**AI Based Diabetes Prediction System**

**Phase 3 – Development Part 1**

**STEPS INVOLVED IN BUILDING THE PROJECT:**

**Step 1: Installing Required Libraries**

In this step, we import various Python libraries that will be used throughout our data preprocessing and analysis process. Each library has a specific purpose:

numpy and pandas are fundamental libraries for numerical operations and data manipulation, respectively. statsmodels is used for statistical modeling and hypothesis testing. seaborn and matplotlib are visualization libraries which are used to create meaningful plots and graphs. sklearn (Scikit-learn) is a powerful machine learning library, which offers tools for data preprocessing, modeling, and evaluation.

warnings module is used to manage warnings that might occur during the execution of our code.

**Step 2: Uploading the Dataset**

In this step, we use the pandas library to read a CSV file containing our dataset. The data set is saved in the variable df as a DataFrame, a two-dimensional labeled data structure with columns of various types.

**Step 3: Exploratory Data Analysis**

**Displaying the First Few Rows**

In this phase, the df.head() method is used to display the dataset's initial few rows. It displays the first few rows of actual data along with the column names.

Checking the Shape of the Dataset

Using df.shape, we obtain the dimensions of the dataset. It returns a tuple representing the number of rows and columns in the DataFrame.

**Dataset Information**

df.info() provides a concise summary of the DataFrame, including the total number of non-null entries in each column, the data type of each column, and memory usage information. This helps in understanding the structure and data types of the dataset.

**Step 4: Data Preprocessing**

**Handling Missing Values**

In this step, we handle the missing values in the dataset. Missing values can negatively impact the performance of machine learning models, so it is important to handle them appropriately.

**Replacing 0 with NULL to Identify Missing Value**

We replace zero values in specific columns (Glucose, BloodPressure, SkinThickness, Insulin, BMI) with NaN (Not a Number) to indicate that these are missing values.

**Missing Value Statistics**

We use df.isnull().sum() to count the number of missing values in each column. This provides information about which columns have missing data.

**Replacing Null Values with Median Value**

We substitute the median value for each column's missing values. This is a common technique to handle missing numerical data.

**Step 5: Data Preprocessing**

**Handling Outliers**

Outliers can impact statistical analyses and machine learning models. In this step, we identify and handle outliers in the dataset.

**Finding Outliers**

We use the Interquartile Range (IQR) method to detect outliers in each feature. If a feature has values outside the calculated bounds, it is considered to have outliers.

**Treating the Outliers**

We use the Local Outlier Factor (LOF) algorithm to detect and remove outliers. This method assigns an "outlier score" to each data point, and points with scores above a certain threshold are considered as outliers.

**Step 6: Selecting Relevant Features**

Feature selection is important for building accurate predictive models. In this step, we create new features based on existing ones.

Creating New Columns (NewBMI, NewInsulinScore, NewGlucose)

NewBMI: Categorizes BMI into different ranges (Underweight, Normal, Overweight, Obesity 1, Obesity 2, Obesity 3).

NewInsulinScore: Classifies insulin levels as either "Normal" or "Abnormal" based on defined thresholds.

NewGlucose: Categorizes glucose levels as "Low", "Normal", "Overweight", "Secret", or "High".

**Step 7: Final Dataset**

Finally, the shape of the dataset is displayed after all the preprocessing steps. This provides a summary of the dataset's transformation and shows which rows were eliminated as a result of the outlier treatment.

These steps in preparing the data guarantees that the data used to train and test models is of high quality and appropriate for precise predictions.

**IMPLEMENTATION:**

**INSTALLING LIBRARIES NEEDED FOR THE PROJECT:**

import numpy as np

import pandas as pd

import statsmodels.api as sm

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import scale, StandardScaler

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score

from sklearn.metrics import confusion\_matrix, accuracy\_score, mean\_squared\_error, r2\_score, roc\_auc\_score, roc\_curve, classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import GradientBoostingClassifier

from lightgbm import LGBMClassifier

from sklearn.model\_selection import KFold

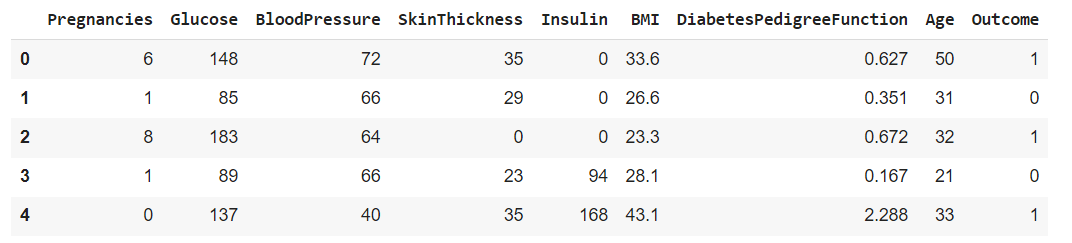
import warnings

warnings.simplefilter(action = "ignore")

**UPLOADING THE DATASET AND MAKING IT AS A DATAFRAME USING PANDAS:**

df = pd.read\_csv("/content/diabetes (1).csv")

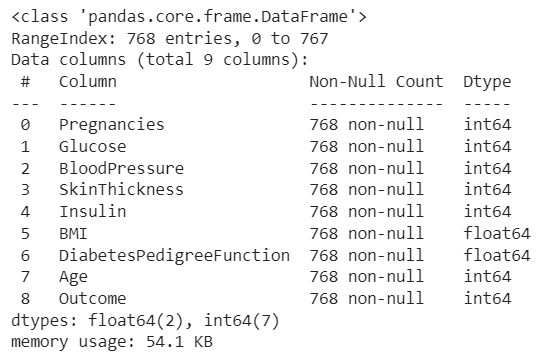
df.head()



df.shape



df.info()



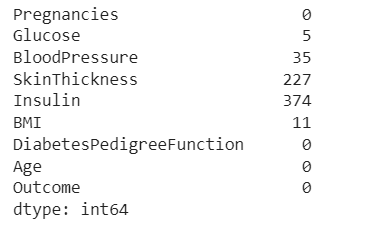
**DATA PREPROCESSING BY TREATING MISSING VALUES, OUTLIERS:**

#Replacing 0 with NULL to identify missing value

df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

#missing value statistics

df.isnull().sum()



#Replacing null values with median value

def median\_target(var):

    temp = df[df[var].notnull()]

    temp = temp[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset\_index()

    return temp

columns = df.columns

columns = columns.drop("Outcome")

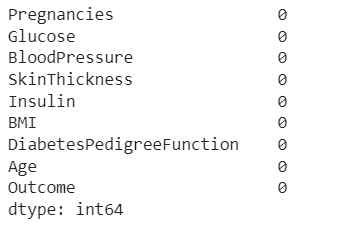
for i in columns:

    median\_target(i)

    df.loc[(df['Outcome'] == 0 ) & (df[i].isnull()), i] = median\_target(i)[i][0]

    df.loc[(df['Outcome'] == 1 ) & (df[i].isnull()), i] = median\_target(i)[i][1]

df.isnull().sum()



#To find which are the columns that has outliers

for feature in df:

    Q1 = df[feature].quantile(0.25)

    Q3 = df[feature].quantile(0.75)

    IQR = Q3-Q1

    lower = Q1- 1.5\*IQR

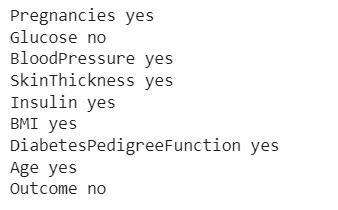
    upper = Q3 + 1.5\*IQR

    if df[(df[feature] > upper)].any(axis=None):

        print(feature,"yes")

    else:

        print(feature, "no")



#Treating the outliers

from sklearn.neighbors import LocalOutlierFactor

lof =LocalOutlierFactor(n\_neighbors= 10)

lof.fit\_predict(df)

df\_scores = lof.negative\_outlier\_factor\_

np.sort(df\_scores)[0:30]

threshold = np.sort(df\_scores)[7]

outlier = df\_scores > threshold

df = df[outlier]

#We see that the no.of rows is reduced as we have removed certain outliers

df.shape



**SELECTING RELEVANT FEATURES:**

#new column NewBMI to give range for BMI like underweight,overweight etc

NewBMI = pd.Series(["Underweight", "Normal", "Overweight", "Obesity 1", "Obesity 2", "Obesity 3"], dtype = "category")

df["NewBMI"] = NewBMI

df.loc[df["BMI"] < 18.5, "NewBMI"] = NewBMI[0]

df.loc[(df["BMI"] > 18.5) & (df["BMI"] <= 24.9), "NewBMI"] = NewBMI[1]

df.loc[(df["BMI"] > 24.9) & (df["BMI"] <= 29.9), "NewBMI"] = NewBMI[2]

df.loc[(df["BMI"] > 29.9) & (df["BMI"] <= 34.9), "NewBMI"] = NewBMI[3]

df.loc[(df["BMI"] > 34.9) & (df["BMI"] <= 39.9), "NewBMI"] = NewBMI[4]

df.loc[df["BMI"] > 39.9 ,"NewBMI"] = NewBMI[5]

#new column NewInsulinScore to tell whether the insulin level is normal or abnormal

def set\_insulin(row):

    if row["Insulin"] >= 16 and row["Insulin"] <= 166:

        return "Normal"

    else:

        return "Abnormal"

df = df.assign(NewInsulinScore=df.apply(set\_insulin, axis=1))

#new column NewGlucose to tell the level of glucose like normal,high etc

NewGlucose = pd.Series(["Low", "Normal", "Overweight", "Secret", "High"], dtype = "category")

df["NewGlucose"] = NewGlucose

df.loc[df["Glucose"] <= 70, "NewGlucose"] = NewGlucose[0]

df.loc[(df["Glucose"] > 70) & (df["Glucose"] <= 99), "NewGlucose"] = NewGlucose[1]

df.loc[(df["Glucose"] > 99) & (df["Glucose"] <= 126), "NewGlucose"] = NewGlucose[2]

df.loc[df["Glucose"] > 126 ,"NewGlucose"] = NewGlucose[3]

#The no.of columns has increased as we have new features to the dataframe

df.shape

