PimaIndia Diabetes Dataset

COURSE PRESENTER:
(DR.Omaima A. Fallatah)

SUBMITTED BY:

Name	ID
Layan Adel Babkur	444002368
Reham Faisal Alsubhi	444003014

DEPARTMENT OF(INFORMATION SCIENCE-DATA SCIENCE)
COLLEGE OF COMPUTER AND INFORMATION SYSTEMS
UMM AL-QURA UNIVERSITY

Table of content

1.Introduction..3

2.ExploratoryDataAnalysis ………………………….4

3NaiveBayesClassifier.……………………………………...9

1.Introduction

This dataset was created for diagnostic purposes It originates from the National Institute of Diabetes and Digestive and Kidney Diseases and includes various medical predictor (independent) variables such as the number of pregnancies, BMI, insulin level, age, and more. The target (dependent) variable is the outcome, which indicates whether or not the patient has diabetes.

The importance of knowing Vital Signs

Vital signs play a crucial role in the management and assessment of diabetes. For individuals with diabetes, monitoring vital signs can help in managing the disease effectively and preventing complications. There are basic signs of diabetes such as blood pressure, body mass index, insulin level, age, etc

The our goals

- The Primary goal of determining whether a patient has diabetes.
- NaiveBayes Model Improvement Attempt.

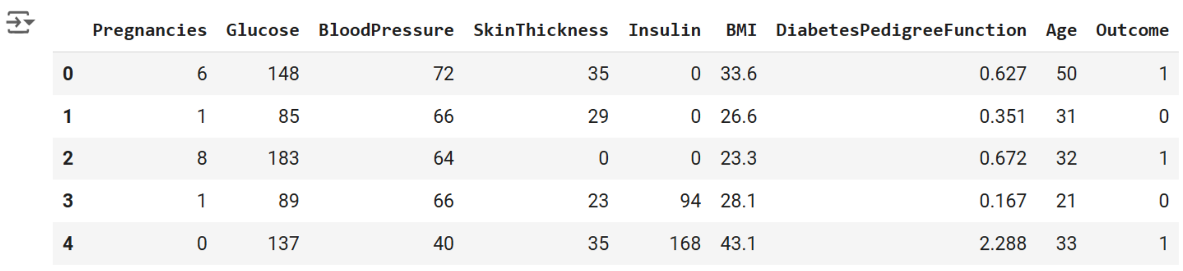
Link dataset

2.Exploratory Data Analysis

Attribute about dataset

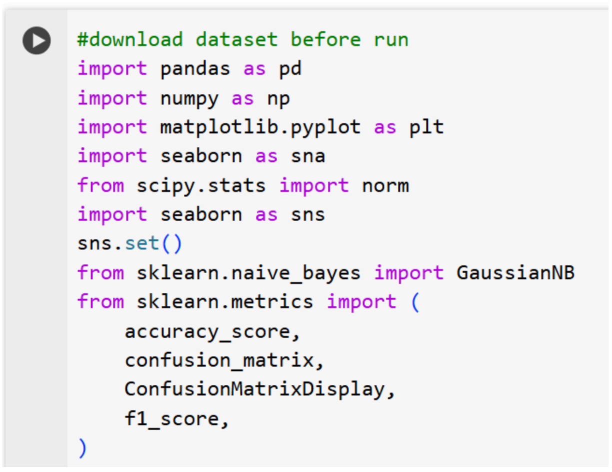
Pregnancies: The total count of the patient's pregnancies
.-Glucose: Plasma glucose concentration (mg/dL) measured two hours after a glucose tolerance test
.-BloodPressure: The measured diastolic blood pressure (mm Hg)
.-SkinThickness: Thickness of the triceps skin fold (mm), indicating subcutaneous fat
.-Insulin: Serum insulin level (mu U/ml) measured two hours post-test.
- Body Mass Index (BMI): Weight in kilograms divided by the square of height in meters.
- Age: The patient's age.
-DiabetesPedigreeFunction: A function indicating the genetic impact on diabetes risk based on family history.
- Outcome: The target variable indicating the patient's diabetes status: 1 (diabetic) or 0 (non-diabetic).

Datasets

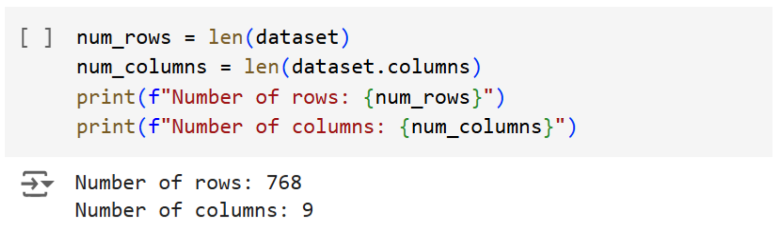


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Libraries used



```python
#download dataset before run
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sna
from scipy.stats import norm
import seaborn as sns
sns.set()
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
)
```

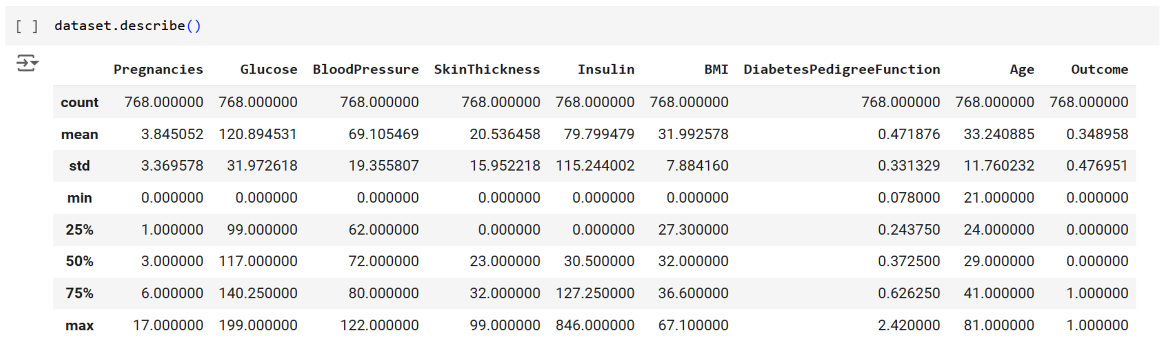


```
[ ] num_rows = len(dataset)
    num_columns = len(dataset.columns)
    print(f"Number of rows: {num_rows}")
    print(f"Number of columns: {num_columns}")
```

```
Number of rows: 768
Number of columns: 9
```

This code displays the number of columns and rows in the dataset.

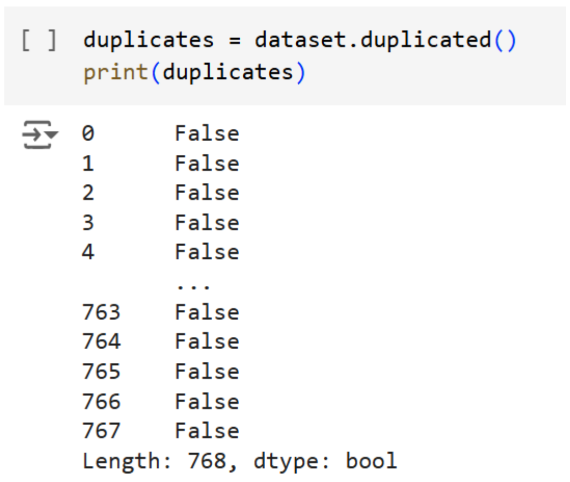
- Number of Columns:9
- Number of rows:768



```
[ ] 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

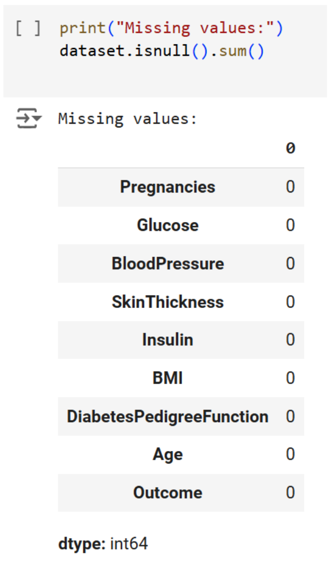
We used the"describe()", This function provides an overview of data distribution and basic statistics for the available variables.



```
[ ] duplicates = dataset.duplicated()
    print(duplicates)
```

```
0      False
1      False
2      False
3      False
4      False
       ...
763    False
764    False
765    False
766    False
767    False
Length: 768, dtype: bool
```

We use the function”duplicates()” from the pandas library,and find duplicate rows in a DataFrame.as show there is no duplication in the dataset.



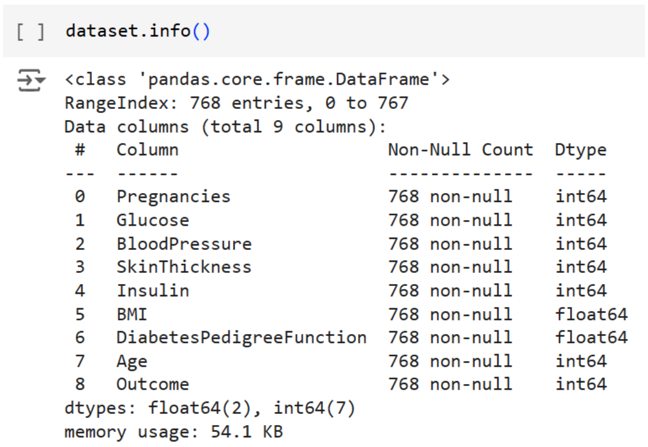
```
[ ] print("Missing values:")
    dataset.isnull().sum()
```

```
Missing values:
```

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

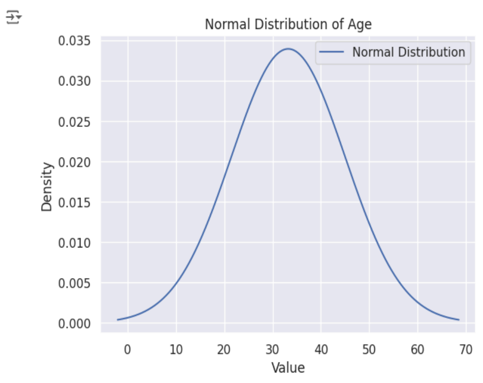
This function checks for missing values. as shown there is no missing value in the dataset.

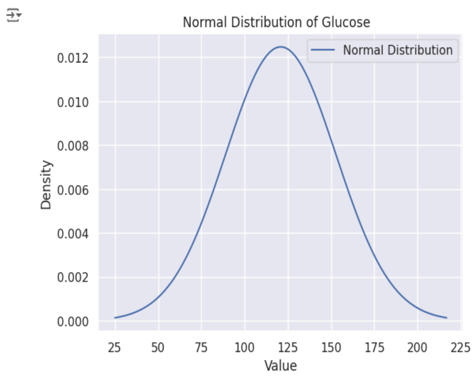


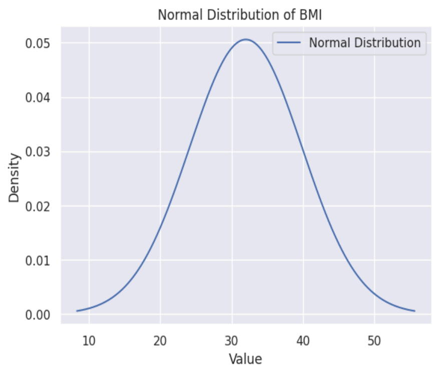
```
[ ] dataset.info()
```

```
<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
```

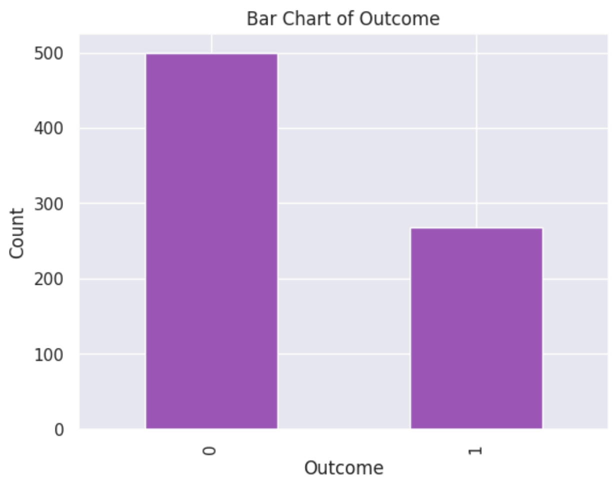
The info command is used to know the data types, and as shown to us, the data type used is integer and float.

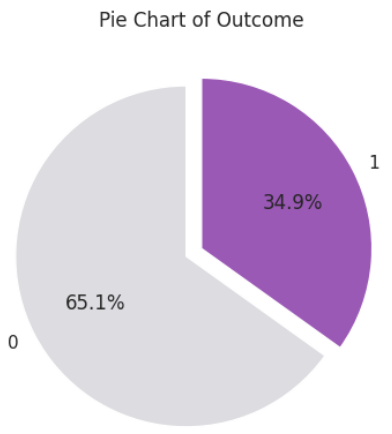




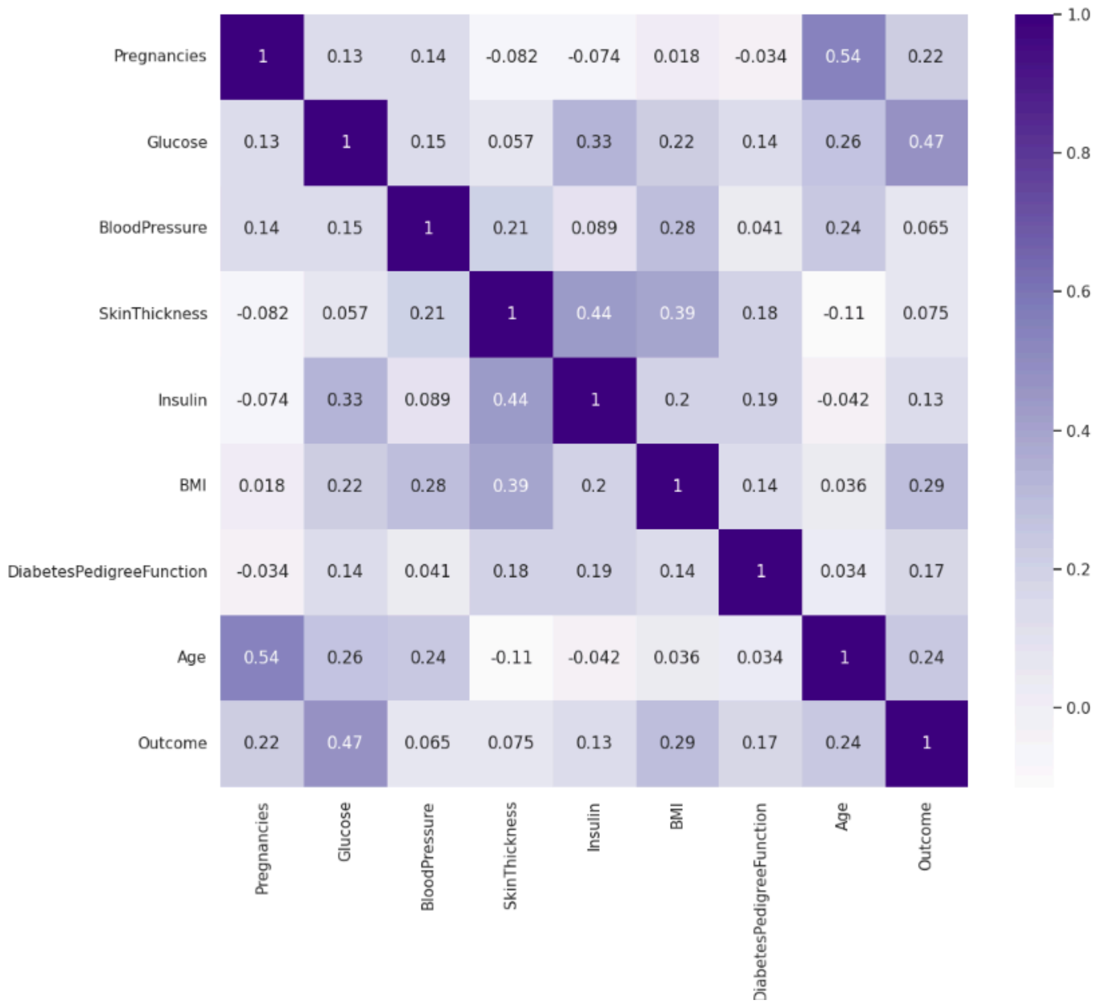


As shown in the figure we also see that the following columns: Age, Glucose, and BMI are normally distributed in the data.





In the graph we compare the number of people with diabetes to those without. The number 1 represents people with diabetes, while the number 0 represents people without diabetes. The graph shows that the percentage of people without diabetes is the largest, and The pie chart represents the percentage.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
SkinThickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
Insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

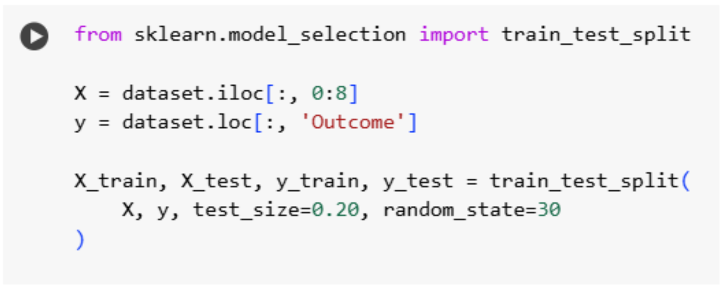
The highest correlation with the (Outcome) (diabetes presence) is with Glucose levels (0.47), indicating that glucose levels are one of the most important factors in predicting diabetes

From the figure we conclude that the most important feature is Glucose.

3- Naive Bayes Mode

We used the Naive Bayesian classification algorithm, which is an algorithm based on Bayes' theorem to calculate the probability that a given item belongs to a class based on its features. The algorithm calculates conditional probabilities for each feature in the data and uses these values to estimate the most likely class for the new item. It is simple and fast.

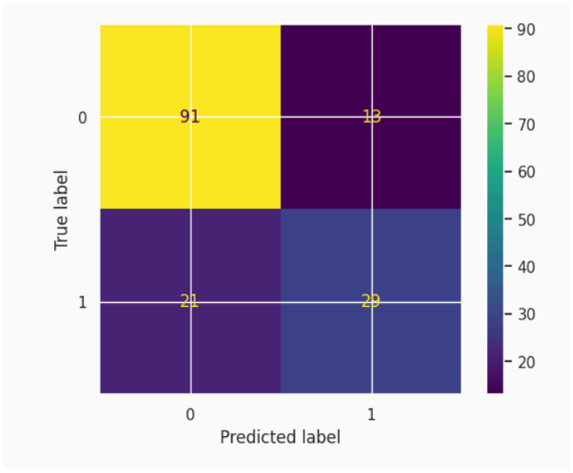
Model 1:

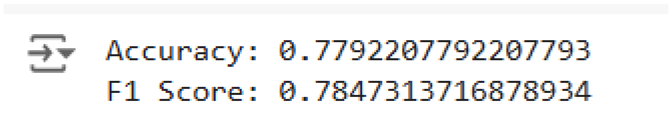


```
from sklearn.model_selection import train_test_split

X = dataset.iloc[:, 0:8]
y = dataset.loc[:, 'Outcome']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=30
)
```

The dataset is divided into two parts: the feature matrix (X), which contains the features that will be used for prediction, and the response vector (y), which represents the dependent variable (the outcome). The columns in the feature matrix (X) represent independent variables, such as pregnancy, glucose, blood pressure, skin thickness, insulin, BMI, diabetes function, and age

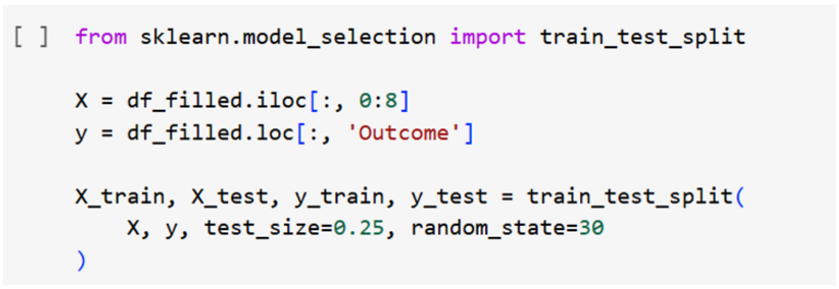
Using train_test_split, the dataset is divided into two groups:
X_train and y_train: the training set (80% of the data).
X_test and y_test: the testing set (20% of the data).
To ensure consistent predictions and a reliable split, the random_state=30 parameter is used, which guarantees that the data is split in a reproducible and stable way.



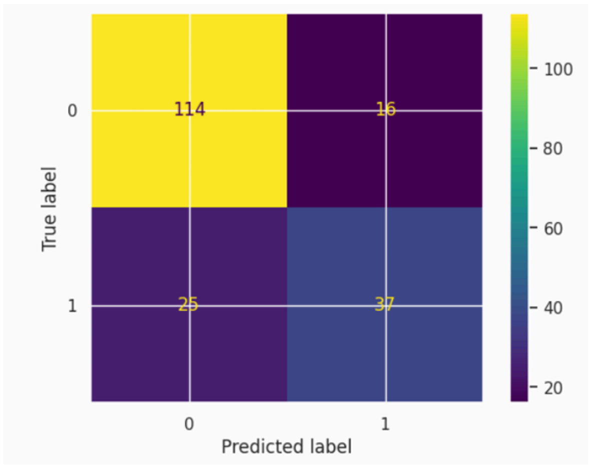


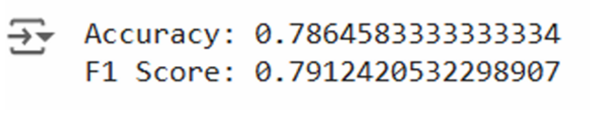
```
Accuracy: 0.7792207792207793
F1 Score: 0.7847313716878934
```

Model 2:



```
[ ] from sklearn.model_selection import train_test_split

    X = df_filled.iloc[:, 0:8]
    y = df_filled.loc[:, 'Outcome']

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=30
    )
```





```
Accuracy: 0.7864583333333334
F1 Score: 0.7912420532298907
```

We improved the model by adjusting the data split between training and test sets. 75% (0.75) of the data was allocated for training, while 25% (0.25) was used for testing.

Model 1	Accuracy: 77.92%	F-measure: 78.47%
Modal 2	Accuracy:78.65%	F-measure: 79.12%

Based on the results, we concluded that the second model performs better than the first model.