DATS SCIENCE INTERNSHIP

Project: Heart Disease Prediction using Logistic Regression

Offered by InlighnTech

Table of Contents:

1. Problem Statement

Develop a machine learning model using Logistic Regression to predict the 10-year risk of Coronary Heart Disease (CHD) in patients based on health metrics.

Dataset: Framingham Heart Disease Dataset

2. Data Preprocessing

Loaded required Python libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

Loaded the dataset and checked its structure:

```
data = pd.read_csv('framingham.csv')
```

First 5 rows of dataset –

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

Last 5 rows of dataset –

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHI
4235	0	48	2.0	1	20.0	NaN	0	0	0	248.0	131.0	72.0	22.00	84.0	86.0	[illegible]
4236	0	44	1.0	1	15.0	0.0	0	0	0	210.0	126.5	87.0	19.16	86.0	NaN	[illegible]
4237	0	52	2.0	0	0.0	0.0	0	0	0	269.0	133.5	83.0	21.47	80.0	107.0	[illegible]
4238	1	40	3.0	0	0.0	0.0	0	1	0	185.0	141.0	98.0	25.60	67.0	72.0	[illegible]
4239	0	39	3.0	1	30.0	0.0	0	0	0	196.0	133.0	86.0	20.91	85.0	80.0	[illegible]

Shape of data:

```
data.shape
```

```
(4240, 16)
```

Data types in dataset:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4240 non-null   int64  
 1   age              4240 non-null   int64  
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64  
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64  
 7   prevalentHyp     4240 non-null   int64  
 8   diabetes         4240 non-null   int64  
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

As we know the dataset mostly comprises of numerical values but columns like ‘education’ , ‘cigsPerDay’ etc. have been designated at data type – float64. It implies there are NaN values present in the columns.

Checking null values:

```
data.isnull().sum()
```

```
male                  0
age                   0
education           105
currentSmoker         0
cigsPerDay           29
BPMeds               53
prevalentStroke       0
prevalentHyp          0
diabetes              0
totChol              50
sysBP                 0
diaBP                 0
BMI                  19
heartRate             1
glucose             388
TenYearCHD            0
dtype: int64
```

Renaming columns:

Columns have been renamed for better comprehension. Given below is a glimpse of the transformed dataset.

```
data = data.rename(columns = {'currentSmoker': 'current_smoker', 'cigsPerDay': 'cigs_per_day', 'BPMeds': 'BP_meds',
                              'prevalentStroke': 'prevalent_stroke', 'prevalentHyp' : 'prevalent_hyp',
                              'totChol': 'total_cholesterol', 'sysBP': 'sys_BP', 'diaBP':'dia_BP', 'heartRate':'heart_rate',
                              'TenYearCHD': 'ten_year_CHD'})
```

	male	age	education	current_smoker	cigs_per_day	BP_meds	prevalent_stroke	prevalent_hyp	diabetes	total_cholesterol	sys_BP	dia_BP	BMI	heart_rate	glucose
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0

Dropped columns –
'education' has been dropped since it does not have a significant impact on the cases of diabetes patients.

Scaling and splitting the dataset:

```
X = df.drop('ten_year_CHD', axis=1)
y = df['ten_year_CHD']
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split
```

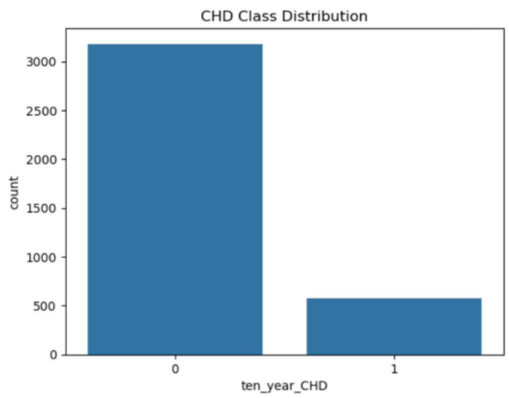
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42, stratify=y)
```

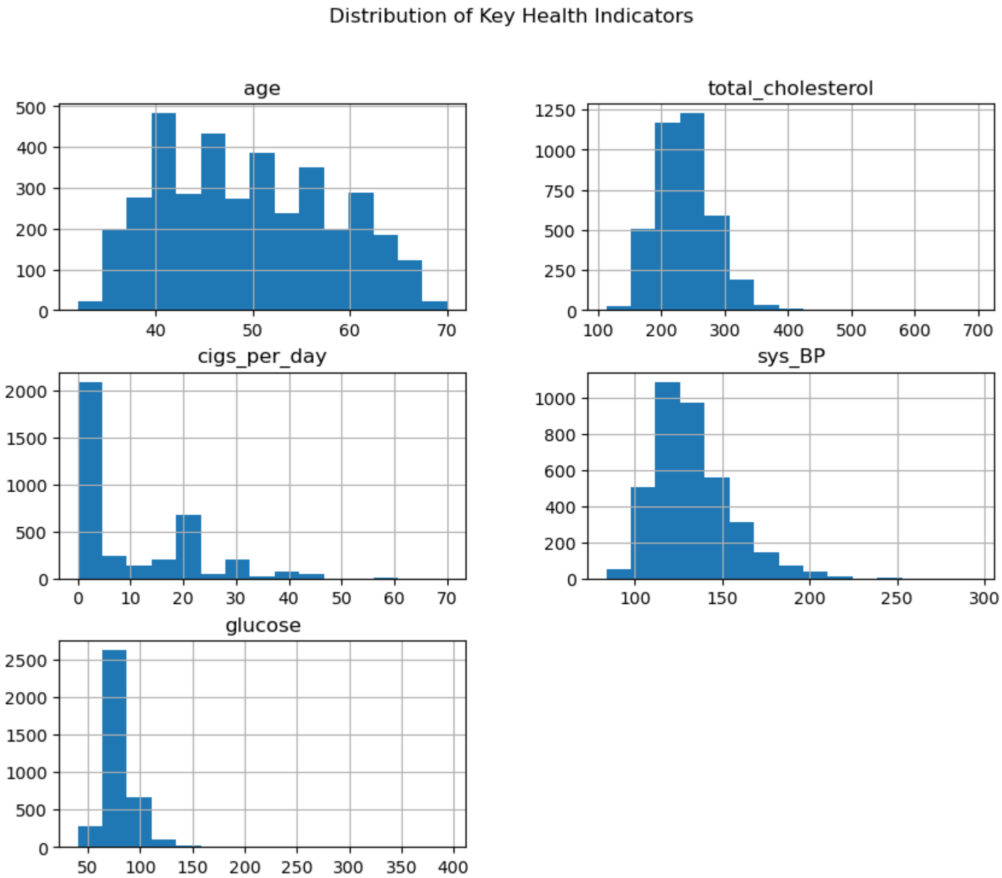
The dataset has been scaled using StandardScaler and split into train and test sets in a ration of 70:30.

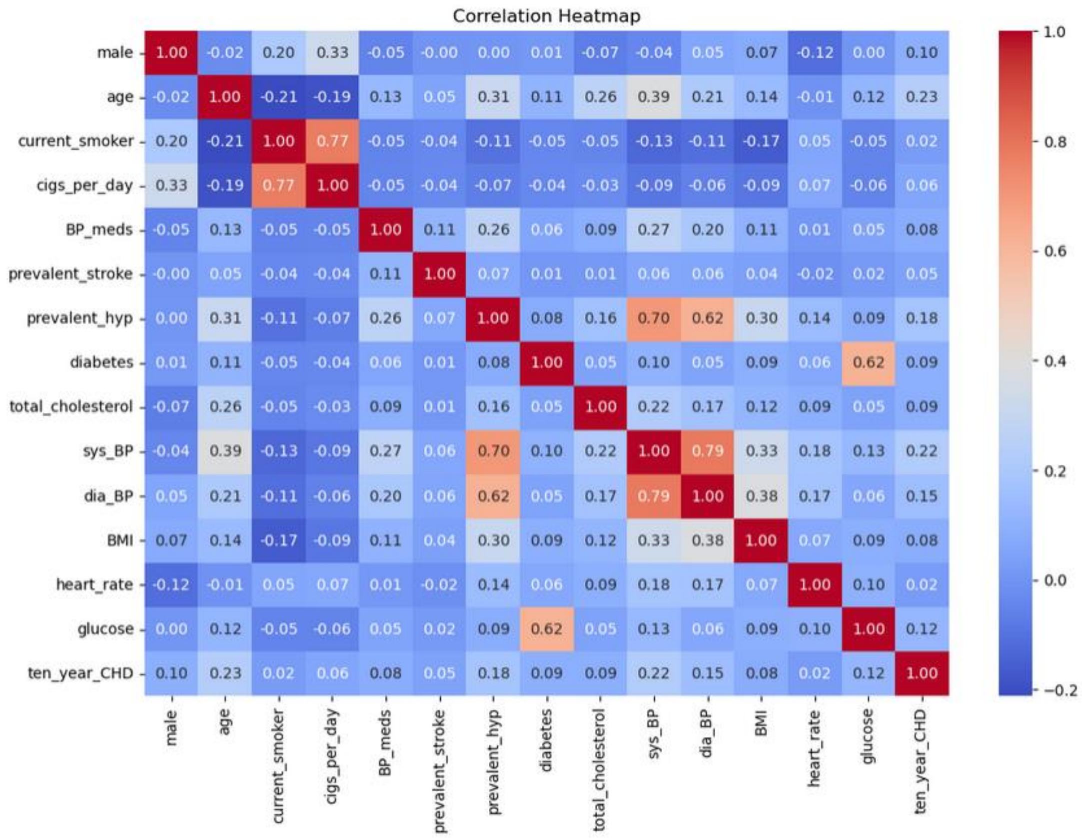
Also, to balance the target column variables, SMOTE method has been used for a better model.

3. Exploratory Data Analysis (EDA)

Distribution of CHD:




Distribution of Key Health Indicators
age
total_cholesterol
cigs_per_day
sys_BP
glucose


Correlation Heatmap
male 1.00 -0.02 0.20 0.33 -0.05 -0.00 0.00 0.01 -0.07 -0.04 0.05 0.07 -0.12 0.00 0.10
age -0.02 1.00 -0.21 -0.19 0.13 0.05 0.31 0.11 0.26 0.39 0.21 0.14 -0.01 0.12 0.23
current_smoker 0.20 -0.21 1.00 0.77 -0.05 -0.04 -0.11 -0.05 -0.05 -0.13 -0.11 -0.17 0.05 -0.05 0.02
cigs_per_day 0.33 -0.19 0.77 1.00 -0.05 -0.04 -0.07 -0.04 -0.03 -0.09 -0.06 -0.09 0.07 -0.06 0.06
BP_meds -0.05 0.13 -0.05 -0.05 1.00 0.11 0.26 0.06 0.09 0.27 0.20 0.11 0.01 0.05 0.08
prevalent_stroke -0.00 0.05 -0.04 -0.04 0.11 1.00 0.07 0.01 0.01 0.06 0.06 0.04 -0.02 0.02 0.05
prevalent_hyp 0.00 0.31 -0.11 -0.07 0.26 0.07 1.00 0.08 0.16 0.70 0.62 0.30 0.14 0.09 0.18
diabetes 0.01 0.11 -0.05 -0.04 0.06 0.01 0.08 1.00 0.05 0.10 0.05 0.09 0.06 0.62 0.09
total_cholesterol -0.07 0.26 -0.05 -0.03 0.09 0.01 0.16 0.05 1.00 0.22 0.17 0.12 0.09 0.05 0.09
sys_BP -0.04 0.39 -0.13 -0.09 0.27 0.06 0.70 0.10 0.22 1.00 0.79 0.33 0.18 0.13 0.22
dia_BP 0.05 0.21 -0.11 -0.06 0.20 0.06 0.62 0.05 0.17 0.79 1.00 0.38 0.17 0.06 0.15
BMI 0.07 0.14 -0.17 -0.09 0.11 0.04 0.30 0.09 0.12 0.33 0.38 1.00 0.07 0.09 0.08
heart_rate -0.12 -0.01 0.05 0.07 0.01 -0.02 0.14 0.06 0.09 0.18 0.17 0.07 1.00 0.10 0.02
glucose 0.00 0.12 -0.05 -0.06 0.05 0.02 0.09 0.62 0.05 0.13 0.06 0.09 0.10 1.00 0.12
ten_year_CHD 0.10 0.23 0.02 0.06 0.08 0.05 0.18 0.09 0.09 0.22 0.15 0.08 0.02 0.12 1.00
male
age
current_smoker
cigs_per_day
BP_meds
prevalent_stroke
prevalent_hyp
diabetes
total_cholesterol
sys_BP
dia_BP
BMI
heart_rate
glucose
ten_year_CHD

We can some strong relationships between columns like cigarettes per day and current smoker status and between blood pressure and hypertension status for diabetes.

4. Model Training using Logistic Regression:

The train dataset has been trained using logistic regression from sklearn.linear_model.

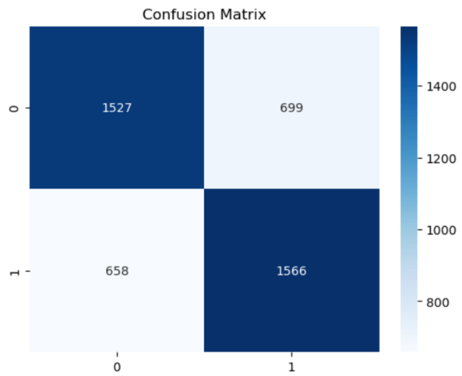
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

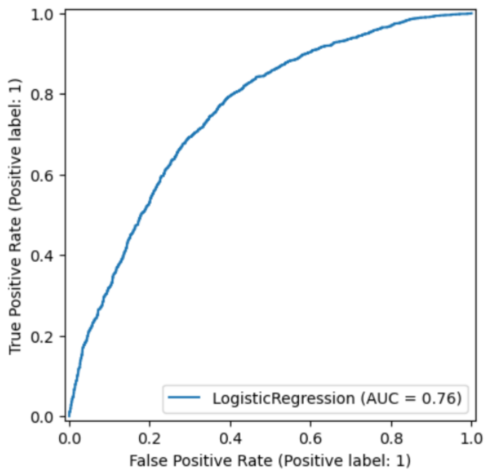
The evaluation metrics are as follows –

```
Accuracy: 0.6950561797752809
Precision: 0.6913907284768211
Recall: 0.704136690647482
F1 Score: 0.697705502339051
ROC-AUC: 0.7550743098243777
```

Confusion matrix for the training set -



ROC curve for training set-

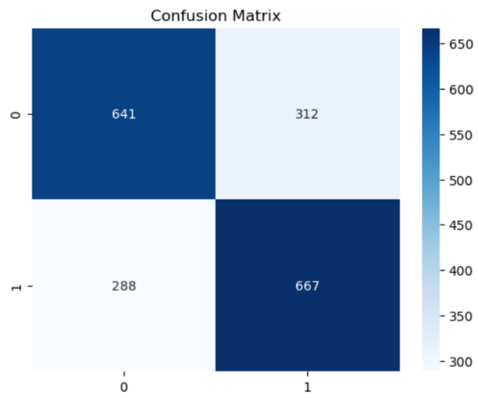


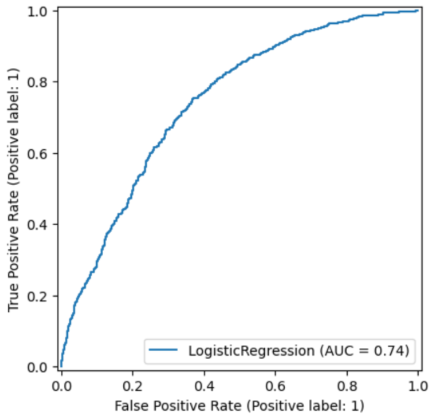
5. Model Evaluation and Prediction:

The evaluation metrics of logistic regression on the test set are as follows –

```
Accuracy: 0.6855345911949685
Precision: 0.6813074565883555
Recall: 0.6984293193717277
F1 Score: 0.6897621509824199
ROC-AUC: 0.743409349367937
```

Confusion matrix for test dataset - ROC curve for test dataset –





As we can see from the evaluation metrics, there are overfitting or underfitting issues but the important metrics like accuracy, recall, f1-score and precision have lesser values. To combat this, we will build models using the XGBoost classifier.

6. <u>Results with XGBoost Classifier</u>

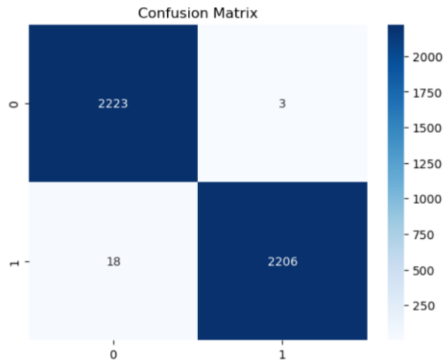
After applying the XGBoost classifier method on the train and test datasets the metrics are as follows:

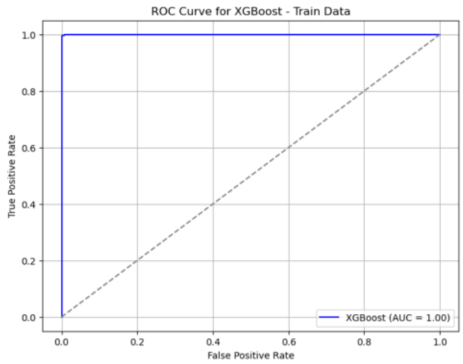
For train set-

```
XGBoost ROC AUC Score: 0.9999402095574215
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2226
           1       1.00      0.99      1.00      2224

    accuracy                           1.00      4450
   macro avg       1.00      1.00      1.00      4450
weighted avg       1.00      1.00      1.00      4450
```

We have a 100% success rate in predicting class 1 i.e., the patients with a heart problem.

The confusion matrix and ROC curve are given below.





For test set-

```
XGBoost ROC AUC Score: 0.9381440806930992
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       953
           1       0.90      0.85      0.87       955

    accuracy                           0.87      1908
   macro avg       0.88      0.87      0.87      1908
weighted avg       0.88      0.87      0.87      1908
```

We can see that the model is working well on the test set as well – 85% recall for class 1. Although it is not foolproof given that prediction rate is lesser indicating a bit of overfitting – the prediction rate is more than that of the linear regression model.

7. <u>Conclusions:</u>

Although there is a bit of overfitting issues in the XGBoost model, the logistic regression model has lesser prediction rates despite having no overfitting/underfitting issues.

We deleted some rows with null values. The model could fair better if those data are provided.

Otherwise, the XGBoost is ready to be used on new datasets – not enclosed with the project files.