HEALTH ANALYSIS USING ML

A report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

By

VARUN PROHIT (20163)

SCHOOL OF COMPUTING

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA
HIMACHAL PRADESH

DECEMBER 2021

BONAFIDE CERTIFICATE

This is to certify that the project titled HEALTH ANALYSIS USING ML is a bonafide record of the work done by

VARUN PROHIT (20163)

in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in COMPUTER SCIENCE ENGINEERING of the INDIAN INSTITUTE OF INFORMATION TECHNOLOGY UNA, HIMACHAL PRADESH, during the year 2021 - 2022.

under the guidance of

Mr. SAHIL

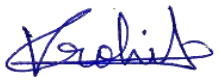
Project viva-voce held on: _____

Examiner (Mr Sahil)

ORIGNALITY / NO PLAGARISM DECLARATION

We certify that this project report is our original report and no part of it is copied from any published reports, papers, books, articles, etc. We certify that all the contents in this report are based on our personal findings and research and we have cited all the relevant sources which have been required in the preparation of this project report, whether they be books, articles, reports, lecture notes, and any other kind of document. We also certify that this report has not previously been submitted partially or as whole for the award of degree in any other university in India and/or abroad.

We hereby declare that, we are fully aware of what constitutes plagiarism and understand that if it is found at a later stage to contain any instance of plagiarism, our degrees may be cancelled.



VARUN PROHIT (20163)

ABSTRACT

Machine Learning and Artificial Intelligence became vital part of everyday life. We have different applications of it. Health sector is also dependent on it for diagnosing of diseases using some pre -defined symptoms and criteria.

There are many Machine Learning Algorithms/ Models to do all these tasks. In this project we are going to compare some of these algorithms and check their accuracy and efficiency for a Heart Disease Analyser. We will also try to identify the reason of suitability of one over the other and also find out the conditions where the given model failed or have very less accuracy.

***Keywords*:** Machine Learning, Heart Disease Analyser, Algorithm, Accuracy and efficiency

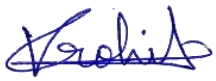
ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

We would like to express our sincere gratitude and heartfelt thanks to Mr. Sahil for their unflinching support and guidance, valuable suggestions and expert advice. Their words of wisdom and expertise in subject matter were of immense help throughout the duration of this project.

We also take the opportunity to thank our Director and all the faculty of School of Computing/Electronics, IIIT Una for helping us by providing necessary knowledge base and resources.

We would also like to thank our parents and friends for their constant support.



VARUN PROHIT (20163)

TABLE OF CONTENTS

LIST OF ACRONYMS

KNN	K Nearest Neighbour
ML	Machine Learning
SVM	Support Random Machine

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Machine Learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is powered by four critical concepts and is **Statistics, Linear Algebra, Probability, and Calculus**. While statistical concepts are the core part of every model, calculus helps us learn and optimize a model.

1.1.1 Goal of ML

The goal of machine learning is to adapt to new data independently and make decisions and recommendations based on thousands of calculations and analyses. It's done by infusing artificial intelligence machines or systems with the ability to learn from the data they're fed. The systems learn, identify patterns, and make decisions with minimal intervention from humans. Ideally, machines increase accuracy and efficiency and remove (or greatly reduce) the possibility of human error.

1.1.2 ML in Real Life

ML has applications in all types of industries, including manufacturing, retail, healthcare and life sciences, travel and hospitality, financial services, and energy, feedstock, and utilities.

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. It is also frequently used for facial recognition within an image.

Table 1.1: Applications of ML in Different Sectors

Sl.No	Sectors	Application
1	Manufacturing	Predictive maintenance and condition monitoring
2	Retail	Upselling and cross-channel marketing
3	Healthcare and life sciences	Disease identification and risk satisfaction
4	Travel and hospitality	Dynamic pricing
5	Financial services	Risk analytics and regulation
6	Energy	Energy demand and supply optimization

1.2 Problem Definition

Health Analysis System using ML: It's aim is to analyze the health status of the person in the society using their health details data like blood pressure, pulse rate, age etc. using different ML algorithms.

1.2.1 Core Objectives

- To understand the behavior of different supervised learning techniques.
- To understand the appropriateness of different supervised learning techniques in different scenarios.
- To compare the efficiency of different supervised learning techniques.

1.3 Pre – Requisite

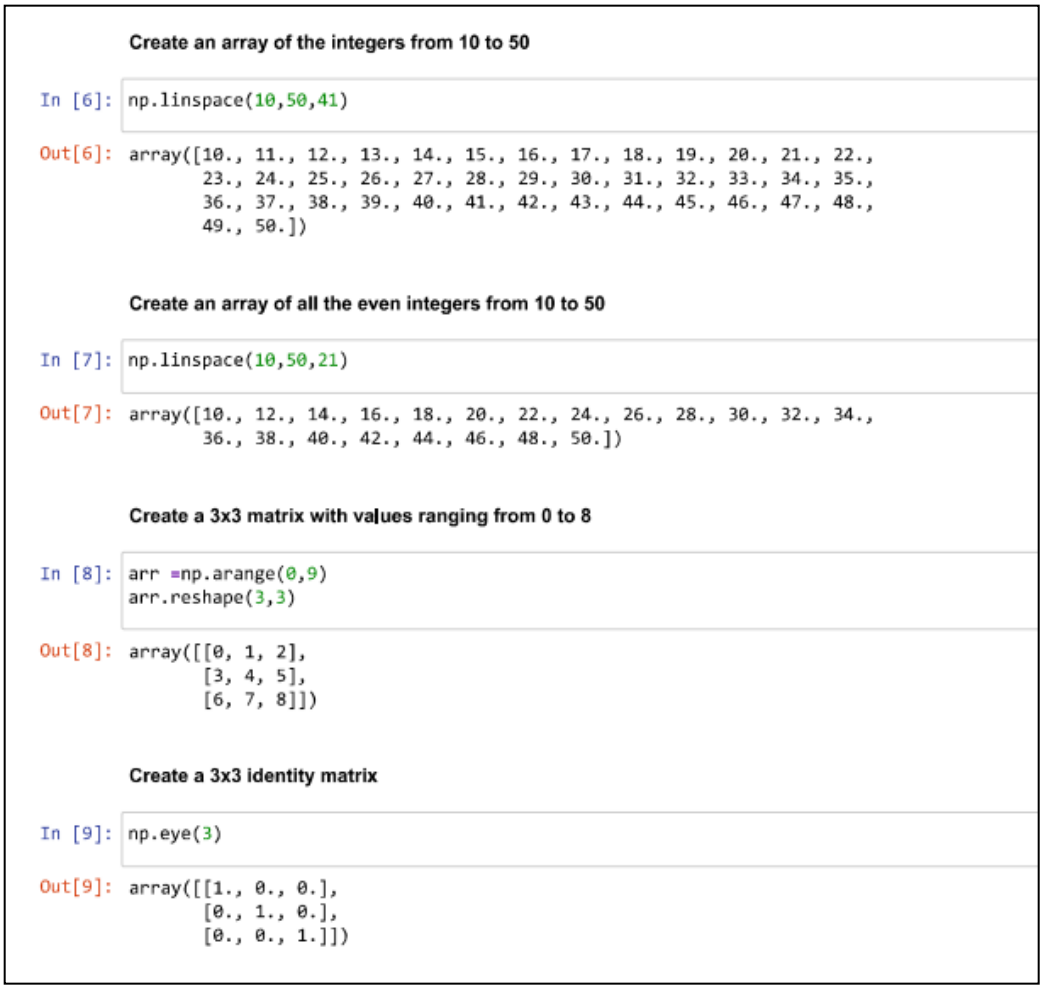
There are basically two pre – requisite of using ML, which are as follows: -

1.3.1 Python and Data Analysis

We must have basic knowledge of Python and its syntax and function declaration of string, list, map, dictionaries and mathematical and logical operations performed on data stored in it.

We have used Python 3.3 in this Practicum and perform all the task only in it.

1.3.1.1 Data Visualization: NumPy: Learnt about built-in methods of NumPy arrays, creation of null matrix, identity matrix etc. and performing logarithmic, exponential and other mathematical operations on these arrays.



Create an array of the integers from 10 to 50

```
In [6]: np.linspace(10,50,41)

Out[6]: array([10., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20., 21., 22.,
               23., 24., 25., 26., 27., 28., 29., 30., 31., 32., 33., 34., 35.,
               36., 37., 38., 39., 40., 41., 42., 43., 44., 45., 46., 47., 48.,
               49., 50.])
```

Create an array of all the even integers from 10 to 50

```
In [7]: np.linspace(10,50,21)

Out[7]: array([10., 12., 14., 16., 18., 20., 22., 24., 26., 28., 30., 32., 34.,
               36., 38., 40., 42., 44., 46., 48., 50.])
```

Create a 3x3 matrix with values ranging from 0 to 8

```
In [8]: arr =np.arange(0,9)
        arr.reshape(3,3)

Out[8]: array([[0, 1, 2],
               [3, 4, 5],
               [6, 7, 8]])
```

Create a 3x3 identity matrix

```
In [9]: np.eye(3)

Out[9]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

Figure 1.1: Operations on NumPy Array

1.3.1.2 Data Visualization: Pandas: In this module we learnt creating series and accessing data stores in data-frames and merging, joining and concatenating to data sets. Multi-indexing, getting missing data and data input output methods features

1.3.1.3 Data Visualization: Matplotlib: In this module we covered creating plot, and their get familiarize with plot properties like: - legends, plot colour, title, size, shape, labels and subplots, DPI settings, saving graph in .jpeg or .png format.

** Now plot (x,y) and (x,z) on the axes. Play around with the linewidth and style**

In [38]:
```
fig,axes = plt.subplots(nrows=1,ncols=2)
axes[0].plot(x,y,color='blue',lw=3,ls='--')
axes[1].plot(x,z,color='red',lw=3,ls='-')
fig
```

Out[38]:

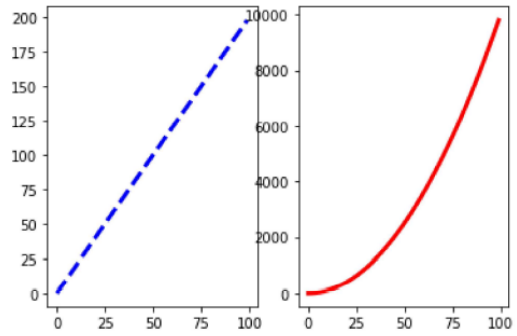


Figure 1.2: Matplotlib Graphs

1.3.1.4 Data Visualization: Seaborn: In this module we covered categorical plot, distribution plot, matrix and regression plots and their styling, spine removing and their implementation on given data set.

In [40]:
```
# CODE HERE
# REPLICATE EXERCISE PLOT IMAGE BELOW
# BE CAREFUL NOT TO OVERWRITE CELL BELOW
# THAT WOULD REMOVE THE EXERCISE PLOT IMAGE!
g = sns.FacetGrid(data=titanic,col='sex')
g.map(plt.hist,'age')
```

Out[40]: <seaborn.axisgrid.FacetGrid at 0x1ed8e3f3f10>

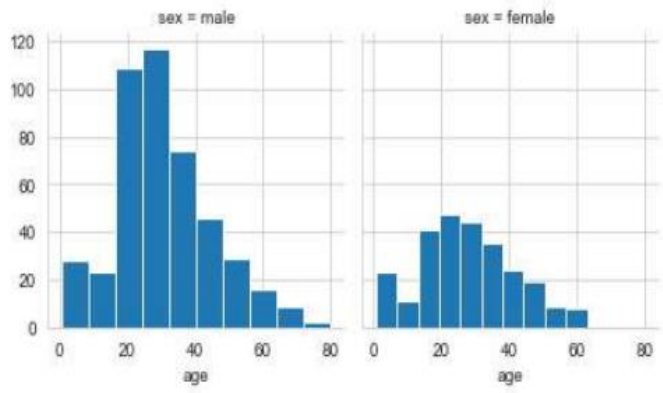


Figure 1.3: Plotting using Seaborn

1.3.1.5 Plotly and Cufflinks: In this module we covered creation of interactive plots (iplot) and geographical plotting.

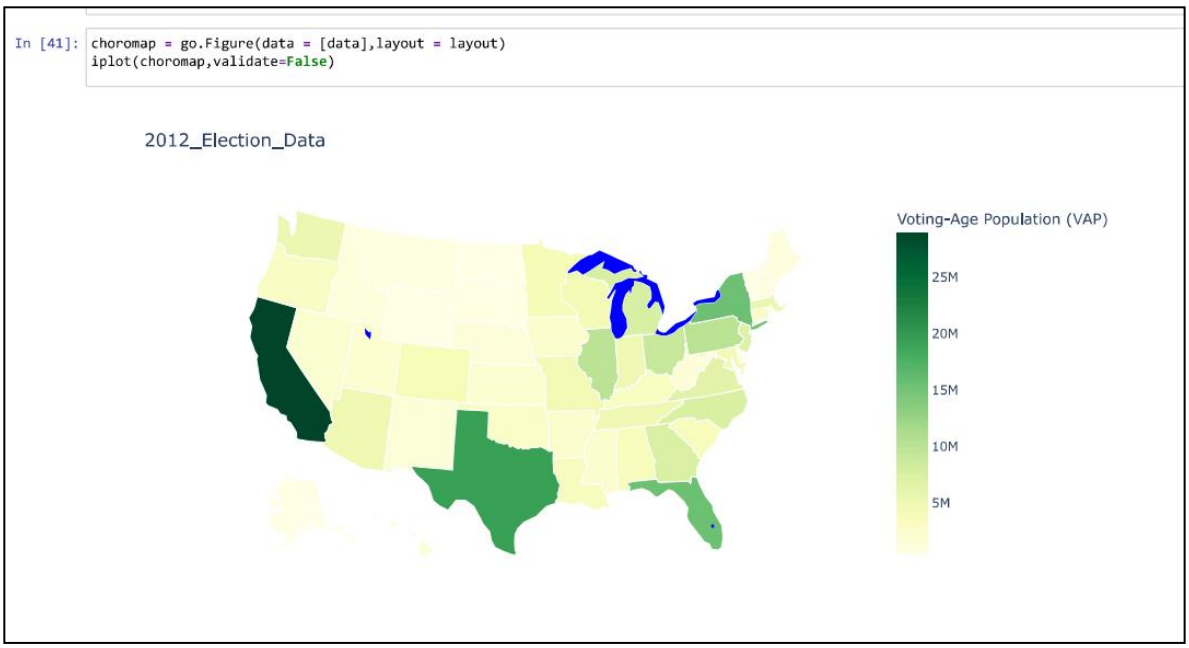


Figure 1.4: Analyzing Voting-Age Population

1.3.2 ML and Its Environment (Anaconda: - Jupyter Notebook)

We have setup local environment using Anaconda and it's Jupyter library and installed required libraries Seaborn, Matplotlib and Scikit -Learn and learn to use the terminal.

1.4 Classification Of ML Models

ML models are broadly classified into three categories: -

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

We covered Supervised Learning Model in this practicum which have different types as follows: -

1.4.1 Linear Regression

Linear regression is a linear approach for modelling the relationship between a scalar respon-

se and one or more explanatory variables (also known as dependent and independent variables). Linear regression analysis is used to predict the value of dependent variable based on the value of independent variable/s.

Given a data set $\{y_i,\ x_{i1},\ \ldots,\ x_{ip}\}_{i=1}^{n}$ of *n* statistical units, a linear regression model assumes that the relationship between the dependent variable *y* and the *p*-vector of regressors **x** is linear. This relationship is modelled through a *disturbance term* or *error variable* ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus, the model takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

where $^{\mathrm{T}}$ denotes the transpose, so that $\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}$ is the inner product between vectors $\mathbf{x}_i$ and $\boldsymbol{\beta}$.

Often these *n* equations are stacked together and written in matrix notation as

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

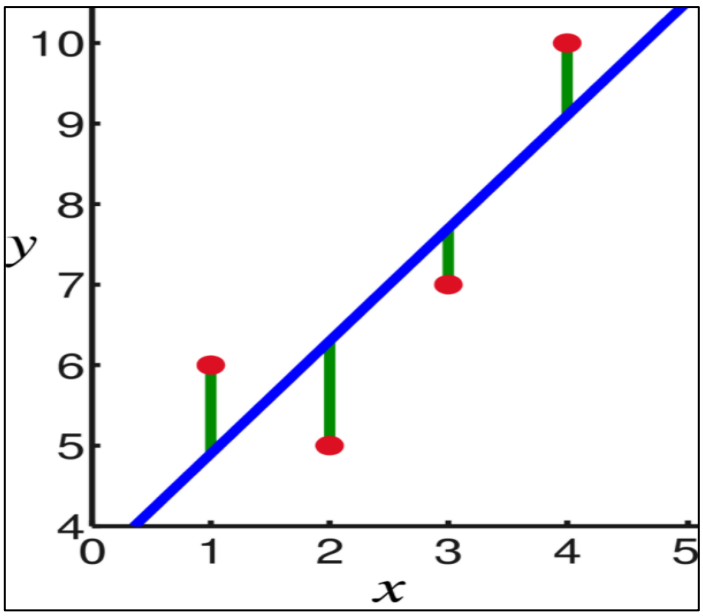


Figure 1.5: Line of Linear Regression

In linear regression (Figure 1.5), the observations (red) are assumed to be the result of random deviations (green) from an underlying relationship (blue) between a dependent variable (*y*) and an independent variable (*x*).

1.4.2 Logistic Regression

It is is a statistical analysis method used to predict a data value based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}} = S_b(\beta_0+\beta_1 x_1+\beta_2 x_2).$$

Where S_b is the sigmoid function with base b.

Generally, sigmoid function is defined as: -

$$f(x) = \frac{1}{1+e^{-x}}$$

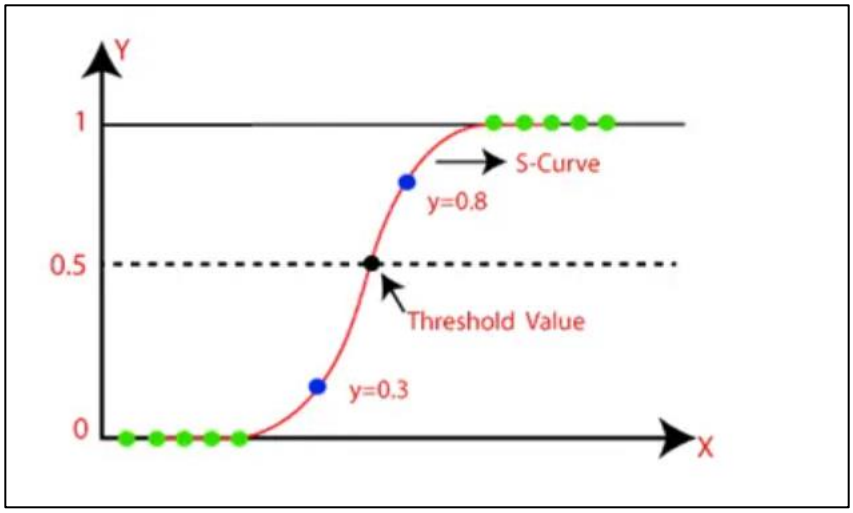


Figure 1.6: Logistic Regression Curve

1.4.3 K Nearest Neighbour

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows.

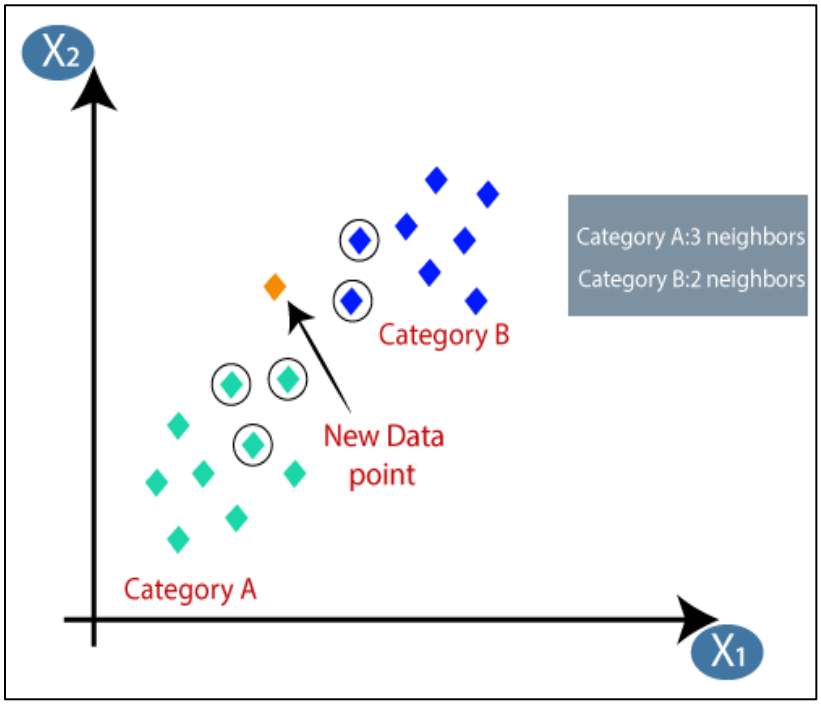


Figure 1.7: KNN Classification

1.4.4 Decision Tree and Random Forest

Decision trees are a type of model used for both classification and regression. Trees answer sequential questions which send us down a certain route of the tree given the answer. The model behaves with “if this than that” conditions ultimately yielding a specific result.

A random forest is simply a collection of decision trees whose results are aggregated into one final result. Their ability to limit overfitting without substantially increasing error due to bias is why they are such powerful models.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

1.4.5 Support Vector Machine

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

The constraint that needs to be satisfied for a training instance to become a support vector. The solution to our problem, i.e., the optimal (maximum-margin) hyperplane remains unchanged if we remove all training instances but the support vectors. That is why they are given the name 'support vectors'.

$$\lambda\|\mathbf{w}\|^2 + \left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right]$$

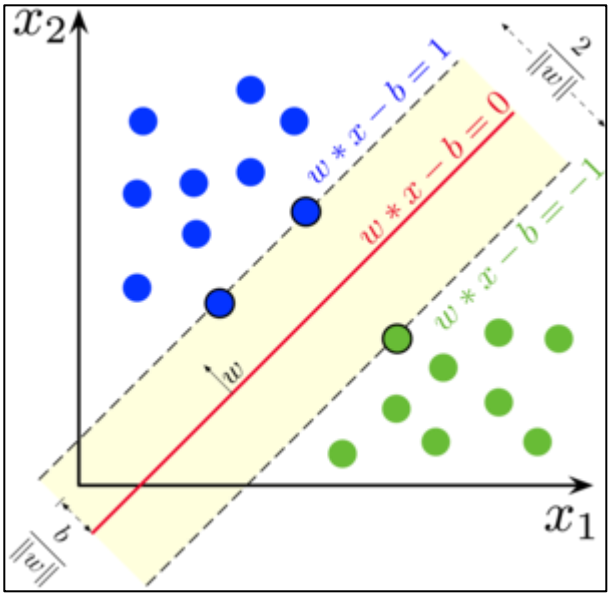


Figure 1.8: SVM Classification

Chapter 2

Implementation

2.1 Data Description

This is a multivariate type of dataset which means providing or involving a variety of separate mathematical or statistical variables, multivariate numerical data analysis. It is composed of 14 attributes which are age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, oldpeak - ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. This database includes 76 attributes, but all published studies relate to the use of a subset of 14 of them. The Cleveland database is the only one used by ML researchers to date. One of the major tasks on this dataset is to predict based on the given attributes of a patient that whether that particular person has heart disease or not and other is the experimental task to diagnose and find out various insights from this dataset which could help in understanding the problem more.

Table 2.1: Data Set

```
In [3]: df.head()
```

Out[3]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

2.2 Data Visualization

Table 2.2: Data Set Column Description

```
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Age             918 non-null    int64
 1   Sex             918 non-null    object
 2   ChestPainType   918 non-null    object
 3   RestingBP       918 non-null    int64
 4   Cholesterol     918 non-null    int64
 5   FastingBS       918 non-null    int64
 6   RestingECG      918 non-null    object
 7   MaxHR           918 non-null    int64
 8   ExerciseAngina  918 non-null    object
 9   Oldpeak         918 non-null    float64
 10  ST_Slope        918 non-null    object
 11  HeartDisease    918 non-null    int64
dtypes: float64(1), int64(6), object(5)
```

From table 2.1, we can see that we can see that five columns namely, 'Sex', 'ChestPainType', 'RestingECG', 'ExcerciseAngina' and 'ST_Slope' are of object type so we have to typecast or make dummy columns for them as our models can't train data on object or string type.

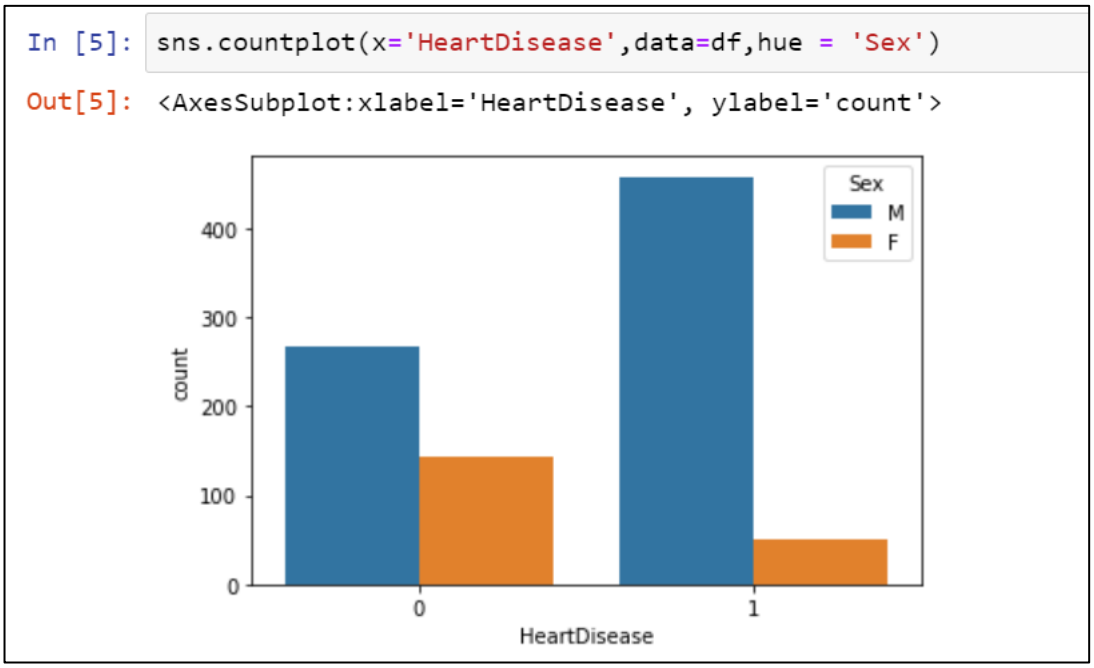


Figure 2.1: Heart Disease Distribution

From the figure above we can easily interpret that 2/3rd of male and 1/4th of female has heart disease.

```
In [6]: sns.countplot(x='ExerciseAngina',data=df,hue = 'Sex')
Out[6]: <AxesSubplot:xlabel='ExerciseAngina', ylabel='count'>
```

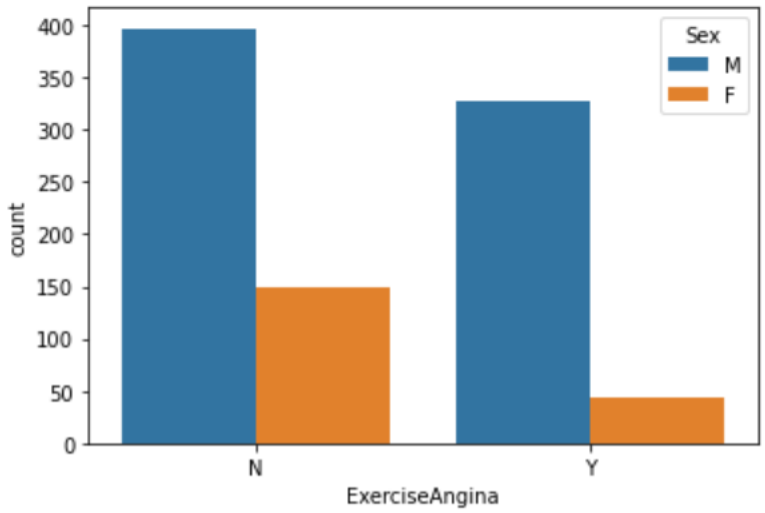


Figure 2.2: Exercise Angina Analysis

From figure 2.3, It is concluded that about 25% of female and 47.2% of male have exercise angina.

```
In [7]: sns.countplot(x='ChestPainType',data=df)
Out[7]: <AxesSubplot:xlabel='ChestPainType', ylabel='count'>
```

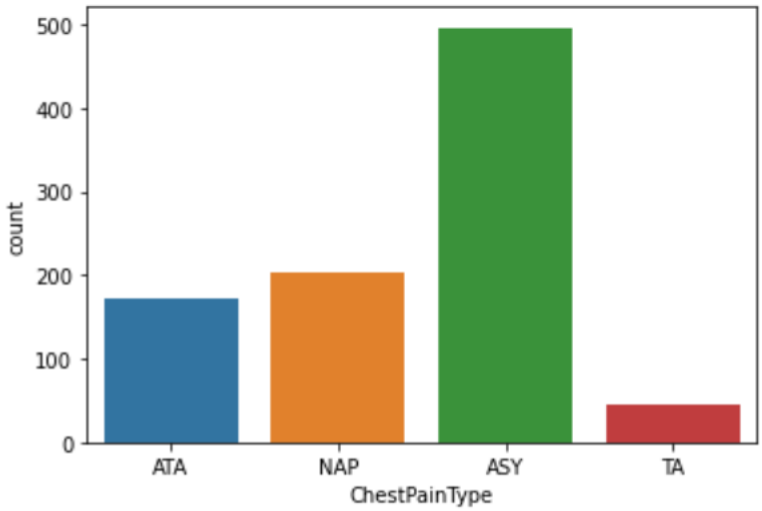


Figure 2.3: Chest Pain Type Distribution

ASY type Chest Pain is commonly found among people and TA type is rarely found.

```
In [8]: sns.countplot(x='ST_Slope',data=df)
Out[8]: <AxesSubplot:xlabel='ST_Slope', ylabel='count'>
```

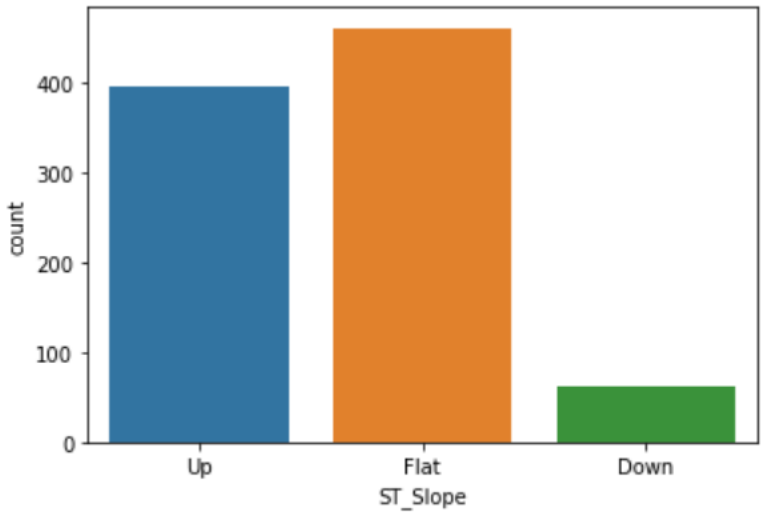


Figure 2.4: ST Slope Distribution

Most of the people have either Up or Flat ST_ Slope only about 5.3% people have Down ST_slope

```
In [9]: sns.countplot(x='RestingECG',data=df)
Out[9]: <AxesSubplot:xlabel='RestingECG', ylabel='count'>
```

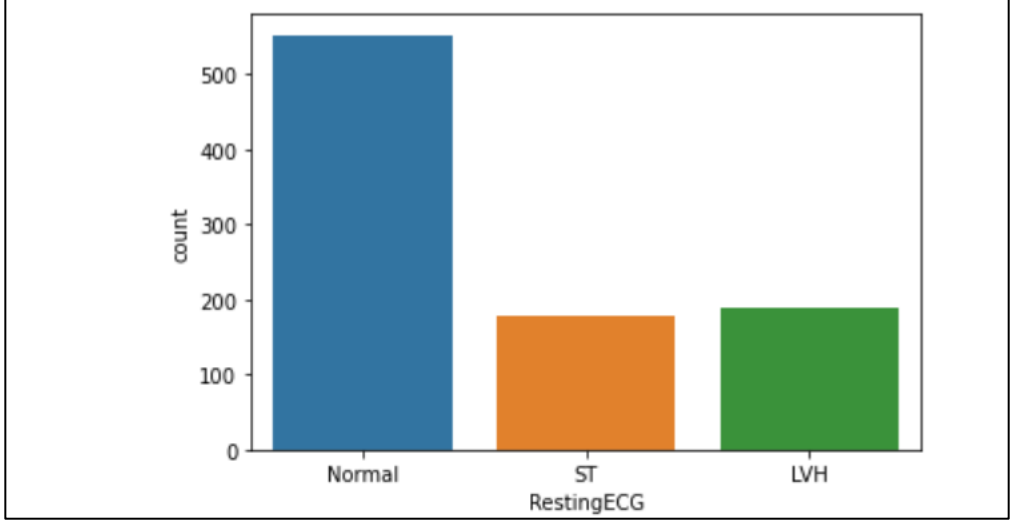


Figure 2.5: Resting ECG Distribution

More than 50% people have normal ECG in resting position while ST and LVH condition is approximately equally distributed among rest of people.

2.3 Data Validation/Cleaning

Before training our model, we have to clean our data by changing all the object type column into integer type by creating their dummy columns and then remove their original columns.

```
#function to create dummy column for Resting_ECG
def ecg(x):
    if x == 'Normal':
        return 0
    elif x=='ST':
        return 1
    else:
        return 2
```

```
df['Restin_ECG'] = df['RestingECG'].apply(ecg)
```

```
#function to create dummy column for STslope
def STslope(x):
    if x == 'Up':
        return 2
    elif x=='Flat':
        return 1
    else:
        return 2
```

```
df['STslope'] = df['ST_Slope'].apply(STslope)
```

```
#function to create dummy column Chest Pain Type
def pain(x):
    if x == 'ATA':
        return 1
    elif x=='NAP':
        return 2
    elif x=='ASY':
        return 3
    else:
        return 4
```

```
df['Chest_paintype'] = df['ChestPainType'].apply(pain)
```

```
#function to create dummy column for Sex
def male(x):
    if x=='M':
        return 1
    else:
        return 0
#function to create dummy column for Exercise Angina
def tr(x):
    if x=='Y':
        return 1
    else:
        return 0
```

```
df['Male']= df['Sex'].apply(male)
df['Ex-Angina'] = df['ExerciseAngina'].apply(tr)
```

```
#Drop all the other columns whose dummy are created
df.drop('ChestPainType',axis=1,inplace=True)
df.drop('Sex',axis=1,inplace=True)
df.drop('RestingECG',axis=1,inplace=True)
df.drop('ST_Slope',axis=1,inplace=True)
df.drop('ExerciseAngina',axis=1,inplace=True)

#Print Data after cleaning is completed
df.head()
```

Table 2.3: Data- Set After Validating

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	HeartDisease	Restin_ECG	STslope	Chest_paintype	Male	Ex-Angina
0	40	140	289	0	172	0.0	0	0	2	1	1	0
1	49	160	180	0	156	1.0	1	0	1	2	0	0
2	37	130	283	0	98	0.0	0	1	2	1	1	0
3	48	138	214	0	108	1.5	1	0	1	3	0	1
4	54	150	195	0	122	0.0	0	0	2	2	1	0

2.4 Model Training

Before training our model, we will split our data set into test and train data set using scikit learn train_test_split.

```
from sklearn.model_selection import train_test_split
```

```
X = df[['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak',
       'Restin_ECG', 'STslope', 'Chest_paintype', 'Male',
       'Ex-Angina']]
y = df['HeartDisease']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

2.4.1 Linear Regression Model

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

```
lm.fit(X_train,y_train)
```

```
LinearRegression()
```

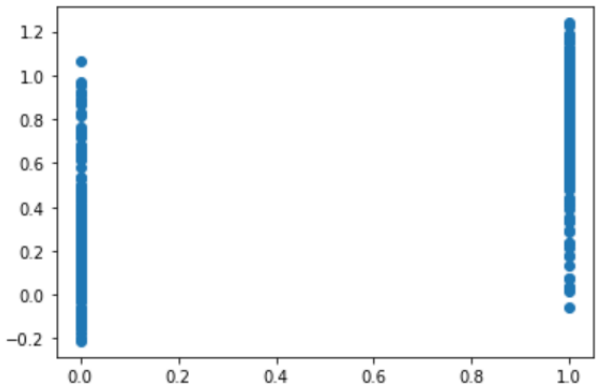
```
print(lm.intercept_)
```

```
0.8783304261790501
```

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

```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x231615502e0>
```



```
from sklearn import metrics
```

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 0.2603202913370167
MSE: 0.1267592682015667
RMSE: 0.3560326785585372
```

Linear Regression Model is not suitable for our data set as in linear regression we have a linear relationship between dependent and independent variable bur here we have categorical distribution we can also be seen in scatter plot.

2.4.2 Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
```

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

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

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.85      0.82      0.83       170
           1       0.85      0.87      0.86       198

    accuracy                           0.85       368
   macro avg       0.85      0.85      0.85       368
weighted avg       0.85      0.85      0.85       368
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[139  31]
 [ 25 173]]
```

This model provides accuracy of 85% which is good.

2.4.3 KNN Model

```
# making copy of our data set
df1 = df
df1['Heart-disease'] = df1['HeartDisease']
df1.drop('HeartDisease',axis=1,inplace =True)
```

```
#scaling the data
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(df1.drop('Heart-disease',axis=1))
```

```
StandardScaler()
```

```
scaled_features = scaler.transform(df1.drop('Heart-disease',axis=1))
```

```
df_feat = pd.DataFrame(scaled_features,columns=df1.columns[:-1])
df_feat.head()
```

Table 2.4: Data After Scaling

	Age	RestingBP	Cholesterol	FastingBS	MaxHR	Oldpeak	Restin_ECG	STslope	Chest_paintype	Male	Ex-Angina
0	-1.433140	0.410909	0.825070	-0.551341	1.382928	-0.832432	-0.749180	1.002181	-1.705573	0.515952	-0.823556
1	-0.478484	1.491752	-0.171961	-0.551341	0.754157	0.105664	-0.749180	-0.997824	-0.530992	-1.938163	-0.823556
2	-1.751359	-0.129513	0.770188	-0.551341	-1.525138	-0.832432	0.492241	1.002181	-1.705573	0.515952	-0.823556
3	-0.584556	0.302825	0.139040	-0.551341	-1.132156	0.574711	-0.749180	-0.997824	0.643588	-1.938163	1.214246
4	0.051881	0.951331	-0.034755	-0.551341	-0.581981	-0.832432	-0.749180	1.002181	-0.530992	0.515952	-0.823556

```
X_train, X_test, y_train, y_test = train_test_split(scaled_features,df['Heart-disease'],
                                                    test_size=0.4,random_state=101)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=1)
```

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=1)
```

```
pred = knn.predict(X_test)
```

```
print(confusion_matrix(y_test,pred))
```

```
[[132  38]
 [ 35 163]]
```

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.78      0.78       170
           1       0.81      0.82      0.82       198

    accuracy                           0.80       368
   macro avg       0.80      0.80      0.80       368
weighted avg       0.80      0.80      0.80       368
```

```
# obtaining suitable value of k
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))
```

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

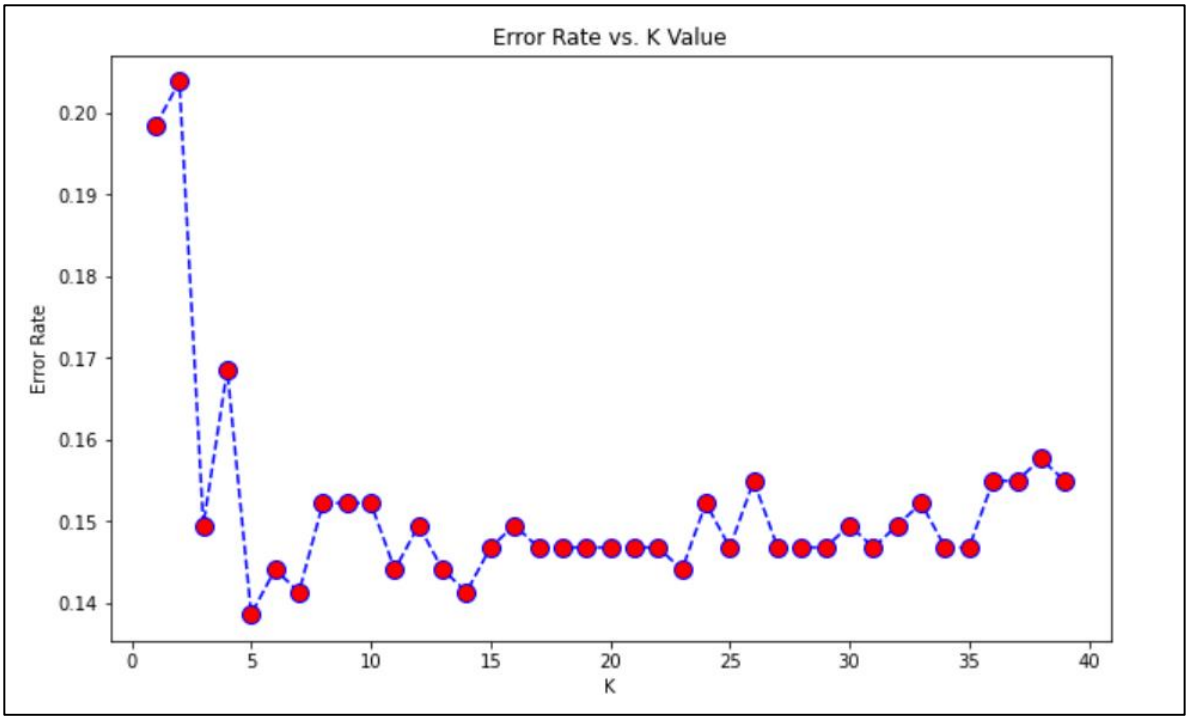


Figure 2.6: Plot of Error Rate vs K

```
# NOW WITH K=5
knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH K=5')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH K=5


[[140  30]
 [ 21 177]]


              precision    recall  f1-score   support

           0       0.87      0.82      0.85       170
           1       0.86      0.89      0.87       198

    accuracy                           0.86       368
   macro avg       0.86      0.86      0.86       368
weighted avg       0.86      0.86      0.86       368
```

In this model we get accuracy of 86% which is 1% more than Logistic Regression Model.

2.4.4 Decision Tree and Random Forests Model

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

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

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       170
           1       0.81      0.81      0.81       198

    accuracy                           0.80       368
   macro avg       0.80      0.79      0.79       368
weighted avg       0.80      0.80      0.80       368
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[132  38]
 [ 37 161]]
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```
rfc_pred = rfc.predict(X_test)
```

```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[138  32]
 [ 22 176]]
```

```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.81      0.84       170
           1       0.85      0.89      0.87       198

    accuracy                           0.85       368
   macro avg       0.85      0.85      0.85       368
weighted avg       0.85      0.85      0.85       368
```

2.4.5 SVM Model

```
from sklearn.svm import SVC
```

```
model = SVC()
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.40,random_state=101)
```

```
model.fit(X_train,y_train)
```

```
SVC()
```

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

```
print(confusion_matrix(y_test,predictions))
```

```
[[115  55]
 [ 60 138]]
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.66      0.68      0.67       170
           1       0.72      0.70      0.71       198

    accuracy                           0.69       368
   macro avg       0.69      0.69      0.69       368
weighted avg       0.69      0.69      0.69       368
```

```
from sklearn.model_selection import GridSearchCV
```

```
param_grid = {'C' : [100000,10000000,100000000],'gamma': [0.000001,0.0000001,0.00000001]}
```

```
gs = GridSearchCV(SVC(),param_grid,refit=True,verbose=3)
```

```
gs.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END .........................C=100000, gamma=1e-06; total time=   0.1s
[CV 2/5] END .........................C=100000, gamma=1e-06; total time=   0.1s
[CV 3/5] END .........................C=100000, gamma=1e-06; total time=   0.0s
[CV 4/5] END .........................C=100000, gamma=1e-06; total time=   0.0s
[CV 5/5] END .........................C=100000, gamma=1e-06; total time=   0.1s
[CV 1/5] END .........................C=100000, gamma=1e-07; total time=   0.0s
[CV 2/5] END .........................C=100000, gamma=1e-07; total time=   0.0s
[CV 3/5] END .........................C=100000, gamma=1e-07; total time=   0.0s
[CV 4/5] END .........................C=100000, gamma=1e-07; total time=   0.0s
[CV 5/5] END .........................C=100000, gamma=1e-07; total time=   0.0s
[CV 1/5] END .........................C=100000, gamma=1e-08; total time=   0.0s
[CV 2/5] END .........................C=100000, gamma=1e-08; total time=   0.0s
[CV 3/5] END .........................C=100000, gamma=1e-08; total time=   0.0s
[CV 4/5] END .........................C=100000, gamma=1e-08; total time=   0.0s
[CV 5/5] END .........................C=100000, gamma=1e-08; total time=   0.0s
[CV 1/5] END ........................C=10000000, gamma=1e-06; total time=   5.9s
[CV 2/5] END ........................C=10000000, gamma=1e-06; total time=   4.2s
[CV 3/5] END ........................C=10000000, gamma=1e-06; total time=   3.3s
[CV 4/5] END ........................C=10000000, gamma=1e-06; total time=   5.6s
[CV 5/5] END ........................C=10000000, gamma=1e-06; total time=   6.2s
[CV 1/5] END ........................C=10000000, gamma=1e-07; total time=   0.4s
[CV 2/5] END ........................C=10000000, gamma=1e-07; total time=   0.7s
[CV 3/5] END ........................C=10000000, gamma=1e-07; total time=   0.3s
[CV 4/5] END ........................C=10000000, gamma=1e-07; total time=   0.3s
[CV 5/5] END ........................C=10000000, gamma=1e-07; total time=   0.5s
[CV 1/5] END ........................C=10000000, gamma=1e-08; total time=   0.0s
[CV 2/5] END ........................C=10000000, gamma=1e-08; total time=   0.0s
[CV 3/5] END ........................C=10000000, gamma=1e-08; total time=   0.0s
[CV 4/5] END ........................C=10000000, gamma=1e-08; total time=   0.0s
[CV 5/5] END ........................C=10000000, gamma=1e-08; total time=   0.0s
[CV 1/5] END .......................C=100000000, gamma=1e-06; total time=   8.1s
[CV 2/5] END .......................C=100000000, gamma=1e-06; total time=   4.0s
[CV 3/5] END .......................C=100000000, gamma=1e-06; total time=  15.6s
[CV 4/5] END .......................C=100000000, gamma=1e-06; total time=  24.7s
[CV 5/5] END .......................C=100000000, gamma=1e-06; total time=  12.0s
[CV 1/5] END .......................C=100000000, gamma=1e-07; total time=   0.3s
[CV 2/5] END .......................C=100000000, gamma=1e-07; total time=   0.5s
[CV 3/5] END .......................C=100000000, gamma=1e-07; total time=   1.3s
[CV 4/5] END .......................C=100000000, gamma=1e-07; total time=   0.6s
[CV 5/5] END .......................C=100000000, gamma=1e-07; total time=   0.4s
[CV 1/5] END .......................C=100000000, gamma=1e-08; total time=   0.0s
[CV 2/5] END .......................C=100000000, gamma=1e-08; total time=   0.0s
[CV 3/5] END .......................C=100000000, gamma=1e-08; total time=   0.0s
[CV 4/5] END .......................C=100000000, gamma=1e-08; total time=   0.0s
[CV 5/5] END .......................C=100000000, gamma=1e-08; total time=   0.0s

GridSearchCV(estimator=SVC(),
             param_grid={'C': [100000, 10000000, 100000000],
                         'gamma': [1e-06, 1e-07, 1e-08]},
             verbose=3)
```

```
gs.best_params_
```

```
{'C': 10000000, 'gamma': 1e-07}
```

```
gs.best_estimator_
```

```
SVC(C=10000000, gamma=1e-07)
```

```
gs_predictions = gs.predict(X_test)
```

```
print(confusion_matrix(y_test,gs_predictions))
```

```
[[141  29]
 [ 29 169]]
```

```
print(classification_report(y_test,gs_predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.83      0.83       170
           1       0.85      0.85      0.85       198

    accuracy                           0.84       368
   macro avg       0.84      0.84      0.84       368
weighted avg       0.84      0.84      0.84       368
```

Here we are getting 84% accuracy.

2.5 Conclusion

Logistic Regression, KNN, Decision Tree and SVM have accuracy of 86%, 85%, 86% and 84% respectively. We are not getting accuracy above 90% for any model also for SVM which is most versatile because we have some points which are too close to classify accurately as can be seen in fig 2.8.

```
sns.pairplot(df,hue='Heart-disease',palette='Dark2')
```

```
<seaborn.axisgrid.PairGrid at 0x2315b63cac0>
```

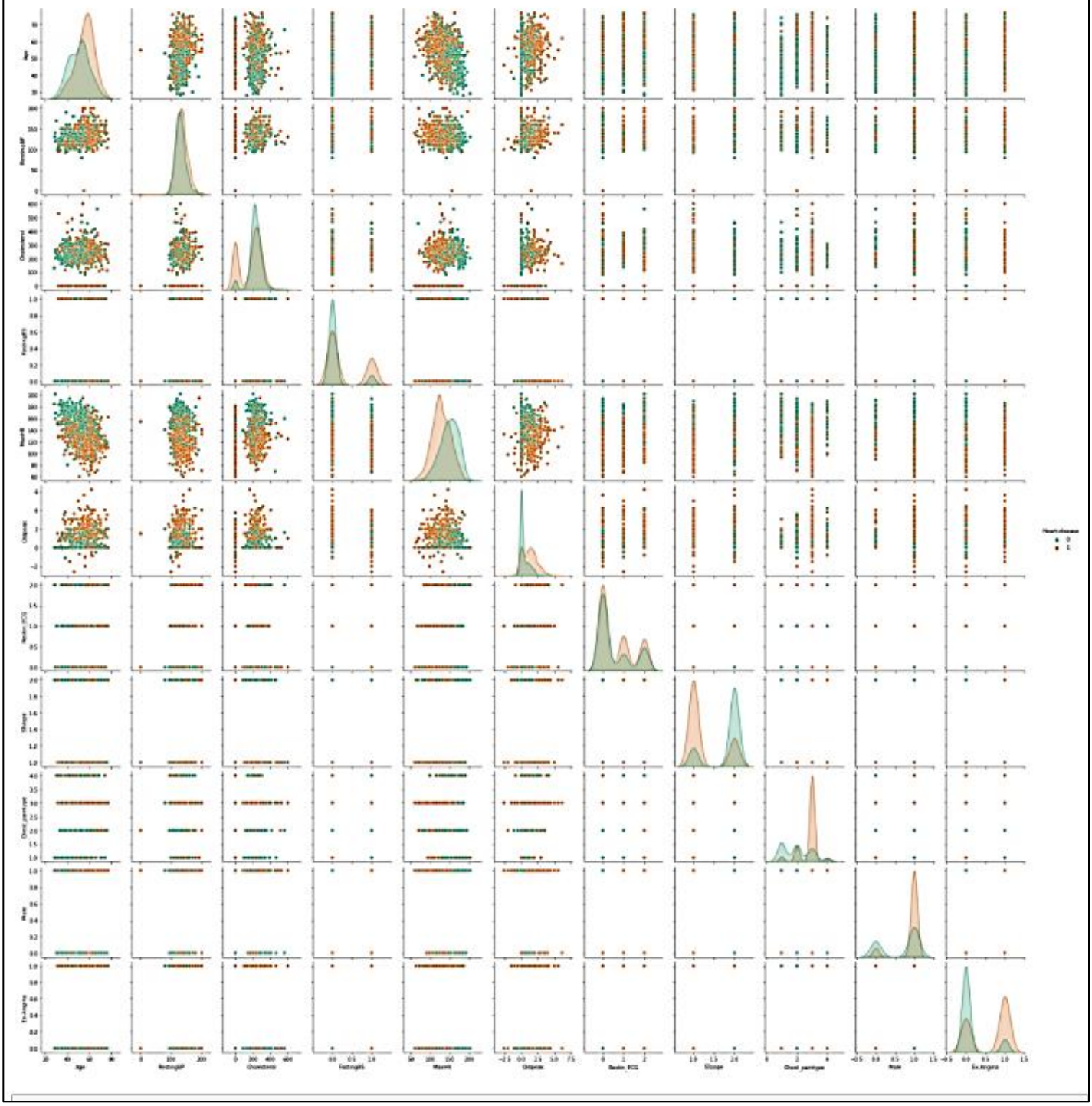


Figure 2.7: Plots between all the Parameters

References

[1] Python for Data Science and Machine Learning Bootcamp by Josh Portilla.

Link: https://www.udemy.com/share/101WaU/