Diabetes Health Prediction

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

TADI SATHVIK

Registration Number : **20B91A05T8**

Under Supervision of Mr. Gundala Nagaraju, Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July ,2022 to 6th September ,2022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM – 534204,

ANDHRA PRADESH

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(Autonomous)

Chinna Amiram, Bhimavaram

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the "**Summer Internship Report"** submitted by **TADI SATHVIK, 20B91A05T8** is work done by him and submitted during 2021 - 2022 academic year, in partial fulfillment of the requirements for the award of the Summer Internship Program for **Bachelor of Technology in COMPUTER SCIENCE ENGINEERING,** at **Henotic Technology Pvt Ltd** from 07.07.2022 to 06.09.2022 .

Department Internship Coordinator **Dean-T & P Cell** **Head of the Department**

Table of Contents

Abstract

Diabetes is considered as one of the deadliest and chronic diseases which causes an increase in blood sugar. Many complications occur if diabetes remains untreated and unidentified. The tedious identifying process results in visiting of a patient to a diagnostic center and consulting doctor. But the rise in machine learning approaches solves this critical problem. The motive of this study is to design a model which can prognosticate the likelihood of diabetes in patients with maximum accuracy.

Machine learning has the potential to disrupt the medical industry by opening up new ways to handle healthcare data, transforming patient care, and streamlining administrative processes. Terabits of medical records, which previously required a human reading, can now be used as input data for machine learning in healthcare projects. ML technology has the potential to become a real game-changer in many areas of healthcare. Machine learning in healthcare is used for Diagnosis identification, medical image analysis, medical record management, Disease prediction, Mental health trends tracking, Drug development and discovery, Robotic surgery, Personalized treatment, Patient engagement.

This dataset is related to Health (Diabetes) and the goal is to determine the presence or absence of Diabetes using medical analysis. we aimed to develop and validate a prediction model for diabetes in medical industry. Based on the results, we have to configure the topmostaccuracy,f1score,roc curve finalize the model and we can predict the future outputs based on inputs of this dataset without performing any test. based on the final data in future we can test the data and predict the approximately the output.

1.0 INTRODUCTION

With the increasing power of computer technology, companies and institutions can now a days store large amounts of data at reduced cost. The amount of available data is increasing exponentially, and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies.

And the goal of this is to predict diabetes using bmi, age, sex etc..

1.1 What are the different types of Machine Learning?

There are classified mainly into three types. They are

Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labelled data. Even though the data needs to be labelled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labelled parameters required for the problem.

This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

Categories of Supervised Machine Learning:

Supervised machine learning can be classified into two types of problems, which are given below:

- Classification
- Regression

Classification:

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "Yes" or No, Male or Female, Red or Blue, etc. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc.

Some popular classification algorithms are given below:

- Random Forest Algorithm
- Decision Tree Algorithm
- Logistic Regression Algorithm
- Support Vector Machine Algorithm

Regression:

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- Simple Linear Regression Algorithm
- Multivariate Regression Algorithm
- Decision Tree Algorithm
- Lasso Regression

Applications of Supervised Learning:

- Image Segmentation
- Medical Diagnosis
- Fraud Detection
- Spam detection
- Speech Recognition

Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

Categories of Unsupervised Machine Learning:

Unsupervised Learning can be further classified into two types, which are given below:

- Clustering
- Association

Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behavior.

Some of the popular clustering algorithms are given below:

- K-Means Clustering algorithm
- Mean-shift algorithm
- DBSCAN Algorithm
- Principal Component Analysis
- Independent Component Analysis

Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production, etc.

Some popular algorithms of Association rule learning are:

- Apriori Algorithm
- Eclat
- FP-growth algorithm.

Applications of Unsupervised Learning:

- Network Analysis
- Recommendation Systems.
- Anomaly Detection
- Singular Value Decomposition

1.2 Benefits of Using Machine Learning in Health Care

There are unlimited opportunities for machine learning in health care and the use of machine learning in health care will be one of the most essential life-saving technologies ever introduced. Machine learning can reduce re-acceptance in a way that is targeted, efficient, and patient-centered. Doctors can receive daily guidelines about which patients are most likely to be accepted again and how they can reduce that risk

Advantages of machine learning in healthcare are:

1. Helps in Identifying Diseases and Diagnosis : - One of the primary advantages of machine learning in healthcare is the identification and diagnosis of diseases and ailments, which are otherwise considered to be as hard to diagnose. This can include anything from cancers that are tough to catch at the time of the initial stage to other genetic diseases

2. Drug Discovery and Manufacturing : - One of the primary clinical benefits of machine learning in healthcare lies in the early-stage drug discovery process.

3. Medical Imaging Diagnosis : - Machine learning and deep learning in health care are both responsible for the breakthrough technology called Computer Vision. This has found one of the best acceptances in the Inner Eye initiative developed by Microsoft

4. Better Radiotherapy : - One of the most sought advantages of machine learning in healthcare is in the field of Radiology. Medical image analysis has many discrete variables that can even arise big at any particular moment. There are many cancer foci, lesions, etc., which cannot be modeled with the help of complex equations.

5. Outbreak Prediction : - predicting and monitoring epidemics across the globe. In today's scenario, the scientist has to access the vast amount of data that is collected from the satellites, website information, real-time social media updates, etc. Artificial Intelligence Neural Networks help to collate this information and predict everything from malaria outbreaks to severe chronic infectious diseases.

6. Personalized Medicine : - Machine learning in health care helps in the customized treatments that can not only be more efficient and effective by pairing individual health with predictive analytics, but it is also ripe for further research and better assessment of the disease

7. Robotic Surgery : - Robotic Surgery is one of the most significant benchmark machine learning applications in the sector of the healthcare market. This application will now also become with some of the promising areas soon. This application can be evenly divided into four subcategories, such as surgical skill evaluation, surgical workflow modeling, automatic suturing, and improvements of the robotic surgical materials. Suturing is the process with the help of sewing up an open wound. The automation of the suturing may help to reduce the surgical process length and surgeon fatigue.

1.3 About Industry (Health Care):

The healthcare industry (also called the medical industry or health economy) is an aggregation and integration of sectors within the economic system that provides goods and services to treat patients with curative, preventive, rehabilitative, and palliative care. It includes the generation and commercialization of goods and services lending themselves to maintaining and re-establishing health.

The healthcare industry is one of the world's largest and fastest-growing industries. Consuming over 10 percent of the gross domestic product (GDP) of most developed nations, health care can form an enormous part of a country's economy. U.S. health care spending grew 4.6 percent in 2019, reaching $3.8 trillion or $11,582 per person. As a share of the nation's Gross Domestic Product, health spending accounted for 17.7 percent. The per capita expenditure on health and pharmaceuticals in OECD countries has steadily grown from a couple of hundred in the 1970s to an average of US$4'000 per year in current purchasing power parities

1.3.1 AI / ML Role in Health Care:

From chronic diseases, like cancer, to radiology, AI is being leveraged to deploy efficient and precise inventions that will help take care of patients suffering from these diseases and hopefully find a cure for them. AI provides several advantages over traditional methods of analytics and making clinical decisions. AI algorithms make the systems more precise as they get the opportunity to understand training data, which further helps humans get unprecedented insights into treatment variability, care processes, diagnostics, and patient results.

Artificial intelligence in healthcare is an overarching term used to describe the use of machine-learning algorithms and software, or artificial intelligence (AI), to mimic human cognition in the analysis, presentation, and comprehension of complex medical and health care data. Specifically, AI is the ability of computer algorithms to approximate conclusions based solely on input data.

The primary aim of health-related AI applications is to analyze relationships between clinical techniques and patient outcomes. AI programs are applied to practices such as diagnostics, treatment protocol development, drug development, personalized medicine, and patient monitoring and care. What differentiates AI technology from traditional technologies in healthcare is the ability to gather data, process it, and produce a well-defined output for the end-user. AI does this through machine learning algorithms and deep learning. These processes can recognize patterns in behavior and create their logic. To gain useful insights and predictions, machine learning models must be trained using extensive amounts of input data. AI algorithms behave differently from humans in two ways: (1) algorithms are literal: once a goal is set, the algorithm learns exclusively from the input data and can only understand what it has been programmed to do, (2) and some deep learning algorithms are black boxes; algorithms can predict with extreme precision, but offer little to no comprehensible explanation to the logic behind its decisions aside from the data and type of algorithm used.

Diabetes Dataset content:

- Gender: Gender of the passengers (Female, IsDiabetic : 0 = no diabetes 1 = pre-diabetes or diabetes
- Age : 13-level age category(1 = 18-24;...;9 = 60-64;13 = 80 or older)
- Education : Education level scale 1-6 1=Never attended school or only kindergarten;2=elementary etc.
- Income : Income level scale 1-8 1 = less than 10,000;5=less than 10,000;5=less than 35,000;8 = $75,000 or more
- HighBP : 0 = no high BP 1 = high BP
- HighChol : 0 = no high cholesterol 1 = high cholesterol
- CholCheck : 0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years
- BMI : Body Mass Index
- Smoker : Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no 1 = yes
- Stroke : (Ever told) you had a stroke. 0 = no 1 = yes
- HeartDiseaseorAttack : coronary heart disease (CHD) or myocardial infarction (MI) 0 = no 1 = yes
- PhysActivity : physical activity in past 30 days - not including job 0 = no 1 = yes
- Fruits : Consume Fruit 1 or more times per day 0 = no 1 = yes
- Veggies : Consume Vegetables 1 or more times per day 0 = no 1 = yes
- HvyAlcoholConsump : (adult men >=14 drinks per week and adult women>=7 drinks per week) 0 = no 1 = yes
- AnyHealthcare : Have any kind of health care coverage, including health insurance, prepaid plans such as HMO, etc. 0 = no 1 = yes
- NoDocbcCost : Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? 0 = no 1 = yes
- GenHlth : Would you say that in general your health is: scale 1-5 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor
- MentHlth : days of poor mental health scale 1-30 days
- PhysHlth : physical illness or injury days in past 30 days scale 1-30
- DiffWalk : Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes
- Sex : 0 = female 1 = male
- Age : 13-level age category 1 = 18-24 9 = 60-64 13 = 80 or older
- Education : Education level scale 1-6 1 = Never attended school or only kindergarten 2 = elementary etc.
- Income : Income scale (scale 1-8 1 = less than 10,000 5=lessthan10,000 5=lessthan35,000 8 = $75,000 or more

2.0 Diabetes Health Prediction:

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. According to International Diabetes Federation 382 million people are living with diabetes across the whole world. By 2035, this will be doubled as 592 million. Diabetes mellitus orimply diabetes is a disease caused due to the increase level of blood glucose. Various traditional methods, based on physical and chemical tests, are available for diagnosing diabetes. However, early prediction of diabetes is quite challenging task for medical practitioners due to complex interdependence on various factors as diabetes affects human organs such as kidney, eye, heart, nerves, foot etc. Data science methods have the potential to benefit other scientific fields by shedding new light on common questions. One such task is to help make predictions on medical data. Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. The aim of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine learning techniques. This project aims to predict diabetes via three different supervised machine learning methods including: SVM, Logistic regression, KNN.

2.1. Main Drivers for AI in Diabetes Health Analysis:

Predictive modelling allows for simultaneous consideration of many variables and quantification of their overall effect. When many satisfactory surveys are analyzed, patterns regarding the characteristics of the satisfaction that drive loss development begin to emerge.

- BP(Blood Pressure)
- Cholesterol
- General Health
- Mental Health
- Physical Health
- Age
- BMI
- Heart Disease or Attack
- Stroke

2.2. Internship Project – Data Link

The internship project data has taken from Kaggle, and the link is
https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

3.0 AI / ML Modelling and Results:

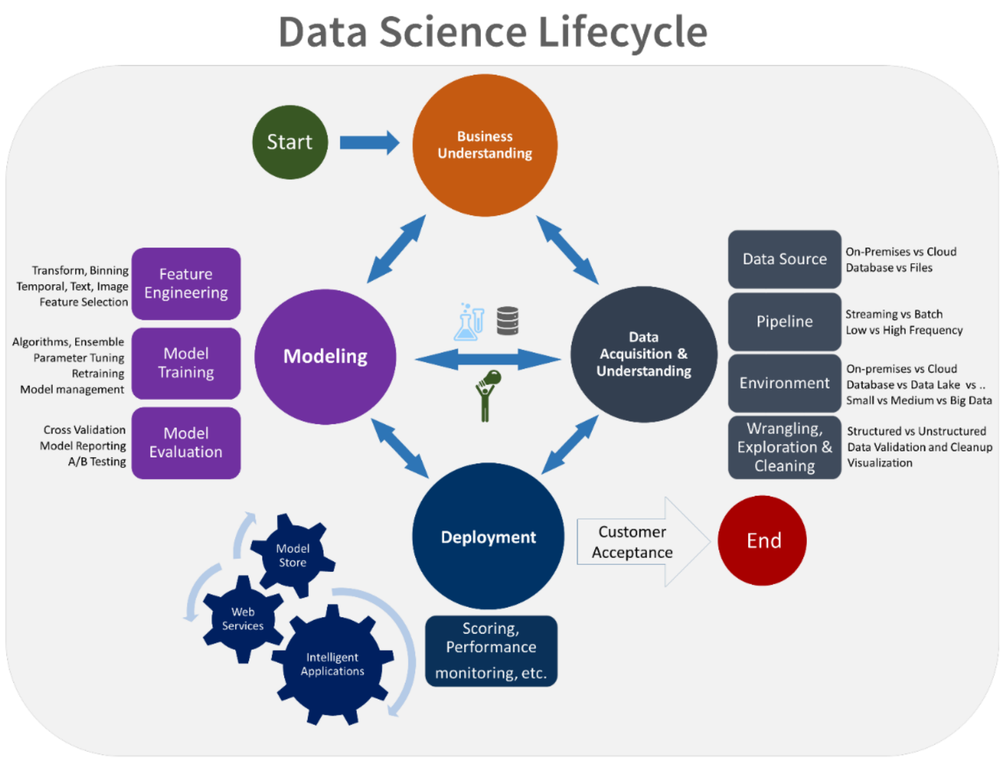
3.1. Problem of Statement:

- This dataset contains 70,000 rows and 22 features along with target variable. In this dataset, the target variable is Diabetes binary. This Diabetes binary contains two possible outcomes 0:Absence of diabetes 1:Presence of diabetes. And Remaining 21 features are BMI, Age, Sex, High BP, High Chol etc.. And this 21 features may affect the target variable. By using the machine learning algorithms like logistic regression, Naïve Bayes, Random Forest etc.. By using those algorithms we have to take the best algorithm that fit into it.
- So, our final moto is to predict the Diabetes result of the patient based on the reports provided.

3.2. Data Science Project Life Cycle:

- What is a Data Science Project Lifecycle?

  Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data.



3.2.1 Data Exploratory Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

3.2.2. Data Pre-processing:

We removed variables which does not affect our target variable (satisfaction)as they may add noise and increase our computation time, we checked the data for anomalous data points and outliers. We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1. Check the Duplicate and low variation data.

You have a dataset and must check there is duplicates or not. The Python pandas library has a method for it, that is **duplicated ().** It checks for the duplicates rows and returns True and False. For the data frame object. If you use the method sum () along with it, then it will return the total number of the duplicates in the dataset.

Now you have known that there are duplicates in the dataset and want to remove the duplicates from the dataset. There are two ways you can remove duplicates. One is deleting the entire rows and other is removing the column with the most duplicates.

3.2.2.2. Identify and address the missing variables:

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset. In Pandas, usually, missing values are represented by **NaN**.

Checking the missing values:

The first step in handling missing values is to look at the data carefully and find out all the missing values.dataframe.isnull().sum() will tell about missing values in the entire column.

Figure Out How to Handle the Missing Data:

Analyze each column with missing values carefully to understand the reasons behind the missing values as it is crucial to find out the strategy for handling the missing values.
There are 2 primary ways of handling missing values:

1. Deleting the Missing values
2. Imputing the Missing Values

3.2.2.3. Handling of Outliers:

As outliers are very different values—abnormally low or abnormally high—their presence can often skew the results of statistical analyses on the dataset. This could lead to less effective and less useful models. But dealing with outliers often requires domain expertise, and none of the outlier detection techniques should be applied *without* understanding the data distribution and the use case.

3.2.2.4. Categorical data and Encoding Techniques:

What is Categorical Data?

Since we are going to be working on categorical variables in this article, here is a quick refresher on the same with a couple of examples. Categorical variables are usually represented as 'strings' or 'categories' and are finite in number. Here are a few examples:

1. The city where a person lives: Delhi, Mumbai, Ahmedabad, Bangalore, etc.
2. The department a person works in: Finance, Human resources, IT, Production.
3. The highest degree a person has: High school, Diploma, Bachelors, Masters, PhD.
4. The grades of a student: A+, A, B+, B, B- etc.

In the above examples, the variables only have definite possible values. Further, we can see there are two kinds of categorical data-

• Ordinal Data: The categories have an inherent order

• Nominal Data: The categories do not have an inherent order

Label Encoding:

• We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence encoding should reflect the sequence.

• In Label encoding, each label is converted into an integer value. We will create a variable that contains the categories representing the education qualification of a person.

Binary Encoding:

• Binary encoding is a combination of Hash encoding and one-hot encoding. In this encoding scheme, the categorical feature is first converted into numerical using an ordinal encoder. Then the numbers are transformed in the binary number. After that binary value is split into different columns.

• Binary encoding works well when there are a high number of categories. For example, the cities in a country where a company supplies its products

3.2.2.5. Feature Scaling:

Why Feature Scaling?

Real Life Datasets have many features with a wide range of values like for example let's consider the house price prediction dataset. It will have many features like no. of. bedrooms, square feet area of the house, etc. As you can guess, the no. of bedrooms will vary between 1 and 5, but the square feet area will range from 500-2000. This is a huge difference in the range of both features.

Many machine learning algorithms that are using Euclidean distance as a metric to calculate the similarities will fail to give a reasonable recognition to the smaller feature, in this case, the number of bedrooms, which in the real case can turn out to be an important metric. There are several ways to do feature scaling. I will be discussing the top 5 of the most used feature scaling techniques.

E.g.: Linear Regression, Logistic Regression, KNN

3.2.3. Selection of Dependent and Independent variables:

The dependent or target variable here is satisfaction Target which tells us a which tells us that the Customer is satisfied, neutral or dissatisfied.

The independent variables are selected after doing exploratory data analysis.

3.2.4 Data Sampling Methods:

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1. Stratified sampling :

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.

It can be performed using strata function from the library sampling.

3.2.4.2. Simple random sampling :

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5 Models Used for Development:

We built our predictive models by using the following ten algorithms:

3.2.5.1. Model 01 (Logistic Regression)

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class. The also gives us p-value of the variables which tells us about significance of each independent variable.

3.2.5.2. Model 02 (Decision Tree Classifier)

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed based on features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

3.2.5.3. Model 03 (Random Forest Classifier)

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Bierman and Adele Cutler. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.
- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.4. Model 04 (Extra Tree Classifier)

This class implements a meta estimator that fits several randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

3.2.5.5. Model 05 (KNN Classifier)

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most like the available categories’-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset KNN-algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much like the new data.

3.2.5.6. Model 06(Gaussian Naive Bayes)

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve Bayes. While other functions are used to estimate data distribution, Gaussian or normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

What is the Naive Bayes Algorithm?

Naive Bayes is a probabilistic machine learning algorithm that can be used in several classification tasks. Typical applications of Naive Bayes are classification of documents, filtering spam, prediction and so on. This algorithm is based on the discoveries of Thomas Bayes and hence its name.

The name “Naïve” is used because the algorithm incorporates features in its model that are independent of each other. Any modifications in the value of one feature do not directly impact the value of any other feature of the algorithm. The main advantage of the Naïve Bayes algorithm is that it is a simple yet powerful algorithm.

It is based on the probabilistic model where the algorithm can be coded easily, and predictions did quickly in real-time. Hence this algorithm is the typical choice to solve real-world problems as it can be tuned to respond to user requests instantly. But before we dive deep into Naïve Bayes and Gaussian Naïve Bayes, we must know what is meant by conditional probability.

3.2.5.7. Model 07 (Bagging classifier)

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

3.2.5.8. Model 08 (Light GBM)

Light GBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of Light GBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.

Gradient-based One Side Sampling Technique for Light GBM:

Different data instances have varied roles in the computation of information gain. The instances with larger gradients (i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drop those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

3.2.5.9. Model 09 (SVC)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

3.2.5.10 Model 10 (Gradient Boosting Classifier)

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n classes_ regression trees are fit on the negative gradient of the loss function, e.g. binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced.

3.3.AI/ ML Models Analysis and Final Results:

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models. We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model.

Code for Data Preprocessing

```
# Importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Ignore harmless warnings
import warnings
warnings.filterwarnings("ignore")
# Set to display all the columns in dataset
pd.set_option("display.max_columns", None)
# Import psql to run queries
import pandasql as psql
# load the dataset
diab = pd.read_csv(r"C:\Users\srini\satvik\diabetes_binary_5050split.csv", header=0)
```

```
# copy the file to back-up file
diab_bk = diab.copy()
# display first 5 records
diab.head()
# Display diabetes data information
diab.info()
# Count the target or dependent variable by '0' & '1' and
# their proportion (> 10 : 1, then the dataset is imbalance dataset)
diabetes_count = diab.Diabetes_binary.value_counts()
print('Class 0:', diabetes_count[0])
print('Class 1:', diabetes_count[1])
print('Proportion:', round(diabetes_count[0] /diabetes_count[1], 2), ': 1')
print('Total loans Trans:', len(diab))
#Missing Values
diab.isnull().sum()
#For outliers
# Identify the numerical and categorical variables
num_vars = diab.columns[diab.dtypes != 'object']
cat_vars = diab.columns[diab.dtypes == 'object']
print(num_vars)
print(cat_vars)
# Eliminate the outlier in 'BMI' and write data to new file
BMI_UL = round(diab.BMI.mean() + 3 * diab.BMI.std(),3)
BMI_LL = round(diab.BMI.mean() - 3 * diab.BMI.std(),3)
diab1 = diab[(diab.BMI > BMI_LL) & (diab.BMI < BMI_UL)]
diab1.shape
# Eliminate the outlier in 'MentHlth' and write data to new file
MentHlth_UL = round(diab1.MentHlth.mean() + 3 * diab1.MentHlth.std(),3)
MentHlth_LL = round(diab1.MentHlth.mean() - 3 * diab1.MentHlth.std(),3)
diab2 = diab1[(diab1.MentHlth > MentHlth_LL) & (diab1.MentHlth < MentHlth_UL)]
diab2.shape
# Eliminate the outlier in 'PhysHlth' and write data to new file
PhysHlth_UL = round(diab2.PhysHlth.mean() + 3 * diab2.PhysHlth.std(),3)
PhysHlth_LL = round(diab2.PhysHlth.mean() - 3 * diab2.PhysHlth.std(),3)
diab3 = diab2[(diab2.PhysHlth > PhysHlth_LL) & (diab2.PhysHlth < PhysHlth_UL)]
diab3.shape
# Eliminate the outlier in 'Age' and write data to new file
Age_UL = round(diab3.Age.mean() + 3 * diab3.Age.std(),3)
Age_LL = round(diab3.Age.mean() - 3 * diab3.Age.std(),3)
diab4 = diab3[(diab3.Age > Age_LL) & (diab3.Age < Age_UL)]
diab4.shape
#Duplicates
# Displaying Duplicate values with in dataset
diabetes_dup =diab4[diab4.duplicated(keep='last')]
diabetes_dup
```

```
# Remove the identified duplicate records
diab4 = diab4.drop_duplicates()
# Display the shape of the dataset
diab4.shape
# Re-setting the row index
diab4 = diab4.reset_index(drop=True)
diab4.head()
#Scaling
# Identify variables for scaling
cols1 = ['PhysHlth','Age','BMI']
# Delete variables which are not influencing the target variable
del diab4['Fruits']
del diab4['AnyHealthcare']
del diab4['NoDocbcCost']
del diab4['Sex']
del diab4['MentHlth']

# Identify the independent and Target (dependent) variables
IndepVar = []
for col in diab4.columns:
    if col != 'Diabetes_binary':
        IndepVar.append(col)
TargetVar = 'Diabetes_binary'
x = diab4[IndepVar]
y = diab4[TargetVar]
# Splitting the dataset into train and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state =42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
# Scaling the features by using MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
mmscaler = MinMaxScaler(feature_range=(0, 1))
x_train[cols1] = mmscaler.fit_transform(x_train[cols1])
x_train = pd.DataFrame(x_train)
x_test[cols1] = mmscaler.fit_transform(x_test[cols1])
x_test = pd.DataFrame(x_test)
```

3.3.1 Logistic Regression Python Code

```
#modeling
from sklearn.linear_model import LogisticRegression
models = LogisticRegression()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
```

```
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.2 Decision Tree Classifier Python Code

```
#modeling
from sklearn.tree import DecisionTreeClassifier
models = DecisionTreeClassifier()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
```

```
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.3 Random Forest Classifier Python Code

```
#modeling
from sklearn.ensemble import RandomForestClassifier
models = RandomForestClassifier()
models.fit(x_train, y_train)
```

```
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
```

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.4 Extra Trees Classifier Python Code

```
#modeling
from sklearn.ensemble import ExtraTreesClassifier
models = ExtraTreesClassifier ()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
```

```
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.5 KNeighbors Classifier Python Code

```
#modeling
from sklearn.neighbors import KNeighborsClassifier
models = KNeighborsClassifier (n_neighbours=5)
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.6 Gaussian Naïve Bayes Classifier Python Code

```
#modeling
from sklearn.naive_bayes import GaussianNB
models = GaussianNB()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
#Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
```

```
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.7 Bagging Classifier Python Code

```
#modeling
from sklearn.ensemble import BaggingClassifier
modelBAG = BaggingClassifier()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.8 LGBM Classifier Python Code

```
#modeling
import lightgbm as lgb
models = LGBMClassifier()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
```

```
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.9 SVC Python Code

```
#modeling
from sklearn.svm import SVC
models = SVC(probability=True)
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

3.3.10 Gradient Boosting Classifier Python Code

```
#modeling
from sklearn.ensemble import GradientBoostingClassifier
models = GradientBoostingClassifier()
models.fit(x_train, y_train)
# Prediction
y_pred = models.predict(x_test)
y_pred_prob = models.predict_proba(x_test)
# Print the model name
print('Model Name: ', models)
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
```

```
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

4.0 Conclusion and Future work:

The model results are in the following order by considering the model accuracy, F1 score and RoC AUC score.

1. **SVC**
2. **Gradient Boosting Classifier**
3. **LGBM Classifier**

We recommend model - SVC as a best fit for the given diabetes health prediction data set. It predicts the outcome of diabetes with high accuracy and F1 score based on the given parameters.

5.0 References

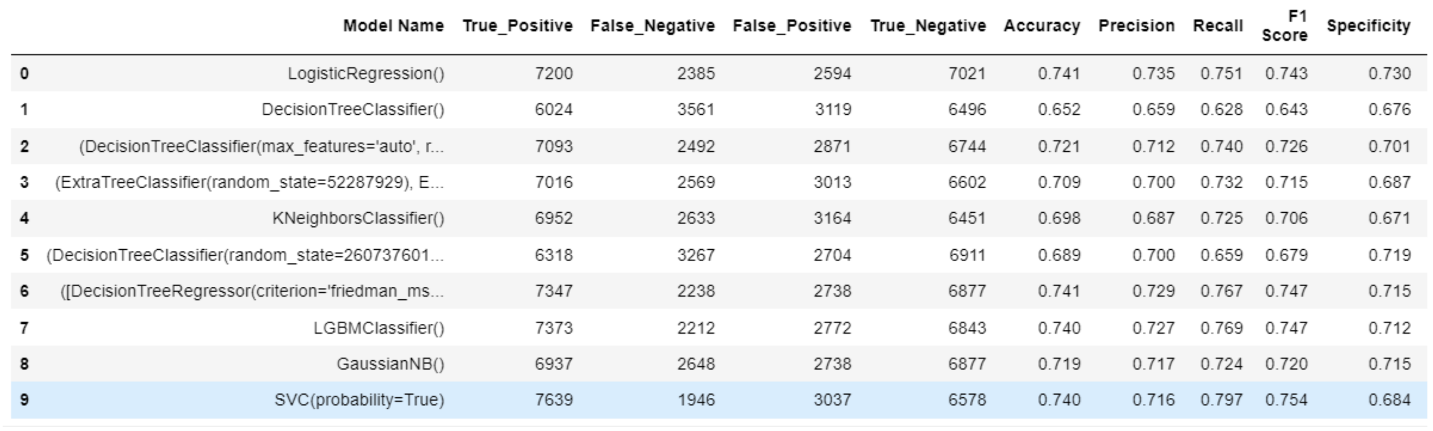
1. MachineLearningAlgorithms-https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/

2. About Diabetes - https://www.healthline.com/health/diabetes

3. Outliers-https://www.freecodecamp.org/news/how-to-detect-outliers-in-machine-learning/#:~:text=Outliers%20are%20those%20data%20points,data%20entry%2C%20or%20erroneous%20observations.

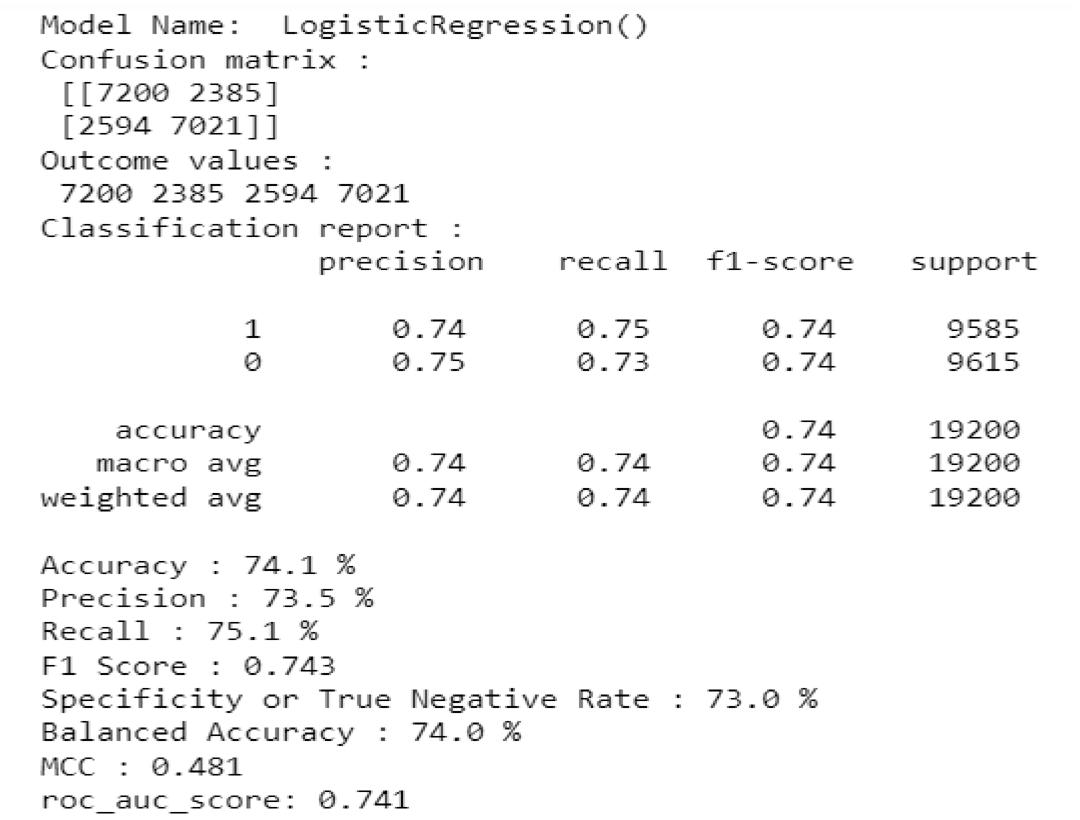
4. Featurescaling - https://www.geeksforgeeks.org/ml-feature-scaling-part-2/

6.0 Appendices

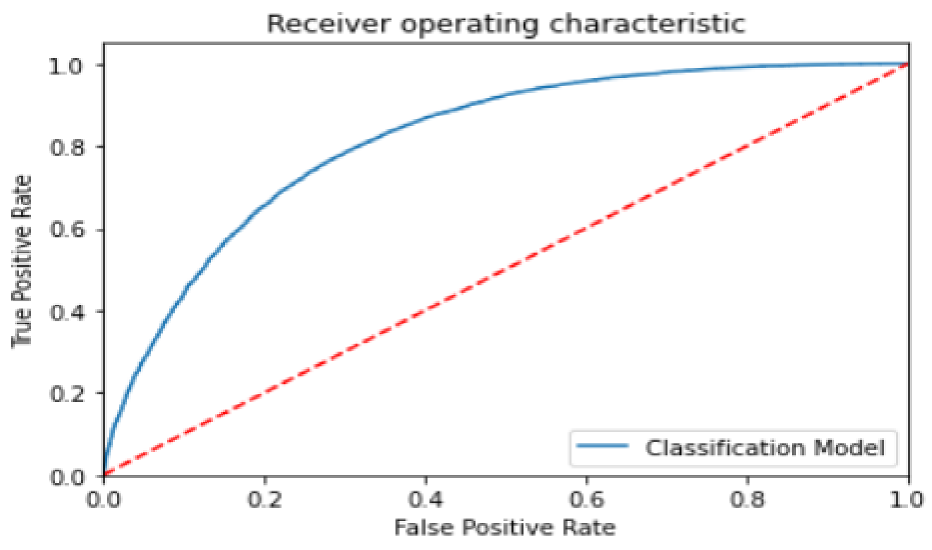


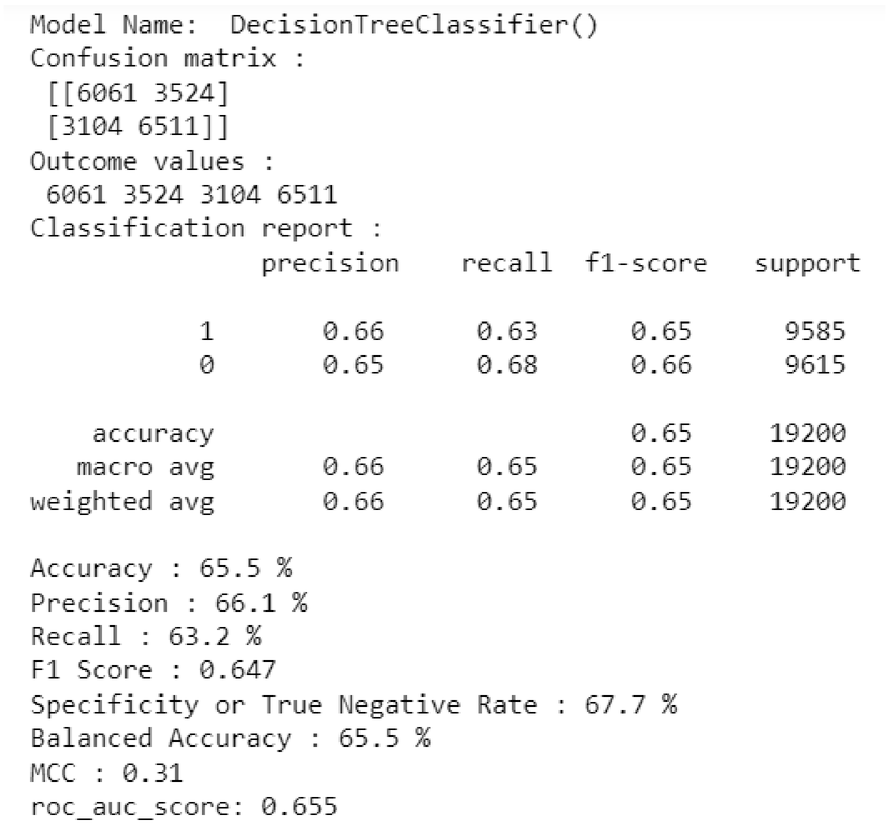
	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity
0	LogisticRegression()	7200	2385	2594	7021	0.741	0.735	0.751	0.743	0.730
1	DecisionTreeClassifier()	6024	3561	3119	6496	0.652	0.659	0.628	0.643	0.676
2	(DecisionTreeClassifier(max_features='auto', r...	7093	2492	2871	6744	0.721	0.712	0.740	0.726	0.701
3	(ExtraTreeClassifier(random_state=52287929), E...	7016	2569	3013	6602	0.709	0.700	0.732	0.715	0.687
4	KNeighborsClassifier()	6952	2633	3164	6451	0.698	0.687	0.725	0.706	0.671
5	(DecisionTreeClassifier(random_state=260737601...	6318	3267	2704	6911	0.689	0.700	0.659	0.679	0.719
6	([DecisionTreeRegressor(criterion='friedman_ms...	7347	2238	2738	6877	0.741	0.729	0.767	0.747	0.715
7	LGBMClassifier()	7373	2212	2772	6843	0.740	0.727	0.769	0.747	0.712
8	GaussianNB()	6937	2648	2738	6877	0.719	0.717	0.724	0.720	0.715
9	SVC(probability=True)	7639	1946	3037	6578	0.740	0.716	0.797	0.754	0.684

6.1 Python Code Results

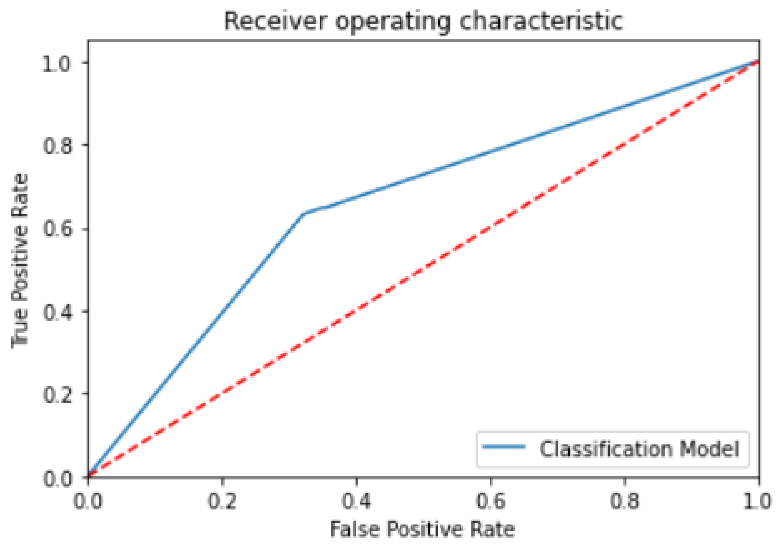


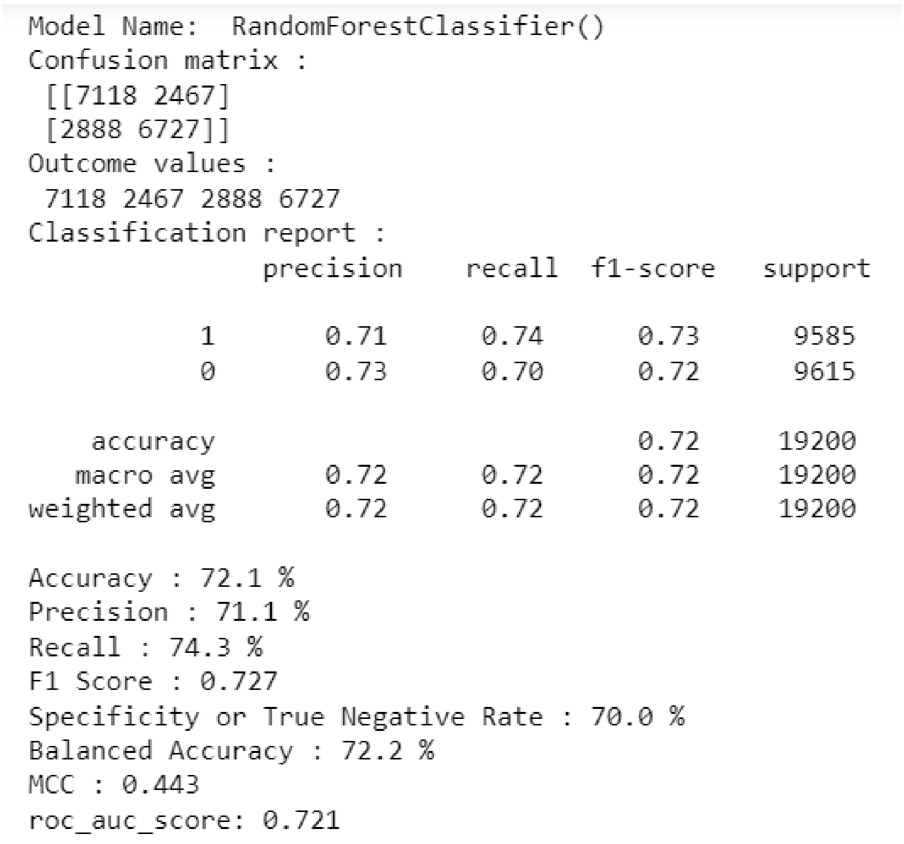
```
Model Name:  LogisticRegression()
Confusion matrix :
 [[7200 2385]
 [2594 7021]]
Outcome values :
 7200 2385 2594 7021
Classification report :
               precision    recall  f1-score   support

           1       0.74      0.75      0.74      9585
           0       0.75      0.73      0.74      9615

    accuracy                           0.74     19200
   macro avg       0.74      0.74      0.74     19200
weighted avg       0.74      0.74      0.74     19200

Accuracy : 74.1 %
Precision : 73.5 %
Recall : 75.1 %
F1 Score : 0.743
Specificity or True Negative Rate : 73.0 %
Balanced Accuracy : 74.0 %
MCC : 0.481
roc_auc_score: 0.741
```



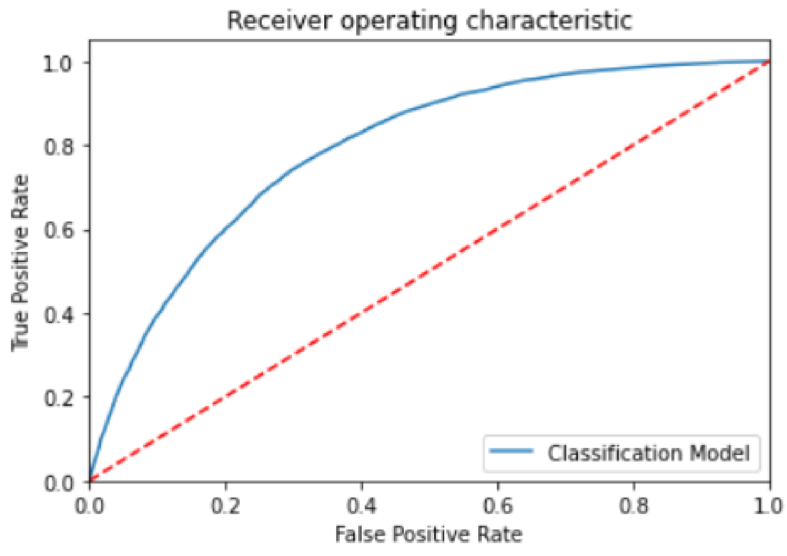


```
Model Name:  DecisionTreeClassifier()
Confusion matrix :
 [[6061 3524]
 [3104 6511]]
Outcome values :
 6061 3524 3104 6511
Classification report :
               precision    recall  f1-score   support

           1       0.66      0.63      0.65      9585
           0       0.65      0.68      0.66      9615

    accuracy                           0.65     19200
   macro avg       0.66      0.65      0.65     19200
weighted avg       0.66      0.65      0.65     19200

Accuracy : 65.5 %
Precision : 66.1 %
Recall : 63.2 %
F1 Score : 0.647
Specificity or True Negative Rate : 67.7 %
Balanced Accuracy : 65.5 %
MCC : 0.31
roc_auc_score: 0.655
```

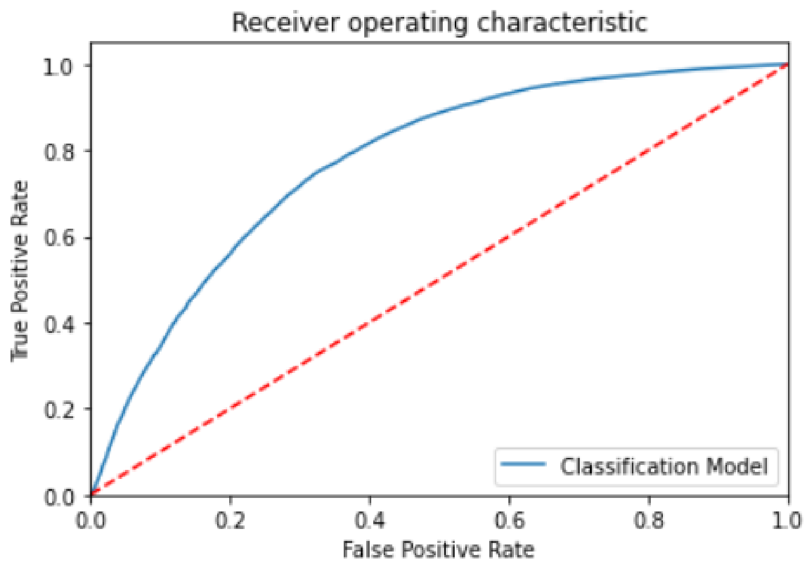




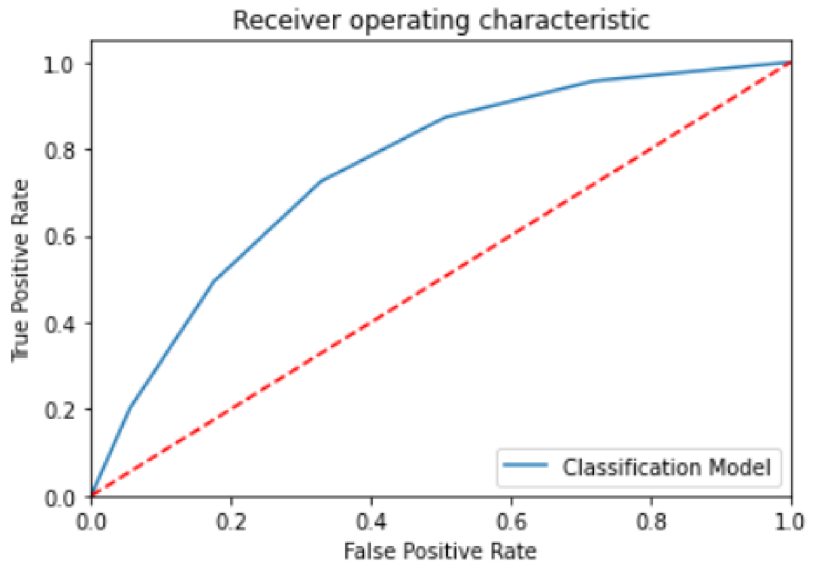
```
Model Name:  RandomForestClassifier()
Confusion matrix :
 [[7118 2467]
 [2888 6727]]
Outcome values :
 7118 2467 2888 6727
Classification report :
               precision    recall  f1-score   support

           1       0.71      0.74      0.73      9585
           0       0.73      0.70      0.72      9615

    accuracy                           0.72     19200
   macro avg       0.72      0.72      0.72     19200
weighted avg       0.72      0.72      0.72     19200

Accuracy : 72.1 %
Precision : 71.1 %
Recall : 74.3 %
F1 Score : 0.727
Specificity or True Negative Rate : 70.0 %
Balanced Accuracy : 72.2 %
MCC : 0.443
roc_auc_score: 0.721
```



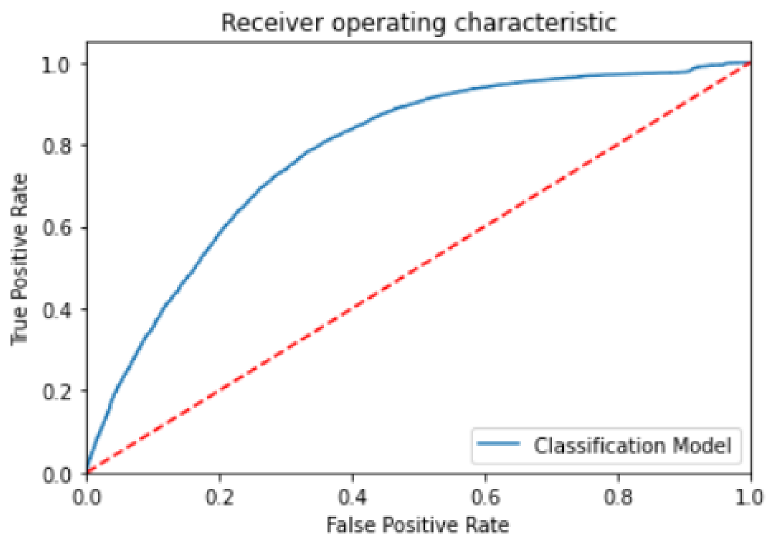
```
Model Name:  ExtraTreesClassifier()
Confusion matrix :
 [[7035 2550]
 [3016 6599]]
Outcome values :
 7035 2550 3016 6599
Classification report :
               precision    recall  f1-score   support

           1       0.70      0.73      0.72      9585
           0       0.72      0.69      0.70      9615

    accuracy                           0.71     19200
   macro avg       0.71      0.71      0.71     19200
weighted avg       0.71      0.71      0.71     19200

Accuracy : 71.0 %
Precision : 70.0 %
Recall : 73.4 %
F1 Score : 0.717
Specificity or True Negative Rate : 68.6 %
Balanced Accuracy : 71.0 %
MCC : 0.421
roc_auc_score: 0.71
```

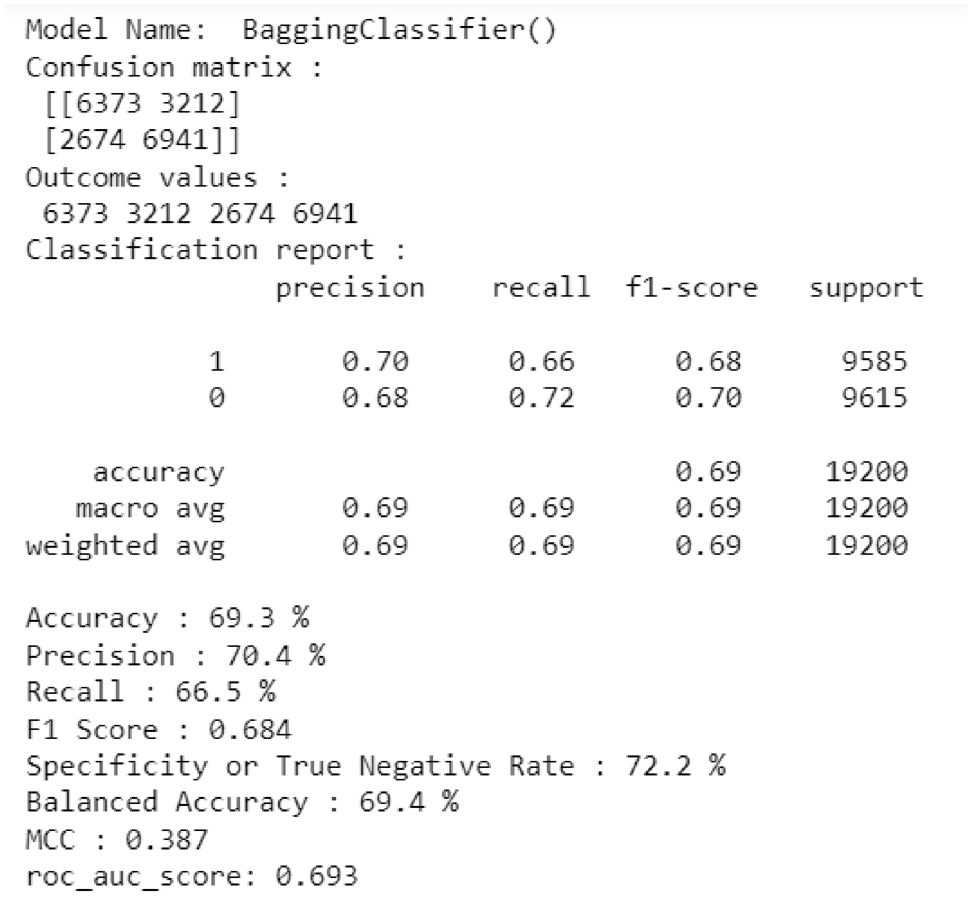


```
Model Name:  KNeighborsClassifier()
Confusion matrix :
 [[6952 2633]
 [3164 6451]]
Outcome values :
 6952 2633 3164 6451
Classification report :
               precision    recall  f1-score   support

           1       0.69      0.73      0.71      9585
           0       0.71      0.67      0.69      9615

    accuracy                           0.70     19200
   macro avg       0.70      0.70      0.70     19200
weighted avg       0.70      0.70      0.70     19200

Accuracy : 69.8 %
Precision : 68.7 %
Recall : 72.5 %
F1 Score : 0.706
Specificity or True Negative Rate : 67.1 %
Balanced Accuracy : 69.8 %
MCC : 0.397
roc_auc_score: 0.698
```

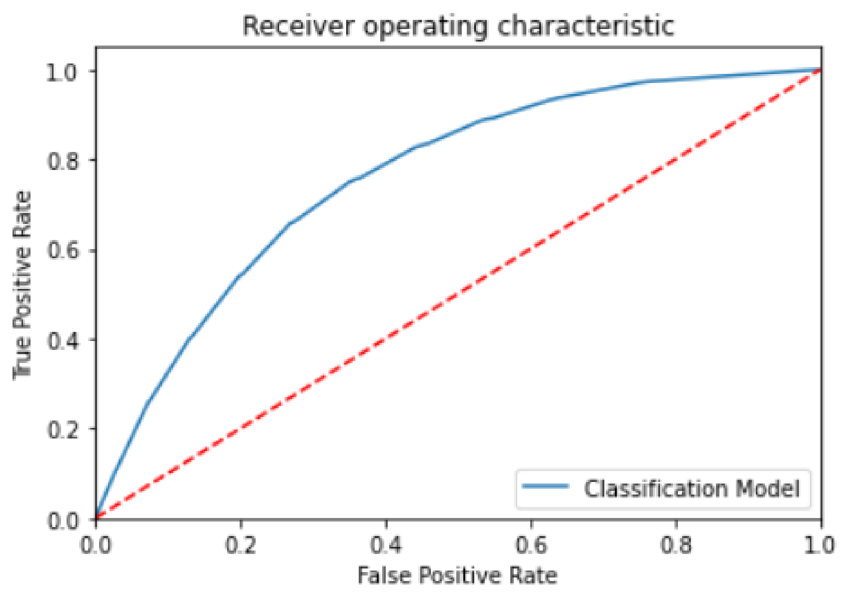


```
Model Name:  GaussianNB()
Confusion matrix :
 [[6937 2648]
 [2738 6877]]
Outcome values :
 6937 2648 2738 6877
Classification report :
               precision    recall  f1-score   support

           1       0.72      0.72      0.72      9585
           0       0.72      0.72      0.72      9615

    accuracy                           0.72     19200
   macro avg       0.72      0.72      0.72     19200
weighted avg       0.72      0.72      0.72     19200

Accuracy : 71.9 %
Precision : 71.7 %
Recall : 72.4 %
F1 Score : 0.72
Specificity or True Negative Rate : 71.5 %
Balanced Accuracy : 72.0 %
MCC : 0.439
roc_auc_score: 0.719
```

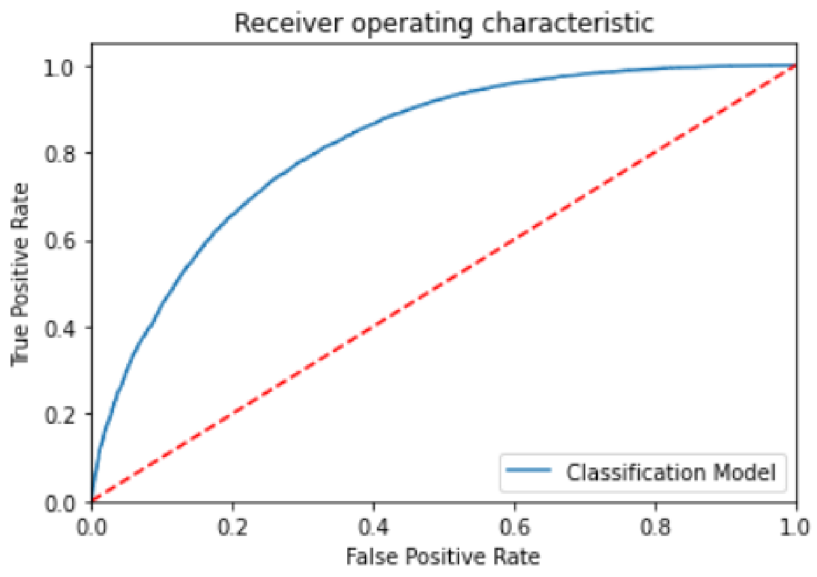




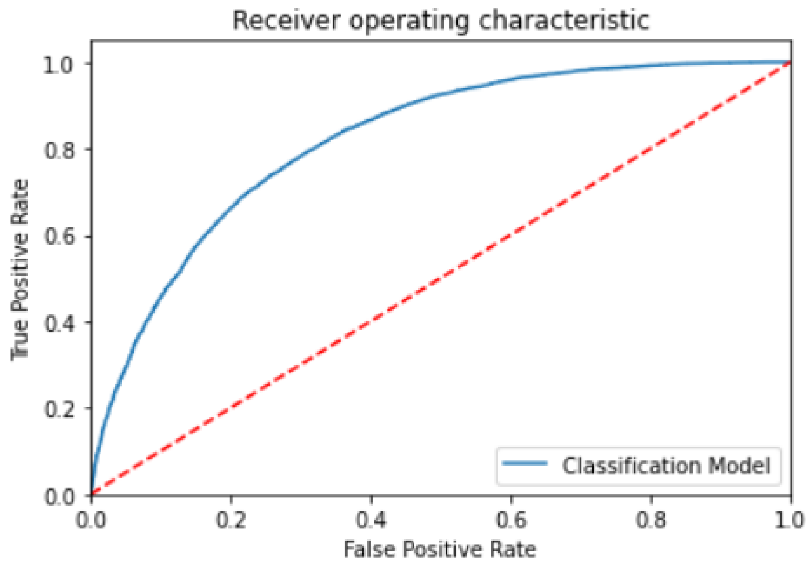
```
Model Name:  BaggingClassifier()
Confusion matrix :
 [[6373 3212]
 [2674 6941]]
Outcome values :
 6373 3212 2674 6941
Classification report :
               precision    recall  f1-score   support

           1       0.70      0.66      0.68      9585
           0       0.68      0.72      0.70      9615

    accuracy                           0.69     19200
   macro avg       0.69      0.69      0.69     19200
weighted avg       0.69      0.69      0.69     19200

Accuracy : 69.3 %
Precision : 70.4 %
Recall : 66.5 %
F1 Score : 0.684
Specificity or True Negative Rate : 72.2 %
Balanced Accuracy : 69.4 %
MCC : 0.387
roc_auc_score: 0.693
```



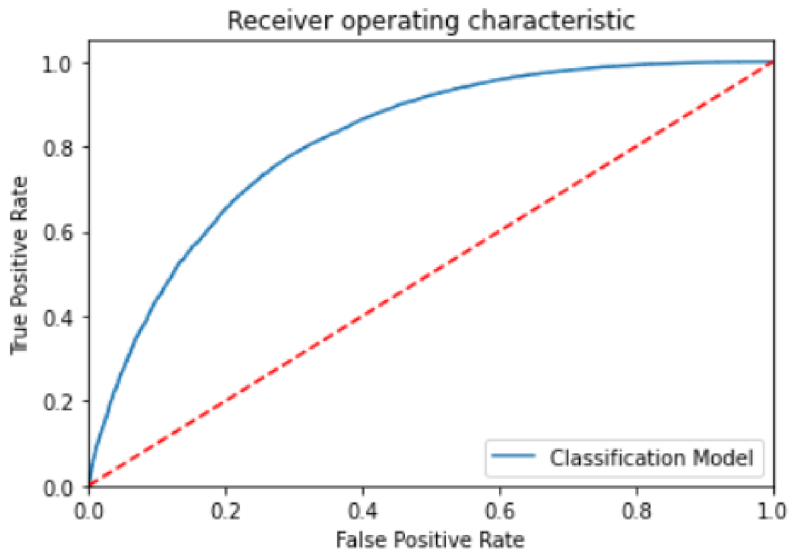
```
Model Name:  LGBMClassifier()
Confusion matrix :
 [[7373 2212]
 [2772 6843]]
Outcome values :
 7373 2212 2772 6843
Classification report :
               precision    recall  f1-score   support

           1       0.73      0.77      0.75      9585
           0       0.76      0.71      0.73      9615

    accuracy                           0.74     19200
   macro avg       0.74      0.74      0.74     19200
weighted avg       0.74      0.74      0.74     19200

Accuracy : 74.0 %
Precision : 72.7 %
Recall : 76.9 %
F1 Score : 0.747
Specificity or True Negative Rate : 71.2 %
Balanced Accuracy : 74.0 %
MCC : 0.482
roc_auc_score: 0.74
```



```
Model Name:  GradientBoostingClassifier()
Confusion matrix :
 [[7347 2238]
 [2738 6877]]
Outcome values :
 7347 2238 2738 6877
Classification report :
               precision    recall  f1-score   support

           1       0.73      0.77      0.75      9585
           0       0.75      0.72      0.73      9615

    accuracy                           0.74     19200
   macro avg       0.74      0.74      0.74     19200
weighted avg       0.74      0.74      0.74     19200

Accuracy : 74.1 %
Precision : 72.9 %
Recall : 76.7 %
F1 Score : 0.747
Specificity or True Negative Rate : 71.5 %
Balanced Accuracy : 74.1 %
MCC : 0.482
roc_auc_score: 0.741
```



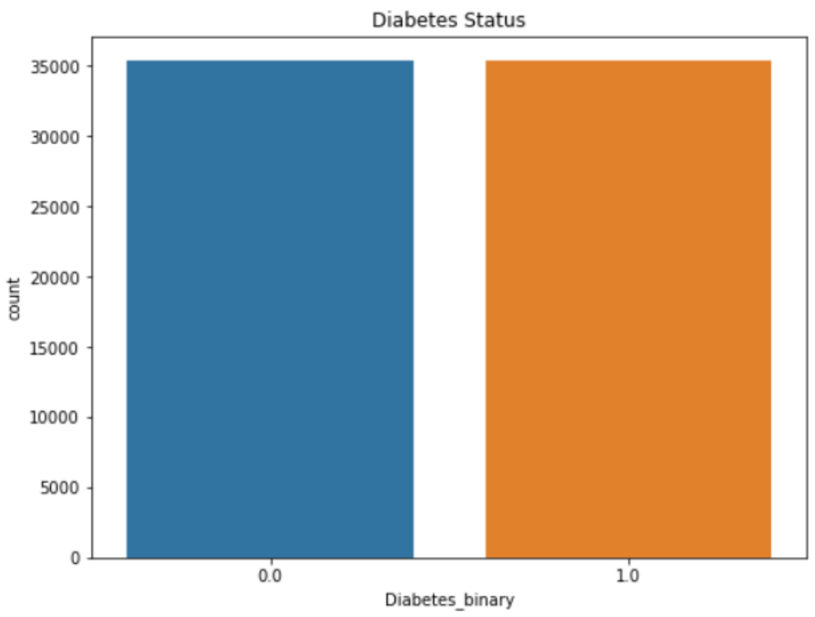
```
Model Name:  SVC(probability=True)
Confusion matrix :
 [[7639 1946]
 [3037 6578]]
Outcome values :
 7639 1946 3037 6578
Classification report :
               precision    recall  f1-score   support

           1       0.72      0.80      0.75      9585
           0       0.77      0.68      0.73      9615

    accuracy                           0.74     19200
   macro avg       0.74      0.74      0.74     19200
weighted avg       0.74      0.74      0.74     19200

Accuracy : 74.0 %
Precision : 71.6 %
Recall : 79.7 %
F1 Score : 0.754
Specificity or True Negative Rate : 68.4 %
Balanced Accuracy : 74.0 %
MCC : 0.484
roc_auc_score: 0.741
```



6.2 List of Charts

6.2.1 Chart 1: Diabetes Plot

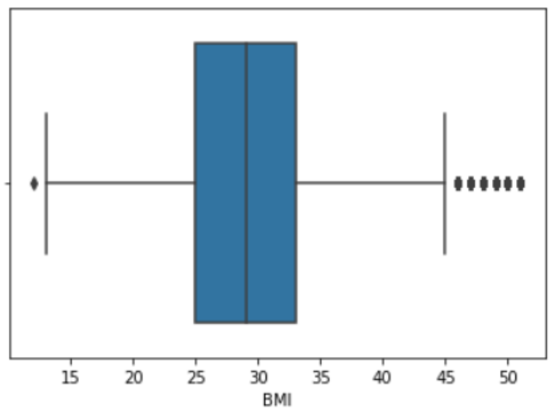
```
plt.figure(figsize=(8,6))
sns.countplot(diab['Diabetes_binary'])
plt.title("Diabetes Status")
plt.show()
```



6.2.2 Chart 2: Verification of Outliers on BMI

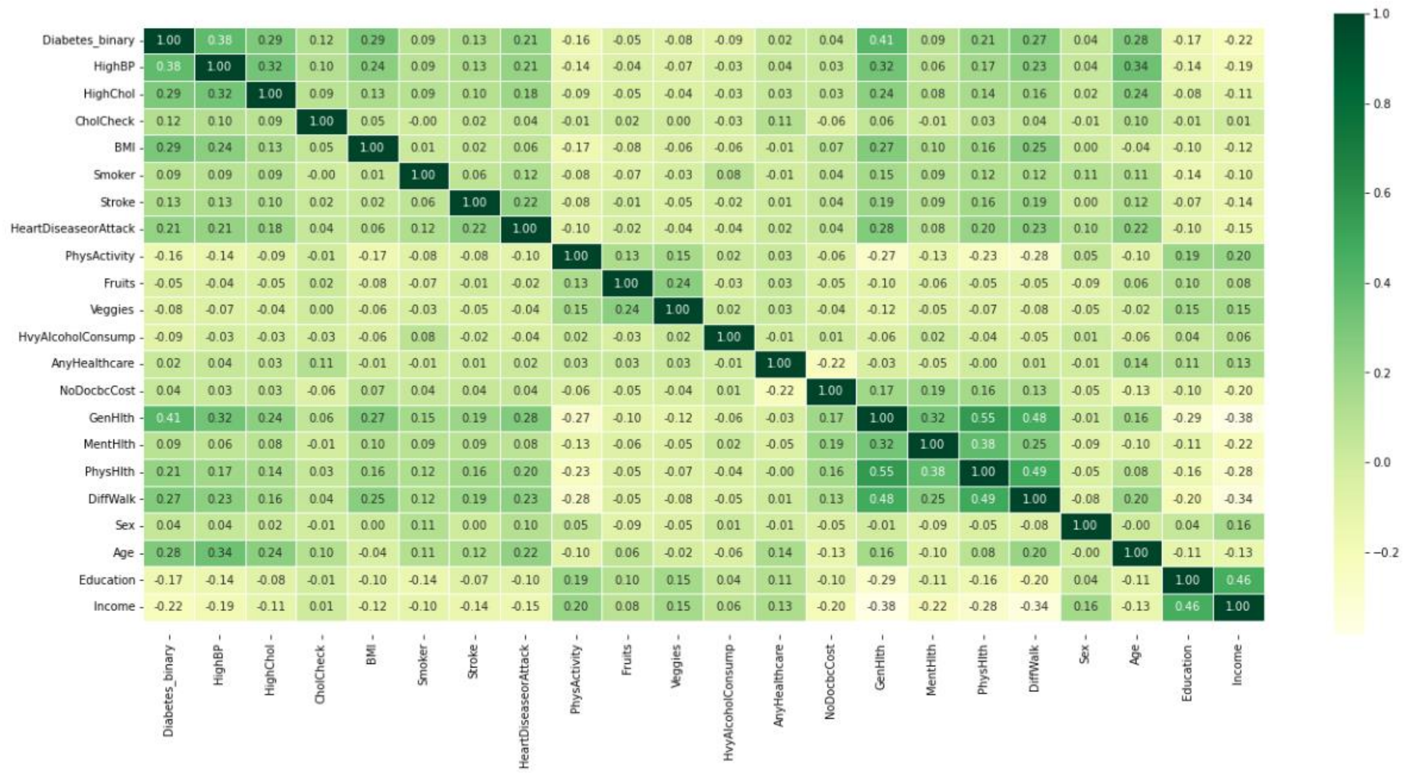
```
# Display the box plot to show the outliers
sns.boxplot(x=diab4["BMI"])
```

```
<AxesSubplot:xlabel='BMI'>
```



6.2.3 Chart 3: heatmap for Correlation between variables

```
corr_matrix = diab.corr()
fig, ax = plt.subplots(figsize=(22, 10))
ax = sns.heatmap(corr_matrix,annot=True,linewidths=0.5,fmt=".2f",cmap="YlGn");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5);
```



	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
Diabetes_binary	1.00	0.38	0.29	0.12	0.29	0.09	0.13	0.21	-0.16	-0.05	-0.08	-0.09	0.02	0.04	0.41	0.09	0.21	0.27	0.04	0.28	-0.17	-0.22
HighBP	0.38	1.00	0.32	0.10	0.24	0.09	0.13	0.21	-0.14	-0.04	-0.07	-0.03	0.04	0.03	0.32	0.06	0.17	0.23	0.04	0.34	-0.14	-0.19
HighChol	0.29	0.32	1.00	0.09	0.13	0.09	0.10	0.18	-0.09	-0.05	-0.04	-0.03	0.03	0.03	0.24	0.08	0.14	0.16	0.02	0.24	-0.08	-0.11
CholCheck	0.12	0.10	0.09	1.00	0.05	-0.00	0.02	0.04	-0.01	0.02	0.00	-0.03	0.11	-0.06	0.06	-0.01	0.03	0.04	-0.01	0.10	-0.01	0.01
BMI	0.29	0.24	0.13	0.05	1.00	0.01	0.02	0.06	-0.17	-0.08	-0.06	-0.06	-0.01	0.07	0.27	0.10	0.16	0.25	0.00	-0.04	-0.10	-0.12
Smoker	0.09	0.09	0.09	-0.00	0.01	1.00	0.06	0.12	-0.08	-0.07	-0.03	0.08	-0.01	0.04	0.15	0.09	0.12	0.12	0.11	0.11	-0.14	-0.10
Stroke	0.13	0.13	0.10	0.02	0.02	0.06	1.00	0.22	-0.08	-0.01	-0.05	-0.02	0.01	0.04	0.19	0.09	0.16	0.19	0.00	0.12	-0.07	-0.14
HeartDiseaseorAttack	0.21	0.21	0.18	0.04	0.06	0.12	0.22	1.00	-0.10	-0.02	-0.04	-0.04	0.02	0.04	0.28	0.08	0.20	0.23	0.10	0.22	-0.10	-0.15
PhysActivity	-0.16	-0.14	-0.09	-0.01	-0.17	-0.08	-0.08	-0.10	1.00	0.13	0.15	0.02	0.03	-0.06	-0.27	-0.13	-0.23	-0.28	0.05	-0.10	0.19	0.20
Fruits	-0.05	-0.04	-0.05	0.02	-0.08	-0.07	-0.01	-0.02	0.13	1.00	0.24	-0.03	0.03	-0.05	-0.10	-0.06	-0.05	-0.05	-0.09	0.06	0.10	0.08
Veggies	-0.08	-0.07	-0.04	0.00	-0.06	-0.03	-0.05	-0.04	0.15	0.24	1.00	0.02	0.03	-0.04	-0.12	-0.05	-0.07	-0.08	-0.05	-0.02	0.15	0.15
HvyAlcoholConsump	-0.09	-0.03	-0.03	-0.03	-0.06	0.08	-0.02	-0.04	0.02	-0.03	0.02	1.00	-0.01	0.01	-0.06	0.02	-0.04	-0.05	0.01	-0.06	0.04	0.06
AnyHealthcare	0.02	0.04	0.03	0.11	-0.01	-0.01	0.01	0.02	0.03	0.03	0.03	-0.01	1.00	-0.22	-0.03	-0.05	-0.00	0.01	-0.01	0.14	0.11	0.13
NoDocbcCost	0.04	0.03	0.03	-0.06	0.07	0.04	0.04	0.04	-0.06	-0.05	-0.04	0.01	-0.22	1.00	0.17	0.19	0.16	0.13	-0.05	-0.13	-0.10	-0.20
GenHlth	0.41	0.32	0.24	0.06	0.27	0.15	0.19	0.28	-0.27	-0.10	-0.12	-0.06	-0.03	0.17	1.00	0.32	0.55	0.48	-0.01	0.16	-0.29	-0.38
MentHlth	0.09	0.06	0.08	-0.01	0.10	0.09	0.09	0.08	-0.13	-0.06	-0.05	0.02	-0.05	0.19	0.32	1.00	0.38	0.25	-0.09	-0.10	-0.11	-0.22
PhysHlth	0.21	0.17	0.14	0.03	0.16	0.12	0.16	0.20	-0.23	-0.05	-0.07	-0.04	-0.00	0.16	0.55	0.38	1.00	0.49	-0.05	0.08	-0.16	-0.28
DiffWalk	0.27	0.23	0.16	0.04	0.25	0.12	0.19	0.23	-0.28	-0.05	-0.08	-0.05	0.01	0.13	0.48	0.25	0.49	1.00	-0.08	0.20	-0.20	-0.34
Sex	0.04	0.04	0.02	-0.01	0.00	0.11	0.00	0.10	0.05	-0.09	-0.05	0.01	-0.01	-0.05	-0.01	-0.09	-0.05	-0.08	1.00	-0.00	0.04	0.16
Age	0.28	0.34	0.24	0.10	-0.04	0.11	0.12	0.22	-0.10	0.06	-0.02	-0.06	0.14	-0.13	0.16	-0.10	0.08	0.20	-0.00	1.00	-0.11	-0.13
Education	-0.17	-0.14	-0.08	-0.01	-0.10	-0.14	-0.07	-0.10	0.19	0.10	0.15	0.04	0.11	-0.10	-0.29	-0.11	-0.16	-0.20	0.04	-0.11	1.00	0.46
Income	-0.22	-0.19	-0.11	0.01	-0.12	-0.10	-0.14	-0.15	0.20	0.08	0.15	0.06	0.13	-0.20	-0.38	-0.22	-0.28	-0.34	0.16	-0.13	0.46	1.00