# Heart Disease Prediction

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**


Under the Guidance of

**(Rajiv Senapati)**

**SRM University-AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh - 522 240**

**[May, 2023]**

## Abstract

Heart disease cases are rising at an alarming rate, and it's critical to be able to predict these diseases in advance. The project focuses on predicting which patients are more likely to have heart disease based on avariety of medical factors. To predict and identify patients with heart disease, we used different algorithms such as Logistic regression, Decision tree, Random forest and KNN. The proposed model's accuracy was quite good, and it was able to predict signs of heart disease in a person. This heart disease predictive method improves patient treatment and makes diagnosing the disease easier along with allowing exploring large data at once.

# 1. Introduction

Machine learning is a form of artificial intelligence that enables the machine to create and implement algorithms that can learn from previous experiences. We used a variety of classifiers from supervised and unsupervised learning to predict and determine the dataset's accuracy. Cardiovascular diseases refer to various disorders that can affect the heart and circulatory system. Heart disease has been common for a long time and is still one of the most severe diseases today. According to WHO, cardiovascular diseases(CVDs) are the number 1 cause of death globally.

Four out of 5 CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age [1]. Our research can identify people who are more likely to be diagnosed with heart disease based on their medical history. It predicts based on factors like sugar level, blood pressure, chest pain, cholesterol, etc. This way, people will be able to know about themselves beforehand and take necessary precautions. To verify the accuracy and explore different models, we used the classifiers KNN, Decision Tree, Random Forest, and logistic regression models. We used a dataset of people with and without heart disease to predict heart disease. We used 14 different attributes of a patient to predict if they are susceptible to heart disease. The more efficient of these algorithms is Random Forest which gives us the best accuracy. We used many graphical representations to present the results of the project's predictions and such.

This project was inspired by a significant amount of work on the detection of CVDs using Machine Learning algorithms. ML algorithms have been used to make several efficient heart disease predictions. Using the previous and current machine learning and deep learning models, the model incorporating IHDPS was able to predict the likelihood of a person getting heart disease pretty accurately. It focused on basic attributes like age, sex, blood pressure, and blood sugar. But new models which use deep learning and neural networks are more efficient, accurate and reliable. A new neural network model had the classification power of 77% to correctly classify the presence of Coronary Heart Disease (CHD) and 81.8% to accurately classify the absence of CHD cases on testing data, which is 85.70% of the total of their datase

## 1.1 Data Source

We collected the dataset from Kaggle to train the model.
The data was gathered from multiple instances. The database is taken from the UCI repository . It initially had 76 attributes but a subset of 14 attributes was used. The dataset includes a variety of individuals and their histories of heart disease, as well as other medical conditions. The dataset consists of the medical history of 1025 different patients of different attributes spread across. This dataset provides details about the patient's medical characteristics, such as age, chest pain types, blood pressure, sugar level, angina, and so on, which enables us in determining whether or not the patient has been diagnosed with heart disease.

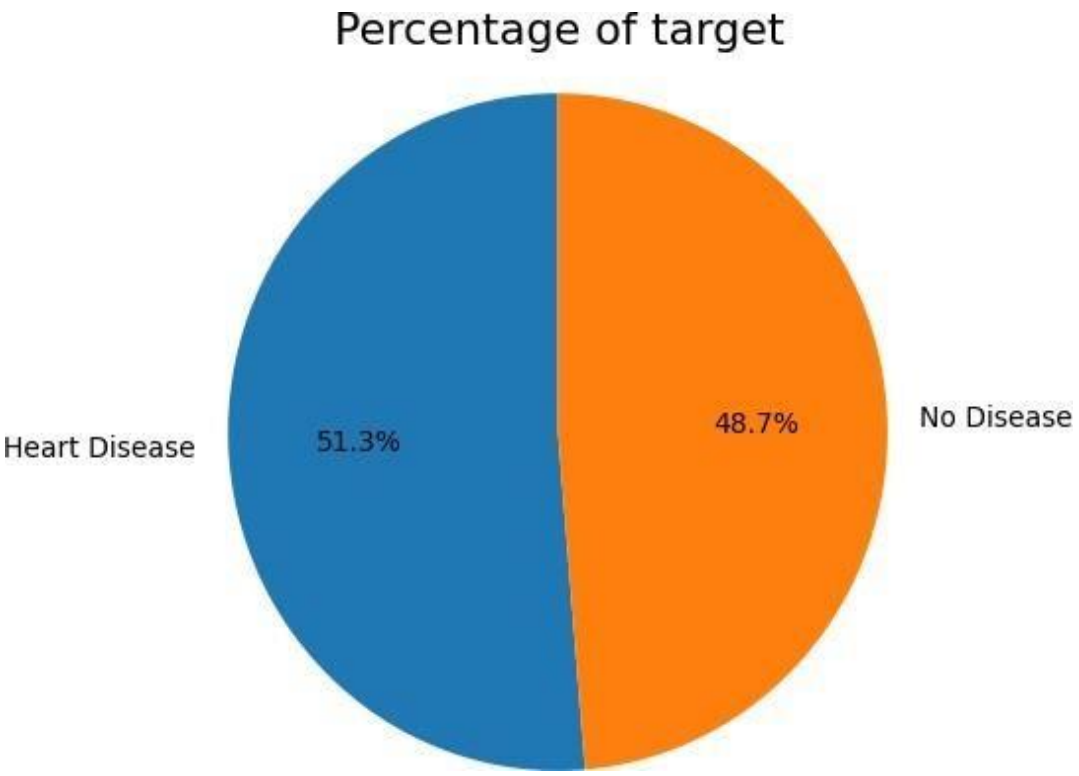
**Table 1. Various attributes used are listed**

| SL. No | Observation | Description |
|---|---|---|
| 1. | Age | Age in years |
| 2. | Sex | Sex of patients |
| 3. | CP | Chest pain |
| 4. | Trestbps | Resting blood pressure |
| 5. | Chol | Serum cholesterol |
| 6. | FBS | Fasting blood pressure |
| 7. | Restecg | Resting electrocardiograph results |
| 8. | Thalach | Maximum heart rate achieved |
| 9. | Exang | Exercise-induced angina |
| 10. | Oldpeak | ST depression induced by exercise relative to rest |
| 11. | Slope | the slope of the peak exercise ST segment |
| 12. | Ca | number of major vessels colored by fluoroscopy |
| 13. | thal | Defect type |
| 14. | target | |

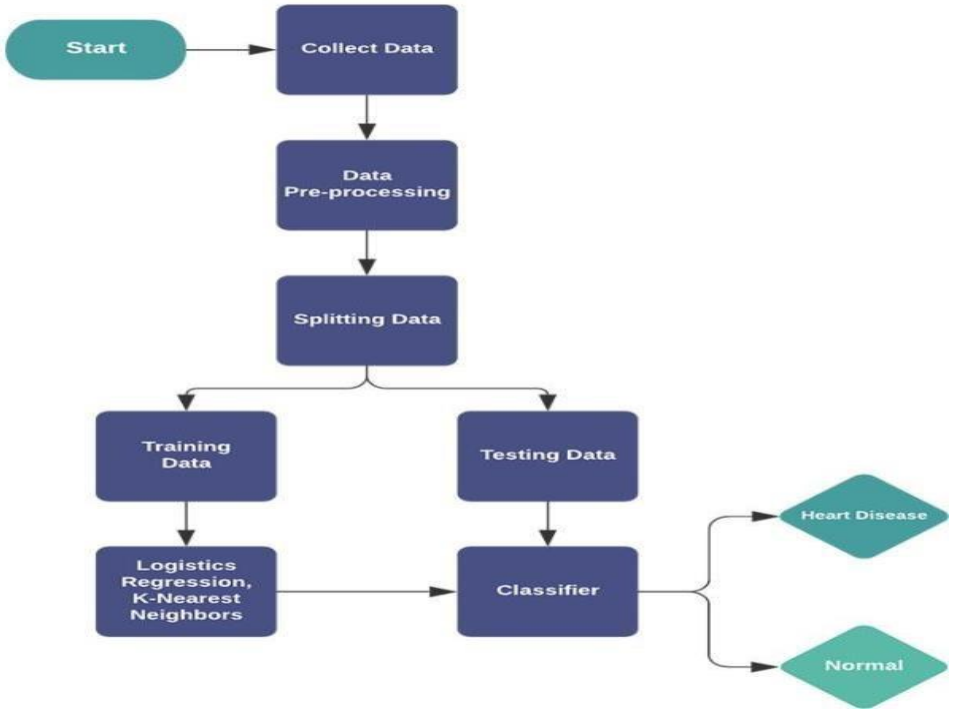
## 2.Methodology

We did an analysis of four machine learning algorithms K nearest neighbors (KNN),Decision Tree, Random Forest and Logistic Regression which are pretty accurate to this certain predicting model. The proposed approach is organized as follows: the first phase is data collection, the second stage is substantial value extraction, and the third stage is data exploration. Depending on the algorithms used, data preprocessing deals with missing values, data cleaning, and normalization.

The classifier used in the proposed models is then used to identify the pre-processed data after it has been pre-processed. Later, we put the proposed model to the test, evaluating it for accuracy and performance using a variety of performance metrics. We used a full dataset to test the model. Percentage of the patient data suffering with heart disease has been calculated and shown using a graphical representation using pie charts to let know the total number of patients with and without heart disease in the used dataset.



Flowchart:



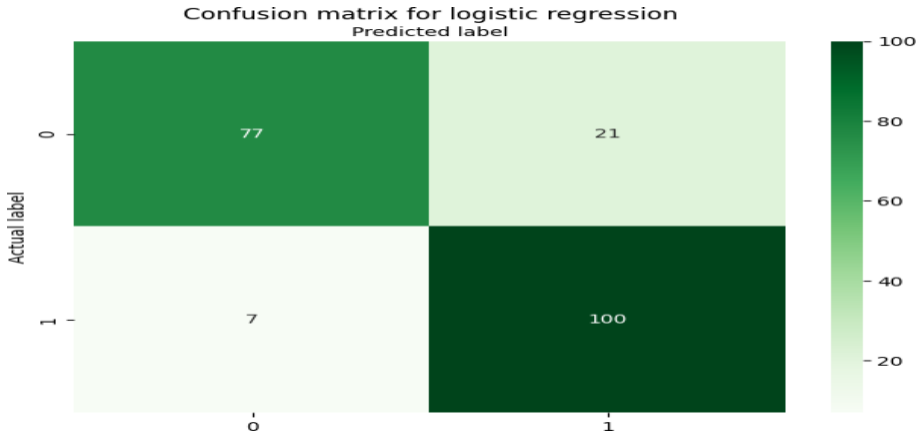

## 2.1 Logistic Regression

Logistic regression is a type of supervised learning in which the probabilityfor classification problems with two outcomes is computed. It can also be used to predict many classes. We used the sigmoid function in the LogisticRegression model,

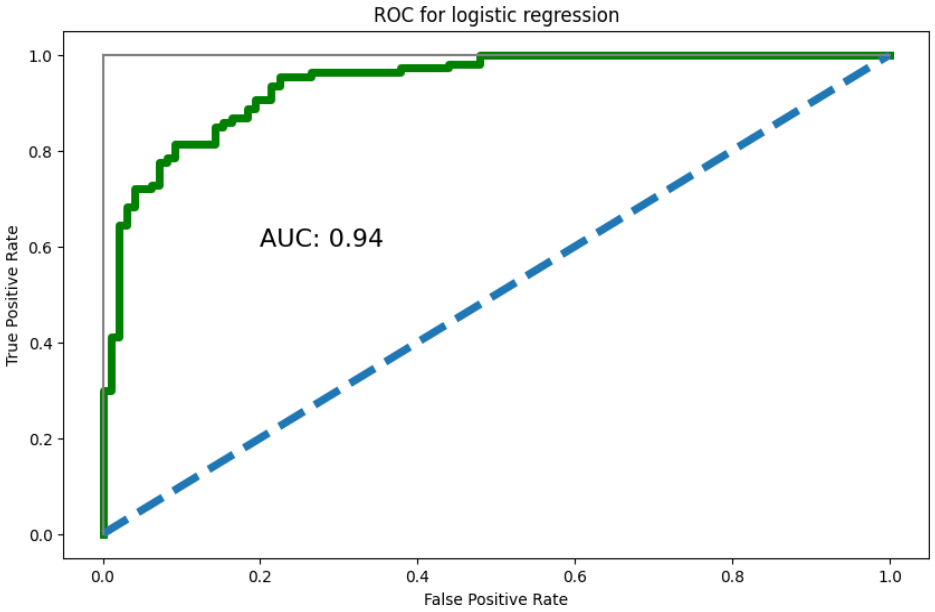
which is: $\boldsymbol{\sigma(z) = 1/1 + e-z}$

This function successfully changes any number into a value between 0 and 1, which we used to calculate the likelihood of correctly guessing classes. For example, there are two types of heart disease: one, those who have it, and the other, those who do not.

Confusion matrix:



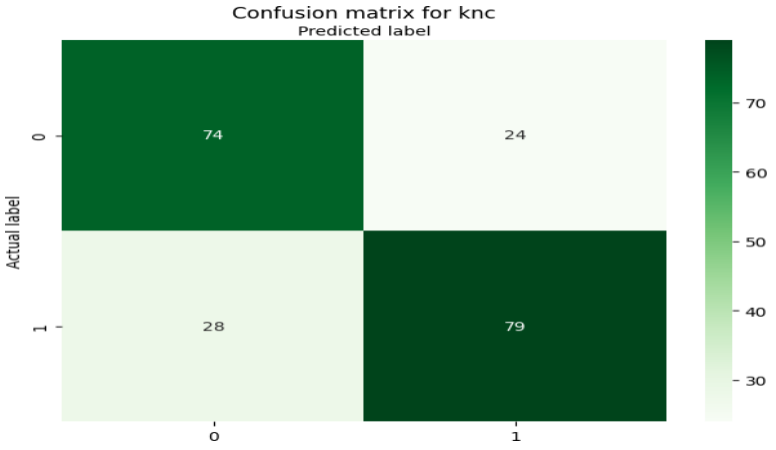
**ROC for Logistic Regression**



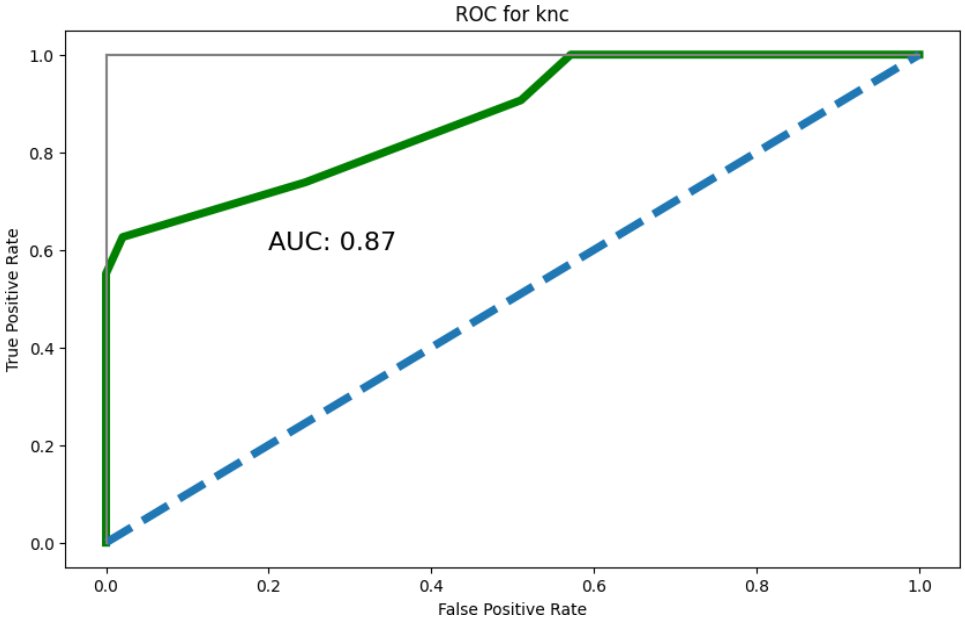

## 2.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a nonparametric, lazy, and basic classifier. When all of the characteristics are continuous, KNN is preferred. KNN, also known as case-based reasoning, has been employed in a wide range of applications, including pattern recognition and statistical estimation. To determine the class of an unknown sample, the nearest neighbor must be identified. Because of its rapid convergence speed and simplicity, KNN is favored over other categorization methods.

**Confusion matrix:**



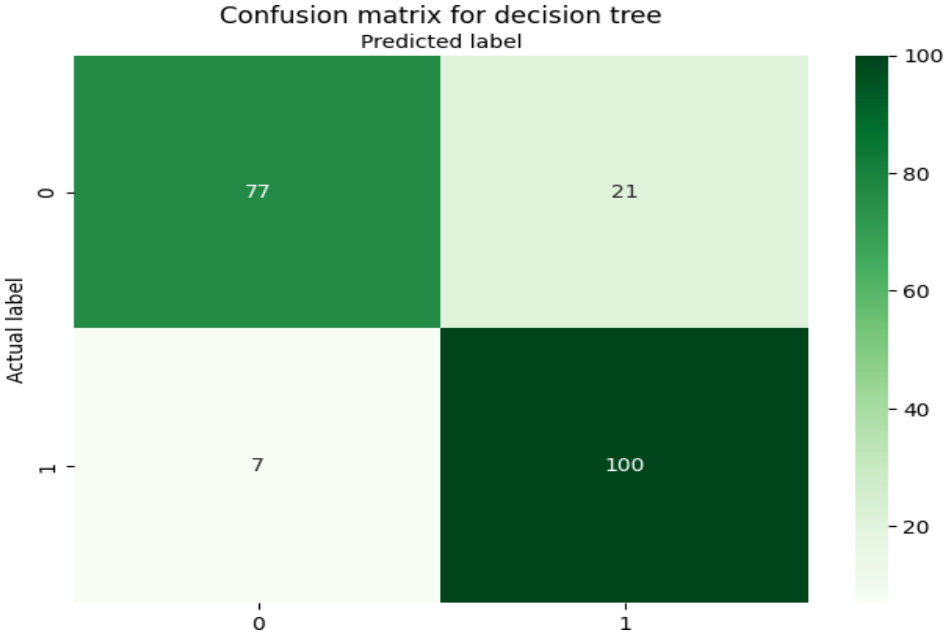
**ROC for KNN**



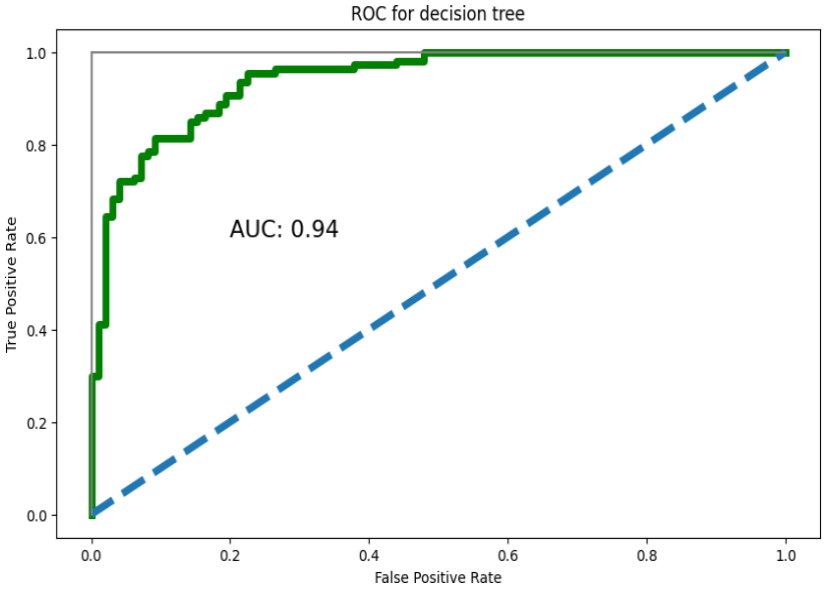

## 2.3 Decision Tree

decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Confusion matrix



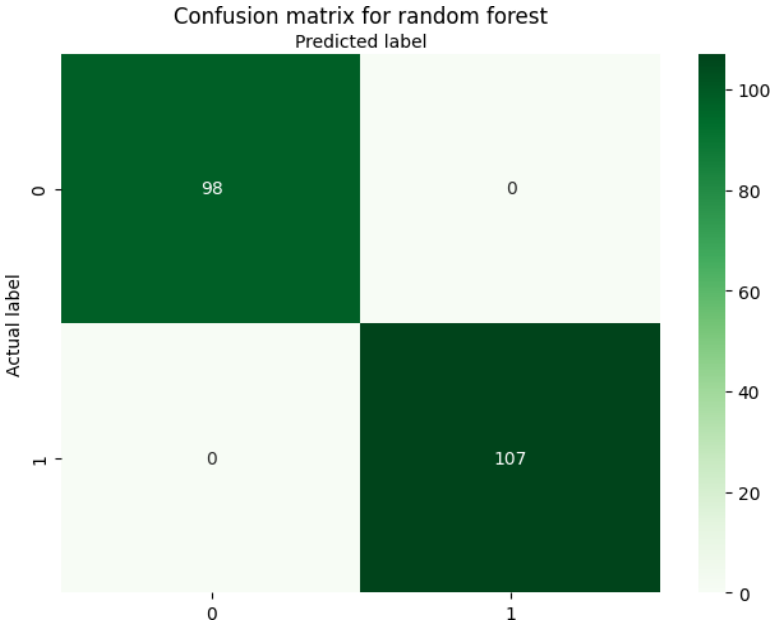
ROC for Decision Tree



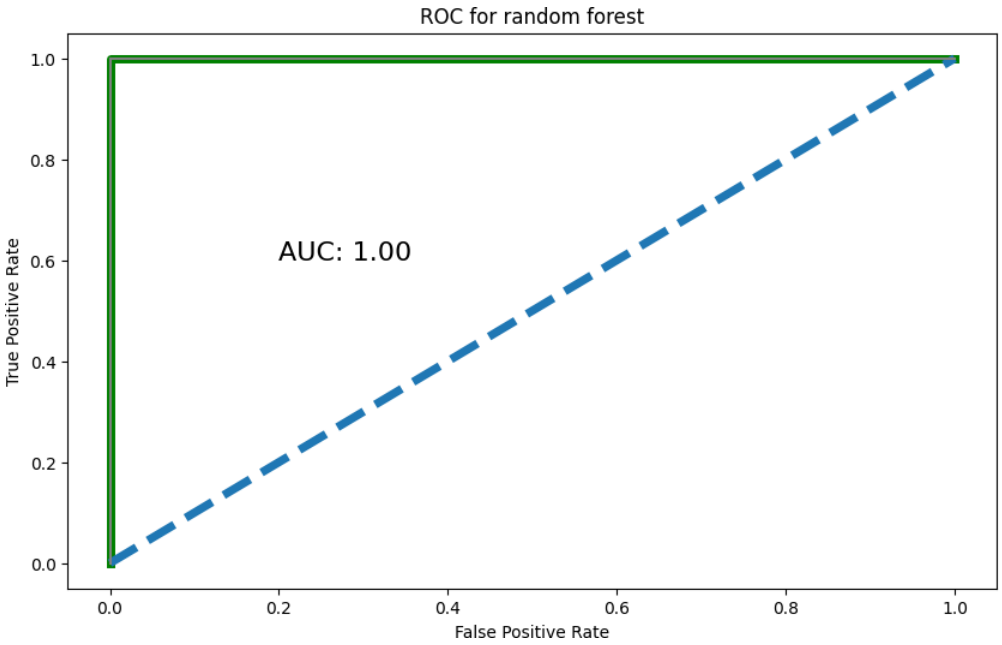

## 2.4 Random Forest

Random forest or random decision forests is an ensemble learning method for classification, regression, and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set.

Confusion Matrix



ROC for Random Forest




# Code:

```
# -*- coding: utf-8 -*-
"""Heart Prediction.ipynb
Automatically generated by Colaboratory.
Original file is located at
    https://colab.research.google.com/drive/10Uepm8Fziguv8Fdoqq3BRs_Efk_hy2o5
"""
# Commented out IPython magic to ensure Python compatibility.
import sys #access to system parameters
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
import matplotlib # collection of functions for scientific and publication-ready
visualization
import matplotlib.pyplot as plt
# %matplotlib inline
import seaborn as sns
import warnings # ignore warnings
warnings.filterwarnings('ignore')
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
```

```
        print(os.path.join(dirname, filename))
df = pd.read_csv('heart.csv')
df.shape
df.head(5)

df.target.value_counts() # df.target.unique()

df.info

disease = len(df[df['target'] == 1])
no_disease = len(df[df['target']== 0])

import matplotlib.pyplot as plt
plt.rcdefaults()
fig, ax = plt.subplots()
y = ('Heart Disease', 'No Disease')
y_pos = np.arange(len(y))
x = (disease, no_disease)
ax.barh(y_pos, x, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(y)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Count')
ax.set_title('Target')
for i, v in enumerate(x):
    ax.text(v + 10, i, str(v), color='black', va='center', fontweight='normal')
plt.show()
import matplotlib.pyplot as plt
y = ('Heart Disease', 'No Disease')
y_pos = np.arange(len(y))
x = (disease, no_disease)
labels = 'Heart Disease', 'No Disease'
sizes = [disease, no_disease]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Percentage of target', size=16)
plt.show() # Pie chart, where the slices will be ordered and plotted counter-clockwise:
```

```
df.isna().sum() # missing values

qualitative = []
quantitative = []
for feature in df.columns:
    if len(df[feature].unique()) <= 8:
        qualitative.append(feature)
    else:
        quantitative.append(feature)

qualitative

quantitative

top = 15
corr = df.corr()
top15 = corr.nlargest(top, 'target')['target'].index
corr_top15 = df[top15].corr()
f,ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr_top15, square=True, ax=ax, annot=True, cmap='coolwarm', fmt='.2f',
annot_kws={'size':12})
plt.title('Top correlated features of dataset', size=16)
plt.show()
ax = sns.distplot(df['thalach']) # histogram distribution
X = df.drop('target',1)
y = df['target']
print('shape of X and y respectively :', X.shape, y.shape)
sns.pairplot(df)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
print('shape of X and y respectively (train) :', X_train.shape, y_train.shape)
print('shape of X and y respectively (test) :', X_test.shape, y_test.shape)

print('Logistic Regression')
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
```

```
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
score = model.score(X_train, y_train)
print('Training Score:', score)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred}) # Heart-Disease yes or no? 1/0
print(output.head())
people = output.loc[output.Predicted == 1]["Predicted"]
rate_people = 0
if len(people) > 0 :
    rate_people = len(people)/len(output)
print("% of people predicted with heart-disease:", rate_people)
score_logreg = score
out_logreg = output
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,Y_pred)
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Greens', fmt = 'g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for logistic regression')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

from sklearn.metrics import roc_auc_score,roc_curve
y_probabilities = model.predict_proba(X_test)[:,1]
false_positive_rate_knn, true_positive_rate_knn, threshold_knn =
roc_curve(y_test,y_probabilities)
plt.figure(figsize=(10,6))
plt.title('ROC for logistic regression')
```

```
plt.plot(false_positive_rate_knn, true_positive_rate_knn, linewidth=5, color='green')
plt.plot([0,1],ls='--',linewidth=5)
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.text(0.2,0.6,'AUC: {:.2f}'.format(roc_auc_score(y_test,y_probabilities)),size= 16)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()


print('DecisionTreeClassifier')
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(max_depth=5)
decision_tree.fit(X_train, y_train)
Y_pred = model.predict(X_test)
score = model.score(X_train, y_train)
print('Training  Score:',  score)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred}) # Heart-Disease yes or no? 1/0
print(output.head())
people = output.loc[output.Predicted == 1]["Predicted"]
rate_people = 0
if len(people) > 0 :
    rate_people = len(people)/len(output)
print("% of people predicted with heart-disease:", rate_people)
score_dtc = score
out_dtc = output
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))

from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,Y_pred)
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Greens', fmt = 'g')
```

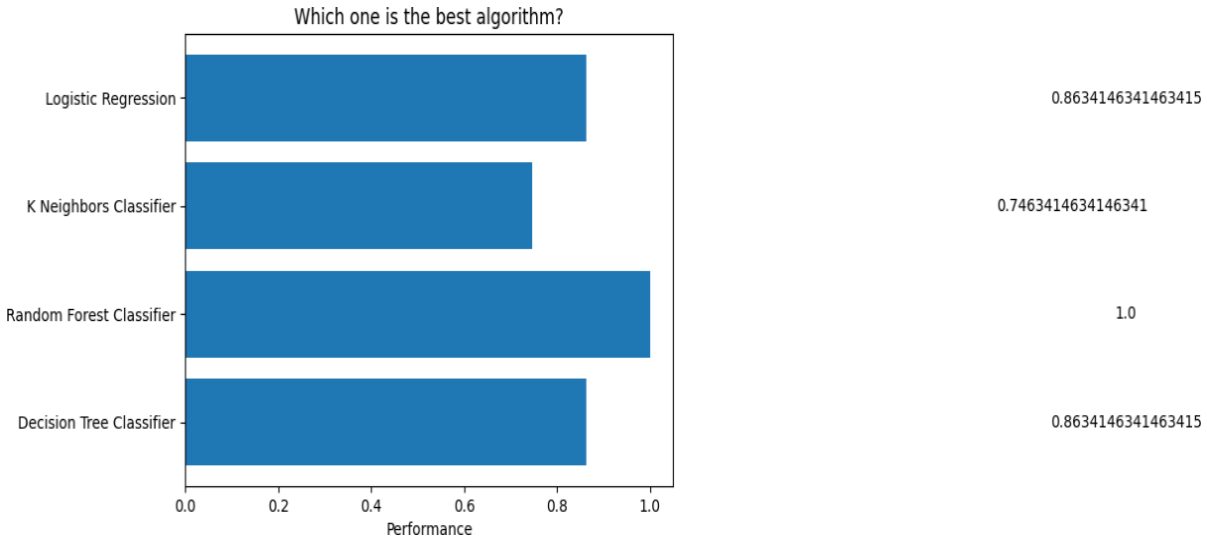
```
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for decision tree')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

from sklearn.metrics import roc_auc_score,roc_curve
y_probabilities = model.predict_proba(X_test)[:,1]
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test,y_probabilities)
plt.figure(figsize=(10,6))
plt.title('ROC for decision tree')
plt.plot(false_positive_rate, true_positive_rate, linewidth=5, color='green')
plt.plot([0,1],ls='--',linewidth=5)
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.text(0.2,0.6,'AUC: {:.2f}'.format(roc_auc_score(y_test,y_probabilities)),size= 16)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print('RandomForestClassifier')
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100) # , max_depth=5, random_state=1
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
score = model.score(X_train, y_train)
print('Training Score:', score)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred}) # Heart-Disease yes or no? 1/0
print(output.head())
people = output.loc[output.Predicted == 1]["Predicted"]
rate_people = 0
if len(people) > 0 :
    rate_people = len(people)/len(output)
print("% of people predicted with heart-disease:", rate_people)
score_rfc = score
```

```
out_rfc = output
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))

from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,Y_pred)
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Greens', fmt = 'g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for random forest')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

from sklearn.metrics import roc_auc_score,roc_curve
y_probabilities = model.predict_proba(X_test)[:,1]
false_positive_rate, true_positive_rate, threshold_knn = roc_curve(y_test,y_probabilities)
plt.figure(figsize=(10,6))
plt.title('ROC for random forest')
plt.plot(false_positive_rate, true_positive_rate, linewidth=5, color='green')
plt.plot([0,1],ls='--',linewidth=5)
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.text(0.2,0.6,'AUC: {:.2f}'.format(roc_auc_score(y_test,y_probabilities)),size= 16)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

print('KNeighborsClassifier')
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X_train, y_train)
Y_pred = model.predict(X_test)
```

```
score = model.score(X_train, y_train)
print('Training Score:', score_logreg)
score = model.score(X_test, y_test)
print('Testing Score:', score)
output = pd.DataFrame({'Predicted':Y_pred}) # Heart-Disease yes or no? 1/0
print(output.head())
people = output.loc[output.Predicted == 1]["Predicted"]
rate_people = 0
if len(people) > 0 :
    rate_people = len(people)/len(output)
print("% of people predicted with heart-disease:", rate_people)
score_knc = score
out_knc = output
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred))

from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test,Y_pred)
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
sns.heatmap(pd.DataFrame(confusion_matrix), annot = True, cmap = 'Greens', fmt = 'g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for knc')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

from sklearn.metrics import roc_auc_score,roc_curve
y_probabilities = model.predict_proba(X_test)[:,1]
false_positive_rate, true_positive_rate, threshold = roc_curve(y_test,y_probabilities)
plt.figure(figsize=(10,6))
plt.title('ROC for knc')
plt.plot(false_positive_rate, true_positive_rate, linewidth=5, color='green')
plt.plot([0,1],ls='--',linewidth=5)
```

```
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.text(0.2,0.6,'AUC: {:.2f}'.format(roc_auc_score(y_test,y_probabilities)),size= 16)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()

plt.rcdefaults()
fig, ax = plt.subplots()
algorithms = ('Logistic Regression', 'K Neighbors Classifier', 'Random Forest Classifier',
'Decision Tree Classifier')
y_pos = np.arange(len(algorithms))
x = (score_logreg, score_knc, score_rfc, score_dtc) # scores
ax.barh(y_pos, x, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(algorithms)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Performance')
ax.set_title('Which one is the best algorithm?')
for i, v in enumerate(x):
    ax.text(v + 1, i, str(v), color='black', va='center', fontweight='normal')
plt.show()
```

## Result:



## Conclusion:

Heart disease prediction is challenging and very important in the medical field. However, the mortality rate can be drastically controlled if the disease is detected at an early stage and preventivemeasures are adopted as soon as possible. The approach has combined the characteristics of the k means clustering decision tree and random forest. In this, we conclude that random forest gives more accurate results in the prediction of heart disease.

## References:


1] World Health Organization. (n.d.). Cardiovascular diseases. World Health Organization. https://www.who.int/health-topics/cardiovascular-diseases/

[2] Palaniappan,S.,& Awang, R. (2008).Intelligent heart disease prediction system usingdata mining techniques. 2008 IEEE/ACS International Conference on Computer SystemsandApplications. https://doi.org/10.1109/aiccsa.2008.4493524

[3] Dutta,A., Batabyal,T.Basu ,M., & Acton, S. T.(2020). An efficient convolutional neural network for coronary heart disease prediction. Expert Systems with Applications, 159, 113408. https://doi.org/10.1016/j.eswa.2020.113408

[4] Ronit, R. (2018, June 25). Heart Disease UCI.Kaggle. https://www.kaggle.com/ronitf/heart-disease-uci

[5] Janosi , A., Detrano , R., Pfisterer , M., & Steinbrunn , W. (1988). UCI Machine Learning Repository: Heart Disease Data Set. https://archive.ics.uci.edu/ml/datasets/Heart+Dis ease/.