**Title of Project**  
Heart Disease Prediction using Machine Learning Algorithms

**Data Source**

The dataset used for this project is the "heart.csv" file, which contains information on patients related to heart disease.

**Import Library**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import cross\_val\_score, train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

**Import Data**

*# Loading the dataset*

df = pd.read\_csv('heart.csv')

**Describe Data**

*# Number of rows and columns in the dataset*

df.shape

*# Column headers*

df.columns

*# Data types for each column*

df.dtypes

*# First 5 rows of the dataset*

df.head()

*# Last 5 rows of the dataset*

df.tail()

*# Check for null values*

df.isnull().any()

*# Basic information about the dataset*

df.info()

*# Basic statistics of numeric columns*

df.describe().T

**Data Visualization**

*# Plotting histogram for the entire dataset*

fig = plt.figure(figsize=(15, 15))

ax = fig.gca()

g = df.hist(ax=ax)

*# Check if the dataset is balanced*

g = sns.countplot(x='target', data=df)

plt.xlabel('Target')

plt.ylabel('Count')

*# Heatmap to visualize correlated features*

corr\_matrix = df.corr()

top\_corr\_features = corr\_matrix.index

plt.figure(figsize=(20, 20))

sns.heatmap(data=df[top\_corr\_features].corr(), annot=True, cmap='RdYlGn')

**Data Pre-processing**

*# Converting categorical variables into dummy variables*

dataset = pd.get\_dummies(df, columns=['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])

*# Feature Scaling*

standScaler = StandardScaler()

columns\_to\_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

dataset[columns\_to\_scale] = standScaler.fit\_transform(dataset[columns\_to\_scale])

*# Displaying the first 5 rows after preprocessing*

dataset.head()

**Define Target Variable (y) and Feature Variable (X)**

*# Splitting the dataset into dependent (y) and independent (X) features*

X = dataset.drop('target', axis=1)

y = dataset['target']

**Train Test Split**

*# Splitting the dataset into training and testing sets (not explicitly done in the provided code but usually follows)*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Modelling**

***KNeighbors Classifier Model***

*# Finding the best accuracy for KNN algorithm using cross\_val\_score*

knn\_scores = []

for i in range(1, 21):

knn\_classifier = KNeighborsClassifier(n\_neighbors=i)

cvs\_scores = cross\_val\_score(knn\_classifier, X, y, cv=10)

knn\_scores.append(round(cvs\_scores.mean(), 3))

*# Plotting the results of knn\_scores*

plt.figure(figsize=(20, 15))

plt.plot([k for k in range(1, 21)], knn\_scores, color='red')

for i in range(1, 21):

plt.text(i, knn\_scores[i-1], (i, knn\_scores[i-1]))

plt.xticks([i for i in range(1, 21)])

plt.xlabel('Number of Neighbors (K)')

plt.ylabel('Scores')

plt.title('K Neighbors Classifier scores for different K values')

*# Training the KNN classifier model with k=12*

knn\_classifier = KNeighborsClassifier(n\_neighbors=12)

cvs\_scores = cross\_val\_score(knn\_classifier, X, y, cv=10)

print("KNeighbours Classifier Accuracy with K=12 is: {}%".format(round(cvs\_scores.mean(), 4)\*100))

***Decision Tree Classifier Model***

# Finding the best accuracy for Decision Tree algorithm using cross\_val\_score

decision\_scores = []

for i in range(1, 11):

decision\_classifier = DecisionTreeClassifier(max\_depth=i)

cvs\_scores = cross\_val\_score(decision\_classifier, X, y, cv=10)

decision\_scores.append(round(cvs\_scores.mean(), 3))

# Plotting the results of decision\_scores

plt.figure(figsize=(20, 15))

plt.plot([i for i in range(1, 11)], decision\_scores, color='red')

for i in range(1, 11):

plt.text(i, decision\_scores[i-1], (i, decision\_scores[i-1]))

plt.xticks([i for i in range(1, 11)])

plt.xlabel('Depth of Decision Tree (N)')

plt.ylabel('Scores')

plt.title('Decision Tree Classifier scores for different depth values')

# Training the Decision Tree classifier model with max\_depth=3

decision\_classifier = DecisionTreeClassifier(max\_depth=3)

cvs\_scores = cross\_val\_score(decision\_classifier, X, y, cv=10)

print("Decision Tree Classifier Accuracy with max\_depth=3 is: {}%".format(round(cvs\_scores.mean(), 4)\*100))

***Random Forest Classifier Model***

# Finding the best accuracy for Random Forest algorithm using cross\_val\_score

forest\_scores = []

for i in range(10, 101, 10):

forest\_classifier = RandomForestClassifier(n\_estimators=i)

cvs\_scores = cross\_val\_score(forest\_classifier, X, y, cv=5)

forest\_scores.append(round(cvs\_scores.mean(), 3))

# Plotting the results of forest\_scores

plt.figure(figsize=(20, 15))

plt.plot([n for n in range(10, 101, 10)], forest\_scores, color='red')

for i in range(1, 11):

plt.text(i\*10, forest\_scores[i-1], (i\*10, forest\_scores[i-1]))

plt.xticks([i for i in range(10, 101, 10)])

plt.xlabel('Number of Estimators (N)')

plt.ylabel('Scores')

plt.title('Random Forest Classifier scores for different N values')

# Training the Random Forest classifier model with n\_estimators=90

forest\_classifier = RandomForestClassifier(n\_estimators=90)

cvs\_scores = cross\_val\_score(forest\_classifier, X, y, cv=5)

print("Random Forest Classifier Accuracy with n\_estimators=90 is: {}%".format(round(cvs\_scores.mean(), 4)\*100))

**Model Evaluation**  
Evaluation was performed using cross-validation scores, as demonstrated in the modelling section. The accuracy of each model was calculated and compared to determine the best-performing model.

**Prediction**  
Not explicitly done in the provided code, but predictions would typically follow after the training step.

-

**Explanation**  
In this project, three machine learning algorithms were applied to predict heart disease: KNeighbors Classifier, Decision Tree Classifier, and Random Forest Classifier. The KNeighbors Classifier achieved the highest accuracy of 85.07% with 12 neighbors. The results were evaluated using cross-validation to ensure reliability. Each model's performance was analyzed, and visualizations were created to better understand the data and correlations between features.