CSE422 Project Report

Heart Disease Prediction

Section: 06

Submitted by:

Sartaj Emon Prattoy (20301326)

Jannatul Ferdoush (20301291)

Md. Imamul Mursalin Sujoy (20301120)

Md. Asif Rahman (20301122)

Submitted to:

A.M. Esfar-E-Alam

Syed Zamil Hasan

Date: 28.04.2023

Table of Contents

Introduction

Heart disease, commonly referred to as cardiovascular disease, continues to be a major cause of death across the globe. Early diagnosis of heart disease is essential for the purpose of providing appropriate medical support. In healthcare research machine learning (ML) methods have recently become more popular. Moreover, with the use of Machine learning prediction of heart disease can be done very efficiently. By identifying the risk factors, patterns and other data analysis, Machine learning can forecast the likelihood of a person developing heart disease. Our aim in this project is to create an accurate prediction model with the use of Machine learning techniques for the early diagnosis of heart disease. The dataset that we collected comes from a renowned health database and by using this kind of dataset in machine learning an accurate and dependable heart disease prediction model can be created. This can dramatically improve patient care and other clinical outcomes. There is a growing need for efficient ML-based methods to assess and use these data for predictive modeling in healthcare as there's a vast accessibility of electronic health records and other sources of health related data. The results of this project have the potential to have a considerable impact on clinical practice, public health strategies, and policy-making as it can provide insightful information for identifying individuals with high risk of getting heart disease. In this project starting from the technique for data processing, comparing and finding correlation between each risk factor and target, training and evaluating the model, as well as the performance metrics of the employed ML models, including accuracy, precision, recall, and F1 score all are explored with visualization detail. All these works come down to our motivation with this project, which is to add to the expanding body of knowledge in this area and offer valuable productive information for healthcare professionals and researchers by offering a thorough examination of ML approaches for heart disease prediction.

Dataset description

Source

The original data came from the Cleveland database from UCI Machine Learning Repository.

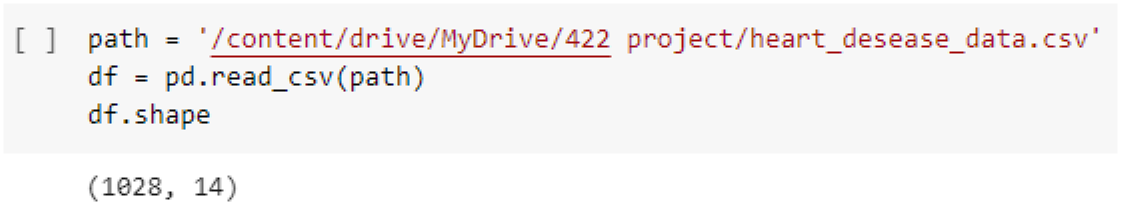
Link: https://archive.ics.uci.edu/ml/datasets/heart+Disease

However, we've downloaded it in a formatted way from Kaggle.

Link: https://www.kaggle.com/datasets/sumaiyatasmeem/heart-disease-classification-dataset

Accessing dataset

The original database contains 76 attributes, but here only 14 attributes will be used. Attributes (also called features) are the variables that we'll use to predict our target variable. In our case, the independent variables are a patient's different medical attributes and the dependent variable is whether or not they have heart disease.



```
[ ]  path = '/content/drive/MyDrive/422 project/heart_desease_data.csv'
     df = pd.read_csv(path)
     df.shape

     (1028, 14)
```

The following are the features we'll use to predict our target variable (heart disease or no heart disease).

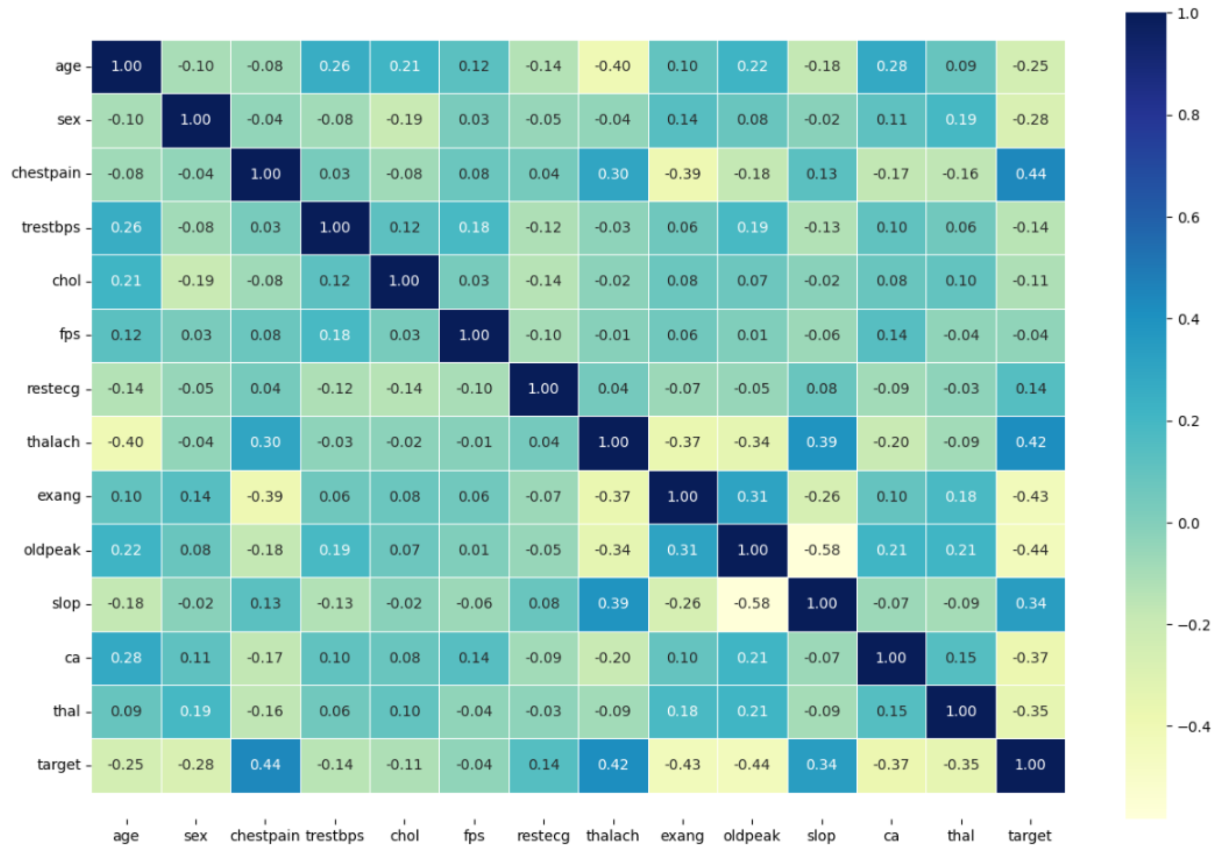
1. age - age in years
2. sex - (1 = male; 0 = female)
3. cp - chest pain type
    - 0: Typical angina: chest pain related decrease blood supply to the heart
    - 1: Atypical angina: chest pain not related to heart
    - 2: Non-anginal pain: typically esophageal spasms (non heart related)
    - 3: Asymptomatic: 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)

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)
```



	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

Data description

Our problem is a classification problem. The problem of determining whether a person has heart disease or not is a classification problem because the task involves assigning discrete class labels to each instance in the dataset. In this case, the classes are "has heart disease" and "does not have heart disease." The goal is to build a predictive model that can learn from the given features (such as age, sex, chest pain type, etc.) and classify new instances into one of these two classes based on the learned patterns and relationships in the data. Classification algorithms are specifically designed to handle this type of problem where the output is categorical in nature.

Number of data points: 1028, both quantitative and categorical.

Quantitative: age, bp, chol etc.

Categorical: sex, chestpaintype etc.

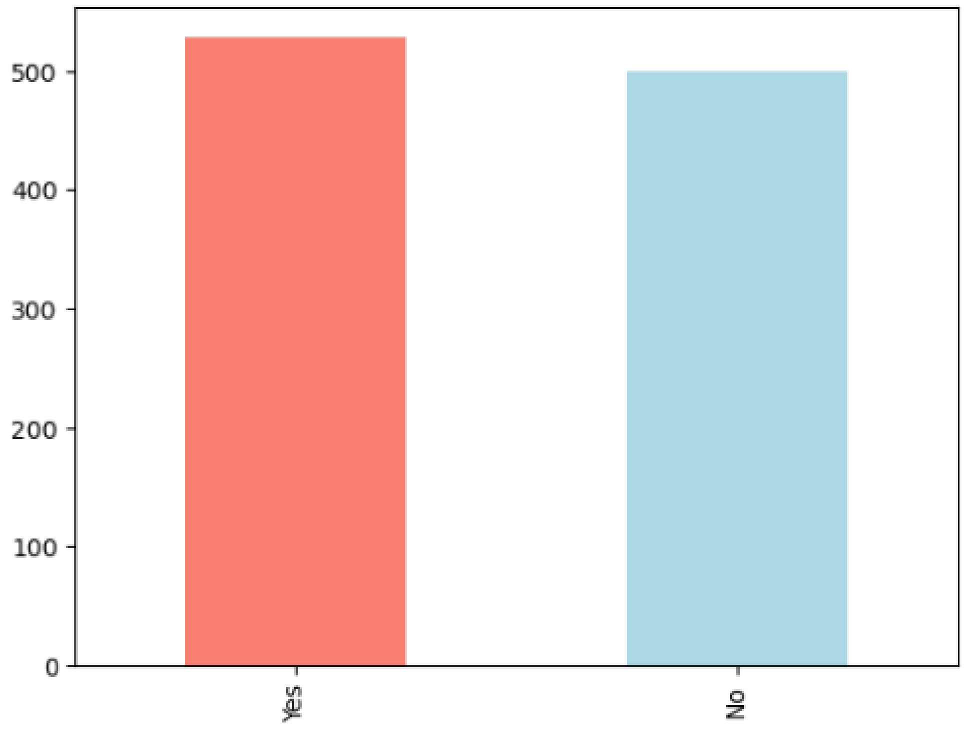
Data distribution

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

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

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

```
<Axes: >
```

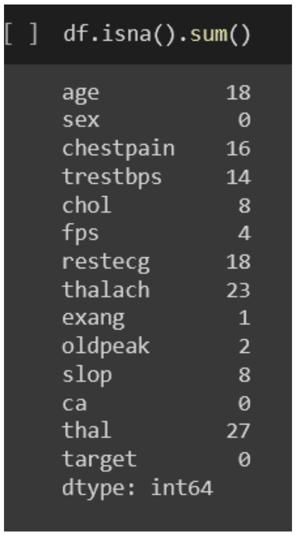


Dataset pre-processing

Fault

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.

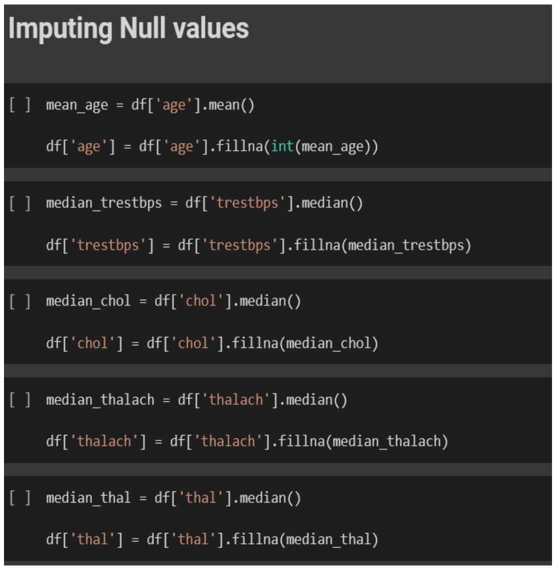


```
[ ]  df.isna().sum()

     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
```

Solution

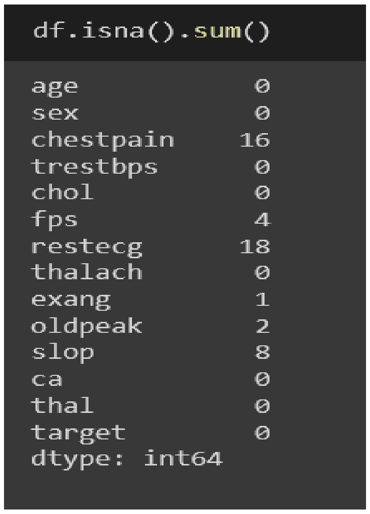
Imputing Null values

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



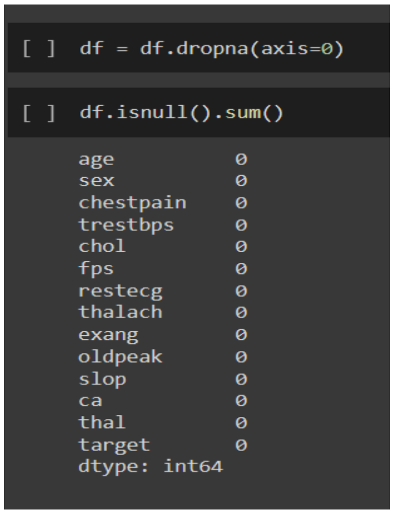
Imputing Null values

```
[ ]  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)
```



```
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
```

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



```
[ ]  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
```

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.

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

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

```
df['chestpain'] = df['chestpain'].map({'Typical angina':0, 'Atypical angina':1, 'Non-anginal pain':2,
       'Asymptomatic':3})
```

```
df['fps'] = df['fps'].astype(int)
```

```
df['restecg'] = df['restecg'].map({'Nothing':0,'ST-T':1,'Possible':2})
```

```
df['exang'] = df['exang'].map({'Yes':1,'No':0})
```

```
df['slop'] = df['slop'].map({'Upsloping':0,'Downsloping':2,'Flatsloping':1})
```

```
df['target'] = df['target'].map({'Yes':1,'No':0})
```

Feature scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

In many machine learning algorithms, features with larger scales may dominate the learning process, resulting in biased models. Feature scaling helps to avoid this issue by transforming the features into a common scale, typically between 0 and 1, or -1 and 1.Common methods of feature scaling include normalization and standardization. Normalization scales the data between 0 and 1, while standardization scales the data to have a mean of 0 and standard deviation of 1.By using feature scaling, the model can make more accurate predictions and generalize better to new, unseen data.

Dataset splitting

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.

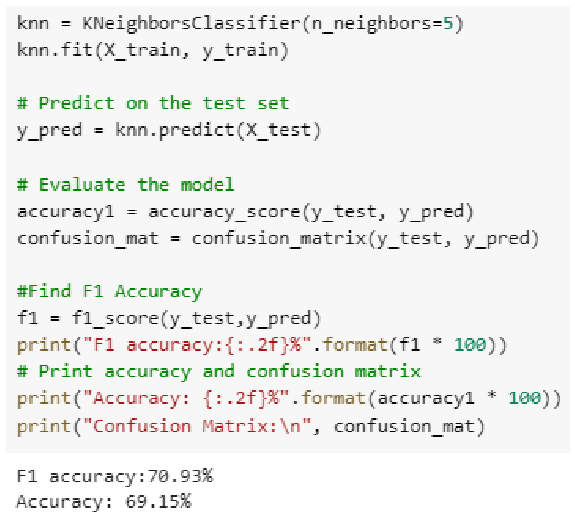
```
np.random.seed(42)

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

Model training & testing

The main idea behind the proposed system architecture was to create a heart disease prediction system based on the inputs datasets. For predicting the accuracy of heart attack, this study analyzed the classification algorithms namely KNN, Decision Tree, SVM and Linear Regression.

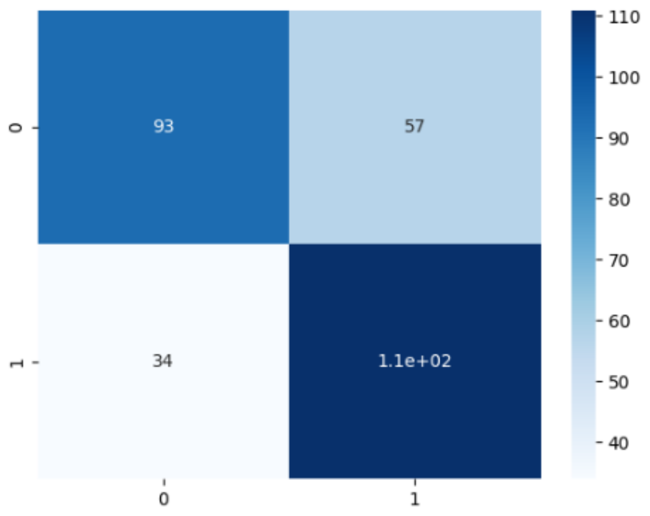
KNN

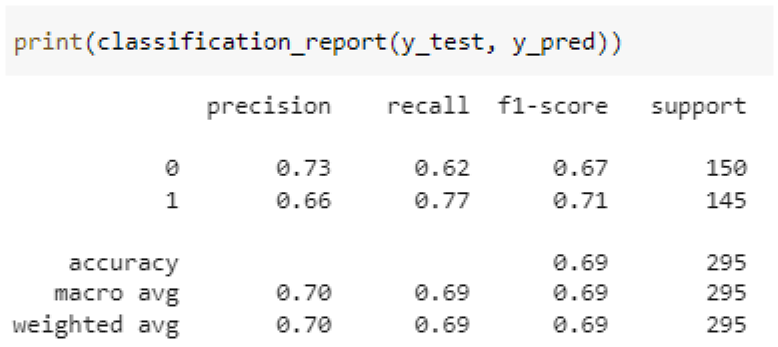
KNN (k-Nearest Neighbors) is a supervised machine learning algorithm that can be used for both classification and regression tasks. In KNN, the prediction for a new data point is based on the k closest points in the training set, where "closest" is defined by some distance metric. For example, in a classification task, given a new data point, KNN would find the k closest data points in the training set and classify the new point based on the majority class among those kneighbors. The choice of k is a hyperparameter that can be tuned to optimize the model's 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%
```



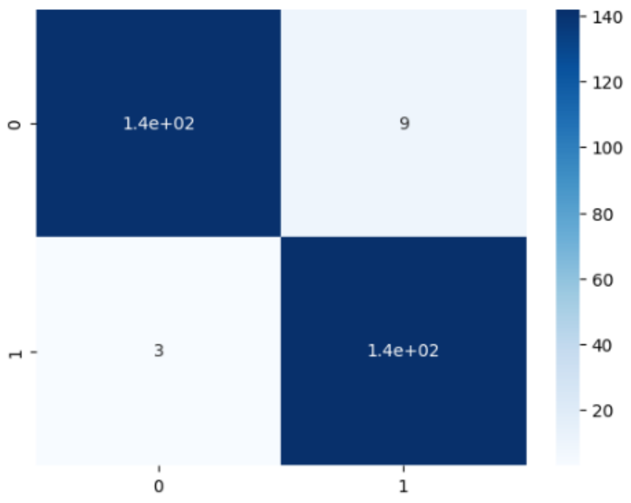


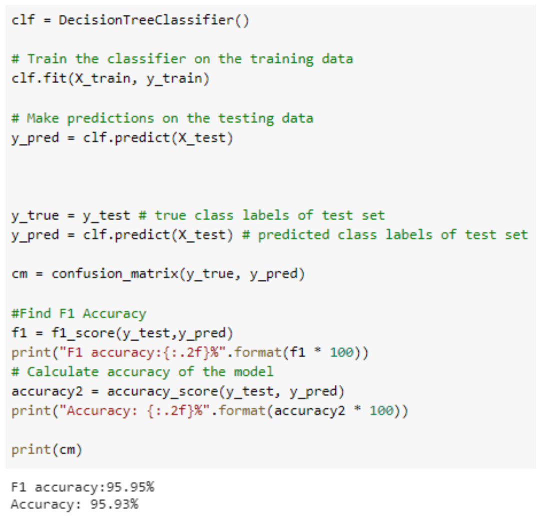
```
print(classification_report(y_test, y_pred))
```

```
              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
```

Decision Tree

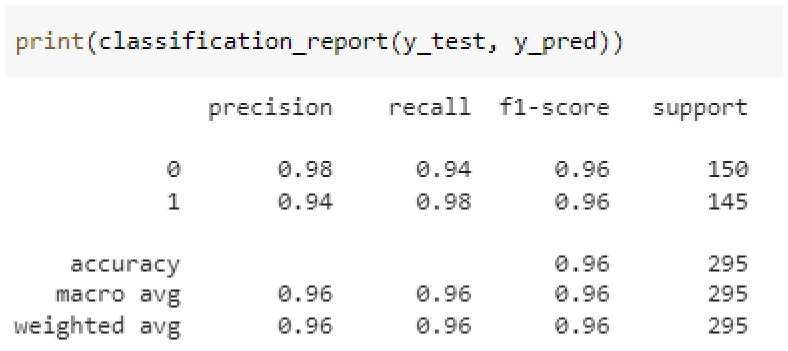
A decision tree is a popular supervised machine learning algorithm used for both classification and regression tasks. It works by recursively partitioning the feature space into smaller and smaller regions, based on the values of the input features, until a decision can be made about the class or value of the output variable. The decision tree algorithm starts at the root node, which represents the entire dataset, and recursively splits the data into smaller subsets based on the values of a particular feature. At each split, the algorithm chooses the feature that best separates the data into the purest possible subsets (i.e., with the least amount of impurity or heterogeneity). This process is repeated for each subset until a stopping criterion is met, such as when the tree reaches a maximum depth or the number of samples in a leaf node falls below a certain threshold.Once the decision tree has been constructed, it can be used to make predictions on new, unseen data by traversing the tree from the root node to a leaf node, based on the values of the input features. The class or value associated with the leaf node represents 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%
```

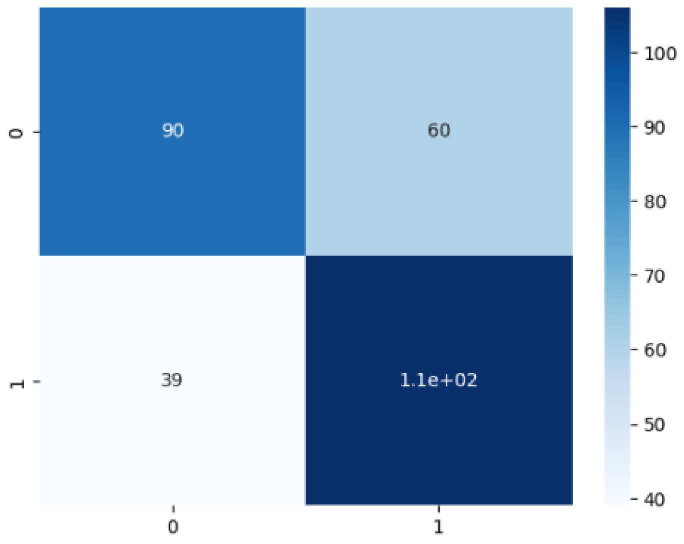


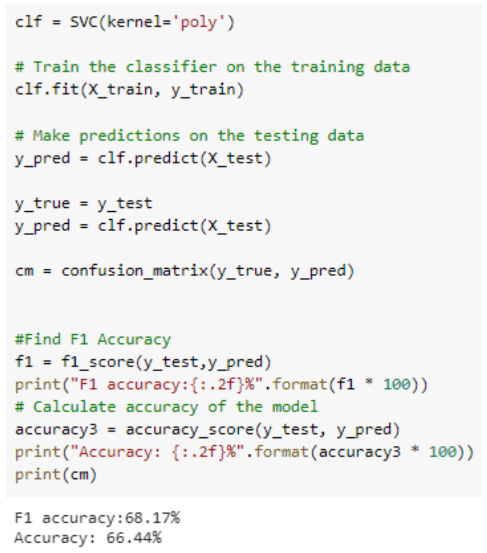
```
print(classification_report(y_test, y_pred))
```

```
              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
```

SVM

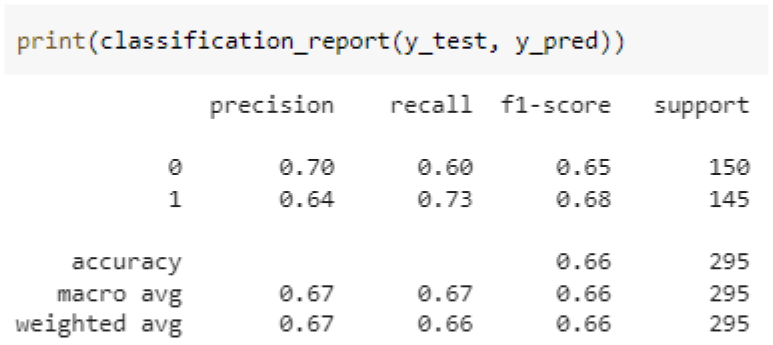
SVM is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiate the two classes very well. SVM is very effective in high dimensional spaces. Also, it is effective in cases where the number of dimensions is greater than the number of samples. Moreover, it uses a subset of training points in the decision function as a result it is also memory efficient. It has also some versatility like different Kernel functions can be 9 specified for the decision function. There are also some common kernels provided, but it is also possible to specify custom kernels.





```
clf = SVC(kernel='poly')

# 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
y_pred = clf.predict(X_test)

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
accuracy3 = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy3 * 100))
print(cm)
```

```
F1 accuracy:68.17%
Accuracy: 66.44%
```

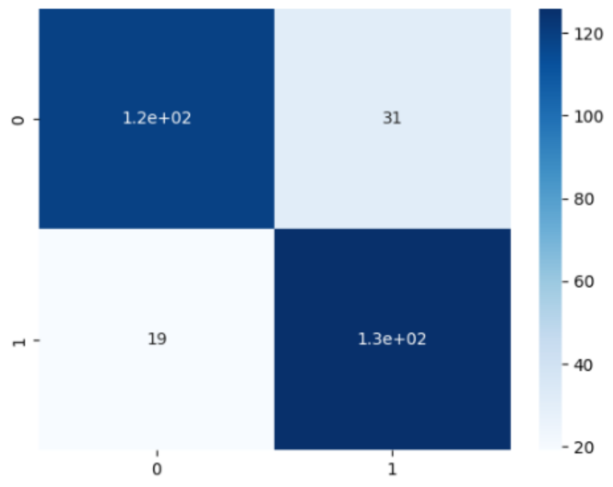


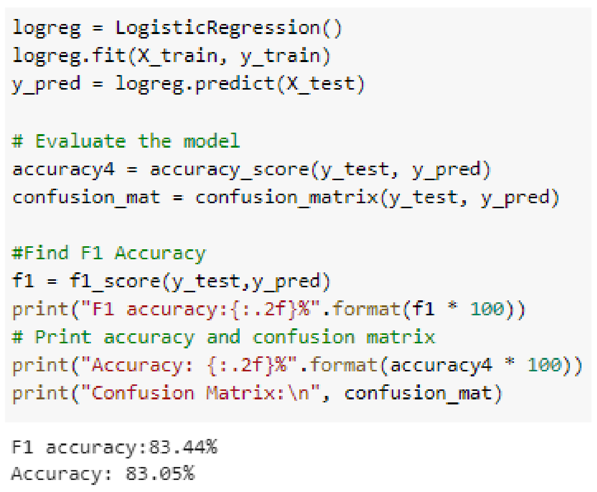
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.70      0.60      0.65       150
           1       0.64      0.73      0.68       145

    accuracy                           0.66       295
   macro avg       0.67      0.67      0.66       295
weighted avg       0.67      0.66      0.66       295
```

Logistic Regression

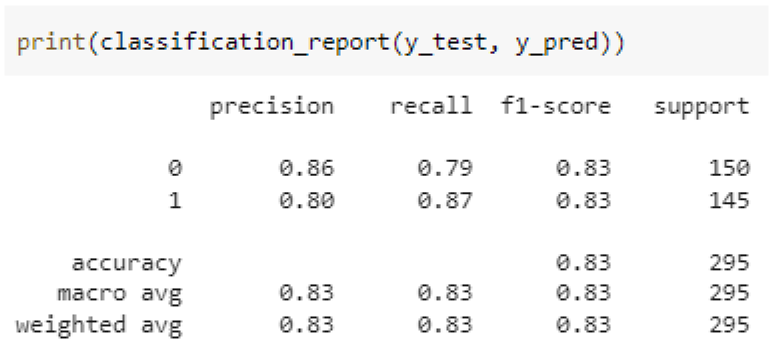
Logistic regression is a statistical method used for binary classification, where the goal is to predict the probability that an input data point belongs to one of two classes. It is a supervised learning algorithm that is commonly used in machine learning and statistics. In logistic regression, the input data points are represented by a set of features, which can be continuous, categorical, or binary. The output is a binary response variable, typically represented as either 0 or 1, which indicates the class membership of the data point. The goal of logistic regression is to estimate the parameters of a logistic function that models the relationship between the input features and the binary response variable. The logistic function, also known as the sigmoid function, maps the input features to a probability value between 0 and 1. The estimated parameters of the logistic function are obtained using a process called maximum likelihood estimation, where the model is trained on a labeled dataset with known class labels. Once the logistic regression model is trained, it can be used to make predictions on new, unseen data points by calculating the probability that the input data point belongs to the positive class based on its feature values. A threshold can be chosen to convert these probabilities into binary class labels.





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

# Evaluate the model
accuracy4 = 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)
```

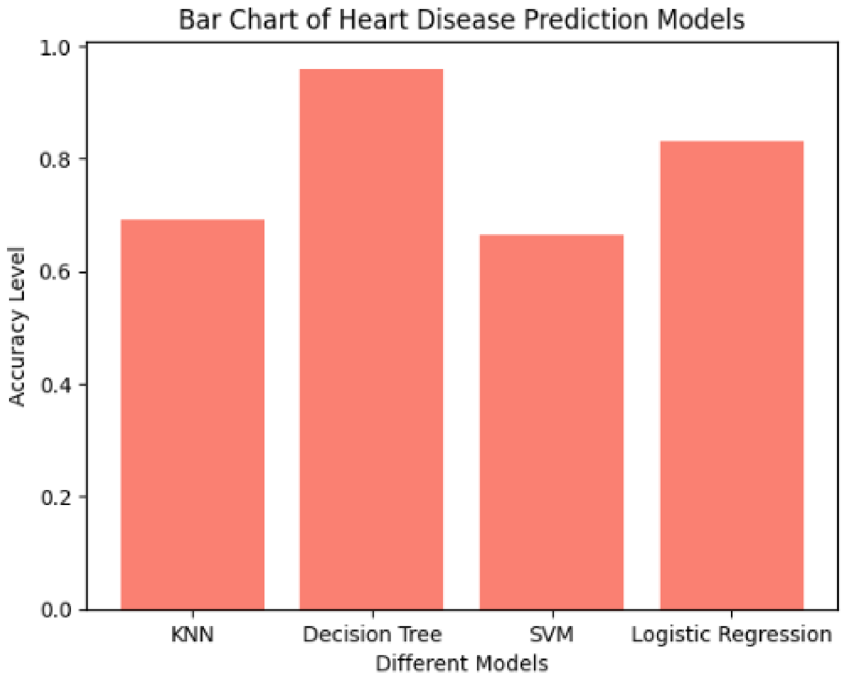
```
F1 accuracy:83.44%
Accuracy: 83.05%
```



```
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
```

Model selection/Comparison analysis



Model Name	F1 Score	Accuracy Rate
KNN	70.93%	69.15%
Decision Tree	95.95%	95.93%
SVM	68.17%	66.44%
Linear Regression	83.44%	83.05%

The decision tree model exhibits the highest accuracy, with an F1 score of 95.95% and overall accuracy of 95.93%. Its ability to capture complex relationships and handle both numerical and categorical features contributes to its strong performance. Logistic regression follows with an F1 score of 83.44% and accuracy of 83.05%, indicating decent performance but not as high as the decision tree. KNN and SVM models show lower accuracies, with F1 scores of 70.93% and 68.17%, and accuracies of 69.15% and 66.44% respectively. The decision tree's success could be attributed to its capability to handle non-linear relationships and feature interactions, making it suitable for this classification problem. The lower accuracy of the SVM model compared to the decision tree and logistic regression models could be due to several factors. Firstly, SVMs are more sensitive to the choice of hyperparameters and the scaling of the features. If the hyperparameters are not properly tuned or the features are not appropriately scaled, the SVM model's performance may suffer. Additionally, SVMs work well with linearly separable data, but their performance can degrade when dealing with complex or overlapping class boundaries. If the heart disease dataset has a complex decision boundary or overlapping classes, the SVM model may struggle to accurately classify the instances. Furthermore, the SVM model may be more prone to overfitting if the dataset is small or if there is a high dimensionality of features, which can negatively impact its generalization ability.

Conclusion

In this project, we developed and evaluated machine learning models for predicting heart disease risk based on comprehensive clinical and demographic features. The findings of this study demonstrate the potential of ML techniques in accurately predicting heart disease and providing valuable insights for early detection and intervention. The performance metrics of the ML models used, including accuracy, precision, recall, and F1 score, indicate the effectiveness of the predictive models in identifying individuals at high risk of heart disease.The results of this project have significant implications for healthcare practitioners, researchers, and policymakers. Accurate heart disease prediction models can assist healthcare practitioners in identifying high-risk individuals and implementing appropriate preventive measures, such as lifestyle interventions, medication, and follow-up care, to reduce the burden of heart disease. Moreover, these models can facilitate resource optimization in healthcare delivery by prioritizing high-risk individuals for further evaluation and intervention, leading to more efficient and cost-effective healthcare strategies.Furthermore, this project contributes to the growing body of research on the application of ML techniques in healthcare, specifically in the field of heart disease prediction. The findings of this study add to the evidence base for the use of ML algorithms in predicting heart disease risk and provide insights into the potential of these techniques for improving patient outcomes. However, it is important to acknowledge the limitations of this study, such as the use of a specific dataset, potential biases, and generalizability to different populations. Further research is needed to validate and refine the predictive models using diverse datasets and populations, as well as to investigate the interpretability and explainability of the ML models for clinical decision-making.

In conclusion, this project highlights the potential of ML techniques for early detection and prediction of heart disease. The outcomes of this study have implications for clinical practice, public health strategies, and policy-making, and contribute to the advancement of knowledge in the field of heart disease prediction. Further research in this area has the potential to improve patient care, reduce healthcare costs, and ultimately save lives.

References

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/#:~:text=Seaborn%20is%20a%20Python%20data,introductory%20notes%20or%20the%20paper.