PROJECT REPORT

ON

HEART DISEASE DIAGNOSIS

in partial fulfillment for the award of the degree of

MASTER'S IN SCIENCE(MS)

IN

COMPUTER SCIENCE (CS)

At

PACE UNIVERSITY

Submitted To:
Prof. Shubham Sawant
(Adjunct Faculty)

Submitted by:
Tejaswini Kandyala
Manognya Raygir
Nikhil Katakam
Drona

Contents

Acknowledgement:

This section is the best section since it allows to give personal acknowledgment to those who helped me and this project bringing it to the stage it is at. “The dream begins with a teacher who believes in you, who tugs and pushes and leads you to the next plateau, sometimes poking you with a sharp stick called ‘truth’”.

The above quotes specifies that teachers are the ones who make you to see a dream and motivates you so when you feel love and get struck in middle,they will help you in getting out of it. So, I would like thank Teachers: -

To Prof. Cha for all his encouragement and appreciation that has given us and this project the much-needed enthusiasm, strength, and confidence. He has been more of a friend then a teacher and has motivated throughout the project.

As we say that “Little things matter much” we would also like to thank our friends who helped us in their own ways they could do. They had always been there to inspire us, and all help we needed.

Finally,to all those who have rendered help to this project directly as well as indirectly, a little word with a never-ending meaning – “Thank You”.

Abstract

This project centered on the application of advanced machine learning techniques to predict heart disease in patients, emphasizing the development of a reliable classification algorithm for early diagnosis. The primary goals included binary prediction of heart disease, experimenting with classification models for optimal accuracy, exploring data trends and correlations, and determining key features influencing heart disease diagnosis.

The tasks encompassed comprehensive data processing and exploration of a dataset with 13 features, involving age, gender, chest pain type, blood pressure, and cholesterol levels. Exploratory data analysis employed correlation matrices, heat maps, and visualizations. Machine learning algorithms, specifically K-Nearest Neighbors (KNN) and Support Vector Machine (SVM), were implemented using tools like Tableau, Python, Pandas, NumPy, Scikit-learn, SciPy, Plotly, and Seaborn.

The analysis revealed significant features influencing predictions, including chest pain type, maximum heart rate achieved, number of major vessels, and ST depression induced by exercise relative to rest. In conclusion, the project successfully deployed machine learning algorithms, achieving notable accuracy rates. The identification of crucial features provides valuable insights for early heart disease diagnosis and proactive healthcare measures.

Introduction

In the vast landscape of global health challenges, cardiovascular diseases, with heart disease at its forefront, persist as formidable adversaries, consistently ranking among the primary causes of morbidity and mortality. The gravity of this pervasive health concern necessitates a profound emphasis on early diagnosis, serving as a pivotal gateway to effective intervention and preventive care. Against this backdrop of escalating cardiovascular concerns, a comprehensive and forward-thinking project takes center stage. This initiative boldly delves into the realm of predictive healthcare, propelled by the transformative capabilities of advanced machine learning techniques. The primary goal: to architect a sophisticated classification algorithm meticulously designed for the early detection of heart disease. These narrative invites exploration into the intricacies of this groundbreaking project, unraveling its significance, elucidating its multifaceted objectives, delving into the methodologies employed, and contemplating the potential impact that it holds in reshaping the landscape of preventive cardiology

Project Overview

In response to the critical need for early diagnosis, this project materializes as a beacon of innovation in the intersection of predictive analytics and healthcare. It is a forward-looking endeavor that seeks to capitalize on the evolving landscape of advanced machine learning techniques. At its core lies the ambitious objective of crafting a classification algorithm tailored with precision for the early detection of heart disease. This algorithm, envisioned as a sentinel of health, is poised to scrutinize a diverse array of medical features, providing timely and accurate predictions regarding the presence or absence of heart disease.

Goal:

The main goal is to develop a predictive model that can help identify people who are at risk of heart disease early on, allowing for prompt intervention and possibly leading to better health outcomes using Artificial Intelligence and Machine learning(AIML).

Objectives of the Project

1. **Binary Outcome Prediction:**
   - At the crux of the project lies the foundational goal of constructing a binary prediction model. This model categorizes patients into positive (indicating a diagnosis of heart disease) or negative (indicating an absence of heart disease) outcomes. This binary framework establishes the practical utility of the predictive tool.
2. **Model Experimentation:**
   - The project unfolds with a commitment to experimentation. It traverses a spectrum of classification models, meticulously assessing their accuracy and reliability. This iterative exploration aims to discern the nuances of each model, ultimately identifying the most effective predictor of heart disease.
3. **Exploratory Data Analysis (EDA):**
   - An indispensable facet of the project involves delving into the dataset through exploratory data analysis. This phase employs statistical analyses, correlation matrices, and visualization tools to unearth intricate relationships within the data. The insights gained lay the foundation for informed model development and feature selection.

4. Identification of Significant Features:

- The project embarks on a journey to unearth the crux of predictive indicators for heart disease. This involves isolating key features such as age, gender, chest pain type, blood pressure, and cholesterol levels. The goal is to distill the essence of diagnostic relevance from the rich tapestry of medical data.

History and Development of Predictive Healthcare for Heart Disease

Early Efforts:

The history of predictive healthcare for heart disease traces back to the mid-20th century when medical researchers and practitioners began recognizing the need for more proactive measures in cardiovascular care. Initial efforts focused on understanding risk factors, such as hypertension and high cholesterol, and their correlation with heart diseases. However, the lack of sophisticated technology limited the depth of analysis.

Introduction of Risk Prediction Models:

The late 20th century witnessed the advent of risk prediction models. Pioneering studies like the Framingham Heart Study, initiated in 1948, laid the foundation for assessing cardiovascular risk factors on a population scale. These models, though revolutionary, were often static and relied on traditional statistical methods.

Integration of Computer Technology:

With the rise of computer technology in the 1980s and 1990s, predictive modeling for heart disease entered a new era. Researchers could now analyze vast datasets more efficiently, leading to the development of risk algorithms incorporating multiple variables. However, these early models were still constrained by the limitations of computing power and the complexity of the data.

Emergence of Machine Learning:

The 21st century marked a paradigm shift with the integration of machine learning into predictive healthcare. Advanced algorithms, including decision trees, neural networks, and ensemble methods, empowered researchers to unravel intricate patterns within extensive datasets. The ability to consider non-linear relationships between variables opened new avenues for more accurate predictions.

Big Data and Electronic Health Records (EHRs):

The proliferation of electronic health records (EHRs) and the era of big data further accelerated the development of predictive healthcare models. These comprehensive datasets, encompassing patient demographics, medical history, and lifestyle factors, became invaluable for training sophisticated machine learning algorithms. This evolution allowed for a more personalized and precise approach to heart disease prediction.

Deep Learning and Neural Networks:

In recent years, deep learning, a subset of machine learning, has gained prominence. Neural networks, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have demonstrated exceptional capabilities in feature extraction and temporal analysis. This has enabled the development of models that can discern subtle patterns indicative of early-stage heart diseases.

Integration of Wearable Technology:

The widespread adoption of wearable technology, equipped with sensors for monitoring vital signs and physical activity, has added a new dimension to predictive healthcare. Real-time data from wearables can be integrated into predictive models, providing continuous monitoring and early detection capabilities.

Challenges and Ethical Considerations:

Despite the remarkable progress, challenges persist. Issues of data privacy, ethical considerations, and the potential for bias in predictive models demand careful scrutiny. Ensuring that these technologies benefit diverse populations equitably remains a critical aspect of ongoing development.

Related Work

A quiet Significant amount of work related to the diagnosis Cardiovascular Heart disease using Machine Learning algorithms has motivated this work. This paper contains a brief literature survey. An efficient cardiovascular disease prediction has been made by using various algorithms some of them include SVM, KNN Etc.

It can be seen in Results that each algorithm has its strength to register the defined objectives.

Materials Science and Engineering 1022 (2021) 012072 IOP Publishing doi:10.1088/1757-899X/1022/1/012072 3 heart disease using artificial neural network and other algorithms of machine and deep learning.

The risk factors of coronary heart disease or atherosclerosis is identified by McPherson using the inbuilt implementation algorithm using uses some techniques of Neural Network and were just accurately able to predict whether the test patient is suffering from the given disease or not. Diagnosis and prediction of heart disease and Blood Pressure along with other attributes using the aid of neural networks was introduced by R. Subramanian. A deep Neural Network was Built incorporating the given attributes related to the disease which were able to produce a output which was carried out by the output perceptron and almost included 120 hidden layers which is the basic and most relevant technique of ensuring a accurate result of having heart disease if we use the model for Test Dataset. The supervised network has been advised for diagnosis of heart diseases. When the testing of the model was done by a doctor using an unfamiliar data, the model used and trained from the previous learned data and predicted the result thereby calculating the accuracy of the given model.

Xing et al conducted a survey of 1000 patients, the results of which showed SVM to have 92.1% accuracy, artificial neural networks to have 91.0% and decision trees with 89.6% using TNF, IL6, IL8, HICRP, MPO1, TNI2, sex, age, smoke, hypertension, diabetes, and survival as the parameters. Similarly, Chen et al compared the accuracy of SVM, neural networks, Bayesian classification, decision tree and logistic regression. Considering 102 cases, SVM had the highest accuracy of 90.5%, neural networks 88.9%, Bayesian 82.2%, decision tree 77.9%, and logistic regression 73.9%.

INPUT:

S.NO	Attribute Name	Description	Range of Values
1	Age	Age of the person in years	
2	Sex	Gender of the person[1:male,0:female]	0,1
3	Cp	Chest pain type [1-Typical Type 1 Angina 2- Atypical type Angina 3-Non-Angina pain 4-Asymptomatic]	1,2,3,4
4	Trestbps	Resting Blood Pressure in mm Hg	94 to 200
5	Chol	Serum Cholestrol in mg/dl	126 to 564
6	Fbs	Fasting Blood Sugar in mg/dl	0,1
7	Restecg	Resting Electrocardiographic Results	0,1,2
8	Thalach	Maximum Heart Rate Achieved	71 to 202
9	Exang	Exercise Induced Angina	0,1
10	Old Peak	ST depression induced by exercise relative to rest	1 to 3
11	Slope	Slope of the Peak Exercise ST segment	1,2,3
12	Ca	Number of major vessels colored by fluoroscopy	0 to 3
13	Thal	3 – Normal, 6- Fixed Defect, 7- reversible defect	3,6,7
14	Target	Class Attribute	0 or 1

dataset consists of 14 attributes and 303 instances. There are 8 categorical attributes and 6 numeric attributes. The description of the dataset is shown in above Table. Patients from age 29 to 79 have been selected in this dataset. Male patients are denoted by a gender value 1 and female patients are denoted by gender value 0. Four types of chest pain can be considered as indicative of heart disease. Type 1 angina is caused by reduced blood flow to the heart muscles because of narrowed coronary arteries. Type 1 Angina is a chest pain that occurs during mental or emotional stress. Non-angina chest pain may be caused due to various reasons and may not often be due to actual heart disease. The fourth type, Asymptomatic, may not be a symptom of heart disease. The next attribute trestbps is the reading of the resting blood pressure. Chol is the cholesterol level. Fbs is the fasting blood sugar level; the value is assigned as 1 if the fasting blood sugar is below 120 mg/dl and 0 if it is above. Restecg is the resting electrocardiographic result, thalach is the maximum heart rate, exang is the exercise induced angina which is recorded as 1 if there is pain and 0 if there is no pain, old peak is the ST depression induced by exercise, slope is the slope of the peak exercise ST segment, ca is the number of major vessels colored by fluoroscopy, thal is the duration of the exercise test in minutes, and num is the class attribute. The class attribute has a value of 0 for normal and 1 for patients diagnosed with heart disease.

Algorithms Used:

KNN:

Nearest neighbor (KNN) is very simple, most popular, highly efficient, and effective algorithm for pattern recognition. KNN makes predictions by averaging the similarity between an input observation and the data already present. KNN is a straightforward classifier, where samples are classified based on the class of their nearest neighbor. Medical data bases are high volume in nature. If the data set contains redundant and irrelevant attributes, classification may produce less accurate result. K-Nearest neighbor (KNN) is a simple, lazy, and nonparametric classifier. KNN is preferred when all the features are continuous. KNN is also called as case-based reasoning and has been used in many applications like pattern recognition, statistical estimation. Classification is obtained by identifying the nearest neighbor to determine the class of an unknown sample. KNN is preferred over other classification algorithms due to its high convergence speed and simplicity.

KNN has 2 stages:

1) Find the *k* number of instances in the dataset that is closest to instance *S*

2) These *k* number of instances then vote to determine the class of instance *S*

The Accuracy of KNN depends on distance metric and K value. Various ways of measuring the distance between two instances are cosine, Euclidian distance. To evaluate the new unknown sample, KNN computes its K nearest neighbors and assign a class by majority voting.

SVM:

A support vector machine is a type of model used to analyze data and discover patters in classification and regression analysis. Support vector machine (SVM) is used when your data has exactly two classes. An SVM classifies data by finding the best hyper plane that separates all data points of one class from those of the other class. The larger margin between the two classes, the better the model is. A margin must have no points in its interior region. The support vectors are the data points that on the boundary of the margin. SVM is based on mathematical functions and used to model complex, and real-world problems. SVM performs well on data sets that have many attributes.

Support Vector Machines map the training data into kernel space. There are many differently used kernel spaces – linear (uses dot product), quadratic, polynomial, Radial Basis Function kernel, Multilayer Perceptron kernel, etc. to name a few. In addition, there are multiple methods of implementing SVM, such as quadratic programming, sequential minimal optimization, and least squares. The challenging aspect of SVM is kernel selection and method selection such that your model is not over optimistic or pessimistic.

Decision tree

A decision tree algorithm is a popular tool in machine learning for classification and regression tasks. It works by recursively partitioning the data into subsets based on the most significant attributes, creating a tree-like structure where each internal node represents a decision based on an attribute, and each leaf node represents a class label or a numerical value.

To apply it to heart disease prediction using Python, you'd start by collecting a dataset with relevant features like age, blood pressure, cholesterol levels, etc., and their corresponding labels indicating the presence or absence of heart disease. Then, you'd use libraries like scikit-learn to build a decision tree model. After training, you can evaluate its performance using metrics like accuracy, precision, and recall to assess its predictive capabilities in diagnosing heart disease.

Logistic Regression

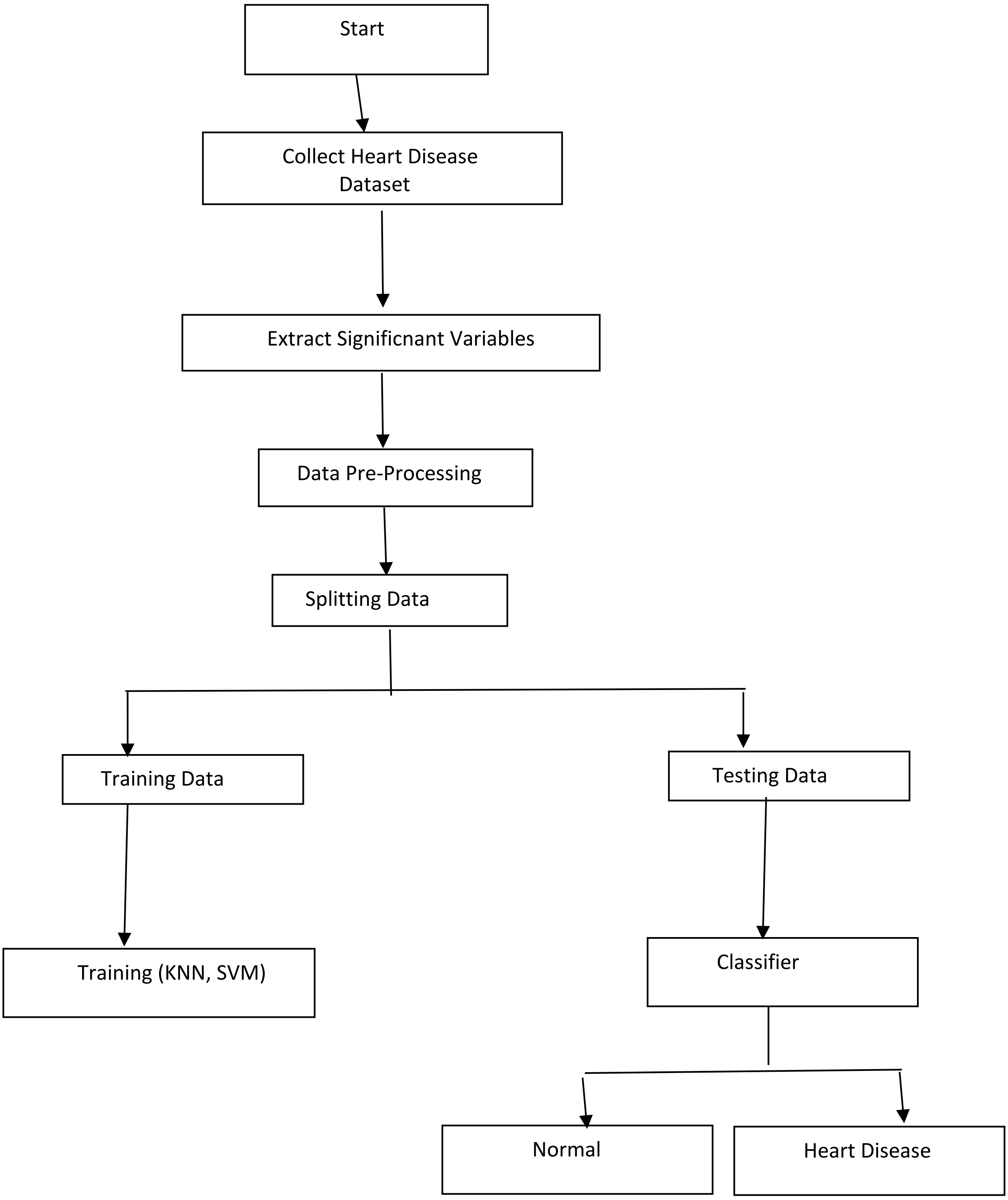
Logistic Regression is a supervised learning algorithm used for binary classification tasks. It models the probability that a given input belongs to a certain category using the logistic function.

In heart disease prediction, logistic regression can analyze various patient features (like age, blood pressure, cholesterol levels) to predict the likelihood of having heart disease. Python libraries like scikit-learn provide efficient implementations for logistic regression.

To apply logistic regression for heart disease prediction in Python, you'd typically start by preparing your dataset, splitting it into training and testing sets. Then, you would instantiate a logistic regression model, fit it to the training data, and evaluate its performance on the testing data using metrics like accuracy, precision, recall, or F1-score. Finally, you can make predictions on new data to assess the likelihood of heart disease.

Methodology :

This paper shows the analysis of various machine learning algorithms, the algorithms that are used in this paper are K nearest neighbors (KNN), SVM which can be helpful for practitioners or medical analysts for accurately diagnose Heart Disease. This paperwork includes examining the journals, published paper and the data of cardiovascular disease of the recent times. Methodology gives a framework for the proposed model. The methodology is a process which includes steps that transform given data into recognized data patterns for the knowledge of the users. The proposed methodology (Figure 1.) includes steps, where first step is referred as the collection of the data than in second stage it extracts significant values than the 3rd is the preprocessing stage where we explore the data. Data preprocessing deals with the missing values, cleaning of data and normalization depending on algorithms used. After pre-processing of data, classifier is used to classify the pre-processed data the classifier used in the proposed model are KNN, Logistic Regression, Random Forest Classifier. Finally, the proposed model is undertaken, where we evaluated our model on the basis of accuracy and performance using various performance metrics. This model uses 13 medical parameters such as chest pain, fasting sugar, blood pressure, cholesterol, age, sex etc. for prediction.


Start
Collect Heart Disease Dataset
Extract Significnant Variables
Data Pre-Processing
Splitting Data
Training Data
Testing Data
Training (KNN, SVM)
Classifier
Normal
Heart Disease

Future Work

The trajectory of predictive healthcare for heart disease is poised for continued innovation. Integration with genetic data, more sophisticated feature engineering, and the exploration of explainable AI are anticipated future developments. Collaborations between healthcare professionals, data scientists, and technology experts will be pivotal in navigating the evolving landscape of predictive healthcare. We can use other algorithms in order to predict the heart disease such as naïve bayes, logistic Regression.

The utilization of KNN algorithm and SVM in heart disease prediction offers significant potential for improving accuracy. By implementing strategies such as addressing data imbalance and optimizing the value of k through cross-validation, we can enhance the reliability of this algorithm in healthcare. Moving forward, further research should focus on exploring additional techniques and integrating advanced machine learning approaches to enhance the prediction accuracy and preventive measures for heart disease.

In summary, the history and development of predictive healthcare for heart disease reflect a journey from rudimentary risk models to the era of advanced machine learning and big data. As technology continues to evolve, the potential for early detection, personalized medicine, and improved patient outcomes stands as a testament to the transformative power of predictive analytics in the realm of cardiovascular health.

In [1]:
```
#implementation of heart attack analysis using KNN algorithum
 import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [3]:
```
df = pd.read_csv(r'D:\project\heart.csv')
```

In [4]:
```
df
```

Out[4]:
```
       age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  ta

0      63   1    3   145       233   1    0        150      0      2.3      0      0   1
1      37   1    2   130       250   0    1        187      0      3.5      0      0   2
2      41   0    1   130       204   0    0        172      0      1.4      2      0   2
3      56   1    1   120       236   0    1        178      0      0.8      2      0   2
4      57   0    0   120       354   0    1        163      1      0.6      2      0   2
...    ...  ...  ... ...       ...  ...  ...      ...      ...    ...      ...  ...    ...
298    57   0    0   140       241   0    1        123      1      0.2      1      0   3
299    45   1    3   110       264   0    1        132      0      1.2      1      0   3
300    68   1    0   144       193   1    1        141      0      3.4      1      2   3
301    57   1    0   130       131   0    1        115      1      1.2      1      1   3
302    57   0    1   130       236   0    0        174      0      0.0      1      1   2

303 rows × 14 columns
```

In [5]:
```
df.describe()
```

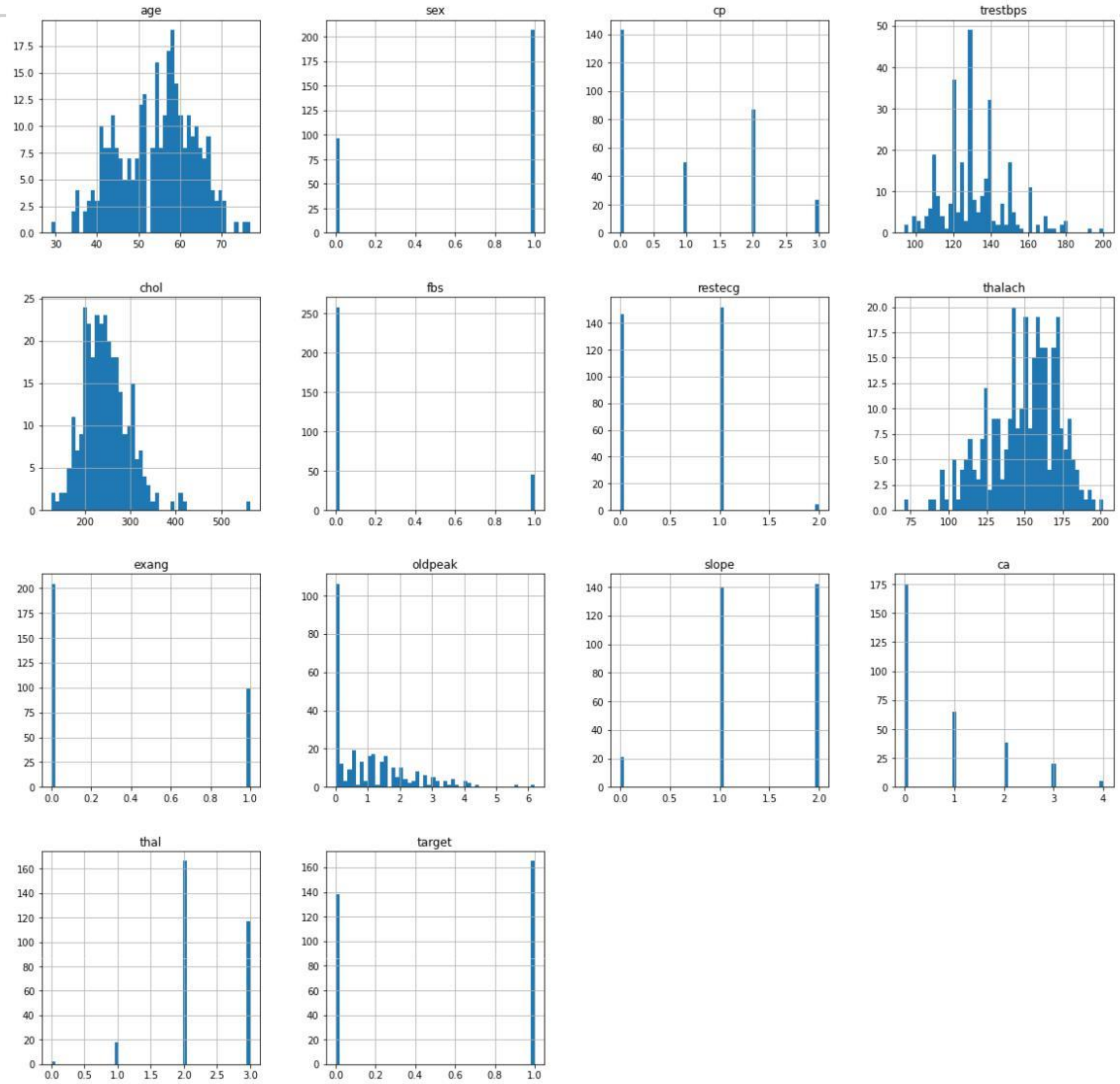
Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000

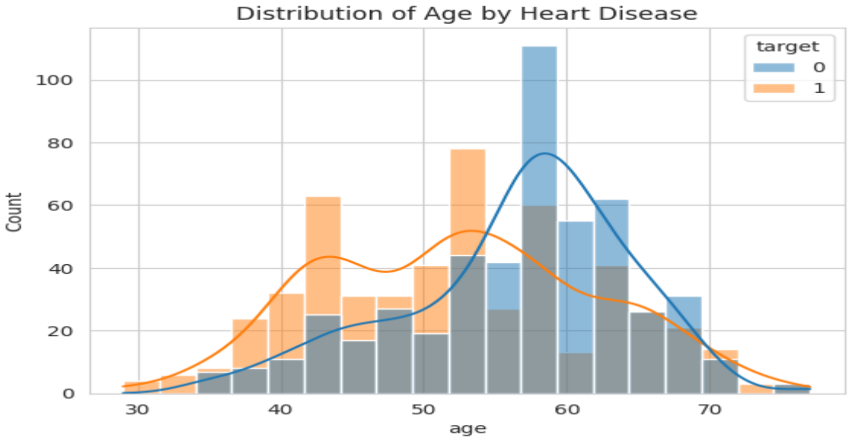
In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data  columns (total 14 columns):
 #    Column     Non-Null Count   Dtype
---   ------     --------------   -----
 0    age        303  non-null    int64
 1    sex        303  non-null    int64
 2    cp         303  non-null    int64
 3    trestbps   303  non-null    int64
 4    chol       303  non-null    int64
 5    fbs        303  non-null    int64
 6    restecg    303  non-null    int64
 7    thalach    303  non-null    int64
 8    exang      303  non-null    int64
 9    oldpeak    303  non-null    float64
 10   slope      303  non-null    int64
 11   ca         303  non-null    int64
 12   thal       303  non-null    int64
 13   target     303  non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3    KB
```

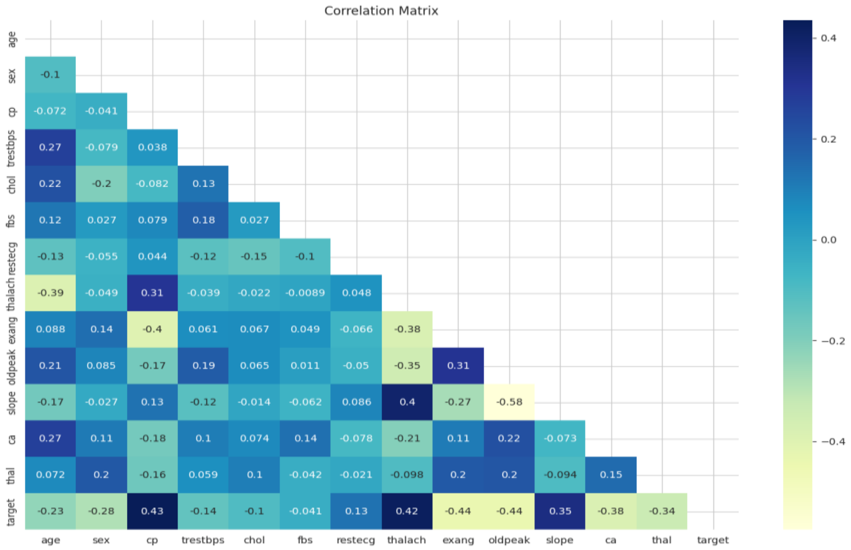
```
In [7]:  df.hist(bins=50, figsize=(20,20))
          plt.show()
```



```
sns.histplot(data=df, x=df['age'], hue='target', kde=True)
plt.title('Distribution of Age by Heart Disease')
plt.xlabel=('age')
plt.ylabel=('count')
plt.show()
```

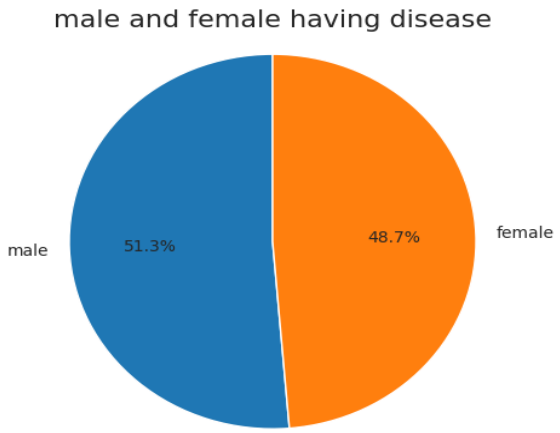


```
corr_matrix = df.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
fig, ax = plt.subplots(figsize=(15, 10))
sns.heatmap(corr_matrix, mask=mask, annot=True, cmap='YlGnBu')
plt.title('Correlation Matrix')
plt.show()
```

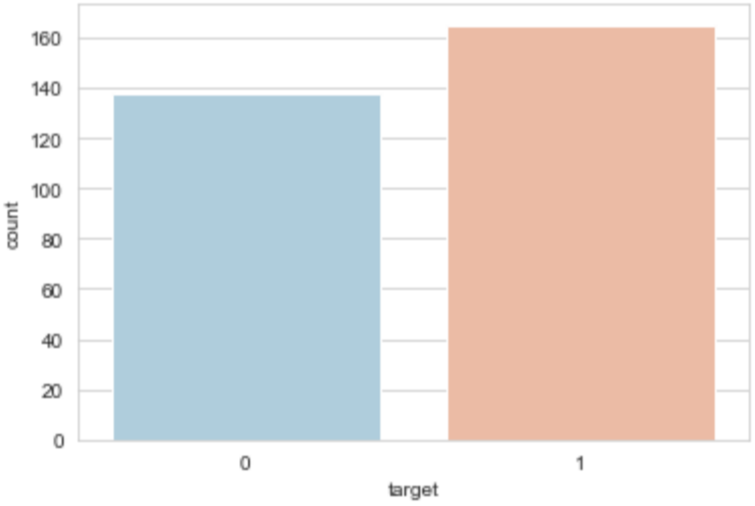


```
male = len(df[df['target'] == 1])
female = len(df[df['target']== 0])
```

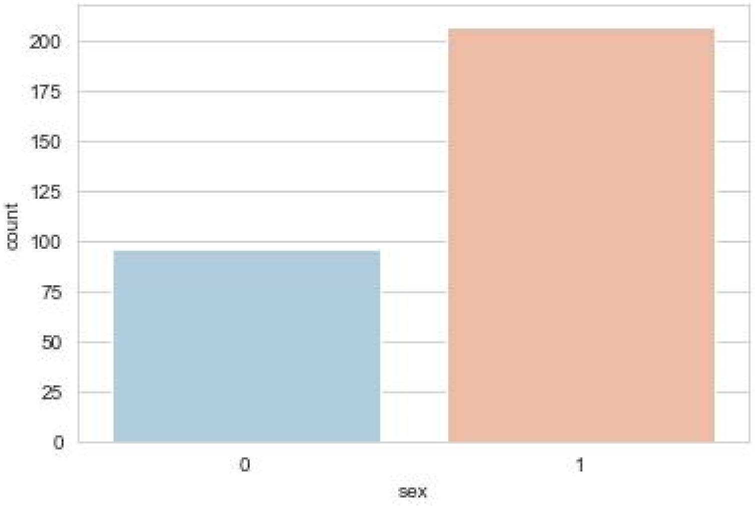
```
y = ('Male', 'Female')
y_pos = np.arange(len(y))
x = (male, female)
labels = 'male', 'female'
sizes = [male, female]
fig1, ax1 = plt.subplots()
ax1.pie(sizes,  labels=labels, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')
plt.title('male and female having disease', size=16)
plt.show()
```



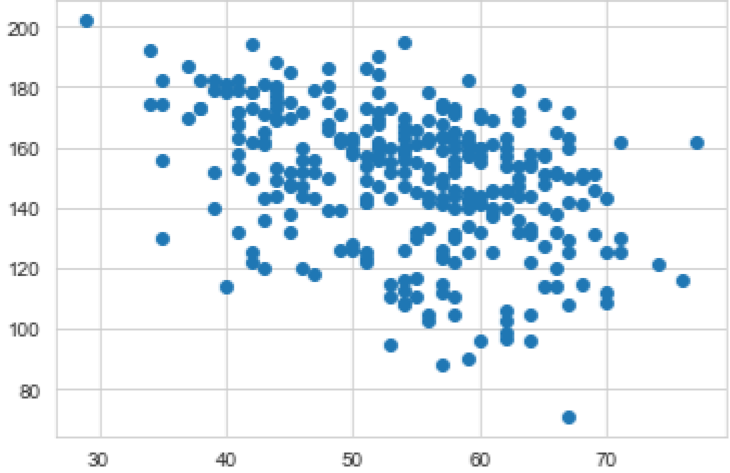
```
In [9]:  sns.set_style('whitegrid')
         sns.countplot(x='target',data=df,palette='RdBu_r')

Out[9]: <AxesSubplot:xlabel='target', ylabel='count'>
```



```
In [10]: #bar plot for male and female having heart desceas
         sns.set_style('whitegrid')
         sns.countplot(x='sex',data=df,palette='RdBu_r')

Out[10]: <AxesSubplot:xlabel='sex', ylabel='count'>
```



```
In [11]: #scatter plot for age and maximum heart rate
         plt.scatter(df.age,df.thalach)

Out[11]: <matplotlib.collections.PathCollection at 0x1c90fa10250>
```



```
In [12]: dataset = pd.get_dummies(df, columns = ['sex', 'cp', 'fbs', 'restecg', 'exan
```

```
In [13]: from sklearn.model_selection import train_test_split
         from  sklearn.preprocessing  import  StandardScaler
         standardScaler = StandardScaler()
         columns_to_scale   =   ['age',   'trestbps',   'chol',   'thalach',   'oldpeak']
         dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_
```

```
In [14]: dataset.head()
```

Out[14]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...
0	0.952197	0.763956	-0.256334	0.015443	1.087338	1	0	1	0	0	...
1	-1.915313	-0.092738	0.072199	1.633471	2.122573	1	0	1	0	0	...
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912	1	1	0	0	1	...
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705	1	0	1	0	1	...
4	0.290464	-0.663867	2.082050	0.583939	-0.379244	1	1	0	1	0	...

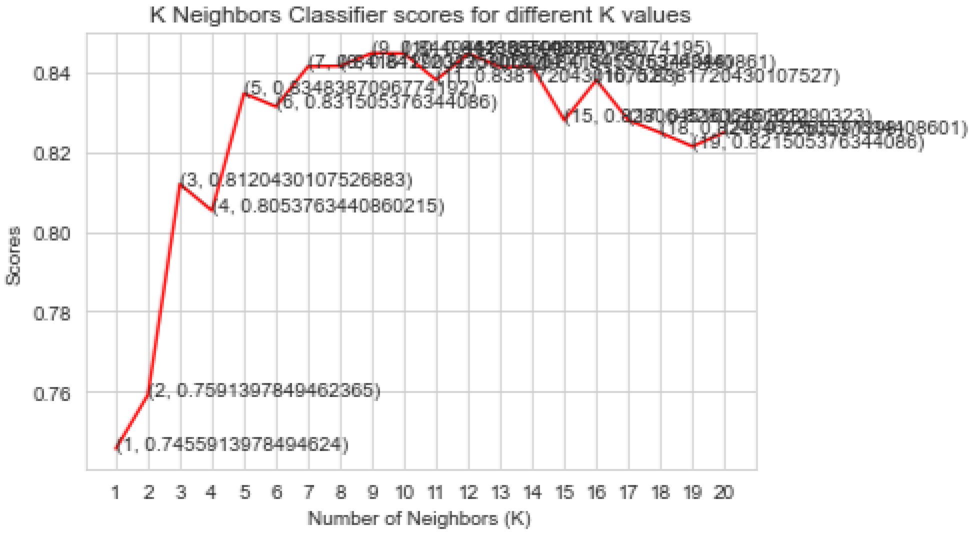
5 rows × 31 columns

```
In [15]: y = dataset['target']
         X = dataset.drop(['target'], axis = 1)
```

```
In [16]: from sklearn.model_selection import cross_val_score
         knn_scores = []
         for k in range(1,21):
             knn_classifier = KNeighborsClassifier(n_neighbors = k)
             score=cross_val_score(knn_classifier,X,y,cv=10)
             knn_scores.append(score.mean())
```

```
In [17]: 
         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')
```

```
Out[17]: Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')
```



```
In [18]:  knn_classifier = KNeighborsClassifier(n_neighbors =12)
         score=cross_val_score(knn_classifier,X,y,cv=10)
```

```
In [19]:  score.mean()
```

```
Out[19]: 0.8448387096774195
```

```
In [20]:  #accuracy of this model using cross validation is 84%
```

```
In [21]:  from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import confusion_matrix, classification_report
```

```
In [22]: from sklearn.model_selection import train_test_split
         X = df.drop('target', axis=1)
         y = df['target']

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

         print("Shape of training set:", X_train.shape)
         print("Shape of test set:", X_test.shape)
```

```
         Shape of training set: (212, 13)
         Shape of test set: (91, 13)
```

In [23]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

In [24]:

```
 knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
predictions2 = knn.predict(X_test)
```

In [25]:

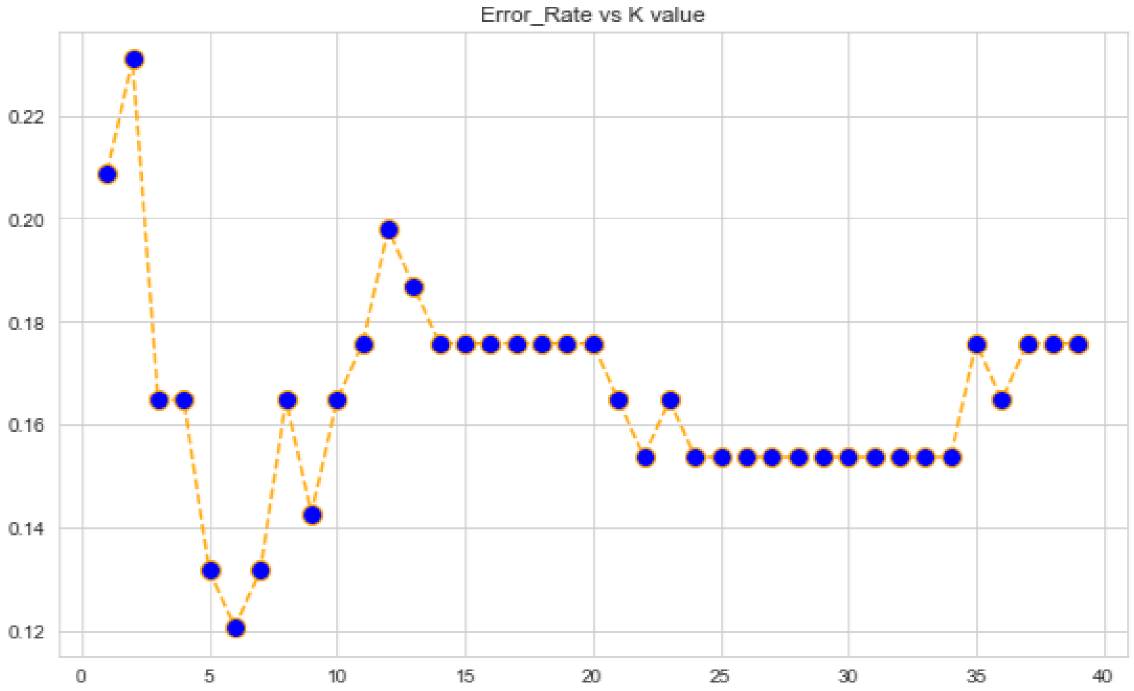
```
 print(confusion_matrix(y_test, predictions2))
print("\n")
print(classification_report(y_test, predictions2))
```

```
[[30 11]
 [ 8  42]]


             precision   recall   f1-score   support

    0        0.79        0.73     0.76       41
    1        0.79        0.84     0.82       50

    accuracy                      0.79       91
macro avg    0.79        0.79     0.79       91
weighted avg 0.79        0.79     0.79       91
```

In [26]:

```
 error_rate = []

for i in range(1,40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [27]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40), error_rate, color='orange', linestyle="--",marker='o'
plt.title('Error_Rate vs K value')
plt.xlabel = ('K')
plt.ylabel = ('Error Rate')
```

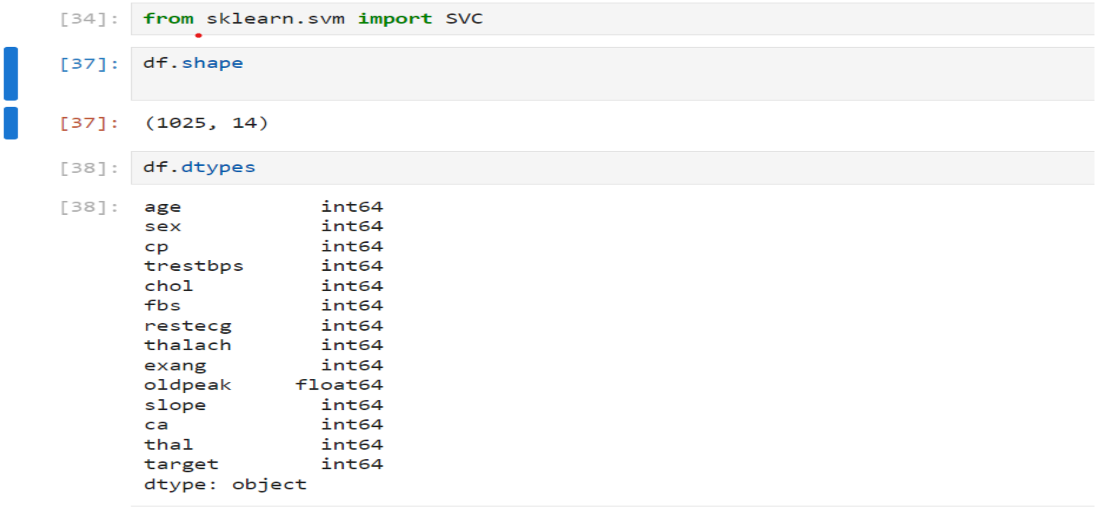


In [36]:
```
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
predictions2 = knn.predict(X_test)
```

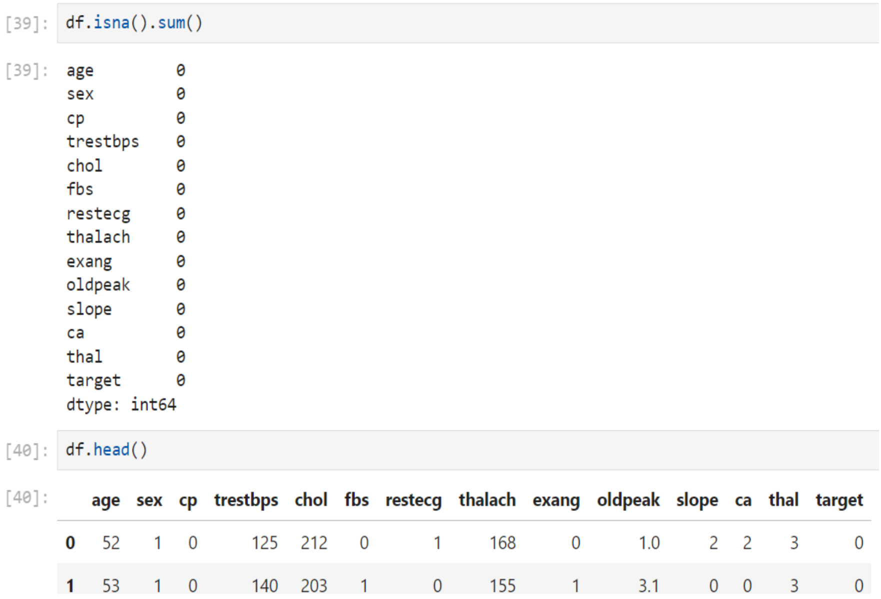
In [37]:
```
print(confusion_matrix(y_test, predictions2))
print("\n")
print(classification_report(y_test, predictions2))
```

```
[[365]
 [ 6 44]]


              precision    recall   f1-score   support

           0       0.86      0.88      0.87        41
           1       0.90      0.88      0.89        50

    accuracy                           0.88        91
   macro avg       0.88      0.88      0.88        91
weighted avg       0.88      0.88      0.88        91
```

In [30]:
```
#accuracy of this mode is 79 to 88
```



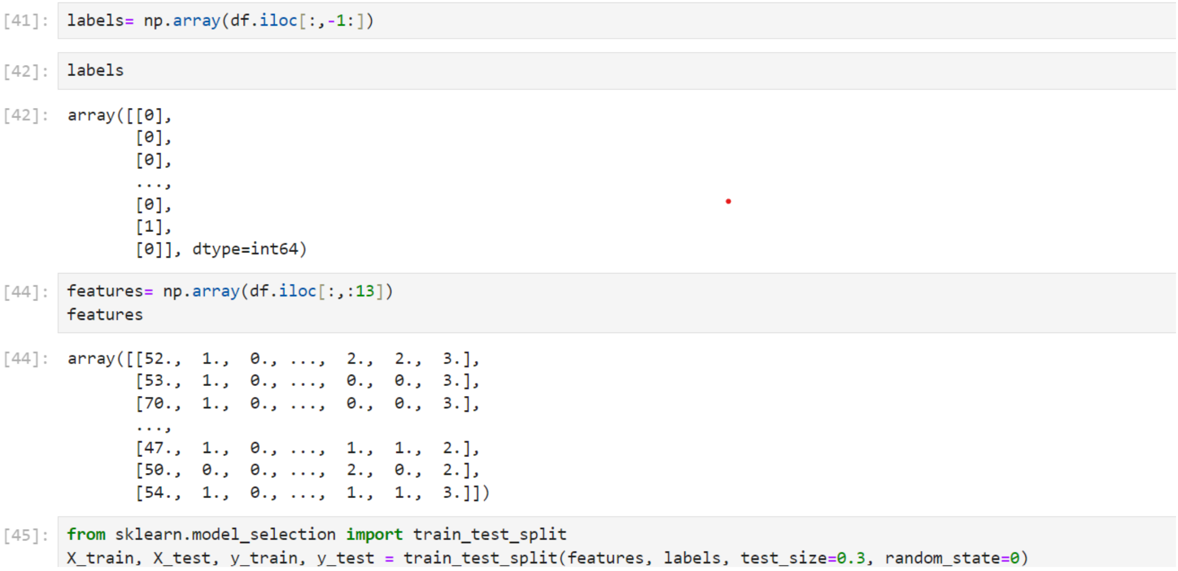
```
[34]: from sklearn.svm import SVC

[37]: df.shape

[37]: (1025, 14)

[38]: df.dtypes

[38]: age           int64
      sex           int64
      cp            int64
      trestbps      int64
      chol          int64
      fbs           int64
      restecg       int64
      thalach       int64
      exang         int64
      oldpeak     float64
      slope         int64
      ca            int64
      thal          int64
      target        int64
      dtype: object
```



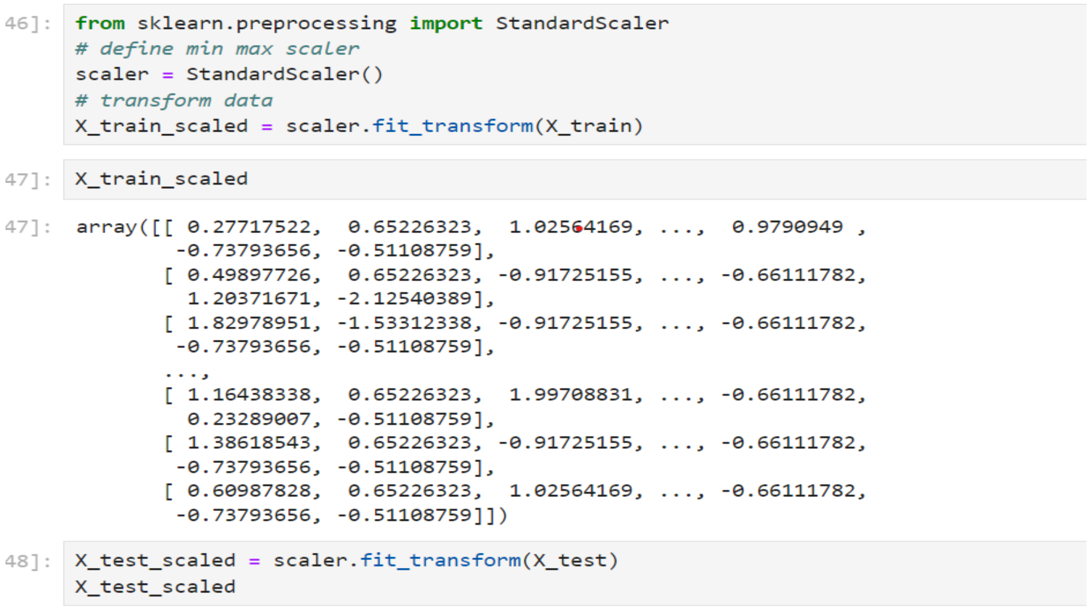
```
[39]: df.isna().sum()

[39]: age         0
      sex         0
      cp          0
      trestbps    0
      chol        0
      fbs         0
      restecg     0
      thalach     0
      exang       0
      oldpeak     0
      slope       0
      ca          0
      thal        0
      target      0
      dtype: int64

[40]: df.head()
```

[40]:

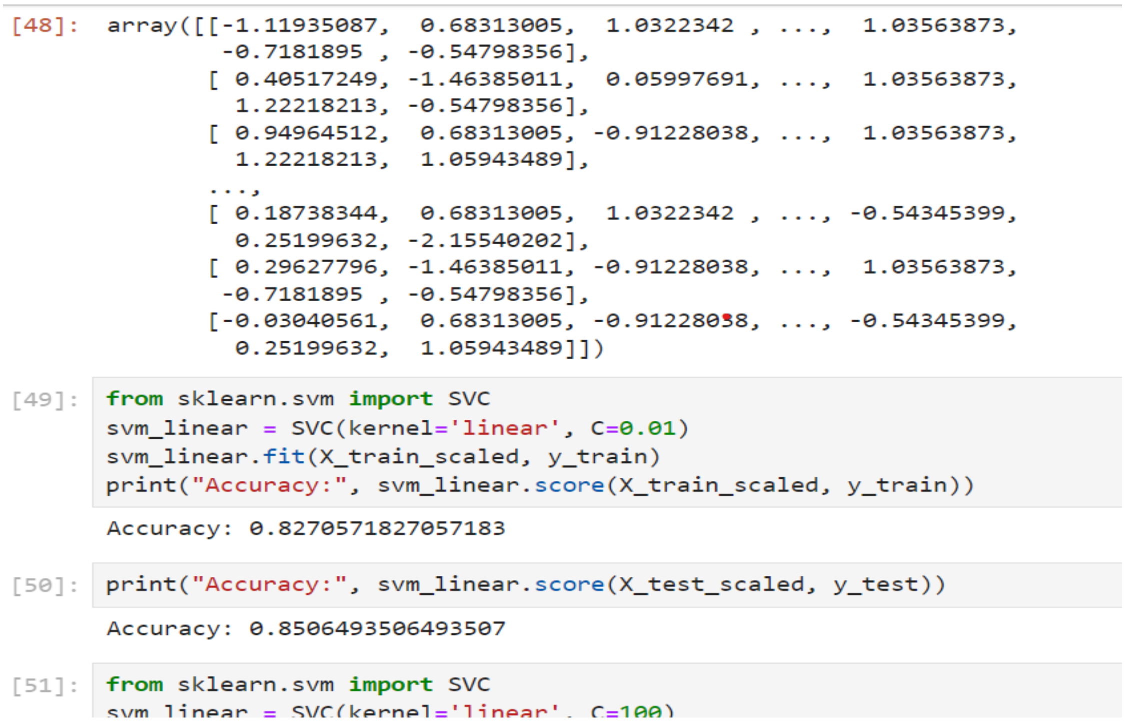
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0



```
[41]: labels= np.array(df.iloc[:,-1:])

[42]: labels

[42]: array([[0],
             [0],
             [0],
             ...,
             [0],
             [1],
             [0]], dtype=int64)

[44]: features= np.array(df.iloc[:,:13])
      features

[44]: array([[52.,  1.,  0., ...,  2.,  2.,  3.],
             [53.,  1.,  0., ...,  0.,  0.,  3.],
             [70.,  1.,  0., ...,  0.,  0.,  3.],
             ...,
             [47.,  1.,  0., ...,  1.,  1.,  2.],
             [50.,  0.,  0., ...,  2.,  0.,  2.],
             [54.,  1.,  0., ...,  1.,  1.,  3.]])

[45]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.3, random_state=0)
```



```
46]: from sklearn.preprocessing import StandardScaler
     # define min max scaler
     scaler = StandardScaler()
     # transform data
     X_train_scaled = scaler.fit_transform(X_train)

47]: X_train_scaled

47]: array([[ 0.27717522,  0.65226323,  1.02564169, ...,  0.9790949 ,
             -0.73793656, -0.51108759],
            [ 0.49897726,  0.65226323, -0.91725155, ..., -0.66111782,
              1.20371671, -2.12540389],
            [ 1.82978951, -1.53312338, -0.91725155, ..., -0.66111782,
             -0.73793656, -0.51108759],
            ...,
            [ 1.16438338,  0.65226323,  1.99708831, ..., -0.66111782,
              0.23289007, -0.51108759],
            [ 1.38618543,  0.65226323, -0.91725155, ..., -0.66111782,
             -0.73793656, -0.51108759],
            [ 0.60987828,  0.65226323,  1.02564169, ..., -0.66111782,
             -0.73793656, -0.51108759]])

48]: X_test_scaled = scaler.fit_transform(X_test)
     X_test_scaled
```



```
[48]: array([[-1.11935087,  0.68313005,  1.0322342 , ...,  1.03563873,
              -0.7181895 , -0.54798356],
             [ 0.40517249, -1.46385011,  0.05997691, ...,  1.03563873,
               1.22218213, -0.54798356],
             [ 0.94964512,  0.68313005, -0.91228038, ...,  1.03563873,
               1.22218213,  1.05943489],
             ...,
             [ 0.18738344,  0.68313005,  1.0322342 , ..., -0.54345399,
               0.25199632, -2.15540202],
             [ 0.29627796, -1.46385011, -0.91228038, ...,  1.03563873,
              -0.7181895 , -0.54798356],
             [-0.03040561,  0.68313005, -0.91228038, ..., -0.54345399,
               0.25199632,  1.05943489]])

[49]: from sklearn.svm import SVC
      svm_linear = SVC(kernel='linear', C=0.01)
      svm_linear.fit(X_train_scaled, y_train)
      print("Accuracy:", svm_linear.score(X_train_scaled, y_train))

      Accuracy: 0.8270571827057183

[50]: print("Accuracy:", svm_linear.score(X_test_scaled, y_test))

      Accuracy: 0.8506493506493507

[51]: from sklearn.svm import SVC
      svm_linear = SVC(kernel='linear', C=100)
```

```
class DataAnalysis:
    def __init__(self, data):
        self.data = data

    def train_decision_tree(self):
        try:

            X = self.data.drop(columns=['target'])
            y = self.data['target']

            X_encoded = pd.get_dummies(X)

            X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, 
test_size=0.2, random_state=42)
```

```
            dt_classifier = DecisionTreeClassifier(random_state=42)
            dt_classifier.fit(X_train, y_train)

            dt_predictions = dt_classifier.predict(X_test)


            print("Decision Tree Classifier:")
            print("Accuracy:", accuracy_score(y_test, dt_predictions))
            print("Classification Report:")
            print(classification_report(y_test, dt_predictions))
        except Exception as e:
            print("Error occurred during Decision Tree training:", e)

data_analysis = DataAnalysis(df)

data_analysis.train_decision_tree()
```

```
Decision Tree Classifier:
Accuracy: 0.9853658536585366
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
 weighted avg       0.99      0.99      0.99       20
```

```
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import ConfusionMatrixDisplay , classification_report,
accuracy_score, precision_recall_curve
from imblearn.over_sampling import SMOTE
```

```
parameters = {'C': [0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2']}
logreg = LogisticRegression(solver='liblinear')
clf = GridSearchCV(logreg, parameters, cv=5)
clf.fit(X_train, y_train)
print("Best parameters:", clf.best_params_)
```

Best parameters: {'C': 1, 'penalty': 'l1'}

```

model = LogisticRegression(C=10, penalty='l1', solver='liblinear')
model.fit(X_train, y_train)
```

LogisticRegression

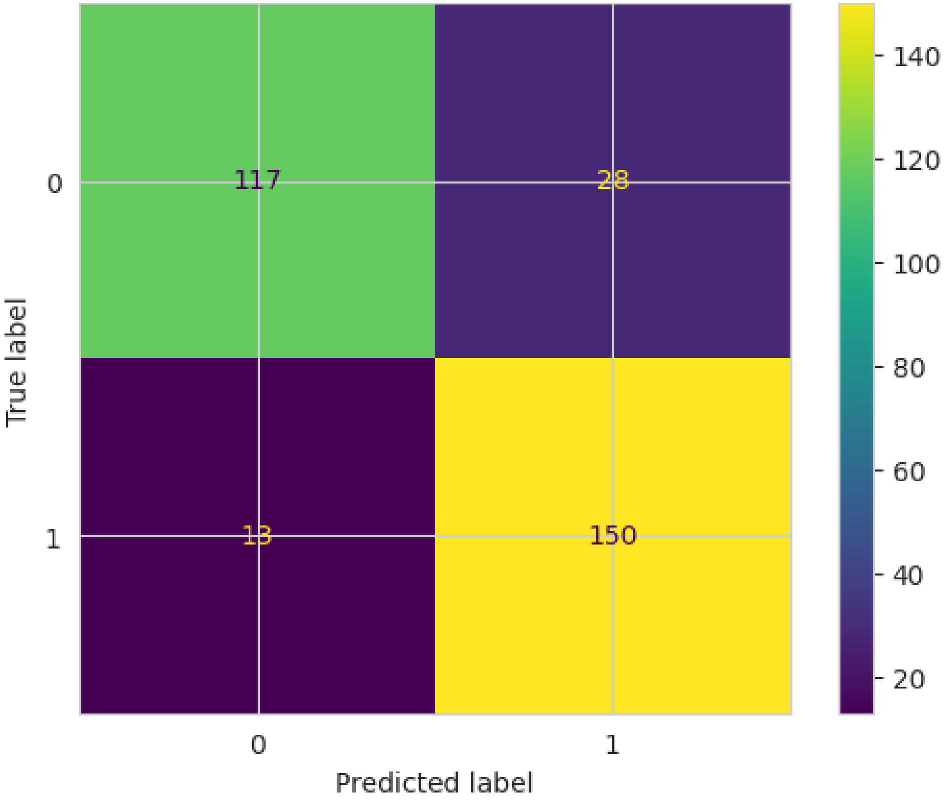
```
LogisticRegression(C=10, penalty='l1', solver='liblinear')
```

```
predictions = model.predict(X_test)
predictions
```

```
testing_acc = accuracy_score(y_test, predictions)

print(f"Testing accuracy : {testing_acc: .4f}")
```

```
Testing accuracy :  0.8669
```

```
ConfusionMatrixDisplay.from_estimator(model,X_test,y_test)
```



Conclusion

In tracing the evolution of predictive healthcare for heart disease, we traverse a remarkable journey marked by paradigm shifts and technological revolutions. From the pioneering studies of the mid-20th century to the advent of risk prediction models and the integration of computer technology, each phase has contributed to our understanding and approach to cardiovascular care. The 21st century, characterized by the rise of machine learning, big data, and wearable technology, has propelled predictive models to unprecedented heights.

The marriage of advanced algorithms with vast datasets has ushered in an era where predictive healthcare goes beyond risk assessment, delving into the realm of personalized and precise early detection. Neural networks, deep learning, and the integration of real-time wearable data exemplify the cutting-edge tools driving this transformative journey.

However, with progress comes responsibility. Ethical considerations, data privacy, and the potential for bias underscore the need for a thoughtful and inclusive approach to further development. As we stand at the cusp of future innovations, collaborations between healthcare professionals, data scientists, and technologists become indispensable for navigating the complex landscape of predictive healthcare.

Looking ahead, the future promises even greater strides. Integration with genetic data, more nuanced feature engineering, and a commitment to equitable healthcare delivery will shape the trajectory of predictive models. In this dynamic landscape, the journey continues, guided by the pursuit of early detection, personalized medicine, and ultimately, improved cardiovascular outcomes for individuals across diverse communities.

REFERENCES

- ○Yanwei Xing, Jie Wang and Zhihong Zhao Yonghong Gao 2007 “Combination data mining methods with new medical data to predicting outcome of coronary heart disease” Convergence Information Technology, 2007. International Conference November 2007, pp 868-872.
- Jianxin Chen, Guangcheng Xi, Yanwei Xing, Jing Chen, and Jie Wang 2007 “Predicting Syndrome by NEI
- Specifications: A Comparison of Five Data Mining Algorithms in Coronary Heart Disease” Life System Modeling and Simulation Lecture Notes in Computer Science, pp 129-135.
- Seema K,Bomare DS,Vaishnavi N. Heart disease prediction using KNN based handwritten text. AISC 2016; 49-56.
- Diagnosis and prediction of heart disease and Blood Pressure along with other attributes using the aid of neural networks- R. Subramanian
- Jabbar MA,Deekshatulu BL,Priti C.Heart disease classification using nearest neighbor classifier with feature subset selection. Annals Computer Science 2013
- https://arxiv.org/abs/2112.06459
- M. Abdar, S. R. N. Kalhori, T. Sutikno, I. M. I. Subrotoand G. Arji, Comparing performance of data mining algorithms in prediction heart diseases, vol. 5, no. 6, pp. 1569–1576, 2015. https://doi.org/10.11591/ijece.v5i6.pp1569-1576
- A. Mustaqeem, S. M. Anwar, A. R. Khan, and M. Majid, A statistical analysis based recommender model for heart disease patients,Int. J. Med. Inform., vol. 108, October, pp. 134–145, 2017. https://doi.org/10.1016/j.ijmedinf.2017.10.008
- M. Rivki and A. M. Bachtiar, Implementasi algoritma k-nearest neighbor dalam pengklasifikasian follower twitter yang menggunakan bahasa indonesia, no. 112, 2017.
- M. Ary and D. A. F. Rismiati, SATIN – Sains dan teknologi informasi ukuran akurasi klasifikasi penyakit mesothelioma menggunakan algoritma k-nearest neighbor dan backward elimination, vol. 5, no. 1, 2019.
- T . M. Manik, Analisis karakteristik fungsi lagrange dalam menyelesaikan permasalahan optimasi berkendala, vol. 1, no. 1, pp. 0–7, 2018. https://doi.org/10.32734/st.v1i1.187
- T. Setiyorini and R. T. Asmono, Komparasi metode neural network , support vector machine dan linear regression pada estimasi kuat tekan, vol. 15, no. 1, pp. 51–56, 2018.
- S. Aulia, S. Hadiyoso and D. N. Ramadhan, Analisis perbandingan knn dengan svm untuk klasifikasi penyakit diabetes retinopati berdasarkan citra eksudat dan mikroaneurisma, vol. 3, no. 1, pp. 75–90, 2015.

- A. Rohman, V . Suhartono and C. Supriyanto,
  Penerapan algoritma c4.5 berbasis adaboost untuk prediksi penyakit jantung, vol. 13, pp. 13–19, 2017.