Diabetes Prediction Using Machine Learning

Final Report

Table Of Contents

Introduction

In the modern world, diabetes is a common term and a major problem in both developed and developing nations . Glucose can go from food into the bloodstream thanks to the pancreas' secretion of the hormone insulin in the body. Diabetes develops when the pancreas isn't producing enough of the insulin, which can lead to coma, renal and retinal failure, pathological death of pancreatic beta cells, cardiovascular dysfunction, cerebral vascular dysfunction, peripheral vascular illnesses, sexual dysfunction, joint failure, weight loss, ulcers, and pathogenic effects on immunity . The World Health Organization (WHO) estimates that 422 million people worldwide had diabetes in 2014. Furthermore, diabetes was directly responsible for 1.6 million deaths worldwide in 2016 . As per research on diabetes patients, the prevalence of diabetes among adults (over 18 years old) increased from 4.7% to 8.5% between 1980 and 2014, respectively, and is rising quickly in second- and third-world nations . Worldwide, 451 million individuals have diabetes, and that number is expected to rise to 693 million by 2045, according to statistics from 2017 . Another statistical study demonstrates the severity of diabetes. They found that half a billion people globally have diabetes, and that percentage will rise to 25% and 51%, respectively, in 2030 and 2045. Although diabetes cannot be permanently cured, it can be controlled and prevented if an accurate early diagnosis is possible.

Causes of diabetes

The primary cause of diabetes is genetics. It is carried on by at least two defective genes on chromosome 6, a chromosome that influences how the body reacts to different antigens. The development of type 1 and type 2 diabetes might also be influenced by viral infection. As per studies, having viruses such hepatitis B, CMV, mumps, rubella, and coxsackievirus increases the risk of acquiring diabetes.

Types of diabetes

Type1-When a person has type 1 diabetes, their immune system is weakened and their cells are unable to make enough insulin. There have been no convincing studies that demonstrate the causes of type 1 diabetes, and there are also no viable preventative measures at this time.

Type2-Diabetes type 2 is defined as either a low level of insulin production by the cells or improper insulin utilization by the body. Since it is the most prevalent kind of diabetes, 90% of those who are diagnosed with it also have it. Both genetic and lifestyle factors contribute to its occurrence.

Gestational diabetes:-Pregnant women with abrupt spikes in blood sugar are diagnosed with gestational diabetes. It will recur in two-thirds of the instances during consecutive pregnancies. Following a pregnancy in which gestational diabetes was present, there is a high likelihood that type 1 or type 2 diabetes may develop.

Both type 1 and type 2 diabetes are the two primary subtypes. In type 1 diabetes, also referred to as insulin-dependent or childhood-onset, the body does not produce enough insulin, necessitating the daily prescription of insulin, but in type 2 diabetes, commonly referred to as non-insulin-dependent or adult-onset, the body is unable to use insulin properly. Controlling diabetes to enhance quality of life has drawn attention to the need for more assistance and information for type 2 diabetes patients. The difficulties of managing diabetes on one's own are overwhelming for most people, despite the fact that new medicines and technological advancements have helped many people control their disease. Diabetes is a chronic disorder that requires patient self-management in order to be controlled. Monitoring blood sugar levels, taking medicine, following a balanced diet, and exercising frequently are self-management practices that are necessary.

Machine learning and data mining techniques can be used on datasets related to diabetes to lower the likelihood of experiencing some major consequences due to diabetes. Machine learning is a subfield of artificial intelligence and computer science that focuses on using data and algorithms to mimic human learning. The two primary categories of machine learning are supervised and unsupervised learning. In all situations, the key objective is to use a given dataset to deepen our comprehension of the data and uncover insightful insights. Supervised machine learning can be used for classification or regression tasks and is defined by the use of labeled data to train its algorithms.The objective of classification is to categorize every unknown event into one of several classes or categories in order to make predictions or diagnoses.

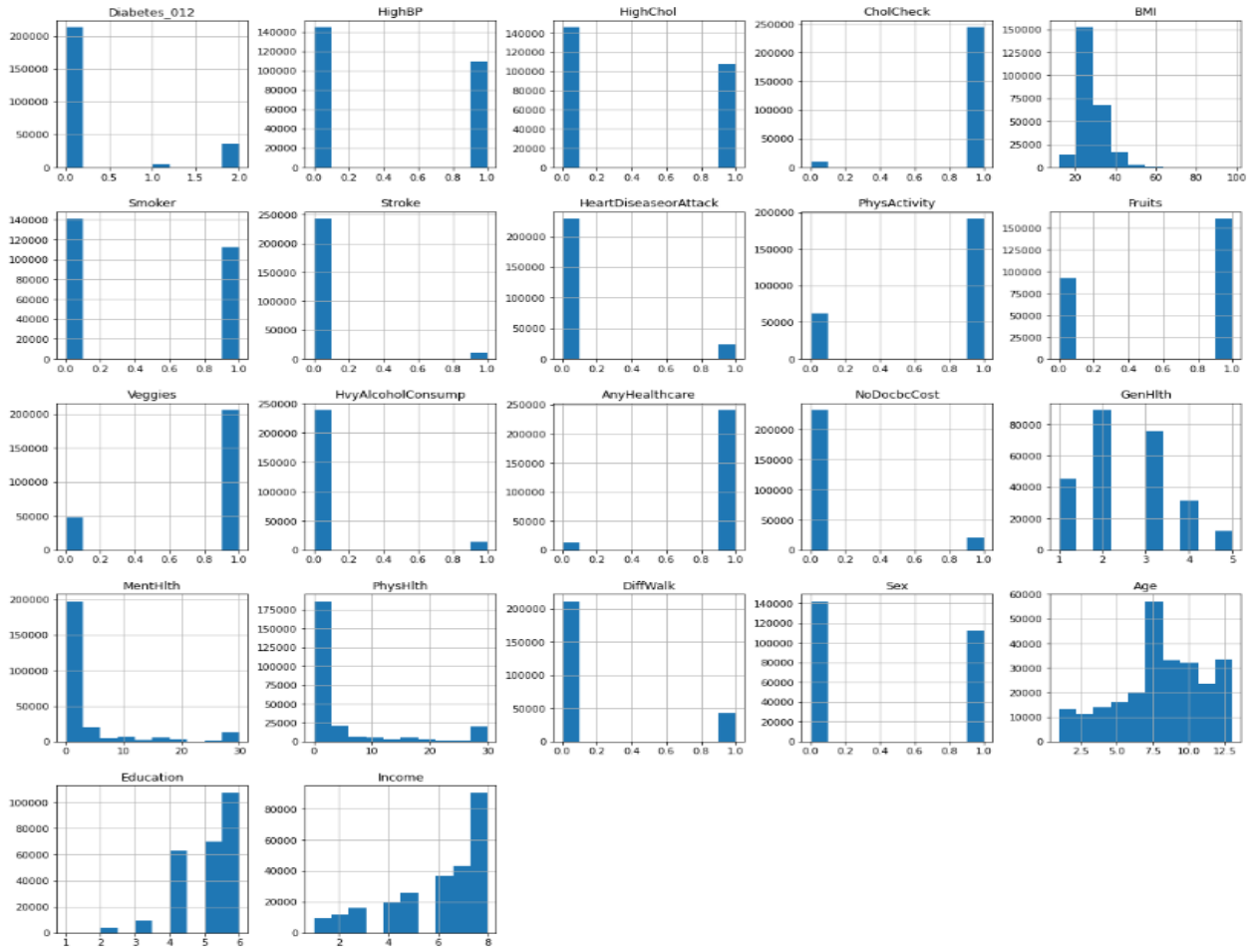
The proposed thesis implants various supervised learning algorithms like RandomForestClassifier() ,CatBoost(),GaussianNB(),Logistic regression , KNN etc. to predict the presence of diabetes.

Problem statement

Diabetes can cause a lot of issues if it is not properly managed and diagnosed by a doctor. In the past, patients would visit a diagnostic facility, speak with their doctor, and then unwind for a day or more while awaiting the results. Additionally, they must squander their money whenever they need to get their diagnosis report in vain. By employing this technique, users can determine their level of diabetes, such as normal, type 1 or type 2, and its treatments, such as dietary restrictions and yogic exercises.

For treatment, doctors rely on common knowledge. Studies are evaluated after a certain number of instances have been researched when there is a lack of common knowledge. However, this procedure takes time, whereas machine learning enables earlier pattern recognition. A significant amount of data is needed to use machine learning. Depending on the disease, there may not be a lot of information available. In addition, there are far more samples with no disorders than there are ones with actual diseases.With the aid of recent and historical data, machine learning algorithms and stats are applied to forecast the disease. Doctors can predict diabetes in the initial stages with the help of Machine learning. Medical records from diabetic patients as well as other algorithms are added to the dataset for experimental study. In order to determine if a patient is diabetic using diagnostic measures, we employ logistic regression, random

forest, decision tree classifiers, and gradient boosting. Performance and accuracy of the used algorithms are examined and contrasted. Since the distribution of classes for all attributes is not linearly separable as shown in image below, predicting diabetes is a difficult task.




```
: #EDA
```

Aims and Objectives

The aim of this project is to analyze and visualize the Diabetes Dataset and use supervised machine learning algorithms like, Logistic Regression,SupportVectorMachine,NaïveBayes,K-NearestNeighbours algorithms,RandomForestClassifier(),XG-boost,Cat-boost classifier for prediction and to develop a prediction engine.

Hypothesis

The goal of this study is to suggest a new classification model for diabetes. To attain the best classification accuracy, a variety of methods and approaches were used, including conventional machine-learning algorithms, ensemble learning approaches, and dataset Balancing. To learn from the machine learning models created in the past and try to increase the overall accuracy .

Motivation

Health problems are becoming more and more prevalent today. Many people begin getting comprehensive medical exams in their late 40s or early 50s. However, our way of life has a significant negative impact on our health, leading to diabetes and many other health issues. Number of deaths can be reduced by early diabetes identification. By employing machine learning techniques, we can detect sickness at an early level and help to totally cure it at no cost, as most of our health care companies aim to diagnose these diseases in their early phases. In order to diagnose a normal individual and produce a report that indicates whether or not the person has diabetes, as well as classify the person's level of diabetes, a trained ML model is developed using an appropriate dataset.

Literature Review

Data mining can be used in a variety of disciplines, including business, healthcare, and education. Applications of data mining in healthcare enable disease diagnosis, prognosis, and a thorough knowledge of medical data [7]. For instance, it might help us comprehend the relationships between several chronic conditions [6], such as diabetes mellitus (DM), a major health issue and a leading cause of death.

The diagnosis and prognosis of DM have drawn a lot of attention. Using the PIMA Indians Diabetes dataset, Hasan et al. [9] suggested a novel method for diabetes prediction. The dataset consists of 768 female patients, including 268 diabetes patients (positive) and 500 non-diabetic individuals (negative), with eight different attributes: pregnancy, glucose, hypertension, triceps, insulin, BMI, pedigree, and age. Preprocessing, which includes outlier rejection, substitution with the mean for missing values, data standardization, feature selection, and k-fold cross-validation, is the key to obtaining state-of-the-art findings, as the article indicates .In this study, decision trees, k-NN, AdaBoost, random forest, naive Bayes, and XGBoost were all employed and put to the test. The authors also employed an ensemble technique, which intended to improve performance by combining several classifiers. When using ensemble methods, the prediction's accuracy can be increased by combining the results of various models. AdaBoost and XGBoost were the ideal models to combine. The performance metric selected was the area under the curve (AUC). The paper beat previous trials with an AUC score of 0.95.

Sisodia et al[10] .'s goal was to predict the likelihood of diabetes in patients with the highest degree of accuracy. The PIDD dataset used in this study is the same as the one used in the prior study [9]. In order

to identify diabetes at an early stage, this experiment utilized a decision tree, SVM, and naive Bayes. The F-score was employed together with accuracy, precision, and recall.to evaluate the performance of the ideal model. According to the article, naive Bayes had the best performance outcomes, with a maximum accuracy of 76.3 percent.

K.VijiyaKumar et al. [1] suggested a random Forest algorithm for the prediction of diabetes to create a system that can accurately forecast the onset of diabetes in a patient early on utilizing the machine learning method. The suggested model provides the greatest results for diabetic prediction, and the findings demonstrated that the prediction system is capable of accurately, quickly, and most significantly, instantly forecasting the disease.

Three data mining models for predicting diabetes or prediabetes were compared in terms of performance in [11]. Logistic regression (LR), artificial neural networks (ANNs), and decision trees were the data mining models (DT). The 735 patients and 752 healthy controls that form the balanced dataset are used. Gender, age, marital status, education level, family history of diabetes, BMI, coffee consumption, physical activity, sleep length, work stress, consumption of fish, and predilection for salty foods were the 12 traits that went into creating the models. A questionnaire was used to acquire all of the prior characteristics. The C5.0 decision tree, according to the authors, has the best classification accuracy.

By utilizing the PIDD dataset, Abdulhadi et al. [12] developed a number of machine learning models to forecast the occurrence of diabetes in females. The mean substitution method was used by the authors to address the issue of missing values, and a standardization method has been used to rescale all of the attributes. The models

were built with LR, LDA, SVM (linear and polynomial), and RF. The paper claims that the RF model has an accuracy score of 82 percent at its highest.

Byoung Geol Choi, Seung-Woon Rha, Suhng Wook Kim, Jun Hyuk Kang, Ji Young Park, and Yung-Kyun Noh[26] effectively designed and verified a T2DM prediction system utilizing machine learning and an EMR database, and it predicted the 5-year occurrence of T2DM comparable to with a traditional prediction model. The usage of Least Square Support Vector Machine (LS-SVM) and Generalized Discriminant Analysis (GDA) to diagnose diabetes disease is discussed in Kemal Polat's work[13]. New cascade learning system based on generalized discriminant analysis and least square support vector machine was also proposed. Two steps are present in the suggested system. They employed generalized discriminant analysis in the first stage as a pre-processing step to separate feature variables between healthy and sick (diabetes) data.In the following stage, they classified the diabetes dataset using LS-SVM. The proposed system termed GDA-LS-SVM obtained 82.05 percent classification accuracy using 10-fold cross validation, which is extremely promising when compared to the previously published classification techniques, whereas LS-SVM obtained 78.21 percent classification accuracy using the same method.

In this study, the dimensionality was reduced using principal component analysis (PCA) and minimum redundancy maximum relevance (mRMR) by Quan Zou, Kaiyang Qu, Yamei Luo, Dehui Yin, Ying Ju, and Hua Tang[14]. According to the findings, including all the variables allowed random forest prediction to have the maximum accuracy (ACCURACY = 0.8084). With the help of a neural network algorithm, Thangara[15] predicts the development of diabetes from a clinical database. With the aid of MDVP, Chitkara[16] conducted

studies on how to identify the vocal characteristics of people with type 2 diabetes. This work [17] discusses machine learning on administrative data, which offers a potent new tool for population health and clinical hypothesis creation for risk factor discovery, enabling population-level risk assessment that may assist steer interventions to the most at-risk population.It is possible to identify patients who are likely to develop type 2 diabetes using the method presented here, with a Prediction Of Type 2 Diabetes From Claims Data 285 at least 67 percent better PPV than with conventional risk assessment approaches for the next 0–2 years. The wide collection of risk indicators that our method collected, for various disease beginning stages, can serve as a basis for further hypothesis testing in medical research facilities. Last but not least, our method is broad enough to be used for a range of outcomes of interest, to create forecasts for a range of years into the future, and to examine risk factors as they manifest themselves at various stages prior to the commencement.

By exploring and examining the patterns that emerge in the data through classification analysis utilizing Decision Tree and Naive Bayes algorithms, Aiswarya et al[2] .'s goal is to find ways to diagnose diabetes. In order to help patients receive timely treatment, the research aims to provide a quicker and more effective approach of diagnosing the illness. The study found that the J48 method provides an overall accuracy of 74.8 percent whereas the naïve Bayes classifier provides an accuracy of 79.5 percent by adopting a 70:30 split, using the PIMA dataset and cross validation approach.

Lee et al. focus on applying a decision tree algorithm named as CART on the diabetes dataset after applying the resample filter over the data.The author places a strong emphasis on the issue of class imbalance and the need to address it before implementing any method in order to increase accuracy rates. Class imbalance is most common in datasets with dichotomous values, which implies that the

class variable has two alternative outcomes and may be readily managed if discovered earlier in the data preprocessing stage. This will help to increase the predictive model's accuracy.

For the purpose of diagnosing diabetes mellitus in this paper[53], we employed the K-nearest neighbor technique. For K=3, 5, we have determined accuracy and error rates. The outcome demonstrates that accuracy rate and error rate will both rise as the value of k rises. One of the most useful artificial intelligence algorithms for diagnosing problems is KNN. KNN can be used to provide outcomes that are more precise and effective. Different categorization strategies were used in the data mining procedure in this work [18]. These techniques have been used to separate the data into various sets, making it simple to identify relationships between various attributes. The diagnosis of diabetic disease has been aided by the use of various data mining tools.Additionally, data mining methods including support vector machines, bagging algorithms, automatically generated groups, and kernel densities are employed. In this paper[19], we employ three distinct classification algorithms—ZeroR, OneR, and Naive Bayes—that were utilized in this experiment. This experiment demonstrates that Naive Bayes is the fastest and ZeroR is the slowest. Weka's data mining technology is used to find the performance comparison. Each model is transformed into a set of rules, which are then implemented into this programme. It is discovered that Naive Bayes performs better than OneR and ZeroR. With the aid of cutting-edge computational techniques and the availability of a sizable amount of epidemiological and genetic diabetes risk dataset, machine learning has the potential to completely transform the estimation of the risk of developing diabetes[20].Early diabetes detection is essential for effective treatment. In this study, a machine learning method for forecasting diabetes levels was described. The method may also

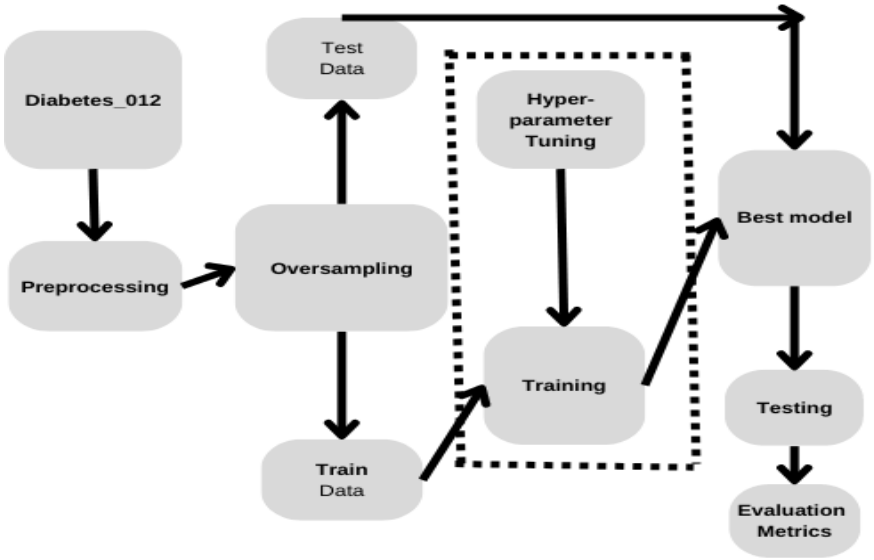
assist researchers in creating a precise and useful tool that will sit at the clinicians' table and assist them in making better decisions regarding the status of the disease.

In his study, Yunsheng[9] employed a novel method called DISKR (reduce the size of the training set for K-nearest neighbors) to remove outliers/OOBs (out of bag) from the KNN algorithm. Additionally, the study minimized storage space. As a result, after eliminating factors or instances that have little bearing on the research, the space complexity is made less complex and more effective. For the prediction and treatment of diabetic disease, V. Kumar and L. Velide[21] used data mining techniques. They employed Naive Bayes, JRip, J48 (4.5), DT, and NN methods. For implementation, they made use of the WEKA tool. For the J48 algorithm, they obtained an accuracy level of 68.5 percent.The study [22] provides a thorough analysis of the data mining techniques currently in use for diabetes prediction. Additionally, information is provided regarding Type 1, Type 2, and Type 3 diabetes. The goal of diabetes is to forecast diabetes with the aid of data mining techniques such the K-Nearest Neighbor Algorithm, Bayesian Classifier, Naive Bayesian Classifier, and Bayesian Network. All of these techniques are utilized for diabetes prediction. Diabetes' impacts on patients are also discussed in this paper. In this paper[23], the suggested methodology aims at offering an effective hybrid classification framework for predicting and monitoring the Diabetes condition.The fundamental goal of this research is to find and create models that will effectively aid medical professionals, enabling individuals to live longer lives on our planet.

Proposed solution

Proposed Framework

The preparation of raw data is a crucial stage in the pipeline suggested in this thesis's proposed methodology, since the quality of the data may encourage classifiers to learn by doing.Then data is feeded to the machine learning algorithms to train on it. This is depicted in the figure below.



Data-set

A clean dataset of 253,680 survey responses to the CDC's (BRFSS2015) available as diabetes_012_health_indicators_BRFSS2015.CSV . Three classes make up the target variable Diabetes_012. The three classes are no

diabetes, pre-diabetes and diabetes. In this dataset, there is a disparity in class. There are 22 feature variables in this dataset . We can get an overview of the dataset by using the pandas profiling library as shown below.

```
In [15]: profile = ProfileReport(df)
         profile
```

Summarize dataset: 100% 71/71 [01:29<00:00, 3.38it/s, Completed]

Generate report structure: 100% 1/1 [00:06<00:00, 6.26s/it]

Render HTML: 100% 1/1 [00:03<00:00, 3.06s/it]

Dataset statistics

Number of variables	22
Number of observations	253680
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	11187
Duplicate rows (%)	4.4%
Total size in memory	42.6 MiB
Average record size in memory	176.0 B

Variable types

Categorical	16
Numeric	6

We can get a very basic understanding by looking at the dataset that has 22 variables and there are 253680 observations present in the dataset .The dataset have many independent variables but have one dependent variable diabetes_012 which have 3 different levels , each one indicating if the person is non-diabetic(0),pre-diabetic(1) or diabetic(2).

Now we can further explore the dataset to get more information out of it by using the describe() function to get an idea of standard deviation, mean and count of the attributes of the classes.

```
In [6]: df.describe()
```

Out[6]:

	Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity
count	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000
mean	0.296921	0.429001	0.424121	0.962670	28.382364	0.443169	0.040571	0.094186	0.756544
std	0.698160	0.494934	0.494210	0.189571	6.608694	0.496761	0.197294	0.292087	0.429169
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	1.000000	27.000000	0.000000	0.000000	0.000000	1.000000
75%	0.000000	1.000000	1.000000	1.000000	31.000000	1.000000	0.000000	0.000000	1.000000
max	2.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000

```
In [6]: df.describe()
```

Out[6]:

Fruits	Veggies	HvyAlcoholConsump	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex
253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000
0.634256	0.811420	0.056197	0.951053	0.084177	2.511392	3.184772	4.242081	0.168224	0.440342
0.481639	0.391175	0.230302	0.215759	0.277654	1.068477	7.412847	8.717951	0.374066	0.496429
0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
0.000000	1.000000	0.000000	1.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
1.000000	1.000000	0.000000	1.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
1.000000	1.000000	0.000000	1.000000	0.000000	3.000000	2.000000	3.000000	0.000000	1.000000
1.000000	1.000000	1.000000	1.000000	1.000000	5.000000	30.000000	30.000000	1.000000	1.000000

Correlation

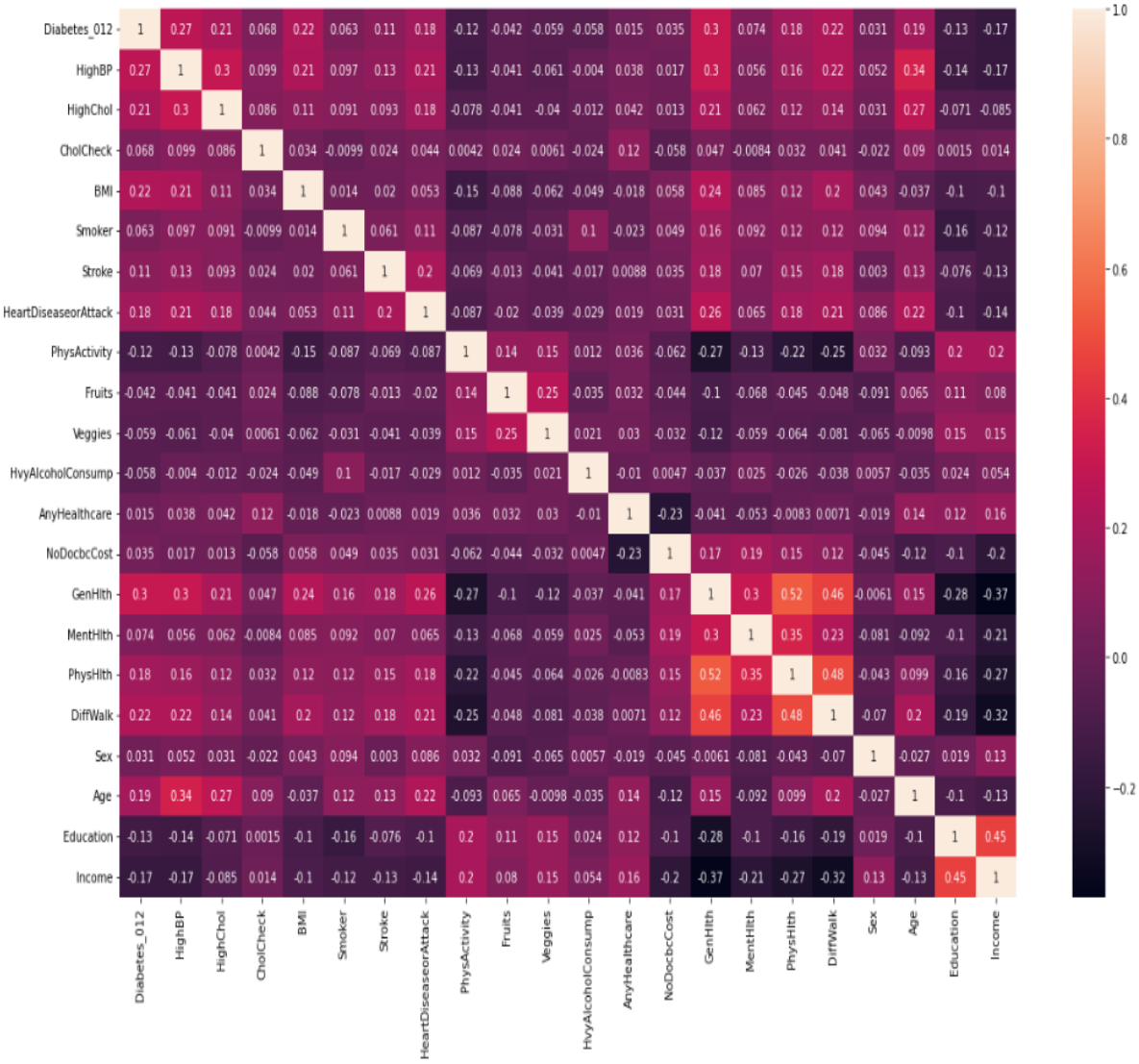
Understanding the connections between various variables and properties in your dataset is possible through data correlation. You can learn some things by using correlation, such as:

- One or more traits are dependent upon or act as a driver for other attributes.
- With other attributes, one or more traits may be connected.

Correlation can help us to predict one attribute with the help of another and it can also help in determining the presence of a casual relationship. There are several problems that arise due to high correlation because the independent variables have a propensity to change simultaneously, the model finds it challenging to estimate the link between each independent variable and the dependent variable separately.

Now we will find the correlation among the features of the dataset with the help of a heatmap of correlation using the seaborn library, as shown in the figure below.

```
In [8]: #Heatmap of correlation
        plt.figure(figsize = (20,12))
        sns.heatmap(df.corr(),annot=True)

Out[8]: <AxesSubplot:>
```

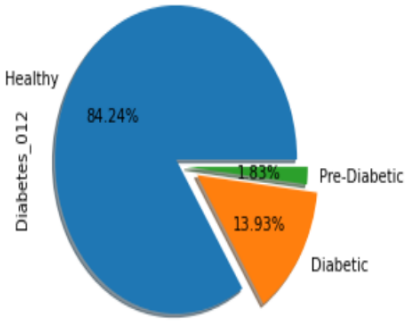


We can infer from the dataset that no single feature has very correlation with our target variable diabetes_012.Some of the qualities have an inverse relationship with the target variable while others have a positive relationship.

We will plot a pie chart to get an idea of how much of the surveyed people are healthy, pre-diabetic and diabetic.to get a broader perspective of our data. As shown in the figure below , 84.24% people are healthy , 13.93% people are diabetic and 1.83% are pre-diabetic.

```
In [12]: #pie chart
         labels = 'Healthy','Diabetic', 'Pre-Diabetic'
         ex=[0.1,0.1,0]
         df.Diabetes_012.value_counts().plot.pie(labels=labels, autopct='%1.2f%%',shadow=True,explode=ex)
```

```
Out[12]: <AxesSubplot:ylabel='Diabetes_012'>
```



Pre-Processing

Preprocessing in the proposed framework entails outlier rejection (P), filling in missing values (Q), standardization (R), and feature selection of the attribute, which are simply explained as follows:

Data Cleaning

The dataset must be cleaned by deleting any extraneous records and characteristics using a methodical methodology.There were no features present in the dataset with confidential values or unnecessary values. Furthermore there were no missing or null values in the dataset , so for now no data changes are made to the dataset, as shown in the figure below.

```
In [13]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   Diabetes_012          253680 non-null  float64
 1   HighBP                253680 non-null  float64
 2   HighChol              253680 non-null  float64
 3   CholCheck             253680 non-null  float64
 4   BMI                   253680 non-null  float64
 5   Smoker                253680 non-null  float64
 6   Stroke                253680 non-null  float64
 7   HeartDiseaseorAttack  253680 non-null  float64
 8   PhysActivity          253680 non-null  float64
 9   Fruits                253680 non-null  float64
 10  Veggies               253680 non-null  float64
 11  HvyAlcoholConsump     253680 non-null  float64
 12  AnyHealthcare         253680 non-null  float64
 13  NoDocbcCost           253680 non-null  float64
 14  GenHlth               253680 non-null  float64
 15  MentHlth              253680 non-null  float64
 16  PhysHlth              253680 non-null  float64
 17  DiffWalk              253680 non-null  float64
 18  Sex                   253680 non-null  float64
 19  Age                   253680 non-null  float64
 20  Education             253680 non-null  float64
 21  Income                253680 non-null  float64
dtypes: float64(22)
memory usage: 42.6 MB
```

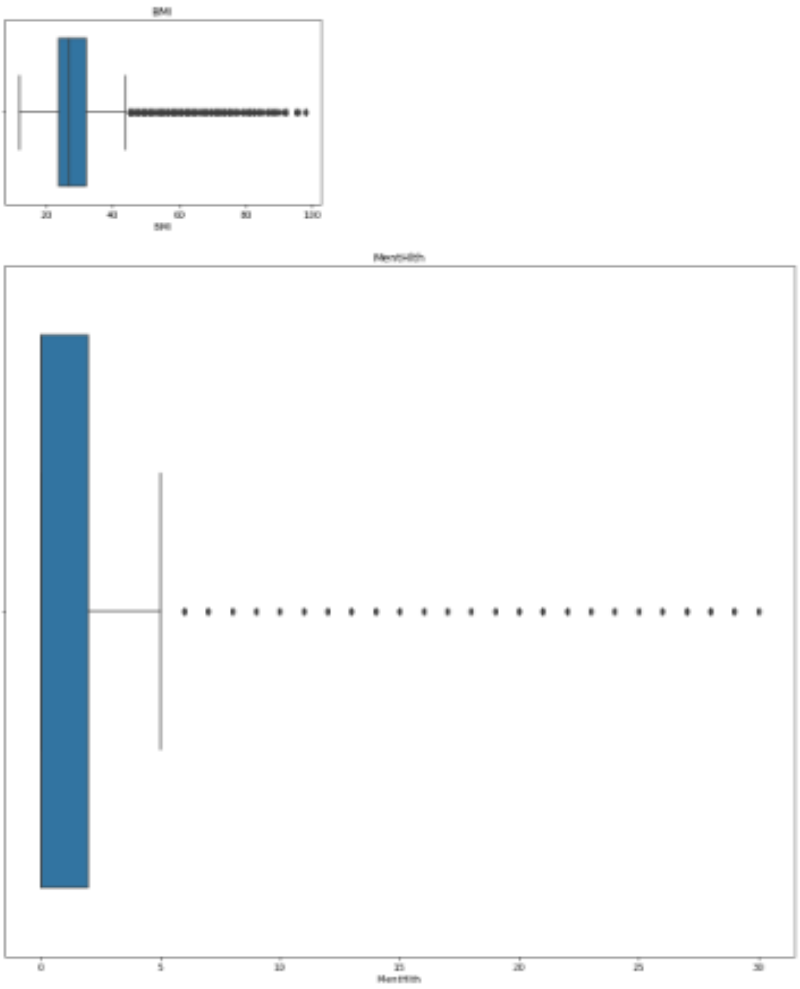
Data Modification(Outliers Rejection)

As the classifiers are particularly sensitive to the data range and distribution of the features, it must be excluded from the data distribution. In this literature, the mathematical formulation for the outlier rejection can be expressed as,

$$P(x) = \begin{cases} x, & \text{if } Q_1 - 1.5 \times IQR \leq x \leq \\ reject, & \text{otherwise} \end{cases}$$

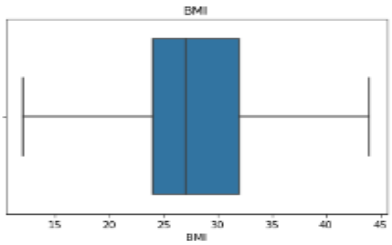
where x represents specific instances of the feature vector, xϵR^n, that are located in n-dimensional space. The first quartile, third quartile, and interquartile range of the qualities, expressed as Q1,Q3, and IQR, respectively, where Q1,Q3, and IQRϵR^n.We will evaluate the outliers in the continuous features as shown below.

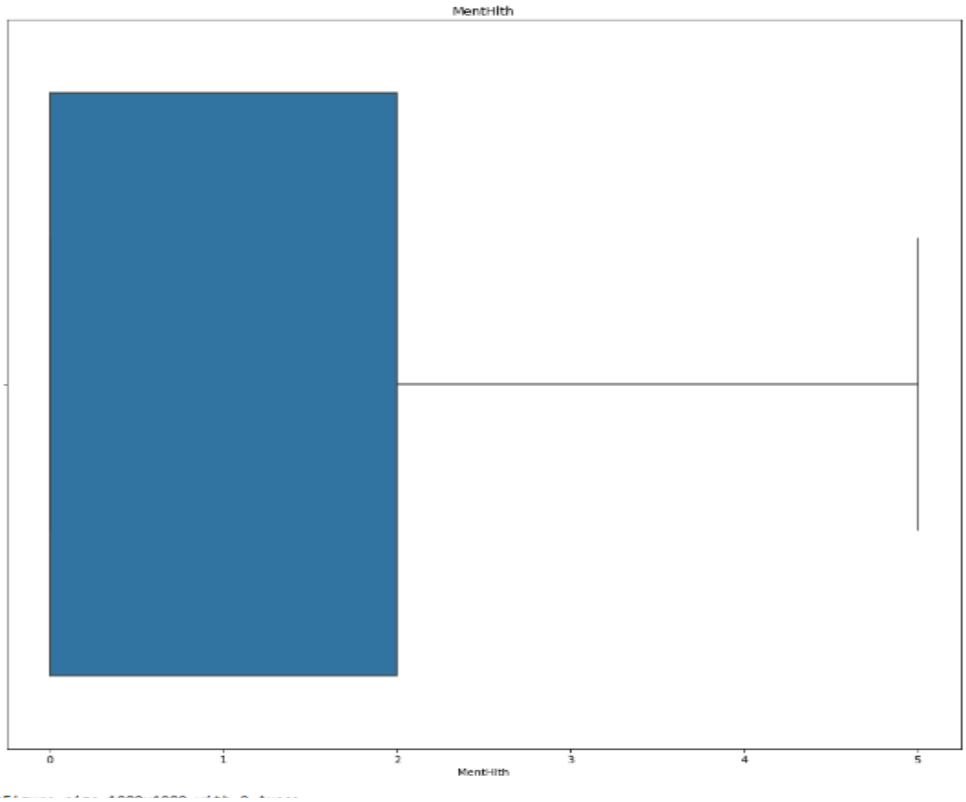
In [27]:
```
#A for loop is used to plot a boxplot for all the continuous features to see the outliers
for feature in continuous_feature:
    data=df.copy()
    sns.boxplot(data[feature])
    plt.title(feature)
    plt.figure(figsize=(15,15))
```



The outliers were treated using the Outlier rejection formula as expressed above and the Boxplot was plotted to observe the changes in the dataset as shown in figure below,

```
In [29]: IQR=df.BMI.quantile(0.75)-df.BMI.quantile(0.25)
         lower_bridge=df.BMI.quantile(0.25)-(IQR*1.5)
         upper_bridge=df.BMI.quantile(0.75)+(IQR*1.5)
         print(lower_bridge, upper_bridge)

         12.0 44.0

In [30]: df.loc[df['BMI']>=44.0,'BMI']=44.0
         df.loc[df['BMI']<=12.0,'BMI']=12.0

In [31]: IQR=df.MentHlth.quantile(0.75)-df.MentHlth.quantile(0.25)
         lower_bridge=df.MentHlth.quantile(0.25)-(IQR*1.5)
         upper_bridge=df.MentHlth.quantile(0.75)+(IQR*1.5)
         print(lower_bridge, upper_bridge)

         -3.0 5.0

In [32]: df.loc[df['MentHlth']>=5.0,'MentHlth']=5.0
         df.loc[df['MentHlth']<=-3.0,'MentHlth']=-3.0

In [33]: for feature in continuous_feature:
             data=df.copy()
             sns.boxplot(data[feature])
             plt.title(feature)
             plt.figure(figsize=(15,15))
```



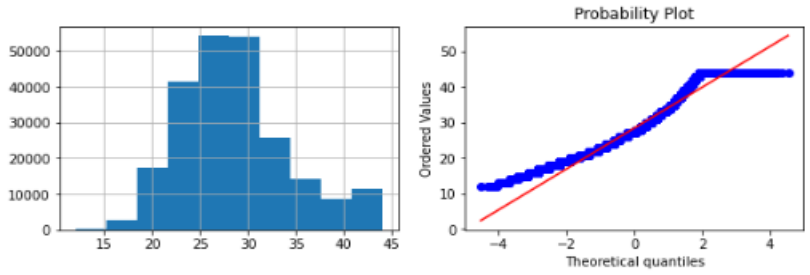


After outlier rejection From the continuous features, we plotted qq plots to observe the reduced skewness in the data features.

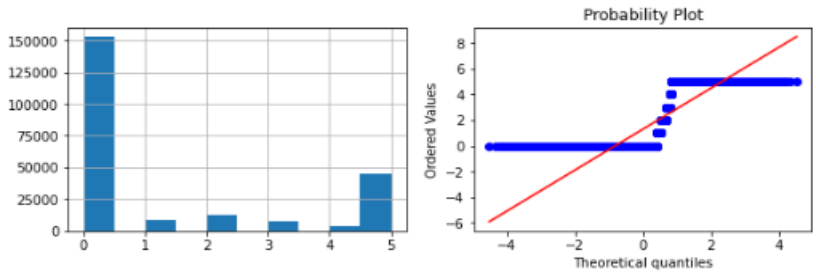
```
In [84]: def qq_plots(df, variable):
             plt.figure(figsize=(10,3))
             plt.subplot(1, 2, 1)
             df[variable].hist()
             plt.subplot(1, 2, 2)
             stats.probplot(df[variable], dist="norm", plot=plt)
             plt.show()
```

```
In [85]: for feature in continuous_feature:
             print(feature)
             plt.figure(figsize=(10,3))
             plt.subplot(1, 2, 1)
             df[feature].hist()
             plt.subplot(1, 2, 2)
             stats.probplot(df[feature], dist="norm", plot=plt)
             plt.show()
```

```
BMI
```




```
MentHlth
```



We can see from the dataset that with the help of outlier rejection we have reduced the skewness of the data in case of the feature BMI .

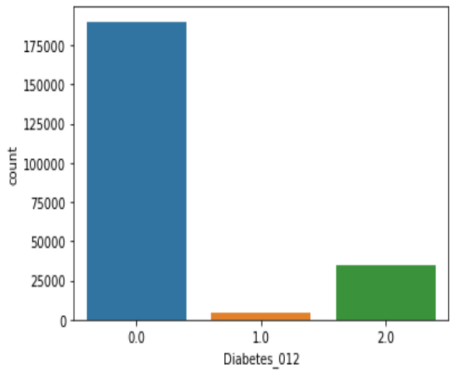
As the attribute's dimension is increased, the classifiers' accuracy rises. When the attribute's dimension rises without increasing the sample size, the classifiers' performance will suffer. In machine learning, such a situation is known as a "curse of dimensionality." Due to the "curse of dimensionality," the feature space gets sparser and sparser, which drives classifiers to overfit by losing their generalizing power.In this thesis, the most popular feature selection techniques,

Principle Component Analysis , were used for denoising and data compression using the Diabetes_012 dataset.

Data Balancing (Oversampling)

Oversampling can be a great method to treat imbalanced classified datasets. As we have very high dimensionality in our dataset we have to increase samples to avoid “curse of dimensionality” and to prepare the suitable dataset for our model in order to attain the highest accuracy.The target variable count of labels healthy , diabetic and pre-diabetic people were evaluated as shown in the figure below.

```
In [18]: sns.countplot(df['Diabetes_012'])

G:\anaconda_software\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(

Out[18]: <AxesSubplot:xlabel='Diabetes_012', ylabel='count'>
```



These labels were oversampled to decrease the “dimensionality of the dataset” and to prepare a suitable dataset.

```
In [20]: non_diabetes= df[df['Diabetes_012'] == 0]
         pre_diabetes = df[df['Diabetes_012'] == 1]
         diabetes = df[df['Diabetes_012'] == 2]

         pre_diabetes_os = pre_diabetes.sample(len(non_diabetes), replace=True)
         diabetes_os = diabetes.sample(len(non_diabetes), replace=True)

         df_new = pd.concat([pre_diabetes_os,diabetes_os, non_diabetes], axis=0)
```

```
In [21]: df_new['Diabetes_012'].value_counts()
```

```
Out[21]: 1.0    190055
         2.0    190055
         0.0    190055
         Name: Diabetes_012, dtype: int64
```

```
In [22]: df_new.shape
```

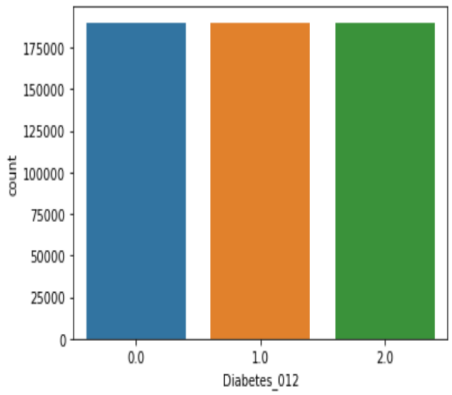
```
Out[22]: (570165, 21)
```

After oversampling the dataset we will plot a countplot to observe the changes.

```
In [23]: sns.countplot(df_new['Diabetes_012'])
```

```
G:\anaconda_software\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[23]: <AxesSubplot:xlabel='Diabetes_012', ylabel='count'>
```



After over sampling the dataset we used principal component analysis to further denoise and compress the dataset diabetes_012.Principal component analysis, or PCA, is a statistical technique that enables you to condense the amount of data included within the massive data

tables into a more manageable number of "summary indices" for easier visualization and analysis.PCA serves as the foundation for multivariate data analysis using projection techniques. In order to identify trends, jumps, clusters, and outliers, it is crucial to represent a multidimensional data table as a smaller number of variables. Before training the model we used Principal component analysis (PCA) as shown below.

```
In [55]: from sklearn.decomposition import PCA

         pca = PCA(n_components = 20)

         x_train = pca.fit_transform(x_train)
         x_test = pca.transform(x_test)

         explained_variance = pca.explained_variance_ratio_
```

Splitting Data

The dataset is prepared for training and testing after data cleaning and preprocessing. In the train/split approach, we divided the dataset into a training set and a testing set at random. We splitted the dataset into two parts of 80:20 golden ratio, where training data was 80% and testing data was 20%, as shown below.

```
In [43]: #Spliting the data in 80:20 training to testing
         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

Implementation of machine learning algorithms

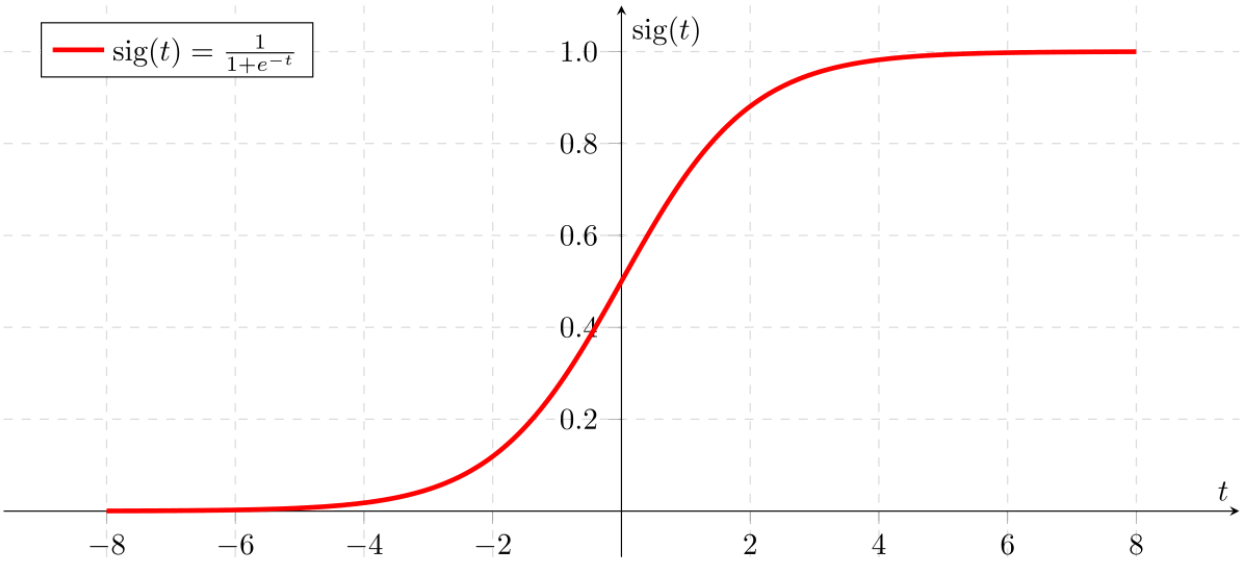
When the data is ready, machine learning techniques are used. For the purpose of predicting diabetes, we employ various classification and ensemble algorithms. The procedures used on the diabetes_012 dataset. The primary objective is to use machine learning techniques to examine the effectiveness of various methods, determine their accuracy, and identify the responsible/important characteristic that is crucial for prediction. The Principle component analysis (PCA) which is an unsupervised learning algorithm is used to reduce the dimensionality of the dataset and then the following algorithms were applied on the datasets . The methods are as followed:-

Logistic regression

A good learning-based classification approach is logistic regression. It is used to estimate the likelihood of a binary response based on one or more predictors. Both continuous and discrete ones are possible. When we wish to classify or separate some data items into categories, we employ logistic regression.

It classifies the data in binary form, which contains only the digits 0 and 1, which relate to patients who are positive or negative for diabetes.

The main goal of logistic regression is to find the optimal fit that best describes the relation between the target and predictor variables. The linear regression model is the foundation of logistic regression. The sigmoid function is used in the logistic regression model to forecast the likelihood of the positive and negative classes.



Logistic Regression uses following algorithm:-

- Data preparation phase
- Making Logistic Regression Fit to the Training Set
- estimating the test outcome
- Verify the result's accuracy
- displaying the outcome of the test set

We will train the model with the help of logistic regression as shown below.

In [40]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(x_train, y_train)
```

Out[40]:
```
LogisticRegression
LogisticRegression(random_state=0)
```

In [41]:
```
y_pred = classifier.predict(x_test)
```

In [42]:
```
#Performace Evaluation
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[24515  6694  6860]
 [11120 11419 15465]
 [ 6816  8849 22295]]
```

RandomForestClassifier

It is a particular kind of ensemble learning methodology and is employed in both classification and regression applications. When compared to other models, it provides greater accuracy. Large datasets can be handled by this strategy with ease. Leo Bremen has created Random Forest. It is a well-known ensemble learning technique. By lowering variance, Random Forest enhances the performance of Decision Tree. It works by building a large number of decision trees during the training period, and it outputs the class that represents the mean of all classes, or mean classification, or mean regression, of all individual trees.

The random forest classifier uses following algorithm:-

- The initial step is to choose the R features where, R>>M from the total m features.
- Create the decision trees linked to the chosen data points (Subsets).
- For any decision trees you intend to construct, select N.
- Replicate steps 1 and 2.
- When dealing with fresh data points, locate each decision tree's predictions and categorize them according to whatever group receives the most votes.

Using the Gin-Index Cost Function, the random forest determines the optimal split as followed :-

The first stage entails looking at options, using the bases of each randomly formed decision tree to forecast the conclusion, and storing the predicted outcome at intervals around the desired location. Calculate the votes for each predicted target, and then, as a consequence of the final prediction made using the random forest algorithm, accept the predicted target with the highest number of votes. Some of the Random Forest solutions provide accurate forecasts for a variety of applications.

Now,we will train the model with the help of Randomforestclassier as shown below.

In [47]:
```
from sklearn.ensemble import RandomForestClassifier

model_1 = RandomForestClassifier(n_estimators = 300, criterion = 'entropy',
                                 min_samples_split=10, random_state=0)

# fitting the model on the train data
model_1.fit(x_train, y_train)

# predicting values on test data
predictions = model_1.predict(x_test)
```

In [48]:
```
from sklearn import metrics
print("Random forest model accuracy(in %):", metrics.accuracy_score(y_test, predictions)*100)
```

```
Random forest model accuracy(in %): 95.01109328001543
```

In [49]:
```
#Performace Evaluation
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, predictions)

print ("Confusion Matrix : \n", cm)
```

```
Confusion Matrix :
 [[33286   316  4467]
 [    0 38004     0]
 [  793   113 37054]]
```

KNN Classifier

Another supervised machine learning algorithm is K - nn. K - nearest neighbor aids in the resolution of the classification and regression issues. KNN is a slack prediction method. K - nn assumes that related things are located close to one another. Identical data points are frequently found close together. K - nn aids in classifying fresh work using a similarity metric. The K - nn algorithm collects all the data and categorizes it based on how similar the data is. Uses a tree-like structure to determine the distance between the spots. The method locates the nearest data points in the training data set to create a prediction for a new data point. Here, K is always a positive integer and stands for "number of close neighbors." Neighbors value is picked from a list of class values. The closeness is determined by euclidean distance and it is calculated as shown below,

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

The K-Nearest Neighbor uses following algorithm:-

- choose the value K of the neighbors
- Determine the Euclidean distance between K neighbors.
- Pick the K closest neighbors based on the Euclidean distance estimation.
- Count the number of data points in each category among these k neighbors.
- Assign the additional data points to the category where the neighbor count is at its highest.

Now,we will train the model with the help of knn as shown below,

```
In [62]: knn = KNeighborsClassifier(n_neighbors=3)
         knn.fit(X_train_res, y_train_res)

Out[62]: KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=3)

In [64]: y_pred4 = knn.predict(x_test)
         print(confusion_matrix(y_test,y_pred4))
         print(accuracy_score(y_test,y_pred4))
         print(classification_report(y_test,y_pred4))

[[27599  1491  8979]
 [   19 37937    48]
 [ 1288   219 36453]]
0.8943814509834872
              precision    recall  f1-score   support

         0.0       0.95      0.72      0.82     38069
         1.0       0.96      1.00      0.98     38004
         2.0       0.80      0.96      0.87     37960

    accuracy                           0.89    114033
   macro avg       0.90      0.89      0.89    114033
weighted avg       0.90      0.89      0.89    114033
```

Catboost Classifier

A machine learning library called CatBoost was created to hasten the training of deep neural networks. As a way to determine if a piece of writing fits inside a particular topic or not, it can also be used to classify texts. This method is based on the use of decision trees and it is created to make predictions with the aid of a training data set.By modifying the weights in accordance with the data distribution and taking into account prior knowledge about the data set, the CatBoost algorithm increases accuracy. This can lessen overfitting and boost general effectiveness.

It is particularly effective in two ways:

- It produces cutting-edge findings without the substantial data training that other machine learning techniques generally require, and,
- gives strong out-of-the-box support for the more descriptive data formats that go along with many business issues.

The name "CatBoost" is a combination of the phrases "Category" and "Boosting."

Since this library is based on the gradient boosting library, the term "Boost" refers to the machine learning algorithm used to boost gradients. Gradient boosting is a potent machine learning method that is frequently used to solve various business problems like fraud detection, product suggestion, and forecasting. It also works effectively. In addition, it can produce excellent results with much less data than DL models, which must learn from enormous amounts of data.

The Catboost algorithm is better than other boosting algorithms in following ways:-

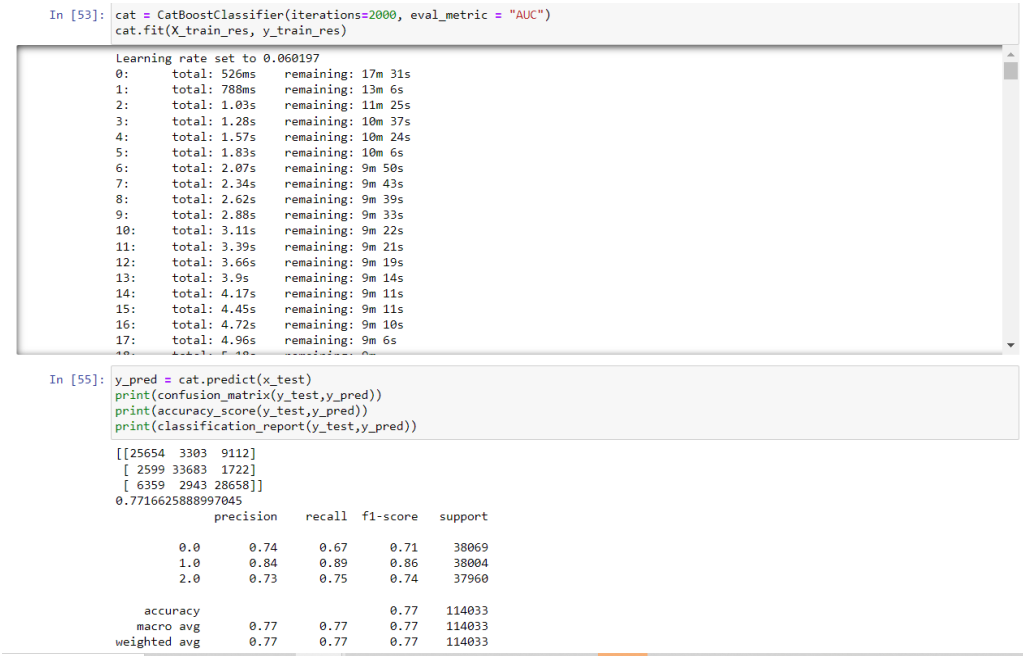
- Performance: CatBoost offers cutting-edge results and compares favorably with any top machine learning algorithm in terms of performance.
- Handling Automatic categorization: CatBoost allows us to transform categories into numbers without performing any explicit pre-processing. CatBoost transforms categorical data into numerical values by employing various statistics on categorical feature combinations and categorical and numerical feature combinations.

- Robustness: It lessens the need for significant hyper-parameter adjustment and also lowers the risk of overfitting, which results in more broadly applicable models. The number of trees, learning rate, regularization, tree depth, fold size, bagging temperature, and other parameters are among the many that may be tuned in CatBoost.
- Simple to use: We can access CatBoost from the command line using an intuitive Python and R API.

The Catboost Classifier works in following ways:

- Splits are first calculated.
- categorical features are converted into numerical features.
- converting textual information into numerical features.
- deciding on a tree structure
- figuring out values in leaves.

Now, we will train the model with the help of Catboost classifier , as shown in figure below.



GaussianNB

The linear models and the family of classifiers known as naive Bayes classifiers are pretty similar. They often move even more quickly during training. Naive Bayes models frequently offer generalization performance that is somewhat inferior to that of linear classifiers like LogisticRegression and LinearSVC as a price for this efficiency. GuassianNB() algo works on the naive bayes theorem as shown below,

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Where,

- **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.
- **P(d|h)** is the probability of data d given that the hypothesis h was true.
- **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- **P(d)** is the probability of the data (regardless of the hypothesis)

As you can see, our goal is to determine the posterior probability P(h|d) given the prior probability P(h) and the inputs P(D) and P(d|h).

You can choose the hypothesis with the highest probability after computing the posterior probabilities of several candidate hypotheses. The maximum a posteriori (MAP) hypothesis, which is the most likely explanation, can be used to refer to this.

The following can be written:

MAP(h) = max(P(h|d))

or

MAP(h) = max((P(d|h) * P(h)) / P(d))

or

MAP(h) = max(P(d|h) * P(h))

The assumption that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution is frequently made when working with continuous data. The characteristics' likelihood is predicated on-

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Sometimes assume variance,

- s independent of Y (i.e., σi),
- or independent of Xi (i.e., σk)
- or both (i.e., σ)

The Gaussian Naive Bayes model allows continuous valued features and assumes that all of them follow a Gaussian (normal) distribution.

Assuming that the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between dimensions is one method for building a straightforward model. By simply calculating the mean and standard deviation of the points within each label, this model may be fit.

Now, let's train the model with the help of GuassianNB classifier , as shown below.

```
In [57]: gnb = GaussianNB()
         gnb.fit(X_train_res, y_train_res)

Out[57]: ▾ GaussianNB
         GaussianNB()

In [59]: y_pred3 = gnb.predict(x_test)
         print(confusion_matrix(y_test,y_pred3))
         print(accuracy_score(y_test,y_pred3))
         print(classification_report(y_test,y_pred3))

         [[20629  6683 10757]
          [ 8939  8864 20201]
          [ 5360  6478 26122]]
         0.487709698069857
                       precision    recall  f1-score   support

                  0.0       0.59      0.54      0.57     38069
                  1.0       0.40      0.23      0.30     38004
                  2.0       0.46      0.69      0.55     37960

             accuracy                           0.49    114033
            macro avg       0.48      0.49      0.47    114033
         weighted avg       0.48      0.49      0.47    114033
```

XGBoost Classifier

A Gradient Boosted decision tree implementation is called XGBoost. Decision trees are generated successively in this process. Weights are significant in XGBoost. Each independent variable is given a weight before being fed into the decision tree that forecasts outcomes. Variables that the tree incorrectly predicted are given more weight before being placed into the second decision tree. These distinct classifiers/predictors are then combined to produce a robust and accurate model. It can be used to solve problems including regression, classification, ranking, and custom prediction.

Now, we will train the model with the help of XGBoost classifier.

In [65]:
```
xgb = XGBClassifier()
xgb.fit(X_train_res, y_train_res)
```

Out[65]:
```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, objective='multi:softprob',
              predictor='auto', random_state=0, reg_alpha=0, ...)
```

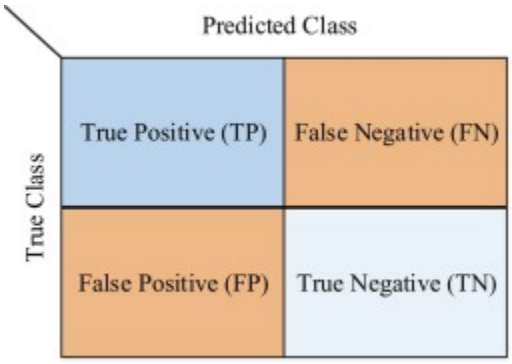
In [72]:
```
y_pred5 = xgb.predict(x_test)
print(confusion_matrix(y_test,y_pred5))
print(accuracy_score(y_test,y_pred5))
print(classification_report(y_test,y_pred5))
```

```
[[24962  4891  8216]
 [ 3891 28353  5760]
 [ 5827  4919 27214]]
0.7061903133303518
              precision    recall  f1-score   support

         0.0       0.72      0.66      0.69     38069
         1.0       0.74      0.75      0.74     38004
         2.0       0.66      0.72      0.69     37960

    accuracy                           0.71    114033
   macro avg       0.71      0.71      0.71    114033
weighted avg       0.71      0.71      0.71    114033
```

Evaluation Metrics

We have used performance matrices to discover the most accurate classifier for diabetes prediction. The confusion matrix and accuracy are given below.

Confusion Matrix:- Confusion matrices are a widely used measurement when attempting to solve classification issues. Both binary classification and multiclass classification issues can be solved with it.



Confusion matrices show counts between expected and observed values. The result "TN" stands for True Negative and displays the number of negatively classed cases that were correctly identified. Similar to this, "TP" stands for True Positive and denotes the quantity of correctly identified positive cases. The term "FP" denotes the number of real negative cases that were mistakenly categorized as positive, while "FN" denotes the number of real positive examples that were mistakenly classed as negative. Accuracy is one of the most often used metrics in classification.We have selected an accuracy matrix to evaluate the effectiveness of each model. The proportion of predictions that were right to all of the predictions made. The formula

below is used to determine a model's correctness (via a confusion matrix).

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

Precision :- The proportion of accurately categorized positive samples (True Positive) to the total number of positively classified samples is known as precision.

Recall :- The recall is determined as the proportion of Positive samples that were correctly identified as Positive to all Positive samples. The recall gauges how well the model can identify positive samples. The more positive samples that are identified, the larger the recall.

F1-score :- A model's accuracy on a dataset is indicated by the F-score, often known as the F1-score. It is employed to assess binary classification schemes that label examples as "positive" or "negative."

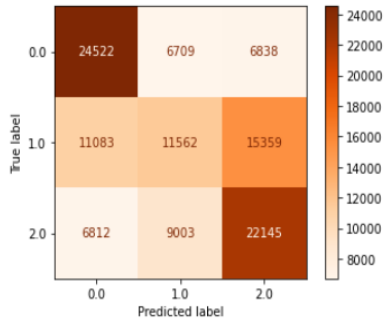
$$\begin{aligned} F_1 &= \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \\ &= \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})} \end{aligned}$$

The Confusion Matrix of each used machine learning Algorithm is given below:-

Logistic Regression

```
In [49]: from sklearn.metrics import plot_confusion_matrix
         plot_confusion_matrix(classifier,x_test,y_test,cmap='Oranges')
         plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



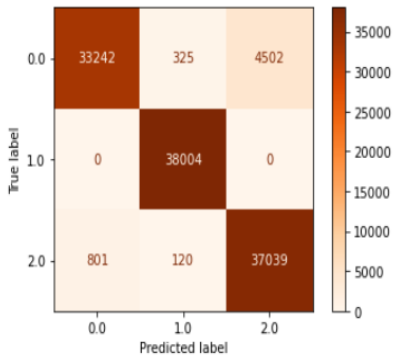
We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.58	0.64	0.61	38069
1.0	0.42	0.30	0.35	38004
2.0	0.54	0.58	0.54	37960
Accuracy			0.51	114033
Macro Avg	0.50	0.51	0.50	114033
Weighted Avg	0.50	0.51	0.50	114033

RandomForestClassifier

```
In [59]: from sklearn.metrics import plot_confusion_matrix
         plot_confusion_matrix(model_1,x_test,y_test,cmap='Oranges')
         plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



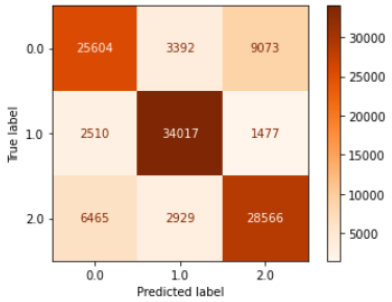
We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.98	0.87	0.92	38069
1.0	0.99	1.00	0.99	38004
2.0	0.89	0.98	0.93	37960
Accuracy			0.95	114033
Macro Avg	0.95	0.95	0.95	114033
Weighted Avg	0.95	0.95	0.95	114033

CatBoost

```
In [68]: from sklearn.metrics import plot_confusion_matrix
         plot_confusion_matrix(cat,x_test,y_test,cmap='Oranges')
         plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



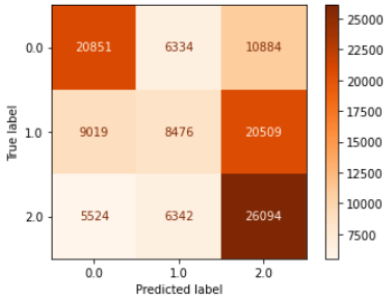
We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.74	0.67	0.70	38069
1.0	0.84	0.90	0.87	38004
2.0	0.73	0.75	0.74	37960
Accuracy	0.77	0.77	0.77	114033
Macro Avg	0.77	0.77	0.77	114033
Weighted Avg	0.77	0.77	0.77	114033

GaussianNB

```
In [73]: from sklearn.metrics import plot_confusion_matrix
         plot_confusion_matrix(gnb,x_test,y_test,cmap='Oranges')
         plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



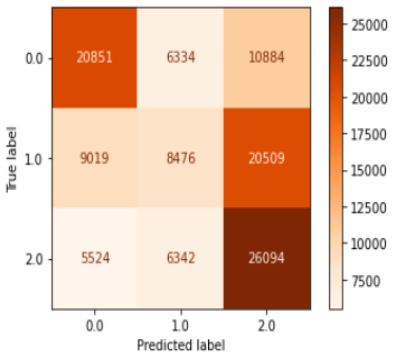
We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.59	0.55	0.57	38069
1.0	0.40	0.22	0.29	38004
2.0	0.45	0.69	0.55	37960
Accuracy			0.49	114033
Macro Avg	0.48	0.49	0.47	114033
Weighted Avg	0.48	0.49	0.47	114033

KneighbourClassifier

```
In [73]: from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(gnb,x_test,y_test,cmap='Oranges')
plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



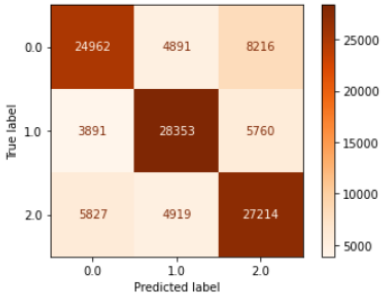
We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.96	0.72	0.82	38069
1.0	0.96	1.00	0.98	38004
2.0	0.80	0.96	0.87	37960
Accuracy		0.89	0.89	114033
Macro Avg	0.90	0.89	0.89	114033
Weighted Avg	0.90	.89	0.89	1140330

XGBoost Classifier

```
In [75]: from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(xgb,x_test,y_test,cmap='Oranges')
plt.grid(False)
```

```
G:\anaconda_software\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprec
ated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: Confusion
MatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



We can determine the precision , recall, F1-score and accuracy with the help of the confusion metrics.

Class	Precision	Recall	F1-Score	support
0.0	0.72	0.66	0.69	38069
1.0	0.74	0.75	0.74	38004
2.0	0.66	0.72	0.69	37960
Accuracy			0.71	114033
Macro Avg	0.71	0.71	0.71	114033
Weighted Avg	0.71	0.71	0.71	114033

Experimentation Results

The table shown below shows us the accuracy of all the classifiers used to train the machine learning model .It is clear from the table shown below that RandomForestClassifier() has the highest accuracy.

Algorithm	Accuracy
K-NN Classifier()	89%
GaussianNB()	49%
CatBoostClassifier()	77%
RandomForestClassifier()	95%
Logistic Regression	51%

Conclusion

Diabetes is a disease that has a lot of potential complications. It might be worthwhile to research how to precisely anticipate and diagnose this condition using machine learning.This project's primary goal was to build and implement methods for predicting diabetes using machine learning, and to assess the effectiveness of those methods.In light of the aforementioned experiments, we discovered that the suggested strategy makes use of a variety of classification learning techniques, including classifiers from KNN, Random Forest, CatBoost, Logistic Regression, and GaussianNB.The best result were obtained using the RandomForestClassifier algorithm for the Diabetes_012 dataset is 0.95, indicating that machine learning can be used to predict diabetes, but it's crucial to identify the right features, classifier, and data mining technique. We are unable to forecast the type of diabetes and if the person can develop the diabetes in future years based on the present data, therefore in the future we plan to do so while also examining the relative importance of each signal, which may increase the precision with which diabetes is predicted.. The experimental results can help medical professionals make early predictions and decisions to treat diabetes and save a patient's life.

Future Scope And Feasibility Study

Future Scope

We can develop an IOT device which can be connected to our Machine learning model which can help us to track diabetes instantly and accurately.The IOT device will take take all the features that that are used to predict the presence of diabetes in our model from the patient and those values can be used as input in our model to detect the presence of diabetes.

Feasibility Scope

A feasibility study determines if a proposed system is practical. It involves locating possible candidates, evaluating them, and choosing the most workable solution. This is done by taking a look at both the fundamental ideas of a new system as well as the system that currently exists in the area in question. It is research that assesses the likelihood of a project's success as well as its possible advantages and disadvantages. It evaluates the usability, organizational influence, user-satisfaction potential, and resource-effectiveness of a system concept. A feasibility study's objective is to understand the scope of the issue rather than to find a solution.

The main factors taken into account in the feasibility study are:

- Technical feasibility
- Operational feasibility
- Economic feasibility

Economic Feasibility

An evaluation of the economic viability determines whether the long-term financial viability of the cost of the new system. The objective of the economic feasibility analysis is to identify the financial advantages that the proposed system will provide to the business. Any capital investment must be expected to reduce costs associated with improving service quality and enhance profit in order to be justified. Economic viability is typically determined using the technical cost-benefit analysis as a base. Because it doesn't require any additional production or maintenance expenses, this project is economical. Python's free open-source software contains all the capabilities a typical user would need.The programme will have a more professional phase in the upcoming update, which will greatly enhance the effectiveness of the diabetic therapy.

Technical Feasibility

It is necessary to create an outline design of the system's input, output, file, database, software, and procedures in order to assess a system's technical viability. This can be described in terms of data volumes, trends, frequency of updating, cycles of activities, and so on in order to give a general overview of the technical system.The objective of the technical feasibility study is to gain a deeper understanding of available technological resources and their applicability to the anticipated requirements of the proposed system. It's an evaluation of the hardware and software to see how well it complies with the demands of the suggested system. All necessary libraries, packages, and instructions are already included in our project. Any device with an internet connection is able to use our software after it has been hosted on a site. As a result, it may operate without any type of external support, which makes it technically possible.

Operational Feasibility

Operational feasibility is a metric for how successfully a proposed system addresses issues, takes advantage of opportunities identified

during scope definition, and satisfies the standards established during the requirements analysis stage of system development. The suggested system is easy to use and may be run by anyone with just a computer device. mainly due to the software's user-friendly design, which is both straightforward and easy to use. For the purpose of using the software, the users are not required to undergo any further training. A thorough investigation into the suggested system's operational viability was undertaken, and it was discovered that the system is viable for the treatment of diabetes.

References

[1]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.
[2]https://www.researchgate.net/publication/291313554_A_Method_for_Classification_Using_Machine_Learning_Technique_for_Diabetes
[3]https://scikit-learn.org/stable/
[4]https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset
[5]https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/
[6]https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb
[7]Sharma, R.; Singh, S.N.; Khatri, S. Medical Data Mining Using Different Classification and Clustering Techniques: A Critical Survey. In Proceedings of the 2016 Second International Conference on Computational Intelligence & Communication Technology (CICT), Ghaziabad, India, 12–13 February 2016; pp. 687–691.
[9]Hasan, M.K.; Alam, M.A.; Das, D.; Hossain, E.; Hasan, M. Diabetes Prediction Using Ensembling of Different Machine Learning Classifiers. IEEE Access 2020, 8, 76516–76531.
[10]Sisodia, D.; Sisodia, D.S. Prediction of Diabetes using Classification Algorithms. Procedia Comput. Sci. 2018, 132, 1578–1585.
[11]Meng, X.-H.; Huang, Y.-X.; Rao, D.-P.; Zhang, Q.; Liu, Q. Comparison of three data mining models for predicting diabetes or prediabetes by risk factors. Kaohsiung J. Med. Sci. 2013, 29, 93–99.
[12]Abdulhadi, N.; Al-Mousa, A. Diabetes Detection Using Machine Learning Classification Methods. In Proceedings of the 2021

International Conference on Information Technology (ICIT), Amman, Jordan, 14–15 July 2021; pp. 350–354.
[13] K. Polat, S. Gunes and A. Aslan.2008.A cascade learning system for classification of diabetes disease: Generalized discriminant analysis and least square support vector machine, Expert Systems with Applications, vol. 34(1), pp. 214–221
[14] Quan Zou, Kaiyang Qu , Yamei Luo , Dehui Yin , Ying Ju and Hua Tang.2018.Frontiers in Genetics .
[15] Thangarasu Gunasekar and Assoc. Prof. Dr. Dominic P.D.D.2104. Prediction of Hidden Knowledge from Clinical Database using Data mining Technique, IEEE 978-1-4799-0059-6.
[16] Ayush Anand, Divya Shakti, Prediction of Diabetes Based on Personal Lifestyle Indicators, IEEE, International Conference on Next Generation Computing Technologies, pp. 673-676, 2015.
[17]Narges Razavian, Saul Blecker, Ann Marie Schmidt, Aaron Smith-McLallen, Somesh Nigam, and David Sontag .2015.PopulationLevel Prediction of Type 2 Diabetes From Claims Data and Analysis of Risk Factors,Original Report.

[20] Tejas N. Joshi, Prof. Pramila M. Chawan.2018.Diabetes Prediction Using Machine Learning Techniques from Journal of Engineering Research and Application ,ISSN: 2248-9622, Vol. 8, Issue 1
[18] Harleen, Dr. Pankaj Bhambri.2016. A Prediction Technique in Data Mining for Diabetes Mellitus from Journal of Management Sciences and Technology, ISSN -2347-5005
[19] M.Mounika, S.D.Suganya, B.Vijayashanthi, S.KrishnaAnand.2015.Predictive Analysis of Diabetic Treatment Using Classification Algorithm from International Journal of Computer Science and Information Technologies.
[21] V. Kumar and L. Velide.2014. A Data mining Approach for Prediction and Treatment Of diabetes Disease from IJCSIT

[22] Sassanian and G. Hari Sekaran.2015. Big Data Analytics Predicting Risk of Readmissions of Diabetic Patients fromInternational Journal of Science and Research, vol. 4.
[23] N.Deepika, Dr.S.Poonkuzhali.2018. Design of Hybrid Classifier for Prediction of Diabetes through Feature Relevance Analysis
from International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 10