# DIABETES PREDICTION

## Project Report

**SUPERVISOR**: Mrs Vandita Grover

SUBMITTED BY

**Mansi Sanwal- 19001570020**

**Ushna Khan- 19001570053**

2022

Department of Computer Science

ACHARYA NARENDRA DEV

COLLEGE

# ACKNOWLEDGEMENT

We would like to express our sincere thanks and gratitude to Mrs Vandita Grover for letting us work on this project. We are very grateful to her for her support and guidance in completing this project.

We are thankful to our parents as well. We were able to successfully complete this project with the help of their guidance and support. Finally, we want to thank all our dear friends as well.

**Mansi Sanwal**

**(19001570020)**

**Ushna Khan**

**(19001570053)**

# ACHARYA NARENDRA DEV COLLEGE

# (University of Delhi)

## CERTIFICATE

This is to certify that Ms Mansi Sanwal and Ms Ushna Khan, students of the Department of Computer Science, Acharya Narendra Dev College, has undergone a Project work titled ‘Diabetes Prediction’.

Supervisor Mrs Vandita Grover

# Table of Contents

# Chapter 1

## PROBLEM STATEMENT

Chronic disease lasts a year or more and requires ongoing medical attention and they have a huge impact on one's life [1]. Diabetes is a chronic disease that occurs when the pancreas refuses to generate enough insulin and the level of sugar in the bloodstream increases drastically [2]. Today a large part of the world's population is suffering from diabetes. Diabetes develops at a younger age in the Asian population than in the white population and the complexity the disease brings with itself impacts the quality of the life of an individual [3]. Diabetes depends on several factors like glucose level, BMI, and so on. With the evolution in Data Science, it has become possible to predict the presence of certain diseases in the patient. This project aims to predict diabetes in the patient with the help of the person's data like BMI, glucose level, number of pregnancies, etc. This module is specifically designed for women patients as their health conditions are mostly unknown due to lack of care and knowledge and this can help them to rectify diabetes on an early basis and then one can start their treatment after that.

# Chapter 2

# DATA MINING TECHNIQUES

The use of enhanced data analysis methods to identify previously unknown, valid patterns and linkages in large data sets is known as data mining. Statistical models, machine learning approaches, and mathematical algorithms. Thus, data mining incorporates analysis and prediction. Various main data mining techniques, such as association, classification, clustering, prediction, sequential patterns, and regression, have been developed and applied in current data mining initiatives.[4]

### 2.1.1 Classification

It's a data analysis task, which entails developing a model that characterizes and differentiates data classes. On the basis of a training set of data comprising observations and whose categories membership is known, classification is the issue of determining which of a set of categories a new observation belongs to.[5]

### 2.1.2 Association

The discovery of interesting connections and links among vast collections of data objects is made possible by association rule mining. This rule indicates how often an item set appears in a transaction.[6]

### 2.1.3 Clustering

Clustering is the process of dividing a set of abstract items into classes of comparable things. It aids marketers in identifying various groups within their consumer base and allowing them to describe those groups using purchase habits.[7]

## 2.2 Data mining technique used for this project:

### Classification

The classification technique has been used in this project for predicting if the patient falls into the diabetic or non-diabetic class. KNN, Naive Bayes and Decision tree classifiers have been used to yield the optimal model with the highest accuracy and precision.

### 2.2.1 K-NN

The KNN algorithm is a supervised machine learning method that may be used to solve both classification and regression predicting problems. However, in industry, it is mostly utilized to solve classification and prediction issues.[9]

### 2.2.2 Naive Bayes

The Bayes' Theorem is used to create a set of classification algorithms known as Naive Bayes classifiers. It is a family of algorithms that share a similar idea, namely that each pair of characteristics being categorized is independent of the others.[11]

### 2.2.3 Decision Tree

Decision trees are one of the most common and widely used classification techniques. The objective is to learn basic decision rules from data attributes to develop a model that predicts the value of a target variable.
Decision trees use gini or entropy as a criterion to select the features on which splitting is to be performed. As the name suggests, the Decision tree has root nodes, leaf nodes and branches.

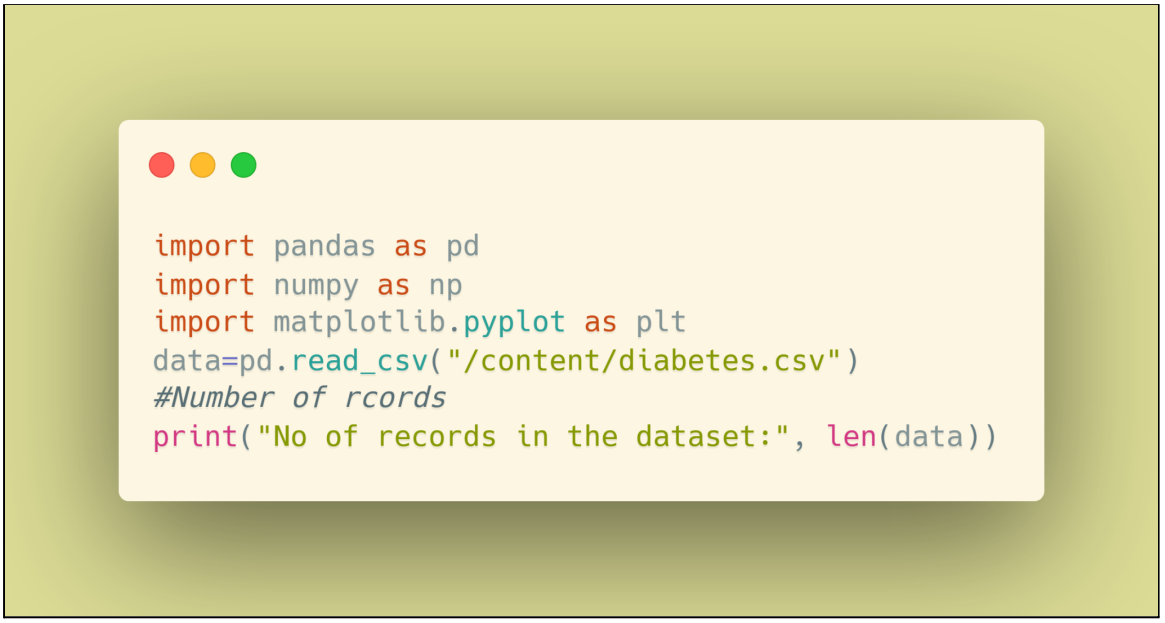
# Chapter 3

# Dataset Description

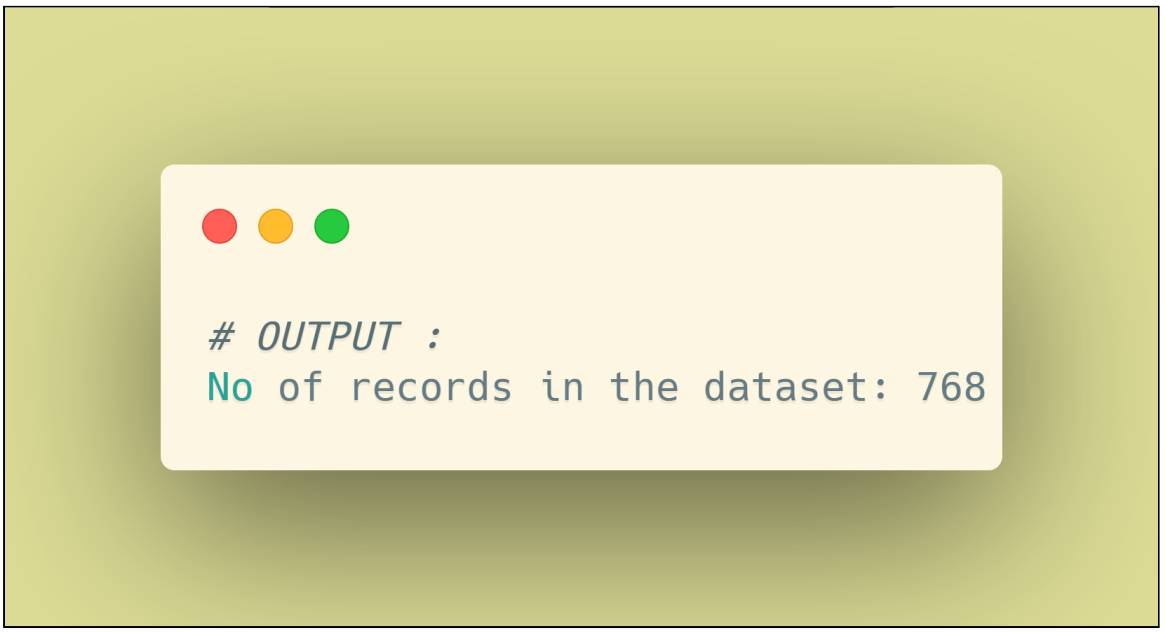
## 3.1 Dataset

The National Institute of Diabetes and Digestive and Kidney Diseases provided this data. The dataset's goal is to diagnose whether a patient has diabetes using diagnostic metrics provided in the collection. The selection of these cases from a wider database was subjected to many limitations. All of the patients at this clinic are Pima Indian women who are at least 21 years old. There are various medical predictor factors in the dataset, as well as one goal variable, Outcome. The number of pregnancies the patient has had, their BMI, insulin level, age, and other factors are all predictor variables.

### 3.1.1 Number of Records

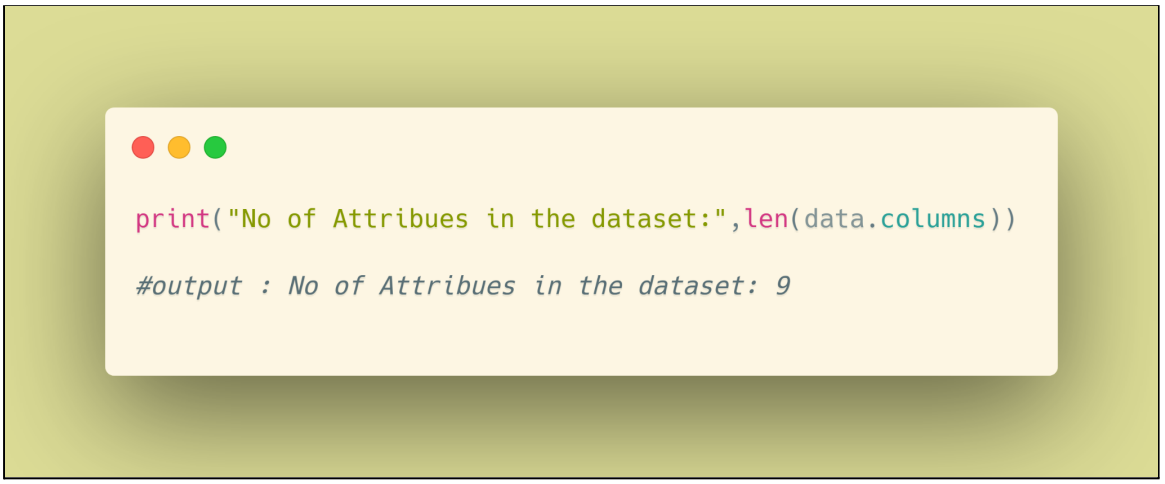


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("/content/diabetes.csv")
#Number of rcords
print("No of records in the dataset:", len(data))
```



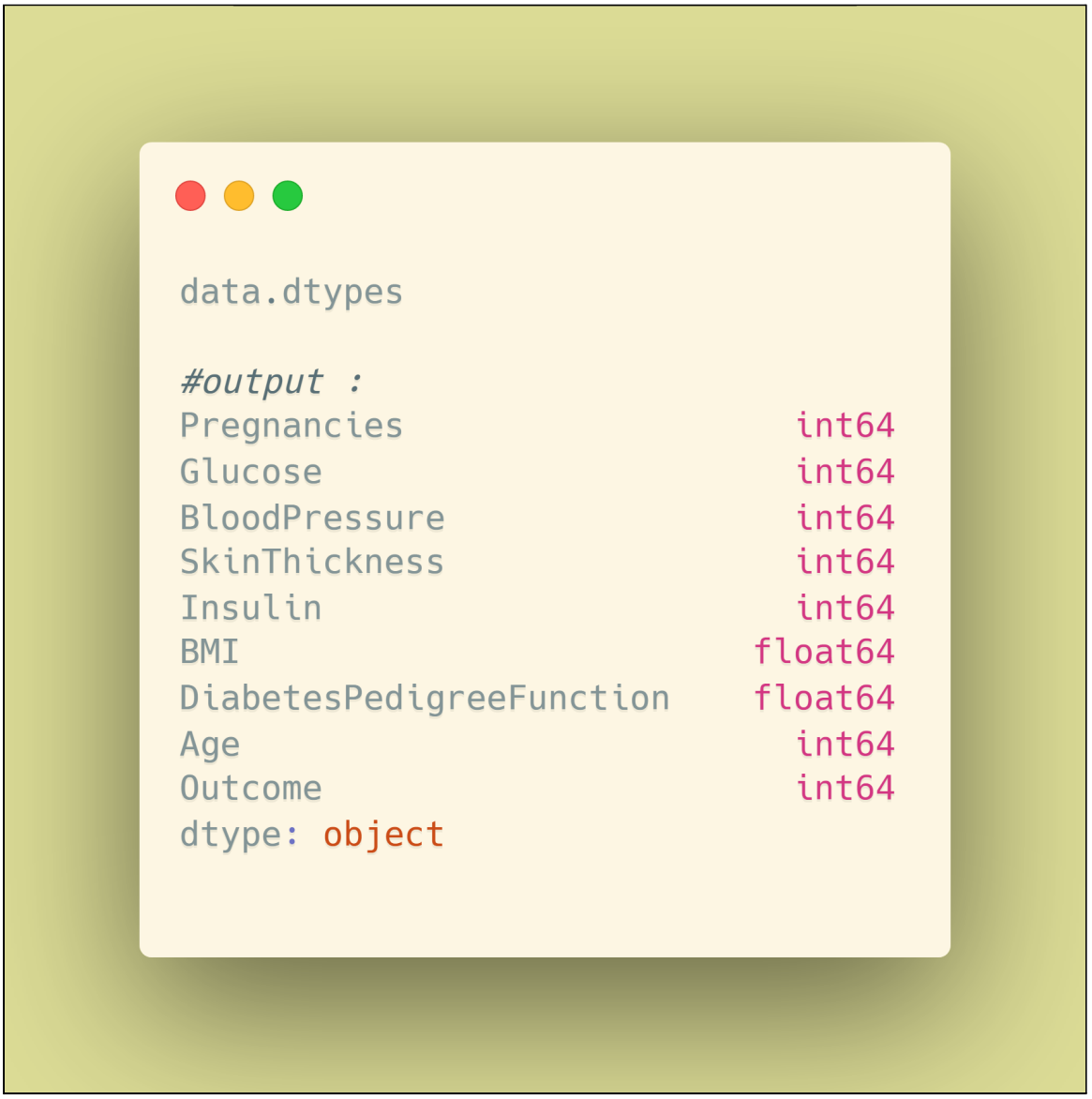

There are 768 records in the diabetes dataset.

### 3.1.2 Number of Attributes




There is a total of 9 attributes out of which one is the target variable and the rest are independent variables. The attributes have been described later in this section

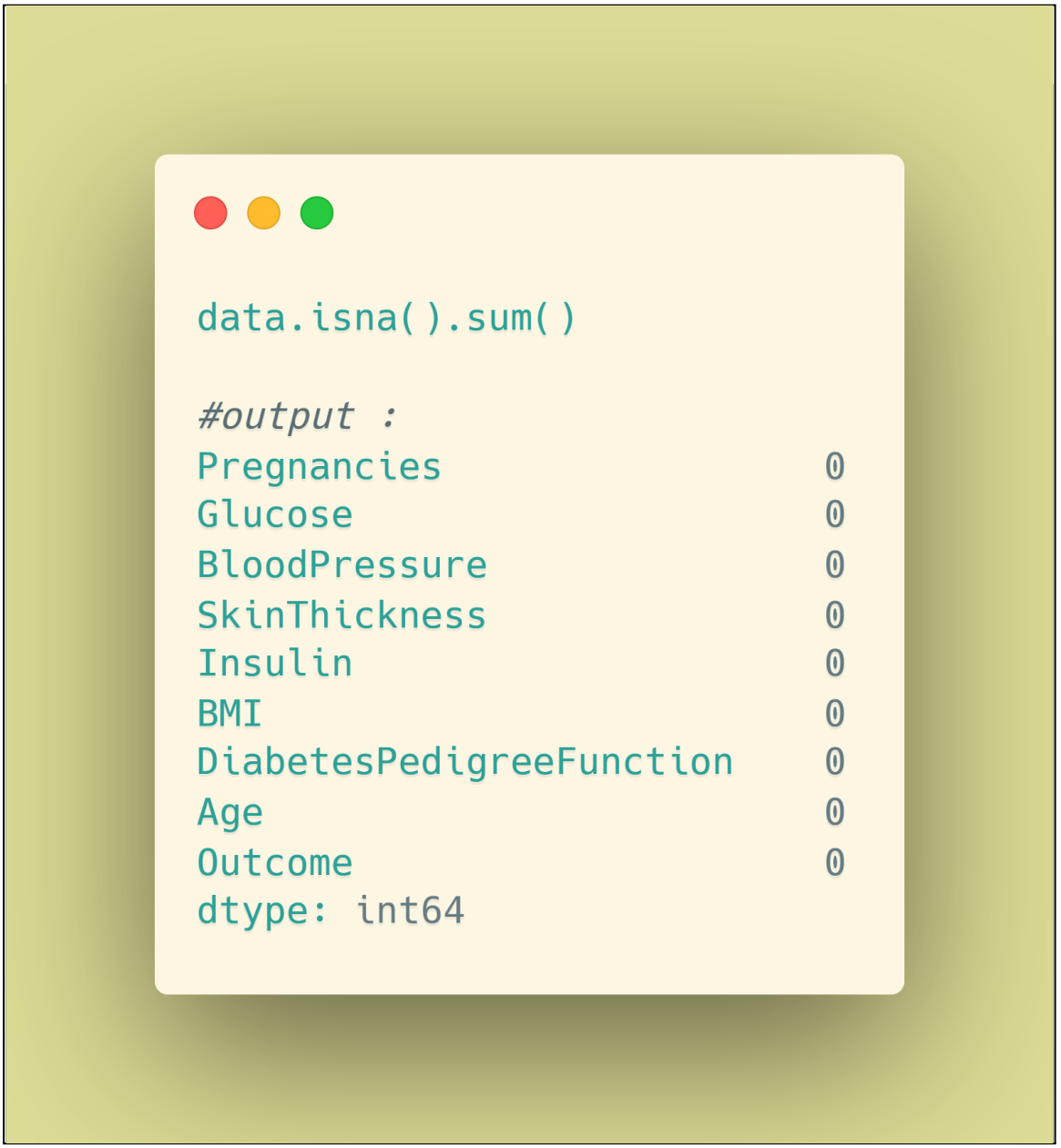
### 3.1.3 Types of Attributes




As can be seen, all the attributes are numerical with int and float values.

### 3.1.4 Missing Values or Nulls

```
data.isnull().sum()

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




There are no missing or null values in the dataset.

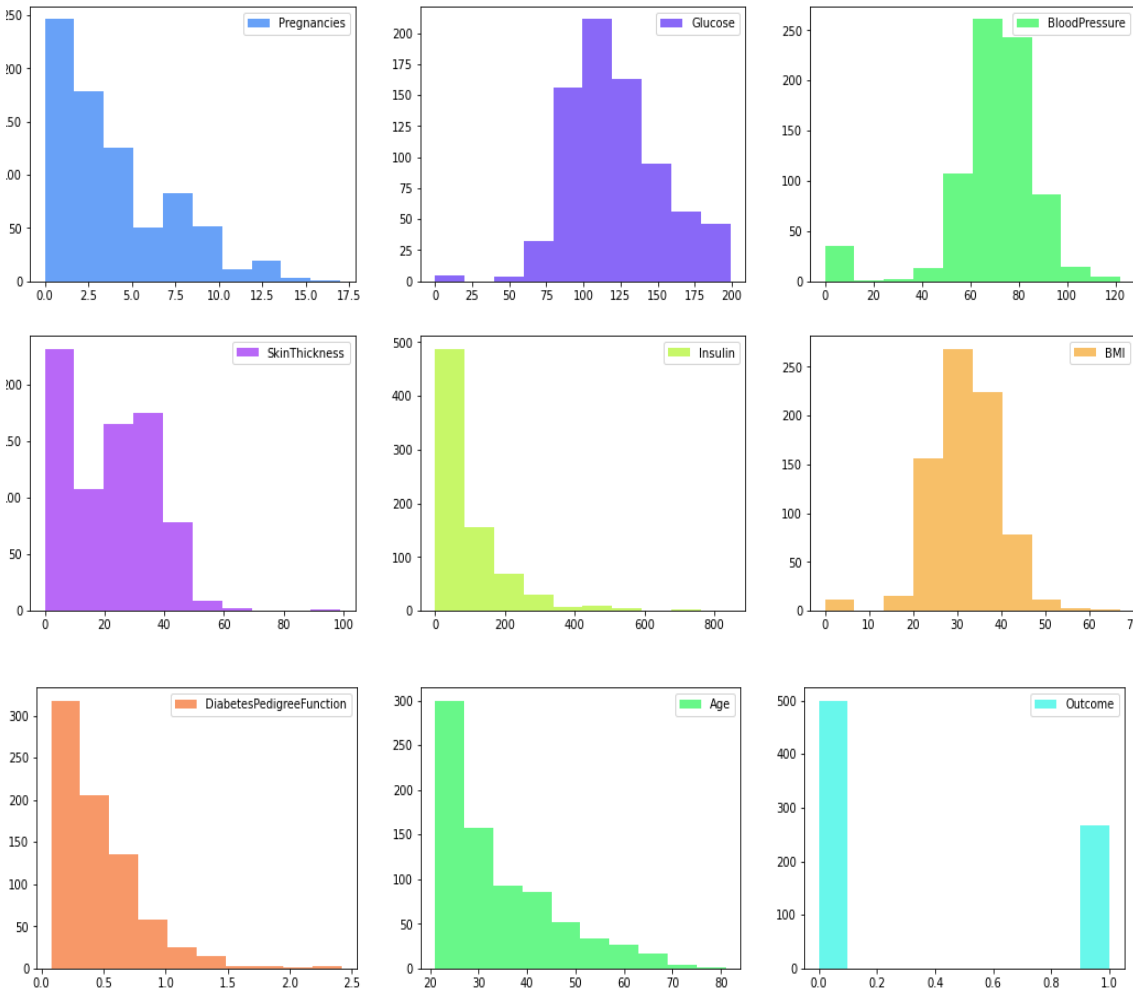
### 3.1.5 Attributes Description

1. **Pregnancies-** Number of times pregnant
2. **Glucose-** Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. **Blood pressure-** Diastolic blood pressure (mm Hg)
4. **SkinThickness-** Triceps skinfold thickness (mm)

5. **Insulin-** 2-Hour serum insulin (mu U/ml)
6. **BMI-** Body mass index (weight in kg/(height in m)^2)
7. **DiabetesPedigreeFunction-** Diabetes pedigree function
8. **Age-** Age (years)
9. **Outcome-** Class variable (0 or 1) 268 of 768 are 1(diabetic), the others are 0(non-diabetic)

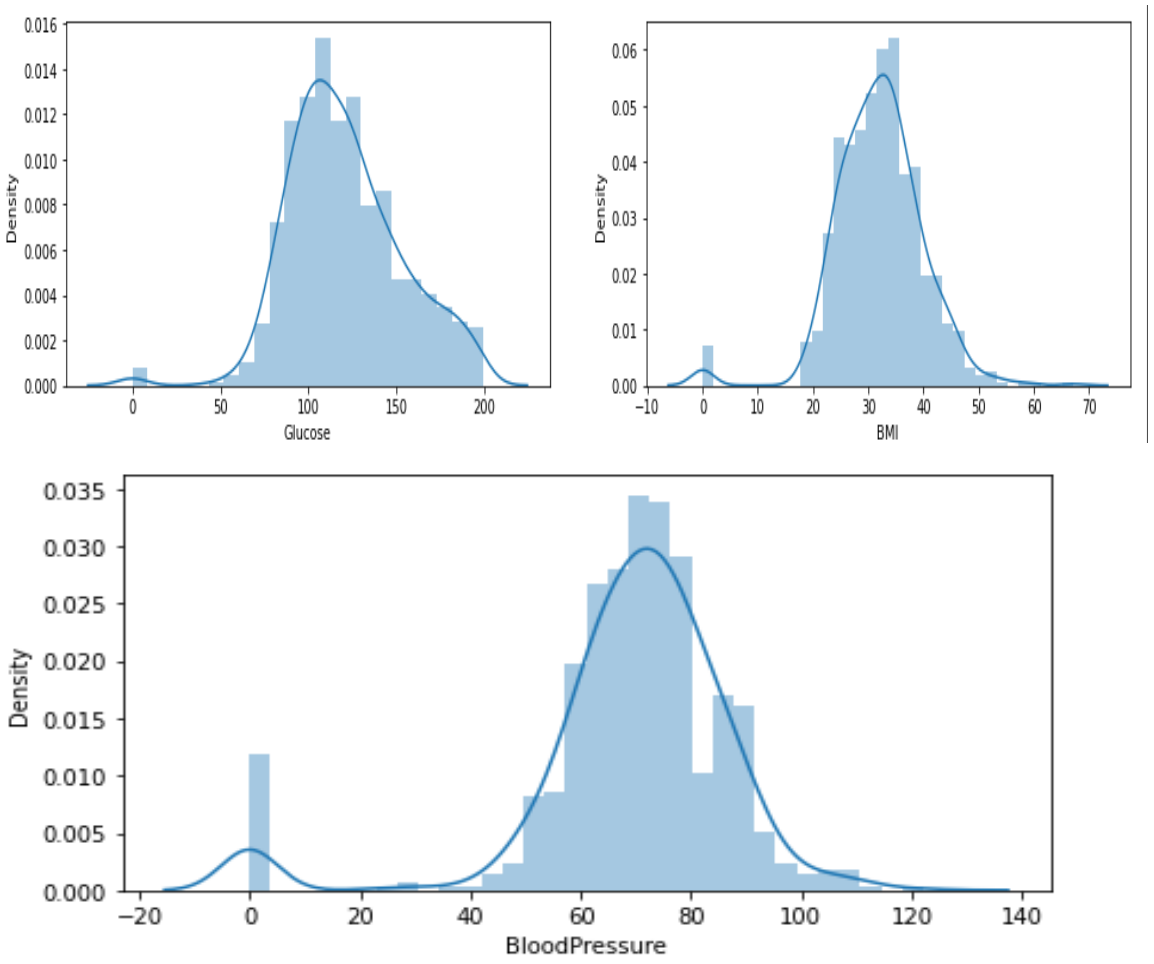
### 3.1.6 Distribution/Histograms

```
fig, axes = plt.subplots(nrows = 3, ncols =3, figsize = (18, 14))
colors=["#428af5","#6c42f5","#42f566","#a742f5","#b9f542","#f5b042","#f57e42","#42f56c","#42f5e6"]
for index, column in enumerate(data.columns):
    ax = axes.flatten()[index]
    ax.hist(data[column], color = colors[index], label = column,alpha=0.8)
    ax.legend(loc = "best")
plt.suptitle("Distribution", size = 20)
plt.show()
```




For features like Glucose, Blood Pressure, Skin Thickness. BMI, etc. the distribution seems almost Gaussian. Though some outliers can be detected in a few features.
The outcome feature suggests the data is highly imbalanced as the number of patients belonging to the non-diabetic class exceeds the diabetic class.

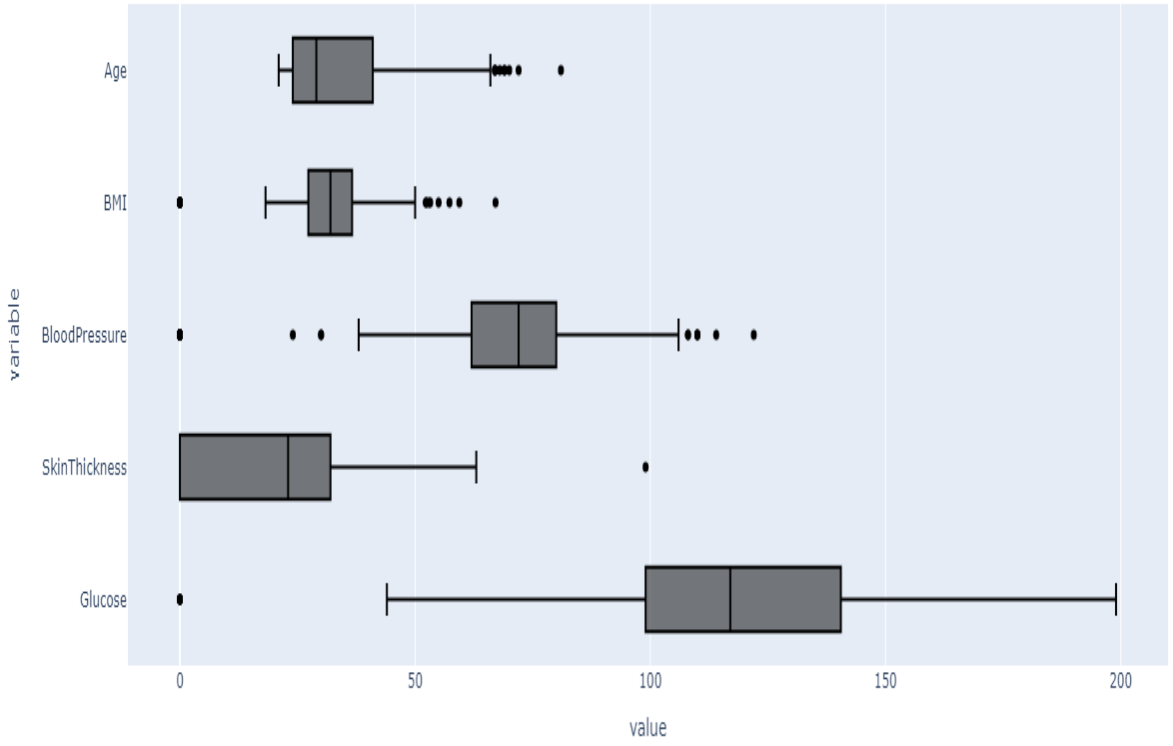
```
fig,ax=plt.subplots(2,2,figsize=(16,8))
sns.distplot(data['Glucose'],ax=ax[0,0])
sns.distplot(data['BMI'],ax=ax[0,1])
sns.distplot(data['BloodPressure'],ax=ax[1,0])
plt.show()
```

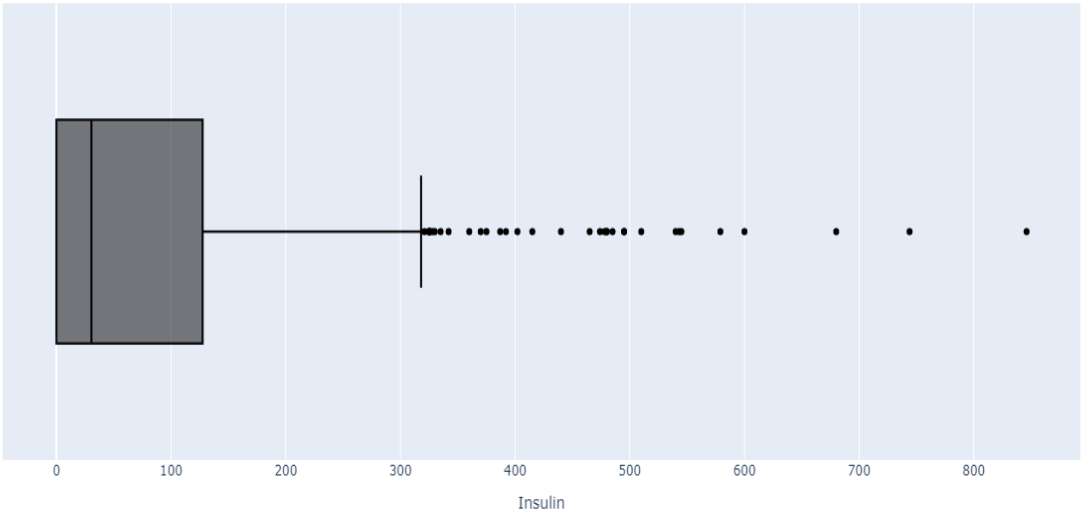


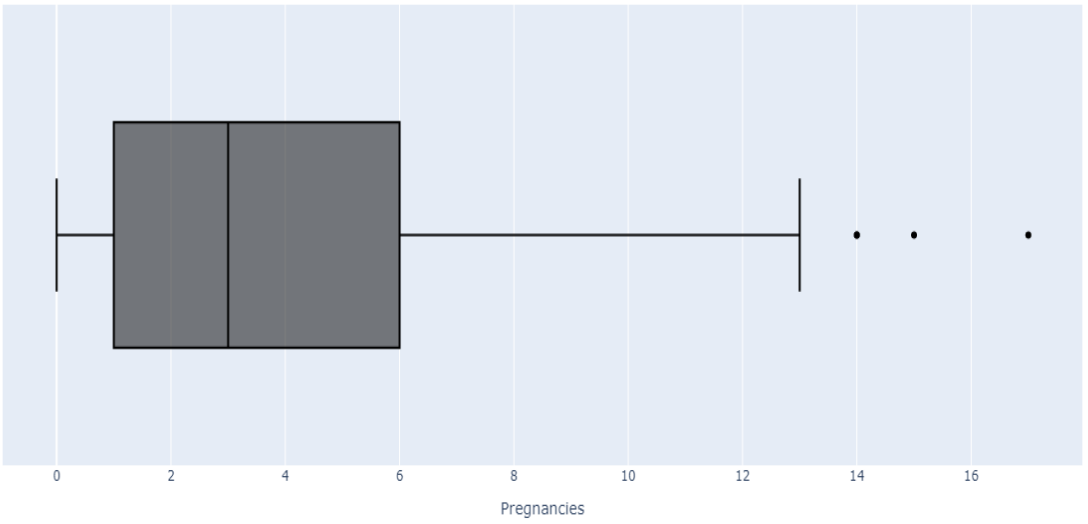

The distplot of features Glucose, BMI and Blood Pressure further confirms the normal distribution of the data in these features.

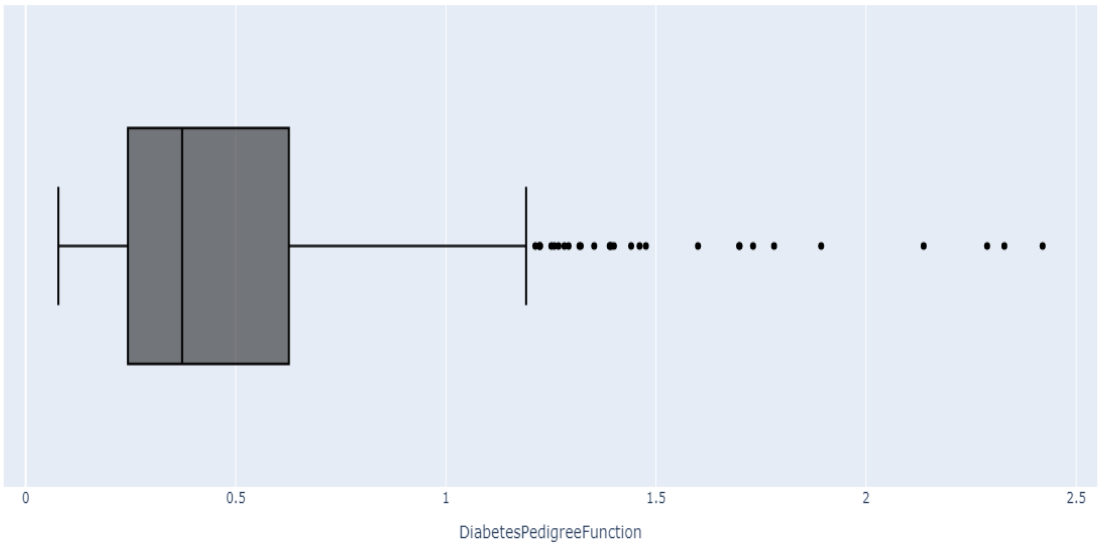
## 3.1.7 Detecting Outliers

```
import plotly.express as px
px.box(data_frame=data,x=
['Glucose','SkinThickness','BloodPressure','BMI','Age'],color_discrete_sequence=['black'])
px.box(data_frame=data,x='Insulin',color_discrete_sequence=['black'])
px.box(data_frame=data,x='Pregnancies',color_discrete_sequence=['black'])
px.box(data_frame=data,x='DiabetesPedigreeFunction',color_discrete_sequence=['black'])
```




0
100
200
300
400
500
600
700
800
Insulin


0
2
4
6
8
10
12
14
16
Pregnancies




BMI, Glucose, Insulin and Blood Pressure exhibit values near to zero which clearly implies that these values are outliers or maybe the missing values as no person can have 0 BMI or Glucose.

For features like Diabetes Pedigree, Insulin and Blood Pressure a high number of outliers can be detected while other features like Pregnancies, Age, Glucose and Skin Thickness consist of fewer outliers.

# Chapter 4

# DATA PREPROCESSING

Data preprocessing is a critical step in any data mining process since it directly affects the project's success rate. Because data in the actual world is filthy, this minimizes the complexity of the data under investigation.
If there are missing attributes, attribute values, noise or outliers, and duplicate or incorrect data, the data is considered to be dirty. If any of these are present, the quality of the findings will suffer.[14]
The purpose of data preprocessing is to ensure that the data is of good quality. The following criteria can be used to assess quality:
Accuracy: To determine whether or not the data entered is correct.
Completeness: To determine if the data is accessible or not recorded.
Consistency: Determine if the same data is retained in all locations that match or do not match.

## Major Tasks in Data Preprocessing:

1. Data cleaning

   The practice of removing erroneous, incomplete, and inaccurate data from datasets, as well as replacing missing information, is known as data cleaning.

2. Data integration

   Combining several sources into a single dataset. One of the most important aspects of data management is data integration.

3. Data reduction

   This procedure aids in the decrease of data volume, making analysis easier while producing the same or almost the same results. This decrease also aids in the reduction of storage space**.**

4. Data transformation

   Data transformation is the process of changing the format or organization of data. Depending on the needs, this process might be easy or difficult. [16]

## 4.1 HANDLING NULL VALUES

The null values or the missing values in the dataset can reduce the accuracy of the models. Therefore it is highly important to detect and handle null values before training the models. The data pre-processing includes the removal of any such value which might affect the accuracy of the model negatively. It is a good practice to remove any special character like “?”,”#”, etc. to retain the purity of data. The data in real life is messy and need cleaning before actually undergoing training.
There are numerous ways which one might consider removing null values according to the datasets.

1. Deleting Rows with missing values
   Delete the rows or columns with null values to deal with missing values. If more than half of the rows in a column are null, the column might be eliminated entirely. Rows with null values in one or more columns can be discarded as well. The only downside of this approach is when the dataset consists of fewer records then deleting rows might affect the accuracy of the models.

2. Impute missing values with Mean/Median
   Columns containing numeric continuous values in the dataset can be replaced with the mean, median, or mode of the remaining values in the column.
   While most of the time this approach is used but extra care must be taken when dealing with data with heavy outliers that might affect the mean in the numerical columns.

3. Imputation method for categorical columns
   When missing values are found in categorical columns (string or numerical), the most common category might be used to fill in the gaps.

Null/Missing values in this project:

```
data=data[data['BloodPressure']!=0]
data_column=['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
for i in data_column:
   data[i] = data[i].replace(0, np.NaN)     #replacing all zeroes to nan

data.isna().sum()
# OUTPUT: Pregnancies                   0
Glucose                     5
BloodPressure               0
SkinThickness             194
Insulin                   339
BMI                         4
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Since a lot of values in the Blood Pressure feature consists of 0, the rows which contain non-zero blood pressure have been filtered out.
For features- Glucose, BMI, Skin Thickness and Insulin, the '0' values have been replaced with NaN.

```
data['Glucose'].fillna(data['Glucose'].mean(),inplace=True)
data['SkinThickness'].fillna(data['SkinThickness'].mean(),inplace=True)
data['Insulin'].fillna(data['Insulin'].median(),inplace=True)
data['BMI'].fillna(data['BMI'].mean(),inplace=True)
```

In Glucose, Skin Thickness and BMI the NaN values have been replaced with their mean as Skin Thickness had only one outlier(99), which didn't impact the mean of the column and Glucose didn't have any outlier beyond 200 so it is convenient to take their mean.
In Insulin the NaN values have been replaced with its mode.
In BMI there were not many outliers that might distort the mean, so its NaN values have been addressed by filling them with the mean of the entire column.

## 4.2 FEATURE SCALING

Feature scaling is a technique for normalizing a set of independent variables or data components. It is also known as data normalization in data processing and is usually done during the data preparation stage.

Let's have a look at the various ways of feature scaling. The following are the most popular approaches available:

### 4.2.1 Normalization

The simplest approach, also known as min-max scaling or min-max normalisation, consists of rescaling the range of features to scale the range in [0, 1]. The general normalisation formula is as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Here, max(x) and min(x) are the maximum and the minimum values of the feature respectively.

```
# Normalization

from sklearn.preprocessing import MinMaxScaler
data2=data.drop(columns=['Outcome'],axis=1)
mmc=MinMaxScaler()
mmc.fit(data2)
data2=mmc.transform(data2)
```

### 4.2.2 Standardization

Feature standardization reduces the variance and mean of each feature in the data to zero. The usual technique of computation is to get the distribution mean and standard deviation for each feature and then use the following formula to generate the new data point:

$$x' = \frac{x - \bar{x}}{\sigma}$$

Here, σ is the standard deviation of the feature vector, and x̄ is the average of the feature vector. [18]

```
# Standardization

from sklearn.preprocessing import StandardScaler
data2=data.drop(columns=['Outcome'],axis=1)
sc=StandardScaler()
data2=sc.fit_transform(data2.values)
```

### 4.2.3 Feature Scaling technique used in this project:

As most of the feature's distribution was seemed to be Gaussian, in the Diabetes dataset, the Standardization technique of feature scaling has been used.

```
# Standardization

from sklearn.preprocessing import StandardScaler
data2=data.drop(columns=['Outcome'],axis=1)
sc=StandardScaler()
data2=sc.fit_transform(data2.values)
```

## 4.3 FEATURE SELECTION AND CONVERSION

Feature Selection and conversion are yet another important step of data pre-processing.
Selecting features that best define the dataset is known as feature selection. While increasing the number of features in the dataset may improve its accuracy but too many features can do reverse. Selecting the right features and in the right numbers is important for making the best model.
In some datasets, features can be selected by using mere common sense or by having the domain knowledge of the dataset, in other cases feature selection is done using some pre-defined techniques like Filters Method, Embedded Method, Wrappers method, etc
Some Machine learning models work best with numerical data like KNN, Naive Bayes etc while others can do well with both categorical and numerical like Decision Trees.

In the Diabetes dataset used in this project, all the features are relevant to the problems and contribute to building the most optimal model on which people can rely.

### 4.3.1 Categorical to Numerical

Converting categorical data to numerical is a necessity for most of the models, which give the best results when fed with numerical data but it is not true for all the models. Some models work great with categorical data as well like the Decision tree classifier.

Since in the Diabetes dataset, there was no categorical data that could be

converted to numerical but for demonstration, a small data sample has been taken and converted into categorical as can be seen in the code below which later is converted to numerical.

```
# categorical to numerical

datax=data2.head(100)
datax['Outcome'].replace({0:"Non Diabetic",1:"Diabetic"},inplace=True)
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
datax['Outcome']=le.fit_transform(datax['Outcome'])
```

The outcome variable is changed to categorical data, the 0 represents Non-Diabetic and 1 represents Diabetic.
Now, the categorical column(Outcome) has been encoded into the numerical column.

## 4.3 DATA SAMPLING AND SUBSETTING

Part of analyzing data mining algorithms involves separating data into training and testing sets. When a data set is divided into a training set and a testing set, the majority of the data is used for training and a smaller amount is utilized for testing. One can test a model by generating predictions against the test set after it has been processed using the training set. It is simple to assess whether the model's estimations are right since the data in the testing set already has known values for the characteristic that one wish to forecast. [20]

### 4.3.1 TRAIN-TEST SPLIT

The dataset is split into two parts that are training set and the testing set. The training set is used to train the model. In the classification models, the training sets consist of the targeting variables as well.
The model is trained using the training dataset and is tested by using the testing dataset. While passing the testing dataset, the target values are not passed rather the model predicts the values and later the predicted values by the model are compared with the target values(of the testing data set).

In this project, 80 % of the training set and 20% of the testing set is used. The

ratios can vary depending on the size of the datasets.

```
X=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random
_state=1)
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

# Chapter 5

# Building Models

Numerous models can be built for the same problem but choosing the right model is what goes inside in finding the optimal solution for a problem. while some models can work best on a certain dataset and other models might be considered good for other sets of problems. There is no thumb rule as to which model is going to work best for a given dataset. Therefore it is important to build all the possible models and compute their accuracy score(or other measures) to select the best model for the dataset.

Since this project deals with the classification problem, all the classification models have been built for the best results.

## 5.1 Model1: K- Nearest Neighbors

K nearest model is one of the widely used classifiers that take the value of k ( number of neighbors) for prediction. The value of K must be chosen wisely for it can drastically impact the accuracy of the model (positively or negatively).

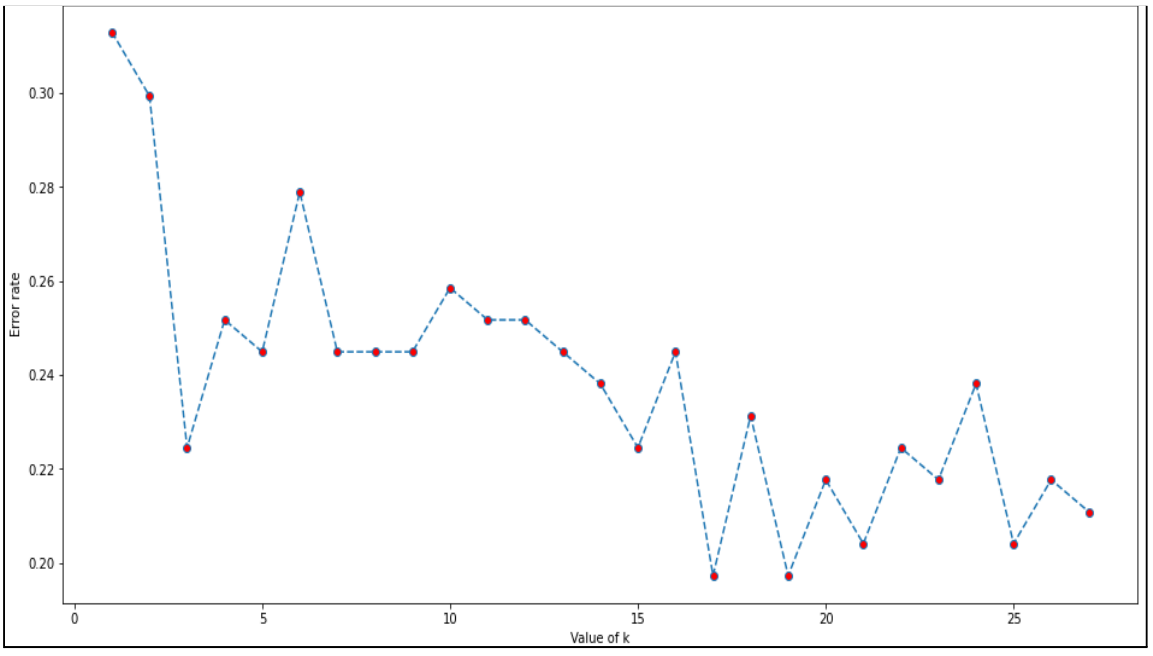
The value of K has been taken 5 and keeping all the parameters to the default mode, the accuracy of the model comes out to be ‘0.7551020408163265’ as can be seen in the code below:

```
from sklearn.neighbors import KNeighborsClassifier
model1= KNeighborsClassifier(n_neighbors=5)
model1.fit(X_train,y_train)
y_pred=model1.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)


# OUTPUT:

0.7551020408163265
```

Since we have chosen the value of k randomly, it is necessary to choose that value of k which minimizes the error rate and maximizes the accuracy. The error rate is simply the wrong predictions made by the model.

So, for different values of K, its respective error rate has been calculated in the code below so as to select the best K for the model.

```
error_rate=[]
for i in range(1,28):
  classifier=KNeighborsClassifier(n_neighbors=i)
  classifier.fit(X_train,y_train)
  y_pred=classifier.predict(X_test)
  error_rate.append(np.mean(y_pred!=y_test))

plt.figure(figsize=(16,8))
plt.plot(range(1,28),error_rate,linestyle='dashed',marker='o',markerfac
ecolor='red')
plt.xlabel("Value of k")
plt.ylabel("Error rate")
print("Minimum error:-",min(error_rate),"at
K=",error_rate.index(min(error_rate))+1)

# OUTPUT:

Minimum error:- 0.19727891156462585 at K = 17
```

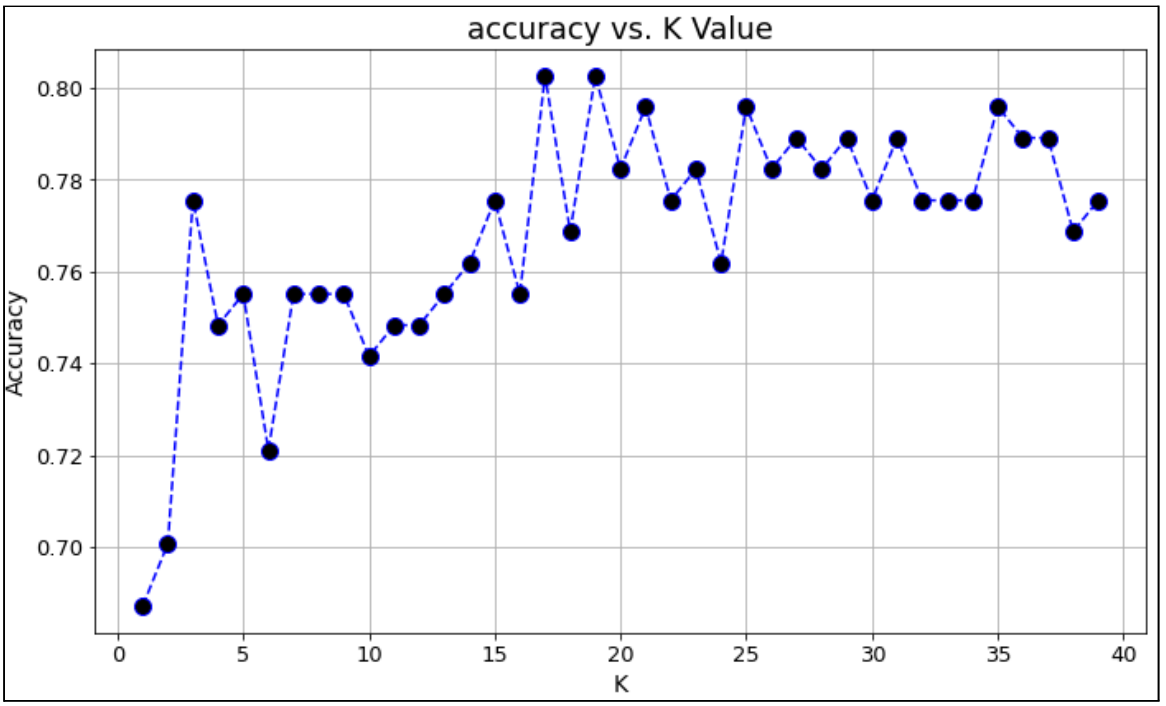



As can be seen at K=17 the error rate is minimum.

Similarly, the highest accuracy score can be calculated for different values of K. In

the code below highest accuracy has been calculated for different values of K, followed by a line graph.

```
plt.style.context('fivethirtyeight')
acc = []
for i in range(1,40):
    neigh = KNeighborsClassifier(n_neighbors = i).fit(X_train,y_train)
    yhat = neigh.predict(X_test)
    acc.append(accuracy_score(y_test, yhat))

plt.figure(figsize=(10,6),tight_layout=True)
plt.plot(range(1,40),acc,color = 'blue',linestyle='dashed',
         marker='o',markerfacecolor='black', markersize=10)
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
plt.grid()

print("Maximum accuracy:-",max(acc),"at K =",acc.index(max(acc))+1
)
# OUTPUT:

Maximum accuracy:- 0.8027210884353742 at K = 17
```



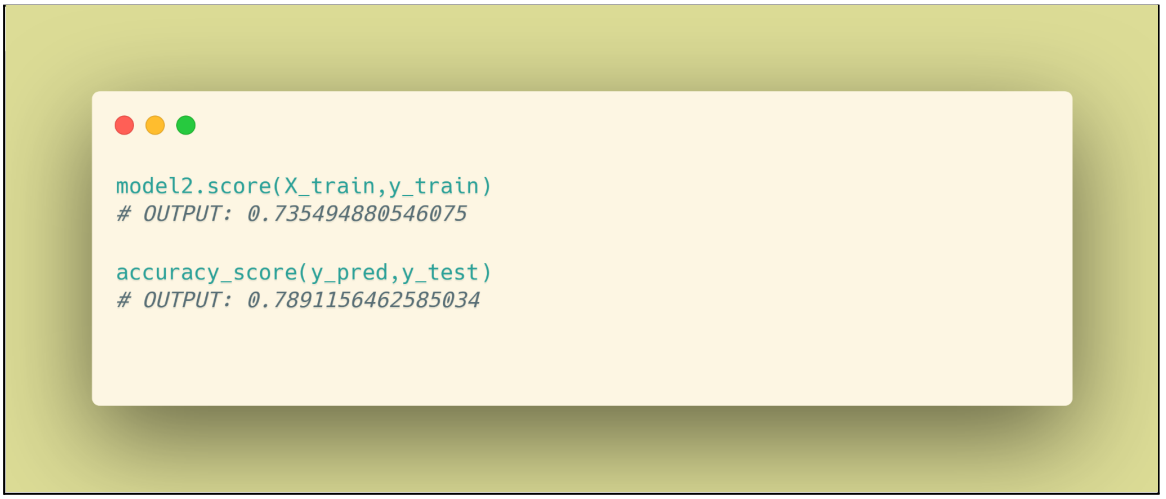

As can be seen, at K=17 the accuracy score is high.

## 5.2 Model2: Naïve Bayes

Naïve Bayes classification algorithm is a simple yet powerful classification technique based on the Bayes theorem. There are three ways of Naive Bayes classification, Gaussian out of all is the most popular and has been used in this project. Below is the code:

```
from sklearn.naive_bayes import GaussianNB
model2=GaussianNB()
model2.fit(X_train,y_train)
y_pred=model2.predict(X_test)
```

Checking for Overfitting:



```
model2.score(X_train,y_train)
# OUTPUT: 0.735494880546075

accuracy_score(y_pred,y_test)
# OUTPUT: 0.7891156462585034
```

The model performed well on the training and testing set, indicating the well-fitting model

## 5.3 Model3: Decision Tree Classifier

Decision tree classifiers are easy to understand and widely used classification algorithm which uses mainly two criterion that are gini and entropy. Both the criterion are important and have been used in this project for the best outcome.

Below is the code for criterion gini:

```
from sklearn.tree import DecisionTreeClassifier
model3=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state
=0)
model3.fit(X_train,y_train) # with gini index as criterion
y_pred=model3.predict(X_test)
```

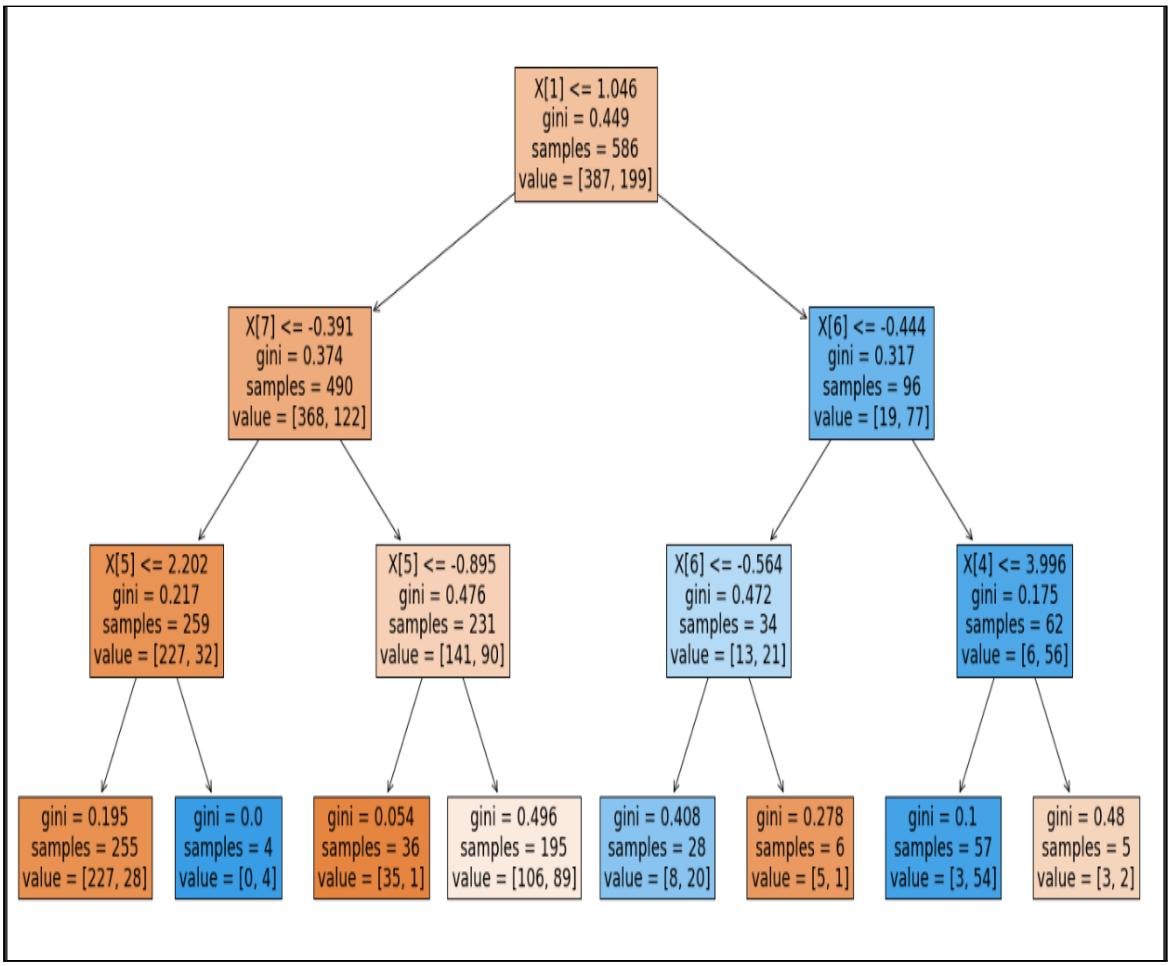
## Checking for Overfitting:

```
model3.score(X_train,y_train)
  #OUTPUT: 0.7747440273037542


accuracy_score(y_test,y_pred)
  #OUTPUT: 0.7074829931972789
```

The model performed quite better on the training set but there is not much difference between the scores of training and testing sets.

The visual representation of the tree with the code:

```
from sklearn import tree
plt.figure(figsize=(25,12))
tree.plot_tree(model3.fit(X_train,y_train),filled=True)
plt.show()
```

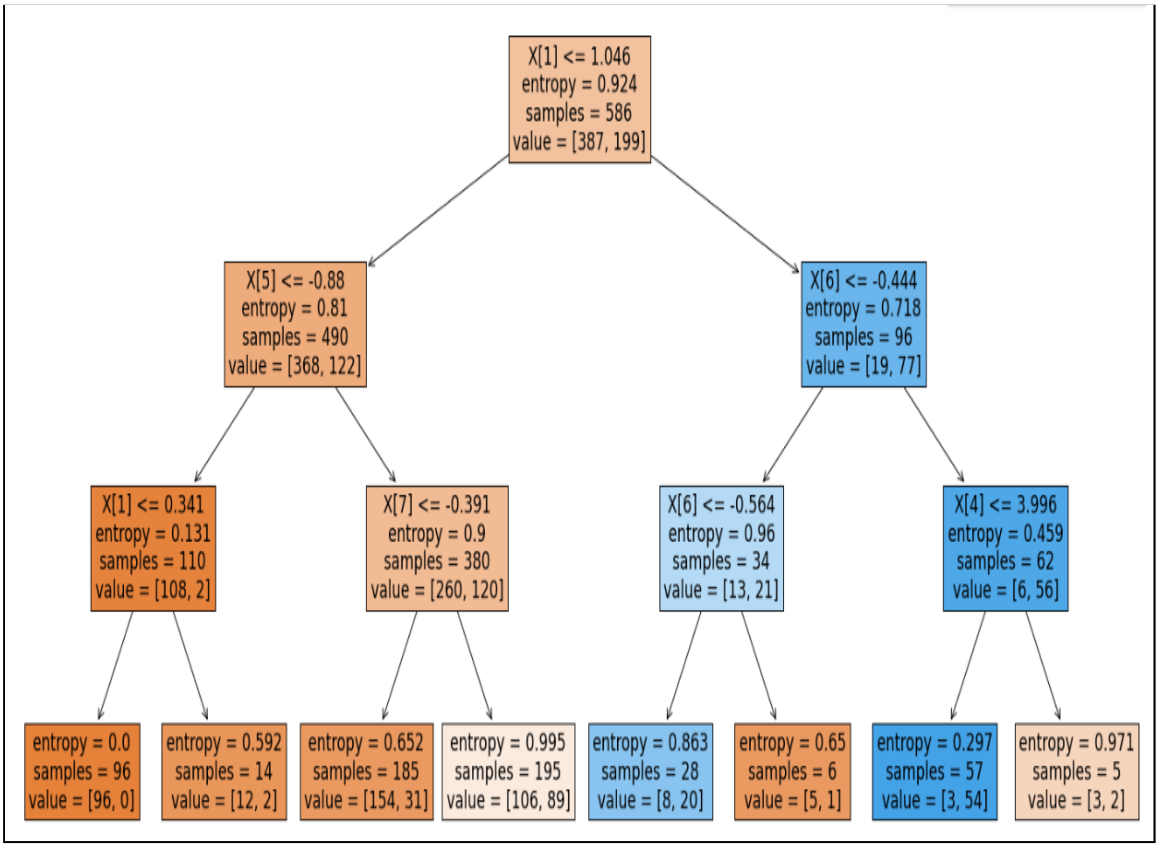



Entropy as the criterion:

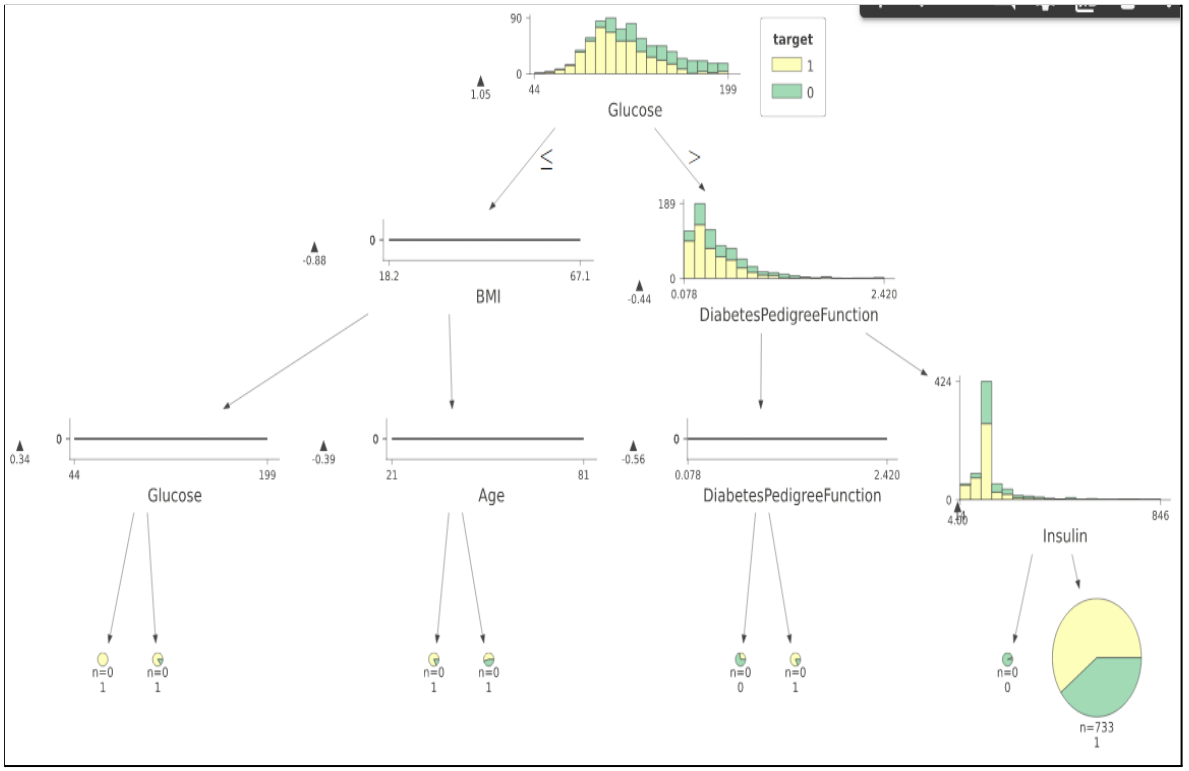
```
model3=DecisionTreeClassifier(criterion='entropy',max_depth=3,random_st
ate=0)
model3.fit(X_train,y_train)
y_pred=model3.predict(X_test)
```

The visual representation of the tree with the code:

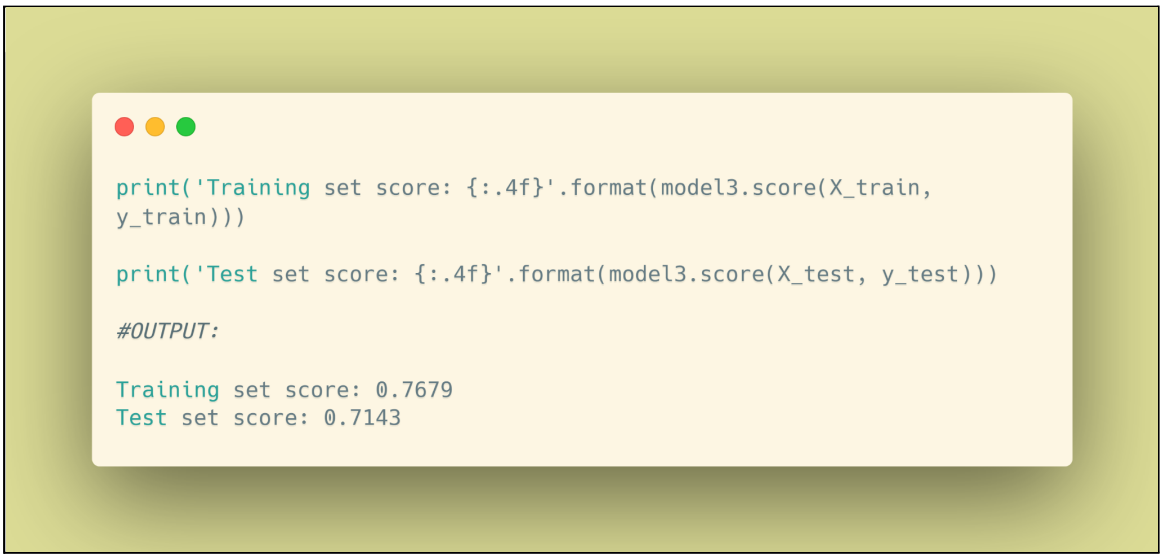
```
plt.figure(figsize=(25,12))
tree.plot_tree(model3.fit(X_train,y_train),filled=True)
plt.show()
```



```
from dtreeviz.trees import dtreeviz

plt.figure(figsize=(25,12))
viz = dtreeviz(model3, X, y,
               target_name="target",
               feature_names=data2.columns,
               class_names=list(data['Outcome']))
```




Checking for Overfitting:




The model performed well on both the training and testing set.

# Chapter 6

## MODEL EVALUATION AND RESULTS

After building the model, evaluating the performance of the model is the crucial step in machine learning. It gives insights into the model as to how well the model is performing. For classification specific problems, many models can be built but choosing the right model for the problem becomes hectic if the model is not evaluated properly.

### 6.1 METRICS

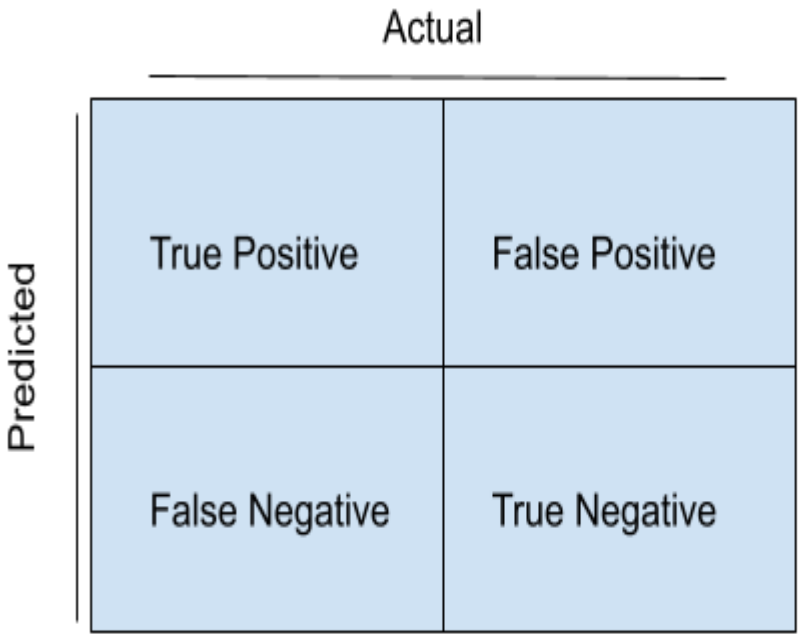
Different performance metrics are used for evaluating the model. Choosing the right metrics for the model evaluation is necessary as it gives insights into the performance of the model. Later the models are compared and selected based on their performance.

#### 6.1.1 CONFUSION MATRIX

A confusion matrix is one of the easiest and simple metrics that is used to find the correctness and accuracy f the model. A confusion matrix is used mostly for classification problems where the target variable can have two or more than two classes.

while the confusion matrix is a widely used metric to evaluate the performance of the models, it is necessary to understand its basics to understand its working.

For a binary classification problem, the confusion matrix has two rows and two columns. The columns are the actual values and rows the predicted(by the model). Following is the diagram for the confusion matrix:




6.1.2 ACCURACY

Accuracy is yet another method for model evaluation that tells how accurate the model has performed. Accuracy is calculated as follows:

$$\frac{TP+TN}{TP+TN+FP+FN}$$

But depending on the accuracy metric alone is not sufficient. Understand this with the following example:
In a dataset of 100 people, a machine is to predict if the patient is cancerous or not. The data contains only 5 patients with cancer and the model predicts all the patients as non-cancerous ( maybe because the data was highly imbalanced). Though the model performed horrendously but its accuracy comes out to be 95% as there were only 5 wrong predictions [21].

6.1.3 PRECISION

The other metric for evaluation is precision. Precision is calculated as below:

$$\frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

6.1.4 RECALL

Recall is calculated as below:

$$\frac{TP}{TP+FN}$$

In the dataset of cancerous patients, if the machine predicts all patients as cancerous. The recall metric will be 100% as it successfully told the cancerous patients as positive.

6.1.5 F-1 SCORE

F1 Score represents both Precision as well as Recall, but its way of computing the mean for precision and recall is slightly different because the normal mean may not give the desired results. The mean F1 Score uses is called harmonic mean which is calculated as below:

where x and y represent Precision and Recall respectively.

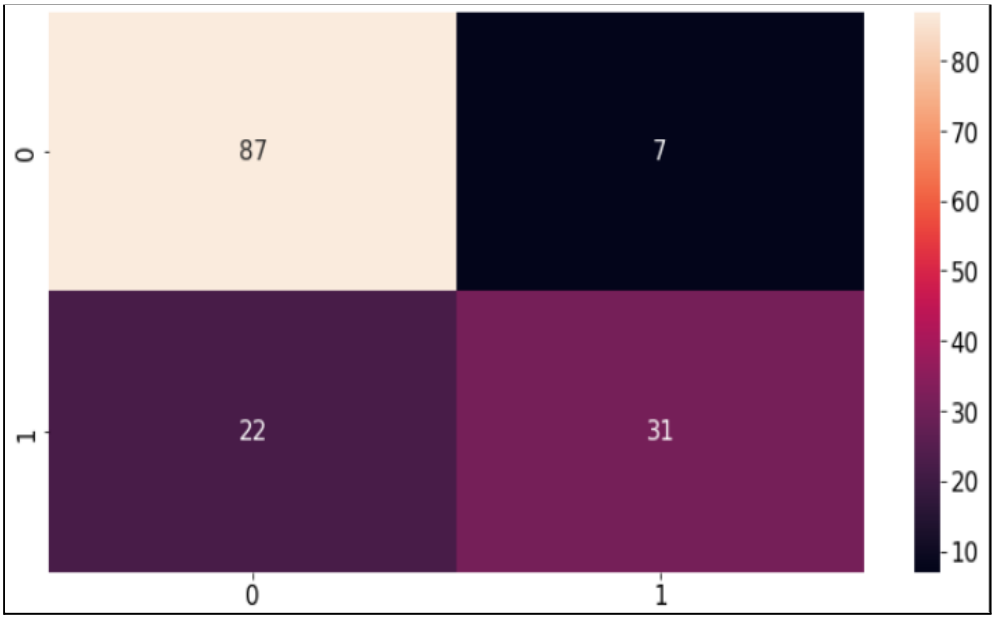
6.2. EXPERIMENTAL RESULTS AND COMPARISON

6.2.1 FOR MODEL 1(K- NEAREST NEIGHBORS):

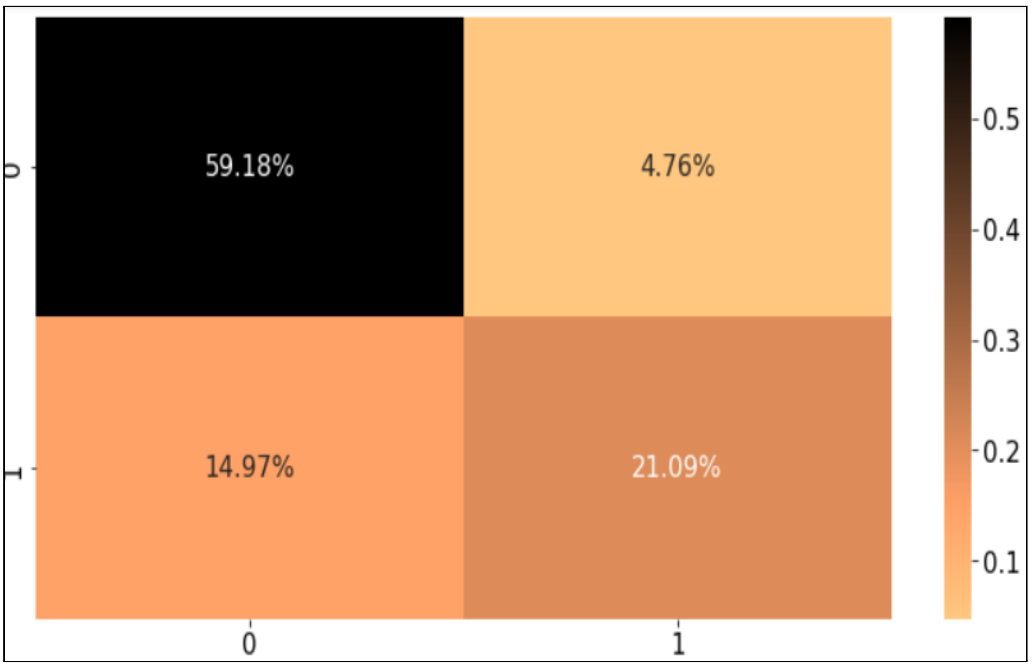
The accuracy for the KNN model with k=17 is 80%, but depending on the accuracy score alone is not sufficient for selecting models. Recall score and precision score reveal a lot about the performance of the model.

The confusion matrix of model 1 along with the code is below:

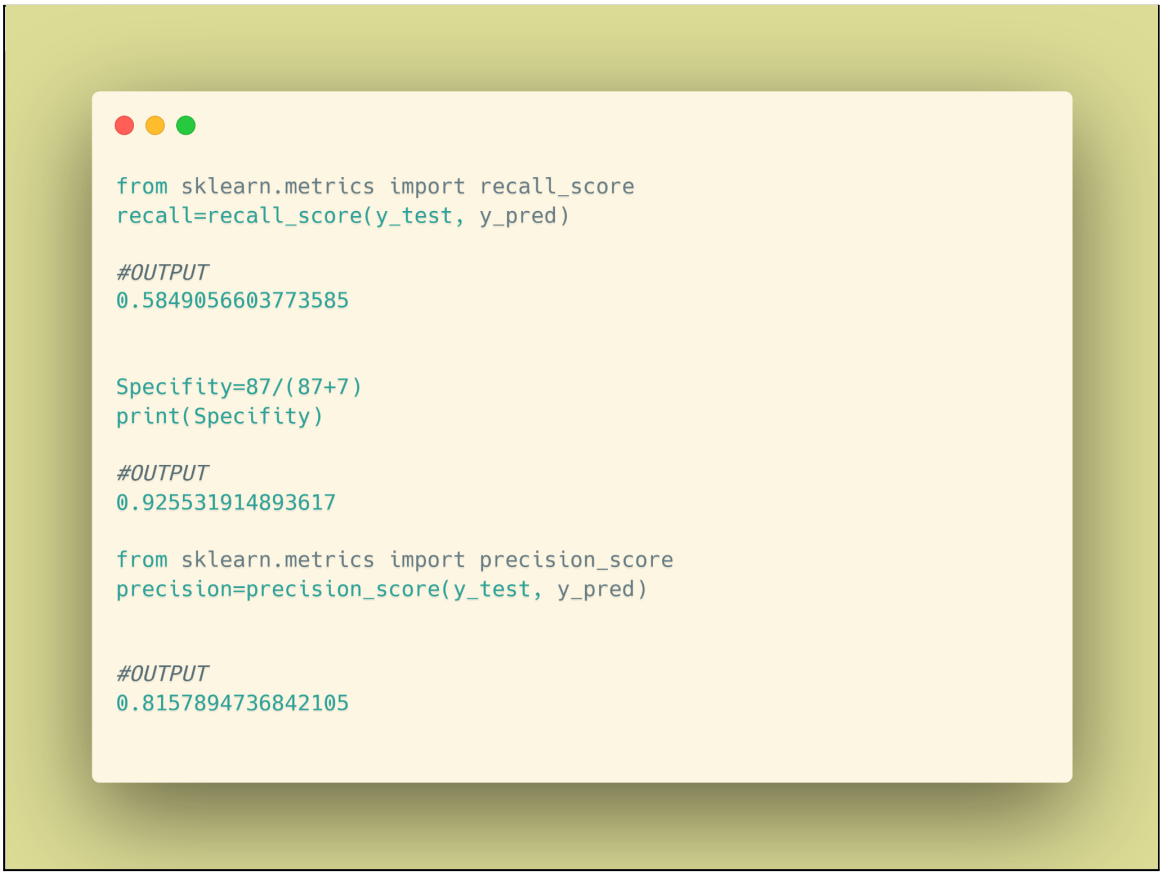
```
from sklearn.metrics import confusion_matrix
cf_matrix=confusion_matrix(y_test,y_pred) #y_test are the real values and
                                          y_pred the one predicted by
                                            model
plt.rcParams['figure.figsize']=(12,6)
plt.rcParams['font.size']=15
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)
plt.show()

sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='copper_r')
plt.show()
```




The following visualization shows what percentage of data lies in each quadrant of the confusion matrix




the bottom right is the true positives and the upper left are the true negatives.
The confusion matrix comes in handy when calculating metrics like Precision, Recall and F1-Score.
The following code shows the Precision, Recall and F1 score for the model1(KNN):

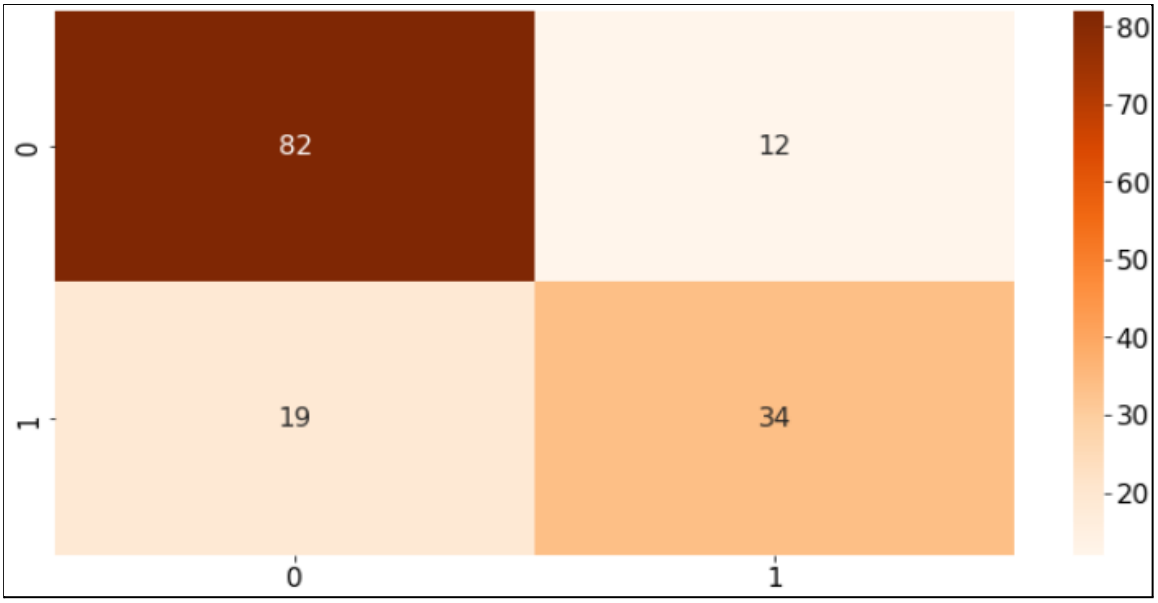


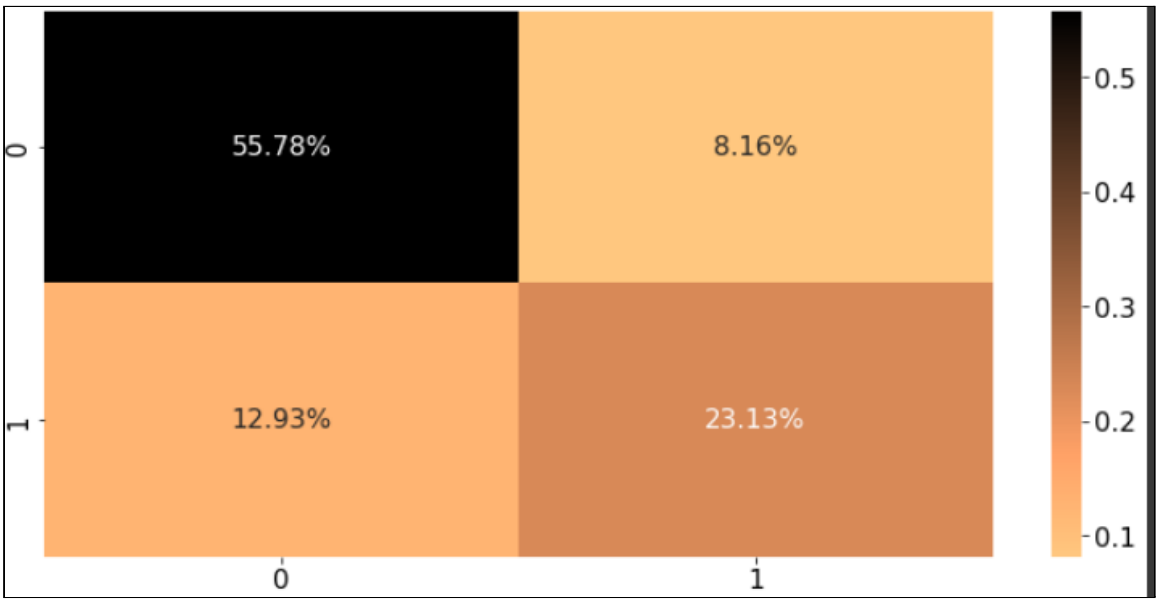
```
from sklearn.metrics import recall_score
recall=recall_score(y_test, y_pred)

#OUTPUT
0.5849056603773585


Specifity=87/(87+7)
print(Specifity)

#OUTPUT
0.925531914893617

from sklearn.metrics import precision_score
precision=precision_score(y_test, y_pred)


#OUTPUT
0.8157894736842105
```

As can be seen, the recall score for model 1 is 58% and the precision score for the same is 91%.
The classification threshold can also be adjusted from 0.5 to lower to increase the sensitivity of the model but the specificity will decrease(as the True negatives will decrease). Specificity and sensitivity are opposite to each other.

6.2.2 FOR MODEL 2( Naïve Bayes ):
The accuracy score for model 2 is 78%, which is quite low as compared to model 1(KNN) but selecting models based on accuracy score is not sufficient.
The matrix creates a better picture for understanding the performance of the models and other metrics are also calculated through the confusion matrix that plays an important role in model evaluation.
The confusion matrix along with the code is below:

```
cf_matrix=confusion_matrix(y_test,y_pred)
sns.heatmap(cf_matrix, annot=True,cmap='Oranges')
plt.show()

sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='copper_r')
plt.show()
```






As can be seen, in the confusion matrix of model 2 the percentage of True positives increased by 2%, on the other hand, the False positives almost got doubled and false negatives were reduced by 2%.
The following code shows the Precision, Recall and F1 score for the model2:

```
recall=recall_score(y_test, y_pred)
precision=precision_score(y_test, y_pred)

print(recall)
#OUTPUT
0.6415094339622641


print(precision)
#OUTPUT
0.7391304347826086
```

```
f1=f1_score(y_test, y_pred)

print(f1)
#OUTPUT
0.6868686868686867
```

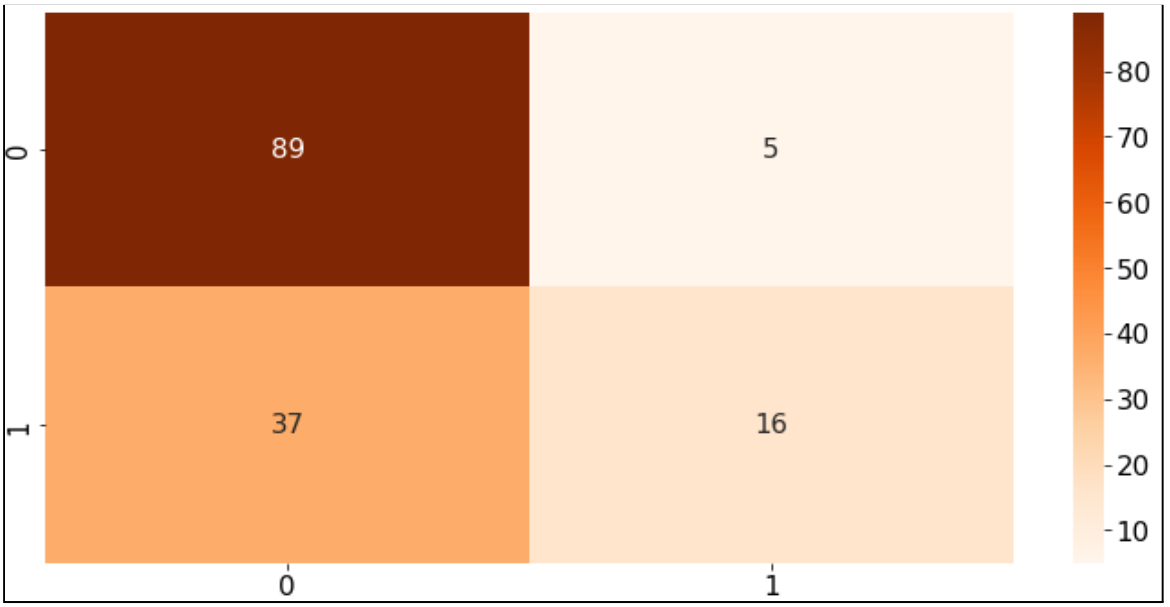
The recall score increased to 64% while the precision dropped to 73%

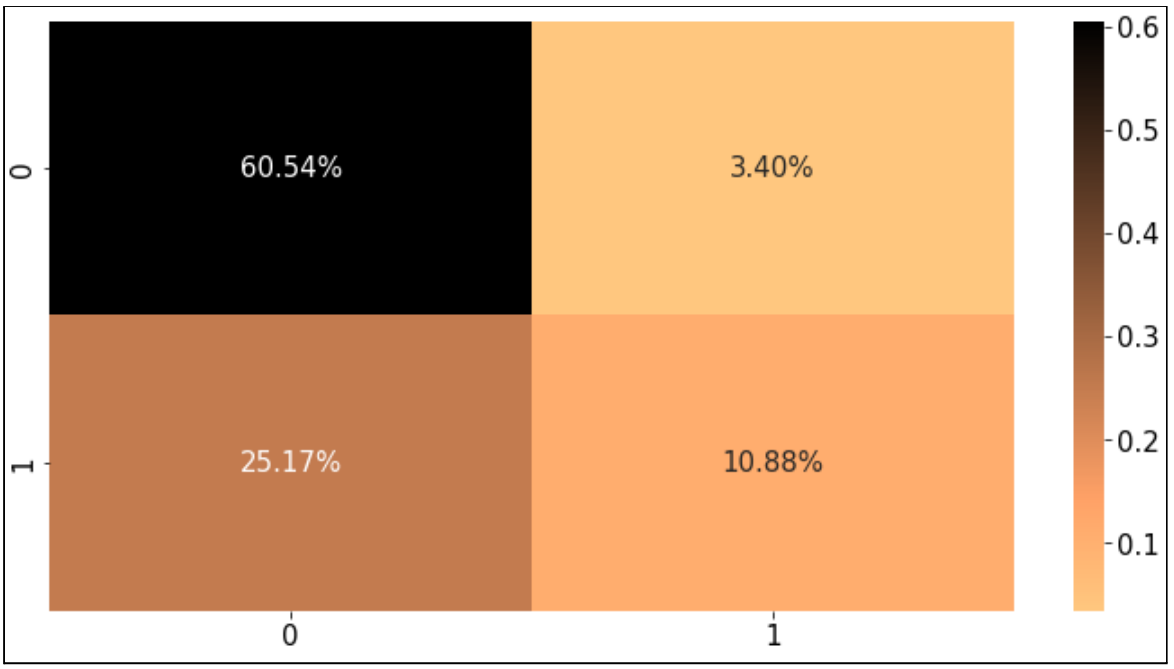
6.2.3 FOR MODEL 3( Decision Tree Classifier):

The accuracy score for model 3 is 71%(when criterion=entropy) which is lower than both KNN and Naive Bayes.

The confusion matrix for the Decision tree classifier along with the code is below:

```
cf_matrix=confusion_matrix(y_test,y_pred)
sns.heatmap(cf_matrix, annot=True,cmap='Oranges')
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,
            fmt='.2%', cmap='copper_r')
```




Clearly, the true positive percentage has dropped down and false negatives have increased which suggests the model is labelling more patients as non-diabetic when in reality they are diabetic.
The following code shows the Precision, Recall and F1 score for model 3:

```
precision=precision_score(y_test, y_pred)
recall=recall_score(y_test, y_pred)
f1=f1_score(y_test, y_pred)

print(precision)
  #OUTPUT
    0.7619047619047619

print(recall)
  #OUTPUT
    0.3018867924528302


print(f1)
  #OUTPUT
    0.43243243243243246
```

The recall score has dropped drastically which suggests the poor performance of Model 3. The F1-Score has also dropped down.

# Chapter 7

# INFERENCES AND CONCLUSION

This project aimed at classifying patients as diabetic or non-diabetic using the PIMA dataset which included various features like BMI, Insulin, Blood Pressure level etc. The classification technique was used since it was a binary classification problem. KNN, Naive Bayes and Decision Tree Classifiers were used to build the models.
All the classifiers performed well. The accuracy of KNN, Naive Bayes and Decision tree were 80%,78% and 71% respectively but choosing models on accuracy score alone is not sufficient and hence other metrics were taken into account to get the insights of the model.
The KNN classifier seemed to perform well as its precision score was close to 80% when the value of K was chosen 17 where it gave the least error rate and recall score was 58% and the model performed well on both training and testing sets hence there was no overfitting.
The Naive Bayes Classifier performed well on both the training and testing sets, hence no issues of overfitting were encountered. Its accuracy was 78% but the recall score of 64%, was better than KNN and the percentage of the True positive increased by 2%.
The Decision tree classifier raised no issue of overfitting or underfitting, it performed well on both the training and testing set with an accuracy score of 71% when entropy was taken as a criterion. Although the recall score dropped drastically to 30% and the percentage of true positives were also decreased, indicating the poor performance of the model which couldn't have been pointed out if accuracy score was the only measure to evaluate models.
Decreasing False Positives or False Negatives solely depends on the problem, since the project aimed at classifying the patients as diabetic and non-diabetic, a model with the least false negatives is preferred. The least false negatives percentage was provided by the Naive Bayes classifier with the recall score of 64% and F1-Score of 68%. Hence, the Naive Bayes Classifier is chosen for classifying patients as diabetic and non-diabetic despite the fact its accuracy score was less than that of the KNN classifier.

# Reference


[1]https://www.cdc.gov/chronicdisease/about/index.htm#:~:text=Chronic%20diseases%20are%20defined%20broadly,disability%20in%20the%20United%20States

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

[3] https://www.sciencedirect.com/science/article/abs/pii/S0140673609609375

[4] https://www.javatpoint.com/data-mining-techniques

[5] https://www.geeksforgeeks.org/basic-concept-classification-data-mining/

[6] https://www.geeksforgeeks.org/clustering-in-data-mining/

[7] https://www.geeksforgeeks.org/association-rule/

: [8] https://www.upgrad.com/blog/classification-in-data-mining/

[9https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

[10] https://www.geeksforgeeks.org/k-nearest-neighbours/

[11] https://www.geeksforgeeks.org/naive-bayes-classifiers/

[12] https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[13] https://scikit-learn.org/stable/modules/tree.html

[14]Data Mining Introduction — Data Preprocessing | by Saishruthi Swaminathan | Medium.

[15]https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/

[16]https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/

[17]https://www.atoti.io/when-to-perform-a-feature-scaling/#:~:text=What%20is%20Feature%20Scaling%3F,during%20the%20data%20preprocessing%20step.

[18] Why it is important to handle missing data and 10 methods to do it. | by Niwratti Kasture | Analytics Vidhya | Medium

[20]https://docs.microsoft.com/en-us/analysis-services/data-mining/training-and-testing-data-sets?view=asallproducts-allversions#:~:text=Separating%20data%20into%20training%20and,of%20evaluating%20data%20mining%20models.&text=Because%20the%20data%20in%20the,the%20model's%20guesses%20are%20correct.

[21]https://medium.com/@MohammedS/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b

**Books:**

1. Tan, P. N., Steinbach, M., & Kumar, V. (2016). *Introduction to data mining*. Pearson Education India.
2. Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts*

*and techniques*. Elsevier.
3. Kesavaraj, G., & Sukumaran, S. (2013, July). A study on classification techniques in data mining. In *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)* (pp. 1-7). IEEE.