EXPOSYS DATA LABS - DATA SCIENCE INTERN REPORT

DIABETIC DISEASE PREDICTION USING ML

AN INTERNSHIP REPORT

Submitted in partial fulfillment for the
Completion of the Internship
Domain: ***Data Science***

by

Yaswitha Sai Atluri

College: Vellore Institute of Technology, Chennai

Mail: yaswithasai.atluri@gmail.com

Under the Guidance of

EXPOSYS DATA LABS

Bengaluru, Karnataka 560064

6th July 2021 - 6th Aug 2021

TABLE OF CONTENTS

ABSTRACT

Diabetes has become one of the major causes of national disease and death in most countries. According to the International Diabetes Federation report, the figure is expected to rise to more than 642 million in 2040, so early screening and diagnosis of diabetes patients have great significance in detecting and treating diabetes on time. Diabetes is a multi factorial metabolic disease, its diagnostic criteria is difficult to cover all the ethology, damage degree, pathogenesis and other factors, so there is a situation for uncertainty and imprecision under various aspects of medical diagnosis process. With the development of Data mining, researchers find that machine learning is playing an increasingly important role in diabetes research. Machine Learning (ML) techniques are now being used in various fields like education, healthcare, business, recommendation system, etc. Healthcare data is complex and high in dimensionality and contains irrelevant information - due to this, the prediction accuracy is low. The Pima Indians Diabetes Dataset was used in this research, it consisted of 768 records. Machine learning techniques can find the risky factors of diabetes and reasonable threshold of physiological parameters to unearth hidden knowledge from a huge amount of diabetes-related data, which has a very important significance for diagnosis and treatment of diabetes. So this project provides a survey of machine learning techniques that has been applied to diabetes data screening and diagnosis of the disease. In this project, conventional machine learning techniques are described in early screening and diagnosis of diabetes. More over deep learning techniques which have a significance of biomedical effect are also described.

Keywords: Diabetes; Feature Extraction; Machine Learning; KNN; SVM; Diabetic Prediction.

CHAPTER I
INTRODUCTION

Diabetes is one of the most chronic diseases in the world in which the sugar level of blood becomes too high. It has become a fifth-ranked disease for disease related deaths. Due to diabetes, other problems may arise like the increased risk of heart attack and stroke. Unfortunately, these diseases cannot be cured; the only way is to manage the glucose level in the blood. Around 8.8% of adults were diabetic in 2017 around the world and the projected value is 9.9% by 2045. According to the International Diabetes Federation (IDF) statistics, there were 415 million people suffering from diabetes around the world. By 2040 this number is expected to rise to over 642 million, as a consequence, diabetes has become the main cause of national disease and death in most countries. Diabetes is a group of metabolic diseases in which a person has high blood glucose, either because the body does not produce enough insulin, or because cells do not respond to the insulin that is produced. If diabetes patients cannot control blood sugar well, it is effortless to induce cardiovascular, nervous system, eye, foot and other systemic diseases. Patients whose conditions are severe can also suffer from diabetic ketoacidosis, with a high disability. Diabetes has a very great deal of harm to the human body, causing a series of complications, affecting the patient physical and mental health, bringing a heavy burden to family and society.

1.1. DIABETES CLASSIFICATION:

The Diabetic disease is classified/segmented into three categories (i) type I diabetes (ii) type II diabetes and (iii) type III (gestational diabetes). Most of the people having diabetes are of type II. The Pima is one of the most studied populations for diabetic analysis around the world. Most of the Pima population is type-2 diabetic.

- Type 1 diabetes is an autoimmune disease that occurs in childhood. In this type of diabetes, the pancreatic cells that secrete insulin have been destroyed.
- Type 2 diabetes is caused by insulin resistance in various organs, leading to a marked increase in insulin demand, which accounts for almost 90% of the diabetes cases .
- Gestational diabetes tends to occur among pregnant women, as the pancreas does not make sufficient amount of insulin.

The standards of early screening and diagnosis of diabetes are still in the exploratory stage on account of the unclear ethology and pathogenesis of diabetes. Through the continuous understanding of diabetes, the criteria of screening and diagnosis are constantly changing. Early diagnosis of diabetes mainly depends on clinical symptoms and signs. In 1965, the World Health Organization (WHO) first published diabetes diagnostic norm based on the clinical characteristics, but this criteria did not mention the diagnosis threshold of blood sugar levels. With the developing understanding of diabetes, diagnostic criteria gradually increased fasting blood glucose (FPG), oral glucose tolerance test (OGTT), glycosylated haemoglobin (HbA1c) and other physiological parameters. In 1980, the fasting blood glucose level was viewed as the main diagnostic norm. In 1997, the new standard of American Diabetes Association (ADA) increased the OGTT parameters.

1.2. NEED FOR THE PREDICTION MODEL:

Accurate screening and diagnosis of diabetes require more effective features and have a high demand on the judgment which can be closer to the nature of the disease. Some studies found that if we consider metabolic changes in diabetes from the perspective of body metabolism, doctors can better make a diagnosis of the type of diabetes and help patients with the more appropriate diabetic treatment. Metabolomics is a new discipline that has been developed in recent years to analyze all the low molecular weight metabolites of a certain organism or cell qualitatively and quantitatively. Through the change of endogenous metabolites and intermediates in diabetes and the evolution of coping rules, the metabolic status of the body can be further understood. On the basis of the study of early screening and diagnostic criteria for diabetes, diagnostic standards are increased from the initial clinical symptoms and signs to FPG, OGTT, HbA1c and other physiological parameters. Simultaneously clinical and demographic signs are also included in the diagnostic reference, such as sex, age, race/ethnicity, haemoglobin disease/anemia, body mass index (BMI), cardiovascular disease, family history/ Genetic, medication records, etc. However, there is still no way to find out the pathogenesis of diabetes from the field of biology. It is urgent to clarify the pathology and diagnostic criteria of diabetes, it has a great significance in delaying the occurrence and development of diabetes, choosing drugs, reducing the incidence of diabetic complications and extending life expectancy. With the continuous development of artificial intelligence and data mining technology, researchers begin to consider using machine learning techniques to search for the characteristics of diabetes. Machine learning techniques can find implied pathogenic factors in virtue of analyzing and using diabetic data, with a high stability and accuracy in diabetic diagnosis.

Therefore, machine learning techniques which can find out the reasonable threshold of risky factors and physiological parameters provide new ideas for screening and diagnosis of diabetes. Diabetic classification is a challenging issue due to nonlinear and complex data. Some other reasons are that the Pima dataset has null entries and outliers, which causes low performance of machine learning algorithms. Machine learning algorithms are presently being used in almost all fields like finances, marketing, medical, business, etc. Machine learning algorithms are of three types (i) supervised learning (ii) unsupervised and (iii) semi-supervised. In supervised learning, labeled data is available and the machine learns from some part of it and applies the learning to the unseen data. In the unsupervised machine learning algorithm, data is not labeled, the algorithms find an interesting relationship between the data points. Semi-supervised is a combination of supervised and unsupervised algorithms. With respect to diabetic prediction, we have applied various supervised machine learning algorithms as the dataset of Pima is labeled. The classifiers cannot correctly classify, due to the presence of missing values and outliers present. In mathematics, outlier and missing value handling is an important issue which cannot be ignored. The dataset considered goes through two phases before applying the feature selection techniques. In the first phase, the missing values of the dataset are replaced by the median. Further feature selection techniques have been applied.

1.3. ABOUT DATA SET: (PIMA INDIANS DIABETES DATABASE DATA SET)

In India, diabetes is a major issue. Between 1971 and 2000, the incidence of diabetes rose ten times, from 1.2% to 12.1%. 61.3 million people 20–79 years of age in India are estimated living with diabetes (Expectations of 2011). It is expected that by 2030 this number will rise to 101,2 million. In India there are reportedly 77.2 million people with prediabetes. In 2012, nearly 1 million people in India died of diabetes. 1 out of 4 individuals living in Chennai’s urban slums suffer from diabetes, which is about 7 per cent by three times the national average. One third of the deaths in India involve people under non-communicable diseases Sixty years old. Indians get diabetes 10 years before their Western counterparts on average. Changes in lifestyle lead to physical decreases increased fat, sugar and activities activity calories and higher insulin cortisol levels Obesity and vulnerability. In 2011, India cost around $38 billion annually as a result of diabetes.

Context:

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Content:

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

CHAPTER 2

EXISTING MODEL

2.1. Literature Review

Mitushi Soni , Dr. Sunita Varma [1] designed and implemented 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, Random Forest, Decision Tree, Logistic Regression and Gradient Boosting classifiers are used. And 77% classification accuracy has been achieved. The Experimental results can be asst health care to take early prediction and make early decision to cure diabetes and save humans life.

K.VijiyaKumar et al. [2] proposed random Forest algo- rithm for the Prediction of diabetes develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by using Random Forest algorithm in ma- chine learning technique. The proposed model gives the best results for diabetic prediction and the result showed that the prediction system is capable of predicting the diabetes disease effectively, efficiently and most importantly, instantly.

Nonso Nnamoko et al. [5] presented predicting diabetes onset: an ensemble supervised learning approach they used five widely used classifiers are employed for the ensembles and a meta-classifier is used to aggregate their outputs. The results are presented and compared with simi- lar studies that used the same dataset within the literature. It is shown that by using the proposed method, diabetes onset prediction can be done with higher accuracy. Tejas

N. Joshi et al. [4] presented Diabetes Prediction Using Machine Learning Techniques aims to predict diabetes via three different supervised machine learning methods including: SVM, Logistic regression, ANN. This project pro- poses an effective technique for earlier detection of the diabetes disease.

Deeraj Shetty et al. [7] proposed diabe- tes disease prediction using data mining assemble Intelli- gent Diabetes Disease Prediction System that gives analysis of diabetes malady utilizing diabetes patients database. In this system, they propose the use of algorithms like

Bayesian and KNN (K-Nearest Neighbor) to apply on diabetes patients database and analyze them by taking various attributes of diabetes for prediction of diabetes disease.

Muhammad Azeem Sarwar et al. [8] proposed study on prediction of diabetes using machine learning algorithms in healthcare they applied six different machine learning algo- rithms Performance and accuracy of the applied algorithms is discussed and compared. Comparison of the different machine learning techniques used in this study reveals which algorithm is best suited for prediction of diabetes. Diabetes Prediction is becoming the area of interest for researchers in order to train the program to identify the patient are diabetic or not by applying proper classifier on the dataset. Based on previous research work, it has been observed that the classification process is not much improved. Hence a system is required as Diabetes Prediction is important area in computers, to handle the issues identified based on previous research.

CHAPTER 3

PROPOSED METHOD WITH ARCHITECTURE

Goal of the project is to investigate for model to predict diabetes with better accuracy, different classification algorithms to predict diabetes.

3.1. ARCHITECTURE OF THE PROPOSED SYSTEM:

Here we collect the data (in this case we are using PIMA Indians Diabetes Data set is taken) , we do some preprocessing techniques like data cleaning , missing values insertion , Feature selection , after that cleaning the data and analyzing it . After the pre processing steps we spilt the data set into Train and Test data sets (80% and 20% respectively). After spiltting the data set , we apply machine learning algorithms to it and predict the accuracy of that algorithms and detect the most accurate one. This is most important phase which includes model building for prediction of diabetes. In this we have implemented various machine learning algorithms such as KNN, SV , Naive Bias for diabetes prediction.

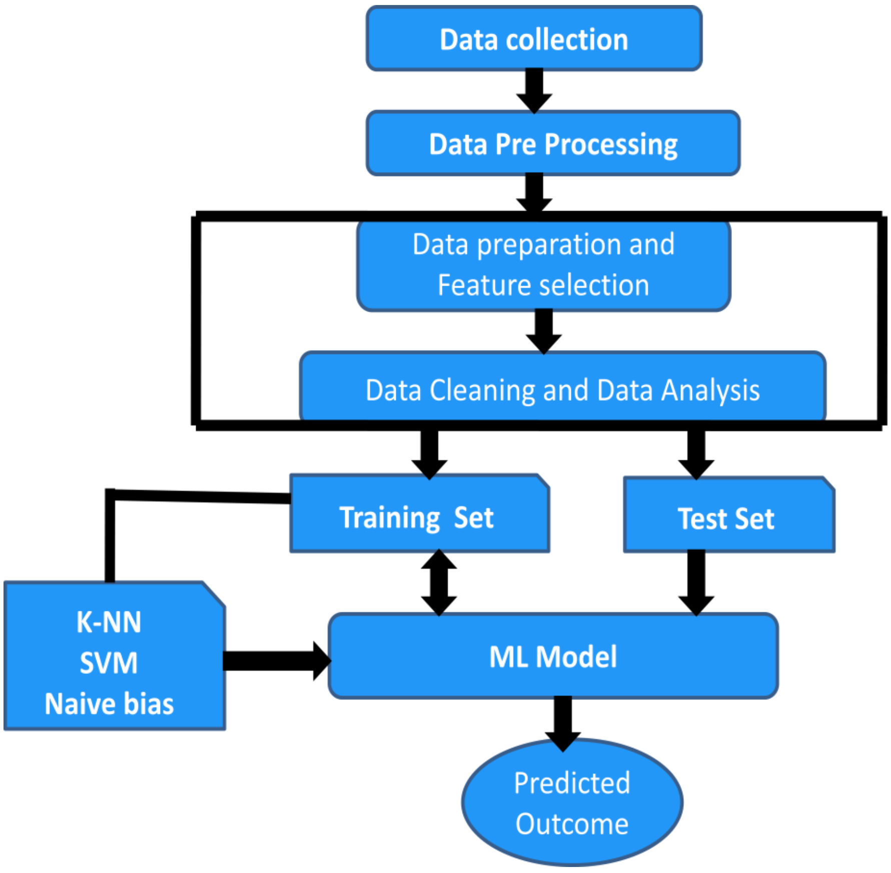


Fig: Architecture of the proposed system / Methodology (Built Model)

3.2. DATA COLLECTION:

The original donor of the data set is the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset was from the website of the University of California, Irvine (UCI) [26]. The data set consists of 768 records (all women) out of which 500 are not diabetic and 268 are diabetic. There are eight attributes in the dataset as shown in Table 1. This dataset does contain zero values corresponding to the: (i) Glucose attribute in five records (ii) 35 records in the blood pressure (iii) 27 records in the BMI (Body Mass Index) attribute (iv) 227 records in the Skin Thickness attribute and (v) 374 records in the Insulin attribute. These zero values do not have significance so were replaced by the median of the corresponding attribute.

Dataset Description- The data is gathered from UCI repository which is named as Pima Indian Diabetes Dataset. The dataset have many attributes of 768 patients.

Table 1: Dataset Description

S No.	Attributes
1	Pregnancy
2	Glucose
3	Blood Pressure
4	Skin thickness
5	Insulin
6	BMI(Body Mass Index)
7	Diabetes Pedigree Function
8	Age

- The 9th attribute is class variable of each data points. This class variable shows the outcome 0 and 1 for diabetics which indicates positive or negative for diabetics.
- Distribution of Diabetic patient- We made a model to predict diabetes however the dataset was slightly imbalanced having around 500 classes labeled as 0 means negative means no diabetes and 268 labeled as 1 means positive means diabetic.

3.3. DATA PREPROCESSING:

Data pre-processing is most important process. Mostly healthcare related data contains missing vale and other impurities that can cause effectiveness of data. To improve quality and effectiveness obtained after mining process, Data preprocessing is done. To use Machine Learning Techniques on the dataset effectively this process is essential for accurate result and successful prediction. For Pima Indian diabetes dataset we need to perform pre processing in two steps.

3.4. MISSING VALUES REMOVAL:

Remove all the instances that have zero (0) as worth. Having zero as worth is not possible. Therefore this instance is eliminated. Through eliminating irrelevant features/instances we make feature subset and this process is called features subset selection, which reduces dimensionality of data and help to work faster.

3.5. SPLITTING OF DATA:

After cleaning the data, data is normalized in training and testing the model. When data is spitted then we train algorithm on the training data set and keep test data set aside. This training process will produce the training model based on logic and algorithms and values of the feature in training data. Basically aim of normalization is to bring all the attributes under same scale.

3.6. APPLY MACHINE LEARNING TECHNIQUES (MODEL BUILDING):

When data has been ready we apply Machine Learning Technique. We use different classification and ensemble techniques, to predict diabetes. The methods applied on Pima Indians diabetes dataset. Main objective to apply Machine Learning Techniques to analyze the performance of these methods and find accuracy of them, and also been able to figure out the responsible/important feature which play a major role in prediction.

The Techniques are follows

3.6.1. SUPPORT VECTOR MACHINE

Support Vector Machine also known as SVM which is a supervised machine learning algorithm. SVM is most popular classification technique. SVM creates a hyper plane that separate two classes. It can create a hyper plane or set of hyper plane in high dimensional space. This hyper plane can be used for classification or regression also. SVM differentiates instances in specific classes and can also classify the entities which are not sup- ported by data. Separation is done through hyper plane performs the separation to the closest training point of any class.

Algorithm:

3.7. Select the hyper plane which divides the class better.

3.8. To find the better hyper plane you have to calculate the distance between the planes and the data which is called Margin.

3.9. If the distance between the classes is low then the chance of miss conception is high and vice versa. So we need to

3.10. Select the class which has the high margin.

3.11. Margin = distance to positive point + Distance to negative point.

3.6.2. K-NEAREST NEIGHBOR

KNN is also a supervised machine learning algorithm. KNN helps to solve both the classification and regression problems. KNN is lazy prediction technique. KNN assumes that similar things are near to each other. Many times data points which are similar are very near to each other.KNN helps to group new work based on similarity measure.KNN algorithm record all the records and classify them according to their similarity measure. For finding the distance between the points uses tree like structure. To make a prediction for a new data point, the algorithm finds the closest data points in the training data set its nearest neighbors.

Here K= Number of nearby neighbors, it’s always a positive integer. Neighbor’s value is chosen from set of class. Closeness is mainly defined in terms of Euclidean distance.

The Euclidean distance between two points P and Q i.e. P (p1,p2, . ,pn) and Q (q1, q2,..qn) is defined by the following equation:-

Algorithm:

3.7. Take a sample dataset of columns and rows named as Pima Indian Diabetes data set.

3.8. Take a test dataset of attributes and rows.

3.9. Find the Euclidean distance by the help of formula.

3.10. Then, Decide a random value of K is the no. of nearest neighbors

3.11. Then with the help of these minimum distance and Euclidean distance find out the nth column of each.

3.12. Find out the same output values.

If the values are same, then the patient is diabetic, other- wise not.

3.6.3. NAIVE BAYES

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as ‘Naive’.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below,

$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

Above,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1. Proposed Methodology:

Step 1: Import required libraries, Import diabetes dataset.

Step 2: Pre-process data to remove missing data.

Step 3: Perform percentage split of 80% to divide dataset as Training set and 20% to Test set.

Step 4: Select the machine learning algorithm i.e. K- Nearest Neighbor, Support Vector Machine, Decision Tree, Logistic regression, Random Forest and Gradient boosting algorithm.

Step 5: Build the classifier model for the mentioned machine learning algorithm based on training set.

Step 6: Test the Classifier model for the mentioned machine learning algorithm based on test set.

Step 7: Perform Comparison Evaluation of the experimental performance results obtained for each classifier.

Step 8: After analyzing based on various measures conclude the best performing algorithm.

4.2. Implementation and Results

Data preprocessing involves the transformation of the raw dataset into an understandable format. Preprocessing data is a fundamental stage in data mining to improve data efficiency. The data preprocessing methods directly affect the outcomes of any analytic algorithm.

Data preprocessing is generally carried out in 7 simple steps:

4.3. Steps In Data Preprocessing:

1. Gathering the data
2. Import the dataset & Libraries
3. Dealing with Missing Values
4. Divide the dataset into Dependent & Independent variable
5. dealing with Categorical values
6. Split the dataset into training and test set
7. Feature Scaling

1. Gathering the data

Data is raw information, it’s the representation of both human and machine observation of the world. Dataset entirely depends on what type of problem you want to solve. Each problem in machine learning has its own unique approach.

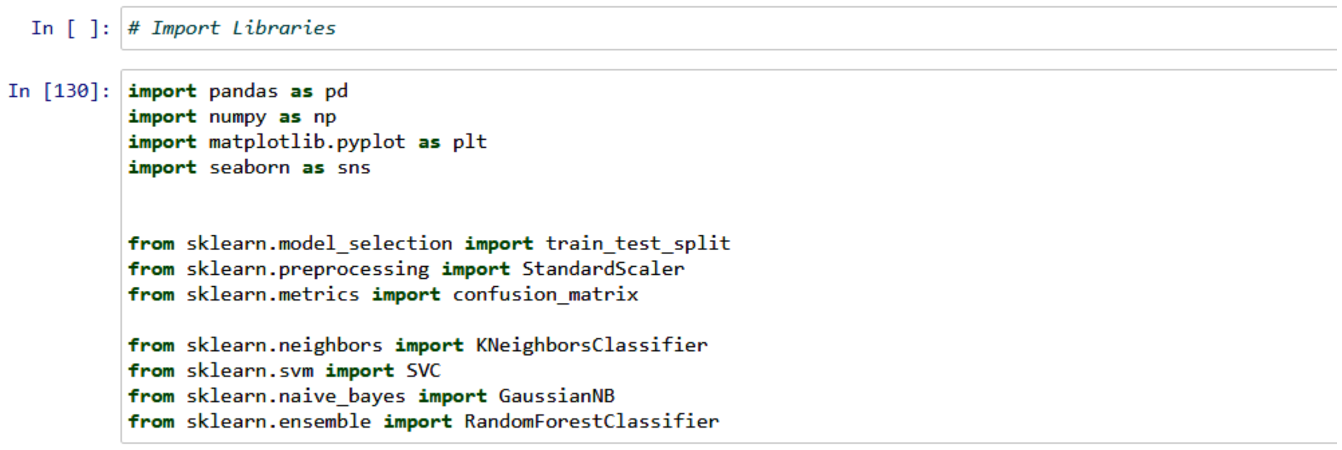
Kaggle: Kaggle is one of the best platforms to get the dataset.

https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv

2. Import the dataset & Libraries

First step is usually importing the libraries that will be needed in the program. A library is essentially a collection of modules that can be called and used.

And can be import the libraries in python code with the help of '*import*' keyword.

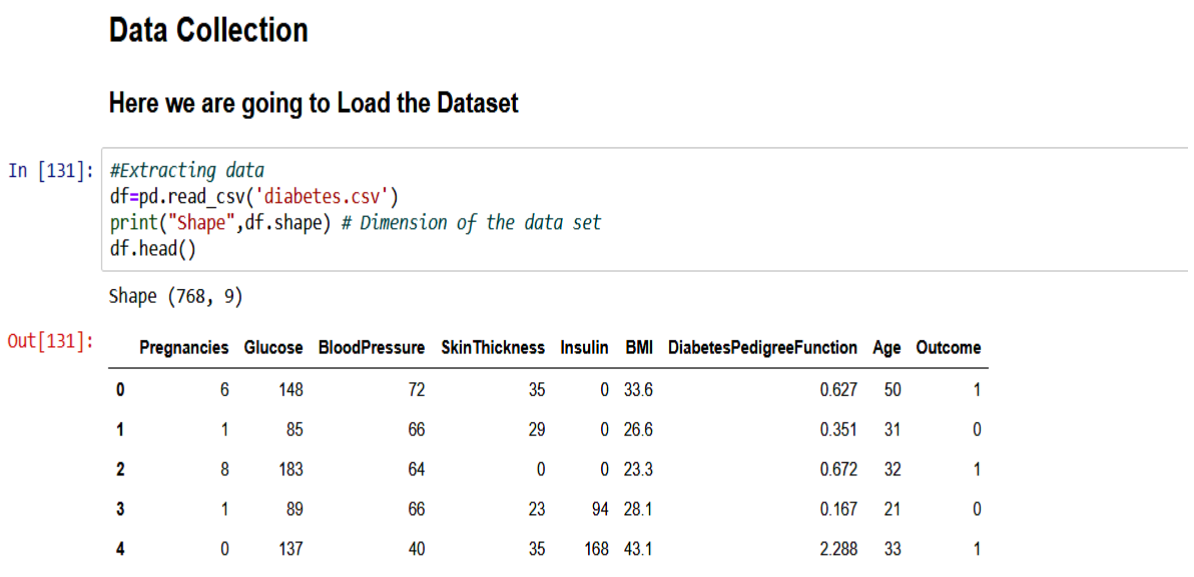


```
In [ ]: # Import Libraries
```

```
In [130]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns


          from sklearn.model_selection import train_test_split
          from sklearn.preprocessing import StandardScaler
          from sklearn.metrics import confusion_matrix

          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.svm import SVC
          from sklearn.naive_bayes import GaussianNB
          from sklearn.ensemble import RandomForestClassifier
```

Importing the dataset

The first and foremost step is collection of data i.e loading the dataset using pandas library ***'read_csv'*** method. Here we have data in csv format, there is many kind of file can be read by using pandas library as shown below:



Data Collection

Here we are going to Load the Dataset

```
In [131]: #Extracting data
          df=pd.read_csv('diabetes.csv')
          print("Shape",df.shape) # Dimension of the data set
          df.head()
```

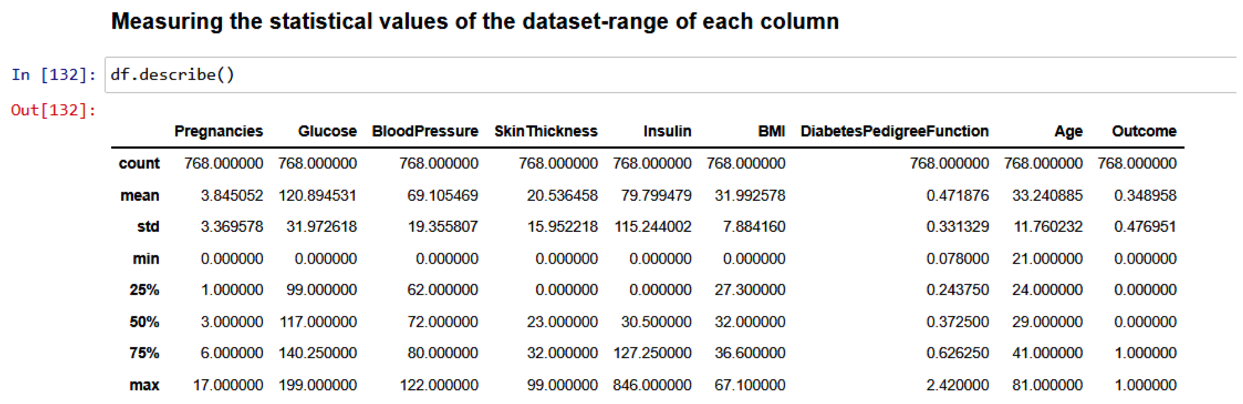
```
Shape (768, 9)
```

Out[131]:

	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

3. Dealing with Missing Values

Sometimes we may find some data are missing in the dataset. if we found then we will remove those rows or we can calculate either **mean, mode or median** of the feature and replace it with missing values. This is an approximation which can add variance to the dataset.



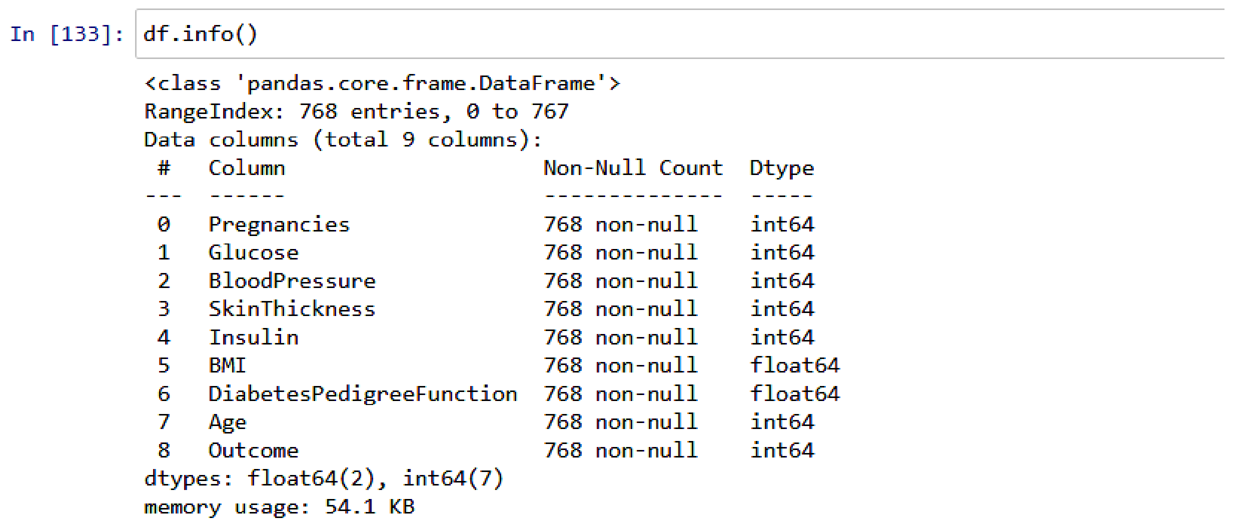
Measuring the statistical values of the dataset-range of each column

```
In [132]: df.describe()
Out[132]:
```

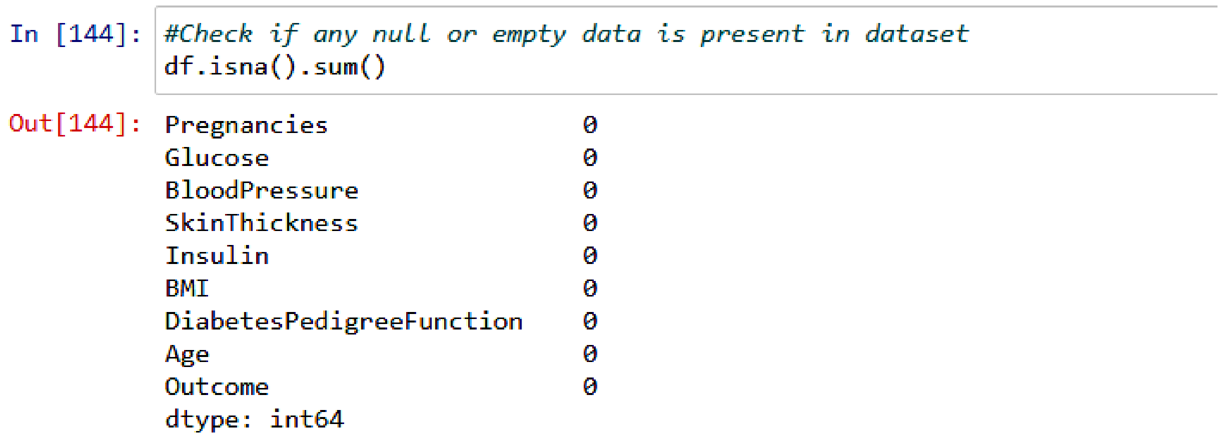
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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

Check for null values:

we can check the null values in our dataset with pandas library as below.With the help of *info()* we can found total number of entries as well as count of non-null values with datatype of all features.we also can use *dataset.isna()* to see the of null values in our dataset.But usually we work on large dataset so it will be a good thing to get the count of all null values corresponding to each features and it will be done by using *sum().*



```
In [133]: 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
```


```
In [144]: #Check if any null or empty data is present in dataset
          df.isna().sum()

Out[144]: Pregnancies                 0
          Glucose                     0
          BloodPressure               0
          SkinThickness               0
          Insulin                     0
          BMI                         0
          DiabetesPedigreeFunction    0
          Age                         0
          Outcome                     0
          dtype: int64
```

Drop Null values:

Pandas provide a **dropna()** function that can be used to drop either row or columns with missing data. We can use *dropna()* to remove all the rows with missing data. But this is **not always** a good idea. Sometime we have small dataset, as we used in our example and removing the whole row means somewhere we are deleting some valuable information from dataset.

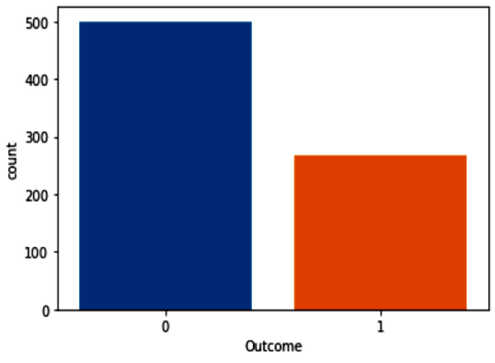
Replacing Null values with Strategy: For replacing null values we use the strategy that can be applied on a feature which has numeric data. We can calculate the *Mean, Median or Mode* of the feature and replace it with the missing values. It will affect the entire data-set and replaces every variable null values with their respective mean, and '*inplace =True*' indicates to affect the changes to dataset. If we need to replace particular variable with the strategies then we can use above line of code.

→ In the output screenshot it specifies only the top five rows from the whole data frame by making use of head() method.

Visualization made using seaborn countplot() method to show the count of the observations based in each categorical bin using bars.

Proportion of non diabetic cases is more i.e. 500

```
In [134]: #Counting values of outcomes having 0 or 1, 0 means non diabetic and 1 means diabetic
          sns.countplot(x='Outcome',data=df)

Out[134]: <AxesSubplot:xlabel='Outcome', ylabel='count'>
```



→ The value of Outcome ‘**0’**represents – non-diabetic

→ The value of Outcome ‘**1’**represents – diabetic

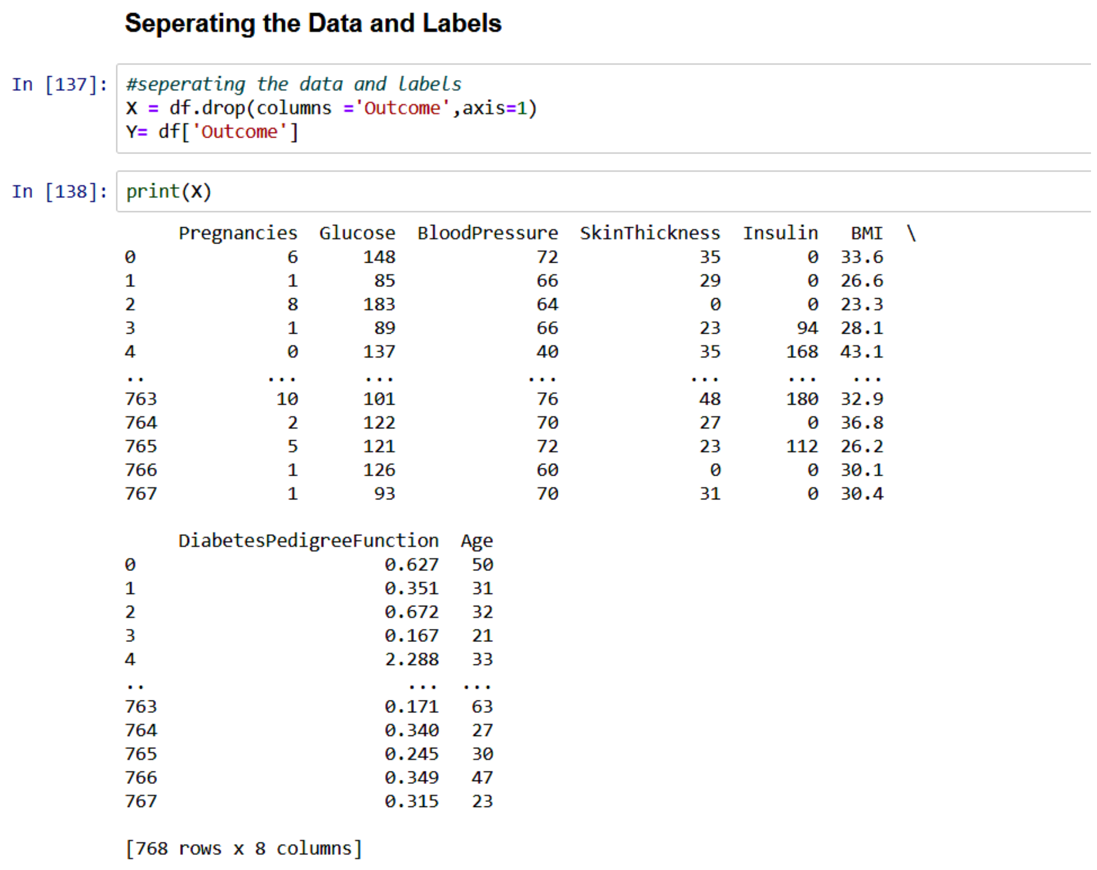
Representing the count of patients based on the Outcome either ‘0’ or ‘1’.

```
In [135]: df['Outcome'].value_counts()

Out[135]: 0    500
          1    268
          Name: Outcome, dtype: int64

In [136]: df.groupby('Outcome').mean()

Out[136]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Outcome								
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

→ Also calculating the average of all of the remaining individual columns by grouping them based on ‘Outcome’ column.

Seperating the output column (‘Outcome’) from the remaining columns of the data frame and store it into separate variable.



Seperating the Data and Labels

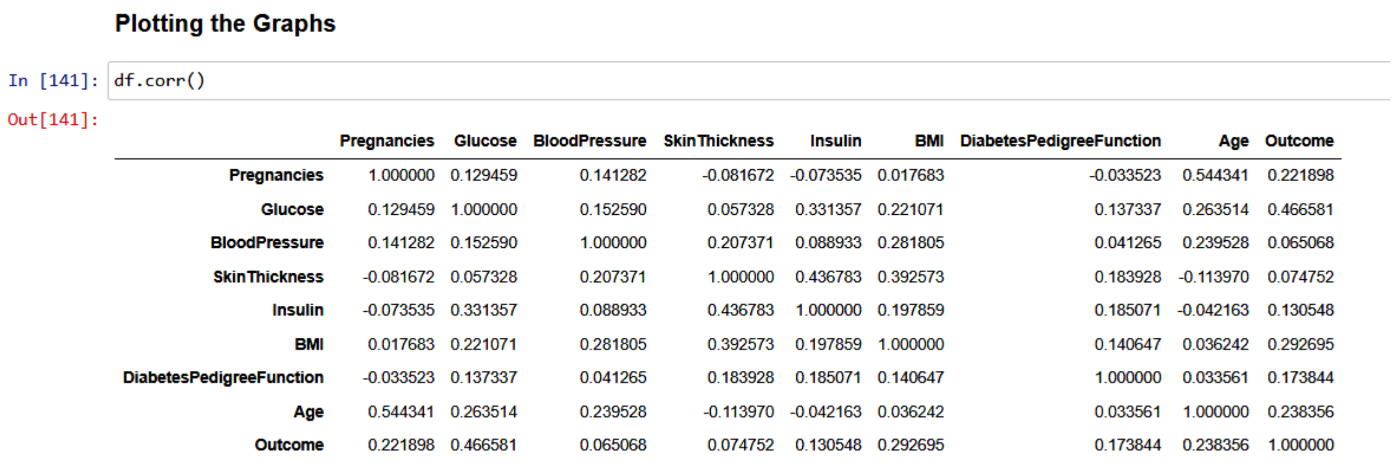
```
In [137]: #seperating the data and labels
          X = df.drop(columns ='Outcome',axis=1)
          Y= df['Outcome']
```

```
In [138]: print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  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
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

Finding Correlation:

Applying Correlation function using corr() method in order to find the pairwise correlation in all the columns in the dataframe and returning correlation coefficients which are in the range [0,1].



Plotting the Graphs

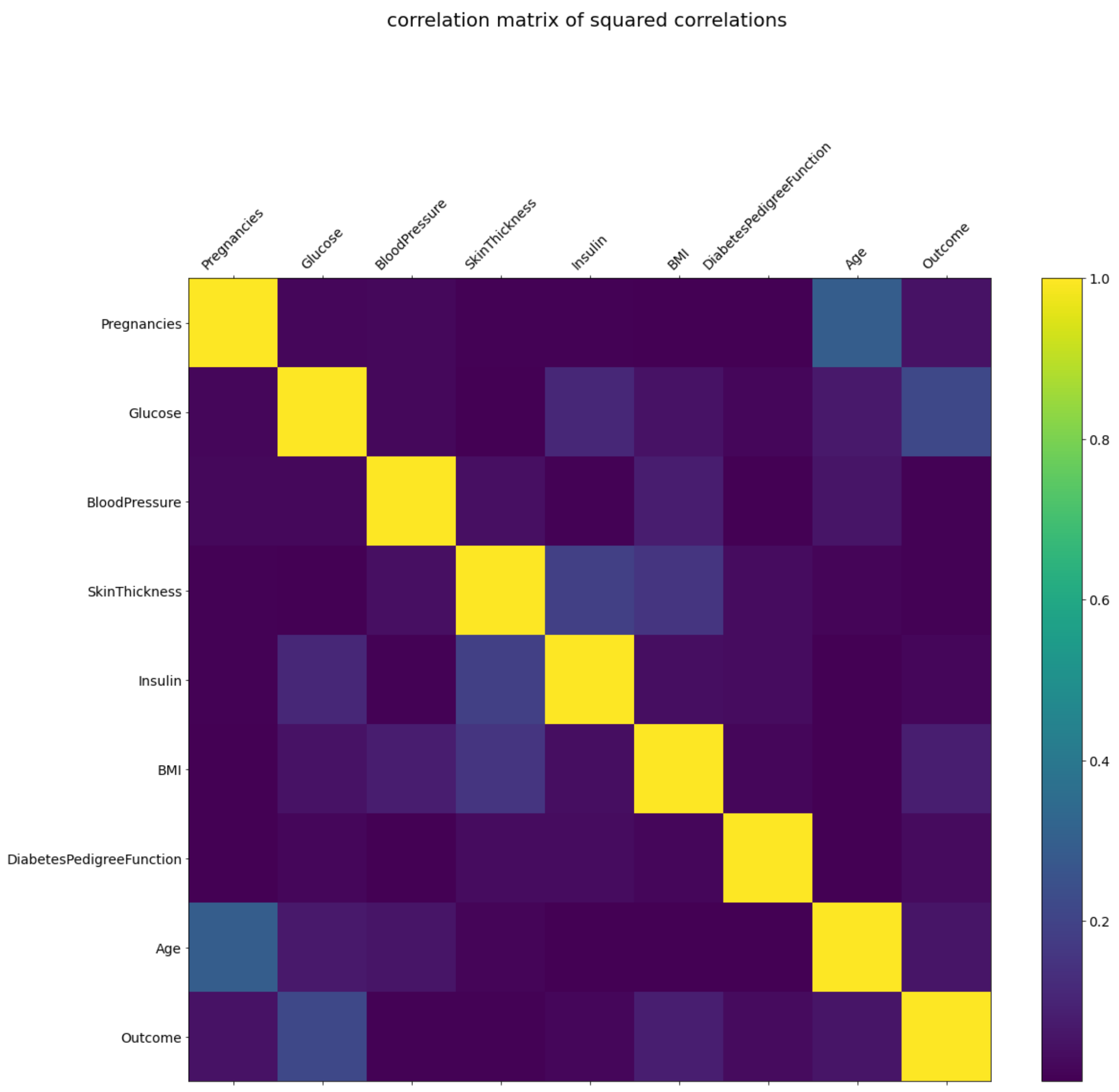
```
In [141]: df.corr()
Out[141]:
```

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

Visualization of correlation coefficients of all the columns by colorbar() method.

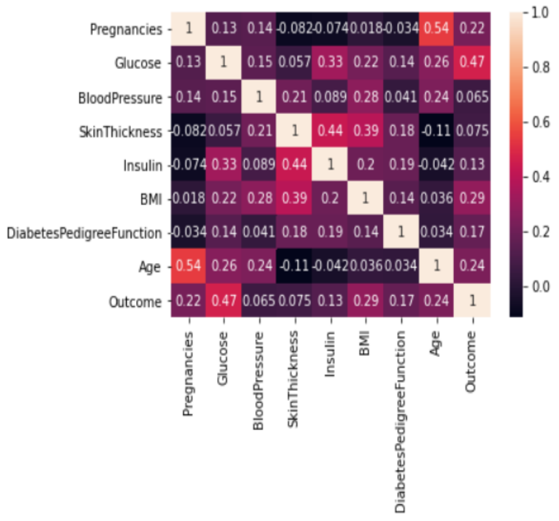
```
In [142]: fig = plt.figure(figsize = (19,15))
          plt.matshow(df.corr()**2,fignum = fig.number)
          plt.xticks(range(df.shape[1]),df.columns,fontsize = 14, rotation = 45)
          plt.yticks(range(df.shape[1]),df.columns,fontsize = 14)
          cb = plt.colorbar()
          cb.ax.tick_params(labelsize = 14)
          plt.title("correlation matrix of squared correlations \n\n\n\n\n",fontsize =20)

Out[142]: Text(0.5, 1.0, 'correlation matrix of squared correlations \n\n\n\n\n')
```

correlation matrix of squared correlations



Representing the correlation matrix using sns heatmap in order to show the correlation between two variables.So, the obtained correlation between Age and Outcome is upto 54% that means the output majorly depends on Age as it is the highest value out of all the correlation coefficients. So, the obtained correlation between Glucose and Outcome is upto 47% that means the output majorly depends on Glucose as it is also the highest value out of all the correlation coefficients.

```
In [143]: #Correlation matrix to show correlation between two variables, 0.x means x% similar
          corr_mat=df.corr()
          sns.heatmap(corr_mat, annot=True)

Out[143]: <AxesSubplot:>
```



	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

Ex: correlation between Glucose and Outcome is 47% that means output depends majorly on Glucose

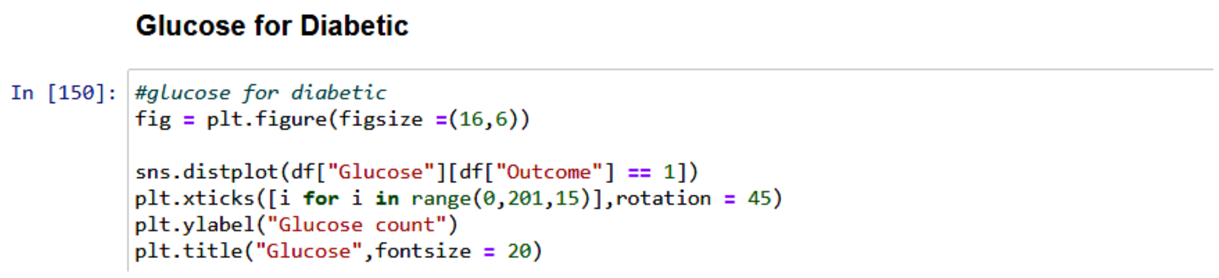
Data Analysis:

Data Analysis

We have to check which columns are useful and which are not

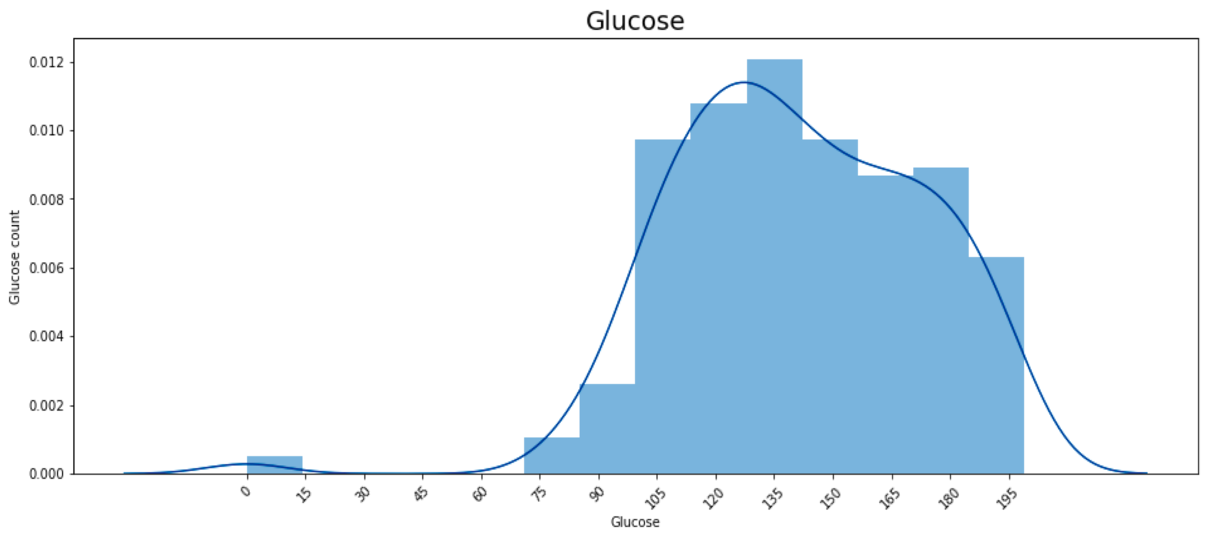
```
In [149]: #1-->diabetic
          #0-->healthy

          print(df["Outcome"].value_counts())
          fig = plt.figure(figsize = (10, 6))

0    500
1    268
Name: Outcome, dtype: int64

<Figure size 720x432 with 0 Axes>
```

This is the representation of a Distribution plot which depicts the variation in the diabetes data distribution where **Glucose** and **Outcome** correlation is ‘1’.

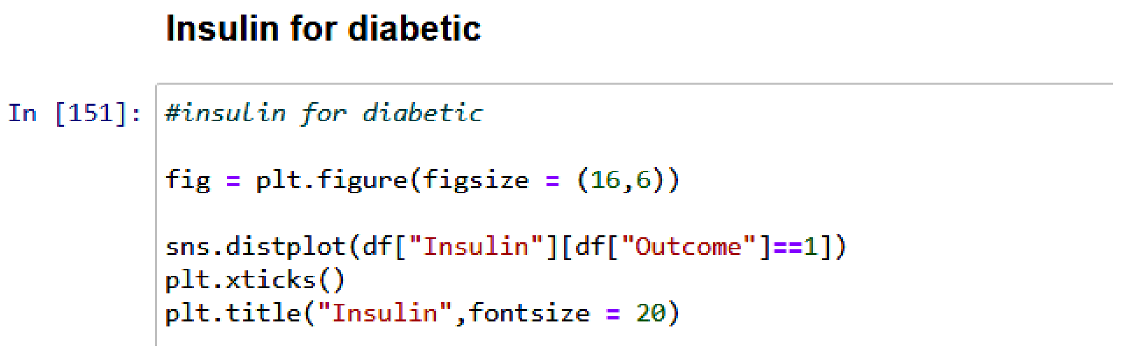


Glucose for Diabetic

```
In [150]: #glucose for diabetic
          fig = plt.figure(figsize =(16,6))

          sns.distplot(df["Glucose"][df["Outcome"] == 1])
          plt.xticks([i for i in range(0,201,15)],rotation = 45)
          plt.ylabel("Glucose count")
          plt.title("Glucose",fontsize = 20)
```

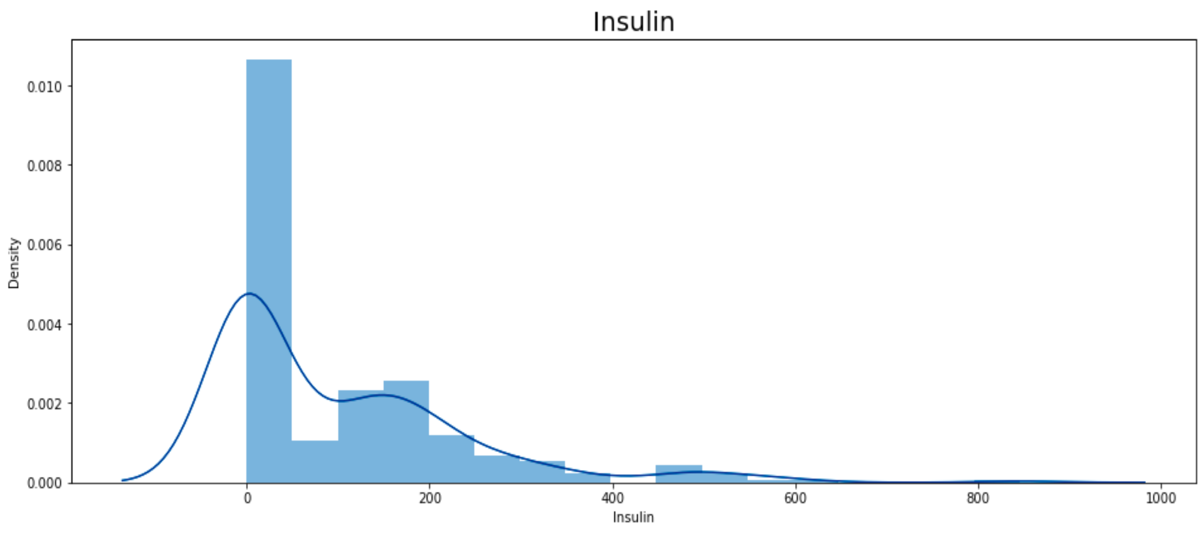


This is the representation of a Distribution plot which depicts the variation in the diabetes data distribution where **Insulin** and **Outcome** correlation is ‘1’.

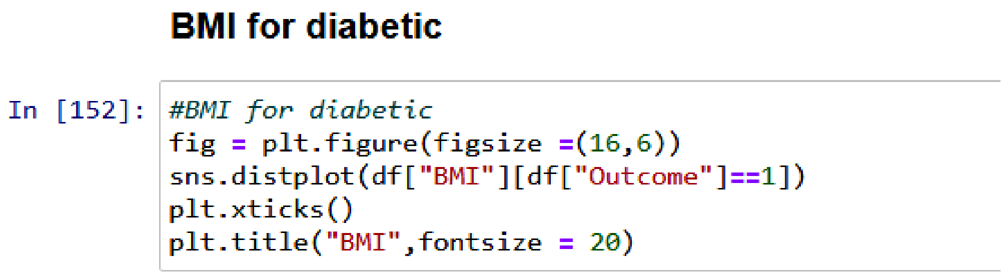


Insulin for diabetic

```
In [151]: #insulin for diabetic

          fig = plt.figure(figsize = (16,6))

          sns.distplot(df["Insulin"][df["Outcome"]==1])
          plt.xticks()
          plt.title("Insulin",fontsize = 20)
```

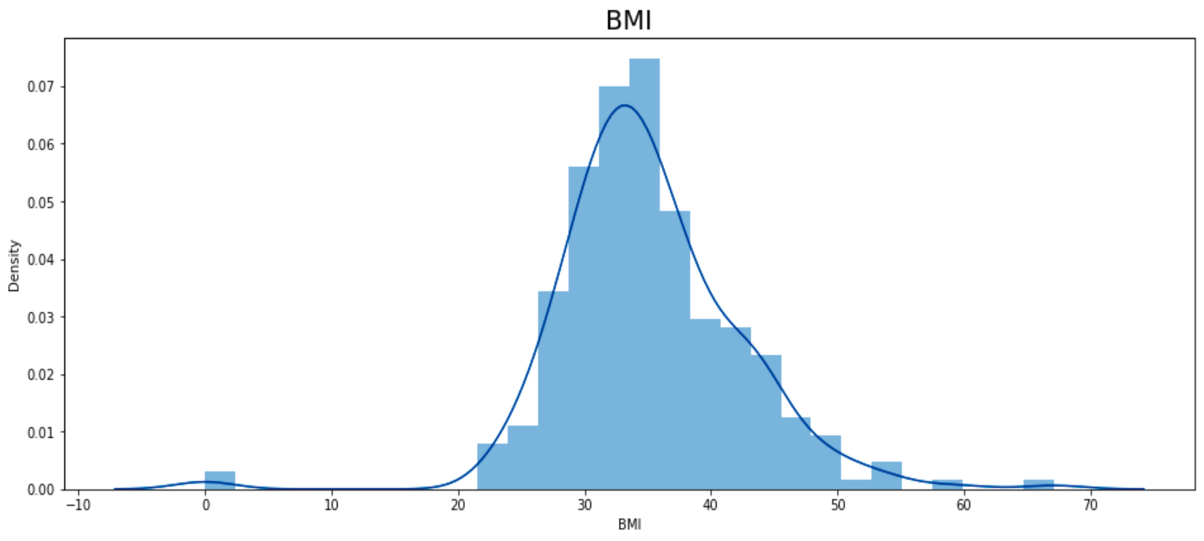


The below screenshot is the representation of a Distribution plot which depicts the variation in the diabetes data distribution where **BMI** and **Outcome** correlation is ‘1’.

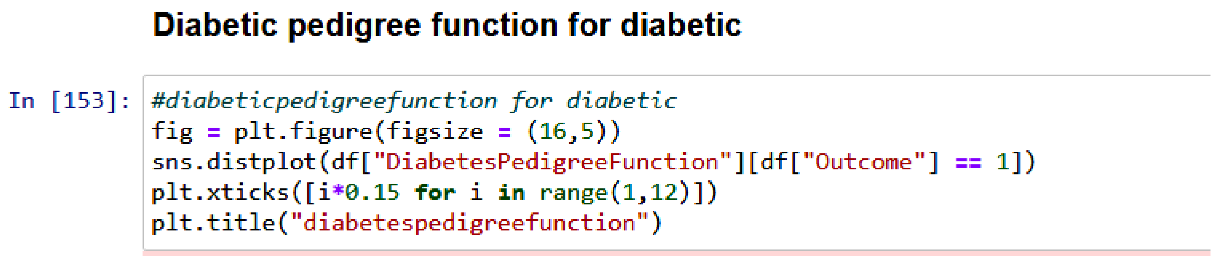


BMI for diabetic

```
In [152]: #BMI for diabetic
          fig = plt.figure(figsize =(16,6))
          sns.distplot(df["BMI"][df["Outcome"]==1])
          plt.xticks()
          plt.title("BMI",fontsize = 20)
```

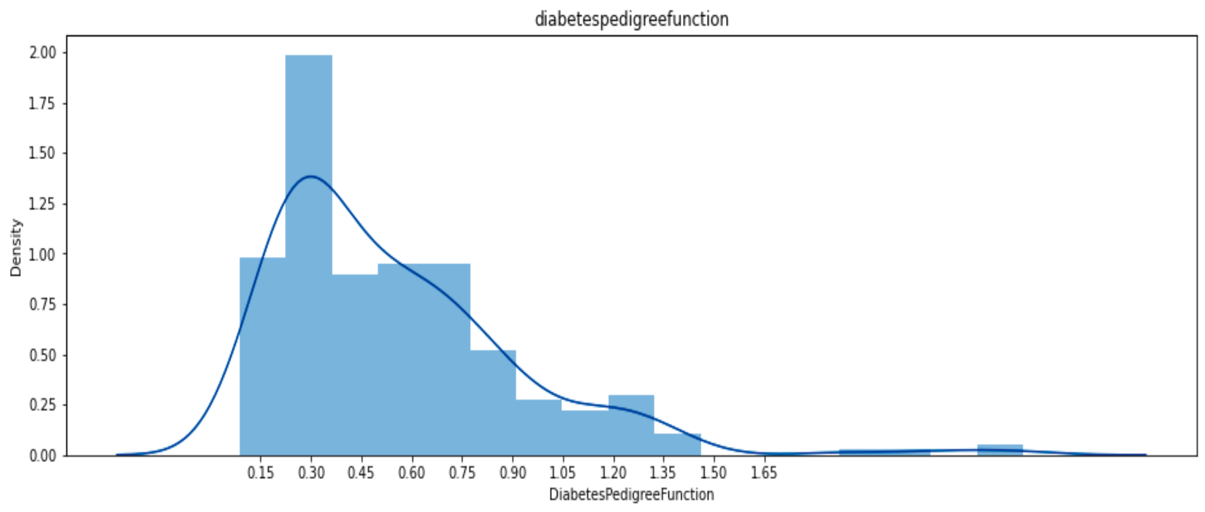


This is the representation of a Distribution plot which depicts the variation in the diabetes data distribution where **diabetespedigreefunction** and Outcome correlation is ‘1’.

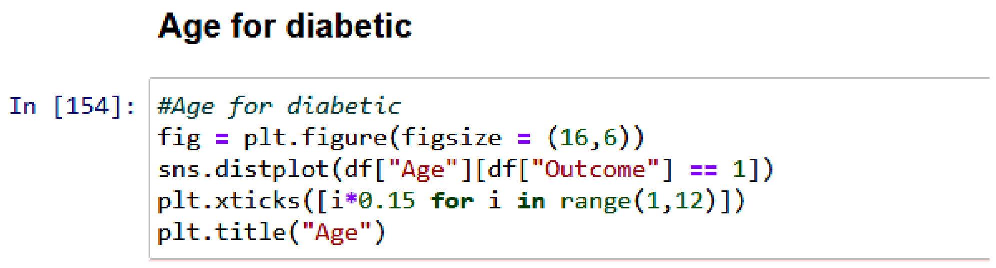


Diabetic pedigree function for diabetic

```
In [153]: #diabeticpedigreefunction for diabetic
          fig = plt.figure(figsize = (16,5))
          sns.distplot(df["DiabetesPedigreeFunction"][df["Outcome"] == 1])
          plt.xticks([i*0.15 for i in range(1,12)])
          plt.title("diabetespedigreefunction")
```

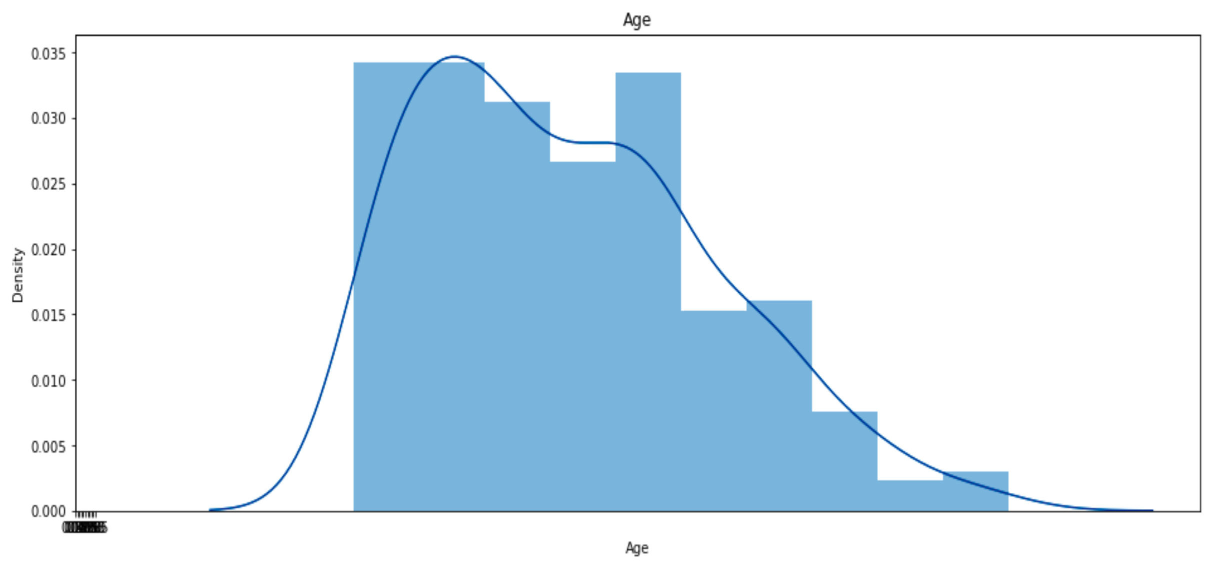


This is the representation of a Distribution plot which depicts the variation in the diabetes data distribution where **Age** and **Outcome** correlation is ‘1’.



Age for diabetic

```
In [154]: #Age for diabetic
          fig = plt.figure(figsize = (16,6))
          sns.distplot(df["Age"][df["Outcome"] == 1])
          plt.xticks([i*0.15 for i in range(1,12)])
          plt.title("Age")
```

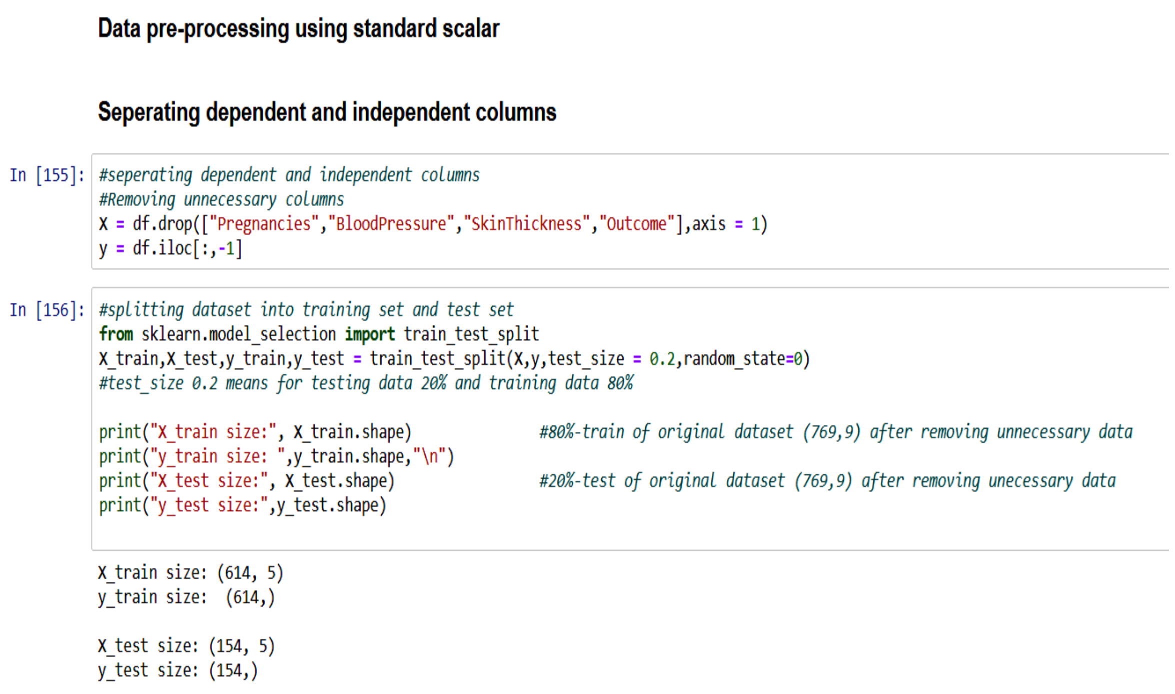


4. Separating Data set into Dependent and Independent variables:

After importing the dataset, the next step would be to identify the independent variable (X) and the dependent variable (Y). *Basically dataset might be labeled or unlabeled.* To read the columns, we will use *iloc* of *pandas* (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].In this step, the purpose is separating certain output columns and placing them into separate variable or data frame and also dropping unnecessary columns because we didn't make use of [‘Pregnancies’, BloodPressure’,’SkinThickness’] anywhere as they are not correlated to the task for checking if the patient is diabetic or not.

5. Split the dataset into training and test set:

In machine learning we usually splits the data into Training and Testing data for applying models. Generally we split the dataset into 70:30 or 80:20 (as per the requirement)it means, 70 percent data taken to train and 30 percent data taken to test. For this task, we will import *train_test_split* from *model_selection* library of scikit. Now to build our training and test sets, we will create 4 sets — X_train (training part of the features), X_test (test part of the features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices) , Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices). We will assign to them the *train_test_split*, which takes the parameters — arrays (X and Y), test_size (An ideal choice is to allocate 20% of the dataset to test set, it is usually assigned as 0.2. 0.25 would mean 25%).



Data pre-processing using standard scalar

Seperating dependent and independent columns

```
In [155]: #seperating dependent and independent columns
          #Removing unnecessary columns
          X = df.drop(["Pregnancies","BloodPressure","SkinThickness","Outcome"],axis = 1)
          y = df.iloc[:,-1]
```

```
In [156]: #splitting dataset into training set and test set
          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=0)
          #test_size 0.2 means for testing data 20% and training data 80%

          print("X_train size:", X_train.shape)          #80%-train of original dataset (769,9) after removing unnecessary data
          print("y_train size: ",y_train.shape,"\n")
          print("X_test size:", X_test.shape)            #20%-test of original dataset (769,9) after removing unecessary data
          print("y_test size:",y_test.shape)
```

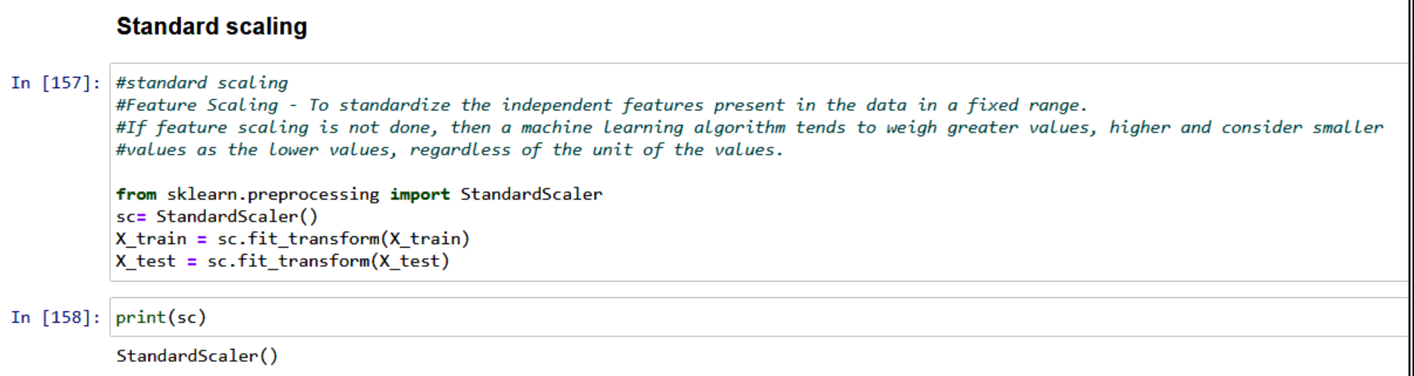
```
X_train size: (614, 5)
y_train size:  (614,)

X_test size: (154, 5)
y_test size: (154,)
```

The further step is followed by performing train-test split for the model which is 80:20 meaning 80% of training data set size and 20% of testing dataset.

So, out of total 768 rows and 5 columns, 614 rows are going to be considered as training data and 154 rows as testing data which is used for prediction.

6. Feature Scaling

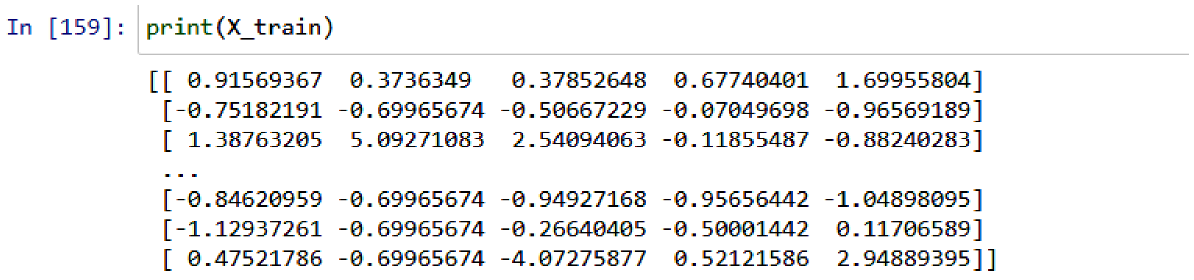
The final step of data preprocessing is to apply the very important feature scaling. Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing.



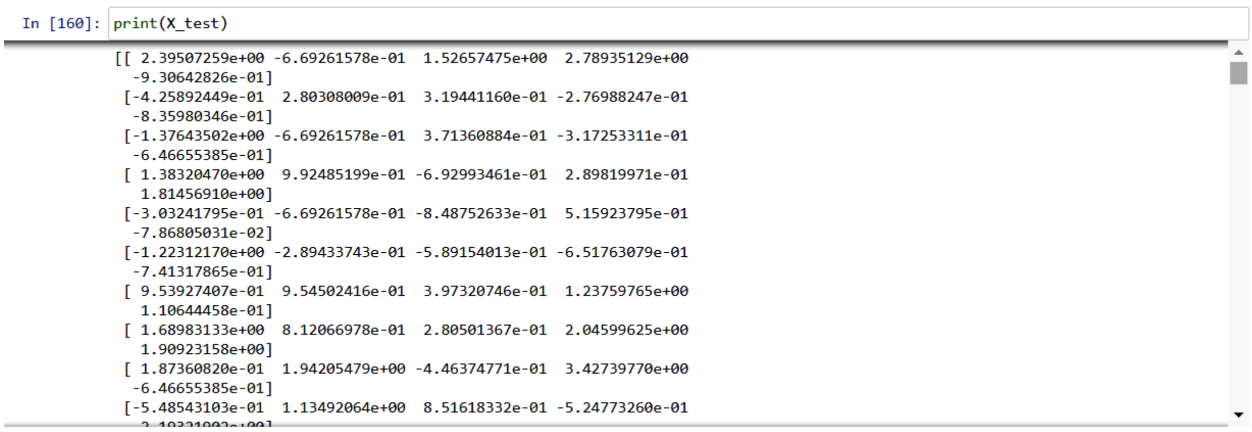
Standard scaling

```
In [157]: #standard scaling
          #Feature Scaling - To standardize the independent features present in the data in a fixed range.
          #If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller
          #values as the lower values, regardless of the unit of the values.

          from sklearn.preprocessing import StandardScaler
          sc= StandardScaler()
          X_train = sc.fit_transform(X_train)
          X_test = sc.fit_transform(X_test)
```

```
In [158]: print(sc)

          StandardScaler()
```



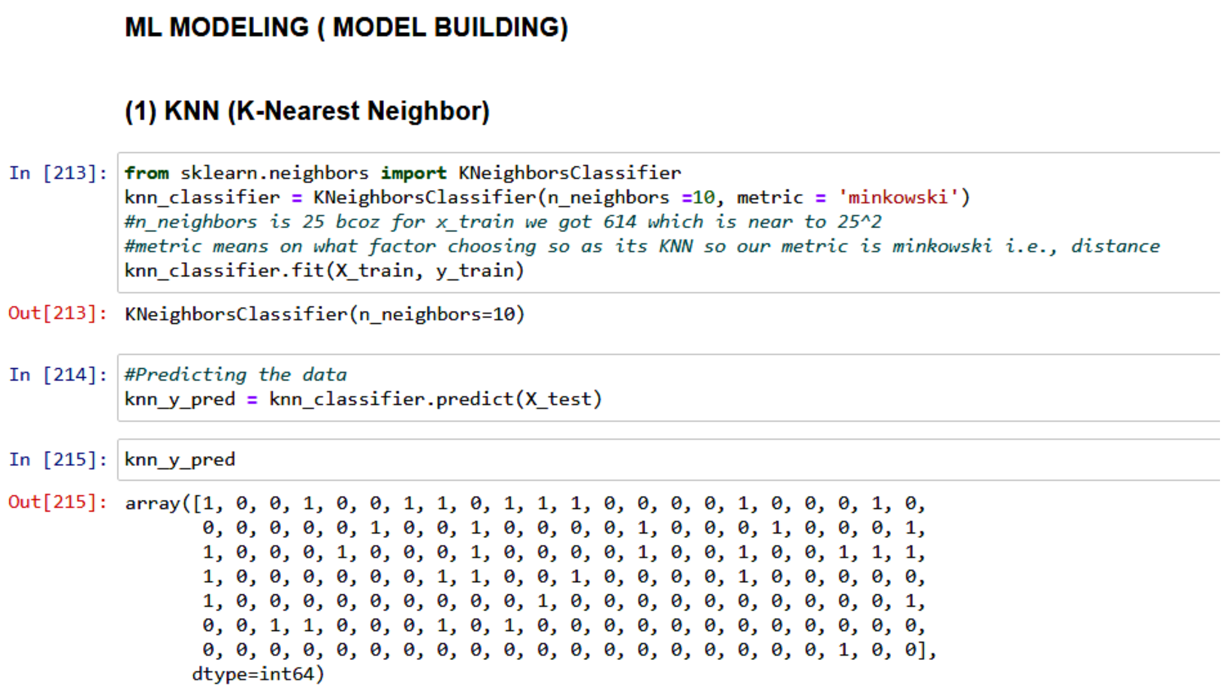
```
In [159]: print(X_train)

          [[ 0.91569367  0.3736349   0.37852648  0.67740401  1.69955804]
           [-0.75182191 -0.69965674 -0.50667229 -0.07049698 -0.96569189]
           [ 1.38763205  5.09271083  2.54094063 -0.11855487 -0.88240283]
           ...
           [-0.84620959 -0.69965674 -0.94927168 -0.95656442 -1.04898095]
           [-1.12937261 -0.69965674 -0.26640405 -0.50001442  0.11706589]
           [ 0.47521786 -0.69965674 -4.07275877  0.52121586  2.94889395]]
```



```
In [160]: print(X_test)

          [[ 2.39507259e+00 -6.69261578e-01  1.52657475e+00  2.78935129e+00
            -9.30642826e-01]
           [-4.25892449e-01  2.80308009e-01  3.19441160e-01 -2.76988247e-01
            -8.35980346e-01]
           [-1.37643502e+00 -6.69261578e-01  3.71360884e-01 -3.17253311e-01
            -6.46655385e-01]
           [ 1.38320470e+00  9.92485199e-01 -6.92993461e-01  2.89819971e-01
             1.81456910e+00]
           [-3.03241795e-01 -6.69261578e-01 -8.48752633e-01  5.15923795e-01
            -7.86805031e-02]
           [-1.22312170e+00 -2.89433743e-01 -5.89154013e-01 -6.51763079e-01
            -7.41317865e-01]
           [ 9.53927407e-01  9.54502416e-01  3.97320746e-01  1.23759765e+00
             1.10644458e-01]
           [ 1.68983133e+00  8.12066978e-01  2.80501367e-01  2.04599625e+00
             1.90923158e+00]
           [ 1.87360820e-01  1.94205479e+00 -4.46374771e-01  3.42739770e+00
            -6.46655385e-01]
           [-5.48543103e-01  1.13492064e+00  8.51618332e-01 -5.24773260e-01
```

7. Model building:

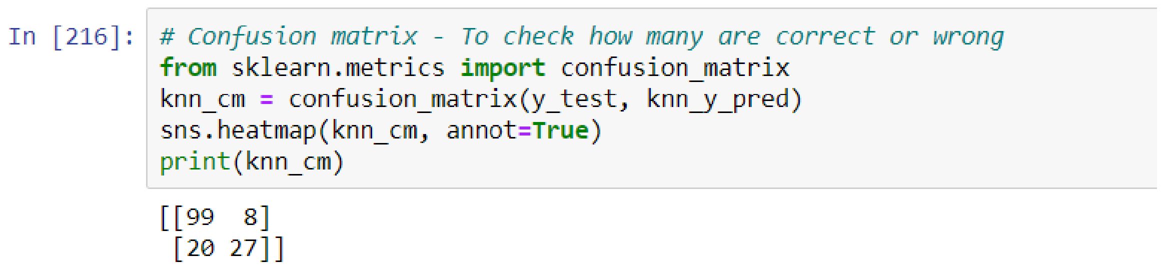
7.1. KNN: Training the model using KNN Classifier and predict the data.



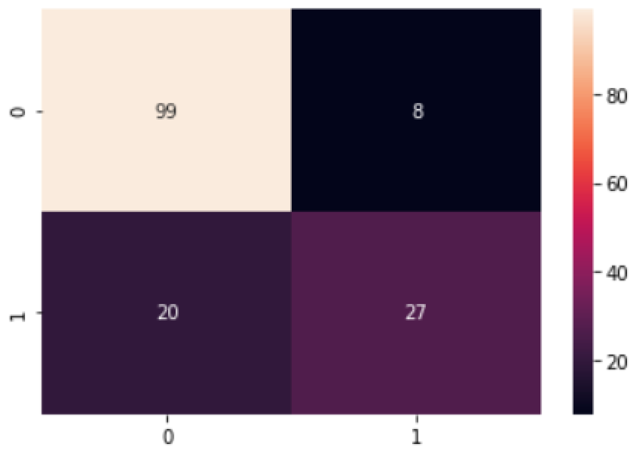
ML MODELING (MODEL BUILDING)

(1) KNN (K-Nearest Neighbor)

```
In [213]: from sklearn.neighbors import KNeighborsClassifier
          knn_classifier = KNeighborsClassifier(n_neighbors =10, metric = 'minkowski')
          #n_neighbors is 25 bcoz for x_train we got 614 which is near to 25^2
          #metric means on what factor choosing so as its KNN so our metric is minkowski i.e., distance
          knn_classifier.fit(X_train, y_train)

Out[213]: KNeighborsClassifier(n_neighbors=10)

In [214]: #Predicting the data
          knn_y_pred = knn_classifier.predict(X_test)

In [215]: knn_y_pred

Out[215]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
                 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
                dtype=int64)
```

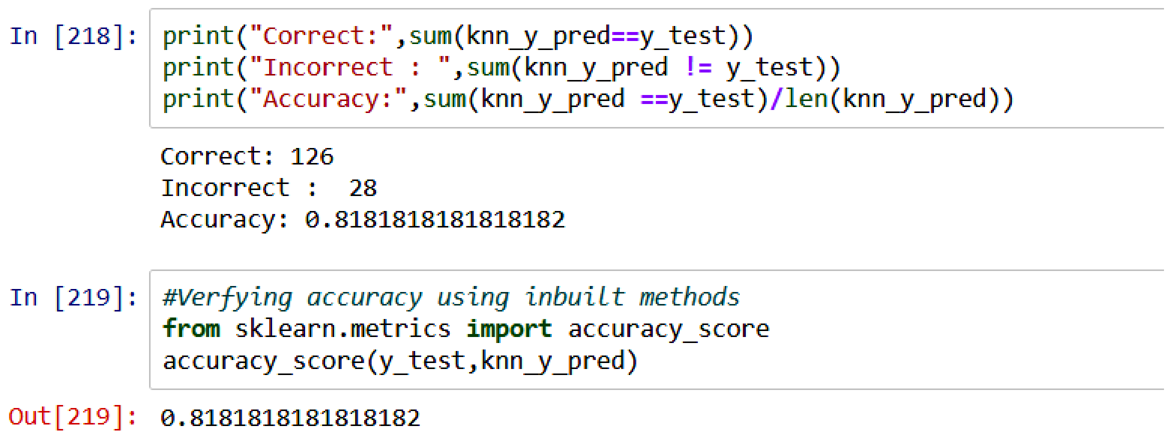


```
In [216]: # Confusion matrix - To check how many are correct or wrong
          from sklearn.metrics import confusion_matrix
          knn_cm = confusion_matrix(y_test, knn_y_pred)
          sns.heatmap(knn_cm, annot=True)
          print(knn_cm)

          [[99  8]
           [20 27]]
```



The above is the confusion matrix obtained when the KNN Classifier is used and by making use of seaborn heatmap the visualization of Confusion matrix is shown.

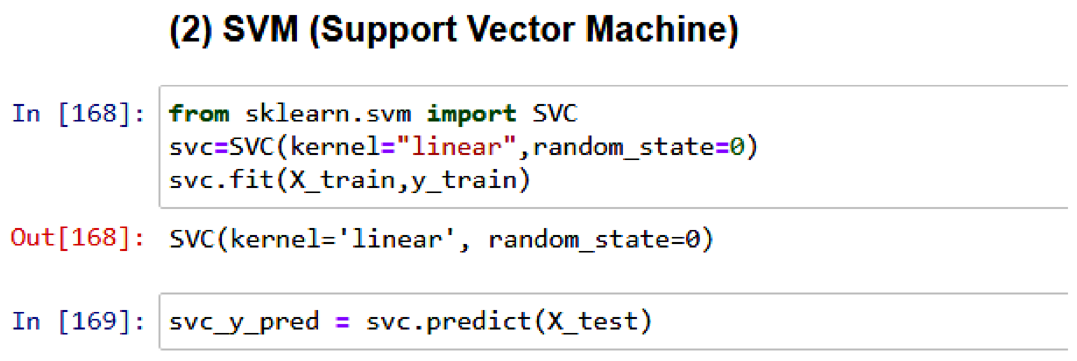
The correct and incorrect values count is shown below and also the measuring metric ‘accuracy’ is calculated.



```
In [218]: print("Correct:",sum(knn_y_pred==y_test))
          print("Incorrect : ",sum(knn_y_pred != y_test))
          print("Accuracy:",sum(knn_y_pred ==y_test)/len(knn_y_pred))

          Correct: 126
          Incorrect :  28
          Accuracy: 0.8181818181818182

In [219]: #Verfying accuracy using inbuilt methods
          from sklearn.metrics import accuracy_score
          accuracy_score(y_test,knn_y_pred)

Out[219]: 0.8181818181818182
```

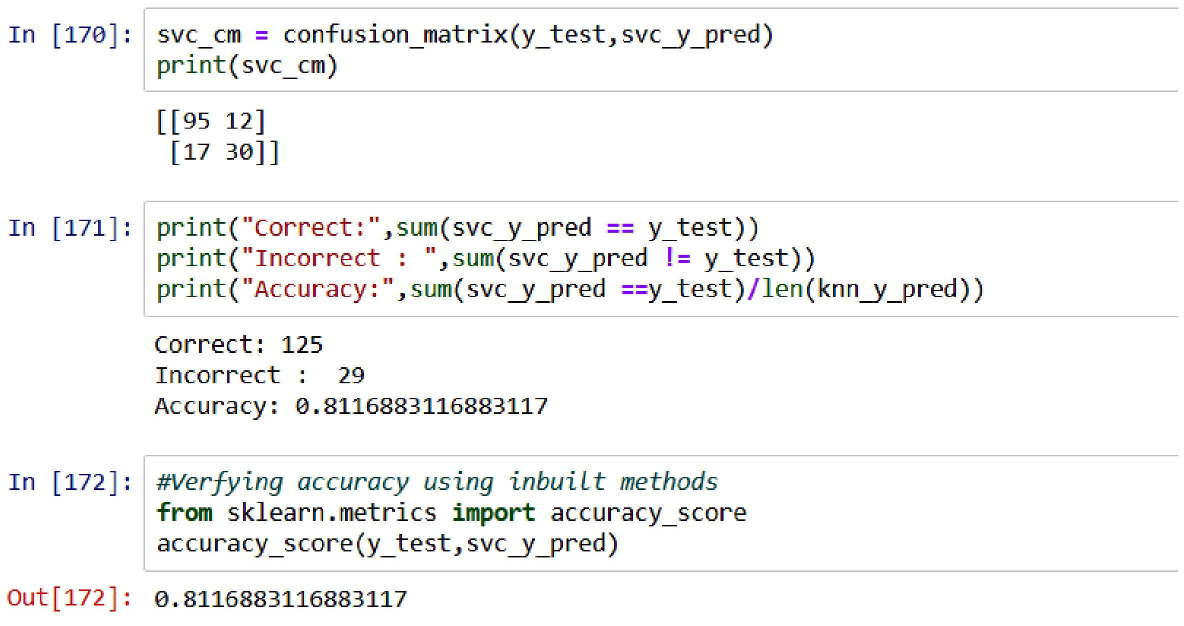
The accuracy score is calculated mathematically as well as using in built scikit learn metrics which is ‘accuracy_score’ and resulted in an accuracy of **81.81%.**

7.2. SVM: Training the model using SVM and predicting the data.



(2) SVM (Support Vector Machine)

```
In [168]: from sklearn.svm import SVC
          svc=SVC(kernel="linear",random_state=0)
          svc.fit(X_train,y_train)

Out[168]: SVC(kernel='linear', random_state=0)

In [169]: svc_y_pred = svc.predict(X_test)
```

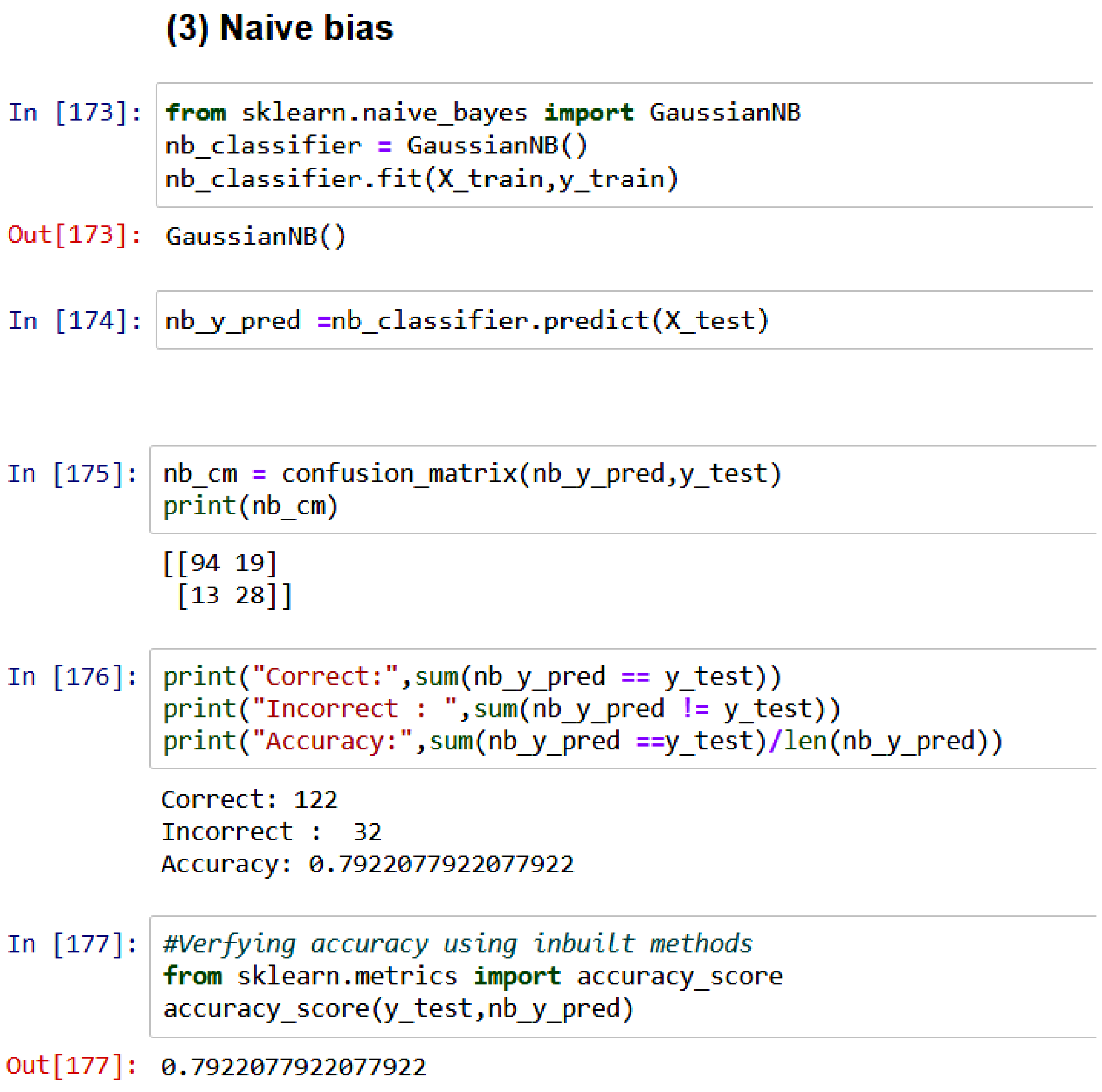


```
In [170]: svc_cm = confusion_matrix(y_test,svc_y_pred)
          print(svc_cm)

          [[95 12]
           [17 30]]

In [171]: print("Correct:",sum(svc_y_pred == y_test))
          print("Incorrect : ",sum(svc_y_pred != y_test))
          print("Accuracy:",sum(svc_y_pred ==y_test)/len(knn_y_pred))

          Correct: 125
          Incorrect :  29
          Accuracy: 0.8116883116883117

In [172]: #Verfying accuracy using inbuilt methods
          from sklearn.metrics import accuracy_score
          accuracy_score(y_test,svc_y_pred)

Out[172]: 0.8116883116883117
```

The model is fit to the parameters X_train and y_train and then predicted the result using y_test. Confusion matrix is calculated for the parameters y_test and the predicted value from SVM on x_test.

Also, the accuracy metrics are measured mathematically as well ad using sklearn metrics which resulted in accuracy of **81.16%.**

7.3. Naive Bayes: Training the model using Naive Bayes Classifier and predicting the data.



(3) Naive bias

```
In [173]: from sklearn.naive_bayes import GaussianNB
          nb_classifier = GaussianNB()
          nb_classifier.fit(X_train,y_train)

Out[173]: GaussianNB()

In [174]: nb_y_pred =nb_classifier.predict(X_test)

In [175]: nb_cm = confusion_matrix(nb_y_pred,y_test)
          print(nb_cm)

          [[94 19]
           [13 28]]

In [176]: print("Correct:",sum(nb_y_pred == y_test))
          print("Incorrect : ",sum(nb_y_pred != y_test))
          print("Accuracy:",sum(nb_y_pred ==y_test)/len(nb_y_pred))

          Correct: 122
          Incorrect :  32
          Accuracy: 0.7922077922077922

In [177]: #Verfying accuracy using inbuilt methods
          from sklearn.metrics import accuracy_score
          accuracy_score(y_test,nb_y_pred)

Out[177]: 0.7922077922077922
```

Resulted in the confusion matrix for the predicted and tested value. Calculated the correct and incorrect values as well as accuracy score which resulted in 79.22%.

Result obtained: The dataset consists of 768 records and 9 columns

The ML model KNN was able to classify patients as diabetic or not with an accuracy of 81.8%

The ML model SVM was able to classify patients as diabetic or not with an accuracy of 81.16%

The Naive Bayes Classifier was able to classify the patients as diabetic or not with an accuracy of 79.22%

CHAPTER 5

CONCLUSION AND FUTURE WORK

The main aim of this project was to design and implement Diabetes Prediction Using Machine Learning Methods and Performance Analysis of that methods and it has been achieved successfully. In this project, various machine learning algorithms are applied on the dataset and the classification has been done using various algorithms

The proposed approach uses various classification and ensemble learning method in which SVM, KNN and Navie Bias classifiers are used with accuracy of 81.8%, 81.8% and 79.22% respectively.

It is clear that the model improves accuracy and precision of diabetes prediction with this dataset compared to existing dataset. Further this work can be extended to find how likely non diabetic people can have diabetes in next few years. The Experimental results can be assist health care to predict and make early decision to cure diabetes and save humans life.

In future we can also develop and app using flask and can predict the chances of having diabetes by simply just providing the values of the glucose, insulin, age and BMI values

CHAPTER 6

REFERENCES

1. Mitushi Soni , Dr. Sunita Varma, “ Diabetes prediction using machine learning techniques “. IJERT, 2278-0181

2. Debadri Dutta, Debpriyo Paul, Parthajeet Ghosh, "Analyzing Feature Importances for Diabetes Prediction using Machine Learning". IEEE, pp 942-928, 2018.

3. K.VijiyaKumar, B.Lavanya, I.Nirmala, S.Sofia Caroline, "Random Forest Algorithm for the Prediction of Diabetes ".Proceeding of International Conference on Systems Compu- tation Automation and Networking, 2019.

4. Md. Faisal Faruque, Asaduzzaman, Iqbal H. Sarker, "Perfor- mance Analysis of Machine Learning Techniques to Predict Diabetes Mellitus". International Conference on Electrical, Computer and Communication Engineering (ECCE), 7-9 Feb- ruary, 2019.

5. Tejas N. Joshi, Prof. Pramila M. Chawan, "Diabetes Prediction Using Machine Learning Techniques".Int. Journal of Engineer- ing Research and Application, Vol. 8, Issue 1, (Part -II) Janu- ary 2018, pp.-09-13

6. Nonso Nnamoko, Abir Hussain, David England, "Predicting Diabetes Onset: an Ensemble Supervised Learning Approach ". IEEE Congress on Evolutionary Computation (CEC), 2018.

7. Deeraj Shetty, Kishor Rit, Sohail Shaikh, Nikita Patil, "Diabe- tes Disease Prediction Using Data Mining ".International Con- ference on Innovations in Information, Embedded and Com- munication Systems (ICIIECS), 2017.

8. Nahla B., Andrew et al,"Intelligible support vector machines for diagnosis of diabetes mellitus. Information Technology in Biomedicine", IEEE Transactions. 14, (July. 2010), 1114-20.

9. A.K., Dewangan, and P., Agrawal, Classification of Diabetes Mellitus Using Machine Learning Techniques, International Journal of Engineering and Applied Sciences, vol. 2, 2015.