Heart Disease Prediction Using Machine Learning Techniques

Priyanka Nelli Komath, SID: 11979142, nellikomap@coventry.ac.uk
Faculty Of Engineering, Environment And Computing, Coventry University
MSc Data Science And Computational Intelligence
Coventry, United Kingdom

Abstract: Heart disease has become one of the leading causes of hospitalization and death in the current era. With the advancements in computer technologies, detecting heart disease has become easier and beneficial for people to take the necessary steps for the treatment or it becomes severe. This paper discusses predicting the result of heart disease. I have applied different data preprocessing techniques and 5 different machine learning techniques - Logistic Regression, Random Forest Classifier, K Neighbors Classifier, Decision Tree Classifier, Gaussian NB. The performance of algorithms are compared using the accuracy score and F1 core on the test data, and is visualized using the confusion matrix. Accuracy of the models were compared with imbalanced, resampled dataset.

Keywords: Machine Learning; Heart disease prediction; Logistic Regression; Random Forest Classifier; K Neighbors Classifier; Decision Tree Classifier; GaussianNB; Feature scaling; Encoding

I. INTRODUCTION

The heart is one of the main parts of the human body, which is responsible to pump blood to the whole body. Cardiac disease can cause multiple function failures and lead to death. The cardio death rate is increasing rapidly every year. According to the report published by the World Health Organization in the year 2016, 31% of death happening around the world is due to heart-related problems (Motarwar, Duraphe, Suganya and Premalatha, 2020).

There are many factors or human habits that can lead to the chance for heart disease. Medical professionals have categorized these into two categories; (1) factors which can be changed and (2) the factors which cannot be changed. The first category includes age category, gender, and hereditary issues (Katarya and Srinivas, 2020). The risk factors which can be changed and could reduce the chances of heart failure are blood pressure, BMI, physical activities, smoking, diabetes, etc.

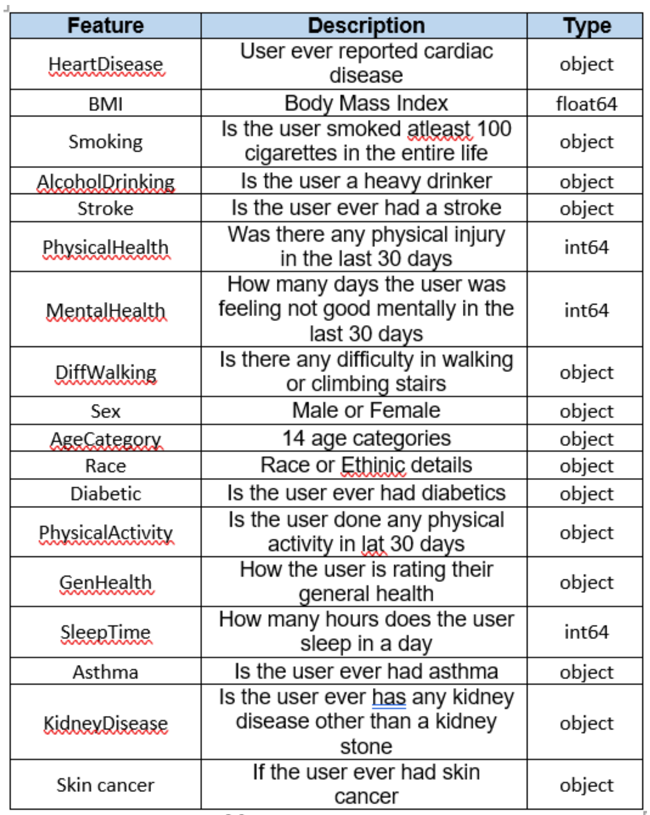
Cardio failure is a life-threatening disease, which needs to be diagnosed or predicted in an early stage. There are many methods that medical practitioners follow, but these procedures are time-consuming and expensive. With the advancement in the information technology field, machine learning algorithms can be used to predict or detect heart disease in an early stage, which is less expensive and beneficial for people to act at an early stage which potentially reduces the threat to life.

Medical conditions and lifestyle details of users are collected and analyzed using different machine learning techniques, which will detect certain patterns and correlations between attributes and this will be used in predicting the heart disease (Motarwar, Duraphe, Suganya and Premalatha, 2020).

There are multiple machine learning methods used for illness predictions; Random forest, Decision tree, KNN, CNN, Support vector machine, Naïve Bayes, logistic regression etc. In this study, the prediction of cardiac illness using 5 different machine learning models, the prediction accuracy, and the best model with high accuracy and F1 score is discussed.

II. THE DATASET

The dataset used for the course work is taken from Kaggle repository, which was originally from the CDC and BRFSS systems, and data was collected through the telephonic survey conducted on the health status of US Residents. It has 319795 instances and 18 attributes. The dataset has no missing or null values. Summary of the dataset with the type and the details of questions used for the survey is explained in *Table 1*.



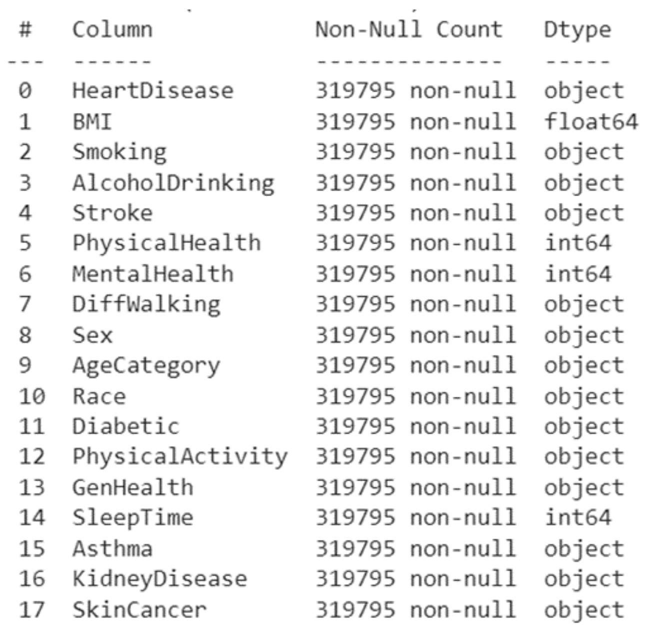
Feature	Description	Type
HeartDisease	User ever reported cardiac disease	object
BMI	Body Mass Index	float64
Smoking	Is the user smoked atleast 100 cigarettes in the entire life	object
AlcoholDrinking	Is the user a heavy drinker	object
Stroke	Is the user ever had a stroke	object
PhysicalHealth	Was there any physical injury in the last 30 days	int64
MentalHealth	How many days the user was feeling not good mentally in the last 30 days	int64
DiffWalking	Is there any difficulty in walking or climbing stairs	object
Sex	Male or Female	object
AgeCategory	14 age categories	object
Race	Race or Ethinic details	object
Diabetic	Is the user ever had diabetics	object
PhysicalActivity	Is the user done any physical activity in lat 30 days	object
GenHealth	How the user is rating their general health	object
SleepTime	How many hours does the user sleep in a day	int64
Asthma	Is the user ever had asthma	object
KidneyDisease	Is the user ever has any kidney disease other than a kidney stone	object
Skin cancer	If the user ever had skin cancer	object

Table 1: Dataset Features

III. DATA ANALYSIS

The data analysis process is done to understand the features and instances to have better data preparation.

After analysing the dataset, there are no missing or null values (figure 1) found. So, we do not have to do any adjustments to the data.



```
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   HeartDisease      319795 non-null  object
 1   BMI               319795 non-null  float64
 2   Smoking           319795 non-null  object
 3   AlcoholDrinking   319795 non-null  object
 4   Stroke            319795 non-null  object
 5   PhysicalHealth    319795 non-null  int64
 6   MentalHealth      319795 non-null  int64
 7   DiffWalking       319795 non-null  object
 8   Sex               319795 non-null  object
 9   AgeCategory       319795 non-null  object
 10  Race              319795 non-null  object
 11  Diabetic          319795 non-null  object
 12  PhysicalActivity  319795 non-null  object
 13  GenHealth         319795 non-null  object
 14  SleepTime         319795 non-null  int64
 15  Asthma            319795 non-null  object
 16  KidneyDisease     319795 non-null  object
 17  SkinCancer        319795 non-null  object
```

Figure 1 : Dataset information

When comparing the numbers, the percentile of people who have heart disease is only 8.5%, which brings an unbalanced data issue.

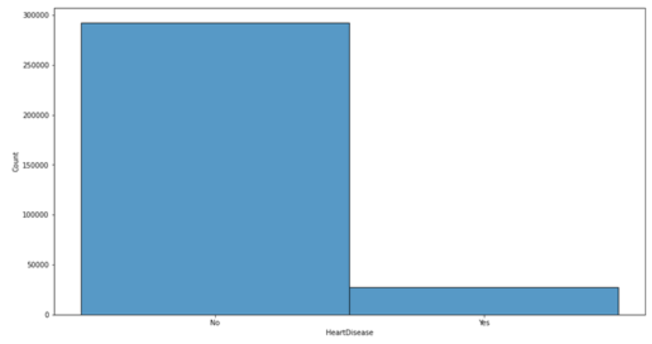


Figure 2: Imbalanced dataset

After visualising the pie-chart of multiple attributes, we can see that few features affect people with heart disease more than people who don't have cardiac disease. For example, in the given 'Stroke' (figure 3) chart it is evident that 16% of people who reported heart disease had a stroke while the percentage of people who don't have heart disease is comparatively small. This is one of the helpful features for cardiac disease prediction.

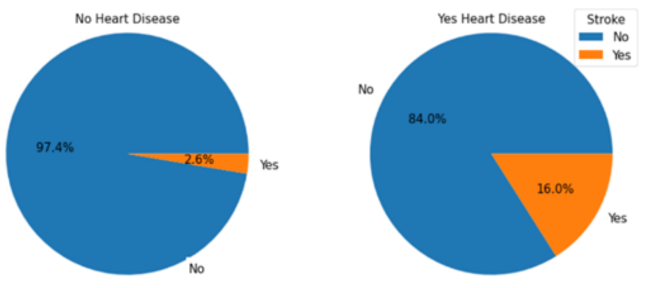


Figure 3: Users with and without heart disease

Figure 4 represents the correlation between the attributes. From the figure, it is evident that the attributes DriftWalking, Stroke, PhyicalHealth, Diabetic and AgeCategory have a fair correlation with the output variable 'HeartDisease'.

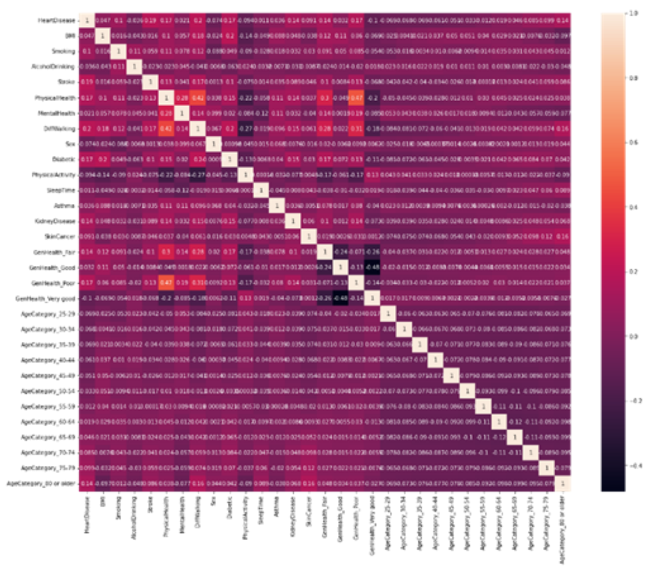


Figure 4: Correlation Map

IV. METHODOLOGY

This research paper discusses the prediction of cardiac disease. The dataset contains 319795 instances and 18 attributes, the output attribute 'HeartDisease' with a value '1' represents the patients with cardiac disease and '0' for people without the disease.
5 different machine learning algorithms are used for the prediction.

- Logistic Regression
- Random Forest
- KNN - K Neighbors Classifier
- Decision Tree
- Gaussian NB

(1) Logistic Regression

Logistic regression is a statistical model mainly used for the analysis of classification problems. This is a simple and efficient method for linear and binary classification problems to achieve very good performance with linearly separable classes (Subasi, 2020).

$$Logistic\ function = \frac{1}{1+e^{-x}}$$

(2) Random Forest

Random forest is a cluster of decision trees. Each individual tree has an accuracy rate. The prediction method varies depending on the problem type (Education, 2022).

(3) KNN - K Neighbors Classifier

KNN is a supervised ML algorithm and can be used on both regression and classification problems. Model checks for the entire dataset to locate the K nearest instance and then output the mode for the selected problem. The accuracy of this model varies from instance to instance when compared with other machine learning algorithms (Farzana and Veeraiah, 2022).

(4) Decision Tree

The decision tree is one of the predictive models used in machine learning. Algorithms identify different ways to split data based on conditions. It is one of the most widely used methods for supervised learning and can be used for both regression and classification problems (Zhang, 2021).

(5) Gaussian NB

Gaussian Naive Bayes algorithm is a special type of naïve bayes algorithm which is used for the data with continuous values. The model assumes that there is no covariance between the dimensions (Gaussian Naive Bayes, 2022). The formula for Bayes theorem is as follows:

V. EXPERIMENTAL SETUP

After analyzing the data, necessary pre-processing steps were carried out to make the data ready in the acceptable format of the machine learning model algorithms chosen.

(A) DATA-PREPROCESSING

(1) Missing Data

Many of the real-world data sets will have missing values. Handling these values is important otherwise it will have poor relationship with the target value prediction. Missing data can be handled by removing null valued instances from the dataset.

We do not have missing value in our dataset, so it is not required to do any adjustments in the chosen set.

(2) Duplicate Data

There were originally 319795 instances in the dataset. During the analysis 18078 instances were found duplicates and removed from the set. After the clean-up, the dataset has 301717 samples and 18 attributes.

(3) Feature Selection

To find the most important features to be used in the heart disease prediction feature selection algorithm is used. After analysis, unwanted attributes are dropped from the original dataset. It is important to apply feature selection algorithms to the dataset for the following reasons:

- It helps in identifying the most suitable variables which essentially reduces the input attributes used in predicting the target value.

- For improved accuracy.

- To reduce the training and testing time

After implementing the feature selection, it is decided to drop the attribute 'Race' as it is not an important variable for the heart disease prediction.

(4)Feature Scaling

The standard Scaler method which is imported from the sklearn library is used in this research to standardize the range of feature data. The method of calculation is to find the distribution mean and standard deviation for each of the features and use the given formula to calculate the new data point.

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

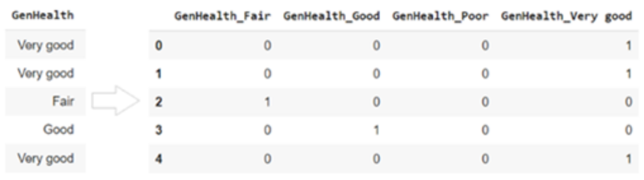
To implement the scaling method, our dataset is divided into 2 lists based on the unique values available in each attribute. 'StandardScaler' method is then used and the features are centered on the mean with a standard deviation, which will potentially reduce the distance.

(5) Categorical Feature Encoding

Most of the machine learning models work only with numerical data, so it is important to convert categorical data into numerical values for the algorithms to run and process data.

We have 13 categorical columns in the original dataset. After analysis, attributes that contain 2 unique category values are directly replaced with 0 and 1. For example, attribute 'Diabetic' had values 'Yes' and 'No' which were replaced by 1 and 0. After this process, we are left with 2 categorical attributes 'AgeCategory' and 'GenHealth'. We have 13 unique categories if we take AgeCatgory as an example. If we directly assign numbers to these categories they will vary from 0 to 12, which will lead the machine learning algorithm to give more weightage to the category with a value 12.

To avoid this issue, the 'Dummy Encoding' method is used to convert categorical values to binary values (figure 5). After implementing the dummy method the result will be concatenated with the original data and potentially remove the column for correlation.



GenHealth		GenHealth_Fair	GenHealth_Good	GenHealth_Poor	GenHealth_Very good
Very good	0	0	0	0	1
Very good	1	0	0	0	1
Fair	2	1	0	0	0
Good	3	0	1	0	0
Very good	4	0	0	0	1

Figure 5 : Categorical Feature Encoding

(6) Imbalanceded data handling

Imbalanced data is one of the common problems in machine learning classification. This arises when there is a disproportionate ratio of observations in the classes.

There are multiple methods to handle imbalanced data and they are SMOTE, NearMiss, Random over and under sampling or also known as resampling methods.

In this research we are using random over and under-sampling techniques to resample the dataset. I have created a dataset of 80% training data and applied both techniques to train different models and compare the accuracy score. After comparing both oversampling and undersampling methods on the selected algorithm we could see that the accuracy rate of each induvial classification method has gone down where in the score was growing up when we used the undersampling method. Therefor, I am using undersampled data for the hyper parameter tuning

After implementing the under-sampling method the dataset is balanced now and the graphical representation of balanced dataset is given in the figure

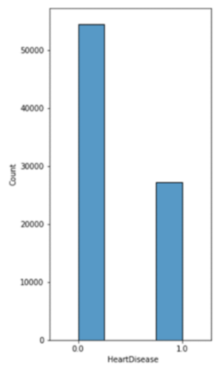


Figure 6: Balanced data

(B) IMPLEMENTATION METHODS

In this research paper, Python is used to implement the selected machine learning algorithms, and Google colab was used as an IDE. Original dataset was uploaded in google drive. Uploaded csv file read after mounting the drive in the code and then the data pre-processing steps were implemented

After pre-processing steps, the data was split into testing and training sets and assigned 20% of the set for testing and 80% for training; Then imported required algorithms from the sklearn library. I have trained all 5 models with imbalanced, over and under-sampled datasets and validated the results using the test data set.

I have started with Logistic Regression and trained the model with the training part of our dataset. Hypertuning to the parameter is done in the dataset as there was a huge imbalanced set. Prediction of test data and the accuracy calculation is done after fitting the train data to the model using fit function. The accuracy score for the Logistic Regression model on imbalanced test data is 91.09% and the accuracy is increased to 91.12% after implementing the resampling technique and training the method using the under-sampled dataset.

K nearest neighbor classification is another efficient model used for the prediction in this research. There are 2 ways to implement the KNN model. In the traditional method, we have to choose the number of neighbours and assign it to the variable K. Next we have to calculate the 'Euclidean Distance' between the chosen points to get the nearest neighbor. The disadvantage of this method is that the process of finding the K value becomes complex as the number of datapoints in the chosen dataset grows. In this research, KNeighborsClassifier was imported from sklearn library, and the prediction of test data was done after fitting the train data to the model. The accuracy score for the KNN model on our imbalanced test data is 90.13% where it increased to 91.6% after resampling the set using the under sampling method.

Next, we used the Decision Tree method, which was imported from the sklearn library. Here it splits a node into multiple sub-nodes and selects the most relevant sub nodes. We got an accuracy of 85.5% which I rounded to 86% on our imbalanced test set. There was a significant improvement in the accuracy score (96.7%) after implementing the resampling technique and training the method using the under-sampled dataset.

We have then tested the prediction using the random forest method, which is essentially the cluster of decision trees. The result will be determined by taking the average of individual decision trees for the classification issue. After fitting the test data to the method and prediction on imbalanced test data we got an accuracy score of 89.47% and an accuracy of 97.5 % on the under-sampled dataset. This is the highest accuracy score received when comparing with other algorithms on the resampled data.

Last method we have used in the research is Gaussian Naïve Baye algorithm. After implementing the model we got an accuracy score of 74.3 % and 74. % on imbalanced and under-sampled datasets respectively. This is the lowest accuracy score we received compared to the other algorithm used for the heart disease prediction in this research.

VI. RESULTS

In this research, we have successfully implemented 5 machine learning algorithms to predict heart disease on the allocated 20% test data. Here we used Accuracy, F1 Score, Precision, and Recall to evaluate the model.

Accuracy: The accuracy value indicates the number of correctly predicted values from the chosen dataset. This is calculated by adding the number of true positives and negatives divided by the sum of true and false positives and negatives.

F1 Score: F1 Score is the mean between recall and precision, which is a key indicator to rate the performance of a model. For the imbalanced dataset F1 score is more important than the accuracy.

Precision and Recall : Precision indicates the percentage of relevant results where in Recall refers to the percent of relevant results which are rightly classified by the chosen algorithm.

Figure 5 shows the accuracy core of each algorithm and the table contains accuracy, F1 core, Precision and Recall value of each machine learning model used in this study.

(1) *Results on imbalanced data*

In this research, we have trained the model with imbalanced data. Figure 6 shows the Accuracy, F1 score, Recall and Precision of all 5 machine learning methods trained using the imbalanced data set. Here it is evident that the accuracy core of Logistic regression is high compared to other models.

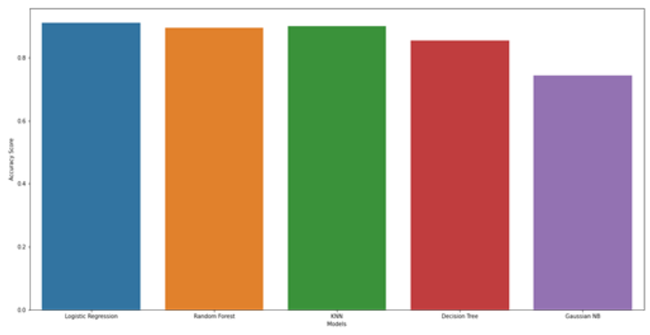
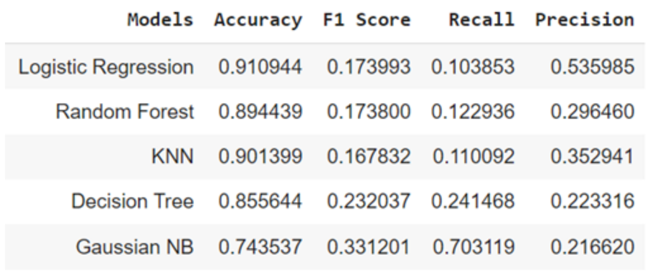


Figure 6 : Accuracy score of ML models

But we cannot conclude the result only based on the accuracy as the F1 score is a better measure for the imbalanced dataset. When we compare F1 score of all 5 methods, Gaussian Naïve Bayes algorithm gives a comparatively higher F1 score than the other 4 models.

Figure 7 shows the Accuracy, Precision, Recall, and F1 score of all 5 machine learning algorithms used on the imbalanced dataset.



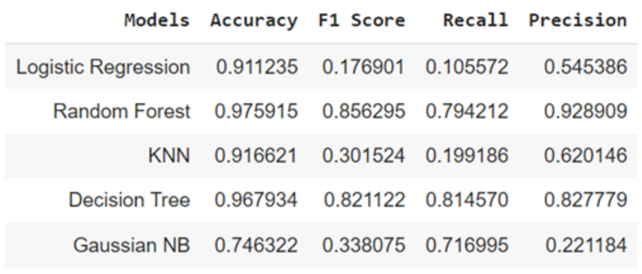
Models	Accuracy	F1 Score	Recall	Precision
Logistic Regression	0.910944	0.173993	0.103853	0.535985
Random Forest	0.894439	0.173800	0.122936	0.296460
KNN	0.901399	0.167832	0.110092	0.352941
Decision Tree	0.855644	0.232037	0.241468	0.223316
Gaussian NB	0.743537	0.331201	0.703119	0.216620

Figure 7 : Accuracy, Recall, Precision and F1 score of imbalanced data

(2) Results on the balanced dataset

The resampling technique is used to change the composition of our imbalanced dataset. The undersampling method is used in this research to balance the set. It removes the instance from the majority class of the training dataset for balancing the class distribution.

Figure shows the scores after balancing the data and calculating the score of machine learning methods. It is very clear from the figure that the 'Random Forest' Classification trained with undersampled data has the highest accuracy score and F1 score combination.



Models	Accuracy	F1 Score	Recall	Precision
Logistic Regression	0.911235	0.176901	0.105572	0.545386
Random Forest	0.975915	0.856295	0.794212	0.928909
KNN	0.916621	0.301524	0.199186	0.620146
Decision Tree	0.967934	0.821122	0.814570	0.827779
Gaussian NB	0.746322	0.338075	0.716995	0.221184

Figure 8 : Accuracy, Recall, Precision and F1 score of balanced data

VII. DISCUSSION AND CONCLUSION

The objective of this coursework was to predict the heart disease which will be beneficial in the medical fields as well as for the people or patients to take an early action based on the predicted result.

The dataset which was chosen from the Kaggle repository was divided into testing and training sets. Necessary pre-processing techniques were performed on the dataset such as feature selection, handling missing and duplicate values, feature encoding and scaling. Undersampling classification was then used to balance the classes.

Five machine learning algorithms were considered to implement solution for the projected problem; Logistic Regression, Random Forest Classifier, K Neighbors Classifier, Decision Tree Classifier, Gaussian NB. I have trained all 5 models with imbalanced, oversampled, and undersampled data sets. After comparing the result it was clear that undersampled set worked well on the machine learning models and concluded that the 'Random Forest classification' is the best model with a high accuracy value and F1 score of 0.975 and 0.856 respectively on the testing dataset.

VII. REFERENCES

1. Subasi, A., 2020. Practical Machine Learning for Data Analysis Using Python,.

2. Education, I., 2022. What is Random Forest?. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/random-forest>

3. S. Farzana and D. Veeraiah, "Dynamic Heart Disease Prediction using Multi-Machine Learning Techniques," 2020 5th International Conference on Computing, Communication and Security (ICCCS), 2020, pp. 1-5, doi: 10.1109/ICCCS49678.2020.9277165.

4. Zhang, X., 2021. Using Data Visualization to Analyze the Correlation of Heart Disease Triggers and Using Machine Learning to Predict Heart Disease. 2021 3rd International Conference on Intelligent Medicine and Image Processing,.

5. OpenGenus IQ: Computing Expertise & Legacy. 2022. Gaussian Naive Bayes. [online] Available at: <https://iq.opengenus.org/gaussian-naive-bayes/>

6. Motarwar, P., Duraphe, A., Suganya, G. and Premalatha, M., 2020. Cognitive Approach for Heart Disease Prediction using Machine Learning. [online]

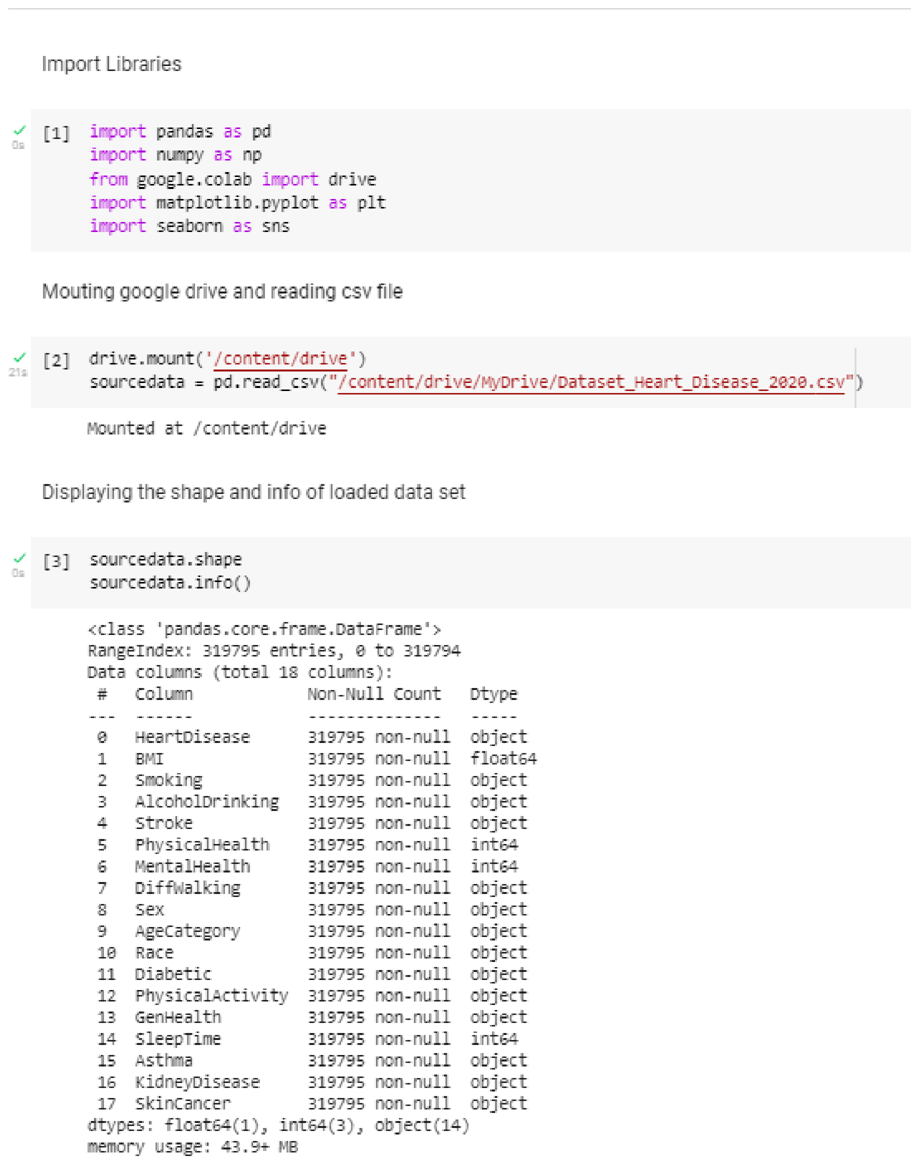
Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9077680>

7. R. Katarya and P. Srinivas, "Predicting Heart Disease at Early Stages using Machine Learning: A Survey," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 302-305, doi: 10.1109/ICESC48915.2020.9155586.

8. Kaggle.com. 2022. Personal Key Indicators of Heart Disease. [online] Available at: <https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease?select=heart_2020_cleaned.csv>

IX. APPENDIX

Link to the code : https://github.com/priyankaharidasnk/MachineLearning



Import Libraries

```
[1] import pandas as pd
    import numpy as np
    from google.colab import drive
    import matplotlib.pyplot as plt
    import seaborn as sns
```

Mouting google drive and reading csv file

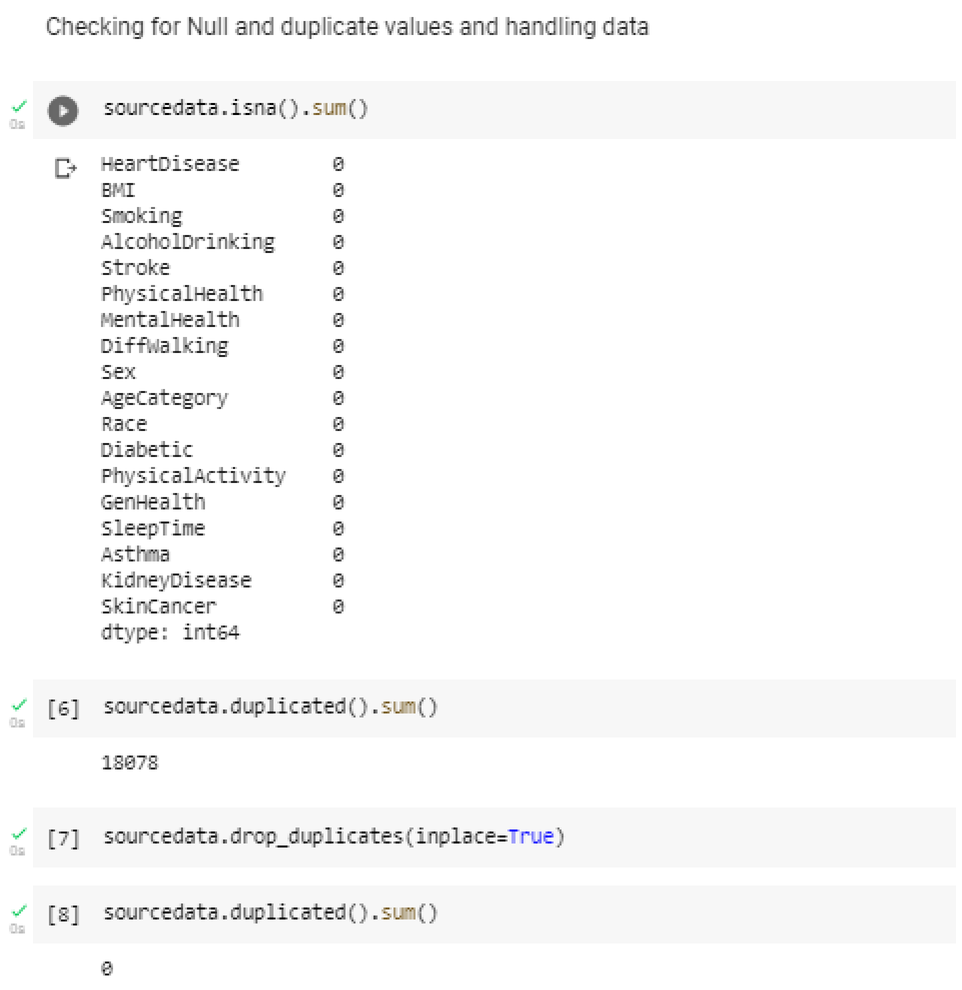
```
[2] drive.mount('/content/drive')
    sourcedata = pd.read_csv("/content/drive/MyDrive/Dataset_Heart_Disease_2020.csv")

    Mounted at /content/drive
```

Displaying the shape and info of loaded data set

```
[3] sourcedata.shape
    sourcedata.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 319795 entries, 0 to 319794
    Data columns (total 18 columns):
     #   Column            Non-Null Count   Dtype
    ---  ------            --------------   -----
     0   HeartDisease      319795 non-null  object
     1   BMI               319795 non-null  float64
     2   Smoking           319795 non-null  object
     3   AlcoholDrinking   319795 non-null  object
     4   Stroke            319795 non-null  object
     5   PhysicalHealth    319795 non-null  int64
     6   MentalHealth      319795 non-null  int64
     7   DiffWalking       319795 non-null  object
     8   Sex               319795 non-null  object
     9   AgeCategory       319795 non-null  object
     10  Race              319795 non-null  object
     11  Diabetic          319795 non-null  object
     12  PhysicalActivity  319795 non-null  object
     13  GenHealth         319795 non-null  object
     14  SleepTime         319795 non-null  int64
     15  Asthma            319795 non-null  object
     16  KidneyDisease     319795 non-null  object
     17  SkinCancer        319795 non-null  object
    dtypes: float64(1), int64(3), object(14)
    memory usage: 43.9+ MB
```



Checking for Null and duplicate values and handling data

```
sourcedata.isna().sum()
```

```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

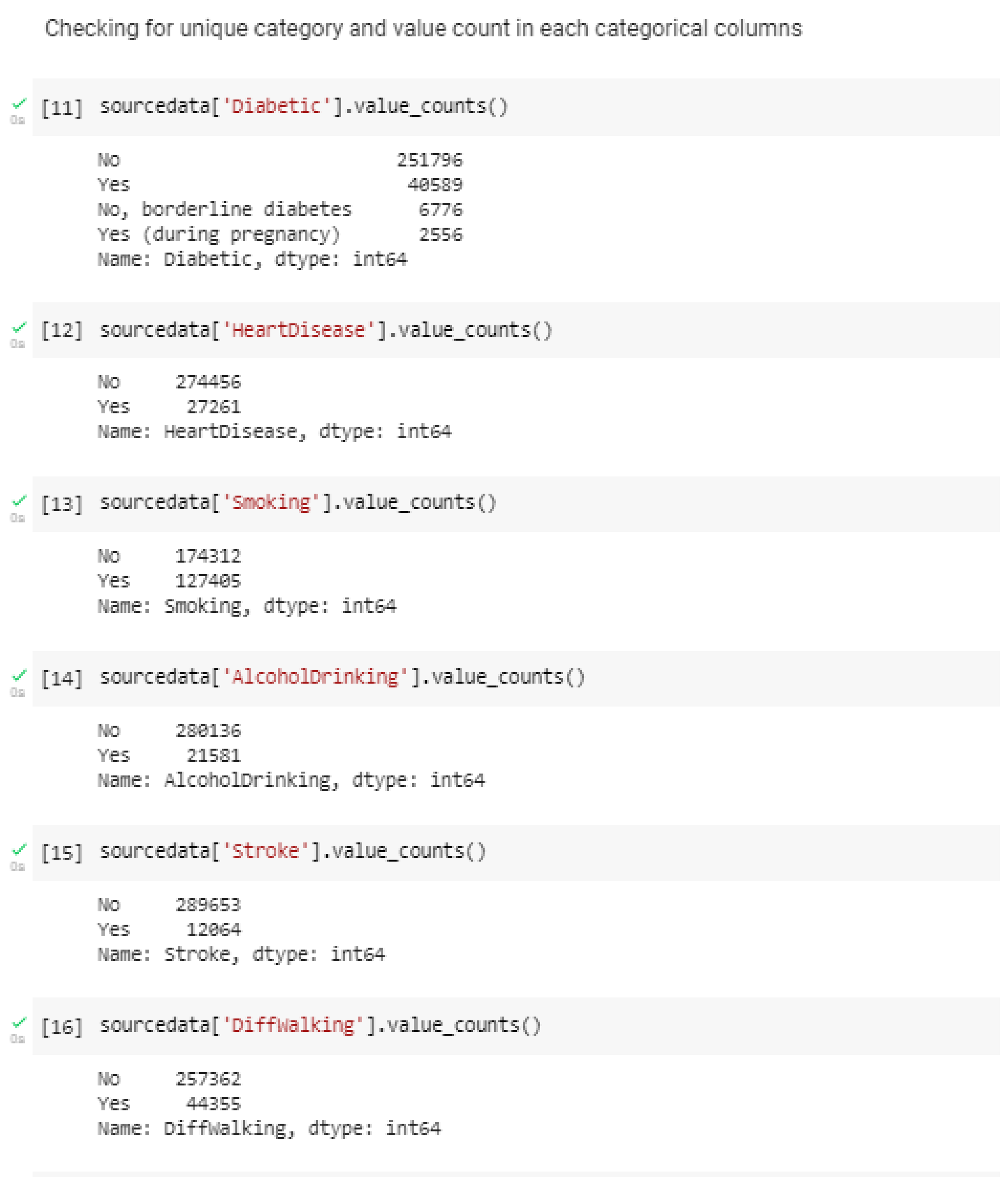
```
[6] sourcedata.duplicated().sum()
```

```
18078
```

```
[7] sourcedata.drop_duplicates(inplace=True)
```

```
[8] sourcedata.duplicated().sum()
```

```
0
```



Checking for unique category and value count in each categorical columns

```
[11] sourcedata['Diabetic'].value_counts()
```

```
No                          251796
Yes                          40589
No, borderline diabetes       6776
Yes (during pregnancy)        2556
Name: Diabetic, dtype: int64
```

```
[12] sourcedata['HeartDisease'].value_counts()
```

```
No     274456
Yes     27261
Name: HeartDisease, dtype: int64
```

```
[13] sourcedata['Smoking'].value_counts()
```

```
No     174312
Yes    127405
Name: Smoking, dtype: int64
```

```
[14] sourcedata['AlcoholDrinking'].value_counts()
```

```
No     280136
Yes     21581
Name: AlcoholDrinking, dtype: int64
```

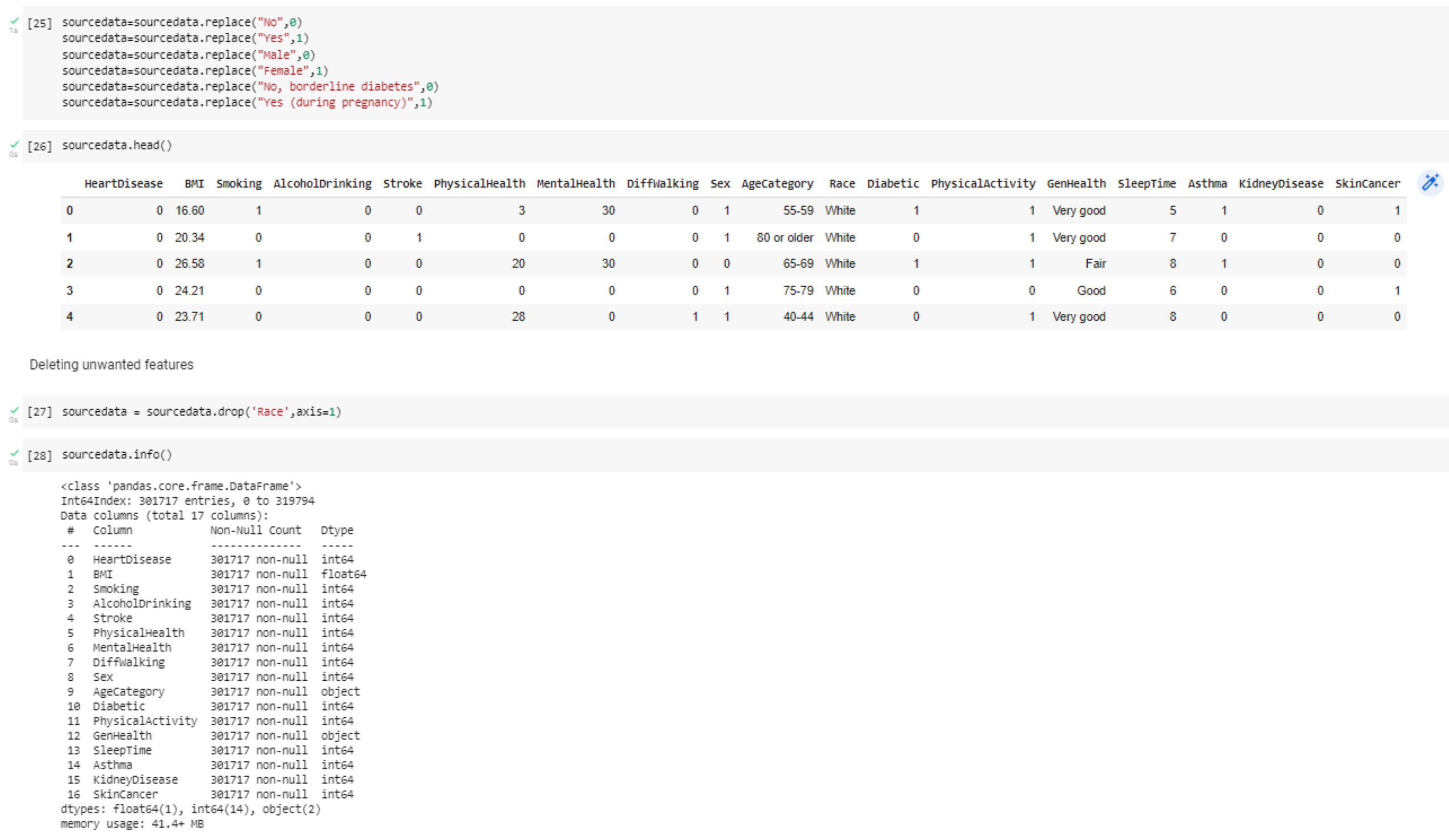
```
[15] sourcedata['Stroke'].value_counts()
```

```
No     289653
Yes     12064
Name: Stroke, dtype: int64
```

```
[16] sourcedata['DiffWalking'].value_counts()
```

```
No     257362
Yes     44355
Name: DiffWalking, dtype: int64
```

Replacing Yes and No values 1 and 0



```
[25] sourcedata=sourcedata.replace("No",0)
     sourcedata=sourcedata.replace("Yes",1)
     sourcedata=sourcedata.replace("Male",0)
     sourcedata=sourcedata.replace("Female",1)
     sourcedata=sourcedata.replace("No, borderline diabetes",0)
     sourcedata=sourcedata.replace("Yes (during pregnancy)",1)
```

```
[26] sourcedata.head()
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity	GenHealth	SleepTime	Asthma	KidneyDisease	SkinCancer
0	0	16.60	1	0	0	3	30	0	1	55-59	White	1	1	Very good	5	1	0	1
1	0	20.34	0	0	1	0	0	0	1	80 or older	White	0	1	Very good	7	0	0	0
2	0	26.58	1	0	0	20	30	0	0	65-69	White	1	1	Fair	8	1	0	0
3	0	24.21	0	0	0	0	0	0	1	75-79	White	0	0	Good	6	0	0	1
4	0	23.71	0	0	0	28	0	1	1	40-44	White	0	1	Very good	8	0	0	0

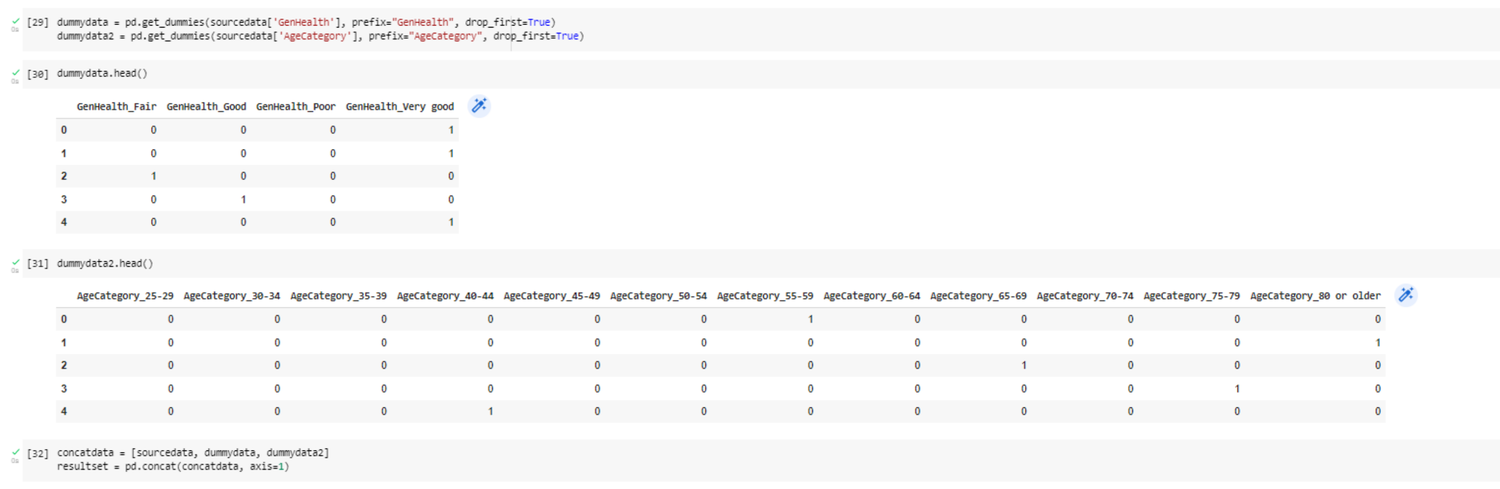
Deleting unwanted features

```
[27] sourcedata = sourcedata.drop('Race',axis=1)
```

```
[28] sourcedata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301717 entries, 0 to 319794
Data columns (total 17 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   HeartDisease      301717 non-null  int64  
 1   BMI               301717 non-null  float64
 2   Smoking           301717 non-null  int64  
 3   AlcoholDrinking   301717 non-null  int64  
 4   Stroke            301717 non-null  int64  
 5   PhysicalHealth    301717 non-null  int64  
 6   MentalHealth      301717 non-null  int64  
 7   DiffWalking       301717 non-null  int64  
 8   Sex               301717 non-null  int64  
 9   AgeCategory       301717 non-null  object 
 10  Diabetic          301717 non-null  int64  
 11  PhysicalActivity  301717 non-null  int64  
 12  GenHealth         301717 non-null  object 
 13  SleepTime         301717 non-null  int64  
 14  Asthma            301717 non-null  int64  
 15  KidneyDisease     301717 non-null  int64  
 16  SkinCancer        301717 non-null  int64  
dtypes: float64(1), int64(14), object(2)
memory usage: 41.4+ MB
```

Categorical feature encoding



```
[29] dummydata = pd.get_dummies(sourcedata['GenHealth'], prefix="GenHealth", drop_first=True)
     dummydata2 = pd.get_dummies(sourcedata['AgeCategory'], prefix="AgeCategory", drop_first=True)
```

```
[30] dummydata.head()
```

	GenHealth_Fair	GenHealth_Good	GenHealth_Poor	GenHealth_Very good
0	0	0	0	1
1	0	0	0	1
2	1	0	0	0
3	0	1	0	0
4	0	0	0	1

```
[31] dummydata2.head()
```

	AgeCategory_25-29	AgeCategory_30-34	AgeCategory_35-39	AgeCategory_40-44	AgeCategory_45-49	AgeCategory_50-54	AgeCategory_55-59	AgeCategory_60-64	AgeCategory_65-69	AgeCategory_70-74	AgeCategory_75-79	AgeCategory_80 or older
0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	1	0	0	0	0	0	0	0	0

```
[32] concatdata = [sourcedata, dummydata, dummydata2]
     resultset = pd.concat(concatdata, axis=1)
```

Deleting original fields(AgeCategory,GenHealth) after applying dummies

```
[34] resultset = resultset.drop('AgeCategory',axis=1)
     resultset = resultset.drop('GenHealth',axis=1)
```

```
[35] resultset.info()
```

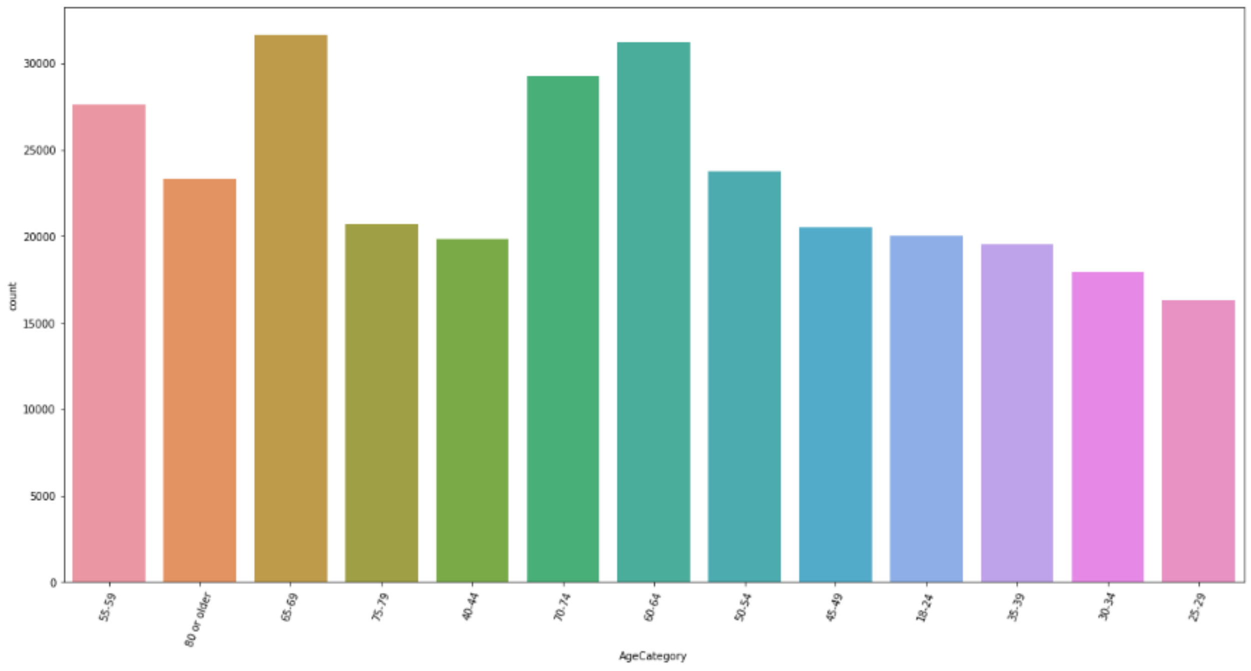
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301717 entries, 0 to 319794
Data columns (total 31 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   HeartDisease          301717 non-null  int64  
 1   BMI                   301717 non-null  float64
 2   Smoking               301717 non-null  int64  
 3   AlcoholDrinking       301717 non-null  int64  
 4   Stroke                301717 non-null  int64  
 5   PhysicalHealth        301717 non-null  int64  
 6   MentalHealth          301717 non-null  int64  
 7   DiffWalking           301717 non-null  int64  
 8   Sex                   301717 non-null  int64  
 9   Diabetic              301717 non-null  int64  
 10  PhysicalActivity      301717 non-null  int64  
 11  SleepTime             301717 non-null  int64  
 12  Asthma                301717 non-null  int64  
 13  KidneyDisease         301717 non-null  int64  
 14  SkinCancer            301717 non-null  int64  
 15  GenHealth_Fair        301717 non-null  uint8  
 16  GenHealth_Good        301717 non-null  uint8  
 17  GenHealth_Poor        301717 non-null  uint8  
 18  GenHealth_Very good   301717 non-null  uint8  
 19  AgeCategory_25-29     301717 non-null  uint8  
 20  AgeCategory_30-34     301717 non-null  uint8  
 21  AgeCategory_35-39     301717 non-null  uint8  
 22  AgeCategory_40-44     301717 non-null  uint8  
 23  AgeCategory_45-49     301717 non-null  uint8  
 24  AgeCategory_50-54     301717 non-null  uint8  
 25  AgeCategory_55-59     301717 non-null  uint8  
 26  AgeCategory_60-64     301717 non-null  uint8  
 27  AgeCategory_65-69     301717 non-null  uint8  
 28  AgeCategory_70-74     301717 non-null  uint8  
 29  AgeCategory_75-79     301717 non-null  uint8  
 30  AgeCategory_80 or older  301717 non-null  uint8  
dtypes: float64(1), int64(14), uint8(16)
memory usage: 41.4 MB
```

Plotting categorical attributes

```
[36] import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[37] plt.rcParams['figure.figsize'] = (20, 10)
     sns.countplot(x='AgeCategory',data=sourcedata)
     plt.xticks(rotation=70)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 <a list of 13 Text major ticklabel objects>)
```



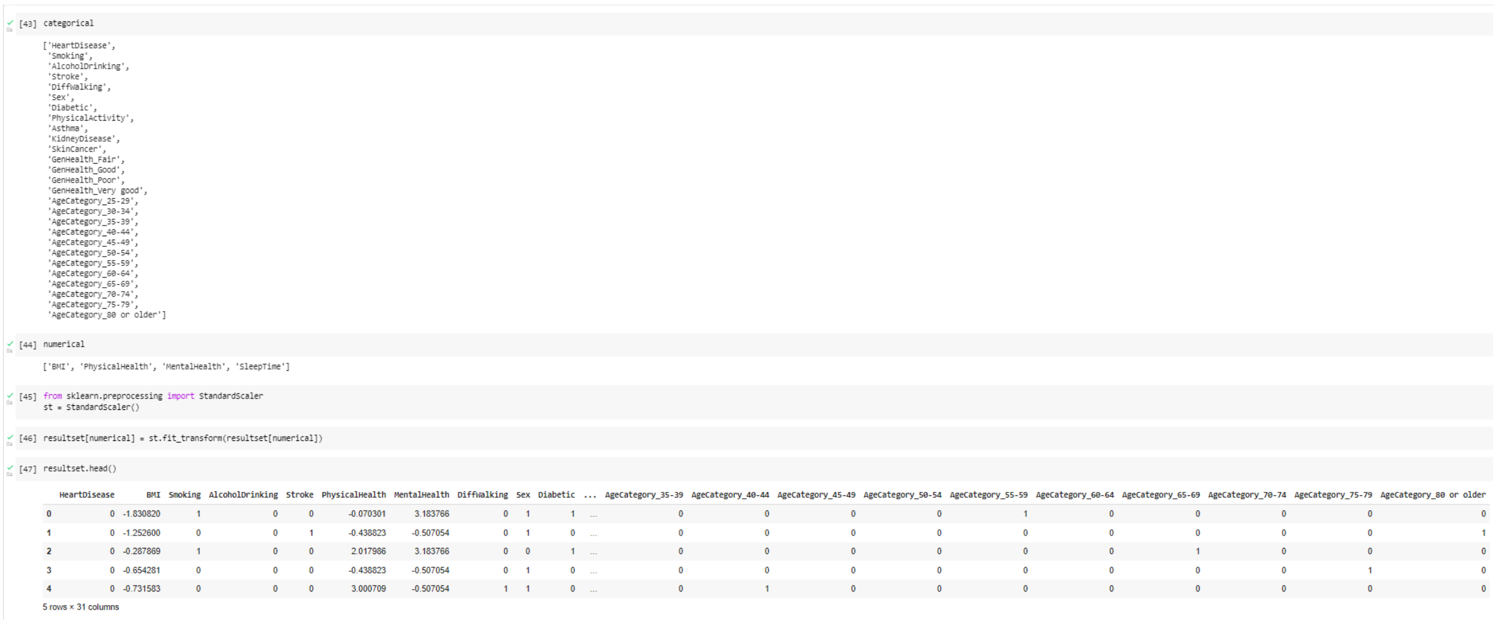
Feature Scaling

```
[40] resultset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301717 entries, 0 to 319794
Data columns (total 31 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   HeartDisease             301717 non-null  int64  
 1   BMI                      301717 non-null  float64
 2   Smoking                  301717 non-null  int64  
 3   AlcoholDrinking          301717 non-null  int64  
 4   Stroke                   301717 non-null  int64  
 5   PhysicalHealth           301717 non-null  int64  
 6   MentalHealth             301717 non-null  int64  
 7   DiffWalking              301717 non-null  int64  
 8   Sex                      301717 non-null  int64  
 9   Diabetic                 301717 non-null  int64  
 10  PhysicalActivity         301717 non-null  int64  
 11  SleepTime                301717 non-null  int64  
 12  Asthma                   301717 non-null  int64  
 13  KidneyDisease            301717 non-null  int64  
 14  SkinCancer               301717 non-null  int64  
 15  GenHealth_Fair           301717 non-null  uint8  
 16  GenHealth_Good           301717 non-null  uint8  
 17  GenHealth_Poor           301717 non-null  uint8  
 18  GenHealth_Very good      301717 non-null  uint8  
 19  AgeCategory_25-29        301717 non-null  uint8  
 20  AgeCategory_30-34        301717 non-null  uint8  
 21  AgeCategory_35-39        301717 non-null  uint8  
 22  AgeCategory_40-44        301717 non-null  uint8  
 23  AgeCategory_45-49        301717 non-null  uint8  
 24  AgeCategory_50-54        301717 non-null  uint8  
 25  AgeCategory_55-59        301717 non-null  uint8  
 26  AgeCategory_60-64        301717 non-null  uint8  
 27  AgeCategory_65-69        301717 non-null  uint8  
 28  AgeCategory_70-74        301717 non-null  uint8  
 29  AgeCategory_75-79        301717 non-null  uint8  
 30  AgeCategory_80 or older  301717 non-null  uint8  
dtypes: float64(1), int64(14), uint8(16)
memory usage: 49.5 MB
```

```
[41] categorical=[]
     numerical=[]
```

```
[42] for column in resultset.columns:
         if resultset[column].nunique() < 5:
             categorical.append(column)
         else:
             numerical.append(column)
```



```
[43] categorical

['HeartDisease',
 'Smoking',
 'AlcoholDrinking',
 'Stroke',
 'DiffWalking',
 'Sex',
 'Diabetic',
 'PhysicalActivity',
 'Asthma',
 'KidneyDisease',
 'SkinCancer',
 'GenHealth_Fair',
 'GenHealth_Good',
 'GenHealth_Poor',
 'GenHealth_Very good',
 'AgeCategory_25-29',
 'AgeCategory_30-34',
 'AgeCategory_35-39',
 'AgeCategory_40-44',
 'AgeCategory_45-49',
 'AgeCategory_50-54',
 'AgeCategory_55-59',
 'AgeCategory_60-64',
 'AgeCategory_65-69',
 'AgeCategory_70-74',
 'AgeCategory_75-79',
 'AgeCategory_80 or older']
```

```
[44] numerical

['BMI', 'PhysicalHealth', 'MentalHealth', 'SleepTime']
```

```
[45] from sklearn.preprocessing import StandardScaler
     st = StandardScaler()
```

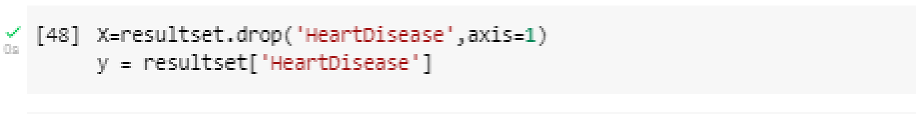
```
[46] resultset[numerical] = st.fit_transform(resultset[numerical])
```

```
[47] resultset.head()
```

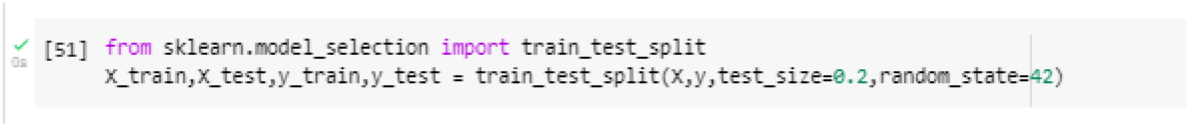
	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	Diabetic	...	AgeCategory_35-39	AgeCategory_40-44	AgeCategory_45-49	AgeCategory_50-54	AgeCategory_55-59	AgeCategory_60-64	AgeCategory_65-69	AgeCategory_70-74	AgeCategory_75-79	AgeCategory_80 or older
0	0	-1.830820	1	0	0	-0.070301	3.183766	0	1	1	...	0	0	0	0	1	0	0	0	0	0
1	0	-1.252600	0	0	1	-0.438823	-0.507054	0	1	0	...	0	0	0	0	0	0	0	0	0	1
2	0	-0.287869	1	0	0	2.017986	3.183766	0	0	1	...	0	0	0	0	0	0	1	0	0	0
3	0	-0.654281	0	0	0	-0.438823	-0.507054	0	1	0	...	0	0	0	0	0	0	0	0	1	0
4	0	-0.731583	0	0	0	3.000709	-0.507054	1	1	0	...	0	1	0	0	0	0	0	0	0	0

5 rows × 31 columns

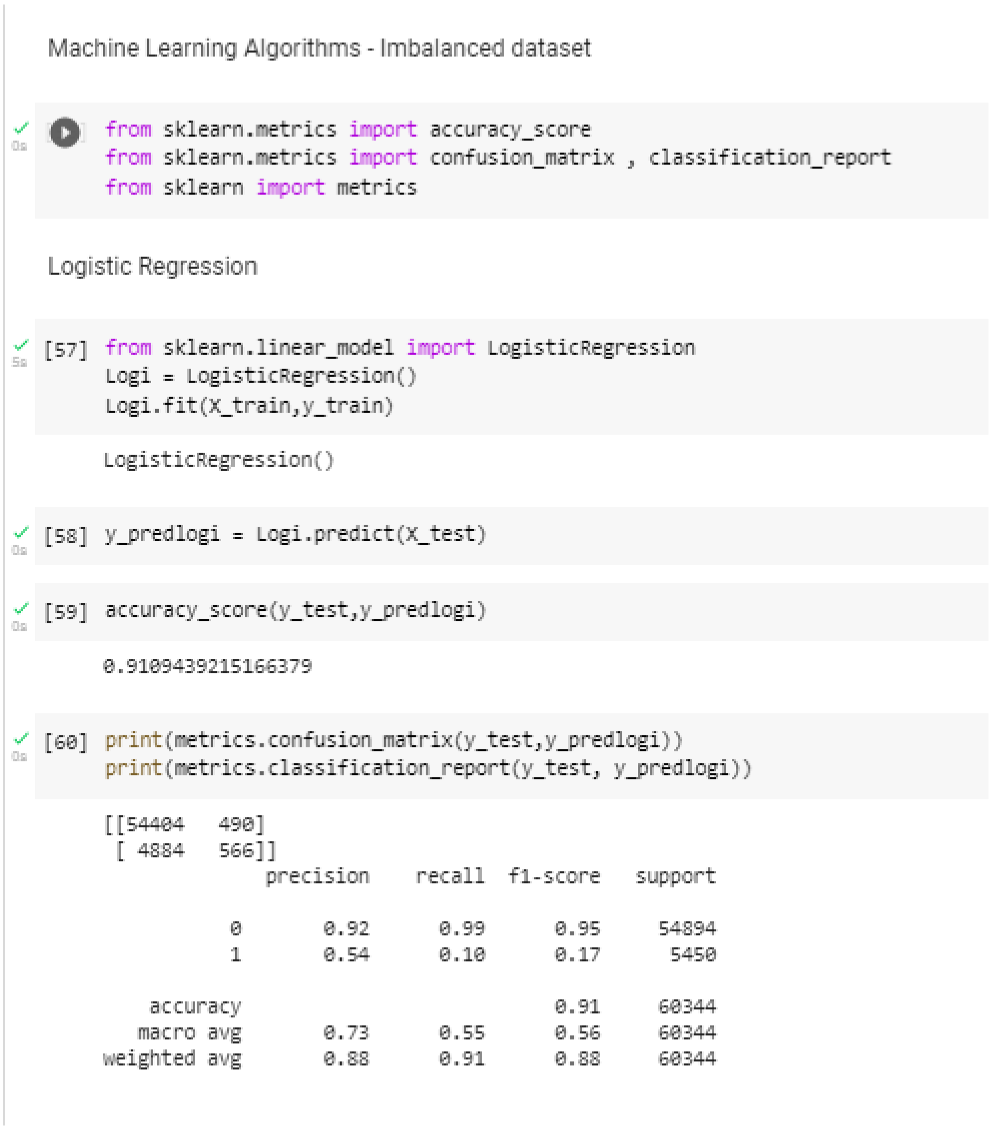
Test and Train data split



```
[48] X=resultset.drop('HeartDisease',axis=1)
     y = resultset['HeartDisease']
```



```
[51] from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```



Machine Learning Algorithms - Imbalanced dataset

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix , classification_report
from sklearn import metrics
```

Logistic Regression

```
[57] from sklearn.linear_model import LogisticRegression
     Logi = LogisticRegression()
     Logi.fit(X_train,y_train)
```

```
LogisticRegression()
```

```
[58] y_predlogi = Logi.predict(X_test)
```

```
[59] accuracy_score(y_test,y_predlogi)
```

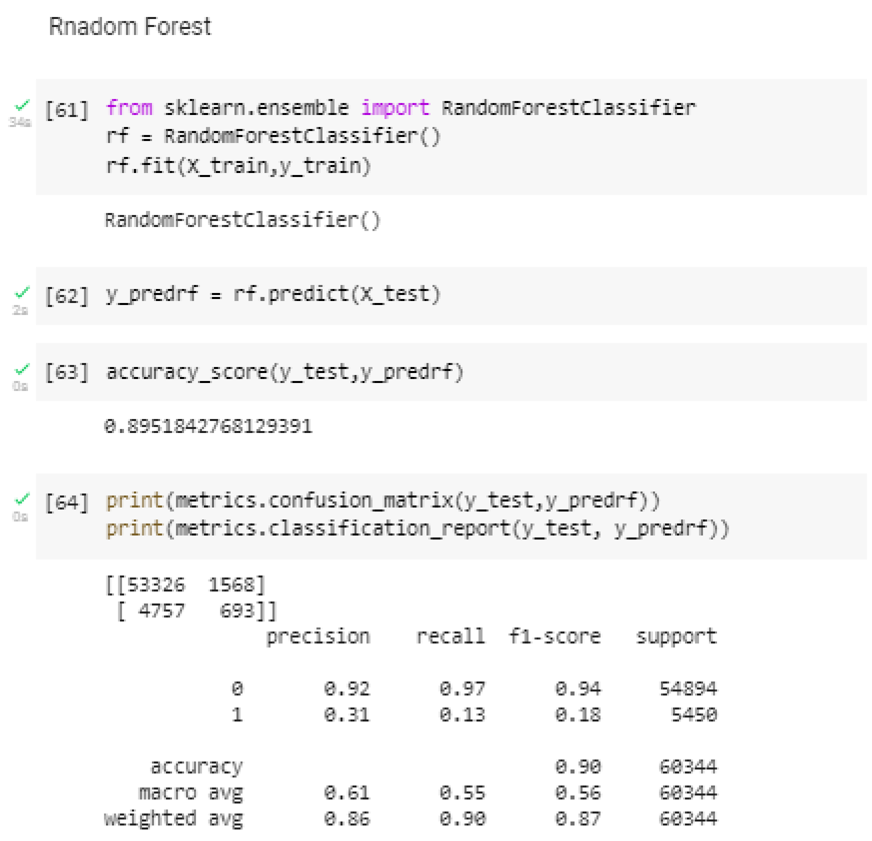
```
0.9109439215166379
```

```
[60] print(metrics.confusion_matrix(y_test,y_predlogi))
     print(metrics.classification_report(y_test, y_predlogi))
```

```
[[54404   490]
 [ 4884   566]]
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     54894
           1       0.54      0.10      0.17      5450

    accuracy                           0.91     60344
   macro avg       0.73      0.55      0.56     60344
weighted avg       0.88      0.91      0.88     60344
```



Rnadom Forest

```
[61] from sklearn.ensemble import RandomForestClassifier
     rf = RandomForestClassifier()
     rf.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
[62] y_predrf = rf.predict(X_test)
```

```
[63] accuracy_score(y_test,y_predrf)
```

```
0.8951842768129391
```

```
[64] print(metrics.confusion_matrix(y_test,y_predrf))
     print(metrics.classification_report(y_test, y_predrf))
```

```
[[53326  1568]
 [ 4757   693]]
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     54894
           1       0.31      0.13      0.18      5450

    accuracy                           0.90     60344
   macro avg       0.61      0.55      0.56     60344
weighted avg       0.86      0.90      0.87     60344
```

KNN

```
[65] from sklearn.neighbors import KNeighborsClassifier
     knei = KNeighborsClassifier()
     knei.fit(X_train,y_train)
```

```
KNeighborsClassifier()
```

```
[66] y_predknei = knei.predict(X_test)
```

```
[67] accuracy_score(y_test,y_predknei)
```

```
0.9013986477528835
```

```
[68] print(metrics.confusion_matrix(y_test,y_predknei))
     print(metrics.classification_report(y_test, y_predknei))
```

```
[[53794  1100]
 [ 4850   600]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     54894
           1       0.35      0.11      0.17      5450

    accuracy                           0.90     60344
   macro avg       0.64      0.55      0.56     60344
weighted avg       0.87      0.90      0.88     60344
```

Decision Tree

```
[69] from sklearn.tree import DecisionTreeClassifier
     dt = DecisionTreeClassifier()
     dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

```
[70] y_preddt = dt.predict(X_test)
```

```
[71] accuracy_score(y_test,y_preddt)
```

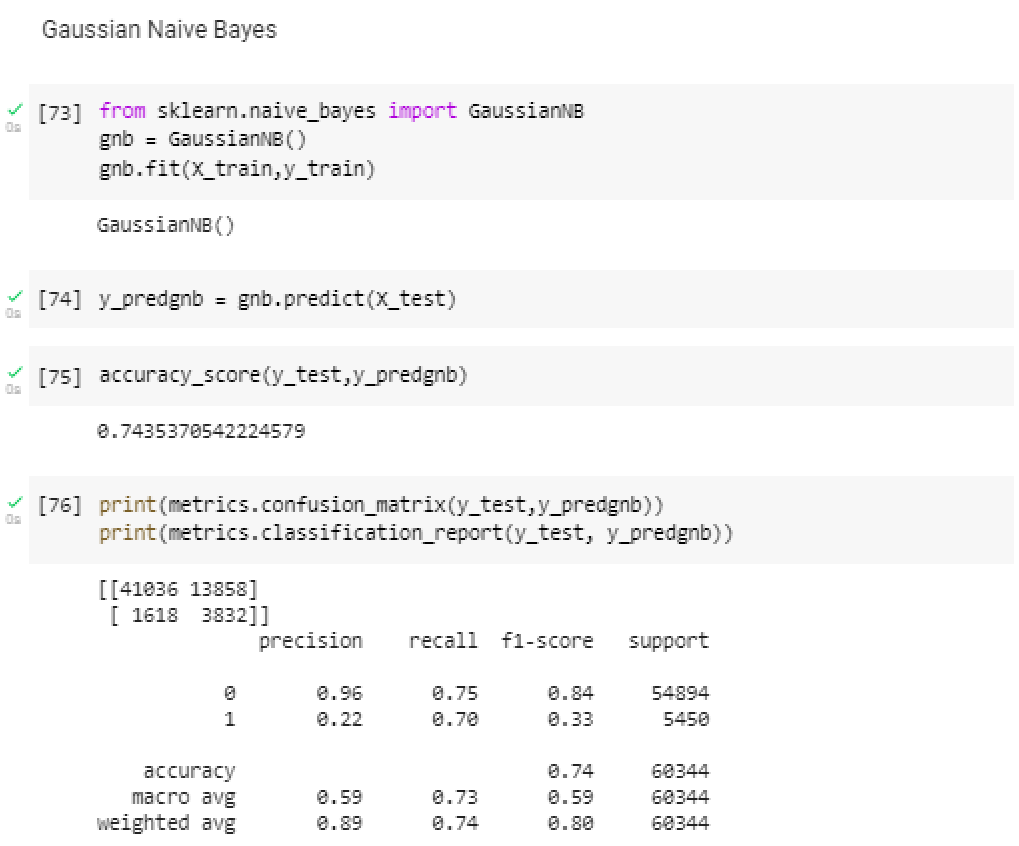
```
0.854136285297627
```

```
[72] print(metrics.confusion_matrix(y_test,y_preddt))
     print(metrics.classification_report(y_test, y_preddt))
```

```
[[50256  4638]
 [ 4164  1286]]
              precision    recall  f1-score   support

           0       0.92      0.92      0.92     54894
           1       0.22      0.24      0.23      5450

    accuracy                           0.85     60344
   macro avg       0.57      0.58      0.57     60344
weighted avg       0.86      0.85      0.86     60344
```



Gaussian Naive Bayes

```
[73] from sklearn.naive_bayes import GaussianNB
     gnb = GaussianNB()
     gnb.fit(X_train,y_train)
```

```
GaussianNB()
```

```
[74] y_predgnb = gnb.predict(X_test)
```

```
[75] accuracy_score(y_test,y_predgnb)
```

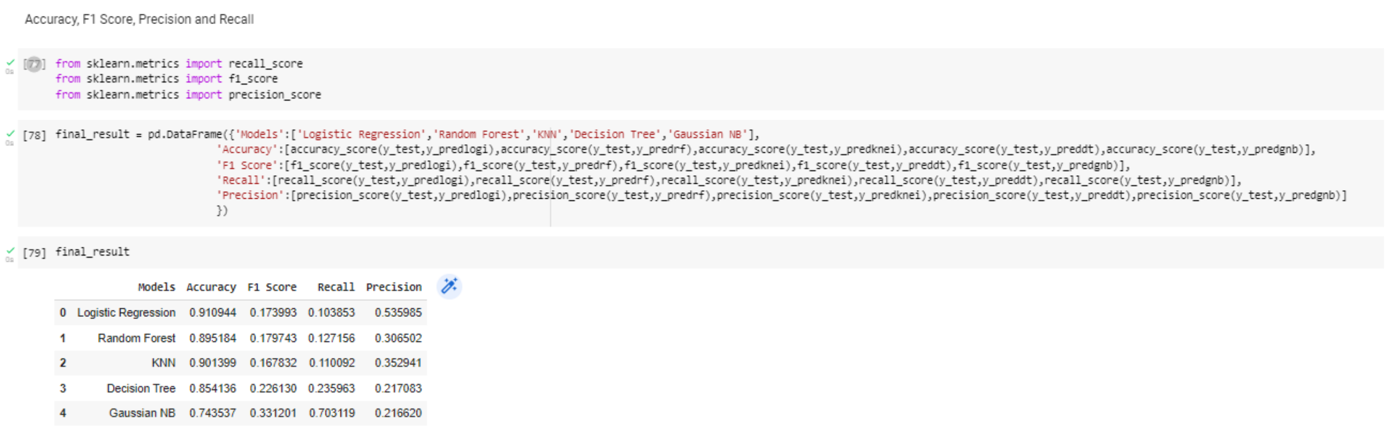
```
0.7435370542224579
```

```
[76] print(metrics.confusion_matrix(y_test,y_predgnb))
     print(metrics.classification_report(y_test, y_predgnb))
```

```
[[41036 13858]
 [ 1618  3832]]
              precision    recall  f1-score   support

           0       0.96      0.75      0.84     54894
           1       0.22      0.70      0.33      5450

    accuracy                           0.74     60344
   macro avg       0.59      0.73      0.59     60344
weighted avg       0.89      0.74      0.80     60344
```



Accuracy, F1 Score, Precision and Recall

```
[77] from sklearn.metrics import recall_score
     from sklearn.metrics import f1_score
     from sklearn.metrics import precision_score
```

```
[78] final_result = pd.DataFrame({'Models':['Logistic Regression','Random Forest','KNN','Decision Tree','Gaussian NB'],
                                  'Accuracy':[accuracy_score(y_test,y_predlogi),accuracy_score(y_test,y_predrf),accuracy_score(y_test,y_predknei),accuracy_score(y_test,y_preddt),accuracy_score(y_test,y_predgnb)],
                                  'F1 Score':[f1_score(y_test,y_predlogi),f1_score(y_test,y_predrf),f1_score(y_test,y_predknei),f1_score(y_test,y_preddt),f1_score(y_test,y_predgnb)],
                                  'Recall':[recall_score(y_test,y_predlogi),recall_score(y_test,y_predrf),recall_score(y_test,y_predknei),recall_score(y_test,y_preddt),recall_score(y_test,y_predgnb)],
                                  'Precision':[precision_score(y_test,y_predlogi),precision_score(y_test,y_predrf),precision_score(y_test,y_predknei),precision_score(y_test,y_preddt),precision_score(y_test,y_predgnb)]
                                  })
```

```
[79] final_result
```

	Models	Accuracy	F1 Score	Recall	Precision
0	Logistic Regression	0.910944	0.173993	0.103853	0.535985
1	Random Forest	0.895184	0.179743	0.127156	0.306502
2	KNN	0.901399	0.167832	0.110092	0.352941
3	Decision Tree	0.854136	0.226130	0.235963	0.217083
4	Gaussian NB	0.743537	0.331201	0.703119	0.216620

OverSampling using SMOTE

```
[80] from collections import Counter
     from imblearn.over_sampling import SMOTE
```

```
[81] smt = SMOTE()
```

```
[82] x_train_smt, y_train_smt = smt.fit_resample(X_train, y_train)
```

Logistic Regression

```
[83] y_predlogismt = Logi.predict(x_train_smt)
     accuracy_score(y_train_smt,y_predlogismt)
```

```
0.5225653801659668
```

Random Foreest

```
[84] y_predrfsmt = rf.predict(x_train_smt)
     accuracy_score(y_train_smt,y_predrfsmt)
```

```
0.7192342026398011
```

Decision Tree

```
[85] y_preddtsmt = dt.predict(x_train_smt)
     accuracy_score(y_train_smt,y_preddtsmt)
```

```
0.7401462912525847
```

Under Sampling

```
[86] from imblearn.under_sampling import RandomUnderSampler
```

```
[87] under_sampler = RandomUnderSampler(sampling_strategy=0.5)
     x_undersamp = resultset.iloc[:, 1:]
     y_undersamp = resultset['HeartDisease']
     x_under, y_under = under_sampler.fit_resample(x_undersamp, y_undersamp)

     print(f'Before undersampling: {Counter(resultset["HeartDisease"])}')
     print(f'After undersampling: {Counter(y_under)}')
```

```
Before undersampling: Counter({0: 274456, 1: 27261})
After undersampling: Counter({0: 54522, 1: 27261})
```

Logitic regression

```
[ ]  y_predlogiun = Logi.predict(x_undersamp)
     accuracy_score(y_undersamp,y_predlogiun)

     0.9112347000666187
```

Random forest

```
[ ]  y_predrfun = rf.predict(x_undersamp)
     accuracy_score(y_undersamp,y_predrfun)

     0.9758382855457266
```

Decision tree

```
[ ]  y_preddtun = dt.predict(x_undersamp)
     accuracy_score(y_undersamp,y_preddtun)

     0.9679633563902598
```

KNN

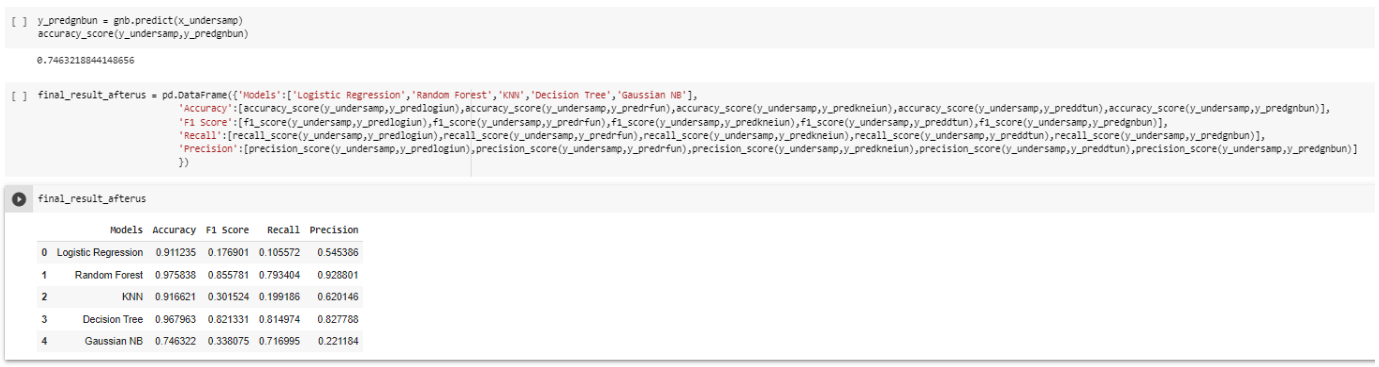
```
[ ]  y_predkneiun = knei.predict(x_undersamp)
     accuracy_score(y_undersamp,y_predkneiun)

     0.9166205417659595
```

Naive bayes

```
[ ]  y_predgnbun = gnb.predict(x_undersamp)
     accuracy_score(y_undersamp,y_predgnbun)

     0.7463218844148656
```



```
[ ]  y_predgnbun = gnb.predict(x_undersamp)
     accuracy_score(y_undersamp,y_predgnbun)

     0.7463218844148656
```

```
[ ]  final_result_afterus = pd.DataFrame({'Models':['Logistic Regression','Random Forest','KNN','Decision Tree','Gaussian NB'],
                          'Accuracy':[accuracy_score(y_undersamp,y_predlogiun),accuracy_score(y_undersamp,y_predrfun),accuracy_score(y_undersamp,y_predkneiun),accuracy_score(y_undersamp,y_preddtun),accuracy_score(y_undersamp,y_predgnbun)],
                          'F1 Score':[f1_score(y_undersamp,y_predlogiun),f1_score(y_undersamp,y_predrfun),f1_score(y_undersamp,y_predkneiun),f1_score(y_undersamp,y_preddtun),f1_score(y_undersamp,y_predgnbun)],
                          'Recall':[recall_score(y_undersamp,y_predlogiun),recall_score(y_undersamp,y_predrfun),recall_score(y_undersamp,y_predkneiun),recall_score(y_undersamp,y_preddtun),recall_score(y_undersamp,y_predgnbun)],
                          'Precision':[precision_score(y_undersamp,y_predlogiun),precision_score(y_undersamp,y_predrfun),precision_score(y_undersamp,y_predkneiun),precision_score(y_undersamp,y_preddtun),precision_score(y_undersamp,y_predgnbun)]
                          })
```

```
final_result_afterus
```

	Models	Accuracy	F1 Score	Recall	Precision
0	Logistic Regression	0.911235	0.176901	0.105572	0.545386
1	Random Forest	0.975838	0.855781	0.793404	0.928801
2	KNN	0.916621	0.301524	0.199186	0.620146
3	Decision Tree	0.967963	0.821331	0.814974	0.827788
4	Gaussian NB	0.746322	0.338075	0.716995	0.221184