Machine learning approach for predicting heart disease

Mitisha Tasnim Rahman
ID:20301327
Department of CSE
Brac University
Dhaka, Bangladesh
mitisha.tasnim.rahman@g.bracu.ac.bd

Rubaiya Islam
ID:20201188
Department of CSE
Brac University
Dhaka, Bangladesh
rubaiya.islam2@g.bracu.ac.bd

***Abstract*—This paper investigates machine-learning based heart disease prediction. In this project we utilized a complete dataset obtained from a reliable health database,which we will use to train our machine-learning algorithms so that when we train these algorithms it can predict heart disease by given features (such as age, sex, chest pain type, etc.)**

***Keywords*—classification, logistic regression, machine-learning, decision tree, normalization, standardization.**

I. Introduction

Heart disease continues to be the world's largest cause of mortality, highlighting the criticality of early detection in the context of successful medical care. Machine learning (ML) techniques are becoming more and more common in healthcare research because they provide accurate cardiac disease prediction by examining risk variables and trends. With the help of data from a reputable health database, our project seeks to create an accurate machine learning-based prediction model that could enhance clinical outcomes and patient care. The need for ML-based predictive modeling in healthcare is rising due to the volume of health-related data and electronic health records. The project's identification of people at high risk of heart disease could have a substantial impact on clinical practice, public health initiatives, and policy-making. We examine several project phases, including data processing, correlation analysis, model training, and evaluation, complete with a thorough visual representation of performance parameters like accuracy, precision, recall, and F1 score. Our ultimate objective is to improve knowledge and implementation of ML techniques for heart disease prediction by offering insightful information to researchers and healthcare practitioners.

II. Dataset description

A. Source

The original data came from the Cleveland database from UCI Machine Learning Repository.
Link: https://archive.ics.uci.edu/dataset/45/heart+disease
However, we've downloaded it in a formatted way from Kaggle.
Link:
https://www.kaggle.com/datasets/sumaiyatasmeem/heart-disease-classification-dataset

B. Dataset Description

Approximately 14 of the 76 attributes in the original dataset will be used in this project. Features or attributes are the variables that will be utilized to predict the desired result. In this case, the dependent variable represents whether the patient has heart disease, though the independent variables cover a range of their medical parameters. The following are the features we'll use to predict our target variable (heart disease or no heart disease).

1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
● 0: Typical angina: chest pain related decrease blood supply to the heart
● 1: Atypical angina: chest pain not related to heart
● 2: Non-anginal pain: typically esophageal spasms (non heart related)
● 3: Asymptomatic: chest pain not showing signs of disease
4. trestbps - resting blood pressure (in mm Hg on admission to the hospital)
● anything above 130-140 is typically cause for concern
5. chol - serum cholestoral in mg/dl
● serum = LDL + HDL + .2 * triglycerides
● above 200 is cause for concern
6. fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
● '>126' mg/dL signals diabetes
7. restecg - resting electrocardiographic results
● 0: Nothing to note
● 1: ST-T Wave abnormality
- ○ can range from mild symptoms to severe problems
- ○ signals non-normal heart beat

● 2: Possible or definite left ventricular hypertrophy
- ○ Enlarged heart's main pumping chamber

8. thalach - maximum heart rate achieved
9. exang - exercise induced angina (1 = yes; 0 = no)
10. oldpeak - ST depression induced by exercise relative to rest
● looks at stress of heart during exercise
● unhealthy heart will stress more
11. slope - the slope of the peak exercise ST segment
● 0: Upsloping: better heart rate with exercise (uncommon)
● 1: Flat Sloping: minimal change (typical healthy heart)
● 2: Downsloping: signs of unhealthy heart

12. ca - number of major vessels (0-3) colored by fluoroscopy

● colored vessel means the doctor can see the blood passing through

● the more blood movement the better (no clots)

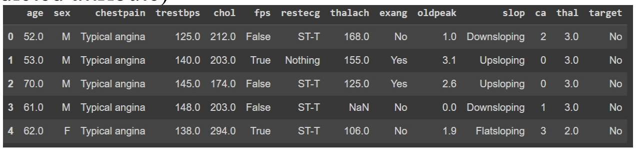
13. thal - thalium stress result

● 1,3: normal

● 6: fixed defect: used to be defect but ok now

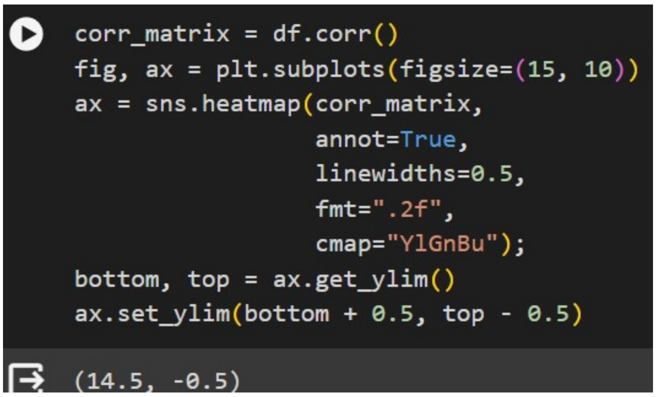
● 7: reversable defect: no proper blood movement when excercising

14. target - have disease or not (1=yes, 0=no) (= the predicted attribute)



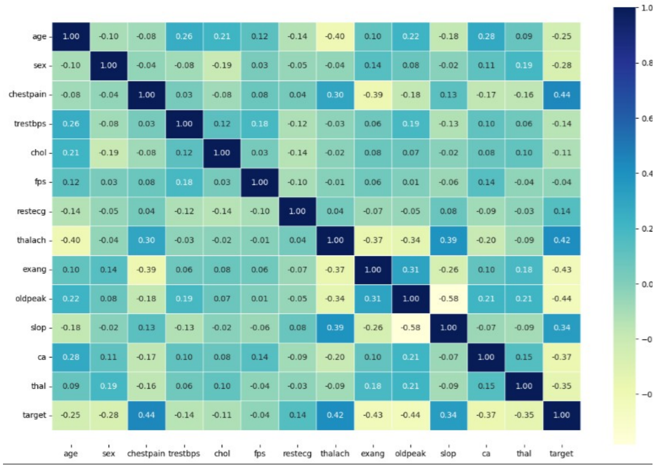
	age	sex	chestpain	trestbps	chol	fps	restecg	thalach	exang	oldpeak	slop	ca	thal	target
0	52.0	M	Typical angina	125.0	212.0	False	ST-T	168.0	No	1.0	Downsloping	2	3.0	No
1	53.0	M	Typical angina	140.0	203.0	True	Nothing	155.0	Yes	3.1	Upsloping	0	3.0	No
2	70.0	M	Typical angina	145.0	174.0	False	ST-T	125.0	Yes	2.6	Upsloping	0	3.0	No
3	61.0	M	Typical angina	148.0	203.0	False	ST-T	NaN	No	0.0	Downsloping	1	3.0	No
4	62.0	F	Typical angina	138.0	294.0	True	ST-T	106.0	No	1.9	Flatsloping	3	2.0	No

C. Correlation



```
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15, 10))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

```
(14.5, -0.5)
```



	age	sex	chestpain	trestbps	chol	fps	restecg	thalach	exang	oldpeak	slop	ca	thal	target
age	1.00	-0.10	-0.08	0.26	0.21	0.12	-0.14	-0.40	0.10	0.22	-0.18	0.28	0.09	-0.25
sex	-0.10	1.00	-0.04	-0.08	-0.19	0.03	-0.05	-0.04	0.14	0.08	-0.02	0.11	0.19	-0.28
chestpain	-0.08	-0.04	1.00	0.03	-0.08	0.08	0.04	0.30	-0.39	-0.18	0.13	-0.17	-0.16	0.44
trestbps	0.26	-0.08	0.03	1.00	0.12	0.18	-0.12	-0.03	0.06	0.19	-0.13	0.10	0.06	-0.14
chol	0.21	-0.19	-0.08	0.12	1.00	0.03	-0.14	-0.02	0.08	0.07	-0.02	0.08	0.10	-0.11
fps	0.12	0.03	0.08	0.18	0.03	1.00	-0.10	-0.01	0.06	0.01	-0.06	0.14	-0.04	-0.04
restecg	-0.14	-0.05	0.04	-0.12	-0.14	-0.10	1.00	0.04	-0.07	-0.05	0.08	-0.09	-0.03	0.14
thalach	-0.40	-0.04	0.30	-0.03	-0.02	-0.01	0.04	1.00	-0.37	-0.34	0.39	-0.20	-0.09	0.42
exang	0.10	0.14	-0.39	0.06	0.08	0.06	-0.07	-0.37	1.00	0.31	-0.26	0.10	0.18	-0.43
oldpeak	0.22	0.08	-0.18	0.19	0.07	0.01	-0.05	-0.34	0.31	1.00	-0.58	0.21	0.21	-0.44
slop	-0.18	-0.02	0.13	-0.13	-0.02	-0.06	0.08	0.39	-0.26	-0.58	1.00	-0.07	-0.09	0.34
ca	0.28	0.11	-0.17	0.10	0.08	0.14	-0.09	-0.20	0.10	0.21	-0.07	1.00	0.15	-0.37
thal	0.09	0.19	-0.16	0.06	0.10	-0.04	-0.03	-0.09	0.18	0.21	-0.09	0.15	1.00	-0.35
target	-0.25	-0.28	0.44	-0.14	-0.11	-0.04	0.14	0.42	-0.43	-0.44	0.34	-0.37	-0.35	1.00

D. Data Description

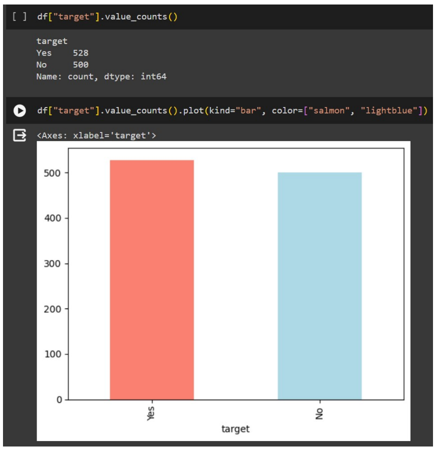
Our problem is a classification problem. Determining if a person has heart disease is a classification challenge, as it requires assigning discrete class labels to each instance in the dataset. In this instance, the classifications are "has heart disease" and "does not have." The goal is to create a predictive model that uses supplied information (e.g. age, gender, chest pain kind) to classify new cases into one of two categories based on learnt patterns and correlations in data. Classification algorithms are specifically developed for categorical output.

There are 1028 data points total, both category and analytical.

Quantitative: chol, bp, age, and more.

Classified: gender, kind of chest discomfort, etc.

E. Data distribution



```
[ ] df["target"].value_counts()
```

```
target
Yes    528
No     500
Name: count, dtype: int64
```

```
df["target"].value_counts().plot(kind="bar", color=["salmon", "lightblue"])
```

```
<Axes: xlabel='target'>
```

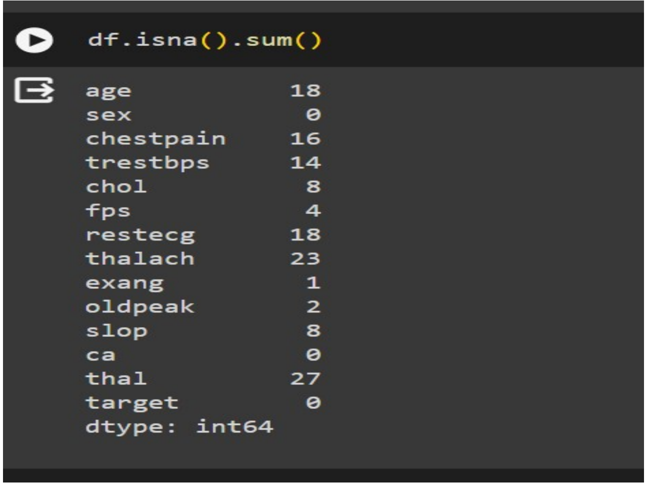
III. PRE-PROCESSING TECHNIQUES

A. Libraries Tools

We are integrating several powerful libraries into our project. NumPy and Pandas are essential tools for data analysis, providing diverse data structures such as DataFrames, which are similar to Excel spreadsheets. Pandas makes it simple to import and process our CSV-formatted data. Scikit-Learn provides us with a comprehensive set of learning algorithms and predictive analytics utilities. For example, we use Scikit-Learn's StandardScaler() to standardize our dataset, which improves model performance.Matplotlib makes it easier to visualize our data graphically. With its simple interface, we can generate smart visualizations like histograms and scatter plots to acquire a better understanding of our dataset.

B. Checking for Null Values

It can be seen that in our dataset total null value in each column is Age-18, Chestpain-16, trestbps-14, chol-8, fps-4, restecg-18, thalach-23,exang-1, oldpeak-2, slop-8, thal-27.

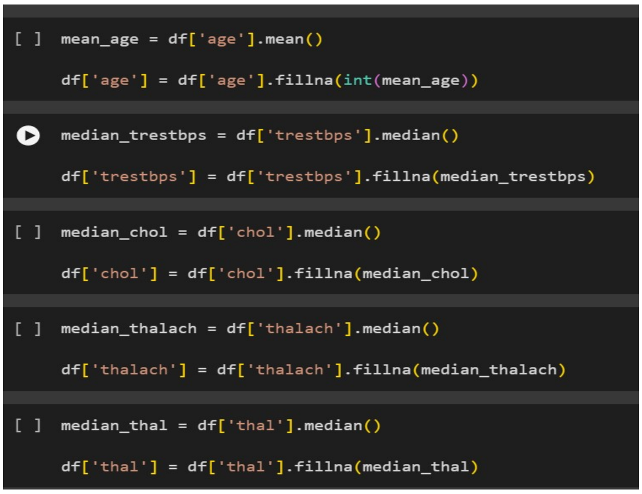


```
df.isna().sum()
```

```
age          18
sex           0
chestpain    16
trestbps     14
chol          8
fps           4
restecg      18
thalach      23
exang         1
oldpeak       2
slop          8
ca            0
thal         27
target        0
dtype: int64
```

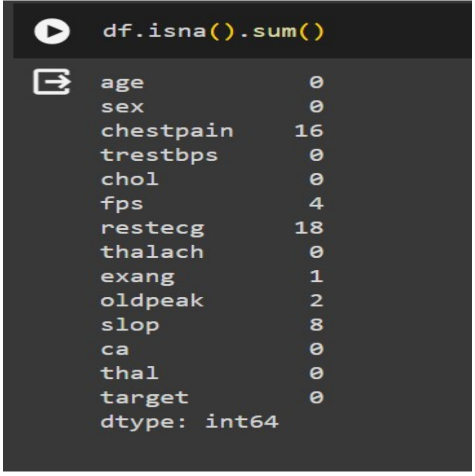
C. Imputing Null Values

We did imputation for all the null values in each column of age, tresbps, chol, thalach and thal with the use of Mean and Median values.



```
[ ]  mean_age = df['age'].mean()

     df['age'] = df['age'].fillna(int(mean_age))

▶    median_trestbps = df['trestbps'].median()

     df['trestbps'] = df['trestbps'].fillna(median_trestbps)

[ ]  median_chol = df['chol'].median()

     df['chol'] = df['chol'].fillna(median_chol)

[ ]  median_thalach = df['thalach'].median()

     df['thalach'] = df['thalach'].fillna(median_thalach)

[ ]  median_thal = df['thal'].median()

     df['thal'] = df['thal'].fillna(median_thal)
```

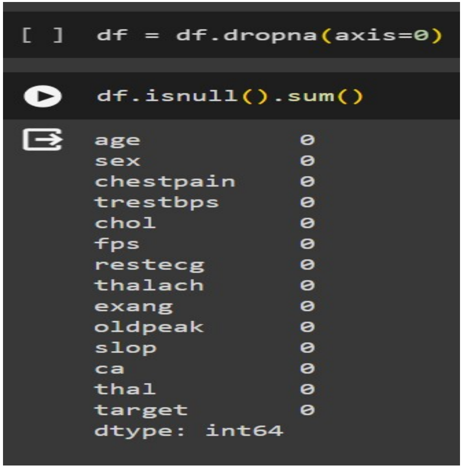
After imputing we can see there is no null value in age, tresbps, chol, thalach and thal.



```
▶  df.isna().sum()

   age            0
   sex            0
   chestpain     16
   trestbps       0
   chol           0
   fps            4
   restecg       18
   thalach        0
   exang          1
   oldpeak        2
   slop           8
   ca             0
   thal           0
   target         0
   dtype: int64
```

D. Removing row/column

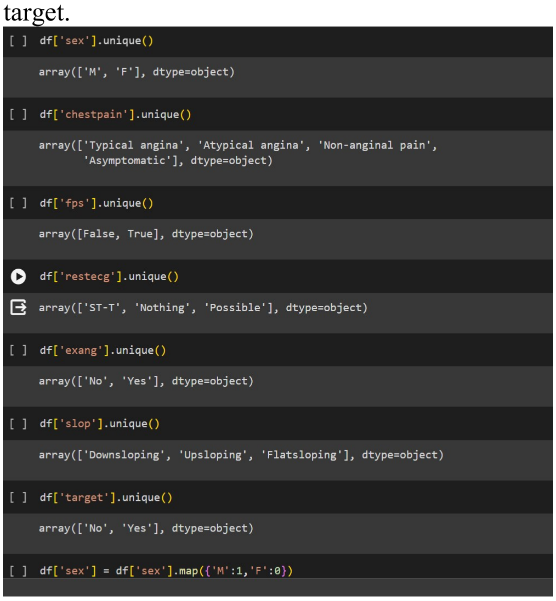
We also used Removal of null values by discarding or deleting the row of each null value containing instances. Thus, all the null values in our dataset were tackled.



```
[ ]  df = df.dropna(axis=0)

▶    df.isnull().sum()

     age            0
     sex            0
     chestpain      0
     trestbps       0
     chol           0
     fps            0
     restecg        0
     thalach        0
     exang          0
     oldpeak        0
     slop           0
     ca             0
     thal           0
     target         0
     dtype: int64
```

E. Encoding

we have encoded sex, chestpain, fps, restecg, exang, slope, target.



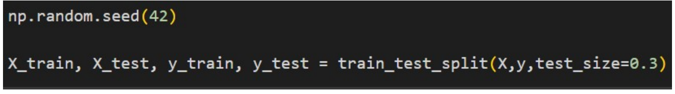
```
[ ]  df['sex'].unique()

     array(['M', 'F'], dtype=object)

[ ]  df['chestpain'].unique()

     array(['Typical angina', 'Atypical angina', 'Non-anginal pain',
            'Asymptomatic'], dtype=object)

[ ]  df['fps'].unique()

     array([False, True], dtype=object)

▶    df['restecg'].unique()

     array(['ST-T', 'Nothing', 'Possible'], dtype=object)

[ ]  df['exang'].unique()

     array(['No', 'Yes'], dtype=object)

[ ]  df['slop'].unique()

     array(['Downsloping', 'Upsloping', 'Flatsloping'], dtype=object)

[ ]  df['target'].unique()

     array(['No', 'Yes'], dtype=object)

[ ]  df['sex'] = df['sex'].map({'M':1,'F':0})
```

IV. Feature Scalling

Feature scaling is a common machine learning technique that normalizes or standardizes the range of independent variables or features in a dataset. The goal is to ensure that the model's predictions are equally influenced by each feature. This preprocessing method is called scaling, and is often used to fit the features of a data set into a specific range. Large-scale features in machine learning algorithms can lead to biased models. Feature scaling converts features to a common scale, usually between 0 and 1 or -1 and 1. This helps avoid this issue.Common ways for feature scaling are normalization and standardization . Normalization scales data between 0 and 1, whereas standardization scales it with a mean of 0 and standard deviation of 1.Feature scaling improves the model's accuracy and ability to generalize to new data.

V. Dataset Spliting

In the field of machine learning, it is mandatory to split a data set into a training set and a testing set to evaluate the performance of a model on unseen data. In this paper, we have splitted the dataset and used 30% data for testing and 70% data for training from the whole dataset to perform the accuracy prediction of multiple machine learning algorithms.



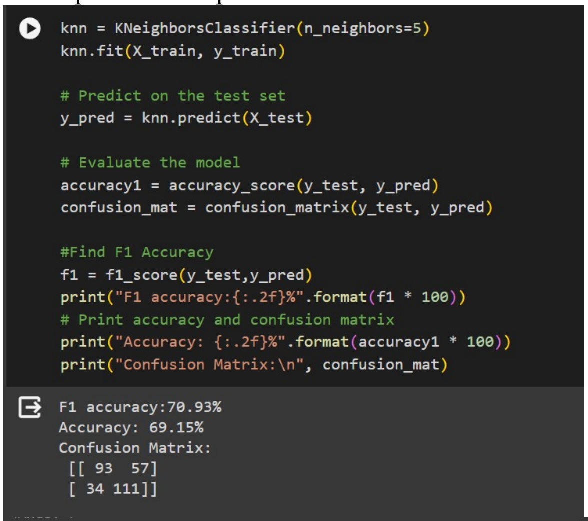
```
np.random.seed(42)

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

VI. Model Training and Testing

We are using three machine learning models to evaluate the accuracy of our heart attack risk prediction research. Our project models include KNN, Decision Tree, and Logistic Regression.

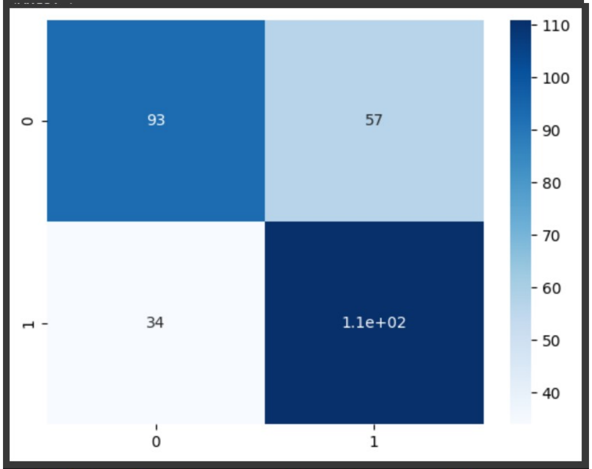
A. KNN

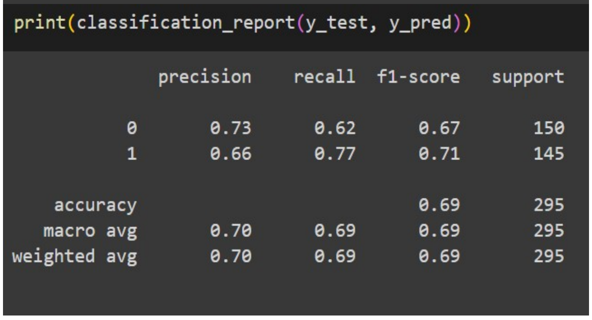
KNN is a supervised machine learning method suitable for classification and regression tasks. KNN predicts new data points based on the k closest points in the training set, with "closest" specified by a distance metric. In a classification task, KNN finds the k closest data points in the training set and classifies the new point based on the majority of those neighbors. Tuning the hyperparameter k can improve model performance.



```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy1 = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Print accuracy and confusion matrix
print("Accuracy: {:.2f}%".format(accuracy1 * 100))
print("Confusion Matrix:\n", confusion_mat)
```

```
F1 accuracy:70.93%
Accuracy: 69.15%
Confusion Matrix:
 [[ 93  57]
 [ 34 111]]
```



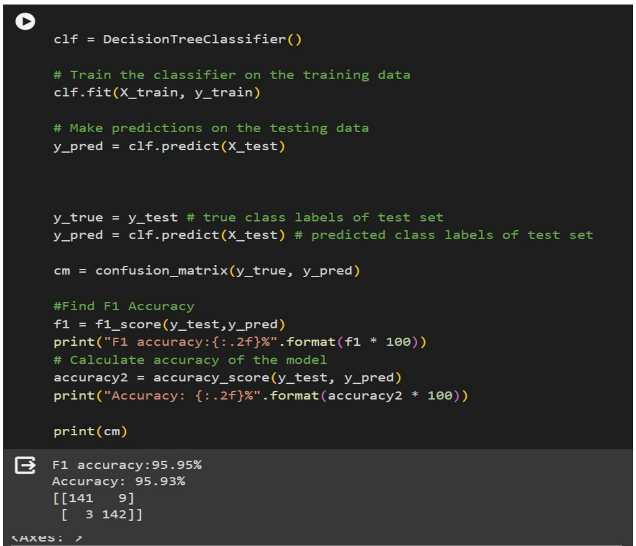


```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.62      0.67       150
           1       0.66      0.77      0.71       145

    accuracy                           0.69       295
   macro avg       0.70      0.69      0.69       295
weighted avg       0.70      0.69      0.69       295
```

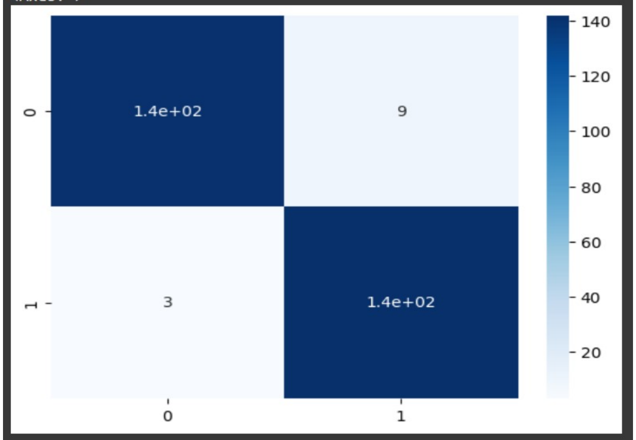
B. Decision Tree

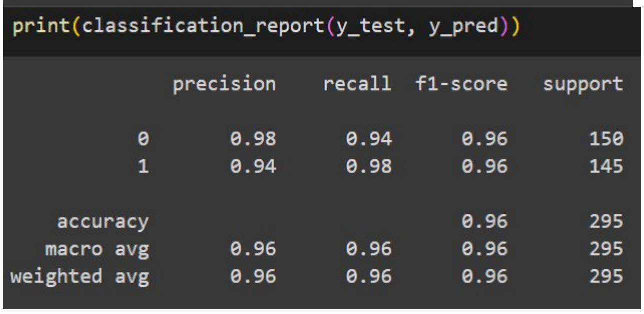
A decision tree is a common supervised machine learning technique used for classification and regression. The algorithm partitions the feature space recursively based on input feature values, ultimately determining the output variable's class or value. The decision tree algorithm begins at the root node, which represents the full dataset, and then recursively divides the data into smaller subgroups based on the values of specific features. At each split, the algorithm selects the feature that best separates the data into the purest possible subsets (those with the least amount of impurity or heterogeneity).This procedure is performed for each subset until a stopping requirement is reached, such as when the tree reaches its maximum depth when the number of samples in a leaf node falls below a specific threshold.To anticipate new data, design a decision tree and traverse it from root to leaf nodes depending on input feature values. The leaf node's class or value corresponds to the predicted output variable.



```
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)



y_true = y_test # true class labels of test set
y_pred = clf.predict(X_test) # predicted class labels of test set

cm = confusion_matrix(y_true, y_pred)

#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Calculate accuracy of the model
accuracy2 = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy2 * 100))

print(cm)
```

```
F1 accuracy:95.95%
Accuracy: 95.93%
[[141   9]
 [  3 142]]
<Axes: >
```



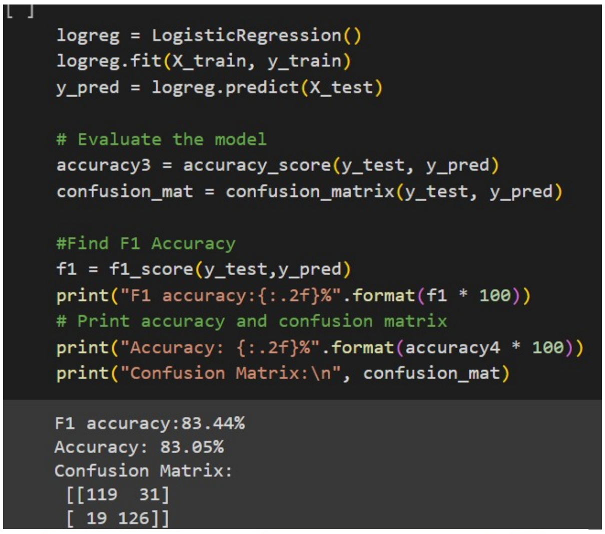


```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       150
           1       0.94      0.98      0.96       145

    accuracy                           0.96       295
   macro avg       0.96      0.96      0.96       295
weighted avg       0.96      0.96      0.96       295
```

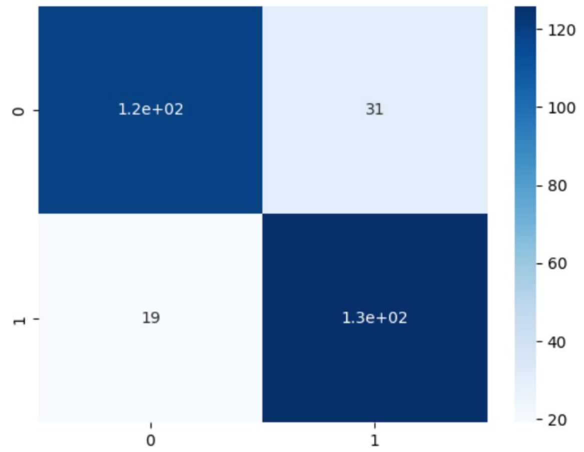
C. Logistic Regression

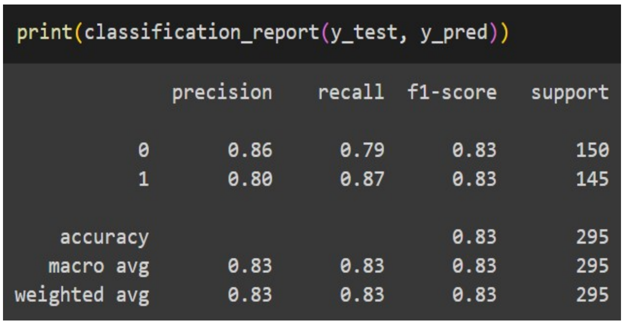
Logistic regression is a statistical approach for binary classification, predicting whether an input data point belongs to one of two groups. This supervised learning algorithm is widely used in machine learning and statistics. Logistic regression involves representing input data points with characteristics that might be continuous, categorical, or binary. The result is a binary response variable, often 0 or 1, that represents the data point's class membership. The purpose of logistic regression is to estimate the parameters of a logistic function that describes the connection between input features and binary response variables. The logistic function, or sigmoid function, relates the input features into a probability value between 0 and 1. The logistic function's parameters are calculated using maximum likelihood estimation, which involves training the model on a labeled dataset with known class labels. Once trained, the logistic regression model may be used to predict new, previously unknown data points by computing the likelihood that the input data point belongs to the positive class based on its feature values. A threshold can be set to transform these probabilities into binary class labels.



```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

# Evaluate the model
accuracy3 = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Print accuracy and confusion matrix
print("Accuracy: {:.2f}%".format(accuracy4 * 100))
print("Confusion Matrix:\n", confusion_mat)
```

```
F1 accuracy:83.44%
Accuracy: 83.05%
Confusion Matrix:
 [[119  31]
 [ 19 126]]
```

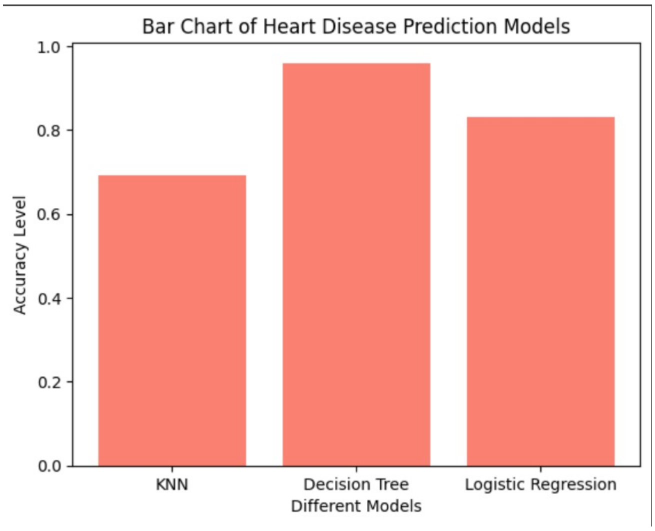




```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.79      0.83       150
           1       0.80      0.87      0.83       145

    accuracy                           0.83       295
   macro avg       0.83      0.83      0.83       295
weighted avg       0.83      0.83      0.83       295
```

VII. MODEL SELECTION/COMPARISON ANALYSIS



Model Name	F1 Score	Accuracy Rate
KNN	70.93%	69.15%
Decision tree	95.95%	95.93%
Logistic regression	83.44%	83.05%

The decision tree model has the highest accuracy, with a total accuracy of 95.93% and an F1 score of 95.95%. Its exceptional performance is largely due to its ability to handle both mathematical and categorical information and to capture complex connections. With an accuracy rate of 83.05% and an F1 score of 83.44%, logistic regression comes second in a close race, showing decent performance but falling short of the decision tree's maximum. On the other hand, when compared with the decision tree and logistic regression analyses, the KNN model's efficiency is lower, with an F1 score of 70.93% and an accuracy of 69.15%, suggesting its limits.The decision tree is a good fit for this type of problem because of its skill at handling unpredictable connections and feature combinations.

VIII. CONCLUSION

In this work, we developed and evaluated machine learning (ML) models that use a wide range of medical and demographic information to predict the risk of heart disease. The study's findings highlight the ability of machine learning methods to accurately project heart disease, providing important information for early identification and treatment. Performance measures such as F1 score, accuracy, precision, and recall confirm how well these prediction models identify those who are more likely to develop heart disease. Our findings have consequences for experts, leaders, and healthcare professionals. Detailed predictions have the potential to assist medical professionals in identifying high-risk patients and implementing appropriate preventative interventions, including lifestyle modifications, medication, and continued care, thereby reducing the impact of heart disease.Furthermore, by giving those at greater risk priority for additional evaluation and therapy, these models have the potential to optimize the deployment of resources in healthcare delivery and promote more effective and economical healthcare practices.

Additionally, our work adds to the rising library of knowledge about the use of machine learning (ML) techniques in healthcare, specifically in the prediction of heart disease. The results add to the body of data supporting the use of ML algorithms to project the risk of cardiovascular illness and provide information on how they could improve the outcomes of patients. It's also critical to look at how understandable and clarified ML models are for healthcare decisions.

In conclusion, this project highlights the possible use of machine learning techniques for the fast evaluation and identification of heart diseases. The results of the project have implications for public health efforts, planning, and medical practice that advance our understanding of heart disease outlook. More research in this area has the opportunity to improve patient outcomes, reduce healthcare expenses, and eventually save lives.

REFERENCE

1) https://www.w3schools.com/python/pandas/pandas_intro.asp#:~:text=Pandas%20is%20a%20Python%20library,by%20Wes%20McKinney%20in%202008

2) https://www.w3schools.com/python/numpy/numpy_intro.asp

3) https://scikit-learn.org/stable/

4) https://seaborn.pydata.org/