

***Faculty of Science and Technology***

**Assignment Coversheet**

|  |  |
| --- | --- |
| **Student ID number &**  **Student Name** | U3242969  Pariwesh Subedi |
| **Unit name** | Software Technology 1 |
| **Unit number** | 4483 |
| **Unit Tutor** | Ms. Linda Ma |
| **Assignment name** | ST1 Capstone Project – Semester 1 2023 |
| **Due date** | Friday 23:55 |
| **Date submitted** | Friday 12th may 2023 |

**You must keep a photocopy or electronic copy of your assignment.**

**Student declaration**

I certify that the attached assignment is my own work. Material drawn from other sources has been appropriately and fully acknowledged as to author/creator, source and other bibliographic details.

**Signature of student: \_P.s\_\_\_\_\_\_\_\_\_\_ Date: \_\_2/05/23\_\_\_\_\_\_\_\_\_**

Table of Contents

[Introduction 1](#_Toc131590041)

[Methodology 2](#_Toc131590042)

[Stage 1: Algorithm Design Stage 2](#_Toc131590043)

[Dataset Description 3](#_Toc131590044)

[Exploratory Data Analysis 3](#_Toc131590045)

[Predictive Data Analytics Stage 11](#_Toc131590046)

[Model Preparation and Development 14](#_Toc131590047)

[Stage 2: Algorithm Implementation Stage 22](#_Toc131590048)

[Stage 3: Software Deployment Stage 22](#_Toc131590049)

[Conclusions 23](#_Toc131590050)

[References 23](#_Toc131590051)

# Introduction

This report describes the details of Python Capstone Project for ST1 unit within the scope of the project requirements provided in the assignment handout [1]. I/we have decided to work on the project using a coronary heart disease dataset available in both UCI [2] and Kaggle data repositories [3].

Coronary heart disease (CHD) also known as CAD (coronary artery disease), is atherosclerotic heart condition, where there is damage within the hearts major blood vessels. In Australia alone, in 2020, an estimated 56,700 people aged 25 and over had an acute coronary event in the form of a heart attack or unstable angina – around 155 events every day [4]. Of these, 6,900 (12%) were fatal. CHD/CAD involves the reduction of blood flow to the heart muscle due to build-up of plaque in the arteries of the heart. It is the most common form of cardiovascular disease.

Currently, invasive coronary angiography represents the gold standard for establishing the presence, location, and severity of CHD/CAD, however this diagnostic method is invasive, costly and associated with morbidity and mortality in patients. Therefore, it would be beneficial to develop a non-invasive alternative to replace the current gold standard. Other less invasive diagnostics methods have been proposed in the scientific literature including exercise electrocardiogram, thallium scintigraphy and fluoroscopy of coronary calcification. However, the diagnostic accuracy of these tests only ranges around 50%. Therefore, it would be beneficial to develop a software tool that could utilize the combined results of these non-invasive tests in conjunction with other patient attributes to boost the diagnostic power of these non-invasive methods with the aim to ultimately replace the current invasive gold standard.

This report presents the details of prototype software platform, in terms of several Python software tools developed as part of this capstone project, based on a data driven scientific approach, involving exploratory data analysis, predictive analytics and implementation as a desktop Tkinter application, and online web-based Flask/ Streamlit application. The details of the methodology used is presented in the next Section.

# Methodology

The methodology used for developing the software platform involves 3 stages as outlined below:

1. Design and development of decision support algorithms based on exploratory data analysis and predictive analytics, for identifying the best performing algorithm for solving a real world problem.
2. Implementation of best performing algorithm as a desktop Tkinter software tool.
3. Deployment of the tool as a web or cloud enabled platform tool.

## Stage 1: Algorithm Design Stage

Stage 1 is most important preliminary stage and depending on the complexity of the problem and dataset used, the design of algorithms for exploratory data analysis and predictive analytics algorithms will vary. However, the workflow for algorithm development will be as outlined in the Figure 1 schematic shown below:

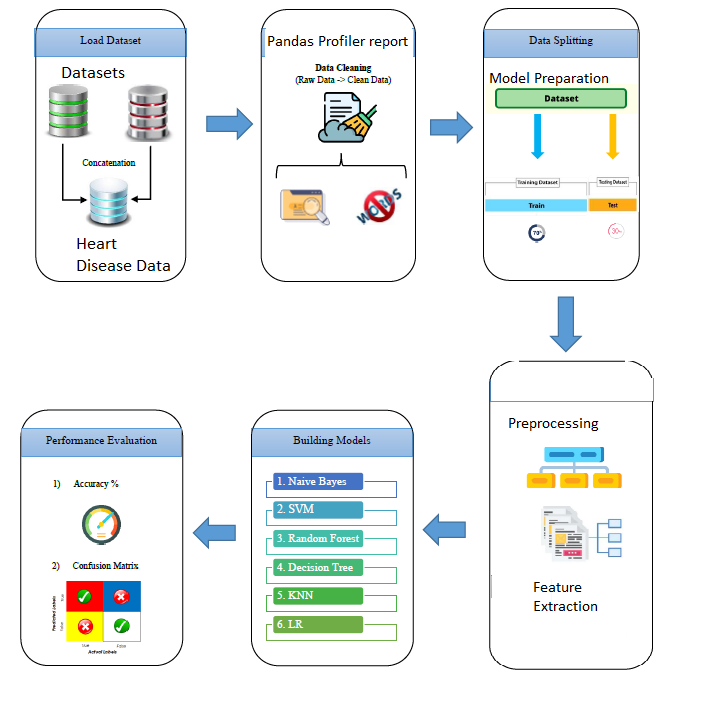


Figure 1: Schematic for Algorithm Design Methodology for Heart Disease Prediction

The details of each building block in Figure 1 schematic for algorithm design is described in the next few sections.

### Dataset Description

There is only one dataset used for this project and it is publicly available from both UCI machine learning repository [2], as well as Kaggle [3]. The dataset consists of 303 observations, 13 features and 1 target/class attribute. The 13 features include the results of the afore mentioned non-invasive diagnostic test results for a patient, along with other relevant patient information. The target/class variable includes the result of the invasive coronary angiogram which represents the presence or absence of coronary artery disease in the patient with 0 representing absence of heart disease and labels 1 representing presence of heart disease. Therefore, the task at hand is to develop a software tool to predict the result of the invasive coronary angiography, the current gold standard, using the results of the afore mentioned non-invasive tests and patient information. The data was originally collected by Robert Detrano, M.D., Ph.D of the Cleveland Clinic Foundation [2].

### Exploratory Data Analysis

The first phase of the software development activity involved understanding the data, basic exploratory data analysis and visualisation. Google Colab was chosen as the experimental environment as it incorporates virtual hardware and resources which does not require additional physical hardware requirement and can be ran directly of a web browser. The python language was used to create the scripts which ran directly on online Jupyter notebook using Google Colab with the help of free google account created, and by saving all the notebook files virtually on google drive without additional configurations. Before the exploratory data analysis can begin, some of the python libraries for EDA need to be imported and dataset acquired, by using the following Python script

from google.colab import drive

drive.mount("/content/drive")

#Import Required Packages for EDA

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

import plotly.graph\_objects as go

import plotly.express as px

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

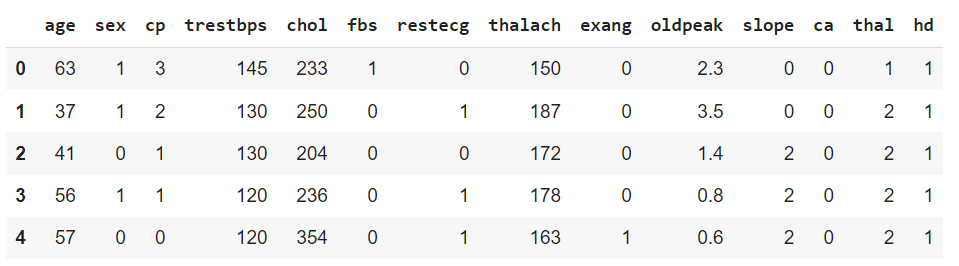
#Read the dataset/s

df = pd.read\_csv('/content/drive/…../heart.csv')

1. The EDA starts with understanding the basic description of data as described next:

#1. Checking description(first 5 and last 5 rows)

df.head()



df.tail() #last 5 rows

Table

Description automatically generated

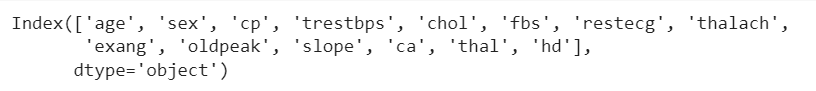
#rows and columns-data shape(attributes & samples)

df.shape

(303, 14)

# name of the attributes

df.columns



#unique values for each attribute

df.nunique()

Table

Description automatically generated

#Complete info about data frame

df.info()

Table

Description automatically generated

#3. Visualising data  distribution in detail

fig = plt.figure(figsize =(18,18))

ax=fig.gca()

df.hist(ax=ax,bins =30)

plt.show()

Chart, box and whisker chart

Description automatically generated

#detecting outliers

df.plot(kind='box', subplots=True,

        layout=(2,7),sharex=False,sharey=False, figsize=(20, 10), color='deeppink');

A picture containing chart

Description automatically generated

#identify the outliers

# define continuous variable & plot

continous\_features = ['age','trestbps','chol','thalach','oldpeak']

def outliers(df\_out, drop = False):

    for each\_feature in df\_out.columns:

        feature\_data = df\_out[each\_feature]

        Q1 = np.percentile(feature\_data, 25.) # 25th percentile of the data of the given feature

        Q3 = np.percentile(feature\_data, 75.) # 75th percentile of the data of the given feature

        IQR = Q3-Q1 #Interquartile Range

        outlier\_step = IQR \* 1.5 #That's we were talking about above

        outliers = feature\_data[~((feature\_data >= Q1 - outlier\_step) & (feature\_data <= Q3 + outlier\_step))].index.tolist()

        if not drop:

            print('For the feature {}, No of Outliers is {}'.format(each\_feature, len(outliers)))

        if drop:

            df.drop(outliers, inplace = True, errors = 'ignore')

            print('Outliers from {} feature removed'.format(each\_feature))

outliers(df[continous\_features])

Text

Description automatically generated

#drop the outliers

outliers(df[continous\_features], drop = True)

Outliers from age feature removed

Outliers from trestbps feature removed

Outliers from chol feature removed

Outliers from thalach feature removed

Outliers from oldpeak feature removed

#check if outliers got removed

df.plot(kind='box', subplots=True,

        layout=(2,7),sharex=False,sharey=False, figsize=(20, 10), color='deeppink');

A picture containing chart

Description automatically generated

#Check data shape after outlier removal

df.shape

(284, 14)

#checking target value distribution

print(df.hd.value\_counts())

fig, ax = plt.subplots(figsize=(5,4))

name = ["Disease", "No\_Disease"]

ax = df.hd.value\_counts().plot(kind='bar')

ax.set\_title("Heart Disease Classes", fontsize = 13, weight = 'bold')

ax.set\_xticklabels (name, rotation = 0)

# To calculate the percentage

totals = []

for i in ax.patches:

    totals.append(i.get\_height())

total = sum(totals)

for i in ax.patches:

    ax.text(i.get\_x()+.09, i.get\_height()-50, \

            str(round((i.get\_height()/total)\*100, 2))+'%', fontsize=14,

                color='white', weight = 'bold')

plt.tight\_layout()

Chart, bar chart

Description automatically generated

#check correlation between variables

sns.set(style="white")

plt.rcParams['figure.figsize'] = (15, 10)

sns.heatmap(df.corr(), annot = True, linewidths=.5, cmap="Blues")

plt.title('Corelation Between Variables', fontsize = 30)

plt.show()

A picture containing table

Description automatically generated

!pip install <https://github.com/pandas-profiling/pandas-profiling/archive/master.zip>

#obtain full profiler report

#restart kernel

#re-run import libraries and data

import pandas as pd

import numpy as np

from pandas\_profiling import ProfileReport

profile = ProfileReport(df,title="Heart Disease EDA",

                        html={'style':{'full\_width':True}})

profile.to\_notebook\_iframe()

Graphical user interface

Description automatically generated

### Predictive Data Analytics Stage

For predictive analytics, several processing steps are required. These include pre-processing, classifier comparison to identify the best machine learning classifier and performance evaluation with different objective metrics, such as accuracy, classification report, confusion matrix, ROC-AUC curve and prediction report was obtained using the Python scikit-learn package. Each of these steps are described next.

* Pre-processing: Since the dataset consists of a combination of continuous and categorical attributes/variables, there is a need to pre-process the data with attribute transformation, standardization and normalisation. We used scikit-learn’s OrdinalEncoder() function to perform attribute transformation.
* Normalisation of the independent values of the dataframe by was done by dropping the target from the dataframe, normalising it, and then reattaching the target to the dataframe:-

#pre-processing

from sklearn.exceptions import DataDimensionalityWarning

#encode object columns to integers

from sklearn import preprocessing

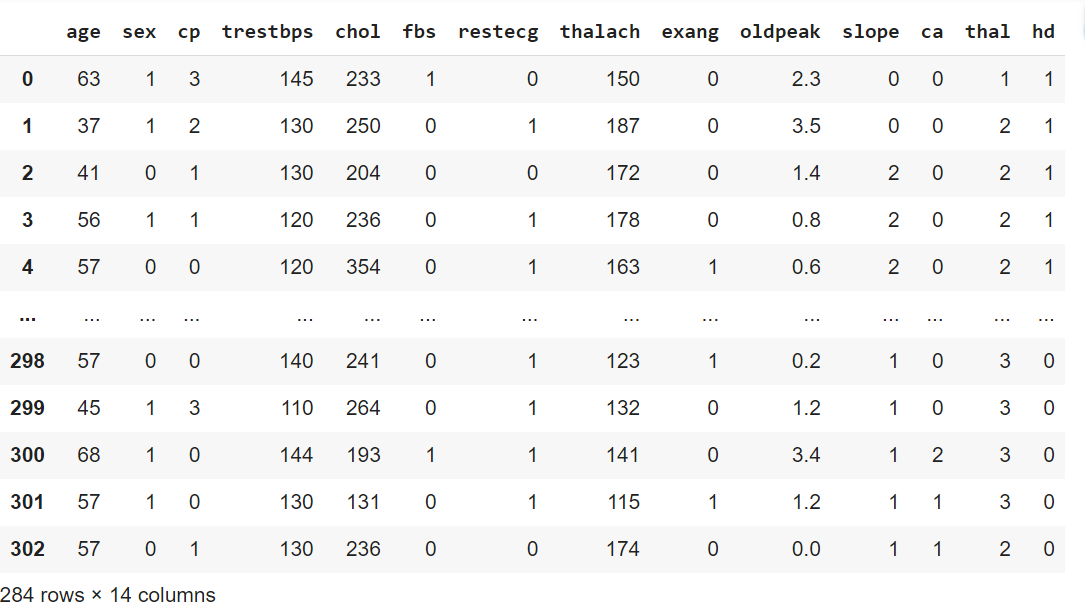
from sklearn.preprocessing import OrdinalEncoder

for col in df:

  if df[col].dtype =='object':

    df[col]=OrdinalEncoder().fit\_transform(df[col].values.reshape(-1,1))

df



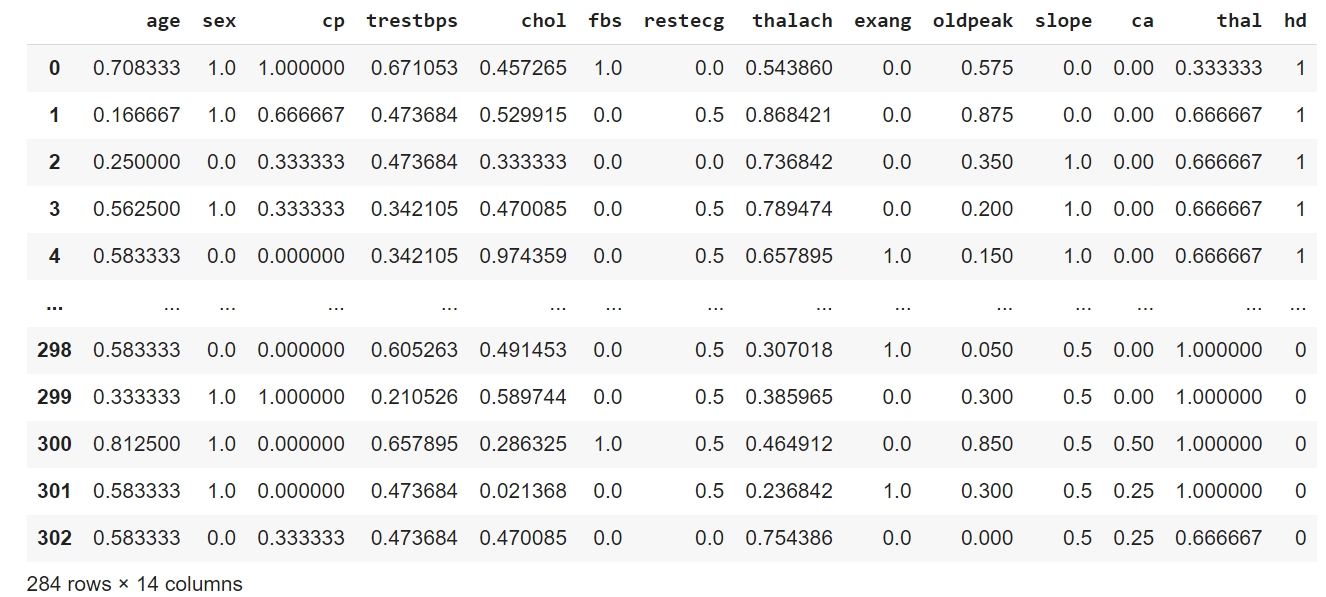
class\_label =df['hd']

df = df.drop(['hd'], axis =1)

df = (df-df.min())/(df.max()-df.min())

df['hd']=class\_label

df



#pre-processing

heart\_data = df.copy()

le = preprocessing.LabelEncoder()

age = le.fit\_transform(list(heart\_data["age"])) # age in years

sex = le.fit\_transform(list(heart\_data["sex"])) # gender (1 = male; 0 = female)

cp = le.fit\_transform(list(heart\_data["cp"])) # chest-pain and chest-pain type

trestbps = le.fit\_transform(list(heart\_data["trestbps"])) # resting blood pressure (mm/Hg)

chol = le.fit\_transform(list(heart\_data["chol"])) # serum cholestrol (mg/dl)

fbs = le.fit\_transform(list(heart\_data["fbs"])) # fasting blood sugar

restecg = le.fit\_transform(list(heart\_data["restecg"])) # resting elctrocardiographic results

thalach = le.fit\_transform(list(heart\_data["thalach"]))

exang = le.fit\_transform(list(heart\_data["exang"]))

oldpeak = le.fit\_transform(list(heart\_data["oldpeak"]))

slope = le.fit\_transform(list(heart\_data["slope"]))

ca = le.fit\_transform(list(heart\_data["ca"]))

thal = le.fit\_transform(list(heart\_data["thal"]))

hd = le.fit\_transform(list(heart\_data["hd"])) # heart-disease 0-not present 1-present

### Model Preparation and Development

Steps used for machine learning model preparation are described below:

* + Convert the dataframe to training and validation/test subsets by taking a random sample of 80% of the data and defining it as train subset. This leaves 20% of the data for validation/testing
  + Create the validation/test set by dropping all of the rows that comprise the training set from the dataframe.
  + Create y\_train by using using the last column of train (target class).
  + Create x\_train by using all of the columns in train except the last one.
  + The validation set of y\_val and x\_val or (y\_test and x\_test), can be created using the same methodology that used to create y\_train and x\_train

x = list(zip(age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal))

y = list(hd)

# Test options and evaluation metric

num\_folds = 5

seed = 7

scoring = 'accuracy'

# Model Test/Train

# Splitting what we are trying to predict into 4 different arrays -

# X train is a section of the x array(attributes) and vise versa for Y(features)

# The test data will test the accuracy of the model created

x\_train, x\_test, y\_train, y\_test = sklearn.model\_selection.train\_test\_split(x, y, test\_size = 0.20, random\_state=seed)

#splitting 20% of our data into test samples. If we train the model with higher data it already has seen that information and knows

#size of train and test subsets after splitting

np.shape(x\_train), np.shape(x\_test)

((227, 13), (57, 13))

# Predictive analytics model development by comparing different Scikit-learn classification algorithms

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import KFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import GridSearchCV

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import accuracy\_score

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import ExtraTreesClassifier

models = []

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

# evaluate each model in turn

results = []

names = []

print("Performance on Training set")

for name, model in models:

  kfold = KFold(n\_splits=num\_folds,shuffle=True,random\_state=seed)

  cv\_results = cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')

  results.append(cv\_results)

  names.append(name)

  msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

  msg += '\n'

  print(msg)

Performance on Training set

DT: 0.717874 (0.065709)

NB: 0.841256 (0.042955)

SVM: 0.718357 (0.040469)

GBM: 0.828502 (0.049942)

RF: 0.819324 (0.081332)

# Compare Algorithms' Performance

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

Chart, box and whisker chart

Description automatically generated

#Model Evaluation by testing with independent/external test data set.

# Make predictions on validation/test dataset

#Model Evaluation by testing with independent/external test data set.

# Make predictions on validation/test dataset

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

models.append(('GBM', GradientBoostingClassifier()))

models.append(('RF', RandomForestClassifier()))

dt = DecisionTreeClassifier()

nb = GaussianNB()

gb = GradientBoostingClassifier()

rf = RandomForestClassifier()

best\_model = rf

best\_model.fit(x\_train, y\_train)

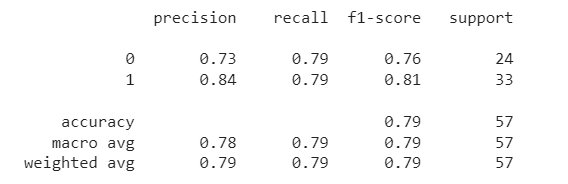
y\_pred = best\_model.predict(x\_test)

print("Best Model Accuracy Score on Test Set:", accuracy\_score(y\_test, y\_pred))

Best Model Accuracy Score on Test Set: 0.7894736842105263

#Model Performance Evaluation Metric 1 - Classification Report

print(classification\_report(y\_test, y\_pred))



#Model Performance Evaluation Metric 2

#Confusion matrix

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

cm = confusion\_matrix(y\_test, y\_pred)

disp = ConfusionMatrixDisplay(confusion\_matrix=cm)

disp.plot()

plt.show()

Chart, square

Description automatically generated

#Model Evaluation Metric 3- ROC-AUC curve

from sklearn.metrics import roc\_auc\_score

from sklearn.metrics import roc\_curve

best\_model = rf

best\_model.fit(x\_train, y\_train)

rf\_roc\_auc = roc\_auc\_score(y\_test,best\_model.predict(x\_test))

fpr,tpr,thresholds = roc\_curve(y\_test, best\_model.predict\_proba(x\_test)[:,1])

plt.figure()

plt.plot(fpr,tpr,label = 'Random Forest(area = %0.2f)'% rf\_roc\_auc)

plt.plot([0,1],[0,1],'r--')

plt.xlim([0.0,1.0])

plt.ylim([0.0,1.05])

plt.xlabel('False positive rate')

plt.ylabel('True positive rate')

plt.title('Receiver Operating Characteristic')

plt.legend(loc='lower right')

plt.savefig('LOC\_ROC')

plt.show()

Chart, line chart

Description automatically generated

#Model Evaluation Metric 4-prediction report

for x in range(len(y\_pred)):

  print("Predicted: ", y\_pred[x], "Actual: ", y\_test[x], "Data: ", x\_test[x],)

Predicted: 0 Actual: 0 Data: (18, 1, 0, 9, 59, 0, 1, 57, 0, 0, 2, 1, 2)

Predicted: 1 Actual: 1 Data: (24, 0, 2, 14, 137, 0, 1, 69, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (12, 1, 2, 34, 60, 0, 1, 44, 0, 33, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (8, 0, 2, 14, 39, 0, 1, 70, 0, 0, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (19, 0, 2, 20, 46, 0, 0, 15, 0, 0, 2, 0, 0)

Predicted: 1 Actual: 1 Data: (8, 1, 2, 22, 16, 0, 1, 47, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (26, 0, 0, 34, 84, 0, 0, 54, 0, 25, 1, 2, 3)

Predicted: 0 Actual: 0 Data: (32, 1, 0, 9, 42, 0, 0, 30, 1, 1, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 1, 14, 107, 0, 0, 57, 0, 17, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (29, 1, 0, 28, 22, 0, 0, 41, 1, 35, 2, 2, 3)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 32, 47, 0, 1, 7, 0, 19, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (33, 1, 0, 14, 58, 0, 0, 27, 1, 25, 1, 2, 3)

Predicted: 0 Actual: 0 Data: (23, 1, 0, 41, 110, 1, 0, 22, 0, 10, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (21, 0, 1, 25, 77, 0, 0, 58, 0, 14, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (5, 1, 0, 13, 48, 0, 1, 37, 0, 12, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (18, 1, 0, 9, 59, 0, 1, 57, 0, 0, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 1, 63, 0, 1, 53, 0, 1, 2, 1, 3)

Predicted: 0 Actual: 0 Data: (30, 1, 2, 28, 136, 0, 1, 55, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (17, 1, 0, 28, 115, 0, 1, 20, 1, 36, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (30, 0, 0, 22, 119, 0, 1, 20, 0, 19, 1, 2, 2)

Predicted: 0 Actual: 0 Data: (29, 1, 0, 22, 80, 0, 0, 44, 0, 14, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (18, 0, 2, 26, 27, 0, 0, 66, 0, 1, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (23, 1, 0, 28, 24, 0, 1, 45, 0, 4, 1, 0, 1)

Predicted: 0 Actual: 0 Data: (27, 0, 0, 31, 123, 0, 0, 43, 1, 10, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (17, 1, 0, 28, 87, 0, 0, 81, 1, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (28, 0, 2, 22, 89, 0, 1, 4, 0, 12, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (20, 1, 1, 7, 125, 0, 1, 53, 0, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (6, 1, 0, 35, 52, 0, 1, 77, 0, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (31, 1, 0, 8, 75, 0, 0, 55, 0, 6, 2, 2, 1)

Predicted: 1 Actual: 1 Data: (4, 1, 2, 27, 12, 0, 1, 70, 0, 0, 2, 4, 2)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 8, 66, 0, 1, 24, 1, 26, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (14, 1, 1, 22, 72, 0, 0, 76, 0, 2, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (26, 1, 0, 22, 79, 0, 1, 41, 1, 14, 2, 1, 3)

Predicted: 0 Actual: 0 Data: (33, 1, 0, 18, 80, 1, 1, 60, 0, 2, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (36, 1, 1, 38, 72, 0, 0, 40, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (14, 1, 0, 17, 99, 0, 0, 63, 0, 5, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (6, 1, 3, 28, 30, 0, 1, 74, 1, 14, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (9, 1, 0, 11, 119, 0, 1, 77, 0, 12, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (12, 1, 2, 34, 60, 0, 1, 44, 0, 33, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (7, 1, 1, 25, 33, 0, 1, 30, 0, 0, 1, 0, 1)

Predicted: 1 Actual: 1 Data: (9, 0, 2, 15, 43, 0, 1, 62, 0, 2, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (24, 1, 2, 5, 67, 0, 0, 51, 1, 6, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (11, 1, 0, 29, 125, 0, 0, 44, 1, 0, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (18, 1, 3, 13, 21, 0, 0, 84, 0, 0, 1, 0, 1)

Predicted: 1 Actual: 1 Data: (39, 0, 2, 28, 28, 0, 2, 16, 0, 11, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (17, 1, 2, 8, 12, 0, 1, 21, 0, 6, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (26, 1, 0, 22, 36, 0, 0, 30, 1, 23, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (31, 1, 0, 14, 14, 0, 1, 37, 0, 4, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (25, 1, 0, 27, 97, 0, 0, 78, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (0, 1, 1, 22, 34, 0, 0, 87, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (18, 1, 3, 35, 115, 1, 1, 74, 0, 12, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (26, 1, 0, 28, 112, 0, 0, 67, 0, 12, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (0, 1, 1, 22, 34, 0, 0, 87, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (8, 1, 1, 14, 114, 0, 1, 59, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 34, 96, 0, 0, 11, 1, 8, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (22, 1, 1, 14, 67, 0, 1, 66, 0, 0, 0, 0, 2)

Predicted: 1 Actual: 1 Data: (19, 0, 0, 22, 90, 0, 0, 40, 0, 4, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (34, 0, 2, 14, 41, 0, 0, 15, 0, 15, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (12, 1, 0, 28, 126, 0, 1, 18, 1, 17, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (29, 0, 2, 25, 78, 0, 0, 69, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (11, 0, 1, 9, 5, 0, 1, 35, 0, 0, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (20, 1, 2, 14, 84, 0, 0, 44, 0, 4, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (7, 1, 1, 14, 4, 0, 1, 78, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (29, 1, 0, 22, 135, 1, 0, 30, 1, 17, 2, 3, 3)

Predicted: 1 Actual: 1 Data: (17, 0, 2, 22, 82, 0, 0, 46, 0, 5, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (33, 1, 0, 39, 108, 0, 0, 9, 1, 15, 1, 3, 2)

Predicted: 0 Actual: 0 Data: (23, 1, 0, 22, 1, 0, 1, 15, 1, 12, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (21, 1, 0, 23, 140, 0, 1, 30, 1, 12, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (30, 0, 0, 22, 119, 0, 1, 20, 0, 19, 1, 2, 2)

Predicted: 0 Actual: 0 Data: (25, 1, 3, 24, 34, 0, 1, 59, 0, 8, 2, 2, 2)

Predicted: 0 Actual: 0 Data: (25, 1, 2, 19, 47, 1, 1, 32, 0, 21, 1, 1, 1)

Predicted: 1 Actual: 1 Data: (5, 1, 2, 28, 131, 0, 0, 78, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (7, 0, 1, 19, 122, 0, 1, 60, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (26, 1, 0, 22, 79, 0, 1, 41, 1, 14, 2, 1, 3)

Predicted: 0 Actual: 0 Data: (6, 1, 0, 8, 7, 0, 0, 14, 1, 19, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (10, 1, 2, 22, 62, 0, 1, 75, 1, 4, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (14, 1, 0, 22, 82, 1, 0, 47, 1, 0, 2, 2, 3)

Predicted: 1 Actual: 1 Data: (32, 0, 2, 32, 103, 0, 0, 49, 0, 0, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (10, 1, 1, 22, 48, 0, 0, 83, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (10, 1, 1, 14, 89, 0, 1, 70, 0, 0, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (22, 1, 1, 22, 50, 0, 0, 60, 0, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (22, 1, 0, 22, 106, 1, 0, 6, 1, 16, 0, 0, 3)

Predicted: 0 Actual: 0 Data: (23, 1, 1, 17, 87, 0, 1, 38, 0, 3, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (32, 0, 2, 32, 103, 0, 0, 49, 0, 0, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (32, 0, 3, 34, 55, 0, 1, 14, 0, 25, 0, 0, 2)

Predicted: 1 Actual: 1 Data: (9, 1, 2, 22, 128, 0, 1, 59, 0, 18, 2, 1, 2)

Predicted: 1 Actual: 1 Data: (24, 1, 1, 18, 49, 0, 1, 41, 0, 4, 1, 4, 3)

Predicted: 0 Actual: 0 Data: (31, 1, 3, 27, 105, 1, 0, 71, 0, 14, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (11, 0, 0, 27, 65, 0, 0, 49, 1, 2, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (31, 0, 0, 34, 54, 0, 0, 14, 0, 10, 1, 3, 3)

Predicted: 0 Actual: 0 Data: (8, 1, 0, 26, 128, 0, 1, 23, 1, 17, 1, 0, 1)

Predicted: 1 Actual: 1 Data: (17, 1, 0, 28, 87, 0, 0, 81, 1, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (20, 0, 2, 39, 32, 0, 1, 60, 0, 0, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (36, 1, 0, 22, 132, 0, 0, 10, 0, 23, 1, 3, 2)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 8, 36, 0, 0, 9, 1, 0, 1, 1, 2)

Predicted: 0 Actual: 0 Data: (13, 1, 0, 8, 100, 0, 0, 17, 1, 10, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (20, 0, 2, 7, 93, 0, 0, 64, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 2, 9, 59, 0, 0, 62, 0, 24, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (23, 0, 0, 28, 68, 0, 1, 21, 1, 2, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 8, 66, 0, 1, 24, 1, 26, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (8, 1, 2, 22, 16, 0, 1, 47, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (8, 0, 2, 14, 39, 0, 1, 70, 0, 0, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (20, 0, 2, 39, 32, 0, 1, 60, 0, 0, 2, 1, 2)

Predicted: 1 Actual: 1 Data: (25, 1, 0, 27, 97, 0, 0, 78, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (24, 0, 0, 1, 75, 0, 0, 20, 0, 10, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (19, 1, 0, 29, 55, 0, 0, 11, 1, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 15, 108, 0, 0, 16, 1, 30, 1, 2, 2)

Predicted: 1 Actual: 1 Data: (35, 0, 3, 28, 66, 0, 1, 48, 0, 17, 2, 2, 2)

Predicted: 0 Actual: 0 Data: (16, 1, 2, 28, 62, 0, 1, 60, 0, 6, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (32, 0, 2, 32, 103, 0, 0, 49, 0, 0, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (17, 1, 2, 8, 12, 0, 1, 21, 0, 6, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (26, 1, 0, 12, 59, 1, 1, 57, 1, 14, 2, 2, 3)

Predicted: 0 Actual: 0 Data: (17, 1, 0, 28, 116, 0, 1, 70, 1, 16, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (13, 1, 2, 22, 79, 0, 1, 75, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (12, 0, 2, 29, 14, 0, 0, 57, 1, 14, 0, 0, 2)

Predicted: 1 Actual: 1 Data: (27, 1, 2, 34, 70, 1, 1, 34, 1, 10, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (27, 1, 0, 27, 6, 0, 0, 23, 1, 33, 1, 1, 2)

Predicted: 0 Actual: 0 Data: (28, 1, 0, 14, 93, 0, 1, 5, 1, 17, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (22, 0, 1, 28, 113, 0, 0, 50, 0, 13, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (25, 1, 3, 42, 109, 0, 0, 56, 0, 2, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (18, 1, 0, 20, 81, 0, 1, 58, 1, 0, 2, 1, 3)

Predicted: 1 Actual: 1 Data: (7, 0, 2, 9, 94, 0, 0, 69, 1, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (32, 1, 0, 14, 118, 0, 0, 48, 0, 4, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 8, 36, 0, 0, 9, 1, 0, 1, 1, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 10, 129, 0, 2, 37, 0, 37, 0, 3, 1)

Predicted: 1 Actual: 1 Data: (12, 0, 0, 27, 70, 0, 0, 49, 1, 0, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (15, 1, 2, 14, 23, 0, 1, 36, 0, 19, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (15, 1, 1, 22, 92, 0, 1, 68, 0, 6, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (6, 1, 0, 35, 52, 0, 1, 77, 0, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (12, 1, 0, 28, 126, 0, 1, 18, 1, 17, 1, 2, 3)

Predicted: 1 Actual: 1 Data: (33, 0, 0, 6, 52, 0, 1, 39, 0, 3, 2, 2, 2)

Predicted: 0 Actual: 0 Data: (27, 0, 0, 31, 123, 0, 0, 43, 1, 10, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 14, 23, 0, 1, 13, 0, 14, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (10, 1, 0, 8, 28, 0, 0, 73, 0, 0, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (23, 1, 2, 20, 58, 0, 0, 47, 0, 4, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (30, 0, 2, 28, 127, 0, 1, 31, 0, 2, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (24, 0, 0, 1, 75, 0, 0, 20, 0, 10, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (15, 0, 0, 22, 95, 0, 1, 60, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (36, 1, 0, 22, 132, 0, 0, 10, 0, 23, 1, 3, 2)

Predicted: 1 Actual: 1 Data: (7, 1, 1, 25, 33, 0, 1, 30, 0, 0, 1, 0, 1)

Predicted: 1 Actual: 1 Data: (11, 0, 1, 22, 63, 0, 0, 72, 0, 6, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (22, 0, 1, 28, 113, 0, 0, 50, 0, 13, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (24, 0, 0, 42, 54, 1, 0, 43, 1, 26, 1, 2, 1)

Predicted: 0 Actual: 0 Data: (20, 1, 0, 17, 92, 0, 0, 10, 1, 21, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 20, 46, 0, 0, 29, 1, 21, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (37, 0, 2, 8, 91, 1, 0, 28, 0, 0, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (24, 0, 1, 26, 130, 1, 0, 49, 0, 0, 2, 2, 2)

Predicted: 0 Actual: 0 Data: (35, 1, 2, 28, 80, 0, 0, 43, 0, 19, 1, 3, 3)

Predicted: 0 Actual: 0 Data: (19, 1, 0, 16, 105, 0, 1, 2, 1, 19, 1, 2, 3)

Predicted: 0 Actual: 0 Data: (36, 1, 0, 22, 132, 0, 0, 10, 0, 23, 1, 3, 2)

Predicted: 0 Actual: 0 Data: (24, 1, 0, 32, 47, 0, 1, 7, 0, 19, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (24, 0, 2, 14, 137, 0, 1, 69, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (20, 1, 0, 28, 66, 0, 1, 57, 0, 12, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (32, 1, 0, 14, 118, 0, 0, 48, 0, 4, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (27, 1, 3, 24, 63, 0, 1, 42, 0, 25, 1, 2, 2)

Predicted: 1 Actual: 1 Data: (8, 1, 0, 28, 55, 0, 1, 74, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (10, 1, 1, 14, 49, 0, 1, 67, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (25, 1, 0, 8, 66, 0, 0, 39, 1, 12, 1, 1, 3)

Predicted: 0 Actual: 0 Data: (15, 1, 2, 14, 23, 0, 1, 36, 0, 19, 1, 3, 3)

Predicted: 1 Actual: 1 Data: (24, 0, 3, 34, 106, 1, 0, 59, 0, 10, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (18, 1, 1, 24, 32, 0, 1, 55, 0, 8, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (28, 0, 0, 34, 71, 0, 1, 51, 1, 14, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (22, 1, 1, 22, 50, 0, 0, 60, 0, 0, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (13, 1, 2, 7, 70, 0, 1, 49, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (17, 0, 2, 22, 82, 0, 0, 46, 0, 5, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (18, 0, 2, 26, 27, 0, 0, 66, 0, 1, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (32, 1, 0, 39, 57, 0, 0, 35, 0, 22, 2, 0, 1)

Predicted: 1 Actual: 1 Data: (5, 0, 2, 27, 49, 0, 1, 49, 0, 0, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (26, 0, 2, 14, 15, 1, 1, 3, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (9, 1, 0, 34, 74, 0, 1, 68, 0, 15, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (2, 1, 1, 15, 24, 0, 1, 71, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (2, 1, 0, 19, 105, 0, 0, 53, 1, 0, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (13, 1, 2, 27, 83, 0, 0, 53, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (24, 0, 2, 14, 137, 0, 1, 69, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (16, 0, 0, 8, 80, 0, 0, 56, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (16, 1, 0, 34, 70, 0, 0, 26, 0, 25, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (20, 1, 1, 7, 125, 0, 1, 53, 0, 0, 2, 0, 3)

Predicted: 1 Actual: 1 Data: (20, 0, 2, 8, 44, 0, 1, 55, 0, 16, 1, 0, 2)

Predicted: 0 Actual: 0 Data: (29, 0, 0, 7, 95, 0, 1, 66, 1, 17, 1, 2, 2)

Predicted: 1 Actual: 1 Data: (20, 1, 2, 34, 61, 0, 0, 62, 0, 16, 2, 0, 3)

Predicted: 0 Actual: 0 Data: (25, 1, 3, 42, 109, 0, 0, 56, 0, 2, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (10, 1, 1, 14, 49, 0, 1, 67, 0, 0, 2, 0, 2)

Predicted: 1 Actual: 1 Data: (9, 1, 2, 22, 128, 0, 1, 59, 0, 18, 2, 1, 2)

Predicted: 1 Actual: 1 Data: (7, 0, 1, 22, 34, 0, 0, 69, 0, 14, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (25, 1, 3, 42, 109, 0, 0, 56, 0, 2, 1, 0, 3)

Predicted: 1 Actual: 1 Data: (25, 1, 0, 25, 63, 0, 1, 58, 0, 5, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (24, 1, 2, 9, 59, 0, 0, 62, 0, 24, 1, 1, 3)

Predicted: 1 Actual: 1 Data: (24, 1, 2, 5, 67, 0, 0, 51, 1, 6, 1, 0, 3)

Predicted: 0 Actual: 0 Data: (18, 1, 0, 9, 59, 0, 1, 57, 0, 0, 2, 1, 2)

Predicted: 0 Actual: 0 Data: (23, 1, 1, 36, 61, 0, 0, 61, 0, 0, 2, 1, 2)

Predicted: 1 Actual: 1 Data: (21, 0, 1, 25, 77, 0, 0, 58, 0, 14, 1, 0, 2)

Predicted: 1 Actual: 1 Data: (36, 1, 1, 38, 72, 0, 0, 40, 0, 0, 2, 0, 2)

Predicted: 0 Actual: 0 Data: (13, 1, 0, 8, 100, 0, 0, 17, 1, 10, 1, 1, 2)

Predicted: 1 Actual: 1 Data: (22, 1, 1, 22, 50, 0, 0, 60, 0, 0, 2, 0, 3)

## Stage 2: Algorithm Implementation Stage

1. Implementation of best performing algorithm as a desktop Tkinter software tool.
2. Deployment of the tool as a web or cloud enabled platform tool.

Once the best performing algorithm and machine learning model for heart disease prediction has been identified from stage 1, the implementation of the algorithm as a desktop software tool using python Tkinter package.

The Pycharm project for the implementation is available at this google drive link:

<https://drive.google.com/drive/folders/1b2LyirWXzKWBCBn6m0eyTz_VNjI_Bwy4?usp=share_link>

## Stage 3: Software Deployment Stage

The deployment of software as a desktop tool as in stage 2, limits its applicability and does not allow wider usage by all care team involved in the managing the chronic disease for the patient. Hence there is a need to deploy this software as a web based tool or cloud based tool. The deployment of heart disease prediction as a web based platform was done using Flask API, a widely used micro web framework for creating APIs in Python. Flask is a simple yet powerful web framework in Python, with an the ability to scale up to complex applications. The Flask project deployment for the Heart Disease prediction is available at this google drive link:

<https://drive.google.com/drive/folders/1b2LyirWXzKWBCBn6m0eyTz_VNjI_Bwy4?usp=share_link>

# Conclusions

This report presents the work done towards the ST1 capstone project for design, development, implementation and deployment of data driven heart disease prediction software platform using Python. This platform allows leveraging the studies of non-invasive clinical tests run on patients, and building a data driven disease prediction platform, where the historical data and insight about a problem or an event can be used to predict the future risk or probability of the event. As can be seen from the outcomes of this project, we can train models that can predict coronary heart disease with substantial agreement with the results of invasive coronary angiography. The availability of predictive analytics tools as both desktop software tool and a web based tool, allows the wider application of this project to clinical, non-clinical and allied health care team for managing the heart disease for the patients.

## References

1. <https://uclearn.canberra.edu.au/courses/13571/assignments/105232>
2. Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J., Sandhu, S., Guppy, K., Lee, S., & Froelicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. American Journal of Cardiology, 64,304--310. <https://archive.ics.uci.edu/ml/datasets/heart+disease>
3. <https://www.kaggle.com/datasets/aavigan/cleveland-clinic-heart-disease-dataset>
4. <https://www.aihw.gov.au/reports/heart-stroke-vascular-diseases/hsvd-facts/contents/summary-of-coronary-heart-disease-and-stroke/coronary-heart-disease>
5. <https://towardsdatascience.com/creating-restful-apis-using-flask-and-python-655bad51b24>