Heart Disease Prediction Using Machine Learning Algorithms

By

Roshan Gautam

Student ID: 22125622

Module: CMP7247 Artificial Intelligence Fundamentals

Coordinator: Dr. Debashish Das

Masters in Computer Science,

Faculty of Computing Engineering and Built Environment
Birmingham City University,

Birmingham, United Kingdom

Date: 30/08/2022

Abstract

There is an indispensable role played by heart in the living organisms. Prediction, as well as diagnosis of heart-related diseases, requires sheer precision, correctness, and perfection as small errors leads to fatigue problem and even leads to the demise of the person. In recent times, there have been several premature death cases which are related to heart attacks and their count is escalating exponentially. For dealing with those issues there must be a prediction system that could make people aware of the scenario and awareness related to the health of their hearts. Machine learning which is one of the units of Artificial Intelligence (AI), makes the use of natural events to train itself so that the relevant future event could be predicted. In this project, the accuracy of three different machine learning algorithms has been done for predicting heart disease, for this purpose K-nearest neighbor, Support Vector Machine, and Linear Regression are implemented by using the Kaggle repository to get the heart dataset to carry out training and testing. Through normal and hyperparameter tuning the trial results verified that the Support Vector Machine algorithm has achieved the maximum accuracy of 98.05% as in comparison to other ML algorithms.

1. Introduction

The heart is the principal organ of the human body which is responsible for the regulation and flow of blood throughout the whole body. Any type of irregularities in the heart has a devouring effect on other parts of the body(Ismaeel, Miri and Chourishi, 2015). Heart disease is coined as something in which there is a disturbance in the normal operation of the heart. In today's world, one of the primary reasons for deaths is heart disease. Several factors lead to heart diseases which are smoking, an unhealthy lifestyle, high intake of fat, and drinking alcohol which leads to hypertension(Wingard and Barrett-connor, 2003). In accordance to the statistics, more than 10 million people die due to the reason of heart disease every year in the world. The only possible way as of today to avoid heart disease is early detection and adopting a healthy lifestyle.

In the present day, the most obstacle in healthcare is the lack of proper access to effective diagnosis and reliable quality services. Even though the most deaths in the world is heart diseases, they are also those which can be managed and controlled effectively. The entire accuracy in which there would be proper management of disease depends upon the proper time of its detection(Ali *et al.*, 2021). The proposed work delineates an approach for detecting heart disease at an early stage so that the disastrous consequences could be mitigated.

There are large records of medical datasets which are made available by experts for extracting and analyzing valuable information from it. The techniques of data mining are the precious means for extracting hidden and valuable information from large sets of available data. Similarly, in most medical datasets there is several discrete information and it becomes very much complex to draw decisions from that discrete information (Saleh Alotaibi, 2019). Thus, the process of decision-making becomes very much complicated when the information is discrete.

The process of extraction of valuable information and data from large sets of a database is called data mining. There are several types of data mining techniques which are clustering, regression, and association rules along with several other classification techniques such as decision tree, Naïve Bayes, K-nearest neighbor, random forest, and other algorithms which are used for the classification of heart diseases(Patel, 2016). These types of algorithms have the potential to analyze a huge amount of data from different fields and one of these important fields is the medical field.

Machine learning is an evolving domain among several other regions of artificial intelligence. In this field of data mining which is Machine Learning is responsible for handling large amounts of datasets effectively and efficiently(Hsieh *et al.*, 2012). In the area of the medical field, the use of Machine Learning can be implemented for handling large-scale data which are well-formatted. In the field of Medical Science, the use of the Machine Learning model can be done for the diagnosis, prediction, and detection of several diseases(Hsieh *et al.*, 2012).

2. Background

Millions of people around the globe have been affected by heart disease and it remains the most catastrophic cause of causing death throughout the world. There must be a reliable, proficient, and efficient system for medical diagnosis aided by the use of computer technology to reduce of cost carrying out diagnosis tests(Jagtap *et al.*, 2019). Machine learning is that domain of Artificial Intelligence (AI) that helps computers to classify several attributes(Kurzweil, 1985). This project makes the use of classification techniques for the prediction of heart disease. In this section, the related subjects of machine learning along with its methods, data pre-processing, dataset description, evaluation measurements along with brief description has been presented accordingly.

2.1 Machine Learning

Machine Learning is an evolving domain of AI (Artificial Intelligence) whose fundamental focus is to carry out the design of systems and make them learn along with making predictions based upon the experience of trained data. It makes the use of machine learning algorithms with the use of training data and creates a model(Yekkala, Dixit and Jabbar, 2018). Here the model makes the use of new inputted datasets for the prediction of events. With the use of Machine Learning the hidden patterns in the data are detected for building the relevant models and also make precise predictions for the newest dataset(Ali *et al.*, 2021). The cleaning of datasets is also done and the relevant missing values are filled accordingly. After that, the model makes use of new input data for heart disease prediction accordingly. The classification of Machine Learning can be done as follows:

2.2 Supervised Learning

In this learning method, the training of model is done on that dataset which is labeled which has its outcomes and input data, and data is split into testing and training datasets. Here, training datasets are responsible for training the model while the functionality of the new dataset is occupied by the testing data for the computing accuracy of the model(Repaka, Ravikanti and Franklin, 2019). There is the existence of datasets within the model and its output. Regression and Classification are examples of Supervised Learning.

2.3 Unsupervised Learning

In this learning, the data which is used for training the model is not labeled and it aims to find the hidden patterns in the data(Rizzo and Berneis, 2008). Here, the model is developed to develop the hidden patterns inside the data and can easily predict the hidden pattern for the new datasets. Besides, upon the exploration of data, it can also conclude the range of datasets for describing the hidden patterns(Hsieh *et al.*, 2012). In this method, there is no observable response in the dataset. Clustering is one example of unsupervised learning.

2.4 Reinforcement Learning

It does not make use of results that are associated with data or labeled datasets, where the model is dependent upon learning from the experience(Wingard and Barrett-connor, 2003). In this method, there is improvement in the presentation of the model based upon the association of the model with the environment and then only figures out to carry out discussion of its faults with for getting right outcome through testing and assessments.(Repaka, Ravikanti and Franklin, 2019)

The most commonly used algorithm for defining the probability of heart disease is the use of supervised learning techniques as the heat data is labelled mostly.

3. Aims and Objectives

The aim of the project is to compare the machine learning algorithms of KNN, SVM and logistic regression on the prediction of heart disease based upon the medical datasets.The objectives of the project are as follows:

- To select heart disease data comprising of several medical attributes such as age, gender, heart rate, etc.
- To carry out exploratory data analysis of the medical attributes upon the target and to analyze the data using several python libraries.
- To compute the accuracy of the algorithms by training them with heart data and through testing.
- To carry out hyperparameter tuning for making the algorithm efficient in the dataset and to compute the corresponding accuracy.

4. Motivation for the work

Several works have been carried out for the prediction of heart diseases using Machine Learning. Different sorts of accuracy have been achieved using several Machine Learning Algorithms which are explained as follows:

(Santhi *et al.*, 2021) have studied several types of ML algorithms that are employable for the prediction of heart disease. The research was carried out to use a Machine learning algorithm for the prediction of a heart attack which has the potential to make people out of the danger zone. They implemented the techniques of KNN and Random Forest algorithm for the prediction of a heart attack in advance. They implemented the technologies of Flask and React for developing the system. This research summarizes that the ML algorithms are useful for the prediction of heart diseases and it was mostly dependent upon dataset instance. Besides, it also concludes that the accuracy of KNN was higher than that of Random Forest and it could be made more efficient by combining several other techniques as well as parameter tunings.

(Saleh Alotaibi, 2019) has implemented an ML model by comparison of five different algorithms. The tool of Rapid Miner was implemented which created greater accuracy as compared to that of Weka Tool and Matlab. In this research, the comparison of accuracies of Logistic Regression, Decision Tree, Naïve Bayes, Random Forest, and SVM classification algorithms was done. Through the implementation, the highest efficiency was yielded by the Decision Tree algorithm.

(Alarsan and Younes, 2019) have proposed the method of ECG classification using the methods of machine learning and features of ECG. The proposed method was implemented based upon classification algorithms of Random Forests, Decision Tree, Gradient-Boosted Trees, etc. and the evaluation was done based upon the database of MIT-BIH Supraventricular Arrhythmia and through the implementation, the overall maximum accuracy obtained for the system was 98.03% for Random Forest Algorithm. This research also develops an inference that the results have been more profound when other sorts of clinical datasets such as heart rate, and stress would have been implemented in the datasets.

(Thomas and Princy, 2016) implemented a survey that included several classification algorithms which were meant for the prediction of heart disease. The classification techniques which were implemented for the research were KNN, Naïve Bayes, Neural Network, and Decision Tree and corresponding accuracies were defined for several attributes.

(Lutimath, Chethan and Pol, 2019) has performed the prediction of heart disease using the techniques of SVM, Naïve Bayes classification. The performance measures which were used in the analysis were Mean Absolute Error, Root Mean Squared Error, and Sum of Squared Error and it was established that the maximum accuracy in the system was generated by the SVM algorithm in terms of accuracy as compared to that of Naïve Bayes.

(Repaka, Ravikanti and Franklin, 2019) proposed a system that implemented Naïve Bayesian methods to carry out the classification of datasets. Besides, they also made use of the Advance Encryption Standard (AES) algorithm so that secure data transfer could be done accordingly.

Through reviewing the above papers it is found that several medical attributes such as restecg, cholesterol level, old peak, slope, etc. have not been encompassed in the datasets of these research, and some of the resources referenced these attributes to be advantageous in creating a better model for prediction of heart disease. Thus, the primary idea of the proposed system after reviewing the above papers is to create an enhanced system for the prediction of heart disease based on the implementation of Logistic Regression, KNN, and Support Vector algorithm by using the latest datasets of heart disease which encompassed different medical attributes which are prominent in defining the health of hearing. Besides, the proposed system also makes use of hyperparameter tuning for enhancing algorithms in predicting heart disease effectively.

5. Dataset Description

Firstly, the heart disease dataset is collected from a popular database which is Kaggle. Here, the dataset which is used in the project is composed of 14 different medical attributes. The variable which needs to be predicted whether a person is healthy or not is the target variable. When the value of the target variable is 1 it states that the person has heart disease and when the value is 0 the person has no heart disease. There are a total 1025 number of rows in the data and 14 columns. Here the 13 variables have the integer value while only one attribute has the old peak has a float value.

```
In [9]: data.shape
Out[9]: (1025, 14)
```

Figure 1: Total Number of rows and columns in the dataset

```
In [13]: pd.set_option("display.float", "{:.3f}".format)
         data.describe()
```

Out[13]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000
mean	54.434	0.696	0.942	131.612	246.000	0.149	0.530	149.114	0.337	1.072	1.385	0.754	2.324	0.513
std	9.072	0.460	1.030	17.517	51.593	0.357	0.528	23.006	0.473	1.175	0.618	1.031	0.621	0.500
min	29.000	0.000	0.000	94.000	126.000	0.000	0.000	71.000	0.000	0.000	0.000	0.000	0.000	0.000
25%	48.000	0.000	0.000	120.000	211.000	0.000	0.000	132.000	0.000	0.000	1.000	0.000	2.000	0.000
50%	56.000	1.000	1.000	130.000	240.000	0.000	1.000	152.000	0.000	0.800	1.000	0.000	2.000	1.000
75%	61.000	1.000	2.000	140.000	275.000	0.000	1.000	166.000	1.000	1.800	2.000	1.000	3.000	1.000
max	77.000	1.000	3.000	200.000	564.000	1.000	2.000	202.000	1.000	6.200	2.000	4.000	3.000	1.000

Figure 2: Statistical computation of the data

```
In [14]: data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

Figure 3: Datatype of the data in the datasets

4.1 Features of the attributes:

Attributes	Features
1. age	Age in years
2. sex	(0= female, 1= male)
3. chest pain (cp)	Chest pain 0: Angina which is typical 1: Atypical angina: chest pain that is not related to heart 2. Non-angina pain: typically related to esophageal spasms not related to heart 3. Asymptomatic: Chest pain that does not shows the signs of heart disease
4. treetops	Resting blood pressure in terms of mm Hg
5. chol	The level of serum cholesterol in mg/dl Serum: HDL+LDL+ 0.2* triglycerides Above 200 is a matter of concern
6. fbs	Fasting blood sugar > 120 mg/dl (0=false, 1= True) .> 126 mg/dl signals diabetes
7. restecg	Results of resting electrocardiographic results 0: nothing for carrying out to note 1: ST-T abnormality in wave • Can range from mild symptoms in case of severe problems • Signals which have no normal heart beat 2. Possibility of definite
8. thalach	Maximum value of heart rate
9. exang	Angina induced from exercise(0=no, 1=yes)
10. oldpeak	Exercise induced ST depression, more stress is there in the unhealthy heart
11. slope	The slope of ST segment for peak exercise 0: Unsloping 1: Flat slopping 2: Downslopes
12. ca	Number of major vessels ranges from 0-3 which is colored by fluoroscopy Colored vessels imply that the doctor is able to see the passing of blood more movement of blood regards better
13. thal	Stress result of thalium 1,3: Normal 7: reversible defect, no proper movement of blood 6: defect which is fixed but it's ok now

14. target	Have heart disease or not (0=no, 1= yes) (= implies the attribute which is predicted

Table 1: Description of attributes of heart disease data

4.2 Feature Types:

1. Categorical Features: It is composed of two or more categories and each value can be categorized accordingly: chest pain(cp), sex
2. Ordinal Features: The variables which have a relative value of ordering or sorting in between the values: fps, slope, thal, target, restecg, ca, exang
3. Continuous Feature: It is that which can take the values in between any two of the points in between maximum or minimum values in the feature column: age, trestbps,chol, thalach , oldpeak

```
In [18]: categor_val = []
         contin_val = []
         for column in data.columns:
             print("----------------------")
             print(f"{column} : {data[column].unique()}")
             if len(data[column].unique()) <= 10:
                 categor_val.append(column)
             else:
                 contin_val.append(column)
```

Figure 4: Code for displaying continuous and categorical values in the data

```
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
----------------------
sex : [1 0]
----------------------
cp : [0 1 2 3]
----------------------
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
----------------------
chol : [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
----------------------
fbs : [0 1]
----------------------
restecg : [1 0 2]
----------------------
thalach : [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
----------------------
exang : [0 1]
----------------------
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
----------------------
slope : [2 0 1]
----------------------
ca : [2 0 1 3 4]
----------------------
thal : [3 2 1 0]
----------------------
target : [0 1]
```

Figure 5: Showing Continuous and Categorical Values of the attributes

Model for Artificial Intelligence

5.1Summary of the approach

The model for the proposed model of this system could be explained following the flow chart given below:

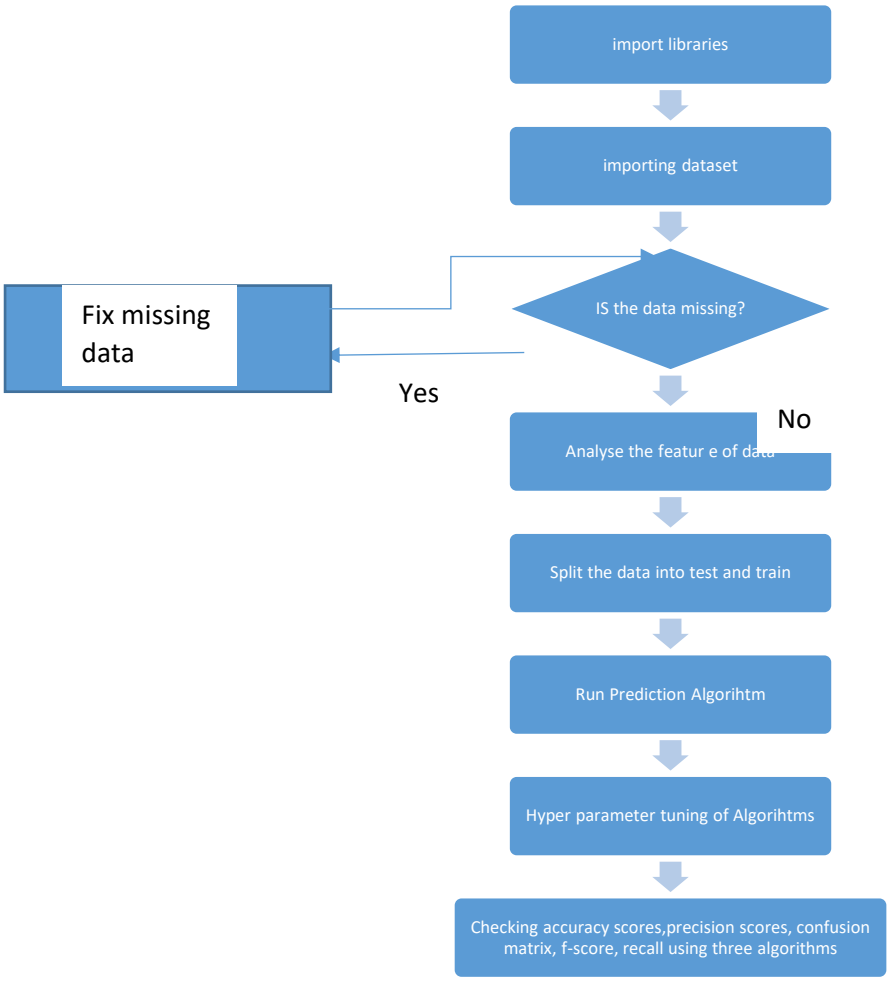


Figure 6: Flow chart of the project

Here at first, the necessary libraries are imported and the heart dataset is also imported accordingly. After that, the statistical analysis of data is done by checking whether any values are missing or not. Then the feature of the dataset is analyzed and data is split into training and testing datasets. After that the prediction algorithm is implemented the hyperparameter tuning of the algorithm is done accordingly and finally, the accuracy scores, precision scores, confusion matrix, f-score, and recall of the three algorithms are computed and the best algorithm among all them is chosen.

5.2 Importing necessary libraries:

The necessary libraries which were used in the project are tabulated below:

Numpy	For working with arrays
Pandas	For working with data frames and csv files
Matplotlib	For creating charts by the use of pyplot.
Warnings	For ignoring all of the warnings which are shown in the notebook due to future/past feature depreciation.
Train_test_split	For splitting the dataset into testing and training data.
StandardScaler	For carrying out the scaling of all of the features, for making the machine learning model adapt better to the dataset.
SciPy	For carrying out the scaling of all of the features, for making the machine learning model adapt better to the dataset.
Seaborn	To carry out the data visualization based upon matplotlib.
RandomizedCV	For carrying out the process of fit and score.
Cufflinks	To bind the power of plotly along with the flexibility of pandas for plotting data easily.

Table 2: Table showing importing of necessary libraries

Heart Disease Prediction

In this machine learning project, I have collected the dataset from Kaggle and I will be using Machine Learning to predict whether any person is suffering from heart disease.

Importing Necessary Libraries

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cufflinks as cf
import hvplot.pandas
from scipy import stats
%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
import warnings
warnings.filterwarnings('ignore')
```

Metrics used for Classification Technique

In [4]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

Scalers

In [5]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import  RandomizedSearchCV, train_test_split
```

Model Building for the system

In [6]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

Figure 7: Necessary Libraries imported for the project

5.3 Data Analysis and Visualization:

The analysis of data has been done accordingly to the medical attributes in the system and the visualization has been done accordingly which is described below:

5.3.1: Importing Dataset:

With the completion of getting data from Kaggle, the data is imported and the quick review of data has been done as follows:

Data import

In [110]:
```
data = pd.read_csv("F:/AI Project Files/Heart Disease Prediction/Datasets/heart.csv")
data.head(6)
```

Out[110]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.000	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.100	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.600	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.000	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.900	1	3	2	0
5	58	0	0	100	248	0	0	122	0	1.000	1	0	2	1

Figure 9: Data importing and quick overlook of the dataset

5.3.2 Analyzing dataset structure and statistical overview of the dataset

Exploratory Data Analysis

```
In [111]: data.shape
Out[111]: (1025, 14)

In [112]: pd.set_option("display.float", "{:.3f}".format)
          data.describe()
```

Out[112]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000	1025.000
mean	54.434	0.696	0.942	131.612	246.000	0.149	0.530	149.114	0.337	1.072	1.385	0.754	2.324	0.513
std	9.072	0.460	1.030	17.517	51.593	0.357	0.528	23.006	0.473	1.175	0.618	1.031	0.621	0.500
min	29.000	0.000	0.000	94.000	126.000	0.000	0.000	71.000	0.000	0.000	0.000	0.000	0.000	0.000
25%	48.000	0.000	0.000	120.000	211.000	0.000	0.000	132.000	0.000	0.000	1.000	0.000	2.000	0.000
50%	56.000	1.000	1.000	130.000	240.000	0.000	1.000	152.000	0.000	0.800	1.000	0.000	2.000	1.000
75%	61.000	1.000	2.000	140.000	275.000	0.000	1.000	166.000	1.000	1.800	2.000	1.000	3.000	1.000
max	77.000	1.000	3.000	200.000	564.000	1.000	2.000	202.000	1.000	6.200	2.000	4.000	3.000	1.000

Figure 10: Understanding the shape of data and statistical computation of data features

Besides the data is also checked whether there is a null value or not

```
In [74]: #Checking Missing Values
         data.isna().sum()

Out[74]: 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
```

Figure 11: Checking null values of data

Here from the above figure, it is clear that there are no null values so the further process of data visualization and model building can be done accordingly.

5.3.3 Computation of correlation between the features:

Correlation for studying the relationship in between different variables which are measured numerically.

memory usage: 112.2 KB

Checking Correlation between various Features

```
In [114]: plt.figure(figsize=(15,12))
          sns.set_context('notebook',font_scale = 1.3)
          sns.heatmap(data.corr(),annot=True,linewidth =1)
          plt.tight_layout()
```

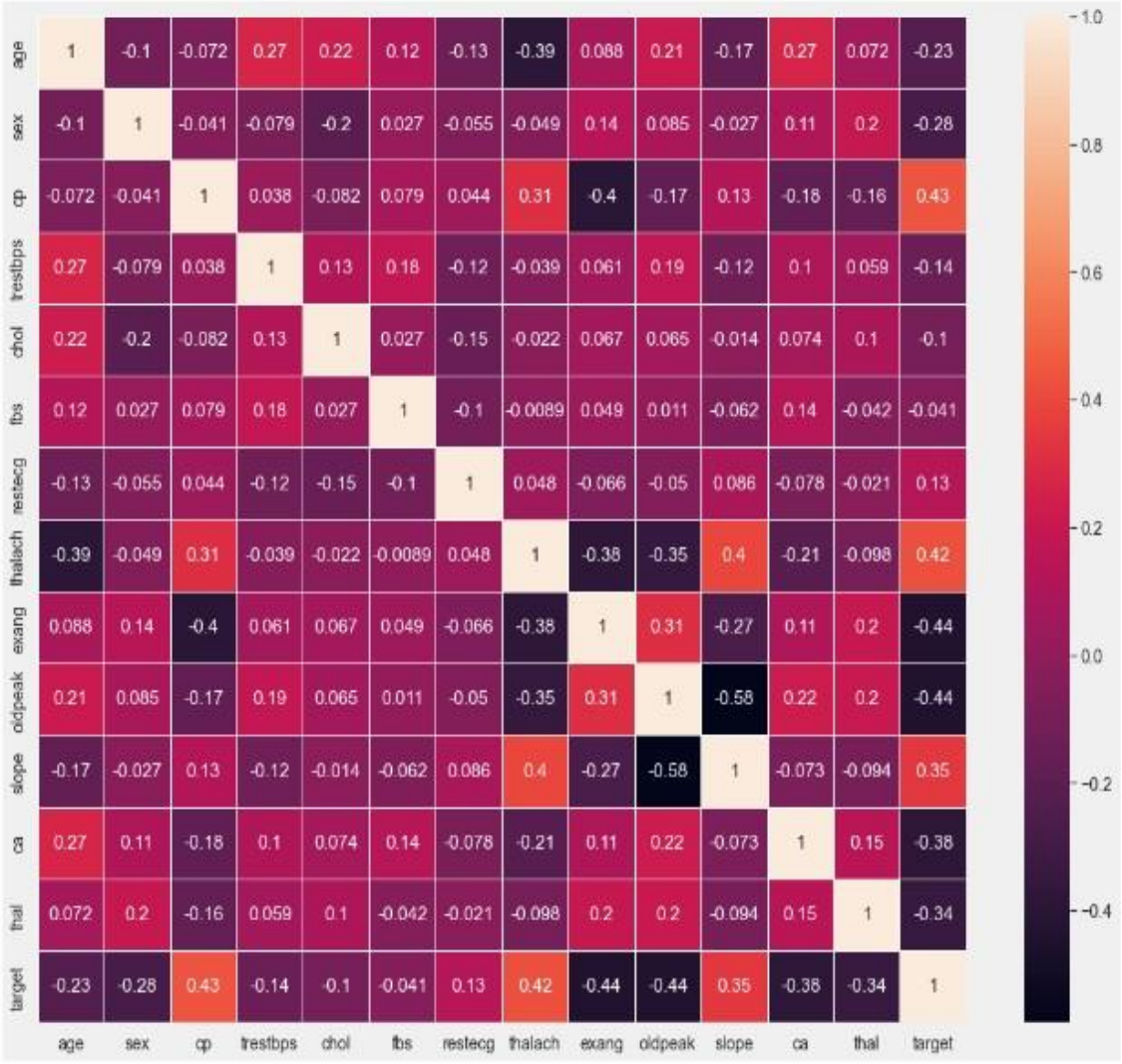


Figure 12: Correlation between the data

5.3.4 Correlation with target value:

```
In [115]: sns.set_context('notebook',font_scale = 2.3)
          data.drop('target', axis=1).corrwith(data.target).plot(kind='bar', grid=True, figsize=(18, 10),
                                                                 title="Correlation with the target feature")
          plt.tight_layout()
```

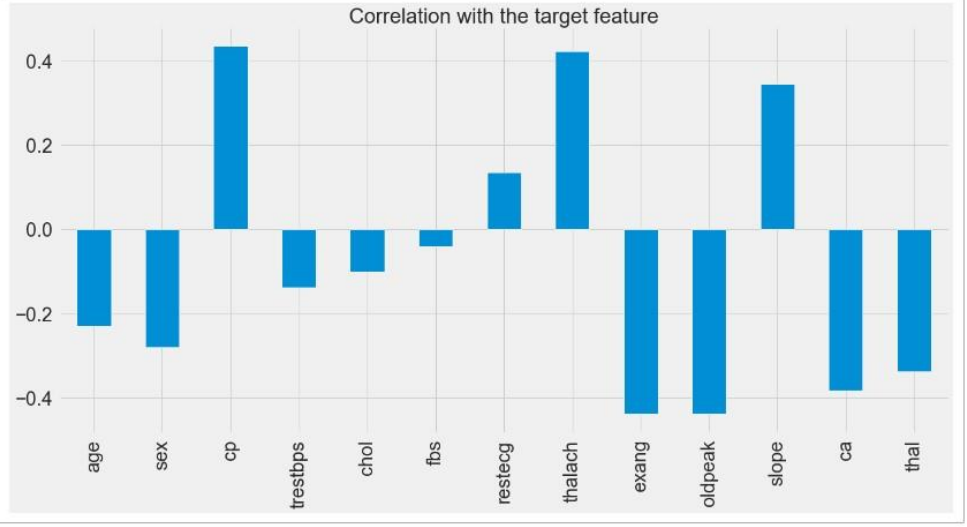


Figure 13: Correlation of other variables with target value

5.3.5 Plotting the correlation of dataset with target value:

```
In [17]: print(data.corr()["target"].abs().sort_values(ascending=False))

target      1.000
oldpeak     0.438
exang       0.438
cp          0.435
thalach     0.423
ca          0.382
slope       0.346
thal        0.338
sex         0.280
age         0.229
trestbps    0.139
restecg     0.134
chol        0.100
fbs         0.041
Name: target, dtype: float64
```

Figure 14: Correlation of target variables with other attributes

The variables are very less correlated and some even have negative correlation.

5.3.6 Checking the range of 10 ages and their corresponding counts

Analysis of Age

```
In [116]: plt.figure(figsize=(25,12))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=data.age.value_counts()[:10].index,y=data.age.value_counts()[:10].values)
plt.tight_layout()
```

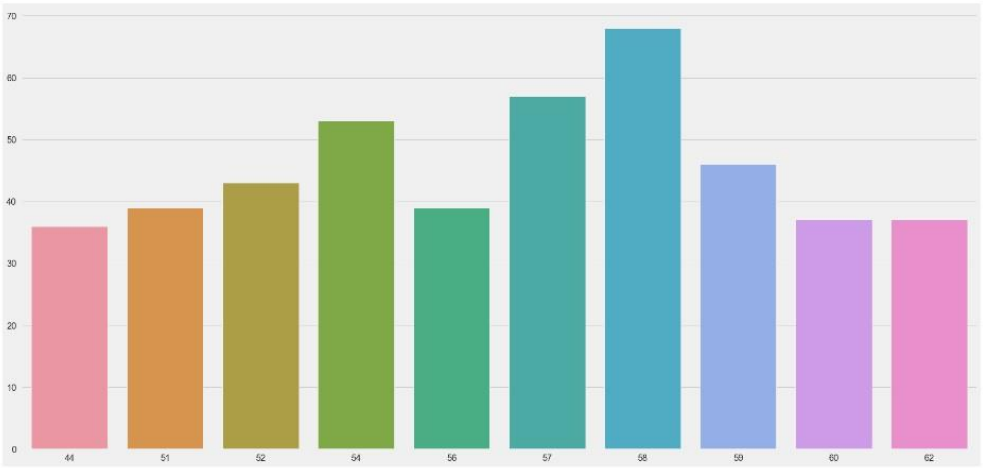


Figure 16: Age analysis

From the above figure, it can be seen that the age column of 58 possesses the highest value of frequency.

5.3.7 Checking the dataset range for age

```
In [117]: #Checking the range of age in the data
minAge=min(data.age)
maxAge=max(data.age)
meanAge=data.age.mean()
print('Min Age :',minAge)
print('Max Age :',maxAge)
print('Mean Age :',meanAge)
```

```
Min Age : 29
Max Age : 77
Mean Age : 54.43414634146342
```

Figure 17: Checking the range of age

Here the minimum age computed is 29 years, the maximum age is 77 years and the average age is 45.32 years.

5.3.8 Dividing the age feature into three different parts which are: young, middle, and elder

```
In [118]: #Dividing the age feature into three parts
Young = data[(data.age>=29)&(data.age<40)]
Middle = data[(data.age>=40)&(data.age<55)]
Elder = data[(data.age>55)]

plt.figure(figsize=(23,10))
sns.set_context('notebook',font_scale = 1.5)
sns.barplot(x=['young ages','middle ages','elderly ages'],y=[len(Young),len(Middle),len(Elder)])
plt.tight_layout()
```

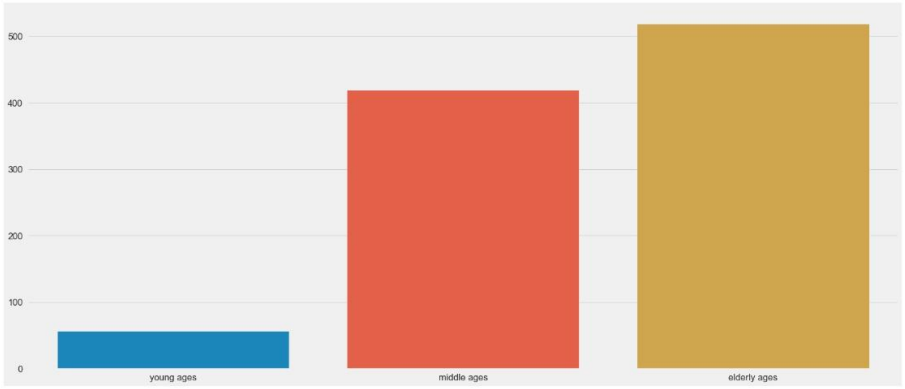


Figure 18: Division of age features into three different categories

From the above figure, it can be seen that the elderly people are the ones who are most affected by heart disease and the young ages are the ones who are least affected. The above statistics can also be drawn by using a pie chart as follows:

```
In [119]: #Plotting Pie Chart
colors = ['blue','green','yellow']
explode = [0,0,0.1]
plt.figure(figsize=(10,10))
sns.set_context('notebook',font_scale = 1.2)
plt.pie([len(Young),len(Middle),len(Elder)],labels=['young ages','middle ages','elderly ages'],explode=explode,colors=colors, au
plt.tight_layout()
```

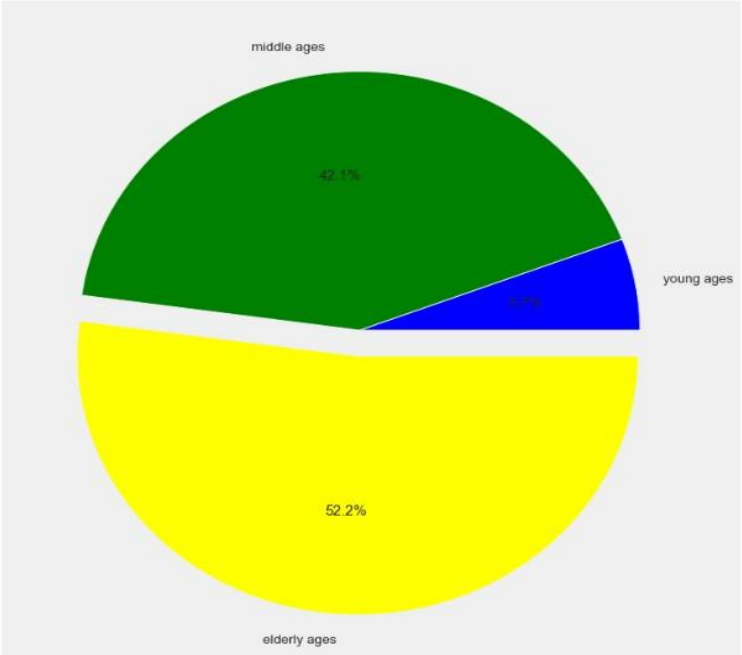


Figure 19: Pie-Chart plotting the figures between different ages

5.3.9 Visualization of the distribution of continuous and categorical attributes of data

```
In [62]: def plotGrid(isCategorial):
    if isCategorial:
        [plotCategorial(x[0], x[1], i) for i, x in enumerate(categorial)]
    else:
        [plotContinuous(x[0], x[1], i) for i, x in enumerate(continuous)]
```

```
In [63]: continuous = [('trestbps', 'blood pressure in mm Hg'),
              ('chol', 'serum cholestoral in mg/d'),
              ('thalach', 'maximum heart rate achieved'),
              ('oldpeak', 'ST depression by exercise relative to rest'),
              ('ca', '# major vessels: (0-3) colored by flourosopy')]
```

```
In [65]: def plotContinuous(attribute, xlabel, ax_index):
    sns.distplot(data[[attribute]], ax=axes[ax_index][0])
    axes[ax_index][0].set(xlabel=xlabel, ylabel='density')
    sns.violinplot(x='target', y=attribute, data=data, ax=axes[ax_index][1])
```

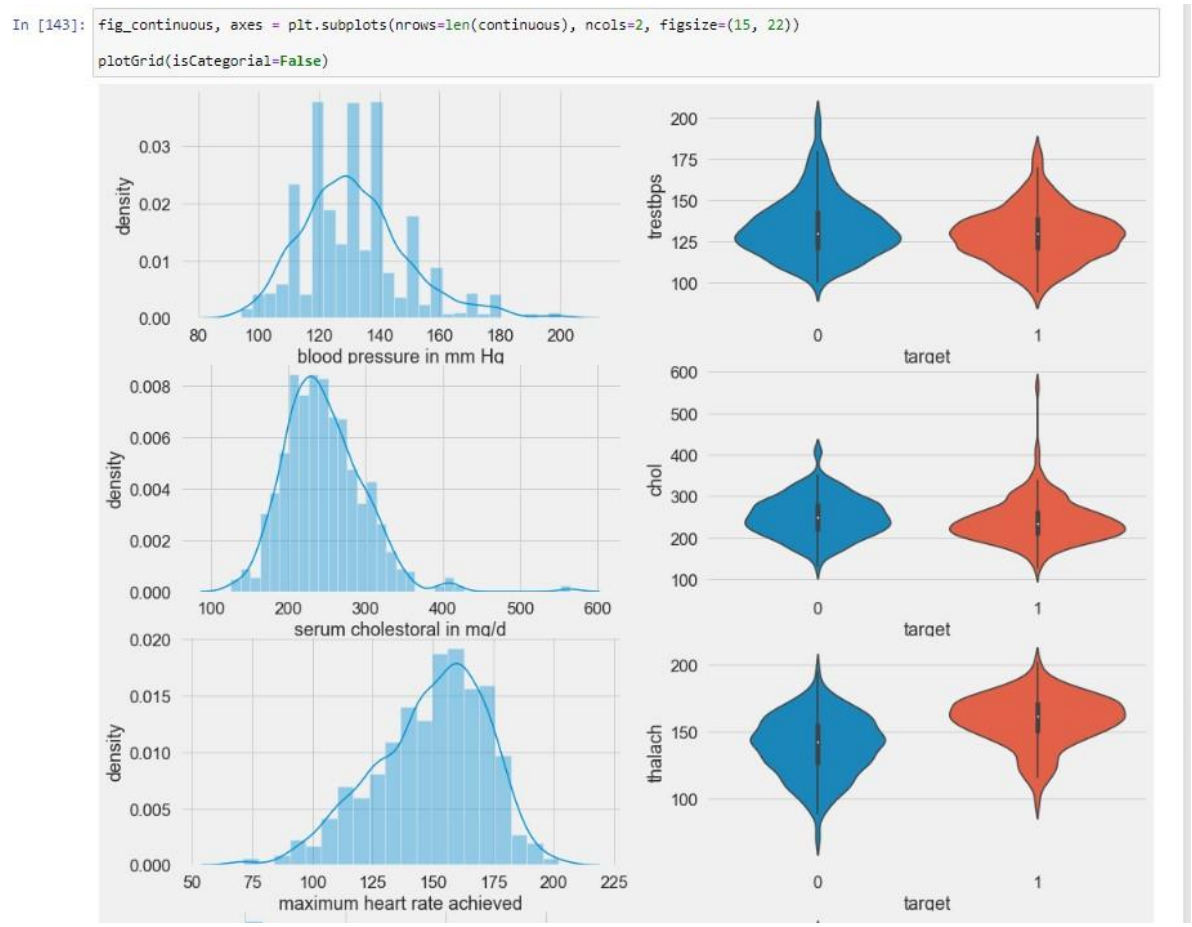


Figure 20: Visualization of the distribution of continuous and categorical attributes of datasets

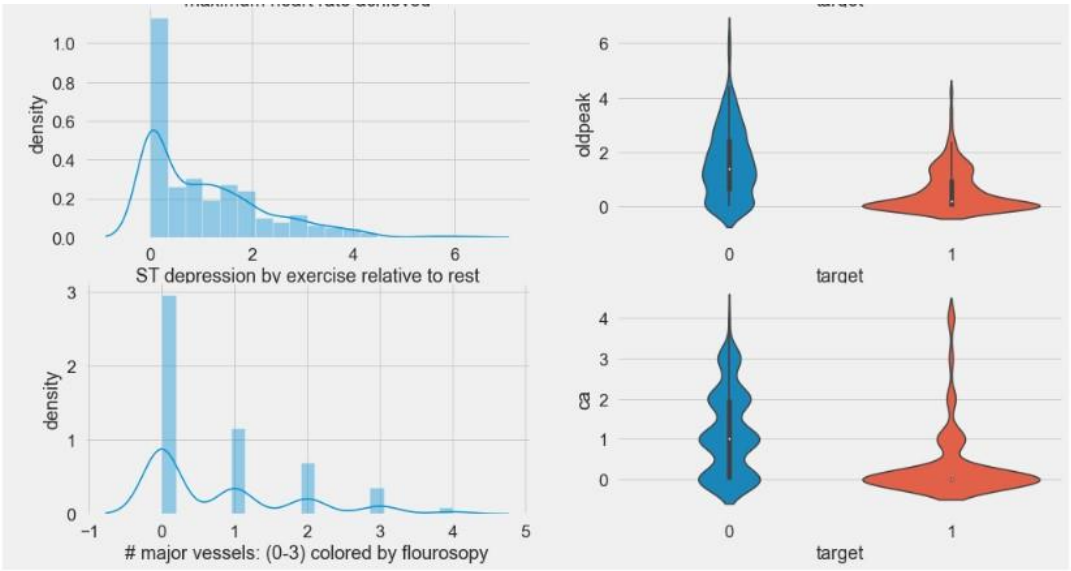


Figure 21: Visualization of continuous and categorical attributes of datasets

5.3.10: Analysis of sex feature

Analysis of Sex Feature

```
In [12]: plt.figure(figsize=(18,9))
         sns.set_context('notebook',font_scale = 1.5)
         sns.countplot(data['sex'])
         plt.tight_layout()
```

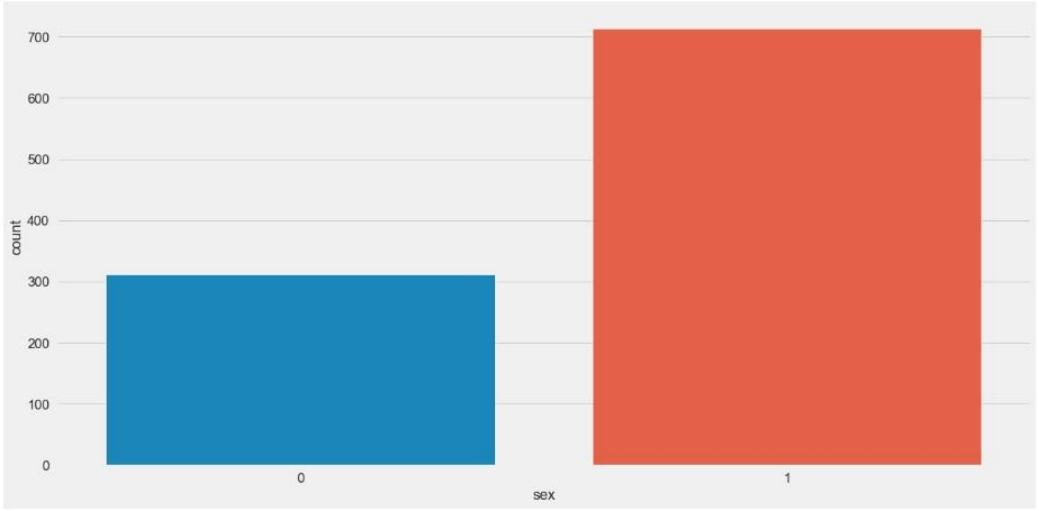


Figure 22: Analysis of Sex Feature

From the above figure, it is seen that the Ratio of Male to females is approximately 2:1.

5.3.11: Plotting the relation between slope and sex

Plotting relation between sex and Slope

```
In [67]: plt.figure(figsize=(18,9))
         sns.set_context('notebook',font_scale = 1.6)
         sns.countplot(data['sex'],hue=data["slope"])
         plt.tight_layout()
```

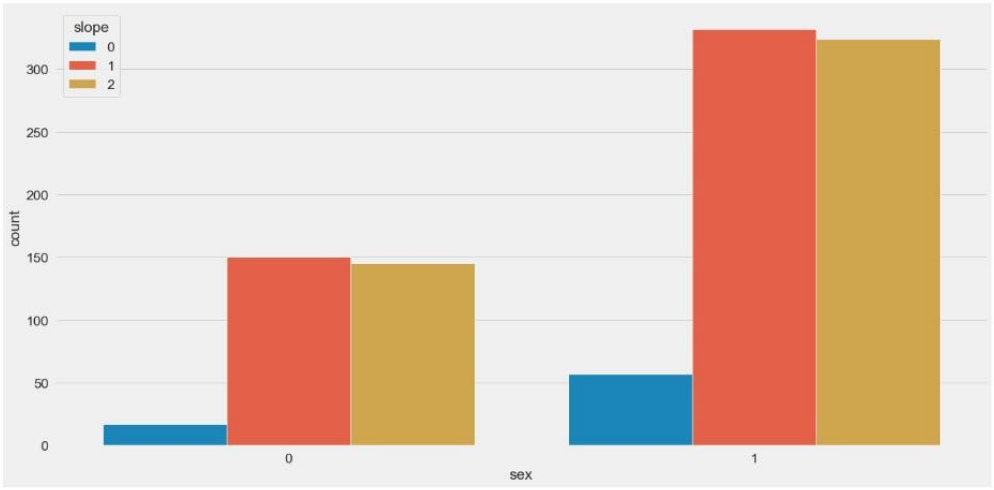


Figure 22: Relation between sex and slope

From the above figure, it is seen that the value of the slope is higher is great in cases of males which is denoted by 1 in the bar graph.

5.3.12 Chest pain (cp) Analysis:

Analysis of Chest Pain

```
In [122]: plt.figure(figsize=(18,9))
          sns.set_context('notebook',font_scale = 1.5)
          sns.countplot(data['cp'])
          plt.tight_layout()
```

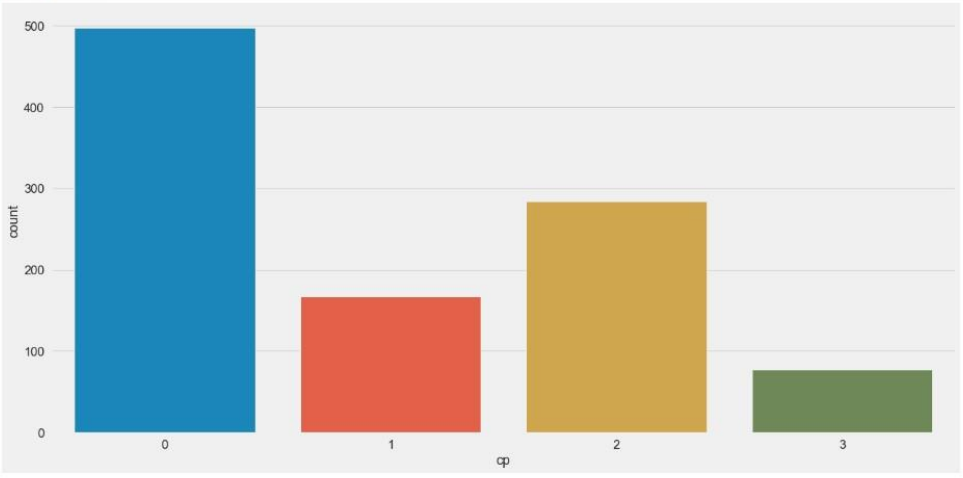


Figure 23: Count Analysis of Chest Pain

Here, 0 represents the least status of chest pain, 1 represents the group who have chest pain condition which is slightly distressed, 2 represents the medium condition and 3 represents the worst condition of chest pain.

5.3.13 Analysis of Chest Pain with target value:

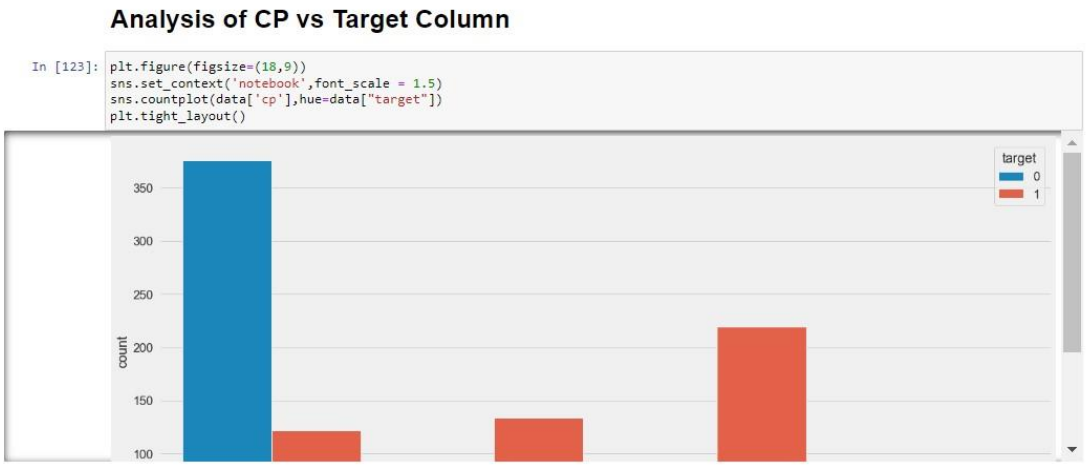


Figure 23: Comparison of Chest Pain with target value

From the above figure, it can be seen that those who have the least chest pain are not probable to have heart disease while on other hand the people who have severe chest pain are likable to have heart disease.

5.3.14 Analysis of Thal:

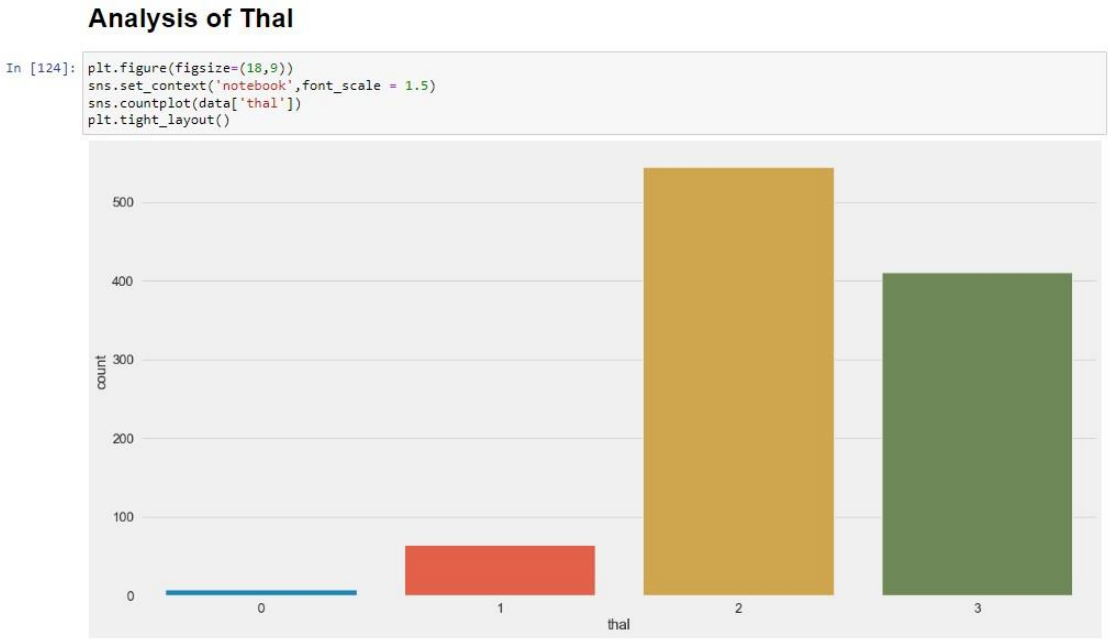


Figure: 24 Thal count visualization

5.3.15 Plotting the relation between heart disease concerning ECG results

```
In [15]: # Plotting the results for heart disease by resting ECG
have_disease = data.loc[data['target']==1, 'restecg'].value_counts().hvplot.bar(alpha=0.4)
no_disease = data.loc[data['target']==0, 'restecg'].value_counts().hvplot.bar(alpha=0.4)

(no_disease * have_disease).opts(
    title="Heart Disease by resting electrocardiographic results", xlabel='resting electrocardiographic results',
    ylabel='Count', width=500, height=450, legend_cols=2, legend_position='top_right'
)
```

Out[15]:

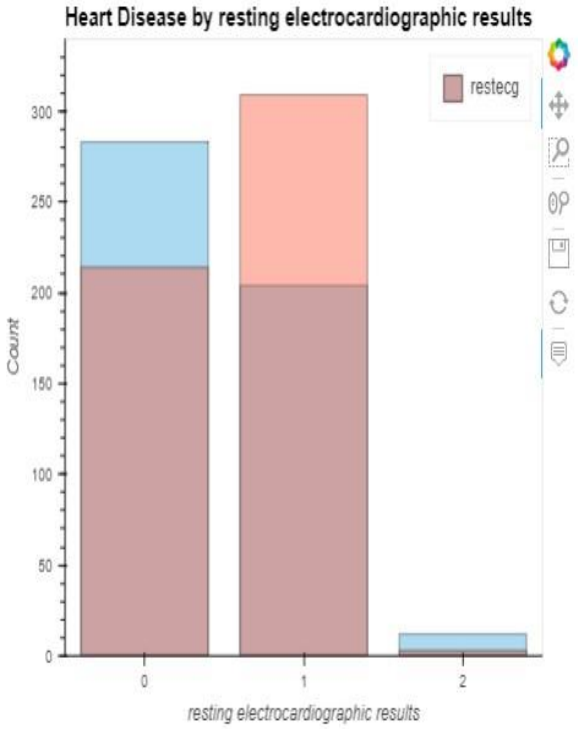


Figure 25: Count of Heart Disease vs ECG results

5.3.16 Plotting the results of count having heart disease or not for all categorical values:

```
In [78]: plt.figure(figsize=(15, 15))

for i, column in enumerate(categor_val, 1):
    plt.subplot(3, 3, i)
    data[data["target"] == 0][column].hist(bins=32, color='green', label='Have Heart Disease = NO', alpha=0.6)
    data[data["target"] == 1][column].hist(bins=32, color='yellow', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```

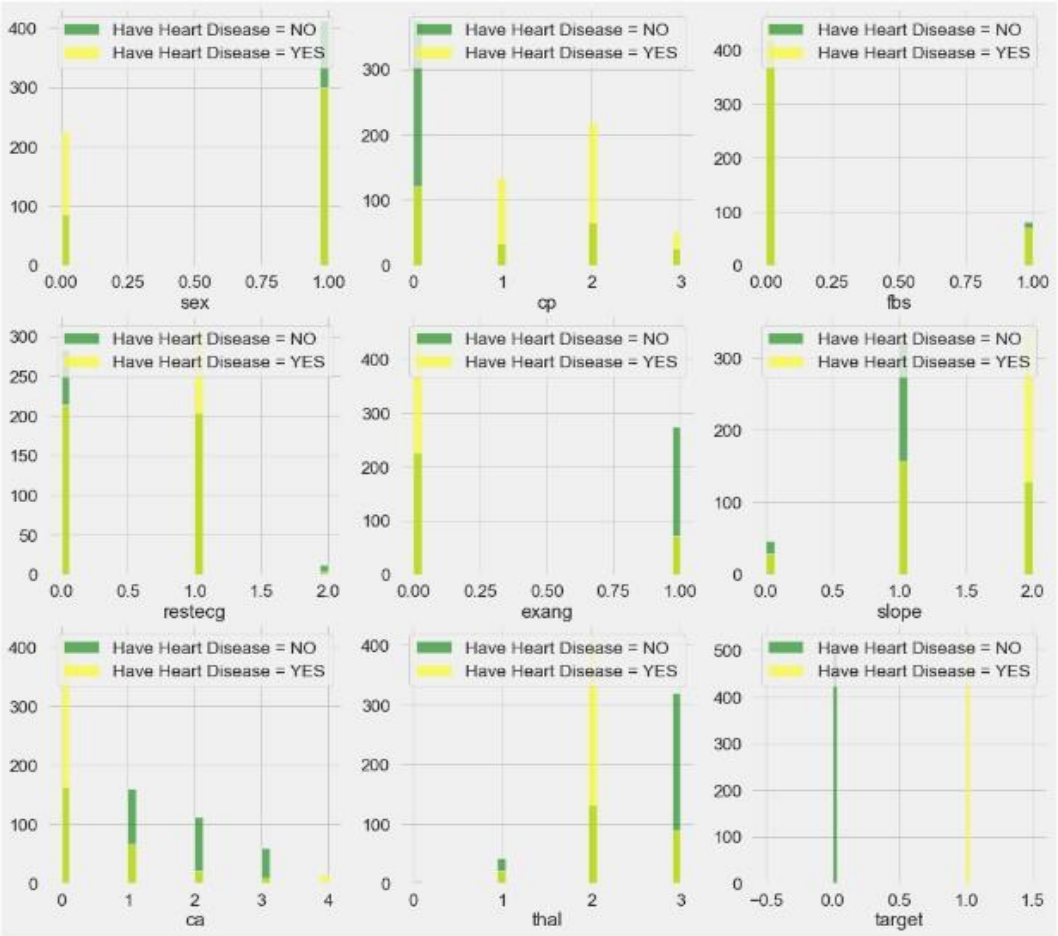


Figure 26: Count of people having heart disease or not based upon categorical features

5.3.17: Visualizing the frequency of age concerning the target value for having heart disease or not

```
In [144]: pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))
          plt.title('Heart Disease Frequency for Ages')
          plt.xlabel('Age')
          plt.ylabel('Frequency')
          plt.savefig('heartDiseaseAndAges.png')
          plt.show()
```

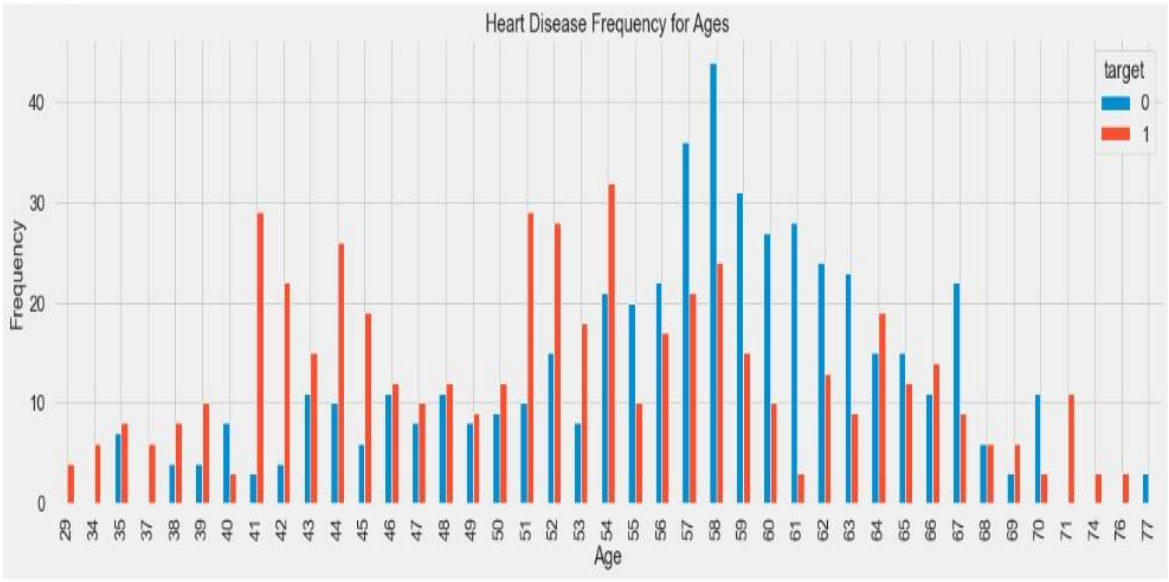


Figure 27: Plot showing the frequency of ages having or not having heart diseases

It can be seen that the maximum frequency of people having heart disease is at the age of 54 and for age 58 minimum frequency of people is having no heart disease.

5.3.18: Visualization of maximum heart rate and heart disease using scatter plot

Heart Disease vs Maximum Heart Rate

In [36]:
```
# Create another figure
plt.figure(figsize=(9, 7))

# Scatter with postivie examples
plt.scatter(data.age[data.target==1],
            data.thalach[data.target==1],
            c="salmon")

# Scatter with negative examples
plt.scatter(data.age[data.target==0],
            data.thalach[data.target==0],
            c="lightblue")

# Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease", "No Disease"]);
```

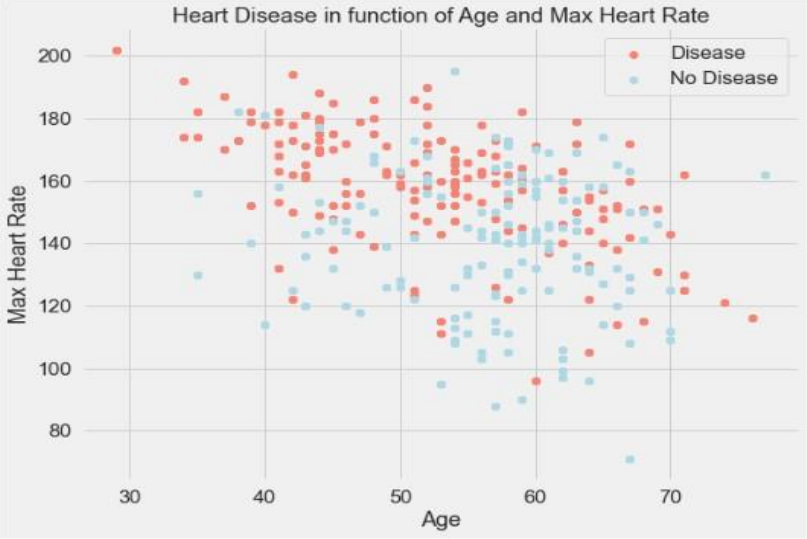


Figure 28: Scatter plot showing maximum heart rate, age, and person having a disease or not

From the above figure, it can be seen that for the age range 40-60 there is the maximum number of people who have heart disease and it also depicts that for this range the density of people having heart disease is maximum for heart rate ranging from 140 to approximately 190.

5.3.19 Visualization of the distribution of Thal(Thalassemia)

Analysing the disorder in blood thalassemia

```
In [147]: data["thal"].unique()

Out[147]: array([3, 2, 1, 0], dtype=int64)

In [148]: sns.distplot(data["thal"])

Out[148]: <AxesSubplot:xlabel='thal', ylabel='Density'>
```

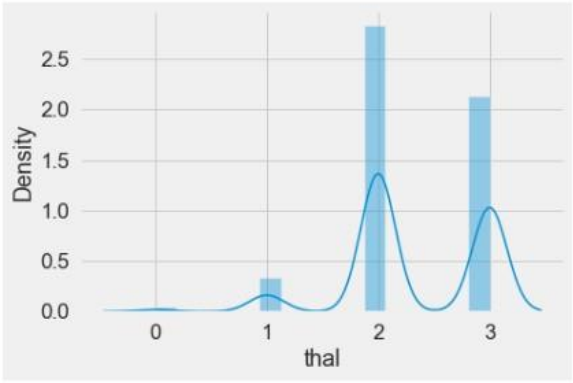


Figure 29: Thalassemia data density plot

5.3.20 Visualization of that with the target variable

Comparing with target

```
In [97]: sns.barplot(data["thal"],y)

Out[97]: <AxesSubplot:xlabel='thal', ylabel='target'>
```

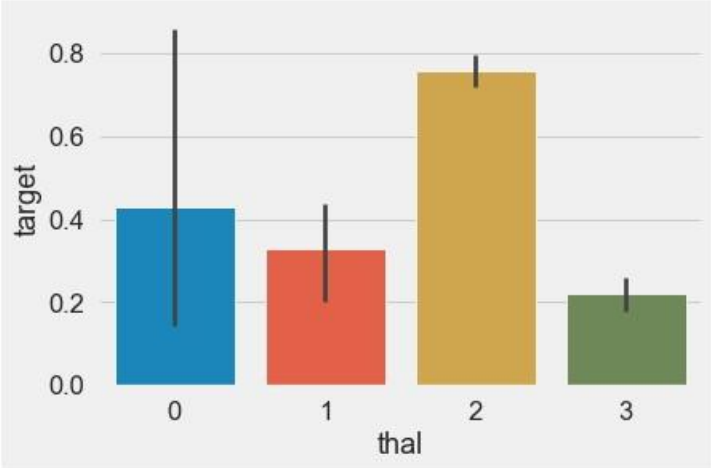


Figure 30: Visualization of Thal with the target variable

5.3.21: Analysis of Target variable:

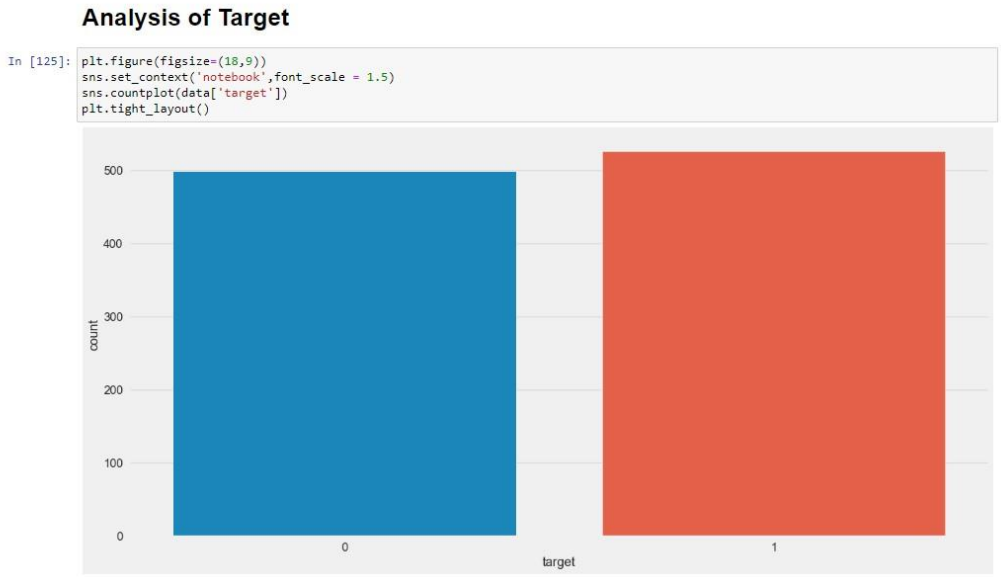


Figure 31: Analysis of count of the target variable

```
In [72]: data.target.value_counts()
Out[72]: 1    526
         0    499
         Name: target, dtype: int64
```

Figure 32: Counts of 0 and 1 value of target variable

Since the ratio of 1 and that of 0 is very much less than that of 1.5, which delineates that there is no imbalance in the target feature. As the data is balanced we can make use of an accuracy score to evaluate the metrics for our proposed model.

5.4 Data Preparation:

Here the array of the categorical and continuous attribute is defined and the attributes of the data are differentiated and stored in the corresponding arrays, Then the arrays are printed which is depicted in the figure below:

Processing of Data

```
In [147]: categor_val = []
          contin_val = []
          for column in data.columns:
              print("**************")
              print(f"{column} : {data[column].unique()}")
              if len(data[column].unique()) <= 10:
                  categor_val.append(column)
              else:
                  contin_val.append(column)
```

```
**************
age : [52 53 70 61 62 58 55 46 54 71 43 34 51 50 60 67 45 63 42 44 56 57 59 64
 65 41 66 38 49 48 29 37 47 68 76 40 39 77 69 35 74]
**************
sex : [1 0]
**************
cp : [0 1 2 3]
**************
trestbps : [125 140 145 148 138 100 114 160 120 122 112 132 118 128 124 106 104 135
 130 136 180 129 150 178 146 117 152 154 170 134 174 144 108 123 110 142
 126 192 115  94 200 165 102 105 155 172 164 156 101]
**************
chol : [212 203 174 294 248 318 289 249 286 149 341 210 298 204 308 266 244 211
 185 223 208 252 209 307 233 319 256 327 169 131 269 196 231 213 271 263
 229 360 258 330 342 226 228 278 230 283 241 175 188 217 193 245 232 299
 288 197 315 215 164 326 207 177 257 255 187 201 220 268 267 236 303 282
 126 309 186 275 281 206 335 218 254 295 417 260 240 302 192 225 325 235
 274 234 182 167 172 321 300 199 564 157 304 222 184 354 160 247 239 246
 409 293 180 250 221 200 227 243 311 261 242 205 306 219 353 198 394 183
 237 224 265 313 340 259 270 216 264 276 322 214 273 253 176 284 305 168
 407 290 277 262 195 166 178 141]
**************
fbs : [0 1]
**************
restecg : [1 0 2]
**************
thalach : [168 155 125 161 106 122 140 145 144 116 136 192 156 142 109 162 165 148
 172 173 146 179 152 117 115 112 163 147 182 105 150 151 169 166 178 132
 160 123 139 111 180 164 202 157 159 170 138 175 158 126 143 141 167  95
 190 118 103 181 108 177 134 120 171 149 154 153  88 174 114 195 133  96
 124 131 185 194 128 127 186 184 188 130  71 137  99 121 187  97  90 129
 113]
**************
exang : [0 1]
**************
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
```

```
**************
oldpeak : [1.  3.1 2.6 0.  1.9 4.4 0.8 3.2 1.6 3.  0.7 4.2 1.5 2.2 1.1 0.3 0.4 0.6
 3.4 2.8 1.2 2.9 3.6 1.4 0.2 2.  5.6 0.9 1.8 6.2 4.  2.5 0.5 0.1 2.1 2.4
 3.8 2.3 1.3 3.5]
**************
slope : [2 0 1]
**************
ca : [2 0 1 3 4]
**************
thal : [3 2 1 0]
**************
target : [0 1]
```

Figure 32: Categorical and Continuous data stored in the corresponding array

Then the target column is removed from the set of categorical features by the use of the method of getting dummies. Besides it also creates a separate definite column for each category.

```
In [98]: categor_val.remove('target')
         dataset = pd.get_dummies(data, columns = categor_val)

In [99]: dataset.head()
```

Out[99]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0
0	52	125	212	168	1.000	0	0	1	1	0	...	1	0	0	1	0	0	0
1	53	140	203	155	3.100	0	0	1	1	0	...	0	1	0	0	0	0	0
2	70	145	174	125	2.600	0	0	1	1	0	...	0	1	0	0	0	0	0
3	61	148	203	161	0.000	0	0	1	1	0	...	1	0	1	0	0	0	0
4	62	138	294	106	1.900	0	1	0	1	0	...	0	0	0	0	1	0	0

5 rows × 31 columns

```
In [100]: print(data.columns)
          print(dataset.columns)

Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
Index(['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'target', 'sex_0',
       'sex_1', 'cp_0', 'cp_1', 'cp_2', 'cp_3', 'fbs_0', 'fbs_1', 'restecg_0',
       'restecg_1', 'restecg_2', 'exang_0', 'exang_1', 'slope_0', 'slope_1',
       'slope_2', 'ca_0', 'ca_1', 'ca_2', 'ca_3', 'ca_4', 'thal_0', 'thal_1',
       'thal_2', 'thal_3'],
      dtype='object')
```

Figure 33: Removing target variable from categorical value and visualizing the data

After that, the method of Standard Scaler is employed for scaling down data so that it wouldn't increase the outliers and also get better accuracy after scaling the data to general units.

```
In [40]: from sklearn.preprocessing import StandardScaler

         s_sc = StandardScaler()
         col_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
         dataset[col_to_scale] = s_sc.fit_transform(dataset[col_to_scale])

In [41]: dataset.head()
```

Out[41]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slope_2	ca_0	ca_1	ca_2	ca_3	ca_4	thal_0	thal_1	thal_2	thal_3
0	-0.268	-0.378	-0.659	0.821	-0.061	0	0	1	1	0	...	1	0	0	1	0	0	0	0	0	1
1	-0.158	0.479	-0.834	0.256	1.727	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
2	1.717	0.765	-1.396	-1.049	1.301	0	0	1	1	0	...	0	1	0	0	0	0	0	0	0	1
3	0.724	0.936	-0.834	0.517	-0.912	0	0	1	1	0	...	1	0	1	0	0	0	0	0	0	1
4	0.834	0.365	0.931	-1.875	0.705	0	1	0	1	0	...	0	0	0	0	1	0	0	0	1	0

5 rows × 31 columns

Figure 34: Standard Scaling of the data

Model Building

Since the primary target of the project is to do the prediction of occurrence of heart disease with maximum accuracy so for achieving it, three ML (Machine Learning) algorithms namely, logistic regression, K-nearest neighbor, and Support Vector Machine algorithm have been implemented. Besides, our problem statement is to classify whether a person has heart disease not based upon characteristic medical attributes, it is a binary classification due to which the classification algorithms have been used rather than other machine learning algorithms.

Here for this purpose, the usage of SciKit Learn and other Libraries has been done so that a general function could be formulated for training our model. Similarly, the data is categorized into two data types which are dependent and independent variables and they are further classified into training and testing data. Here the size of testing data is 30% and training data size is 70%, and the random state is selected to be 44.

The primary purpose of generating accuracy on training and test sets is to find whether the created model under fits or overfits.

Building Model

In [45]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        prediction = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, prediction, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, prediction) * 100:.2f}%")
        print("************************************")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("************************************************")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, prediction)}\n")


    elif train==False:
        prediction = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, prediction, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, prediction) * 100:.2f}%")
        print("************************************************")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("************************************************")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, prediction)}\n")
```

In [53]:
```
from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=44)
```

Figure 35: Building model of the system and splitting data into training and testing

5.6.1 Logistic Regression

It is that which measures the probability of occurring an event based on the dataset which have independent variables. Since the outcome is that of probability, there is a binding value between 0 and 1. The pseudo-code which is implemented in the project is given below:

1. Dividing the classification problem into a total of n+1 number
2. For each class the problem is divided into an n+1 number
3. Prediction of observations and probability which are in the same class
4. Prediction=<math>maximum

Logistic Regression

```
In [84]: from sklearn.linear_model import LogisticRegression

         lr_clf = LogisticRegression(solver='newton-cg')
         lr_clf.fit(X_train, y_train)
         y_pred_rf = lr_clf.predict(X_test)
         print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
         print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
         print( y_pred_lr)

         Train Result:
         ================================================
         Accuracy Score: 88.01%
         ************************************
         CLASSIFICATION REPORT:
                              0       1  accuracy  macro avg  weighted avg
         precision    0.892   0.870     0.880      0.881         0.881
         recall       0.850   0.907     0.880      0.879         0.880
         f1-score     0.871   0.888     0.880      0.879         0.880
         support    341.000 376.000     0.880    717.000       717.000
         ************************************************
         Confusion Matrix:
          [[290  51]
          [ 35 341]]

         Test Result:
         ================================================
         Accuracy Score: 86.04%
         ****************************************
         CLASSIFICATION REPORT:
                              0       1  accuracy  macro avg  weighted avg
         precision    0.897   0.828     0.860      0.862         0.863
         recall       0.823   0.900     0.860      0.861         0.860
         f1-score     0.858   0.863     0.860      0.860         0.860
         support    158.000 150.000     0.860    308.000       308.000
         ****************************************
         Confusion Matrix:
          [[130  28]
          [ 15 135]]

         [1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 0
          1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1
          0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1
          0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1 1 0 0
          1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0 1
          1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0 1 1 1
          0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0
          1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 1
          1 0 1 0 1 1 0 1 0 1 1 0]
```

Figure 36: Logistic Regression implementation with accuracy and classification report

Here from logistic Regression, the accuracy score for testing the data is 86.04%.

```
In [27]: tested_score = accuracy_score(y_test, lr_clf.predict(X_test)) * 100
         trained_score = accuracy_score(y_train, lr_clf.predict(X_train)) * 100

         results_df = pd.DataFrame(data=[["Logistic Regression", trained_score, tested_score]],
                                   columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
         results_df
```

Out[27]:

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	88.006	86.039

```
In [28]: matrix= confusion_matrix(y_test, y_pred_rf)
         sns.heatmap(matrix,annot = True, fmt = "d")
```

Out[28]: <AxesSubplot:>

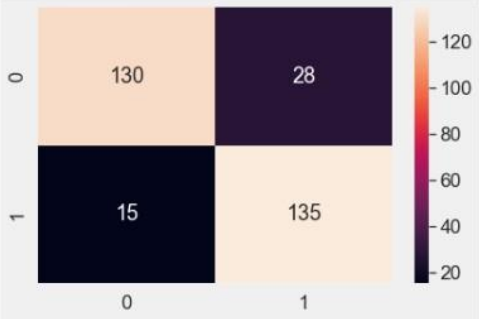


Figure 37: Accuracy and Confusion matrix for testing the algorithm

2. K-nearest Neighbor:

It is that supervised learning which is dependent upon a labeled dataset and it relies on labeled input data for learning the functionality which produces an appropriate output. The corresponding pseudo-code used for this algorithm is given below:

1. Loading the data
2. Initialization carried out to the value of K
3. To get the predicted class, iterating over form 1 – total training data points number
    - Calculation of distance in between each row for training data as well as testing data.
    - Sorting the computed distance in corresponding ascending order
    - Getting the value of the top K rows of the values in the array which is sorted
    - Getting the class of these sorted arrays which is very frequent.
    - Returning to the stipulated class.

K-nearest neighbors

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)
y_pred_lr = knn_clf.predict(X_test)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
print(y_pred_lr)
```

```
Train Result:
================================================
Accuracy Score: 93.31%
************************************
CLASSIFICATION REPORT:
                0       1  accuracy  macro avg  weighted avg
precision   0.925   0.941     0.933      0.933         0.933
recall      0.935   0.931     0.933      0.933         0.933
f1-score    0.930   0.936     0.933      0.933         0.933
support   341.000 376.000     0.933    717.000       717.000
************************************************
Confusion Matrix:
 [[319  22]
 [ 26 350]]

Test Result:
================================================
Accuracy Score: 83.44%
****************************************
CLASSIFICATION REPORT:
                0       1  accuracy  macro avg  weighted avg
precision   0.845   0.824     0.834      0.834         0.835
recall      0.829   0.840     0.834      0.835         0.834
f1-score    0.837   0.832     0.834      0.834         0.834
support   158.000 150.000     0.834    308.000       308.000
****************************************
Confusion Matrix:
 [[131  27]
 [ 24 126]]

[1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 0
 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1
 0 1 0 1 1 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1
 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1 1 1 0 0
 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1
 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 0 1 1 0 0 0 1 1 1
 0 1 1 0 0 1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 0 0
 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 1
 1 0 1 0 1 1 0 1 0 1 1 0]
```

Figure 37: KNN algorithm implementation with accuracy and classification report

Here the accuracy of KNN for the test result is 83.44%.

In [60]:
```
test_score = accuracy_score(y_test, knn_clf.predict(X_test)) * 100
train_score = accuracy_score(y_train, knn_clf.predict(X_train)) * 100

results_df_2 = pd.DataFrame(data=[["K-nearest neighbors", train_score, test_score]],
                            columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
results_df = results_df.append(results_df_2, ignore_index=True)
results_df
```

Out[60]:

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	88.006	86.039
1	K-nearest neighbors	93.305	83.442
2	Support Vector Machine	95.397	91.883
3	K-nearest neighbors	85.774	85.065

In [31]:
```
matrix= confusion_matrix(y_test,y_pred_lr )
sns.heatmap(matrix,annot = True, fmt = "d")
```

Out[31]: <AxesSubplot:>

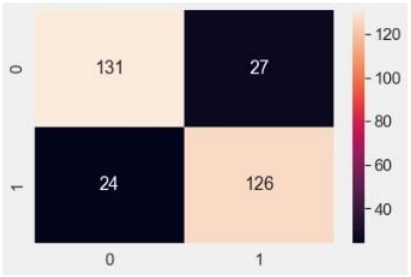


Figure 38: Accuracy and Confusion matrix for testing data of KNN algorithm

Support Vector Machine:

It is that algorithm of supervised machine learning where which is associated with corresponding learning algorithms for analysis of classification data along with carrying out the regression analysis.

Support Vector Machine

```
In [32]: from sklearn.svm import SVC


svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)
svm_clf.fit(X_train, y_train)
y_pred_mr = svm_clf.predict(X_test)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
print (y_pred_mr)
```

```
Train Result:
================================================
Accuracy Score: 95.40%
************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       0.964   0.945     0.954      0.955         0.954
recall          0.938   0.968     0.954      0.953         0.954
f1-score        0.951   0.957     0.954      0.954         0.954
support       341.000 376.000     0.954    717.000       717.000
************************************************
Confusion Matrix:
 [[320  21]
 [ 12 364]]

Test Result:
================================================
Accuracy Score: 91.88%
********************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       0.940   0.898     0.919      0.919         0.920
recall          0.899   0.940     0.919      0.919         0.919
f1-score        0.919   0.919     0.919      0.919         0.919
support       158.000 150.000     0.919    308.000       308.000
********************************************
Confusion Matrix:
 [[142  16]
 [  9 141]]

[1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 0
 1 0 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1
 1 1 0 0 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1
 1 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 0 0
 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1
 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1 1 0
 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 0 0
 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1 0 1
 0 0 1 0 1 1 0 1 0 1 1 1]
```

Figure 39: Support Vector Machine implementation with accuracy and classification report

Here, the accuracy of the support vector machine for the test result is found to be 91.88%

```
In [114]: matrix= confusion_matrix(y_test,y_pred_mr )
          sns.heatmap(matrix,annot = True, fmt="d")

Out[114]: <AxesSubplot:>
```

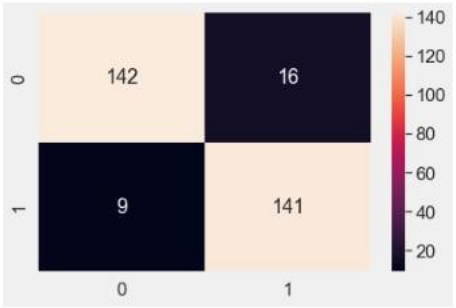


Figure 40: Confusion Matrix for SVM algorithm

Comparative training and testing score the implemented algorithm

```
In [33]: test_score = accuracy_score(y_test, svm_clf.predict(X_test)) * 100
         train_score = accuracy_score(y_train, svm_clf.predict(X_train)) * 100

         results_df_2 = pd.DataFrame(data=[["Support Vector Machine", train_score, test_score]],
                                     columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
         results_df = results_df.append(results_df_2, ignore_index=True)
         results_df
```

Out[33]:

	Model	Training Accuracy %	Testing Accuracy %
0	Logistic Regression	88.006	86.039
1	K-nearest neighbors	93.305	83.442
2	Support Vector Machine	95.397	91.883

Figure 41: Comparative accuracy of training and testing accuracy for logistic regression, K-nearest neighbor, and Support Vector Machine

Hyper Parameter Tuning:

Here the hyperparameter tuning for all of the algorithms is done based upon GridSearchCV and the optimized parameters for all of the algorithms are computed accordingly.

Hyper Parameter Tuning of Logistic Regression:

Logistic Regression Hypter Parameter Tuning

In [85]:
```
from sklearn.model_selection import GridSearchCV

params = {"C": np.logspace(-4, 4, 20),
          "solver": ["newton-cg"]}

lr_clf = LogisticRegression()

lr_cv = GridSearchCV(lr_clf, params, scoring="accuracy", n_jobs=-1, verbose=1, cv=5)
lr_cv.fit(X_train, y_train)
x_pred_mr = lr_cv.predict(X_test)
best_params = lr_cv.best_params_
print(f"Best parameters: {best_params}")
lr_clf = LogisticRegression(**best_params)

lr_clf.fit(X_train, y_train)

print_score(lr_clf, X_train, y_train, X_test, y_test, train=True)
print_score(lr_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best parameters: {'C': 545.5594781168514, 'solver': 'newton-cg'}
Train Result:
================================================
Accuracy Score: 88.42%
************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision   0.901   0.871     0.884      0.886         0.885
recall      0.850   0.915     0.884      0.883         0.884
f1-score    0.875   0.892     0.884      0.884         0.884
support   341.000 376.000     0.884    717.000       717.000
********************************************
Confusion Matrix:
 [[290  51]
 [ 32 344]]

Test Result:
================================================
Accuracy Score: 86.04%
****************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision   0.902   0.824     0.860      0.863         0.864
recall      0.816   0.907     0.860      0.862         0.860
f1-score    0.857   0.863     0.860      0.860         0.860
support   158.000 150.000     0.860    308.000       308.000
****************************************
Confusion Matrix:
 [[129  29]
 [ 14 136]]
```

Figure 42: Hyperparameter tuning of Logistic Regression

Here the optimized testing accuracy is 86.04%

```
In [80]: test_score1 = accuracy_score(y_test, lr_clf.predict(X_test)) * 100
         train_score1= accuracy_score(y_train, lr_clf.predict(X_train)) * 100

         tuning_results_df = pd.DataFrame(data=[["Tuned Logistic Regression", train_score1, test_score1]],
                                          columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
         tuning_results_df
```

Out[80]:

	Model	Training Accuracy %	Testing Accuracy %
0	Tuned Logistic Regression	88.424	86.039

```
In [36]: #Confusion Matrix, Plot
         matrix= confusion_matrix(y_test,x_pred_mr )
         sns.heatmap(matrix,annot = True, fmt = "d")
```

```
Out[36]: <AxesSubplot:>
```

Figure 43: Training and Testing accuracy and Confusion matrix for Logistic Regression Algorithm

Precision score, F-score, recall, and Model's False Negative rate:

#Precision score

```
In [37]: from sklearn.metrics import precision_score
         precision = precision_score(y_test, x_pred_mr)
         print("Precision: ",precision)
```

```
Precision:  0.8242424242424242
```

Recall

```
In [38]: from sklearn.metrics import recall_score
         recall = recall_score(y_test, x_pred_mr )
         print("Recall is: ",recall)
```

```
Recall is:  0.9066666666666666
```

F-score

```
In [39]: print((2*precision*recall)/(precision+recall))
```

```
0.8634920634920634
```

Model's False Negative rate

```
In [40]: CM =pd.crosstab(y_test, x_pred_mr)
         CM
         TN=CM.iloc[0,0]
         FP=CM.iloc[0,1]
         FN=CM.iloc[1,0]
         TP=CM.iloc[1,1]
         fnr=FN*100/(FN+TP)
         fnr
```

```
Out[40]: 9.333333333333334
```

Figure 44: Precision score, F-score, recall, and Model's False Negative rate for logistic regression

KNN Hyperparameter Tuning:

Hyperparameter Tuning of K-nearest neighbors

In [65]:
```
train_score = []
test_score = []
neighbors = range(1, 30)

for k in neighbors:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    train_score.append(accuracy_score(y_train, model.predict(X_train)))
#     test_score.append(accuracy_score(y_test, model.predict(X_test)))
```

In [66]:
```
plt.figure(figsize=(10, 7))

plt.plot(neighbors, train_score, label="Train score")
# plt.plot(neighbors, test_score, label="Test score")
plt.xticks(np.arange(1, 21, 1))
plt.xlabel("Number of Neighbours ")
plt.ylabel("Score gained by model")
plt.legend()

print(f"Maximum KNN score on the test data: {max(train_score)*100:.2f}%")
```

```
Maximum KNN score on the test data: 100.00%
```

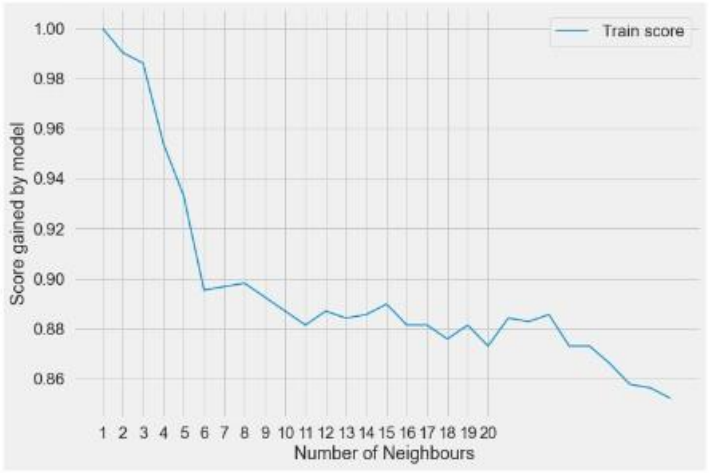


Figure 45: Hyperparameter tuning of Logistic Regression

```
In [47]: knn_clf = KNeighborsClassifier(n_neighbors=27)
         knn_clf.fit(X_train, y_train)
         x_pred_zr = knn_clf.predict(X_test)

         print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
         print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)

Train Result:
================================================
Accuracy Score: 85.77%
************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       0.865   0.851     0.858      0.858         0.858
recall          0.830   0.883     0.858      0.856         0.858
f1-score        0.847   0.867     0.858      0.857         0.858
support   341.000 376.000     0.858    717.000       717.000
****************************************
Confusion Matrix:
 [[283  58]
 [ 44 332]]

Test Result:
================================================
Accuracy Score: 85.06%
****************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       0.878   0.825     0.851      0.852         0.852
recall          0.823   0.880     0.851      0.851         0.851
f1-score        0.850   0.852     0.851      0.851         0.851
support   158.000 150.000     0.851    308.000       308.000
****************************************
Confusion Matrix:
 [[130  28]
 [ 18 132]]
```

Figure 46: Training and Testing accuracy, classification report, and Confusion matrix for KNN

```
In [67]: test_score2 = accuracy_score(y_test, knn_clf.predict(X_test)) * 100
         train_score2 = accuracy_score(y_train, knn_clf.predict(X_train)) * 100

         results_df_2 = pd.DataFrame(data=[["Tuned K-nearest neighbors", train_score2, test_score2]],
                                     columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
         results_df_2
```

Out[67]:

	Model	Training Accuracy %	Testing Accuracy %
0	Tuned K-nearest neighbors	85.774	85.065

```
In [49]: #Confusion Matrix, Plot
         matrix= confusion_matrix(y_test,x_pred_zr )
         sns.heatmap(matrix,annot = True, fmt = "d")

Out[49]: <AxesSubplot:>
```

Figure 47: Training and Testing accuracy and Confusion matrix for KNN

Here the optimized testing accuracy is 85.065%

Precision score, F-score, recall, and Model's False Negative rate:

Precision score

In [68]:
```
from sklearn.metrics import precision_score
precision = precision_score(y_test, x_pred_zr)
print("Precision: ",precision)
```

```
Precision:  0.825
```

Recall

In [69]:
```
from sklearn.metrics import recall_score
recall = recall_score(y_test, x_pred_zr )
print("Recall is: ",recall)
```

```
Recall is:  0.88
```

F-score

In [70]:
```
print((2*precision*recall)/(precision+recall))
```

```
0.8516129032258064
```

Model's False Negative rate

In [71]:
```
CM =pd.crosstab(y_test, x_pred_zr)
CM
TN=CM.iloc[0,0]
FP=CM.iloc[0,1]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
fnr=FN*100/(FN+TP)
fnr
```

Out[71]: 12.0

Figure 48: Precision score, F-score, recall, and Model's False Negative rate for KNN

Hyper Parameter Tuning SVM:

Hypterparameter Tuning of SVM

```
In [72]: svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)

params = {"C":(0.1, 0.5, 1, 2, 5, 10, 20),
          "gamma":(0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1),
          "kernel":('linear', 'poly', 'rbf')}

svm_cv = GridSearchCV(svm_clf, params, n_jobs=-1, cv=5, verbose=1, scoring="accuracy")
svm_cv.fit(X_train, y_train)
x_pred_qr= svm_cv.predict(X_test)
best_params = svm_cv.best_params_
print(f"Best params: {best_params}")

svm_clf = SVC(**best_params)
svm_clf.fit(X_train, y_train)

print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 5 folds for each of 147 candidates, totalling 735 fits
Best params: {'C': 2, 'gamma': 0.25, 'kernel': 'rbf'}
Train Result:
================================================
Accuracy Score: 99.86%
************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       1.000   0.997     0.999      0.999         0.999
recall          0.997   1.000     0.999      0.999         0.999
f1-score        0.999   0.999     0.999      0.999         0.999
support       341.000 376.000     0.999    717.000       717.000
********************************************
Confusion Matrix:
 [[340   1]
 [  0 376]]

Test Result:
================================================
Accuracy Score: 98.05%
****************************************
CLASSIFICATION REPORT:
                    0       1  accuracy  macro avg  weighted avg
precision       0.981   0.980     0.981      0.981         0.981
recall          0.981   0.980     0.981      0.981         0.981
f1-score        0.981   0.980     0.981      0.981         0.981
support       158.000 150.000     0.981    308.000       308.000
****************************************
Confusion Matrix:
 [[155   3]
 [  3 147]]
```

Figure 49: Training and Testing accuracy, classification report, and Confusion matrix for SVM

```
In [73]: test_score3 = accuracy_score(y_test, svm_clf.predict(X_test)) * 100
         train_score3 = accuracy_score(y_train, svm_clf.predict(X_train)) * 100

         results_df_2 = pd.DataFrame(data=[["Tuned Support Vector Machine", train_score3, test_score3]],
                                     columns=['Model', 'Training Accuracy %', 'Testing Accuracy %'])
         results_df_2
```

Out[73]:

	Model	Training Accuracy %	Testing Accuracy %
0	Tuned Support Vector Machine	99.861	98.052

```
In [74]: #Confusion Matrix, Plot
         matrix= confusion_matrix(y_test,x_pred_qr )
         sns.heatmap(matrix,annot = True, fmt = "d")
```

```
Out[74]: <AxesSubplot:>
```

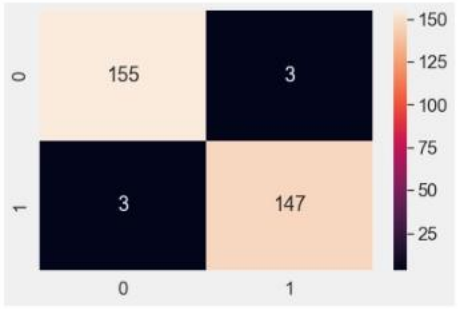
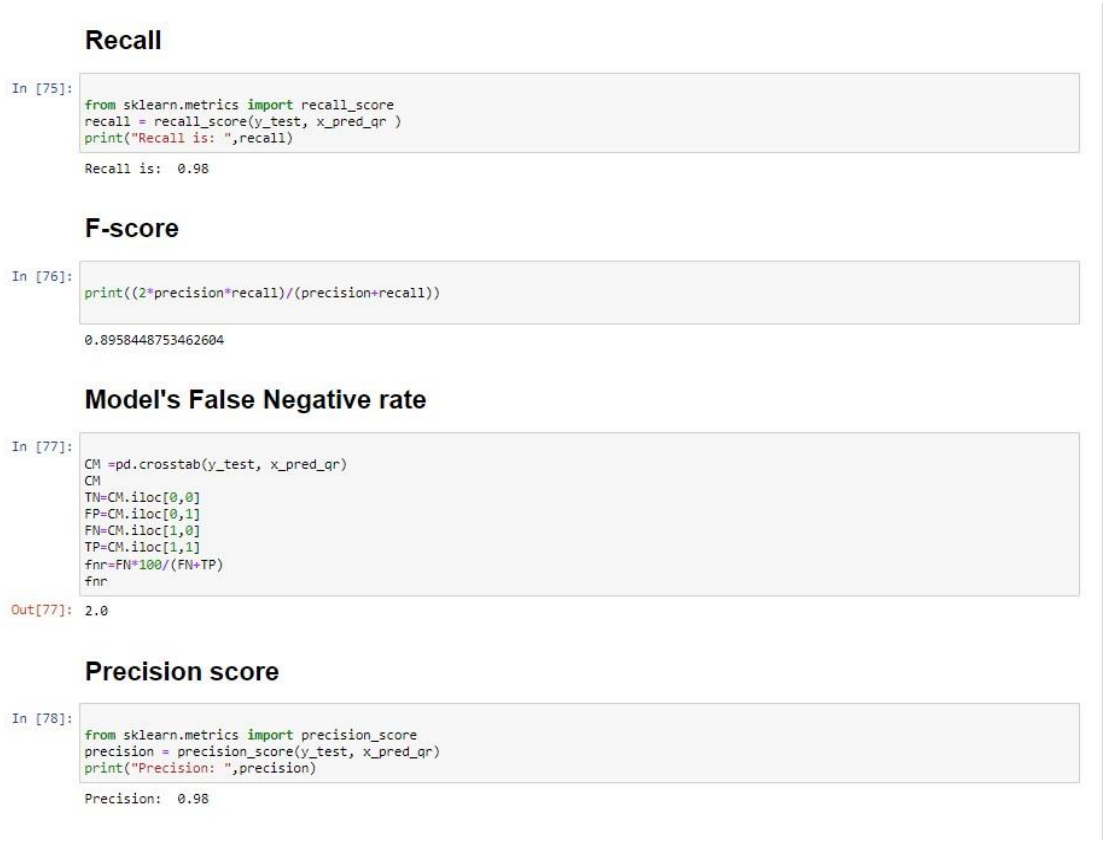


Figure 50: Training, Testing accuracy, and Confusion matrix for SVM

Here the optimized testing accuracy is 98.05%

Precision score, F-score, recall, and Model's False Negative rate:



Recall

In [75]:
```
from sklearn.metrics import recall_score
recall = recall_score(y_test, x_pred_qr )
print("Recall is: ",recall)
```

```
Recall is:  0.98
```

F-score

In [76]:
```
print((2*precision*recall)/(precision+recall))
```

```
0.8958448753462604
```

Model's False Negative rate

In [77]:
```
CM =pd.crosstab(y_test, x_pred_qr)
CM
TN=CM.iloc[0,0]
FP=CM.iloc[0,1]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
fnr=FN*100/(FN+TP)
fnr
```

Out[77]: 2.0

Precision score

In [78]:
```
from sklearn.metrics import precision_score
precision = precision_score(y_test, x_pred_qr)
print("Precision: ",precision)
```

```
Precision:  0.98
```

Figure 51: Precision score, F-score, recall, and Model's False Negative rate for SVM algorithm

Results

Here the bar graph in between the training and testing accuracy of the system is plotted after the optimization which is given below:

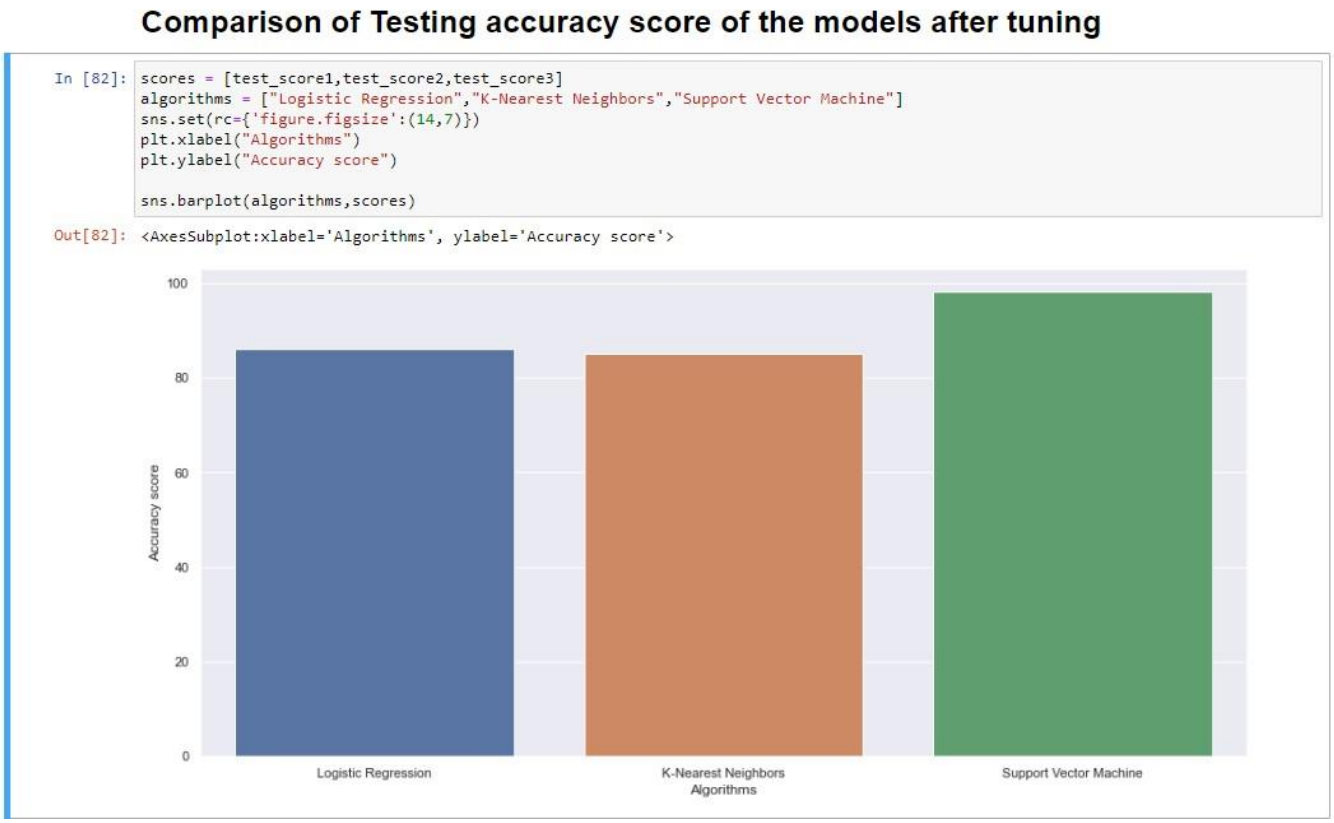


Figure 51: Bar graph between Logistic Regression, KNN, and SVM for testing accuracy after optimization

For the observation of testing accuracy for three algorithms, it is seen that SVM has the highest training accuracy as compared to other algorithms,

Comparison of Training accuracy score of the models after tuning

```
In [81]:
scores = [train_score1,train_score2,train_score3]
algorithms = ["Logistic Regression","K-Nearest Neighbors","Support Vector Machine"]
sns.set(rc={'figure.figsize':(14,7)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)

Out[81]: <AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```

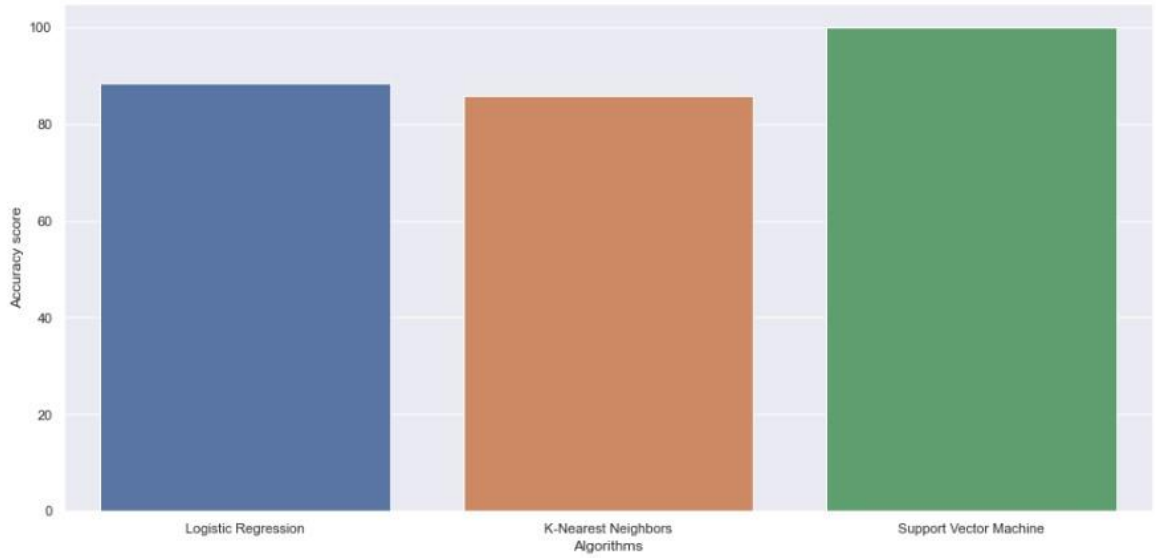


Figure 52: Bar graph between Logistic Regression, KNN, and SVM for training accuracy

For the observation of training accuracy for three algorithms, it is seen that SVM has the highest training accuracy as compared to other algorithms,

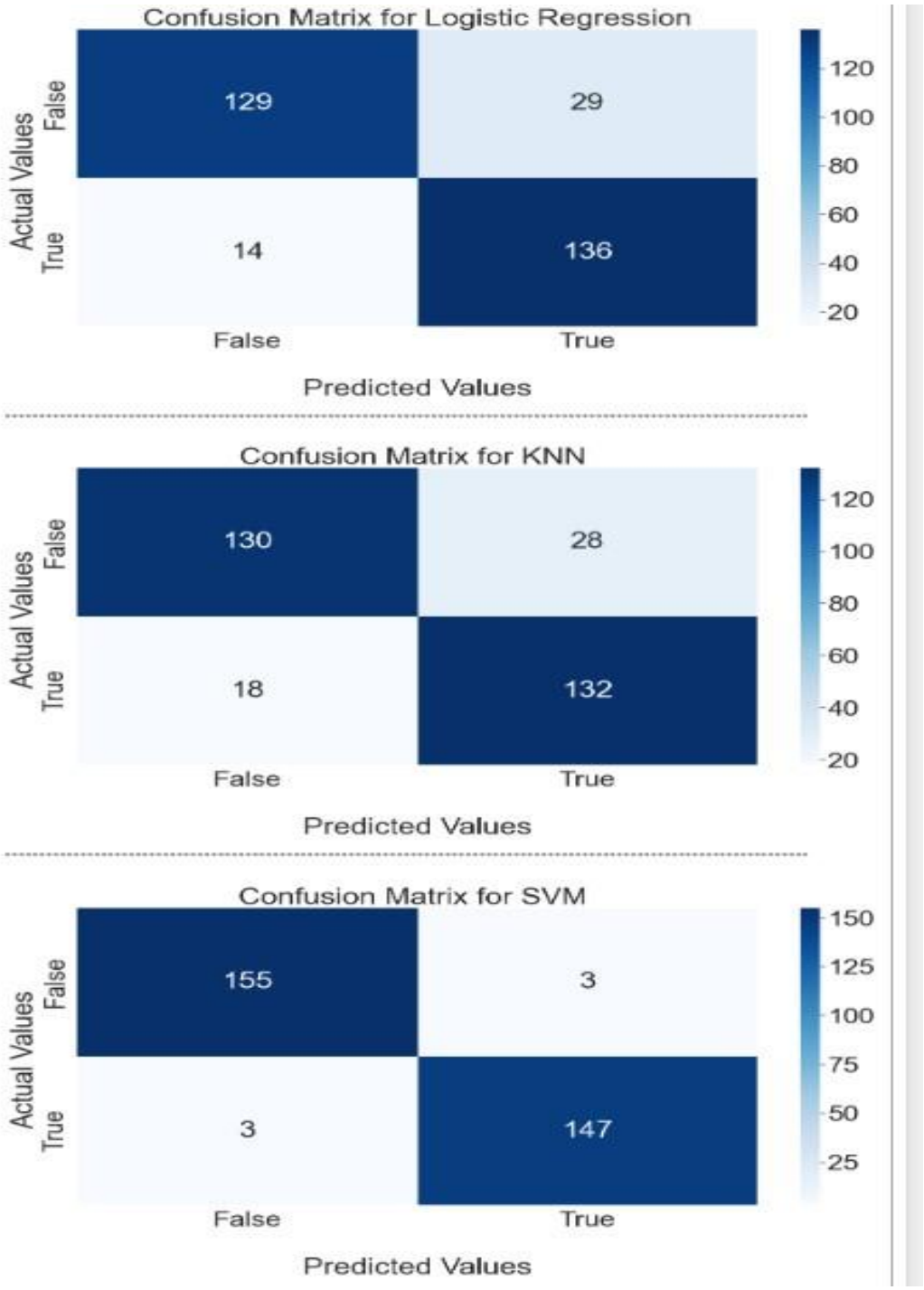


Figure 53: Confusion Matrix in between Logistic Regression, KNN, and SVM for training accuracy

For the confusion Matrix for three algorithms, it is seen that SVM predicts an efficiently greater number of cases of heart disease as compared to other algorithms for the same amount of testing data.

:

Algorithms	Testing Accuracy after optimization	Testing Accuracy Without optimization	Precision score	F-score	Recall	Model's False Negative
SVM	98.05%	86.039%	0.98	0.89	0.98	2.0
KNN	85.065%	83.442%	0.825	0.851	0.88	12
Logistic Regression	86.04%	86.039%	0.824	0.86	0.9066	9.33

Table 54: Showing the value of testing accuracy without optimization, with optimization and Precision score, F-score recall, and Model's False Negative rate

From the above table, it can be concluded that SVM outperformed all other algorithms which have maximum testing accuracy for the prediction of heart disease, similarly, Logistic Regression performed better than KNN and KNN is the worst among all other testing algorithms.

Analysis of AI project Life Cycle in Compliance with AI ethics

Artificial Intelligence (AI) and robotics are those digital technologies that possess a drastic impact on human development in near future. AI ethics is a system of techniques and moral principles which is intended to inform the responsible use and development of the technology of Artificial Intelligence. (Hsieh *et al.*, 2012) Since AI has become very much integral to services and products organizations and governmental bodies are developing policy statements for formally defining the role of artificial intelligence in the seamless development of the human race. The other purpose of AI ethics is also to guide stakeholders during facing an ethical decision about the use of artificial intelligence.

In this project, the algorithms of Machine learning which is the domain of Artificial Intelligence AI has been implemented ethically throughout the lifecycle of the project and the rules and policies have been followed accordingly which is portrayed accordingly in the table below:

	AI Ethics		
Life Cycle of AI Project	**1. Issues raised by Machine Learning**	**2. Living in a digital world**	**3. Metaphysical Issues**
Stage:1 Planning and Data Collection	**Privacy and Data Protection:** No Privacy has been breached.	**Economic Issues:** The project does not affect the disappearance of jobs and other issues.	Since we are in the generation of weak AI there are no issues related to this ethics of AI.
	Safety: No harm to physical integrity.	**Justice and Fairness:** No negative impact on the justice system and unfairness.	
	Reliability: Quality data has been implemented from the popular website.	**Freedom:** No loss in human decision-making and equal access and freedom of information has been ensured.	

	Transparency: Biased data has not been used in the project.	**Broader Societal Issues:** No negative impact on the environment and negative impact on health.	
		Uncertainty Issues: No potential for malicious and criminal issues.	

Table 4: Table for AI Ethics Vs Stage I

	AI Ethics		
Life Cycle of AI Project	**1. Issues raised by Machine Learning**	**2. Living in a digital world**	**3. Metaphysical Issues**
Stage:2 Design and Training of Machine Learning	**Privacy and Data Protection:** No usage of personal data and breaching of privacy.	**Economic Issues:** The project does not affect the disappearance of jobs and other issues.	Since we are in the generation of weak AI there are no issues related to this ethics of AI.
	Safety: No harm to physical integrity.	**Justice and Fairness:** No negative impact on the justice system and unfairness.	
	Reliability: Quality data has been used and accuracy has been maintained for deploying the system.	**Freedom:** No loss in human decision-making and equal access and freedom of	

		information has been ensured.	
	Transparency: There are no gender and racial biases in the data.	**Broader Societal Issues:** No negative impact on the environment and negative impact on health	
		Uncertainty Issues: No potential for malicious and criminal issues.	

Figure 5: Table for AI Ethics Vs Stage II

	AI Ethics		
Life Cycle of AI Project	**1. Issues raised by Machine Learning**	**2. Living in a digital world**	**3. Metaphysical Issues**
Stage:3 Deployment and Maintenance	**Privacy and Data Protection:** Deployment of the model has been done with no breach of personal data.	**Economic Issues:** The project does not affect the disappearance of jobs and other issues.	Since we are in the generation of weak AI there are no issues related to this ethics of AI.
	Safety: No harm to physical integrity.	**Justice and Fairness:** No negative impact on the justice system and unfairness.	
	Reliability: The data is implemented from the reliable database of Kaggle.	**Freedom:** No loss in human decision-making and equal access and freedom of information has been ensured.	
	Transparency: No technologies have been implemented to bias and discriminate against any parts of society.	**Broader Societal Issues:** No negative impact on the environment and negative impact on health	
		Uncertainty Issues: No potential for malicious and criminal issues.	

Table 6: Table for AI Ethics Vs Stage III

Conclusion

Since the overall objective of the project was to carry out the comparison of the accuracy of Machine Learning Algorithms which are KNN, SVM, and Logistic Regression for the prediction of heart disease based upon the medical attributes it is carried out successfully with fewer tests. Through carrying out the project it is concluded the most effective model for the prediction of heart disease among all of these three models is Support Vector Machine Algorithm as it has maximum testing accuracy and it is also not overfitted. Similarly, as compared to KNN, Logistic regression yielded better accuracy and other scores due to which it is found to be a better model than the KNN algorithm. Besides, the system also performed very efficiently even without the aid of retraining.

Recommendations

The recommendations for the project are depicted below:

- Here the dataset can encompass other several attributes rather than the 14 attributes which have been used in the project.
- Besides several other types of data mining techniques could be implemented such as association rules and clustering, Time Series, etc.
- Text mining could also be implemented for the vast amount of unstructured data.

Future work:

This project could be integrated with the web platform for the prediction of heart diseases based upon several types of symptoms in liaison with medical personnel. Here the user could be able to select several types of symptoms and could locate the diseases in terms of probabilistic figures. The improvement in the project can also be done by the implementation of creating an algorithm for suggesting medicine for specific cases of heart disease and feedback from experienced medical professionals could be taken to improve the performance of this algorithm. Besides the usage of live chatting with the doctor could also be done in future projects so that the patient can immediately diagnose the heart disease.

References

Alarsan, F.I. and Younes, M. (2019) 'Analysis and classification of heart diseases using heartbeat features and machine learning algorithms', *Journal of Big Data*, 6(1). Available at: https://doi.org/10.1186/s40537-019-0244-x.

Ali, M.M. *et al.* (2021) 'Heart disease prediction using supervised machine learning algorithms: Performance analysis and comparison', *Computers in Biology and Medicine*, 136(May), p. 104672. Available at: https://doi.org/10.1016/j.compbiomed.2021.104672.

Hsieh, N.C. *et al.* (2012) 'Intelligent postoperative morbidity prediction of heart disease using artificial intelligence techniques', *Journal of Medical Systems*, 36(3), pp. 1809–1820. Available at: https://doi.org/10.1007/s10916-010-9640-7.

Ismaeel, S., Miri, A. and Chourishi, D. (2015) 'Using the Extreme Learning Machine (ELM) technique for heart disease diagnosis', *2015 IEEE Canada International Humanitarian Technology Conference, IHTC 2015*, (1), pp. 1–3. Available at: https://doi.org/10.1109/IHTC.2015.7238043.

Jagtap, A. *et al.* (2019) '09 Ijresm_V2_I2_89', (2).

Kurzweil, R. (1985) 'What Is Artificial Intelligence Anyway', *American Scientist*, 73(3), p. 258.

Lutimath, N.M., Chethan, C. and Pol, B.S. (2019) 'Prediction of heart disease using machine learning', *International Journal of Recent Technology and Engineering*, 8(2 Special Issue 10), pp. 474–477. Available at: https://doi.org/10.35940/ijrte.B1081.0982S1019.

Patel, S.B. (2016) 'H e a r t D i s e a s e P r e d i c t i o n U s i n g M a c h i n e l e a r n i n g a n d D a t a M i n i n g T e c h n i q u e', (March). Available at: https://doi.org/10.090592/IJCSC.2016.018.

Repaka, A.N., Ravikanti, S.D. and Franklin, R.G. (2019) *Design And Implementing Heart Disease Prediction Using Naives Bayesian; Design And Implementing Heart Disease Prediction Using Naives Bayesian*, *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*.

Rizzo, M. and Berneis, K. (2008) 'Clinical utility of different lipid measures for prediction of coronary heart disease in men and women', *Southern Medical Journal*, 101(3), p. 221. Available at: https://doi.org/10.1097/SMJ.0b013e318164de5b.

Saleh Alotaibi, F. (2019) *Implementation of Machine Learning Model to Predict Heart Failure Disease*, *IJACSA) International Journal of Advanced Computer Science and Applications*. Available at: www.ijacsa.thesai.org.

Santhi, P. *et al.* (2021) *A Survey on Heart Attack Prediction Using Machine Learning*, *Turkish Journal of Computer and Mathematics Education*.

Thomas, J. and Princy, R.T. (2016) 'Human heart disease prediction system using data mining techniques', in *Proceedings of IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2016*. Institute of Electrical and Electronics Engineers Inc. Available at:

https://doi.org/10.1109/ICCPCT.2016.7530265.

Wingard, D.L. and Barrett-connor, E. (2003) 'Heart Disease and Diabetes', *Clinical Diabetes*, 21(1), pp. 10–10. Available at: https://doi.org/10.2337/diaclin.21.1.10.

Yekkala, I., Dixit, S. and Jabbar, M.A. (2018) 'Prediction of heart disease using ensemble learning and Particle Swarm Optimization', *Proceedings of the 2017 International Conference On Smart Technology for Smart Nation, SmartTechCon 2017*, pp. 691–698. Available at: https://doi.org/10.1109/SmartTechCon.2017.8358460.