INTRODUCTION

Heart disease remains a pressing global health concern, representing a significant cause of mortality and morbidity. The World Health Organization (WHO) reports that cardiovascular diseases are the leading cause of death worldwide, responsible for nearly 18 million deaths annually. The ability to predict and assess an individual's risk of heart disease is paramount for early intervention, improved patient outcomes, and the efficient allocation of healthcare resources.

Machine learning has emerged as a transformative tool in the field of healthcare, offering the potential to enhance the accuracy and effectiveness of heart disease prediction. By leveraging the vast amounts of healthcare data available, machine learning models can extract patterns and relationships that might escape human analysis, resulting in more accurate risk assessments and informed decision-making.

This project focuses on the development of a machine learning model designed to predict the risk of heart disease. We explore a comprehensive dataset encompassing various clinical, demographic, and diagnostic parameters to build and evaluate the model. By combining advanced machine learning algorithms with careful data preprocessing and model optimization, we aim to contribute to the ongoing efforts to improve heart disease diagnosis and risk assessment.

ABSTRACT

This project aims to develop and evaluate ML models for predicting heart disease. We will use a variety of ML algorithms, including logistic regression, Decision Tree, and ensemble methods. We will also explore the use of feature selection and dimensionality reduction techniques to improve the performance of our models.

We employed a diverse dataset comprising various clinical, demographic, and diagnostic parameters, collected from a cohort of patients from Mendeley Data. The dataset was preprocessed to handle missing values and outliers and was split into training and testing sets. Key steps in our project include feature engineering, and model evaluation using metrics like accuracy, precision. We also conducted cross-validation to ensure the robustness of our model.

This project has the potential to improve the early detection and prevention of heart disease. By providing clinicians with a tool for predicting heart disease risk, we can help them to identify patients who are at high risk for the disease and to initiate appropriate interventions. This project contributes to the ongoing efforts to leverage machine learning for improving heart disease diagnosis and risk assessment.

METHODOLOGY

1. Importing Dependencies

The project begins by importing necessary libraries such as NumPy, Pandas, and scikit-learn for data manipulation and machine learning. Matplotlib and Seaborn are used for data visualization.

```
#Importing the Dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt


import warnings
warnings.filterwarnings('ignore')
```

2. Loading and Exploring the dataset

The dataset, "Indian Heart Disease Dataset.csv," is obtained from “Mendeley Data” and loaded into a Pandas DataFrame for exploration.

```
#Loading the csv data to Pandas DataFrame
heart_data=pd.read_csv("Indian Heart Disease Dataset.csv")
heart_data.head()
```

	patientid	age	gender	chestpain	restingBP	serumcholestrol	fastingbloodsugar	restingrelectro	maxheartrate
0	103368	53	1	2	171	0	0	1	147
1	119250	40	1	0	94	229	0	1	115
2	119372	49	1	2	133	142	0	0	202
3	132514	43	1	0	138	295	1	1	153
4	146211	31	1	1	199	0	0	2	136

3. Exploratory Data Analysis (EDA)

3.1 Checking Initial Dataset Information:

- heart_data.shape returns the total number of rows and columns in the dataset, which is printed to provide an overview.

```
#Number of rows and columns in the dataset
heart_data.shape
```

```
(1000, 14)
```

- heart_data.info() provides a concise summary of the dataset, including the data types of each column and the presence of missing values.

```
#Getting some info about the data
heart_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   patientid           1000 non-null   int64
 1   age                 1000 non-null   int64
 2   gender              1000 non-null   int64
 3   chestpain           1000 non-null   int64
 4   restingBP           1000 non-null   int64
 5   serumcholestrol     1000 non-null   int64
 6   fastingbloodsugar   1000 non-null   int64
 7   restingrelectro     1000 non-null   int64
 8   maxheartrate        1000 non-null   int64
 9   exerciseangia       1000 non-null   int64
 10  oldpeak             1000 non-null   float64
 11  slope               1000 non-null   int64
 12  noofmajorvessels    1000 non-null   int64
 13  target              1000 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 109.5 KB
```

3.2 Checking for Missing Values:

This step is essential to ensure that there are no missing values in the dataset. In this project, there are no missing values.

```
#checking for missing values
heart_data.isnull().sum()
```

```
patientid           0
age                 0
gender              0
chestpain           0
restingBP           0
serumcholestrol     0
fastingbloodsugar   0
restingrelectro     0
maxheartrate        0
exerciseangia       0
oldpeak             0
slope               0
noofmajorvessels    0
target              0
dtype: int64
```

3.3 Descriptive Statistics:

This provides summary statistics such as mean, standard deviation, minimum, maximum, and quartiles for numerical features in the dataset.

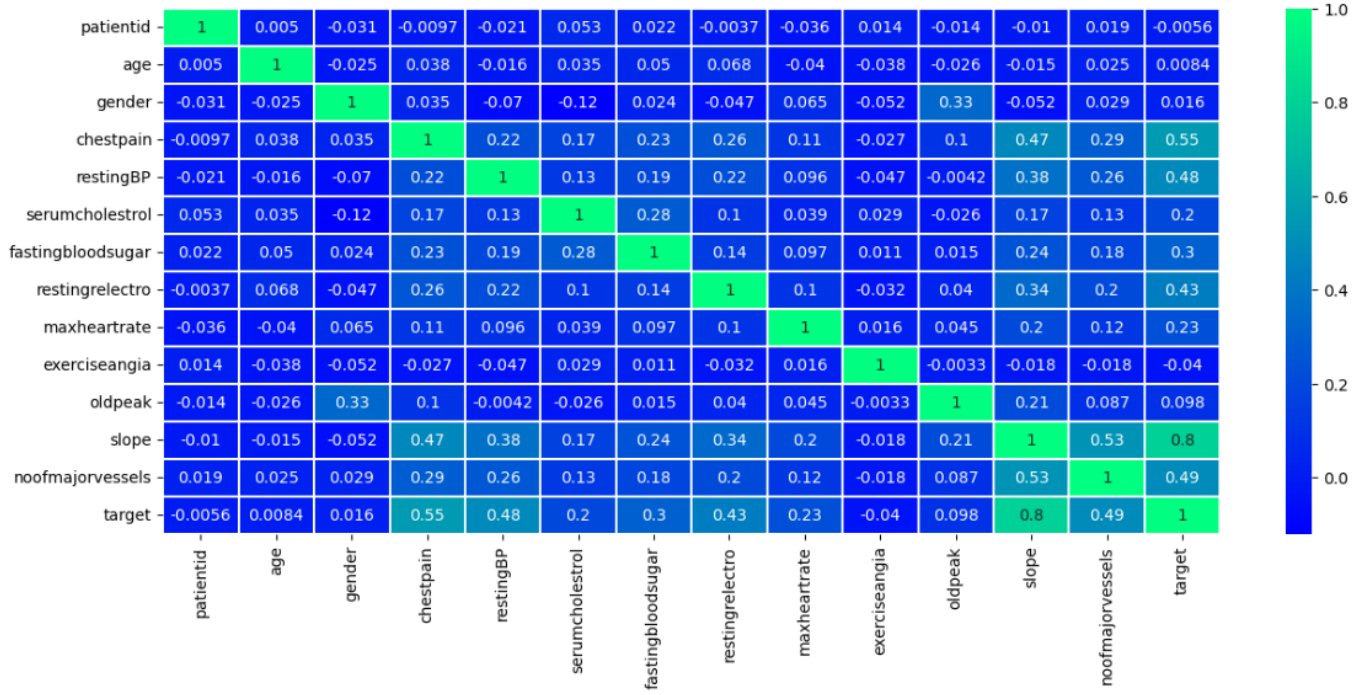
```
#Statistical Measures about the data
heart_data.describe()
```

	patientid	age	gender	chestpain	restingBP	serumcholestrol	fastingbloodsugar	restingrelectro
count	1.000000e+03	1000.00000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	5.048704e+06	49.24200	0.765000	0.980000	151.747000	311.447000	0.296000	0.748000
std	2.895905e+06	17.86473	0.424211	0.953157	29.965228	132.443801	0.456719	0.770123
min	1.033680e+05	20.00000	0.000000	0.000000	94.000000	0.000000	0.000000	0.000000
25%	2.536440e+06	34.00000	1.000000	0.000000	129.000000	235.750000	0.000000	0.000000
50%	4.952508e+06	49.00000	1.000000	1.000000	147.000000	318.000000	0.000000	1.000000
75%	7.681877e+06	64.25000	1.000000	2.000000	181.000000	404.250000	1.000000	1.000000
max	9.990855e+06	80.00000	1.000000	3.000000	200.000000	602.000000	1.000000	2.000000

3.4 Correlation Analysis:

This heat map visualizes the correlation between different features. It helps identify relationships between variables and assess multicollinearity.

```
plt.figure(figsize=(15,6))
sns.heatmap(heart_data.corr(),linewidth=.01,annot=True,cmap="winter")
plt.show()
```



3.5 Target Variable Distribution

Understanding the distribution of the target variable is crucial for evaluating the balance of classes in a classification problem. In this case, it helps to know how many instances are labeled as having heart disease (1) and how many do not (0).

```
#Checking the distribution of Target Variable
heart_data['target'].value_counts()
```

```
1    580
0    420
Name: target, dtype: int64
```

4. Splitting the Data

Splitting the dataset into features (X) and the target variable (Y). Further, splitting the data into training and testing sets. This step is crucial for training the machine learning models on a subset of the data and evaluating their performance on another subset.

```
#Splitting the Features and the Target
X=heart_data.drop(["target","patientid"],axis=1)
Y=heart_data["target"]
```

```
#Splitting the data into Training Data and Test Data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,
                                               stratify=Y,random_state=2)
```

```
print(X.shape,X_train.shape,X_test.shape)
```

```
(1000, 12) (800, 12) (200, 12)
```

5. Model Training and Evaluation

LOGISTIC REGRESSION MODEL

- The Logistic Regression model is instantiated and trained using the training data.

```
#Model Training (Logistic Regression)
LR_model=LogisticRegression()
LR_model.fit(X_train,Y_train)
```

```
LogisticRegression()
```

- The model's accuracy is evaluated on the training data to understand how well it fits the training set.

```
#Accuracy on training data of Logistic Regression
X_train_prediction_LR=LR_model.predict(X_train)
LR_training_data_accuracy=accuracy_score(X_train_prediction_LR,Y_train)
print("Accuracy on training data of Logistic Regression:",LR_training_data_accuracy)
```

```
Accuracy on training data of Logistic Regression: 0.93375
```

- The model's accuracy is evaluated on the test data to assess its generalization performance.

```
#Accuracy on test data of Logistic Regression
X_test_prediction_LR=LR_model.predict(X_test)
LR_test_data_accuracy=accuracy_score(X_test_prediction_LR,Y_test)
print("Accuracy on test data of Logistic Regression:",LR_test_data_accuracy)
```

```
Accuracy on test data of Logistic Regression: 0.94
```

- The classification report provides precision, recall, F1-score, and support for each class. It offers a comprehensive overview of the model's performance.

```
from sklearn.metrics import classification_report
print("Classification report of Logistic Regression:\n",
      classification_report(Y_test,X_test_prediction_LR))
```

```
Classification report of Logistic Regression:
              precision    recall  f1-score   support

           0       0.94      0.92      0.93        84
           1       0.94      0.96      0.95       116

    accuracy                           0.94       200
   macro avg       0.94      0.94      0.94       200
weighted avg       0.94      0.94      0.94       200
```

DECISION TREE MODEL

- The Decision Tree model is instantiated and trained with specified hyperparameters.

```
#Model Training (Decision Tree)
tree_model = DecisionTreeClassifier(max_depth=5,criterion='entropy')
tree_model.fit(X_train, Y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=5)
```

- The accuracy on the training data is calculated.

```
#Accuracy on training data of Decison Tree
X_train_prediction_tree=tree_model.predict(X_train)
tree_training_data_accuracy=accuracy_score(X_train_prediction_tree,Y_train)
print("Accuracy on training data of Decison Tree:",tree_training_data_accuracy)
```

```
Accuracy on training data of Decison Tree: 0.98375
```

- The accuracy on the test data is calculated.

```
#Accuracy on test data of Decison Tree
X_test_prediction_tree=tree_model.predict(X_test)
tree_test_data_accuracy=accuracy_score(X_test_prediction_tree,Y_test)
print("Accuracy on test data of Decison Tree:",tree_test_data_accuracy)
```

```
Accuracy on test data of Decison Tree: 0.955
```

- The classification report provides a detailed analysis of the Decision Tree model's performance.

```
print("Classification report of Decison Tree:\n",
      classification_report(Y_test,X_test_prediction_tree))
```

```
Classification report of Decison Tree:
              precision    recall  f1-score   support

           0       0.94      0.95      0.95        84
           1       0.97      0.96      0.96       116

    accuracy                           0.95       200
   macro avg       0.95      0.95      0.95       200
weighted avg       0.96      0.95      0.96       200
```

RANDOM FOREST MODEL

- The Random Forest model is instantiated and trained with specified hyperparameters.

```
#Model Training (Random Forest)
RF_model=RandomForestClassifier(n_estimators=300,criterion='entropy',
                                max_depth=3,min_samples_split=5)
RF_model.fit(X_train, Y_train)
```

```
RandomForestClassifier(criterion='entropy', max_depth=3, min_samples_split=5,
                       n_estimators=300)
```

- The accuracy on the training data is calculated.

```
#Accuracy on training data of Random Forest
X_train_prediction_RF=RF_model.predict(X_train)
RF_training_data_accuracy=accuracy_score(X_train_prediction_RF,Y_train)
print("Accuracy on training data of Random Forest:",RF_training_data_accuracy)
```

```
Accuracy on training data of Random Forest: 0.95125
```

- The accuracy on the test data is calculated.

```
#Accuracy on test data of Random Forest
X_test_prediction_RF=RF_model.predict(X_test)
RF_test_data_accuracy=accuracy_score(X_test_prediction_RF,Y_test)
print("Accuracy on test data of Random Forest:",RF_test_data_accuracy)
```

```
Accuracy on test data of Random Forest: 0.965
```

- The classification report provides a detailed analysis of the Random Forest model's performance.

```
print("Classification report of Random Forest:\n",
      classification_report(Y_test,X_test_prediction_RF))
```

```
Classification report of Random Forest:
              precision    recall  f1-score   support

           0       0.99      0.93      0.96        84
           1       0.95      0.99      0.97       116

    accuracy                           0.96       200
   macro avg       0.97      0.96      0.96       200
weighted avg       0.97      0.96      0.96       200
```

The Logistic Regression, Decision Tree, and Random Forest models are trained and evaluated. Training accuracy is assessed to understand how well each model fits the training data. Test accuracy is measured to evaluate each model's generalization performance. Classification reports provide detailed metrics for each class, including precision, recall, and F1-score.The Random Forest model demonstrates balanced performance and is recommended for its accuracy on both training and test data.

6. Building a Predictive System

- The project includes a user-friendly predictive system that allows individuals to input their health parameters, and the system predicts the likelihood of heart disease based on the trained Random Forest model.
- The system prompts the user to input various health parameters such as age, sex, chest pain, blood pressure, cholesterol level, etc.
- The user-input data is then converted into a tuple, then into a NumPy array, and reshaped to match the format expected by the machine learning model.
- The trained Random Forest model is used to predict whether the user is likely to have heart disease based on the provided health parameters.

```
#Building Predictive System
age=int(input("Age of the person:"))
sex=int(input("Sex(1 = male; 0 = female):"))
cp=int(input("Describe Chest Pain(0-3):"))
trestbps=int(input("Resting Blood Pressure(in mm)(avg=120):"))
chol=int(input("Cholestoral level (avg=200):"))
fbs=int(input("Is Diabetes level > 120 mg/dl?(1 = true; 0 = false):"))
restecg=int(input("Resting electrocardiographic results (values 0,1,2):"))
thalach=int(input("Maximum heart rate achieved(avg=170):"))
exang=int(input("Exercise induced angina (1 = yes; 0 = no):"))
oldpeak=float(input("ST depression (between 0 & 7):"))
slope=int(input("The slope of ST segment (values 0,1,2):"))
ca=int(input("Number of major vessels (0-4) colored by flourosopy:"))

input_data=(age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca)
input_data

#change the input data to a numpy array
input_data_as_numpy_array=np.asarray(input_data)

#reshape the numpy array as we are predicting for only one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)

prediction=RF_model.predict(input_data_reshaped)

print(prediction)
```

- The system outputs the prediction, indicating whether the person is predicted to have heart disease or not.

```
if (prediction) ==0:
    print("The person does not have heart diesease")
else:
    print("The person has heart Disease ")
```

The predictive system allows users to interactively input their health parameters and receive a quick assessment of the likelihood of heart disease. It leverages the trained Random Forest model to make predictions based on the input data. The system is designed to be user-friendly and provides actionable information for individuals concerned about their cardiovascular health.

SOURCE CODE

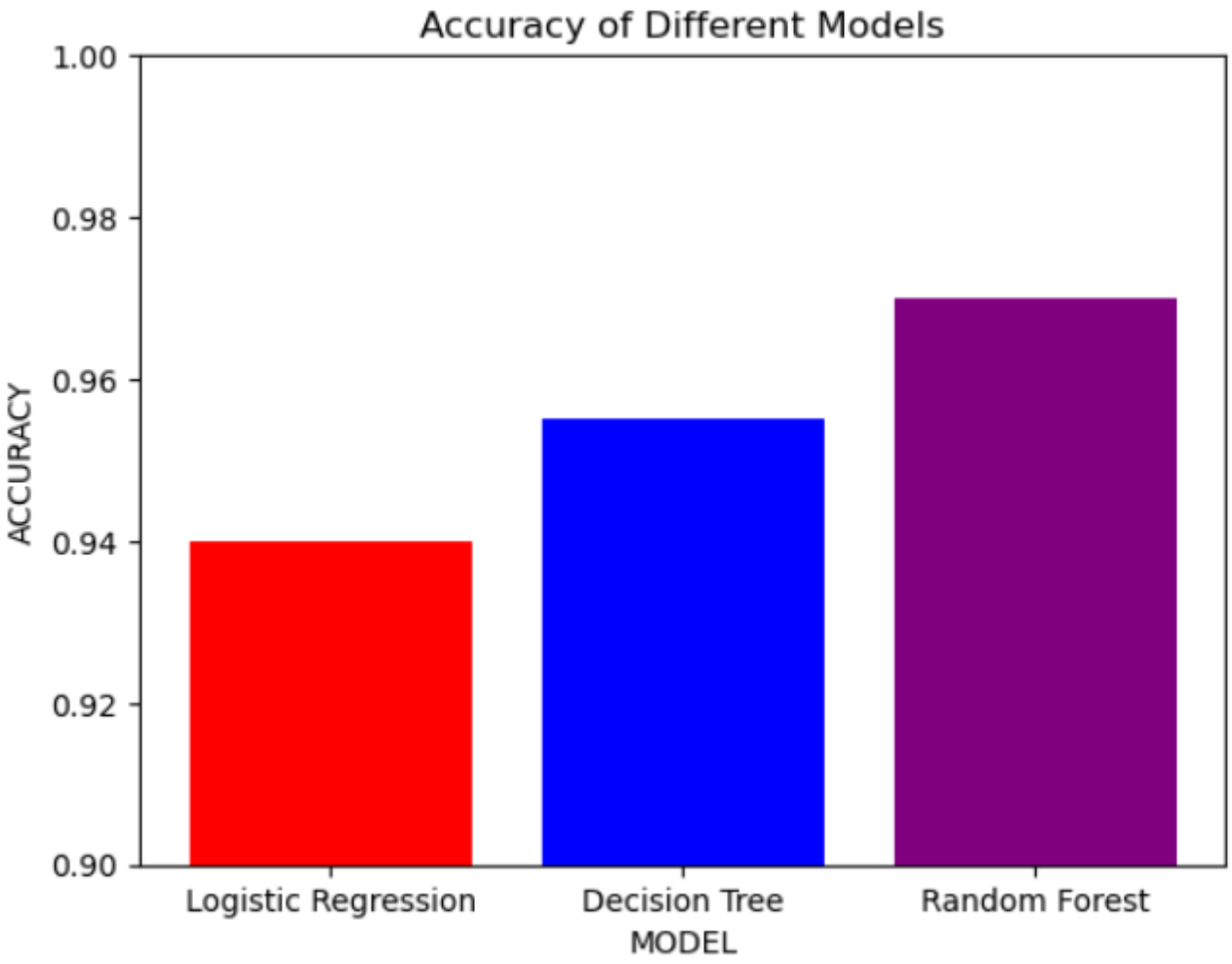
```
#Importing the Dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

#Loading the csv data to Pandas DataFrame
heart_data=pd.read_csv("Indian Heart Disease Dataset.csv")
heart_data.head()
heart_data.tail()

#Number of rows and columns in the dataset
heart_data.shape
#Getting some info about the data
heart_data.info()
#checking for missing values
heart_data.isnull().sum()

#Statistical Measures about the data
heart_data.describe()
plt.figure(figsize=(15,5))
sns.heatmap(heart_data.corr(),linewidth=.01,annot=True,cmap="winter")
plt.show()
```

```
#Checking the distribution of Target Variable
heart_data['target'].value_counts()
# 1--> Defective Heart
# 0--> Healthy Heart
#Splitting the Features and the Target
X=heart_data.drop(["target","patientid"],axis=1)
Y=heart_data["target"]
X
Y
#Splitting the data into Training Data and Test Data
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
print(X.shape,X_train.shape,X_test.shape)

#Model Training (Logistic Regression)
LR_model=LogisticRegression()
LR_model.fit(X_train,Y_train)

#Accuracy on training data of Logistic Regression
X_train_prediction_LR=LR_model.predict(X_train)
LR_training_data_accuracy=accuracy_score(X_train_prediction_LR,Y_train)
print("Accuracy on training data of Logistic Regression:",LR_training_data_accuracy)

#Accuracy on test data of Logistic Regression
X_test_prediction_LR=LR_model.predict(X_test)
LR_test_data_accuracy=accuracy_score(X_test_prediction_LR,Y_test)
print("Accuracy on test data of Logistic Regression:",LR_test_data_accuracy)

from sklearn.metrics import classification_report
print("Classification report of Logistic Regression:\n",classification_report(Y_test,X_test_prediction_LR))

#Model Training (Decision Tree)
tree_model = DecisionTreeClassifier(max_depth=5,criterion='entropy')
tree_model.fit(X_train, Y_train)
```

```
#Accuracy on training data of Decison Tree
X_train_prediction_tree=tree_model.predict(X_train)
tree_training_data_accuracy=accuracy_score(X_train_prediction_tree,Y_train)
print("Accuracy on training data of Decison Tree:",tree_training_data_accuracy)

#Accuracy on test data of Decison Tree
X_test_prediction_tree=tree_model.predict(X_test)
tree_test_data_accuracy=accuracy_score(X_test_prediction_tree,Y_test)
print("Accuracy on test data of Decison Tree:",tree_test_data_accuracy)
print("Classification report of Decison Tree:\n",classification_report(Y_test,X_test_prediction_tree))

#Model Training (Random Forest)
RF_model=RandomForestClassifier(n_estimators=300,criterion='entropy',max_depth=3,min_samples_split
=5)
RF_model.fit(X_train, Y_train)

#Accuracy on training data of Random Forest
X_train_prediction_RF=RF_model.predict(X_train)
RF_training_data_accuracy=accuracy_score(X_train_prediction_RF,Y_train)
print("Accuracy on training data of Random Forest:",RF_training_data_accuracy)

#Accuracy on test data of Random Forest
X_test_prediction_RF=RF_model.predict(X_test)
RF_test_data_accuracy=accuracy_score(X_test_prediction_RF,Y_test)
print("Accuracy on test data of Random Forest:",RF_test_data_accuracy)
print("Classification report of Random Forest:\n",classification_report(Y_test,X_test_prediction_RF))

x=['Logistic Regression','Decision Tree','Random Forest']
y=[LR_test_data_accuracy,tree_test_data_accuracy,RF_test_data_accuracy]
plt.bar(x,y,color=["red","blue","purple"])
plt.xlabel("MODEL")
plt.ylabel("ACCURACY")
plt.title("Accuracy of Different Models")
```

```
plt.ylim([0.9,1])
plt.show()

#Building Predictive System
age=int(input("Age of the person:"))
sex=int(input("Sex(1 = male; 0 = female):"))
cp=int(input("Describe Chest Pain(0-3):"))
trestbps=int(input("Resting Blood Pressure(in mm)(avg=120):"))
chol=int(input("Cholestoral level (avg=200):"))
fbs=int(input("Is Diabetes level > 120 mg/dl?(1 = true; 0 = false):"))
restecg=int(input("Resting electrocardiographic results (values 0,1,2):"))
thalach=int(input("Maximum heart rate achieved(avg=170):"))
exang=int(input("Exercise induced angina (1 = yes; 0 = no):"))
oldpeak=float(input("ST depression (between 0 & 7):"))
slope=int(input("The slope of ST segment (values 0,1,2):"))
ca=int(input("Number of major vessels (0-4) colored by flourosopy:"))

input_data=(age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca)
input_data

#change the input data to a numpy array
input_data_as_numpy_array=np.asarray(input_data)

#reshape the numpy array as we are predicting for only one instance
input_data_reshaped=input_data_as_numpy_array.reshape(1,-1)

prediction=RF_model.predict(input_data_reshaped)

print(prediction)
if (prediction) ==0:
   print("The person does not have heart diesease")
else:
   print("The person has heart Disease ")
```

GRAPHS



From the above graph we can see that Random Forest Model has the highest accuracy than Logistic Regression or Decision Tree with 96%. Therefore we use Random Forest Model in our predictive system to predict the likeliness of having a heart disease.

OUTPUT

Case 1

```
Age of the person:53
Sex(1 = male; 0 = female):1
Describe Chest Pain(0-3):2
Resting Blood Pressure(in mm)(avg=120):171
Cholestoral level (avg=200):170
Is Diabetes level > 120 mg/dl?(1 = true; 0 = false):1
Resting electrocardiographic results (values 0,1,2):2
Maximum heart rate achieved(avg=170):180
Exercise induced angina (1 = yes; 0 = no):1
ST depression (between 0 & 7):3
The slope of ST segment (values 0,1,2):0
Number of major vessels (0-4) colored by flourosopy:0
[1]


The person has heart Disease
```

Case 2

```
Age of the person:18
Sex(1 = male; 0 = female):0
Describe Chest Pain(0-3):0
Resting Blood Pressure(in mm)(avg=120):115
Cholestoral level (avg=200):180
Is Diabetes level > 120 mg/dl?(1 = true; 0 = false):0
Resting electrocardiographic results (values 0,1,2):0
Maximum heart rate achieved(avg=170):138
Exercise induced angina (1 = yes; 0 = no):0
ST depression (between 0 & 7):1.1
The slope of ST segment (values 0,1,2):0
Number of major vessels (0-4) colored by flourosopy:3
[0]


The person does not have heart diesease
```

Case 3

```
Age of the person:78
Sex(1 = male; 0 = female):1
Describe Chest Pain(0-3):3
Resting Blood Pressure(in mm)(avg=120):130
Cholestoral level (avg=200):230
Is Diabetes level > 120 mg/dl?(1 = true; 0 = false):1
Resting electrocardiographic results (values 0,1,2):2
Maximum heart rate achieved(avg=170):180
Exercise induced angina (1 = yes; 0 = no):1
ST depression (between 0 & 7):5.2
The slope of ST segment (values 0,1,2):2
Number of major vessels (0-4) colored by flourosopy:2


The person has heart Disease
```

CONCLUSION

In conclusion, this project aimed to develop a predictive model for identifying the likelihood of heart disease based on various health parameters. The methodology involved data exploration, preprocessing, model training, and the creation of a user-friendly predictive system.

Logistic Regression provided a baseline for binary classification with moderate accuracy. Decision Tree captured non-linear relationships but may be prone to overfitting without proper tuning. Random Forest demonstrated balanced performance, combining multiple decision trees to enhance accuracy on both training and test data. The Random Forest model emerged as the preferred choice due to its robust performance.

This project impacts to empowering individuals to make informed decisions about their cardiovascular health. Providing a tool for quick risk assessment and early detection of potential heart disease.

In summary, this project successfully addressed the goal of developing a predictive model for heart disease identification. The Random Forest model, combined with the user-friendly predictive system, creates a valuable tool for individuals to assess their risk and take proactive measures towards heart health. Ongoing improvements and collaborations with healthcare experts will contribute to the continuous enhancement of the model's accuracy and applicability.

BIBLIOGRAPHY

- https://data.mendeley.com/datasets/dzz48mvjht/1
- https://youtu.be/qmqCYC-MBQo?si=IGpR-_0SWkckOMkI
- https://chat.openai.com/c/74d061a2-5795-4a77-9cb5-8d0d80025b14
- https://www.kaggle.com/code/nourhanelsabawy/heart-disease-prediction-100-acc-with-deployment
- https://www.kaggle.com/code/syedali110/heart-disease-detection