dt.predict(X_test_std)

: Y_pred
: array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
         0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0,
         0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
         1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
         0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
         0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
         1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
         0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
         0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0], dtype=int64)

: Y_test
: 505    0
  391    1
  542    1
  611    1
  532    0
        ..
  729    0
```

2.5.3 Random Forest Model



Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier()
```

```
rfc.fit(X_train_std, Y_train)
```

```
RandomForestClassifier()
```

```
Y_pred = rfc.predict(X_test_std)
```

```
Y_pred
```

```
array([0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

```
 Y_test
```

```
635    1
259    1
231    1
583    0
648    1
      ..
234    0
```

Fig 3.4.6.1 Random Forest Code

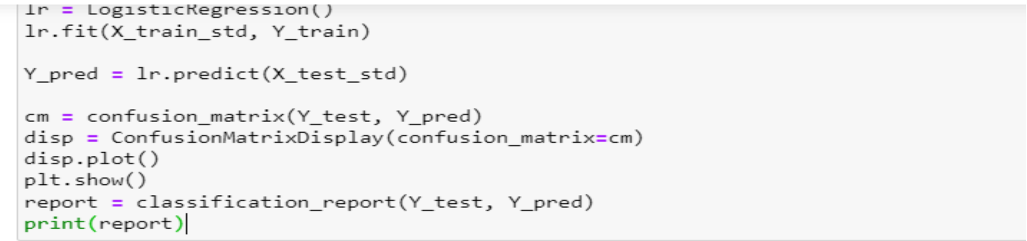
3. TEST RESULTS

4.

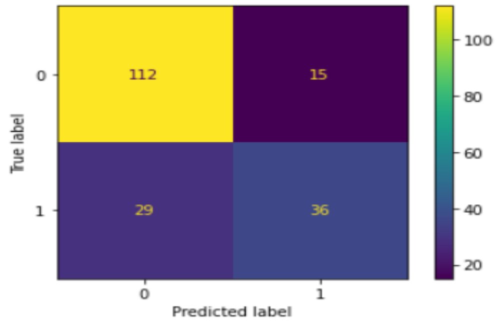
4.1 RESULTS

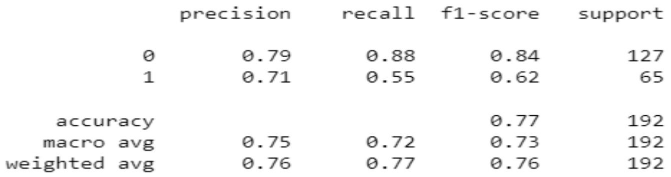
Logistic Regression Model

In the realm of logistic regression, the confusion matrix serves as a cornerstone for evaluating the performance of a classifier. This matrix offers a structured representation of the model's predictions against actual class labels, providing insights into the classification outcomes. Comprising four essential components—true positives, true negatives, false positives, and false negatives—the confusion matrix illuminates the classifier's ability to discern between classes. True positives signify correct predictions of the positive class, while true negatives denote accurate identification of the negative class. Conversely, false positives represent instances where the model wrongly classifies negatives as positives, and false negatives indicate misclassification of positives as negatives. By synthesizing these elements, the confusion matrix furnishes vital metrics such as accuracy, precision, recall, and F1 score, enabling a comprehensive assessment of the logistic regression model's efficacy in classification tasks.



```
lr = LogisticRegression()
lr.fit(X_train_std, Y_train)

Y_pred = lr.predict(X_test_std)

cm = confusion_matrix(Y_test, Y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
report = classification_report(Y_test, Y_pred)
print(report)
```

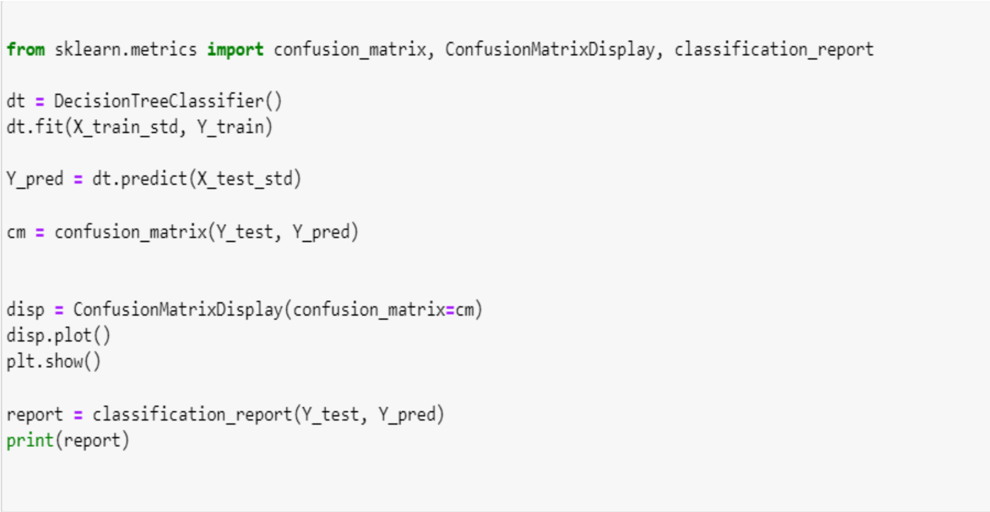




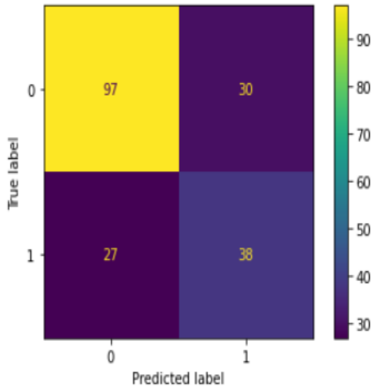
```
              precision    recall  f1-score   support

           0       0.79      0.88      0.84       127
           1       0.71      0.55      0.62        65

    accuracy                           0.77       192
   macro avg       0.75      0.72      0.73       192
weighted avg       0.76      0.77      0.76       192
```

2. Decision Tree

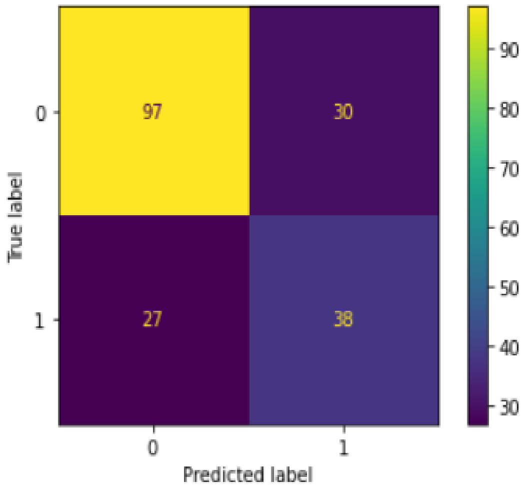
The confusion matrix of a decision tree provides a snapshot of its predictive performance by illustrating the distribution of actual and predicted classes. It's a tabular representation where rows correspond to the true classes and columns to the predicted classes. The four quadrants of the matrix represent different outcomes: true positives (correctly predicted positive instances), true negatives (correctly predicted negative instances), false positives (incorrectly predicted as positive when they are actually negative), and false negatives (incorrectly predicted as negative when they are actually positive). By analyzing this matrix, one can assess the accuracy, precision, recall, and F1-score of the decision tree model, enabling adjustments and optimizations to enhance its effectiveness in classification tasks.

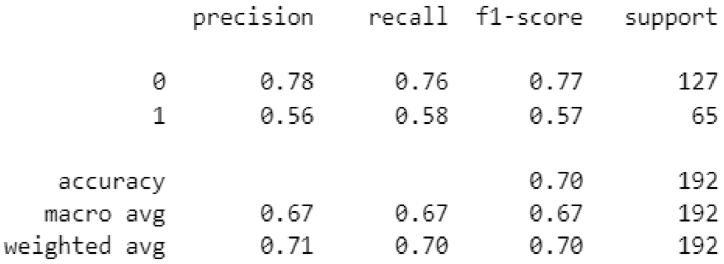


```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

dt = DecisionTreeClassifier()
dt.fit(X_train_std, Y_train)

Y_pred = dt.predict(X_test_std)

cm = confusion_matrix(Y_test, Y_pred)


disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()

report = classification_report(Y_test, Y_pred)
print(report)
```

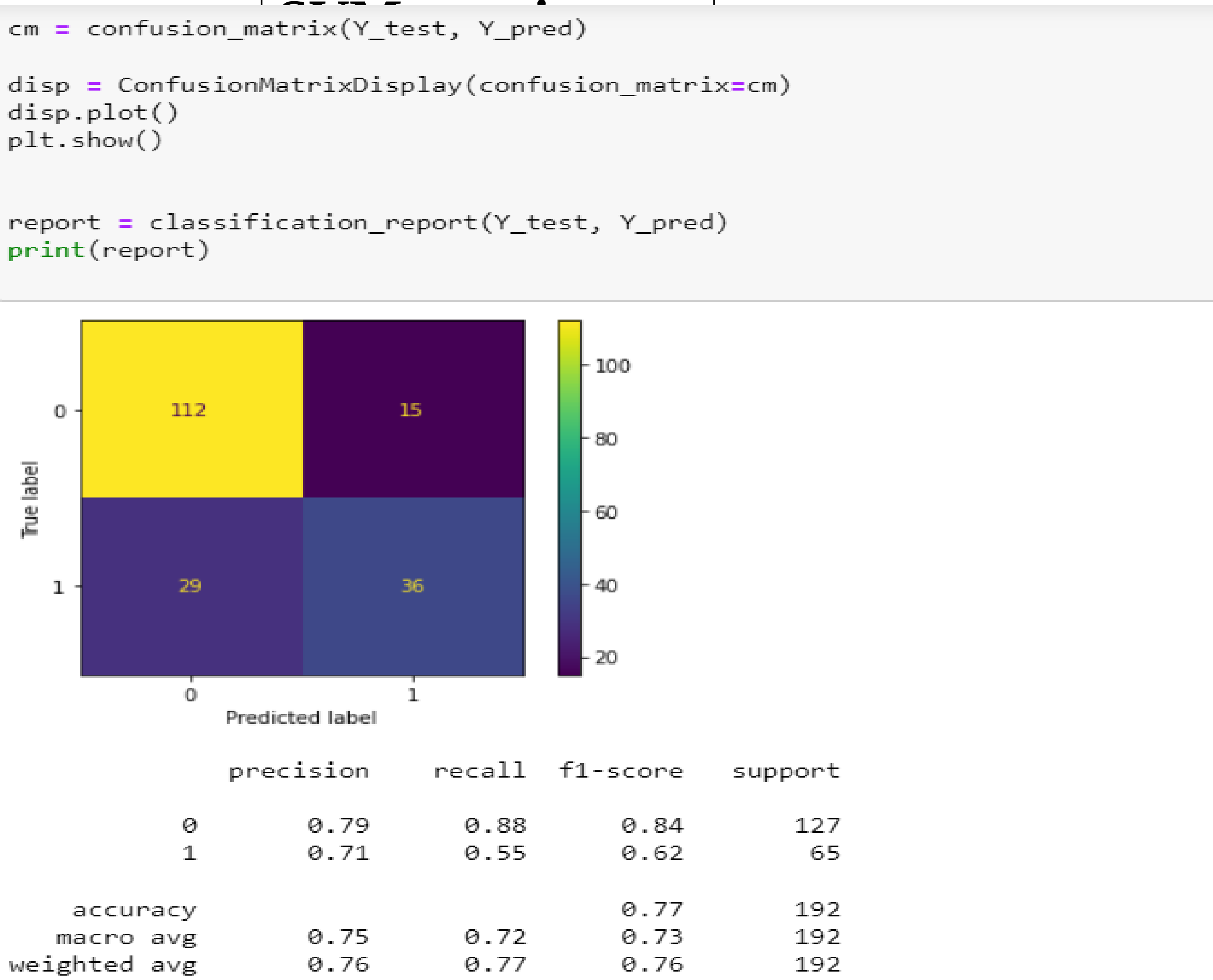


The Confusion Matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known . It shows the error in the model performance in the form of a matrix , hence also known as an error matrix.





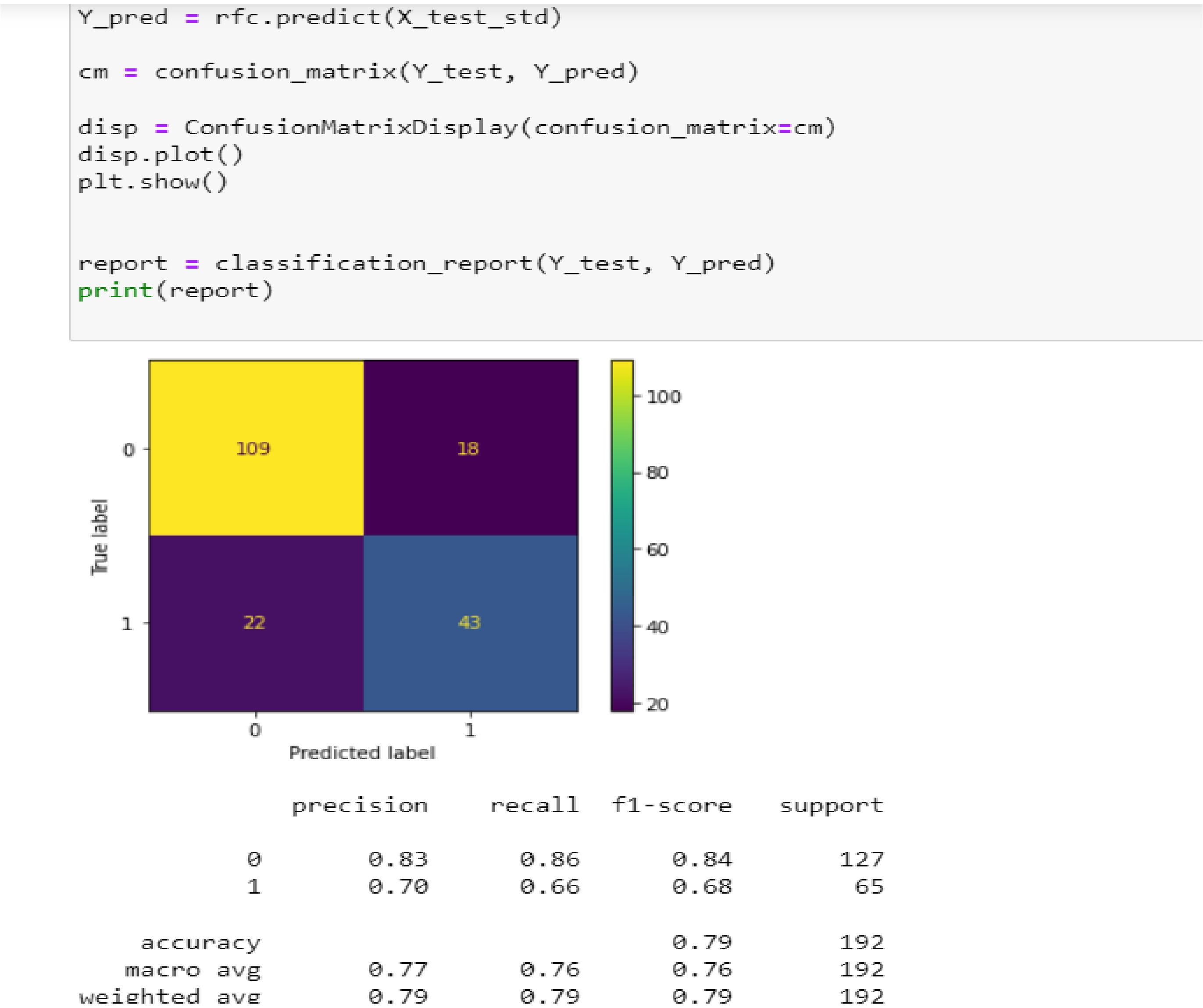
```
              precision    recall  f1-score   support

           0       0.78      0.76      0.77       127
           1       0.56      0.58      0.57        65

    accuracy                           0.70       192
   macro avg       0.67      0.67      0.67       192
weighted avg       0.71      0.70      0.70       192
```



```
cm = confusion_matrix(Y_test, Y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()


report = classification_report(Y_test, Y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.84       127
           1       0.71      0.55      0.62        65

    accuracy                           0.77       192
   macro avg       0.75      0.72      0.73       192
weighted avg       0.76      0.77      0.76       192
```

Random forest confusion matrix



```
Y_pred = rfc.predict(X_test_std)

cm = confusion_matrix(Y_test, Y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()


report = classification_report(Y_test, Y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       127
           1       0.70      0.66      0.68        65

    accuracy                           0.79       192
   macro avg       0.77      0.76      0.76       192
weighted avg       0.79      0.79      0.79       192
```

4.2 TEST REULTS ANALYSIS

Finally, we have trained our models, and summarized table of the metrics of the various models. Objectives were,

1) To attempt to see if it is possible to glean any further information from the data to determine correlation between parameters and diabetes.

2) To attempt to get the best accuracy score using various supervised learning machine learning algorithms.

For the first objective, based on the hypothesis test, we can tell that glucose levels are positively correlated to a person having diabetes, but we are not able to confirm if there is causality. For the second objective, based on the comparison between the various algorithms used, Random Forest seems to produce the best results to me.

The aim of this project is to create a model that can reliably predict the accuracy of diabetes in patients. The main aim of this project is to design and implement Diabetes Prediction Using Machine Learning Methods and Performance Analysis of that methods and it has been achieved successfully.

The proposed approach uses various classification and ensemble learning method in which SVM, Random Forest, Decision Tree, Logistic regression and Neural Networks classifiers are used. A machine learning algorithm must be used to build a framework that provides reliable results while reducing human effort.

The test accuracy of the various models is generally within the same range, from approximately 73% to 81%. Based on Accuracy and Recall score, overly the Random Forest Classifier produced the best results.

CONCLUSION AND FUTURE SCOPES

- Machine learning has the great ability to revolutionize the diabetes prediction with the help of advanced computational methods.

- Detection of Diabetes in its early stage is the key for treatment.

- The technique may also help researchers to develop an accurate and efficient tool that will reach at the table of clinicians to help them make better decisions about the disease.

- More parameters and factors would be involved in the future scope of this project.

- The accuracy will increase even more when the parameters increase.

- Using traditional techniques and algorithms, we can enhance the accuracy by improving the data.

REFERENCES

- ❑ https://www.kaggle.com/uciml/pima-indians-diabetes-database
- ❑ Diabetes Prediction using Machine Learning Algorithms - ScienceDirect
- ❑ Predicting Diabetes Mellitus With Machine Learning Techniques (nih.gov)
- ❑ Diabetes Prediction using Machine Learning Techniques – IJERT
- ❑ https://www.researchgate.net/publication/339543101_Diabetes_Prediction_using_Machine_Learning_Algorithms