Name - Kumar Gaurav

Subject Name - MADSC204 DATA VISUALIZATION LAB

Table of Contents

Introduction

"Data visualization" is the process of representing data and informing it through visual elements such as graphs, charts, and maps. It can help to effectively communicate perception and the hidden patterns in complex data sets. This presentation of data in a visual format makes it easier for the users to quickly understand and interpret the data, which leads to better "decision-making". Data visualization can be useful for various fields, including business, healthcare, science, and education. It plays a vital role in data analysis as well as communication, allowing the users for identifying patterns, trends, and outliers that may otherwise be difficult to discern from raw data.

This research study is based on the data visualization of the diabetes dataset to predict the factors responsible for diabetes and its impacts. Here the project is all about predicting the patient of diabetes.

Problem statement

Diabetes is a ***"chronic metabolic disorder"*** characterized by the help of high blood sugar levels due to the inability of the body to produce and use insulin. It is a high range of health conditions that affect millions of people worldwide. This can lead to severe complications if left untreated (Yahyaoui, *et al,* 2019). Early detection of diabetes is crucial for proper management and prevention of complications. One of the most effective ways of early detection is by using predictive analytics and machine learning models to identify individuals at risk of developing diabetes. The goal of diabetes prediction in Python is to develop a model of machine learning that can accurately predict whether a person is at risk of developing diabetes based on certain risk factors like age, family history, obesity, physical inactivity, and high blood pressure. The model is basically trained on a dataset that includes a set of input features (such as age, BMI, glucose levels, etc.) and a corresponding output label indicating whether the person has diabetes or not. Diabetes prediction in Python is an important application of machine learning and predictive analytics that can help in the early detection and management of diabetes. By developing accurate and good models, professionals in healthcare can identify individuals at risk of developing diabetes. This will provide the necessary arbitration to prevent or manage the condition(Aada, *et al,* 2019).

Aim and objectives

Aim

The aim of diabetes prediction is to identify individuals at high risk for developing diabetes, facilitate early arbitration to prevent or delay the onset of diabetes, and improve overall health outcomes.

Objective

- To identify individuals who are at high possibility of evolving diabetes in the future.
- To take preventive measures to reduce the risk of difficulties related to the disease.
- to accurately predict the probability of an individual who has the chance to develop diabetes.

Result and Analysis

Diabetes prediction is the focus of this project because it is currently crucial to health. Python is utilized in this project to facilitate the application of machine learning. For any Machine Learning, there are a number of stages that must be taken in order to obtain precise and acceptable results. Many steps are also completed in this project. Here, numerous processes are carried out, including data collection, cleaning, transformation, imputation, classification, and accuracy prediction. (Hasan, *et al*, 2020).

Data collection is the first stage in machine learning. The data for this project was gathered through the Kaggle Platform. A CSV file containing patient data for diabetes was then displayed. Using adequate data is crucial to the success of machine learning. The following phase is data pre-processing, during which the data is processed. For the machine learning model to learn correctly and accurately, this phase is necessary. The next step is data cleaning which is used to clean the unnecessary and insufficient data. After that it checks the null values, then there is data transformation where the data is transformed into a float to integer values or categorical values to get a better and more accurate result. Then classification of the dataset, here Logistic Regression, Random Forest, and Decision Tree are used. After all these steps, the accuracy result got. The first step in machine learning is data acquisition. The Kaggle Platform was used to collect the data for this project. Afterward, a CSV file with patient information for diabetes was shown. Having enough data is essential for machine learning to be successful. Data pre-processing is the phase that comes after data processing. This step is essential for the machine learning model to learn reliably and appropriately.

```
▶ 0.0s
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")



from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, recall_score, f1_score
from sklearn.metrics import roc_auc_score, classification_report
```

Figure 1: Import Libraries

(Source: self-made)

Here, in the above figure, the libraries have been imported. The Numpy library has been imported as np and the Pandas library has been imported as PD. The pandas as pd, seaborn as sns, etc.

```
▶ 0.1s
# Reading Dataset
df = pd.read_csv("diabetes.csv")
df.sample(7)
```

Table Raw Visualize Statistics

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
91	4	123	80	15	176	32.0	
74	1	79	75	30	0	32.0	
209	7	184	84	33	0	35.5	
182	1	0	74	20	23	27.7	
33	6	92	92	0	0	19.9	
634	10	92	62	0	0	25.9	
4	0	137	40	35	168	43.1	

Figure 2: Read the Dataset

(Source: self-made)

Form the above figure it is showing "***read.csv***" file has been read. Then it has been stored in the variable named df. Then using sample() the top five columns of the dataset have been read.

```
# shape of dataset
df.shape
```

```
(768, 9)
```

```
0.1s
# Get information from Dataset
df.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
```

Figure 3: Shape of the dataset and Get Information

(Source: self-made)

In the above figure, the info() function is used to get the information of the dataset. With the help of this function, detailed information about the dataset can be accessed and the shape of the data also can be get.

```
0.0s
# get statistical information
df.describe().T
```

Table Raw Visualize Statistics

	count	mean	std	min	25%	50%	
Pre…	768.0	3.8450520833333335	3.3695780626988694	0.0	1.0	3.0	
Glu…	768.0	120.89453125	31.97261819513622	0.0	99.0	117.0	
Blo…	768.0	69.10546875	19.355807170644777	0.0	62.0	72.0	
Ski…	768.0	20.536458333333332	15.952217567727637	0.0	0.0	23.0	
Ins…	768.0	79.79947916666667	115.24400235133817	0.0	0.0	30.5	
BMI	768.0	31.992578124999998	7.884160320375446	0.0	27.3	32.0	
Dia…	768.0	0.471876302083333…	0.3313285950127749	0.078	0.24375	0.3725	
Age	768.0	33.240885416666664	11.760231540678685	21.0	24.0	29.0	
Out…	768.0	0.3489583333333333	0.476951377242798…	0.0	0.0	0.0	

Figure 4: Fetch Statistical Information

(Source: self-made)

Here, in the above figure ***"df.describe()"*** command has been used to fetch the statistical information.

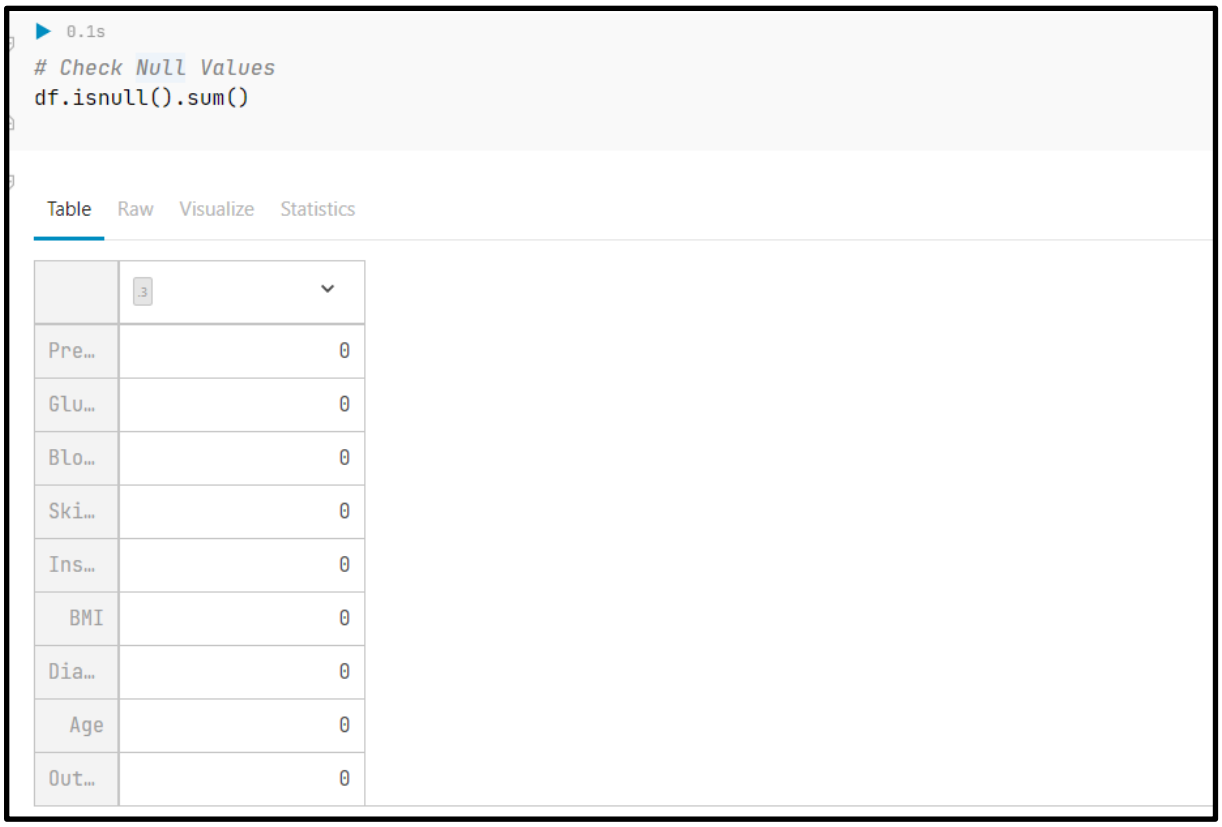

Figure 5: Check the Null Values

(Source: self-made)

The above figure shows the process of using the ***"isnull()"*** function which basically gives information about the columns that are contained null values in the set of data.

```
0.4s
# plot a counter plot
plt.figure(figsize=(10,6))
sns.countplot(x = 'Outcome',data = df, palette="Set3")
plt.show()
```

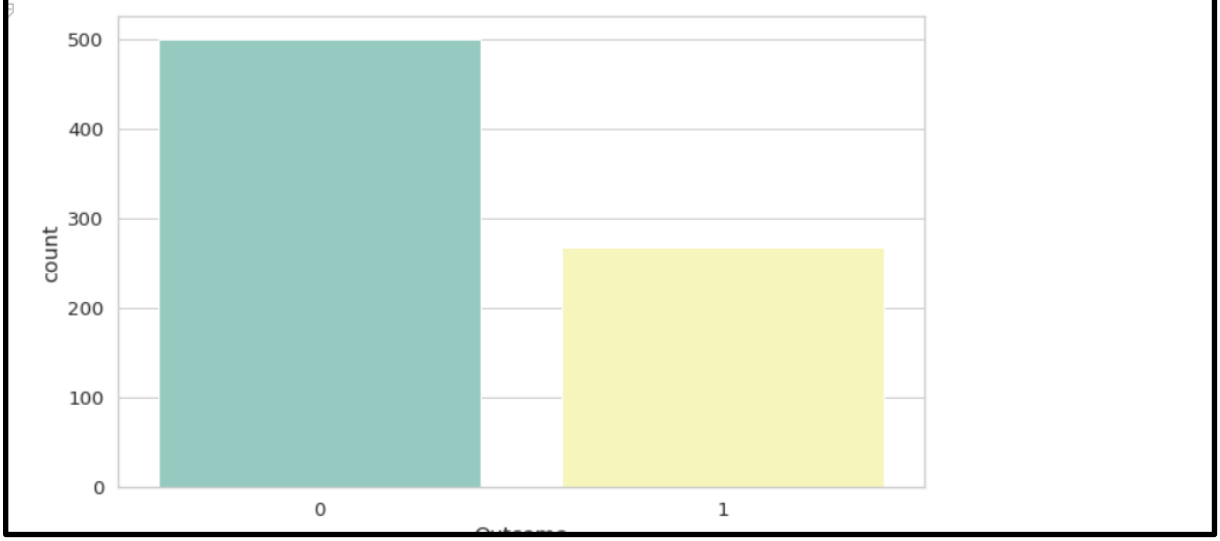

Figure 6: Plot Counter Plot

(Source: self-made)

The above figure, it is showing the counter plot. In diabetes prediction, a counter plot is used to visualize the distribution of diabetes-positive and diabetes-negative individuals for different levels of a given risk factors, such as age or BMI, allowing for insights into the relationship between the predictor and the target variable.

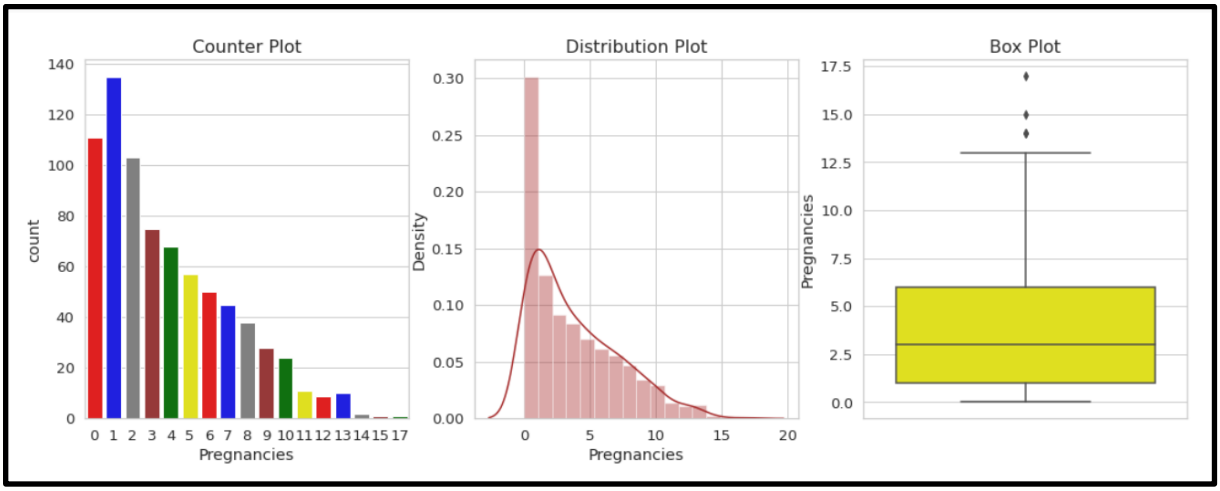

Figure 7: Counter Plot, Distribution plot, and Box plot of Pregnancies

(Source: self-made)

Here, in the above figure, the counter plot, distribution plot, and box plot have been shown for pregnancies.

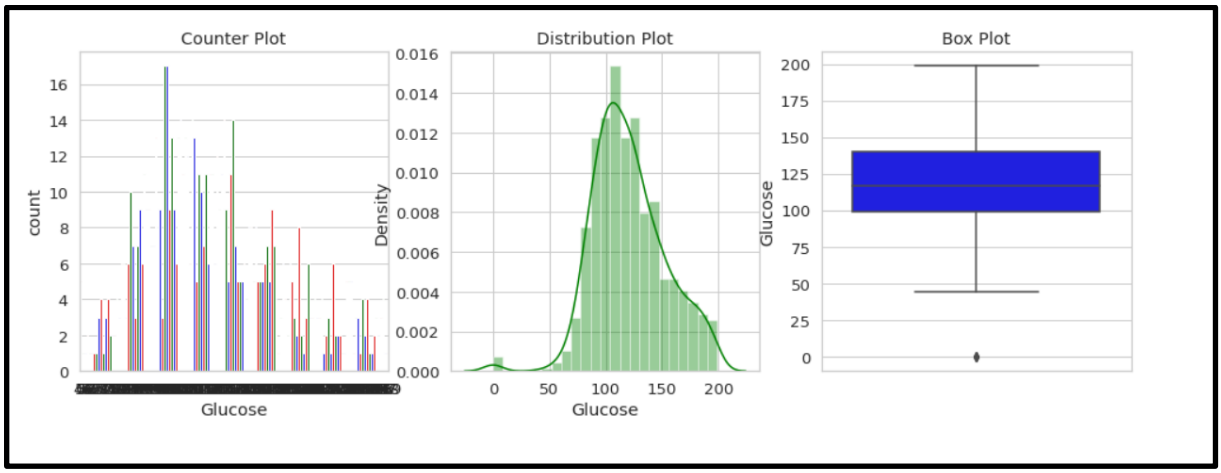

Figure 8: Counter Plot, Distribution plot, and Box plot of Glucose

(Source: self-made)

Here, in the above figure, the counter plot, distribution plot, and box plot have been shown for Glucose.

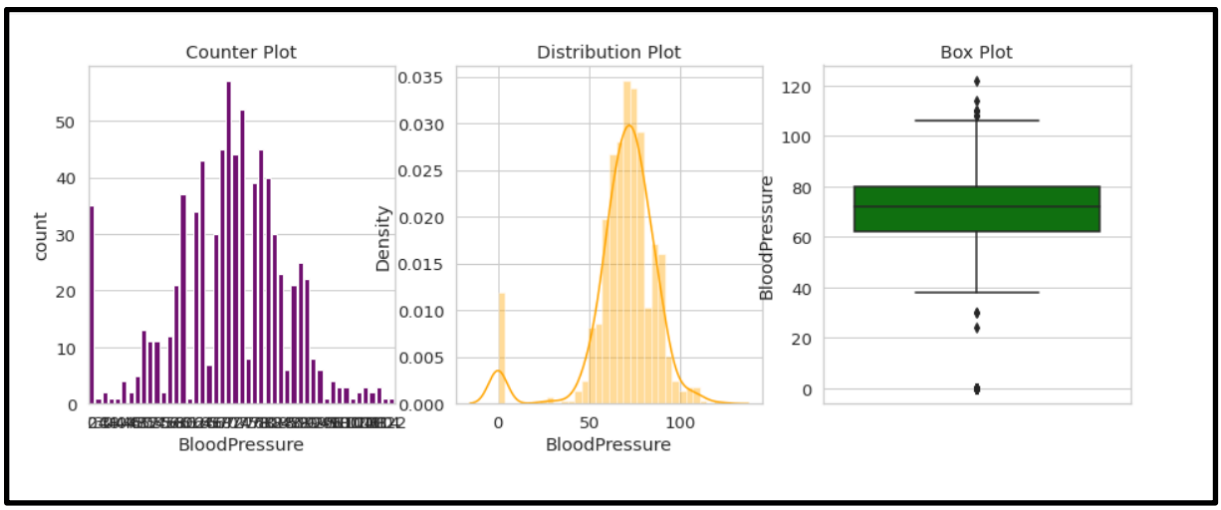

Figure 9: Counter Plot, Distribution plot, and Box plot of Blood Pressure

(Source: self-made)

In the above figure, the counter plot, distribution plot, and box plot have been shown for blood pressure.

```
# HeadMap
fig, ax = plt.subplots(figsize=(15,10))

corr = df.corr()

cmap = sns.diverging_palette(220, 10, as_cmap=True)

heatmap = sns.heatmap(corr, annot=True, cmap=cmap)

plt.show()
```

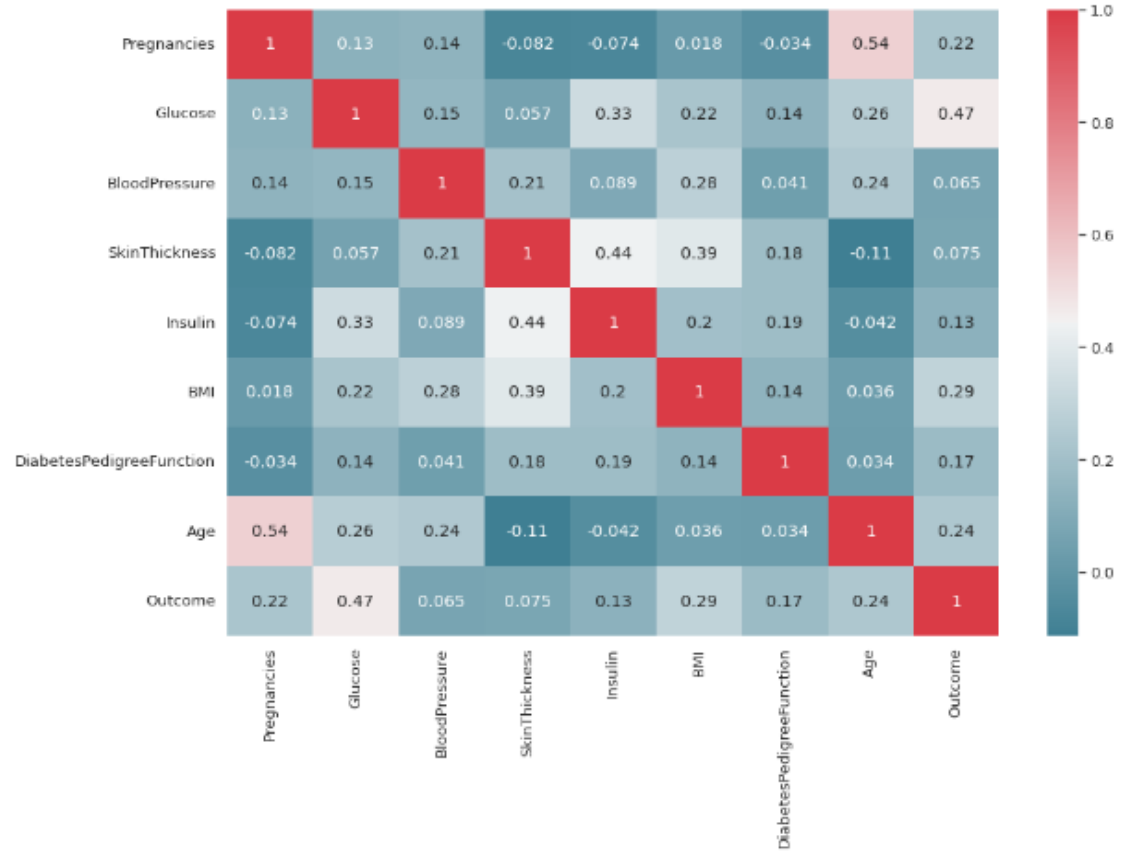

Figure 10: HeatMap

(Source: self-made)

In the above figure, the heatmap has been shown here. It provides a visual summary of the strength and direction of the relationships between different risk factors and the target variable. This can help identify the most important predictors of diabetes and inform feature selection for the machine learning model.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42, stratify = y )
print(" Training dataset: ", X_train.shape)
print(" Testing dataset: ", X_test.shape)
```

```
 Training dataset:  (537, 8)
 Testing dataset:  (231, 8)
```

Figure 11: Training and Testing Dataset

(Source: self-made)

Here, in the above figure, the dataset has been trained and tested.

```
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression()
```

```
▷ 0.1s
lr_model.fit(X_train_scaled, y_train)
y_predict = lr_model.predict(X_test_scaled)
```

```
accuracy = accuracy_score(y_test,y_predict)
recall = recall_score(y_test,y_predict)
f1 = f1_score(y_test,y_predict)

print("Accuracy Score: ", accuracy)
print("Recall Score: ", recall)
print("F1 Score: ", f1)
```

```
Accuracy Score:  0.7402597402597403
Recall Score:  0.5308641975308642
F1 Score:  0.589041095890411
```

Figure 12: Logistic Regression

(Source: self-made)

Here, the Logistic Regression has been done and the accuracy score of the Logistic Regression is 0.74.

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(max_depth=5, random_state=42)
# scalling doest affect decision tree
dt_model.fit(X_test, y_test)
y_pred = dt_model.predict(X_test)
```

```
▷ 0.1s
dt_accuracy = accuracy_score(y_test,y_pred)
dt_recall = recall_score(y_test,y_pred)
dt_f1 = f1_score(y_test,y_pred)

print("Accuracy: ", dt_accuracy)
print("Recall: ", dt_recall)
print("F1: ", dt_f1)
```

```
Accuracy:  0.9047619047619048
Recall:  0.8765432098765432
F1:  0.8658536585365854
```

Figure 13: Decision Tree

(Source: self-made)

Here, the Decision Tree has been done and the accuracy score of the Decision Tree is 0.90.

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=70, random_state=32)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
```

```
▷ 0.1s
rf_accuracy = accuracy_score(y_test,y_pred)
rf_recall = recall_score(y_test,y_pred)
rf_f1 = f1_score(y_test,y_pred)

print("Accuracy: ", rf_accuracy)
print("Recall: ", rf_recall)
print("F1: ", rf_f1)
```

```
Accuracy:  0.7359307359307359
Recall:  0.5308641975308642
F1:  0.5850340136054422
```

Figure 14: Random Forest

(Source: self-made)

Here, the Random Forest has been done and the accuracy score of the Random Forest is 0.73.

```
▷ 0.1s
# print the accuracy scores
print("Random Forest Accuracy:", rf_accuracy)
print("Decision Tree Accuracy:", dt_accuracy)
print("Logistic Regression Accuracy:", accuracy)


Random Forest Accuracy: 0.7359307359307359
Decision Tree Accuracy: 0.9047619047619048
Logistic Regression Accuracy: 0.7402597402597403
```

Figure 15: Accuracy of Random Forest, Decision Tree, and Logistic Regression

(Source: self-made)

Here, the accuracy score of Random Forest, Decision Tree, and Logistic Regression has been done with the help of the ***"print"*** function. Here, the decision tree is the most accurate classification for this diabetes prediction.

```
▷ 0.2s
y_predict = dt_model.predict([[1,148,72,35,79.799,33.6,0.627,50]])
print(y_predict)
if y_predict==1:
    print("Diabetic")
else:
    print("Non Diabetic")


[1]
Diabetic
```

Figure 16: Prediction of whether the patient is diabetic or not

(Source: Created by the learner)

Here, the above figure shows whether the patients are diabetic or not with the help of the ***"if-else"*** function.

Conclusion

In recent years, machine learning has been used for the prediction and diagnosis of diabetes. ML algorithms have proven to be effective in predicting diabetes based on patient data such as

blood glucose levels, body mass index, and family history. In recent years, machine learning has been used to predict and diagnosis of diabetes. The algorithms of ML have proven to be effectual in predicting diabetes based on the data of patients such as blood glucose levels, body mass index, and family history. ML-based diabetes prediction has several advantages, like quick detection of the disease, reduced medical costs, and improved patient outcomes. The early detection can guide to timely intervention and management of diabetes, which can prevent complications and improve the quality of life for patients.

In conclusion, ML has shown promise in predicting diabetes. It is used in healthcare likely to continue to grow in the future. However, further research is needed to refine the accuracy of the algorithms and address any concerns about bias and exploitability.

Reference

Yahyaoui, A., Jamil, A., Rasheed, J. and Yesiltepe, M., 2019, November. A decision support system for diabetes prediction using machine learning and deep learning techniques. In *2019 1st International informatics and software engineering conference (UBMYK)* (pp. 1-4). IEEE. **https://openaccess.izu.edu.tr/xmlui/bitstream/handle/20.500.12436/1906/A_Decision_Support_System_for_Diabetes_Prediction_Using_Machine_Learning_and_Deep_Learning_Techniques.pdf?sequence=1&isAllowed=y**

Hasan, M.K., Alam, M.A., Das, D., Hossain, E. and Hasan, M., 2020. Diabetes prediction using ensembling of different machine learning classifiers. *IEEE Access*, *8*, pp.76516-76531.

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9076634

Ramesh, J., Aburukba, R. and Sagahyroon, A., 2021. A remote healthcare monitoring framework for diabetes prediction using machine learning. *Healthcare Technology Letters*, *8*(3), pp.45-57.
https://ietresearch.onlinelibrary.wiley.com/doi/pdfdirect/10.1049/htl2.12010

Ayon, S.I. and Islam, M.M., 2019. Diabetes prediction: a deep learning approach. *International Journal of Information Engineering and Electronic Business*, *12*(2), p.21.
https://j.mecs-press.net/ijieeb/ijieeb-v11-n2/IJIEEB-V11-N2-3.pdf

Aada, A. and Tiwari, S., 2019. Predicting diabetes in medical datasets using machine learning techniques. *Int. J. Sci. Res. Eng. Trends*, *5*(2), pp.257-267.
https://ijsret.com/wp-content/uploads/2019/03/IJSRET_V5_issue2_154.pdf