A

Project Report

On

Diabetes prediction

Machine Learning(EIPY202C)

Master of Computer Applications(CNCS)

Submitted by

PHILIS FELISTAS MUTAMBA(23SCSE2150009)

Submitted to

Dr AMIT KUMAR

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GREATER NOIDA, UTTAR PRADESH
Winter 2023-2024

ACKNOWLEDGEMENT

I acknowledge the collaborative efforts involved in this project, which would not have been feasible without the generous support of various individuals and organizations. Special gratitude is extended to Dr. Amit Kumar, for his invaluable guidance and ongoing supervision, as well as for providing essential project-related information. I am also grateful to , Head of the Department of Computer Science and Engineering and, Project Coordinator, for their insightful suggestions and unwavering support throughout the project duration. Additionally, I extend my thanks to the faculty and staff members of the Department of Computer Science and Engineering, for their assistance in timely project completion. My appreciation goes out to friends who contributed to project development and to all those who lent their expertise willingly.

ABSTRACT

Diabetes is a prevalent ailment globally, but early detection offers a chance to halt its progression and mitigate complications. Our study focuses on constructing a predictive model to anticipate diabetes development based on diagnostic metrics from datasets sourced from kaggle. The aim of this project is to create a model that can reliably predict the accuracy of diabetes in patients. Dataset splits into four 3 types of dataset. Then classification techniques are implemented. Training Dataset, Dataset sample that is used to fit the model. Validation Dataset, Dataset sample that is used for hyper tuning the parameters, and comparing the accuracy and error rates of the model performance between using the training dataset and the validation dataset. Testing Dataset, Dataset sample that is used to test the model performance (predictive power). To detect diabetes at an early stage, this project employs machine learning classification algorithms: Gaussian Naive Bayes, K Neighbours, SVM, Decision tree and Gradient Boosting Classifier are implemented. The Pima Indians Diabetes Database (PIDD) is used in the experiments. The National Institute of Diabetes and Digestive and Kidney Diseases provided the results. 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 study compares and discusses the accuracy of different algorithms employed. By examining various machine learning techniques, the research determines which algorithm is most effective for predicting diabetes. Through the utilization of machine learning methods, the project seeks to aid doctors and physicians in identifying diabetes early on.

CONTENTS

1. INTRODUCTION

Different classification strategies are used in the medical field to classify data into different classes. Diabetes is a chronic health condition characterized by high blood sugar levels.It occurs when the body either doesn't produce enough insulin or can't effectively use the insulin it produces.High blood sugar is a common symptom of diabetes. If diabetes is not treated at early stages, it can cause serious consequences on health. 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 pregnancies, glucose, blood pressure, skin thickness, BMI, DiabetesPedigreeFunction, age, 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 'National Institute of Diabetes and Digestive Diseases' Pima Indians Diabetes Database (PIDD). Several constraints were taken from the massive database. 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. The testing dataset serves as a sample used to evaluate the model's performance, employing a range of machine learning techniques. A confusion matrix is generated and compared across all classification algorithms to determine the most suitable algorithm for diabetes prediction. Additionally, the correlation between parameters is analyzed, and the highest accuracy score achieved using different supervised machine learning algorithms is identified.

2.BACKGROUND

Diabetes mellitus, commonly referred to as diabetes, is a chronic metabolic disorder characterized by high blood glucose levels (hyperglycemia) due to inadequate insulin production, impaired insulin function, or both. With an increasing global prevalence, diabetes has become a significant public health concern affecting millions of people worldwide. According to the World Health Organization (WHO), an estimated 422 million adults were living with diabetes in 2014, and the number is expected to rise to 642 million by 2040 if current trends continue.

The implications of diabetes are profound, as it significantly increases the risk of developing serious health complications such as cardiovascular disease, kidney failure, blindness, and lower limb amputations. However, timely diagnosis and effective management can help prevent or delay these complications, highlighting the importance of early detection and intervention.

In recent years, advancements in machine learning and artificial intelligence have opened new avenues for predicting and managing chronic diseases like diabetes. Supervised learning, a branch of machine learning, offers promising opportunities for developing predictive models that can analyze diagnostic data and accurately forecast an individual's risk of developing diabetes.

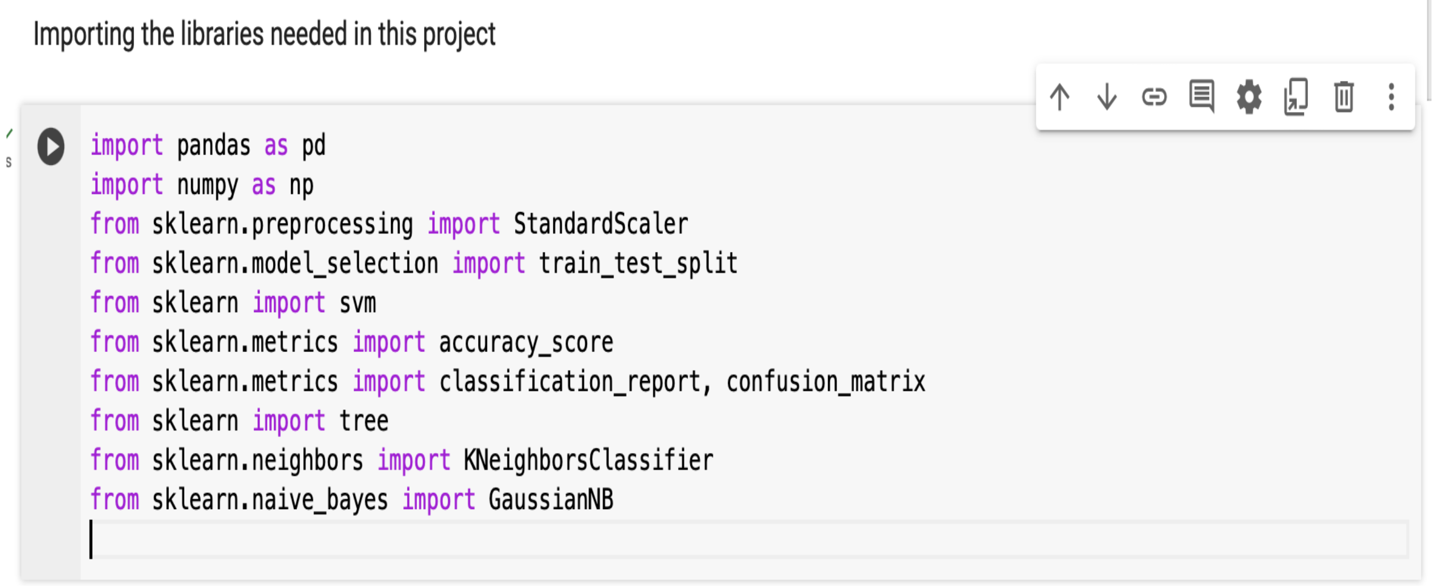
The primary objective of this project is to leverage supervised learning techniques to construct a predictive model capable of identifying individuals at high risk of developing diabetes. By utilizing a dataset containing relevant diagnostic measures, such as blood glucose levels, insulin sensitivity, and body mass index (BMI), we aim to train and evaluate our model's performance. Moreover, by employing the same dataset for both training and testing purposes through a data-splitting approach, we aim to ensure robustness and generalizability of our model.

Through this project, we seek to contribute to the ongoing efforts in diabetes research and healthcare by providing a reliable tool for early detection and intervention, ultimately improving patient outcomes and reducing the burden of diabetes-related complications.

3. METHODOLOGY AND RESULTS

3.1STRUCTURE OF DATA

The dataset originates from the National Institute of Diabetes and Digestive and Kidney Diseases and aims to predict whether a patient has diabetes based on specific diagnostic measurements. The dataset was derived from a larger database, with constraints ensuring that all patients are females of Pima Indian heritage aged at least 21 years old. It includes various medical predictor variables such as the number of pregnancies, BMI, insulin level, and age, along with one target variable, Outcome, indicating the presence or absence of diabetes.

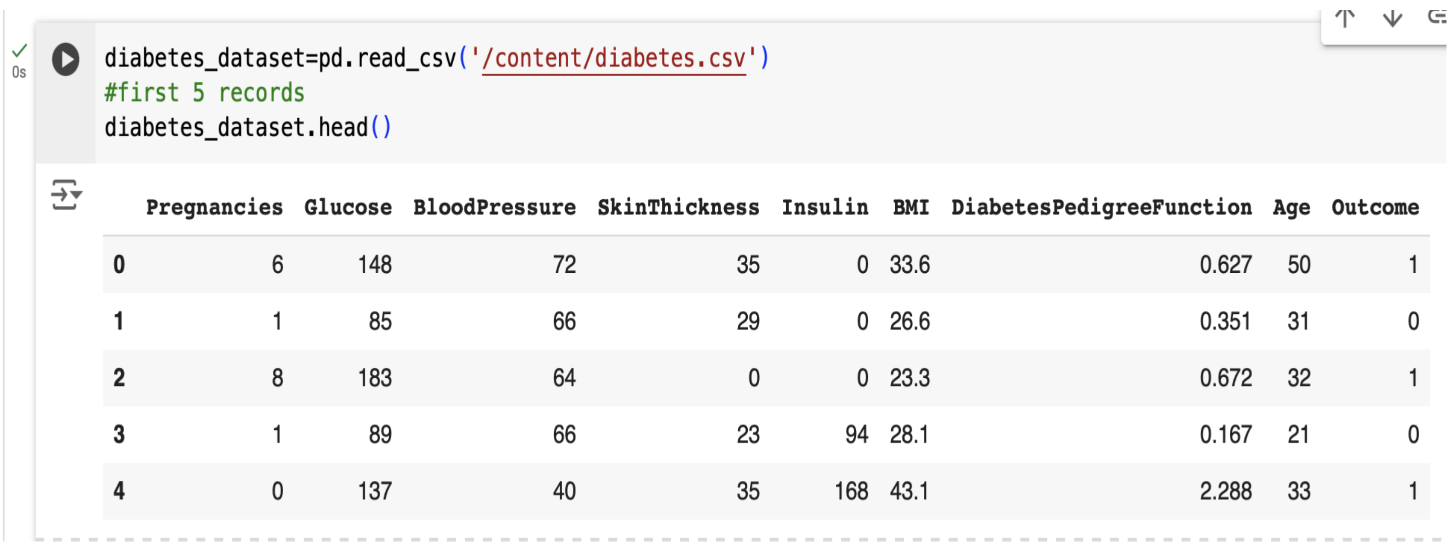


Importing the libraries needed in this project

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
```

Fig3.1.1 Importing Libraries

Fig3.1.1, Importing libraries to implement various machine learning for classification techniques

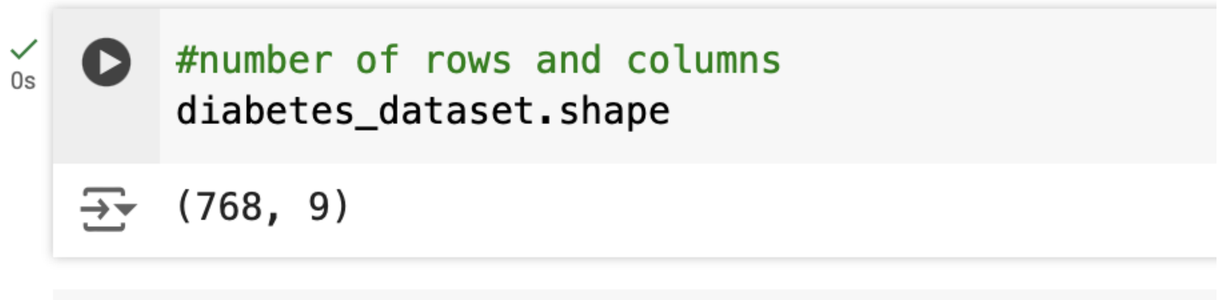


```
diabetes_dataset=pd.read_csv('/content/diabetes.csv')
#first 5 records
diabetes_dataset.head()
```

```
   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
```

Fig3.1.2 Loading Dataset

Fig3.1.2, Loading the dataset to understand data structure.



```
#number of rows and columns
diabetes_dataset.shape
```

```
(768, 9)
```

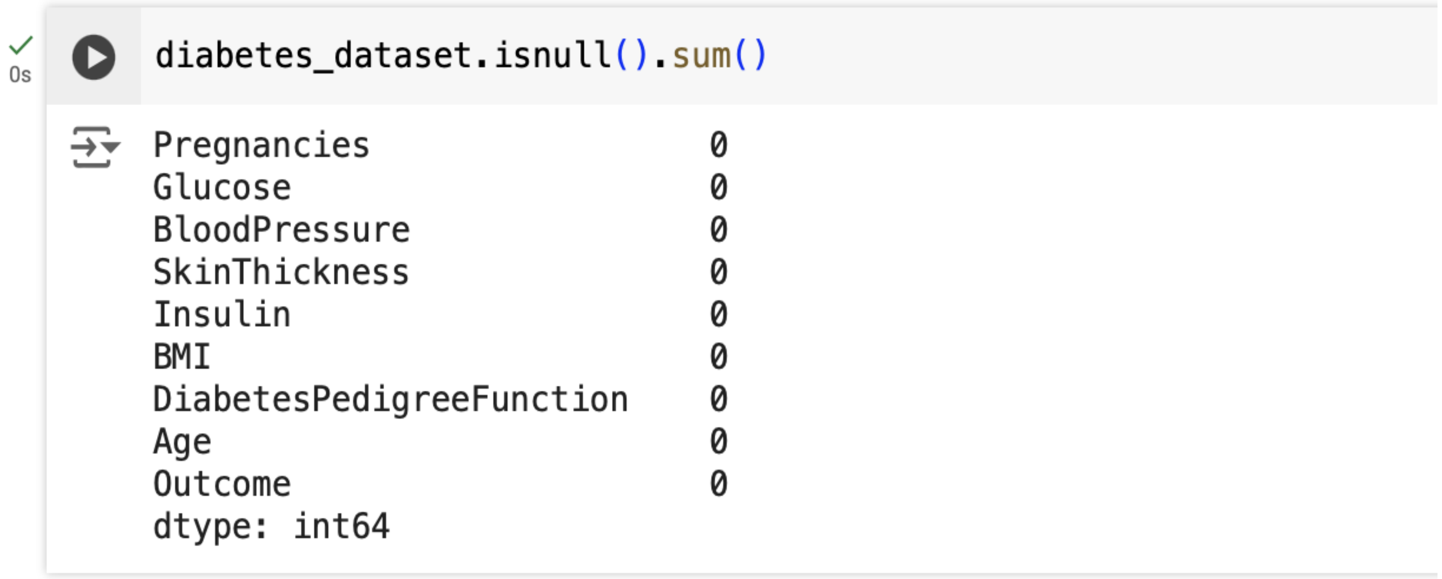
Fig 3.1.3 Shape of dataset

Fig 3.1.3, represent total number of rows and columns in Dataset

3.2 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 11 | Mini Project 2020-2021 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 variable which will be used in finding out. 1 - diabetic, 0 - non-diabetic

3.3 EXPLORATORY DATA ANALYSIS

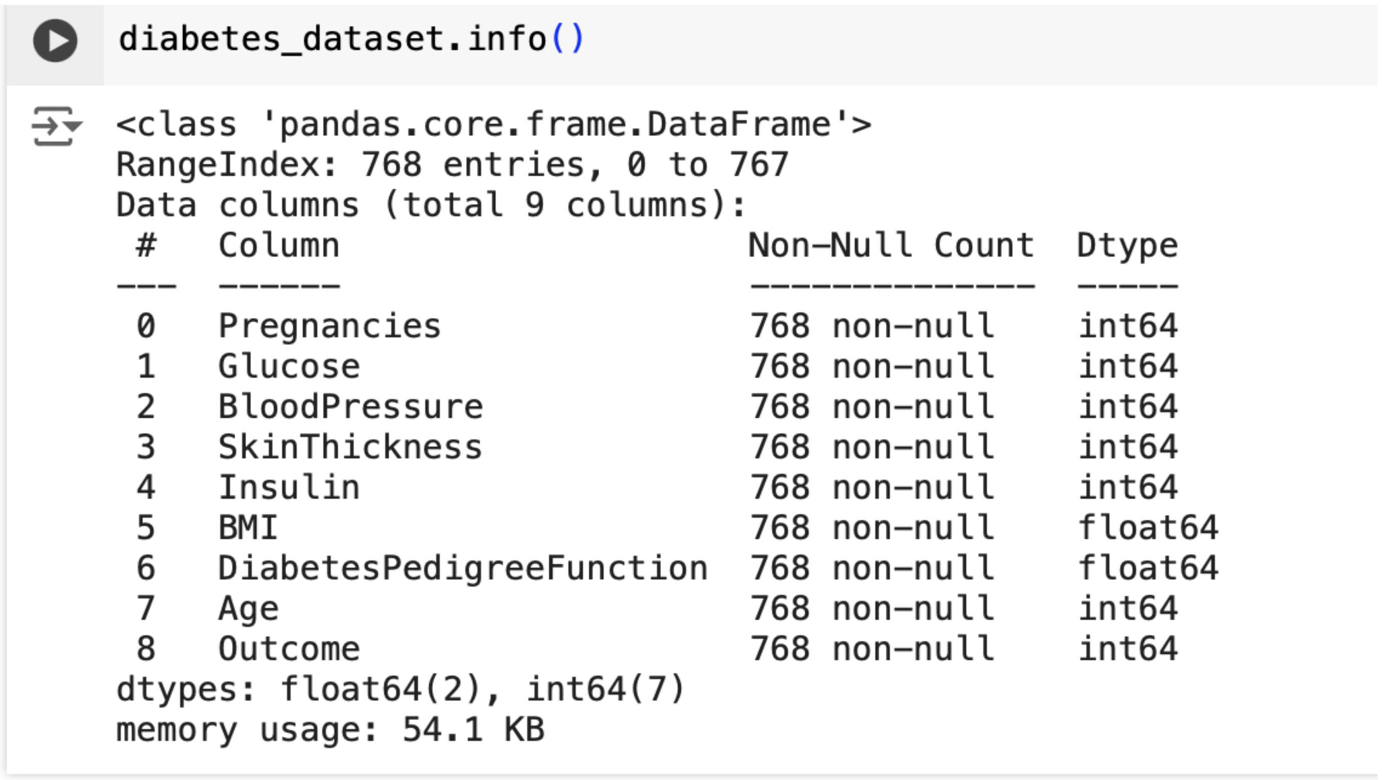


```
diabetes_dataset.isnull().sum()
```

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

Fig 3.3.1 Exploratory Data Analysis

Fig 3.3.1, is checking the dataset and checking any missing values.

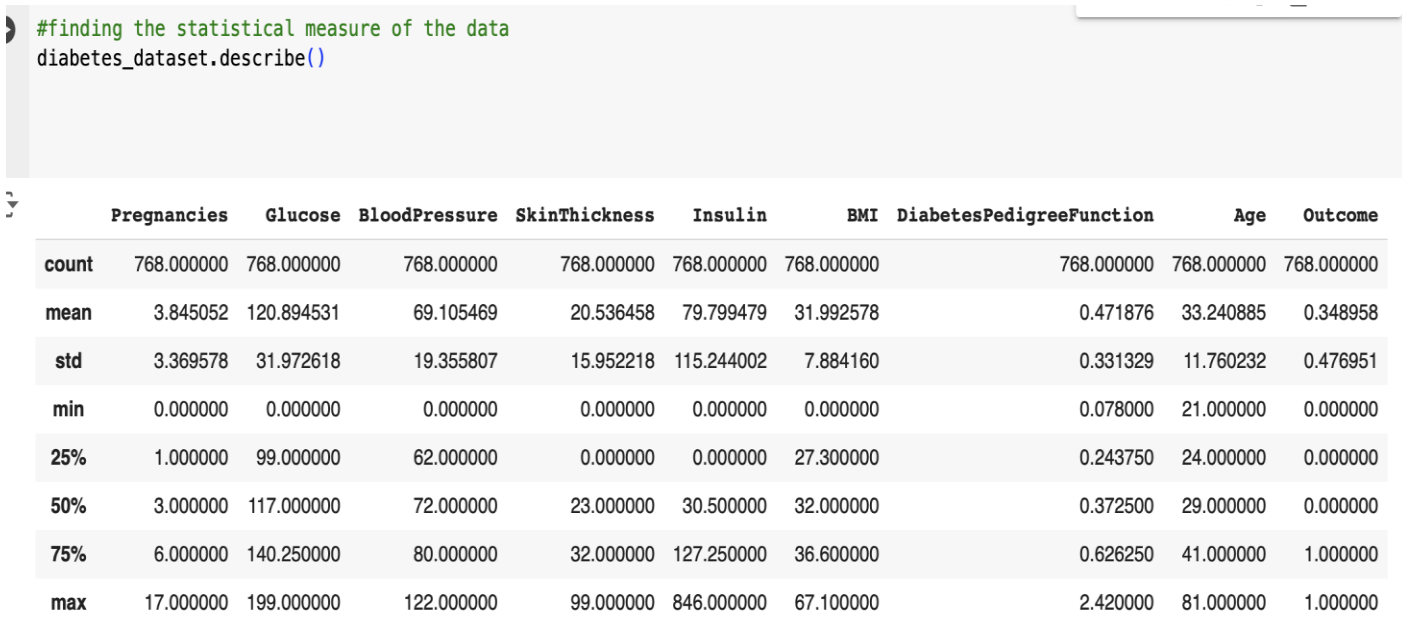


```
diabetes_dataset.info()
```

```
<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 3.3.2 Dataset Information

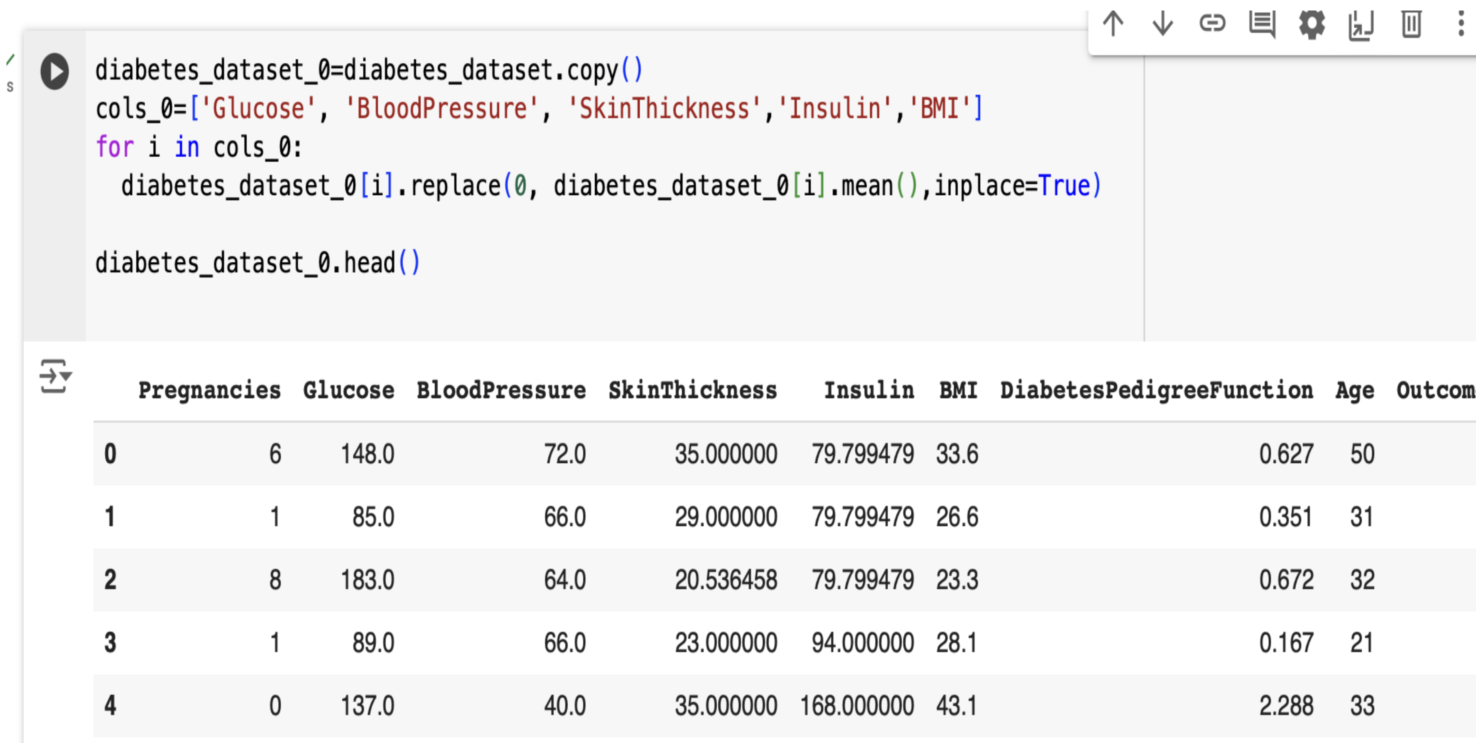
Fig 3.3.2 Dataset information's are checked.



```
#finding the statistical measure of the data
diabetes_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
```

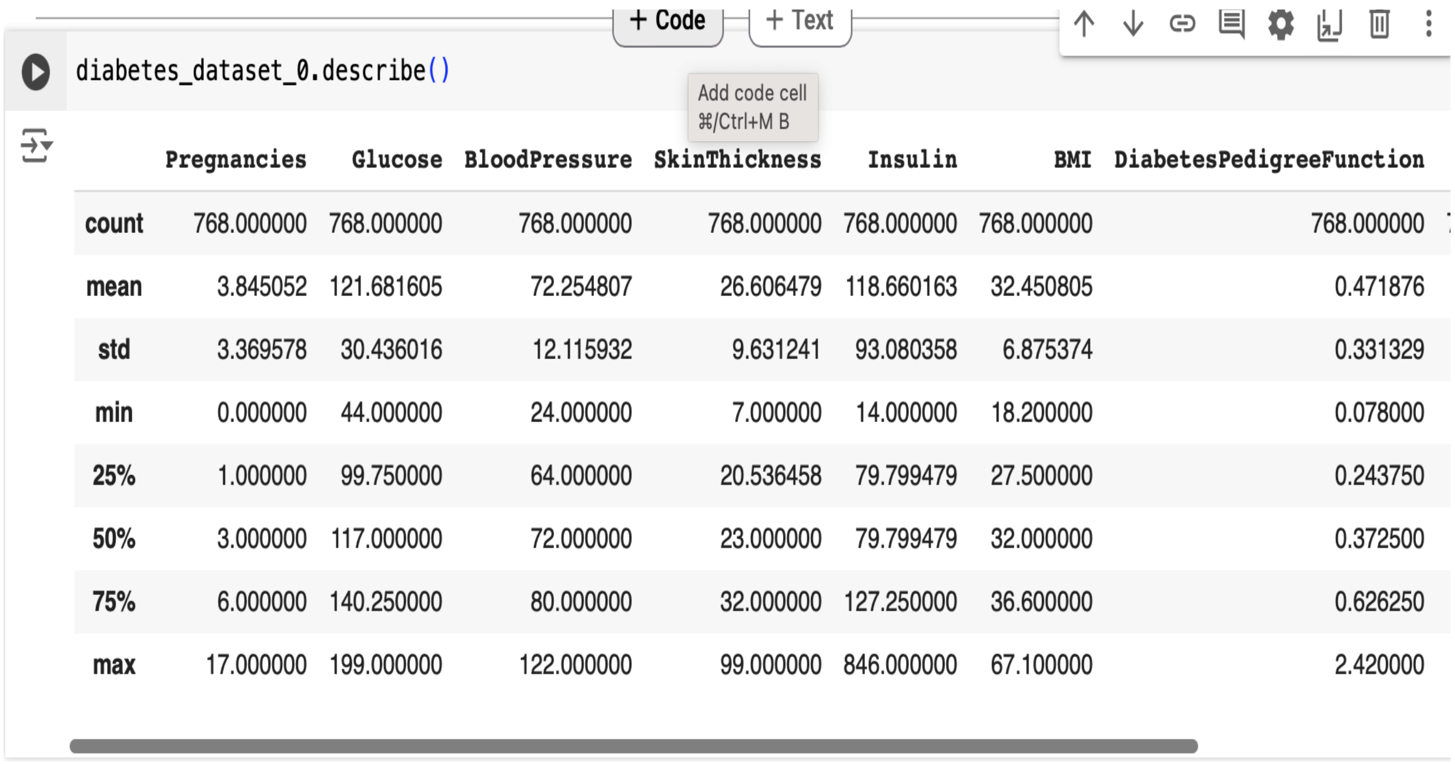
Fig 3.3.3 Finding the statistical measure of the data thus Calculating Mean, Count, Min, Max and Standard Deviation e.t.c



```
diabetes_dataset_0=diabetes_dataset.copy()
cols_0=['Glucose', 'BloodPressure', 'SkinThickness','Insulin','BMI']
for i in cols_0:
  diabetes_dataset_0[i].replace(0, diabetes_dataset_0[i].mean(),inplace=True)

diabetes_dataset_0.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  DiabetesPedigreeFunction  Age  Outcom
0            6    148.0           72.0      35.000000   79.799479  33.6                     0.627   50
1            1     85.0           66.0      29.000000   79.799479  26.6                     0.351   31
2            8    183.0           64.0      20.536458   79.799479  23.3                     0.672   32
3            1     89.0           66.0      23.000000   94.000000  28.1                     0.167   21
4            0    137.0           40.0      35.000000  168.000000  43.1                     2.288   33
```

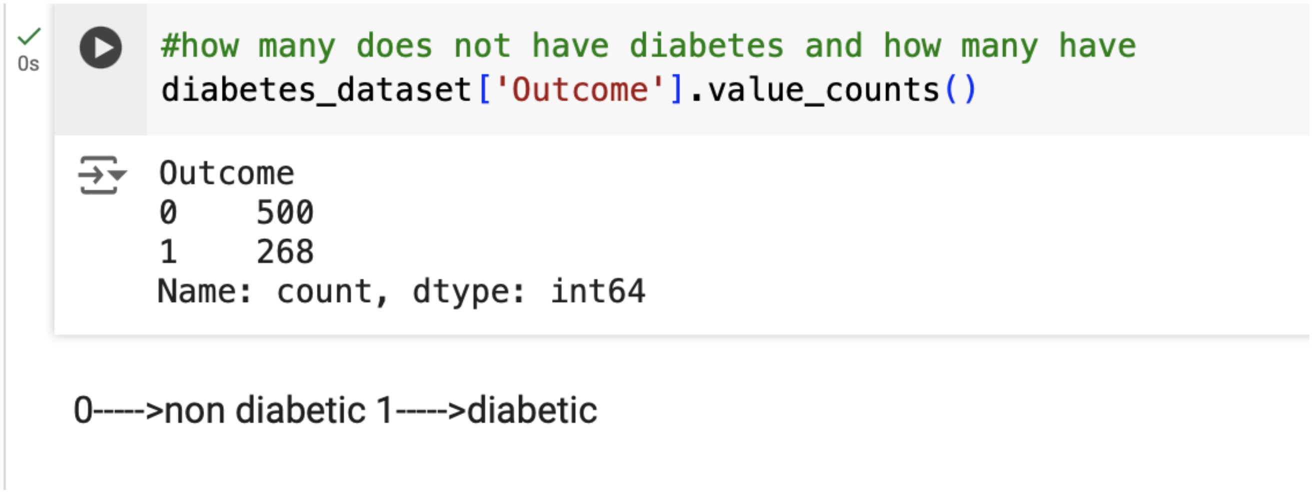
Fig 3.3.4 Based on the understanding of the parameters, it seems highly unlikely that glucose, blood pressure, skin thickness, insulin and BMI levels are 0. So, a copy is created.



```
diabetes_dataset_0.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
count	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
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

Fig 3.3.5 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.



```
#how many does not have diabetes and how many have
diabetes_dataset['Outcome'].value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

0----->non diabetic 1----->diabetic

Fig 3.3.6 How many have diabetes and how many have no diabetes

3.4 HISTOGRAM PLOT OF DATA

```
col = list(diabetes_dataset_0.columns)
diabetes_dataset_0[col].hist(stacked=True, bins=20, figsize=(6,6), layout=(3,3))
plt.tight_layout()
```

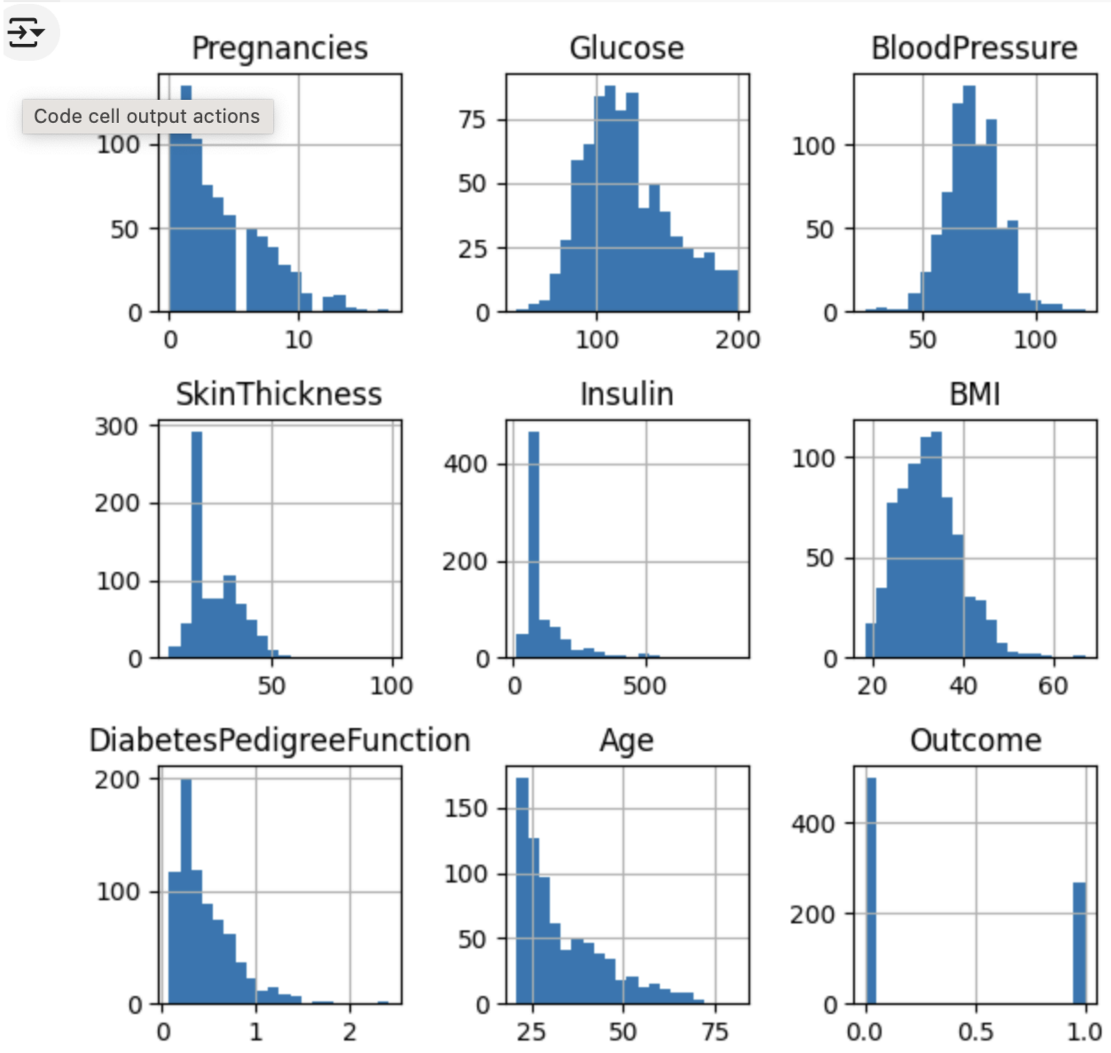
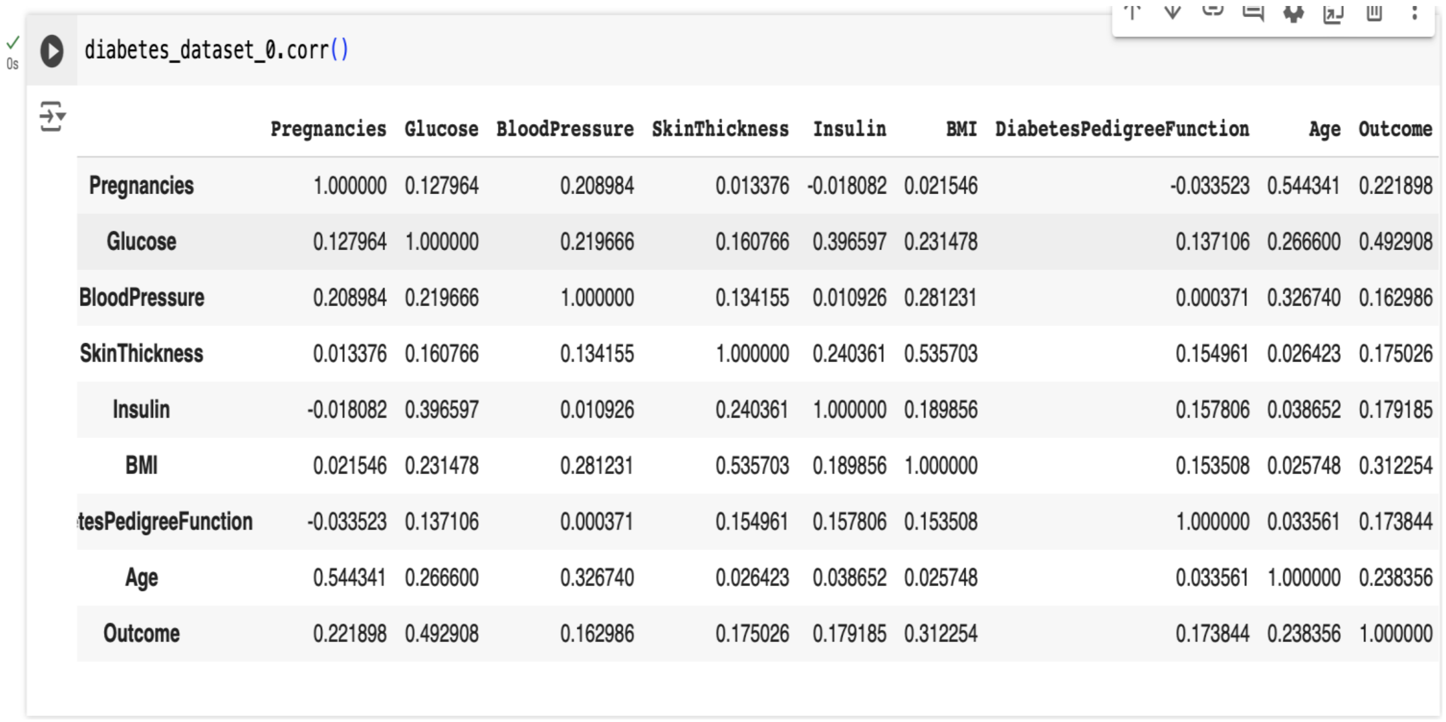


Fig 3.4.1 Histogram The above histogram plots give a high-level view of the bucket distribution of the dataset parameters. At first glance, most of them appear to be positively skewed, with Glucose and Blood Pressure with the closest distribution to a normal distribution. Outcome is a bimodal distribution which is to be expected.

3.5 CORRELATION OF DATA



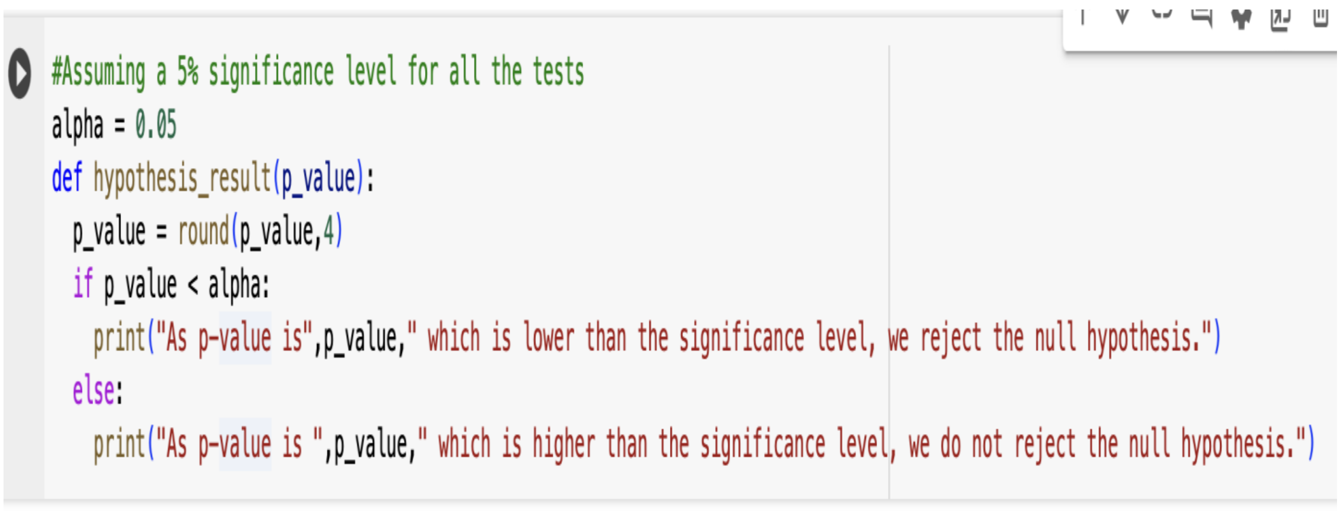
```
diabetes_dataset_0.corr()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546	-0.033523	0.544341	0.221898
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106	0.266600	0.492908
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371	0.326740	0.162986
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961	0.026423	0.175026
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806	0.038652	0.179185
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508	0.025748	0.312254
tesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000	0.033561	0.173844
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561	1.000000	0.238356
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844	0.238356	1.000000

This calculates the correlation between variables in the diabetes_dataset. Specifically, it computes the correlation coefficients between all pairs of numerical columns in the dataframe. The correlation coefficient measures the strength and direction of the linear relationship between two variables. The resulting correlation matrix provides insights into the relationships between different features, which can be useful for feature selection, identifying multicollinearity, and understanding the underlying patterns in the data.The parameter with the highest positive correlation to each other is BMI and Skin Thickness.

3.6HYPOTHESIS TESTING

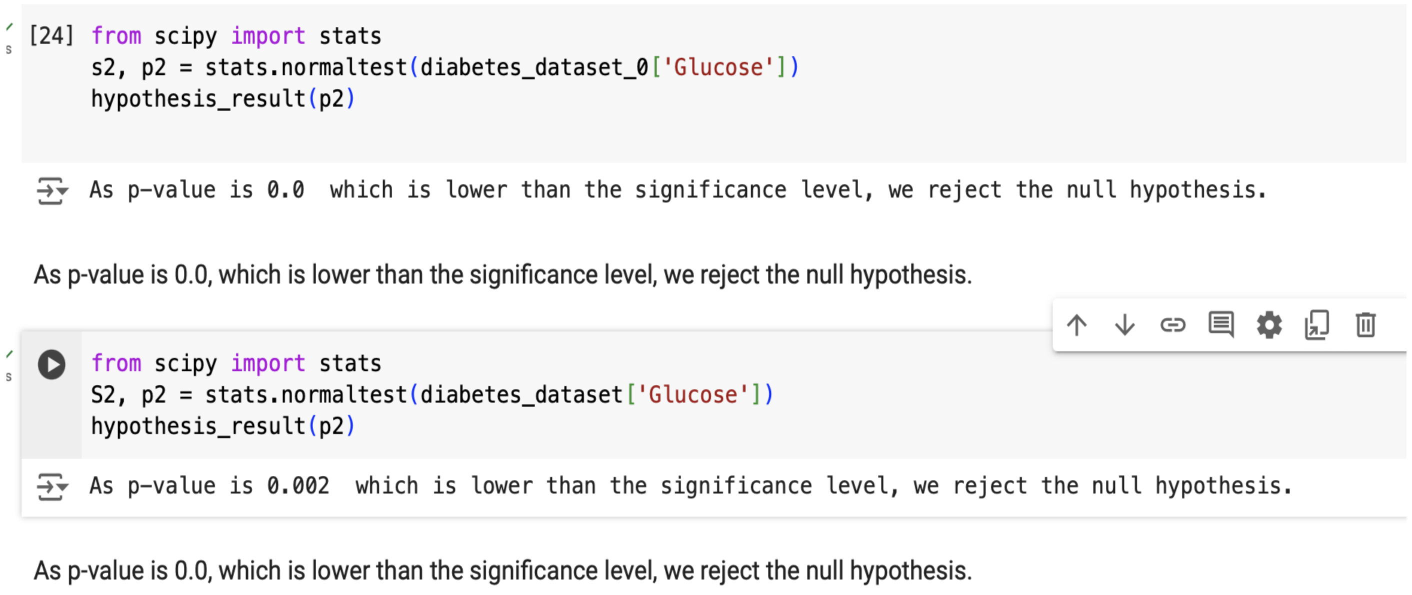
First hypothesis test would be to try and confirm my suspicion if Glucose data has a normal distribution. Next hypothesis test would be that based on the above hypothesis test, I would test the correlation between Glucose and the target outcome.



```
#Assuming a 5% significance level for all the tests
alpha = 0.05
def hypothesis_result(p_value):
  p_value = round(p_value,4)
  if p_value < alpha:
    print("As p-value is",p_value," which is lower than the significance level, we reject the null hypothesis.")
  else:
    print("As p-value is ",p_value," which is higher than the significance level, we do not reject the null hypothesis.")
```

Fig 3.6.1

First Hypothesis Test Null Hypothesis: The sample comes from a normal distribution. Alternative Hypothesis: The sample does not come from a normal distribution



```
[24] from scipy import stats
     s2, p2 = stats.normaltest(diabetes_dataset_0['Glucose'])
     hypothesis_result(p2)
```

```
As p-value is 0.0  which is lower than the significance level, we reject the null hypothesis.
```

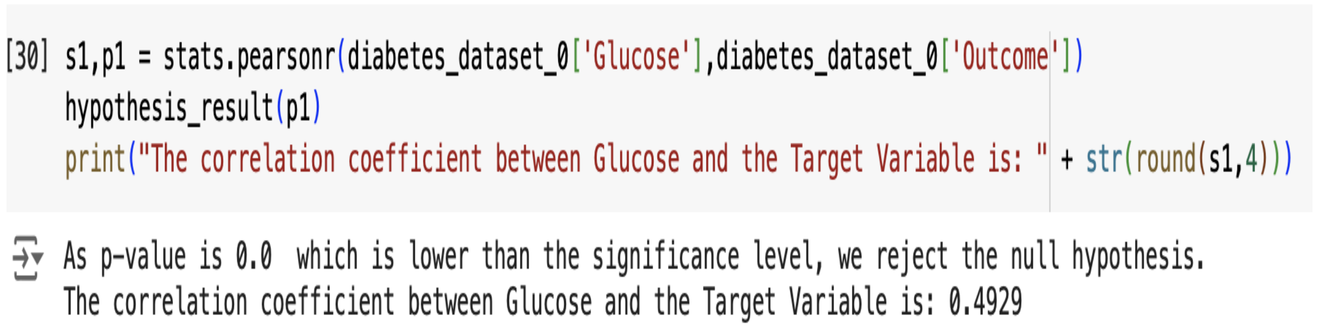
As p-value is 0.0, which is lower than the significance level, we reject the null hypothesis.

```
from scipy import stats
S2, p2 = stats.normaltest(diabetes_dataset['Glucose'])
hypothesis_result(p2)
```

```
As p-value is 0.002  which is lower than the significance level, we reject the null hypothesis.
```

As p-value is 0.0, which is lower than the significance level, we reject the null hypothesis.

Fig 3.6.2. First Hypothesis Testing From the above test result, we reject Glucose having a normal distribution. Second Hypothesis Test Next, we will perform a Pearson correlation test. Null Hypothesis: Both sets of data are uncorrelated. Alternative Hypothesis: Both sets of data are correlated.



```
[30] s1,p1 = stats.pearsonr(diabetes_dataset_0['Glucose'],diabetes_dataset_0['Outcome'])
     hypothesis_result(p1)
     print("The correlation coefficient between Glucose and the Target Variable is: " + str(round(s1,4)))
```

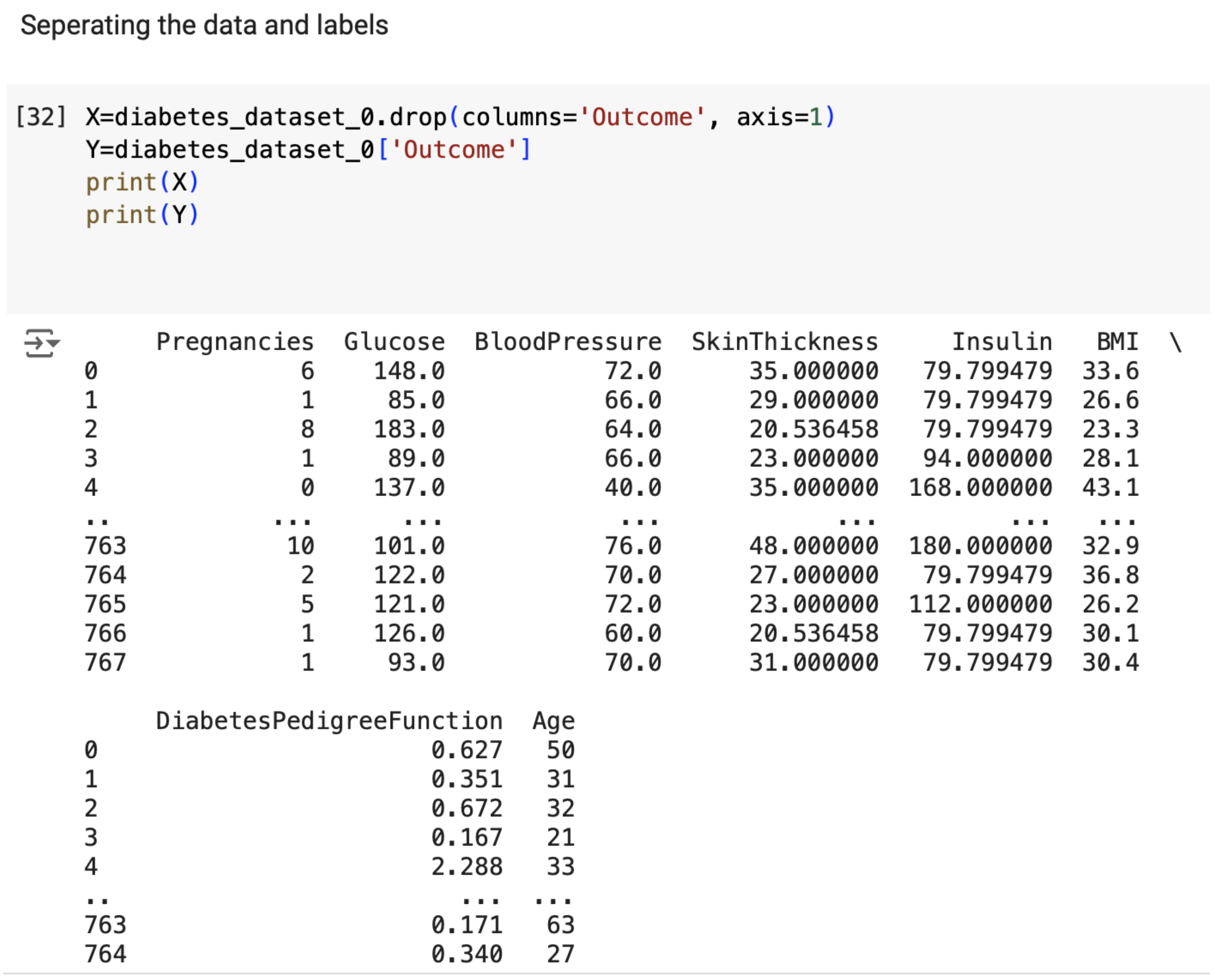
```
As p-value is 0.0  which is lower than the significance level, we reject the null hypothesis.
The correlation coefficient between Glucose and the Target Variable is: 0.4929
```

Fig3.6.3 Second Hypothesis

From the above hypothesis testing, there is indeed some correlation between Glucose levels and diabetes.

3.7 Splitting of dataset(Training/Validation/Testing)

The splitting of the dataset for validation and testing. Training Dataset: Dataset sample that is used to fit the model. Validation Dataset: Dataset sample that is used for hyper tuning the parameters, and comparing the accuracy and error rates of the model performance between using the training dataset and the validation dataset. Testing Dataset: Dataset sample that is used to test the model performance (predictive power).

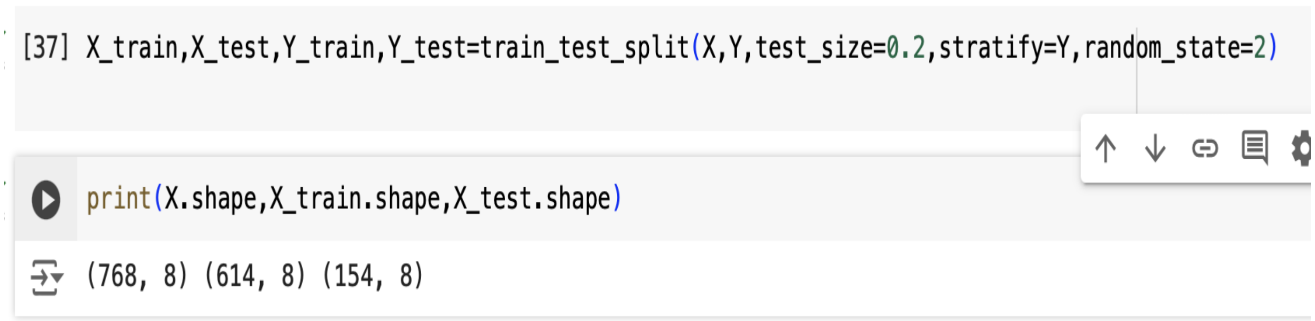


Seperating the data and labels

```
[32] X=diabetes_dataset_0.drop(columns='Outcome', axis=1)
     Y=diabetes_dataset_0['Outcome']
     print(X)
     print(Y)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0              6    148.0           72.0      35.000000   79.799479  33.6   
1              1     85.0           66.0      29.000000   79.799479  26.6   
2              8    183.0           64.0      20.536458   79.799479  23.3   
3              1     89.0           66.0      23.000000   94.000000  28.1   
4              0    137.0           40.0      35.000000  168.000000  43.1   
..           ...      ...            ...            ...         ...   ...   
763           10    101.0           76.0      48.000000  180.000000  32.9   
764            2    122.0           70.0      27.000000   79.799479  36.8   
765            5    121.0           72.0      23.000000  112.000000  26.2   
766            1    126.0           60.0      20.536458   79.799479  30.1   
767            1     93.0           70.0      31.000000   79.799479  30.4   

     DiabetesPedigreeFunction  Age  
0                       0.627   50  
1                       0.351   31  
2                       0.672   32  
3                       0.167   21  
4                       2.288   33  
..                        ...  ...  
763                     0.171   63  
764                     0.340   27  
```

TRAIN TEST SPLIT



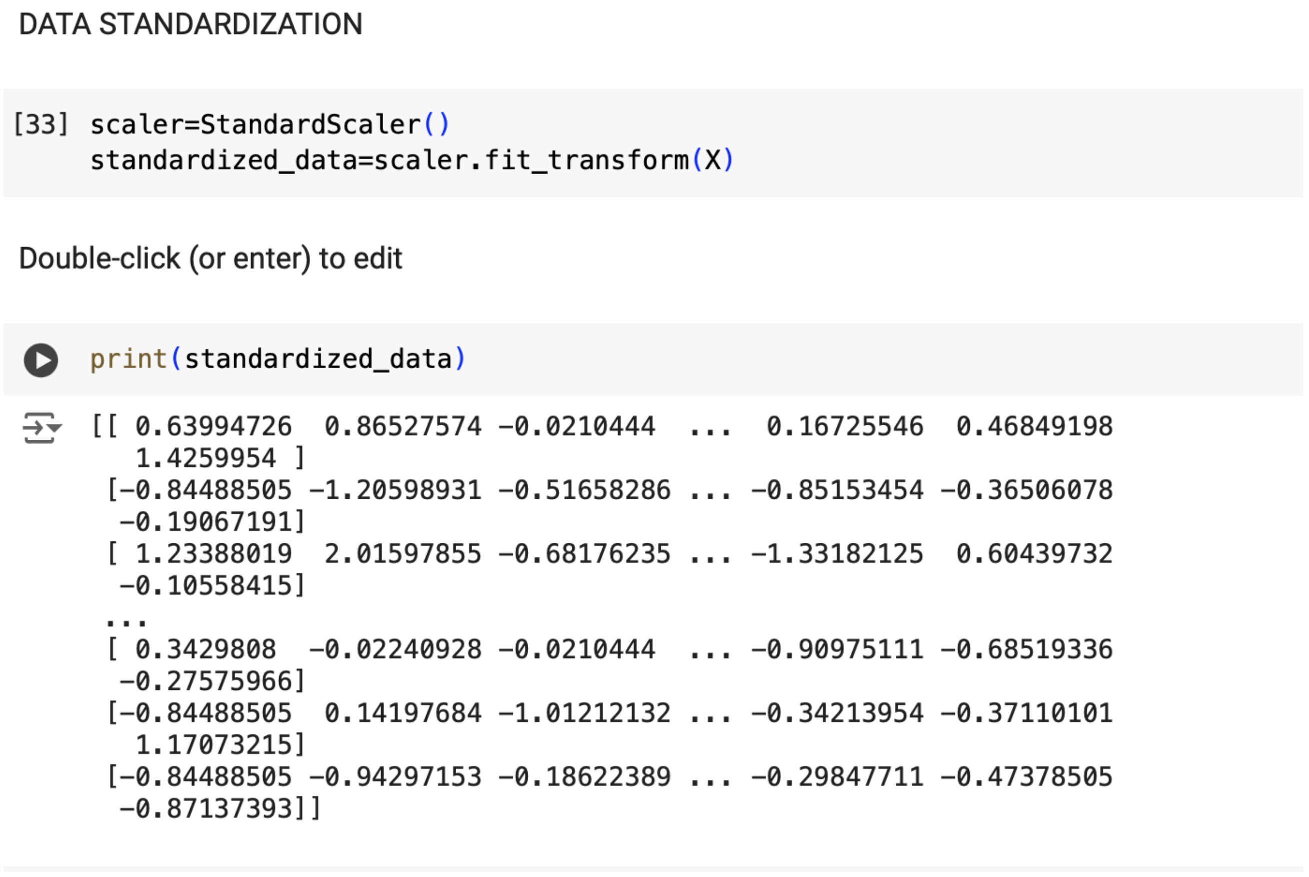
```
[37] X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

3.8 Feature Scaling

Here StandardScaler() is used to perform feature scaling. This will retain the mean and the standard deviation of the sample distribution of the data set, and reuse it to transform the X_train and X_test subsequently. I try to reuse the mean and standard deviation obtained from the training set and apply it to the testing set as well. Standardizing data after data splitting is to prevent data leakage from test dataset into train dataset.



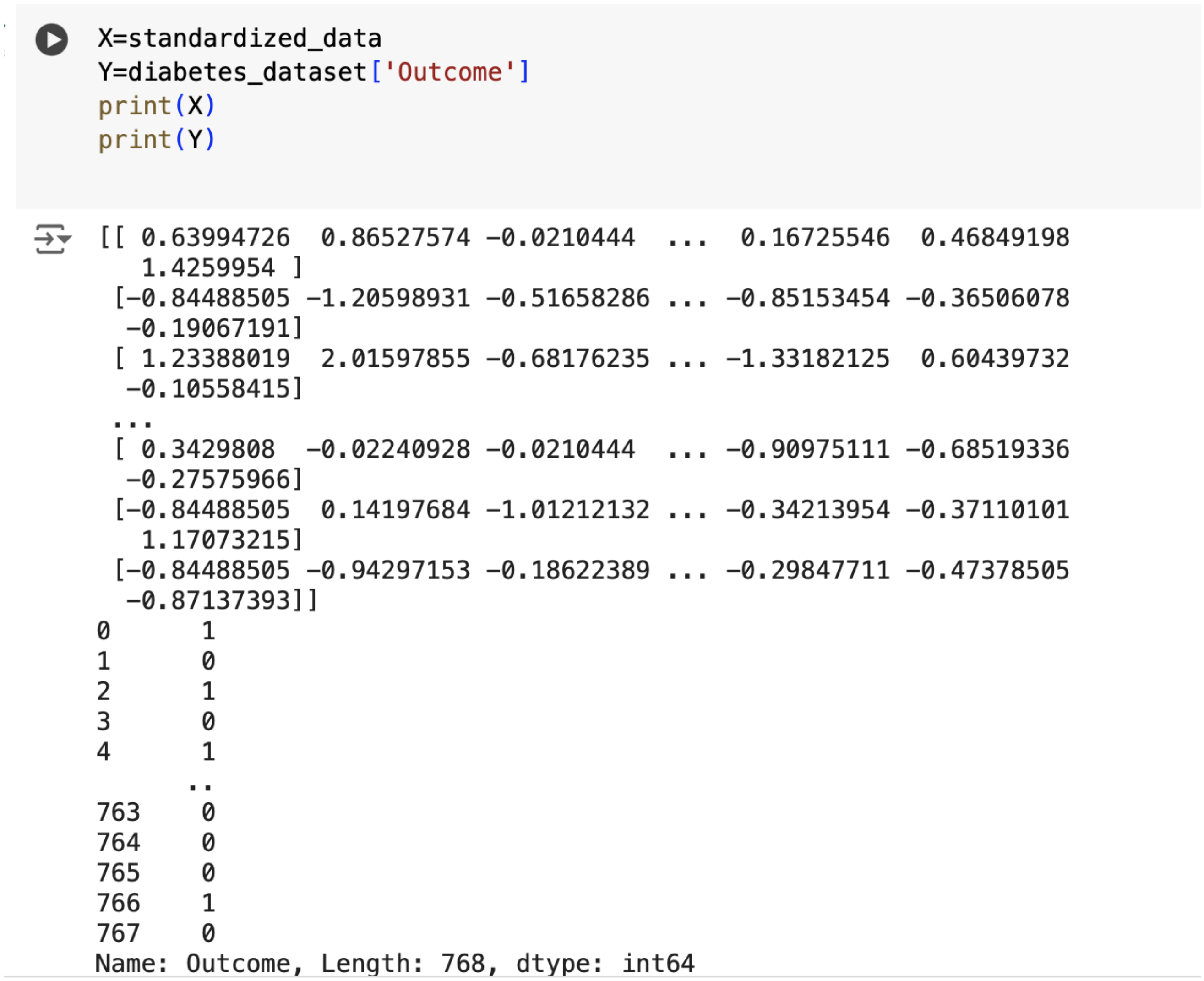
DATA STANDARDIZATION

```
[33] scaler=StandardScaler()
     standardized_data=scaler.fit_transform(X)
```

Double-click (or enter) to edit

```
print(standardized_data)
```

```
[[ 0.63994726  0.86527574 -0.0210444   ...  0.16725546  0.46849198
   1.4259954 ]
 [-0.84488505 -1.20598931 -0.51658286 ... -0.85153454 -0.36506078
  -0.19067191]
 [ 1.23388019  2.01597855 -0.68176235 ... -1.33182125  0.60439732
  -0.10558415]
 ...
 [ 0.3429808  -0.02240928 -0.0210444   ... -0.90975111 -0.68519336
  -0.27575966]
 [-0.84488505  0.14197684 -1.01212132 ... -0.34213954 -0.37110101
   1.17073215]
 [-0.84488505 -0.94297153 -0.18622389 ... -0.29847711 -0.47378505
  -0.87137393]]
```



```
X=standardized_data
Y=diabetes_dataset['Outcome']
print(X)
print(Y)
```

```
[[ 0.63994726  0.86527574 -0.0210444  ...  0.16725546  0.46849198
   1.4259954 ]
 [-0.84488505 -1.20598931 -0.51658286 ... -0.85153454 -0.36506078
  -0.19067191]
 [ 1.23388019  2.01597855 -0.68176235 ... -1.33182125  0.60439732
  -0.10558415]
 ...
 [ 0.3429808  -0.02240928 -0.0210444  ... -0.90975111 -0.68519336
  -0.27575966]
 [-0.84488505  0.14197684 -1.01212132 ... -0.34213954 -0.37110101
   1.17073215]
 [-0.84488505 -0.94297153 -0.18622389 ... -0.29847711 -0.47378505
  -0.87137393]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

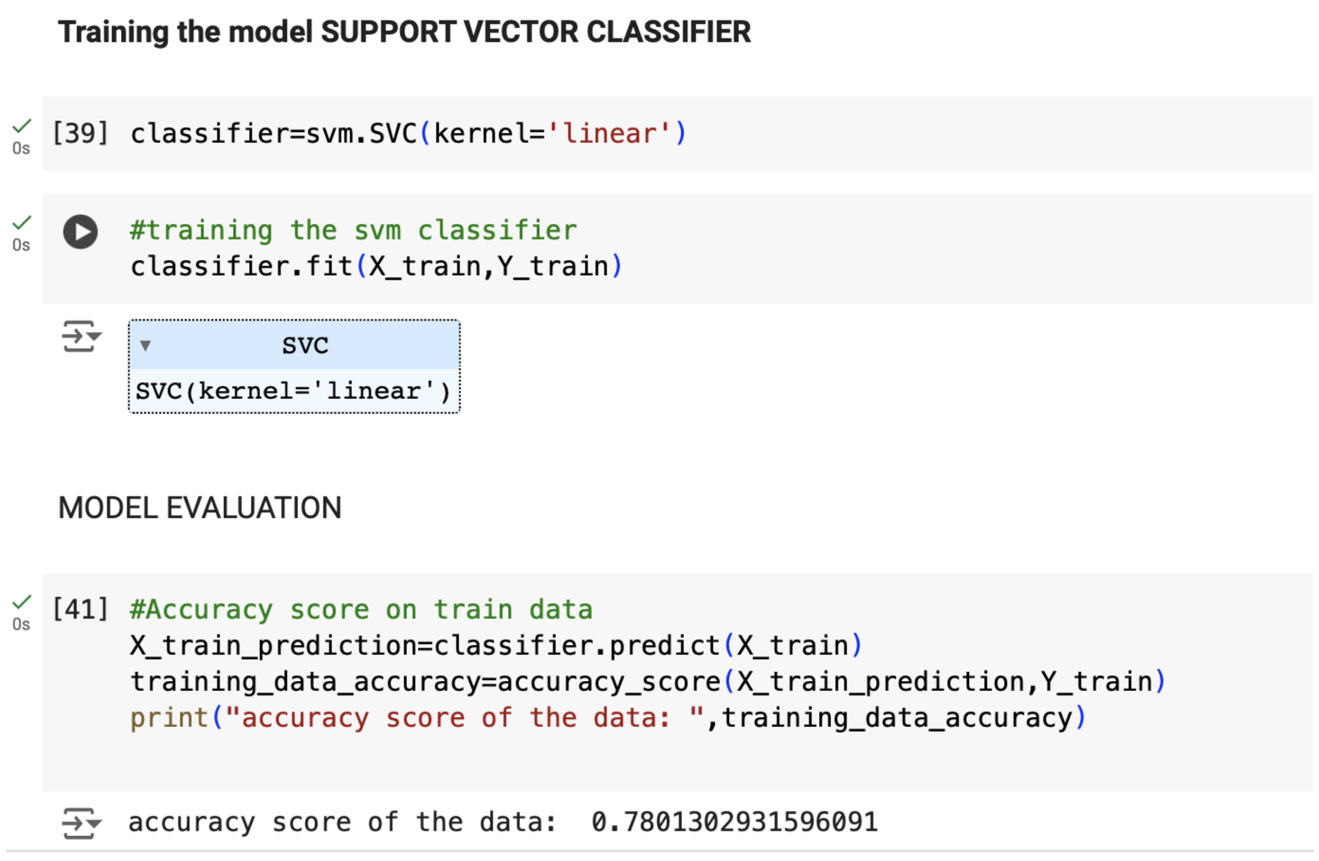
Fig 3.8.1 Scaling

3.9 IMPLEMENTING MACHINE LEARNING ALGORITHMS AND RESULTS

Different machine learning algorithms to try and classify the pima Indian diabetes dataset. Support Vector Machine,

3.9.1 Support Vector Machine

Visual representation of SVM output, demonstrating its ability to delineate decision boundaries and classify data points with high accuracy.



Training the model SUPPORT VECTOR CLASSIFIER

```
[39] classifier=svm.SVC(kernel='linear')
```

```
#training the svm classifier
classifier.fit(X_train,Y_train)
```

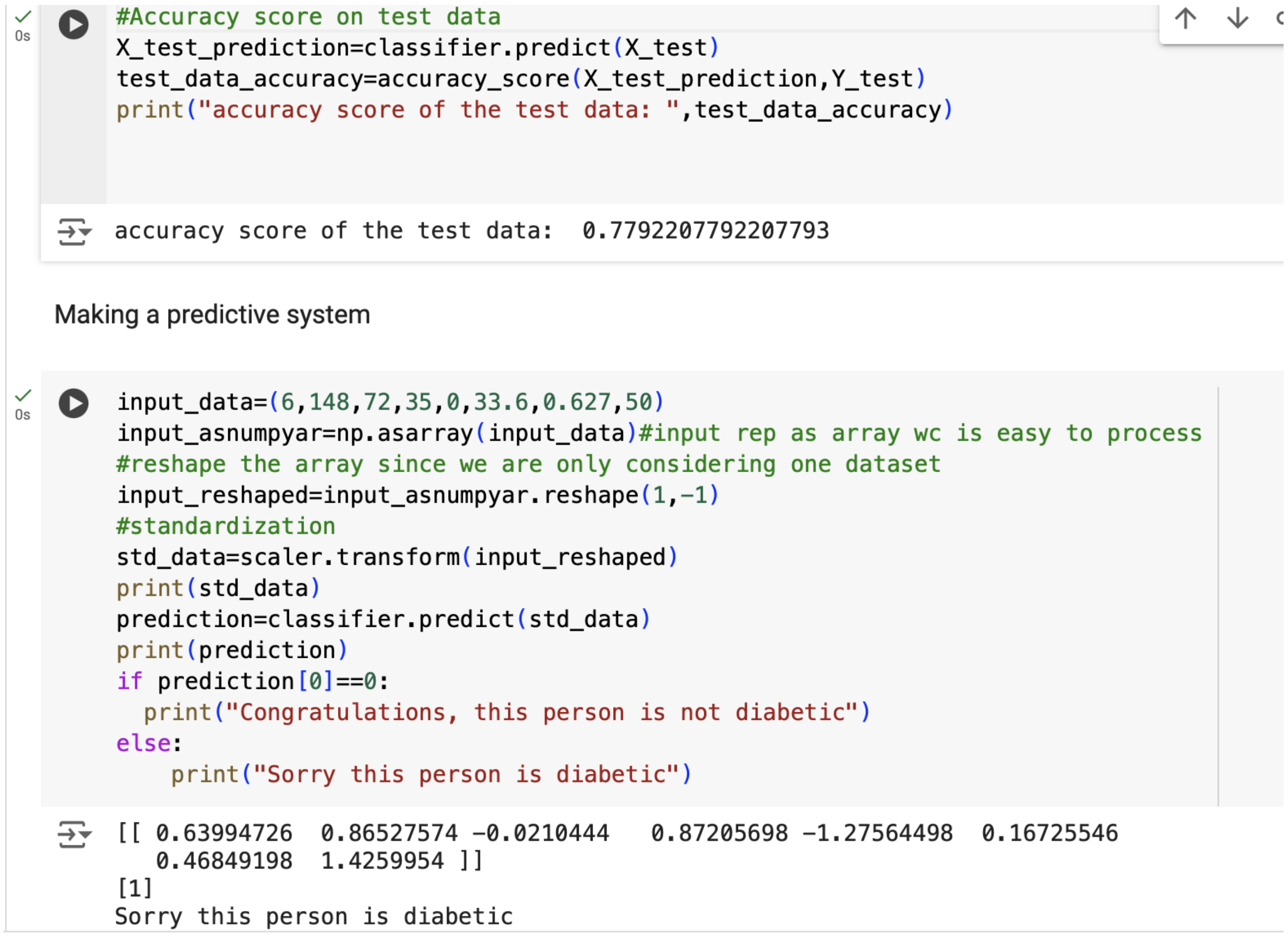
```
▾ SVC
SVC(kernel='linear')
```

MODEL EVALUATION

```
[41] #Accuracy score on train data
X_train_prediction=classifier.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
print("accuracy score of the data: ",training_data_accuracy)
```

```
accuracy score of the data:  0.7801302931596091
```

FIG 3.9.1.1



```
#Accuracy score on test data
X_test_prediction=classifier.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
print("accuracy score of the test data: ",test_data_accuracy)
```

```
accuracy score of the test data:  0.7792207792207793
```

Making a predictive system

```
input_data=(6,148,72,35,0,33.6,0.627,50)
input_asnumpyar=np.asarray(input_data)#input rep as array wc is easy to process
#reshape the array since we are only considering one dataset
input_reshaped=input_asnumpyar.reshape(1,-1)
#standardization
std_data=scaler.transform(input_reshaped)
print(std_data)
prediction=classifier.predict(std_data)
print(prediction)
if prediction[0]==0:
  print("Congratulations, this person is not diabetic")
else:
    print("Sorry this person is diabetic")
```

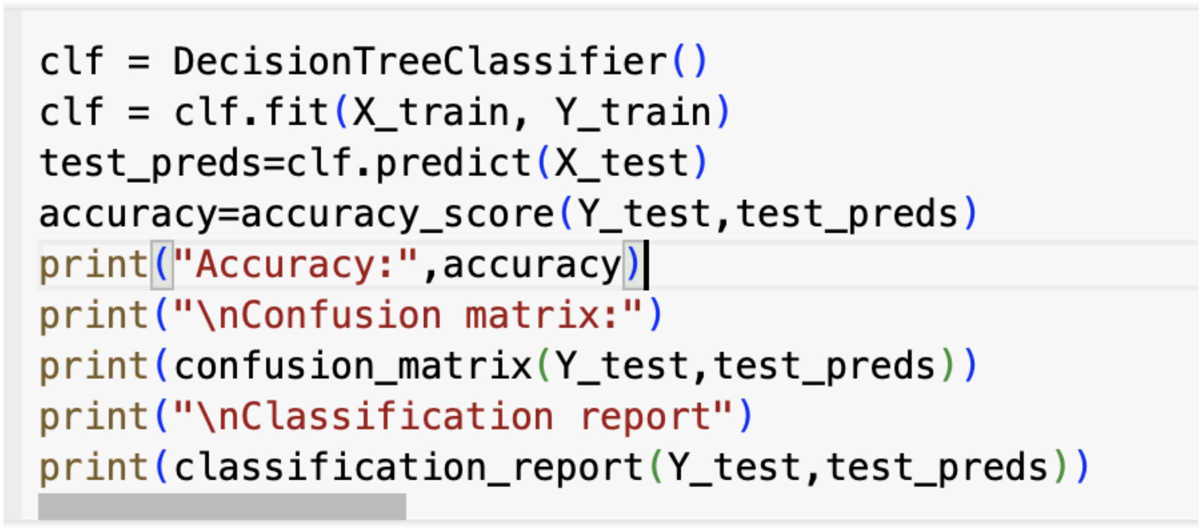
```
[[ 0.63994726  0.86527574 -0.0210444   0.87205698 -1.27564498  0.16725546
   0.46849198  1.4259954 ]]
[1]
Sorry this person is diabetic
```

FIG 3.9.1.2 Making a predictive System

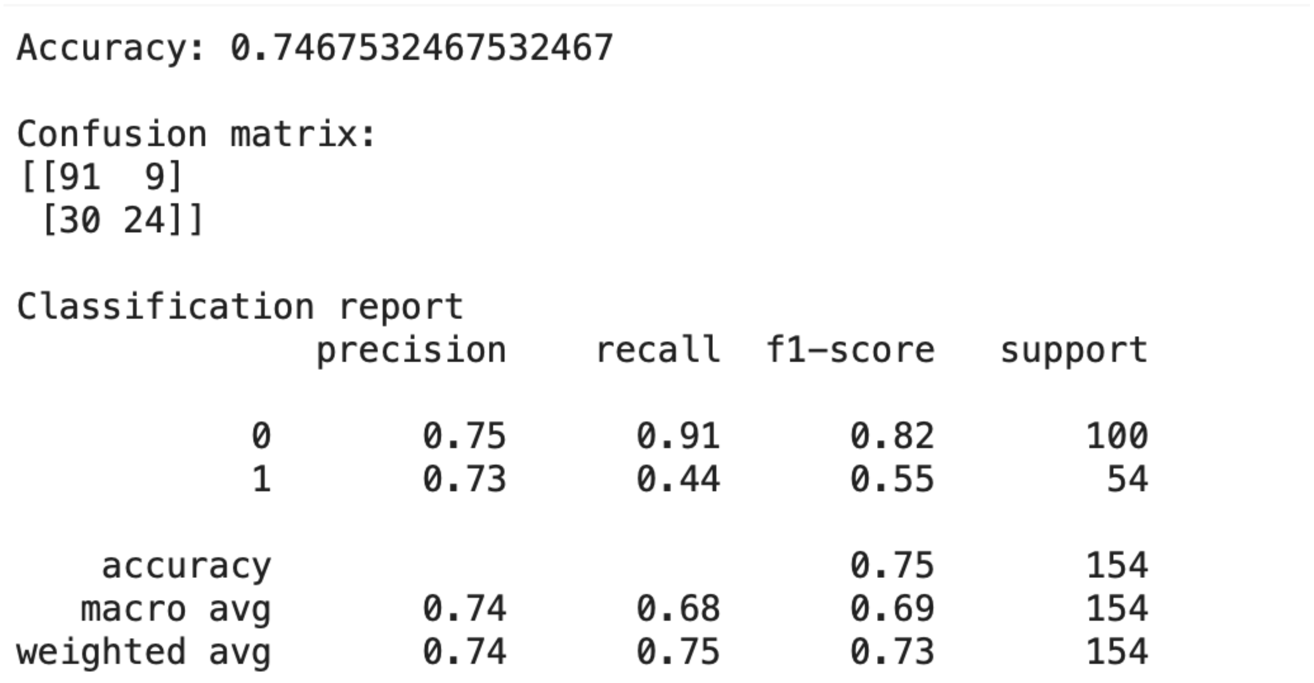
The input_data consist of all the required attributes that can be used to determine whether the person is diabetic or not and by the use of the Support vector machine classifier the system will detect whether the person has diabetes or not by printing the message

3.9.2 DECISION TREE CLASSIFIER

Visual representation of the Decision Tree Classifier output, illustrating its intuitive decision-making process by partitioning the feature space into distinct regions for accurate classification



```
clf = DecisionTreeClassifier()
clf = clf.fit(X_train, Y_train)
test_preds=clf.predict(X_test)
accuracy=accuracy_score(Y_test,test_preds)
print("Accuracy:",accuracy)
print("\nConfusion matrix:")
print(confusion_matrix(Y_test,test_preds))
print("\nClassification report")
print(classification_report(Y_test,test_preds))
```

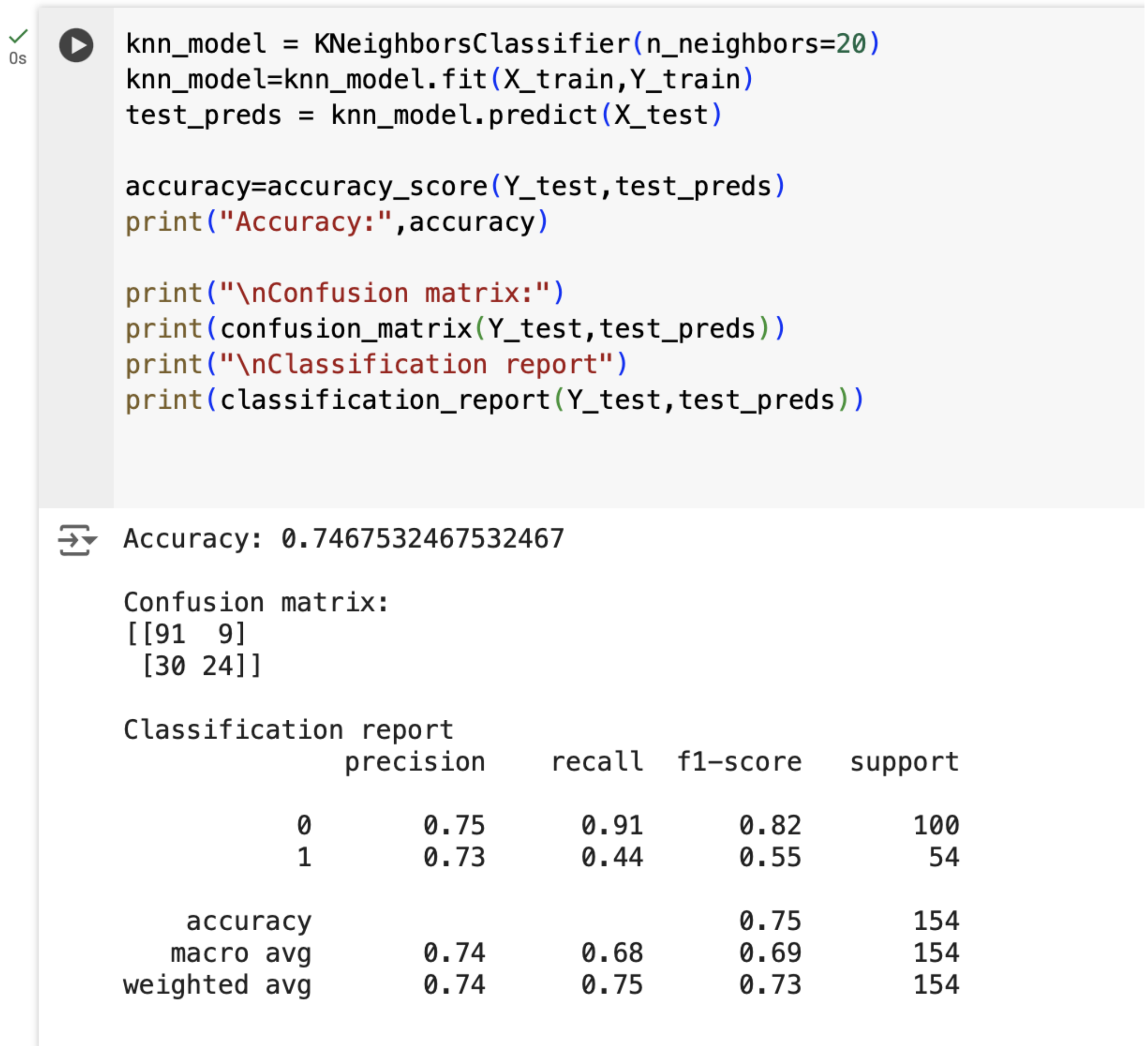


```
Accuracy: 0.7467532467532467

Confusion matrix:
[[91  9]
 [30 24]]

Classification report
              precision    recall  f1-score   support

           0       0.75      0.91      0.82       100
           1       0.73      0.44      0.55        54

    accuracy                           0.75       154
   macro avg       0.74      0.68      0.69       154
weighted avg       0.74      0.75      0.73       154
```

FIG 3.9.2 .1 Shows how thwe Decion Tree Classifier was used to find the perfomance of the system as we calculate Accuracy,Confusion matrix and Classification report

3.9.3 K NEAREST NEIGHBOUR

Visual depiction of the KNN output, demonstrating its non-parametric approach by assigning labels to data points according to the most prevalent class among their closest neighbors. The KNN algorithm relies on distance metrics, such as Euclidean to determine proximity and classify data points accordingly.



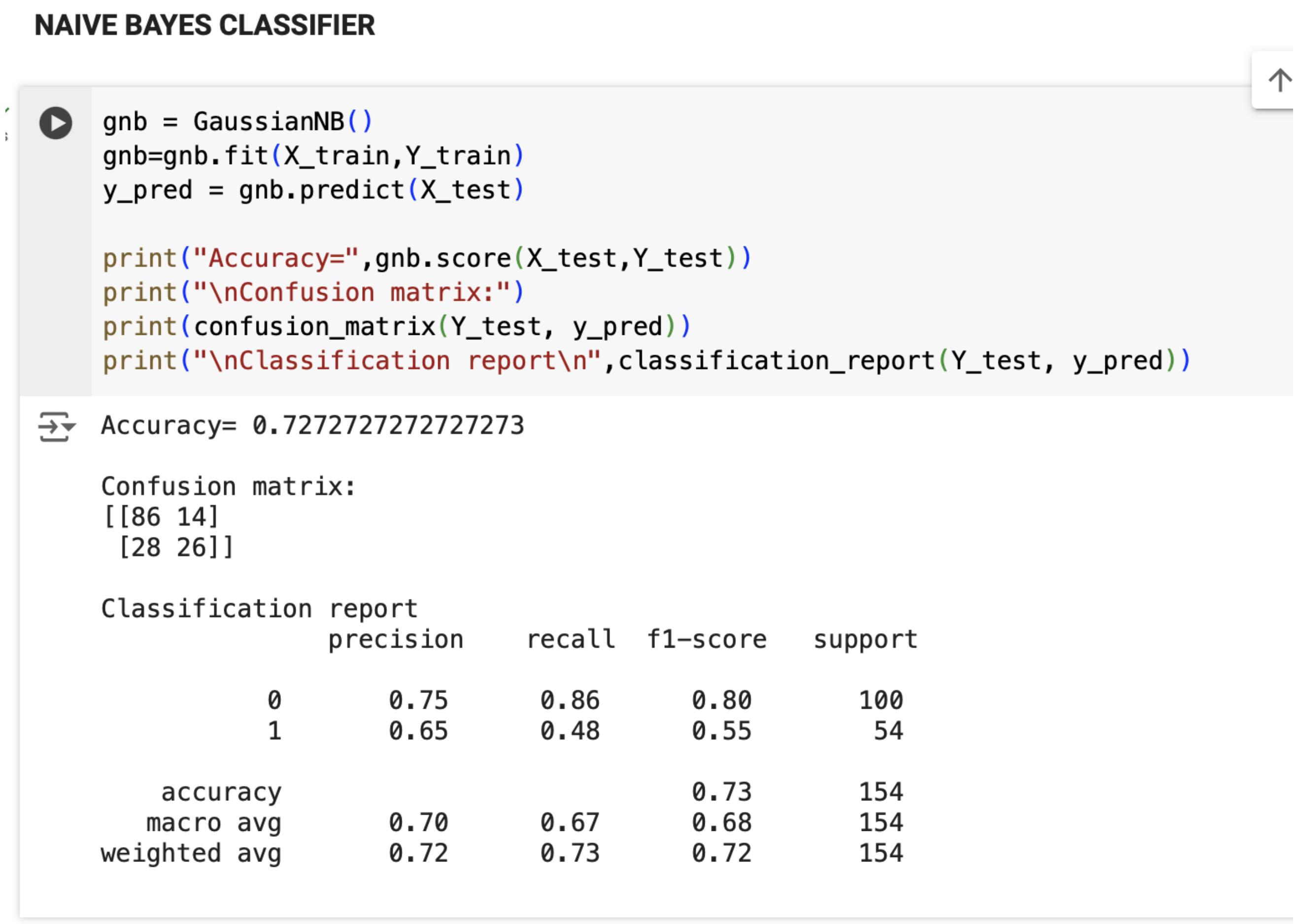
```
knn_model = KNeighborsClassifier(n_neighbors=20)
knn_model=knn_model.fit(X_train,Y_train)
test_preds = knn_model.predict(X_test)

accuracy=accuracy_score(Y_test,test_preds)
print("Accuracy:",accuracy)

print("\nConfusion matrix:")
print(confusion_matrix(Y_test,test_preds))
print("\nClassification report")
print(classification_report(Y_test,test_preds))
```

```
Accuracy: 0.7467532467532467

Confusion matrix:
[[91  9]
 [30 24]]

Classification report
              precision    recall  f1-score   support

           0       0.75      0.91      0.82       100
           1       0.73      0.44      0.55        54

    accuracy                           0.75       154
   macro avg       0.74      0.68      0.69       154
weighted avg       0.74      0.75      0.73       154
```

FIG 3.9.3.1 Shows how the K Nearest Neighbour was used to find the perfomance of the system as we calculate Accuracy,Confusion matrix and Classification report

3.9.4 NAIVE BAYES CLASSIFIER

Visual representation of the Naive Bayes Classifier output, showcasing its ability to predict the class label of data points by maximizing the posterior probability using Bayes' theorem. The Naive Bayes Classifier assumes independence among features, simplifying calculations and making it particularly effective for text classification and other high-dimensional datasets.



NAIVE BAYES CLASSIFIER

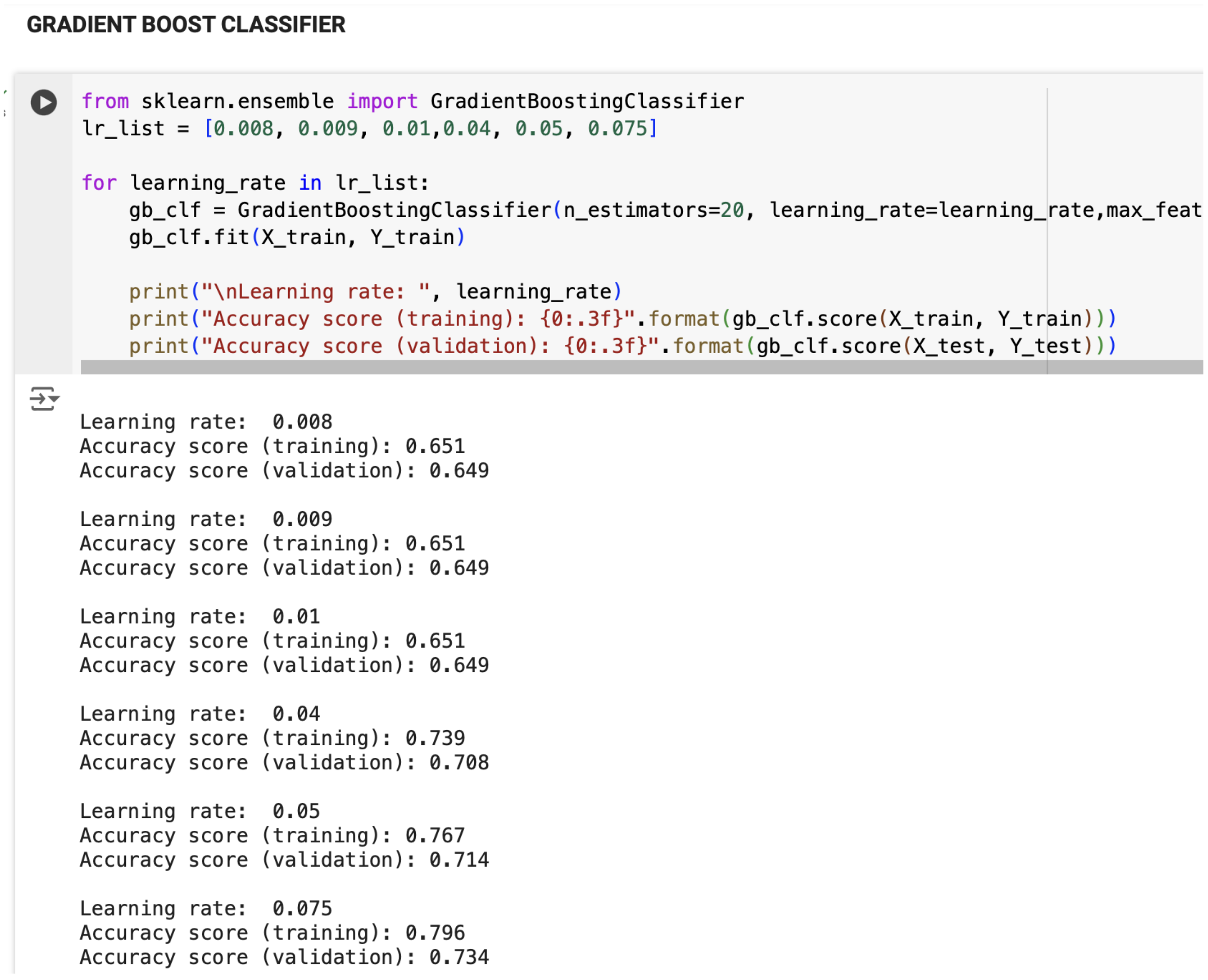
```
gnb = GaussianNB()
gnb=gnb.fit(X_train,Y_train)
y_pred = gnb.predict(X_test)

print("Accuracy=",gnb.score(X_test,Y_test))
print("\nConfusion matrix:")
print(confusion_matrix(Y_test, y_pred))
print("\nClassification report\n",classification_report(Y_test, y_pred))
```

```
Accuracy= 0.7272727272727273

Confusion matrix:
[[86 14]
 [28 26]]

Classification report
               precision    recall  f1-score   support

           0       0.75      0.86      0.80       100
           1       0.65      0.48      0.55        54

    accuracy                           0.73       154
   macro avg       0.70      0.67      0.68       154
weighted avg       0.72      0.73      0.72       154
```

FIG 3.9.2 Shows how the Naive Bayes Classifier was used to find the perfomance of the system as we calculate Accuracy,Confusion matrix and Classification report

3.9.5 GRADIENT BOOST CLASSIFIER

A Visual depiction of the Gradient Boost Classifier output, showcasing its iterative approach of combining the predictions from each weak learner to gradually improve the model's overall accuracy. Gradient Boosting optimizes a loss function by adding new models that complement the errors made by previous models, resulting in a strong predictive model capable of handling complex datasets.



GRADIENT BOOST CLASSIFIER

```
from sklearn.ensemble import GradientBoostingClassifier
lr_list = [0.008, 0.009, 0.01,0.04, 0.05, 0.075]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=learning_rate,max_feat
    gb_clf.fit(X_train, Y_train)

    print("\nLearning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, Y_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_test, Y_test)))
```

```
Learning rate:  0.008
Accuracy score (training): 0.651
Accuracy score (validation): 0.649

Learning rate:  0.009
Accuracy score (training): 0.651
Accuracy score (validation): 0.649

Learning rate:  0.01
Accuracy score (training): 0.651
Accuracy score (validation): 0.649

Learning rate:  0.04
Accuracy score (training): 0.739
Accuracy score (validation): 0.708

Learning rate:  0.05
Accuracy score (training): 0.767
Accuracy score (validation): 0.714

Learning rate:  0.075
Accuracy score (training): 0.796
Accuracy score (validation): 0.734
```

FIG 3.9.5.1 Shows how the Gradient used to find the perfomance of the system as we calculate Accuracy score of training and validation using a different learning rate at each case.

4. TEST RESULTS 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, Support Vector Machine 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, KNN, Decision Tree, Naive Bayes Classifier and Gradient Boosting 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 72% to 79%. Based on Accuracy and Recall score, overly the Support Vector Machine produced the best results.

5.CONCLUSION

- Leveraging advanced computational methods, machine learning holds immense potential to transform diabetes prediction, ushering in a new era of precision healthcare.
- Early detection of diabetes stands as a pivotal factor in facilitating timely treatment interventions.
- Moreover, this technique holds promise for researchers aiming to develop a robust diagnostic tool, empowering clinicians with enhanced decision-making capabilities in managing the disease.
- The project's future trajectory envisions incorporating additional parameters and factors to further refine prediction accuracy.
- As the number of parameters expands, so does the potential for heightened prediction accuracy.
- By employing conventional methodologies and algorithms, there lies an opportunity to augment accuracy through iterative enhancements.

REFERENCES

[1] https://www.kaggle.com/uciml/pima-indians-diabetes-database

[2] Diabetes Prediction using Machine Learning Algorithms - ScienceDirect

[3] Predicting Diabetes Mellitus With Machine Learning Techniques (nih.gov)

[4] Diabetes Prediction using Machine Learning Techniques – IJERT

[5]https://www.researchgate.net/publication/339543101_Diabetes_Prediction_using_
Machi ne_Learning_Algorithms