EMPERICAL ANALYSIS AND PERFORMANCE EVALUATION OF DIFFERENT LUNG CANCER CLASSIFICATION TECNIQUE USING MACHINE LEARNING

(BTCS 603-18)

Submitted in partial fulfilment of the Requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE

Sharpening Minds, Brightening Tomorrow.
Gulzar Group of Institutes

UNDER THE GUIDANCE OF Dr. Sita Rani

BY

SANJOLI (1906144)

SAKSHI JHA (1906138)

SUKRITI SOOD (1906151)

SHIVAM SINHA (1906144)

KAPURTHALA

ਆਈ.ਕੇ. ਗੁਜਰਾਲ ਪੰਜਾਬ ਟੈਕਨੀਕਲ ਯੂਨੀਵਰਸਿਟੀ, ਜਲੰਧਰ

I.K. GUJRAL PUNJAB TECHNICAL UNIVERSITY, JALANDHAR

CANDIDATE'S DECLARATION AND CERTIFICATE

We hereby certify that the work, which is being presented in this report entitled, **Emperical Analysis and performance evaluation of different lung cancer classification technique using machine learning** , in partial fulfillment of the requirements for the degree of **B.TECH** submitted in the **Computer Science and Engineering**, Gulzar Group of Institutes, Khanna, Punjab; by **Sakshi(1906138), Sukriti (1906151), Sanjoli(1906141), Shivam sinha(1906144)** is the authentic record of our own work carried out under the supervision of **Dr.Sita Rani,Computer science Engineering**, Gulzar Group of Institutes, Khanna, Punjab.

We further declare that the matter embodied in this report has not been submitted by us for the award of any other degree.

Candidate(s) Signature

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Signature of HOD **Signature of Supervisor**

HOD Name **Supervisor Name**

Date:

ACKNOWLEDGEMENT

It is our pleasure to acknowledge the contributions of all who have helped us and supported us during this Project report.

First, we thank God for helping us in one way or another and providing strength and endurance to us. We wish to express my sincere gratitude and indebtedness to our supervisor, Supervisor Name, department name, Gulzar Group of Institutes, Khanna, Punjab; for her/his intuitive and meticulous guidance and perpetual inspiration in completion of this report. In spite of his/her busy schedule, he/she rendered help whenever needed, giving useful suggestions and holding informal discussions. Her invaluable guidance and support throughout this work cannot be written down in few words. We also thank her for providing facilities for my work in the department name.

We are also humbly obliged by the support of our group members and friends for their love and caring attitude. The sentimental support they rendered to us is invaluable and everlasting. They have helped us through thick and thin and enabled us to complete the work with joy and vigor. We thank the group members for entrusting in each other and following directions, without them this report would never have been possible.

We are also thankful to our parents, elders and all family members for their blessing, motivation and inspiration throughout our work and bearing with us even during stress and bad temper. They have always provided us a high moral support and contributed in all possible ways in completion of this Capstone report.

ABSTRACT

Lung cancer is one of the leading causes of mortality in every country, affecting both men and women. Lung cancer has a low prognosis, resulting in a high death rate. The computing sector is fully automating it, and the medical industry is also automating itself with the aid of image recognition and data analytics. Lung cancer is one of the most common diseases for human beings everywhere throughout the world. Lung cancer is a disease which arises due to growth of unwanted tissues in the lung and this growth which spreads beyond the lung are named as metastasis which spreads into other parts of the body.

The objective of our project is to inspect accuracy ratio of two classifiers which is Support Vector Machine (SVM), and K Nearest Neighbour (KNN) on common platform that classify lung cancer in early stage so that many lives can be saving. The experimental results show that KNN gives the best result Than SVM . This report discusses the Implementation details of our project.

We have done data Preprocessing ,data cleaning and implements machine leaning algorithm for prediction of lung cancer at early stages through their symptoms.we have used both classification algorithm to find or predict the accuracy ratio.

TABLE OF CONTENTS

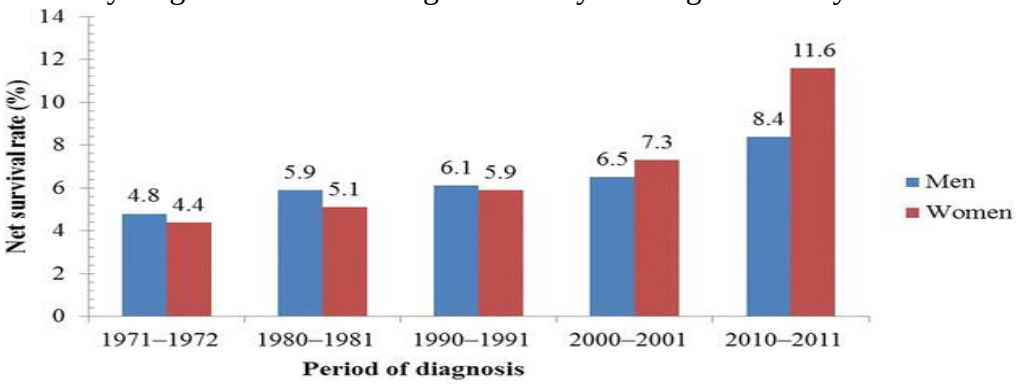
CHAPTER 1

INTRODUCTION

Cancers exist in several organs, and simultaneously, and different types of cancer occur in various organs of the body. The illness may even go unnoticed for long periods of time. According to WHO reports, cancer may be prevented if it is detected early enough.

Lung cancer is one of the most common diseases for human beings everywhere throughout the world. “Lung cancer is a disease which arises due to growth of unwanted tissues in the lung and this growth which spreads beyond the lung are named as metastasis which spreads into other parts of the body”.

It is one of the leading causes of mortality in every country, affecting both men and women. Lung cancer has a low prognosis , resulting in a high death rate. Lung disease which affects the parts of lungs and cause infections such as tuberculosis ,pneumonia and other breathing problems such as asthma. Lung cancer constitutes 12.8% of all cancer types worldwide, also it constitutes 17.8% of the cancer deaths and it increases by 0.5% every year globally worldwide. The cause of cancer in males represents 38.6%, and in females it represents 5.2%. However, it is suggested that 15% of lung cancer patients live 5 years or more after the diagnosis, together with this early diagnosis and used drugs when they are diagnosed early.



The vast majority (85%) of cases of lung cancer are due to long-term tobacco smoking and about 10–15% of the cases occur in people who have never smoked.

Lung cancer occurs when a malignant (cancerous) tumor grows inside the lungs, in structures such as the bronchi (small tubes that connect the windpipe to the inner surfaces of the lungs where gas transfer takes place). Like many other types of cancer, lung cancer is capable of spreading (metastasizing) to other parts of the body. In this case, cancer beginning in the lungs most commonly spreads to the brain, bones, adrenal glands and liver, via any of three mechanisms: direct extension, via the blood vessels, or via the lymph system. Direct extension occurs when a tumor grows rapidly in size such that it begins to touch an adjacent organ or structure, and then begins to penetrate itself into that adjacent organ or structure. Tumor cells are also able to get into the blood and lymph circulatory systems and travel, one by one, to distant structures. Lung cancer is now the most prevalent form of cancer affecting Americans with an estimated 222,500 new cases every year, according to the American Cancer Society (ACS, 2010). Beyond being the most common form of cancer, lung cancer is also often difficult to treat. As a result, lung cancer is the most deadly cancer

with roughly 160,000 Americans dying from it every year. This is about 30% of all cancer deaths! (ACS, 2010).

Although lung cancer is difficult to treat and cure, it is for the most part preventable. Lifestyle choices can be made which can almost eliminate your risk for getting the disease. Your decision to stop smoking and to eat a healthy diet featuring plenty of fresh fruits and vegetables can greatly decrease your risk.

TYPES OF LUNGS CANCER

There are 2 main types of lung cancer and they are treated very differently.

1. Non-small cell lung cancer (NSCLC)

About 80% to 85% of lung cancers are NSCLC. This type of lung cancer occurs mainly in people who currently smoke or formerly smoked, but it is also the most common type of lung cancer seen in people who don't smoke. It is more common in women than in men, and it is more likely to occur in younger people than other types of lung8 cancer.

2. Small cell lung cancer (SCLC)

About 10% to 15% of all lung cancers are SCLC and it is sometimes called oat cell cancer. This type of lung cancer tends to grow and spread faster than NSCLC. About 70% of people with SCLC will have cancer that has already spread at the time they are diagnosed. Since this cancer grows quickly, it tends to respond well to chemotherapy and radiation therapy.

Several techniques are available for lung cancer diagnosis but those techniques are more expensive, time consuming and having less capability for detecting the lung cancer. Hence, a new prediction method is essential to predict the lung cancer in its early stage.

CHAPTER 2

RELATED WORK

Machine learning involves several algorithms such as k-Nearest Neighbours (KNN), support vector machine (SVM), Naive Bayes (NBs), classification tree (C4.5), gradient boosting machines (GBM), etc. While each of these algorithms processes data differently, in this section, a few recently proposed machine learning candidates in the area of malignant growth finding are reviewed chronologically.

Chen et al. (2013) presented a fuzzy system using KNN (FKNN) for Parkinson's disease (PD) diagnose. Besides, they used the principal component analysis to find the most discriminative features on which the optimal FKNN model was built. They compared their system with the SVM algorithm and found that their proposed method performed better. The best classification accuracy of their FKNN reached to 96.07%.

Odajima & Pawlovsky (2014) declared that the precision of the KNN method changes with the number of neighbours and with the level of information utilized for classification. Meanwhile, they showed details about the variation of the maximum and the minimum values of the accuracy with the classification set sizes and the number of neighbours.
Lynch et al. (2017) applied some supervised learning classification techniques such as linear regression, decision trees, GBM, SVM, and a custom ensemble to the SEER database to order lung cancer patients regarding survival. The outcomes demonstrated that among the five individual models used, the most precise was GBM with a root mean square error (RMSE) value of 15.32.
Septiani et al. (2017) compared the performances of C4.5, NBs, and KNN classification algorithms to detect breast cancer diagnosis on 670 data, each with 9 attributes. They showed that while NBs and kNN have the same accuracy of 98.51%, C4.5 is the worst with the accuracy equal to 91.79%.

Hashi et al. (2017) employed decision tree and KNN algorithms to diagnose diabetes disease from the Pima Indians Dataset including 768 data, each with 8 attributes and attained 90.43% and 76.96% accuracy, respectively. This implies that the decision tree is the better-supervised method in terms of the classification accuracy in this case. This dataset had been also used in Iyer et al. (2015), Hayashi and Yukita (2016), Sa'di et al. (2015) and Huang et al. (2015) where they applied the decision tree method and attained 76.96%, 83.83%, 76.52%, and 62.17% accuracies, respectively. Khateeb and Usman (2017) used NB, KNN, J48, and bagging classifiers/ ML classification techniques on a heart disease dataset consisting of 303 instances, each with 14 features. They divided their experimental outcomes into 6 cases and found the highest accuracy of 79.20% by the KNN classifier that utilizes all 14 attributes. Moreover, Tayeb et al. (2017) applied KNN as well to datasets compiled by the University of California to analyze two conditions (chronic kidney failure and heart disease) with an accuracy of roughly 90%.
Pradeep and Naveen (2018) used SVM, NBs, and C4.5 techniques on the North Central Cancer Treatment Group (NCCTG) lung cancer data set to help specialists for better conclusions for cancer survivability rate. The results show that C4.5 performs better in foreseeing lung malignancy with

increment in the training data set. Alharbi (2018) employed a combined genetic-fuzzy algorithm to diagnose lung cancer. He applied the algorithm on 32 patients with 56 attributes without any reduction in dimensions and attained 97.5% accuracy with a 93% confidence. Cherif (2018) developed a new solution to accelerate the kNN algorithm dependent on clustering and attribute separating on the breast cancer database. He compared his proposed algorithm with other classification techniques such as SVM, Artificial Neural Network (ANN), NBs, and KNN. The dataset was isolated into 5 subsets of 113 occurrences, based on which the F-Measure of each technique was calculated five times to achieve an average F-Measure for each. The results demonstrated that while ANN performed the best, its execution time was 2.2 times higher than the proposed algorithm. Joshi and Mehta (2018) employed a well known machine learning algorithm (kNN) to examine its execution on the Wisconsin diagnostic breast cancer dataset. The dataset involved 569 instances with 32 attributes and 2 classes. They used two essential dimensionality reduction strategies (principal component analysis (PCA) and linear discriminant analysis (LDA) and showed that KNN with LDA technique worked better than KNN and KNN with PCA with the accuracies 97.06%, 95.29%, and 95.88%, respectively. Akben (2018) utilized KNN, SVM, and NBs to pre-processed data in order to detect chronic kidney disease (CKD). They first used the methods on raw data and figured out that the classification was not accurate enough to encourage medicinal practitioners. As such, they employed the methods after the data was pre-processed by the k-means clustering approach. The outcomes demonstrated that the accuracy was increased significantly, especially, for the kNN classifier which reached 96%. Lakshmanaprabu et al. (2019) developed a hybrid algorithm involving an optimal deep neural network (ODNN) and a linear discriminate analysis (LDA) to classify lung nodules as either malignant or benign. In their work, the ODNN was first used to extract important features from computed tomography (CT) lung images. Then, LDA was applied to reduce the dimensionality of the features. Finally, a modified gravitational search algorithm was utilized to optimize the ODNN. The sensitivity, specificity, and accuracy of their algorithm were shown to be 96.2%, 94.2%, and 94.56%, respectively. Recently, Alirezaei et al. (2019) deployed four bi-objective meta-heuristic algorithms (multiobjective firefly (MOFA), multi-objective imperialist competitive algorithm (MOICA), non-dominated sorting genetic algorithm (NSGA-II), and multi-objective particle swarm optimization (MOPSO)) to determine the least number of attributes with the highest classification accuracy rate. Because of the importance of data quality, first of all, they utilized some preprocessing methods. Then SVM was used as a classifier. Among the above meta-heuristics, MOFA was the best with 95.12% accuracy.

CHAPTER 3

LIBRARIES

First step is to import all the important libraries.
Libraries are-

- Numpy
- Pandas
- Seaborn
- Matplotlib
- Scikit-learn

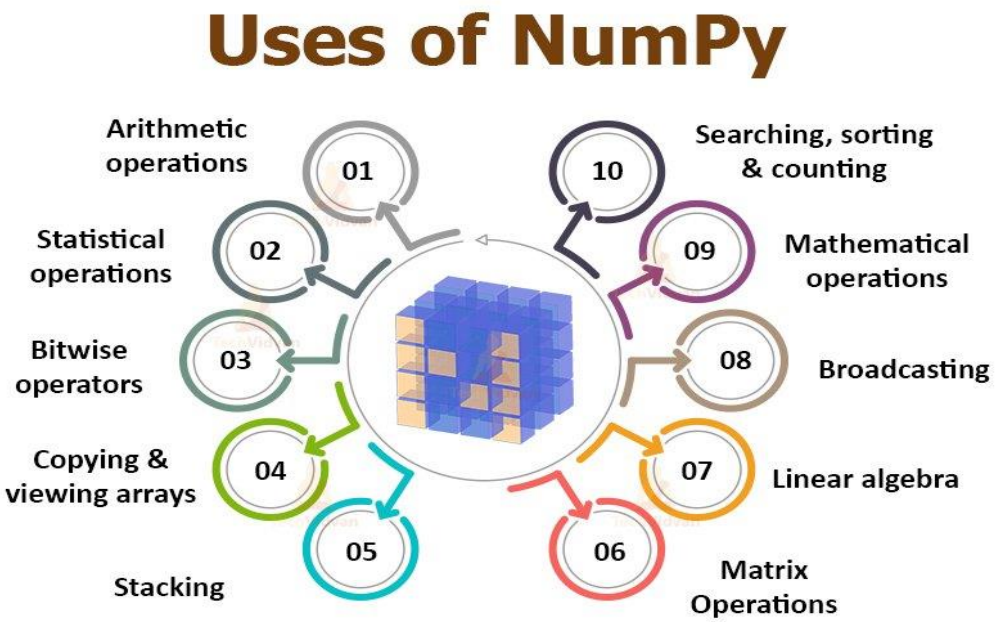
3.1 Numpy

⇨ NumPy is a Python library used for working with arrays.

⇨ It also has functions for working in domain of linear algebra, fourier transform, and matrices.

⇨ NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

⇨ NumPy stands for Numerical Python.

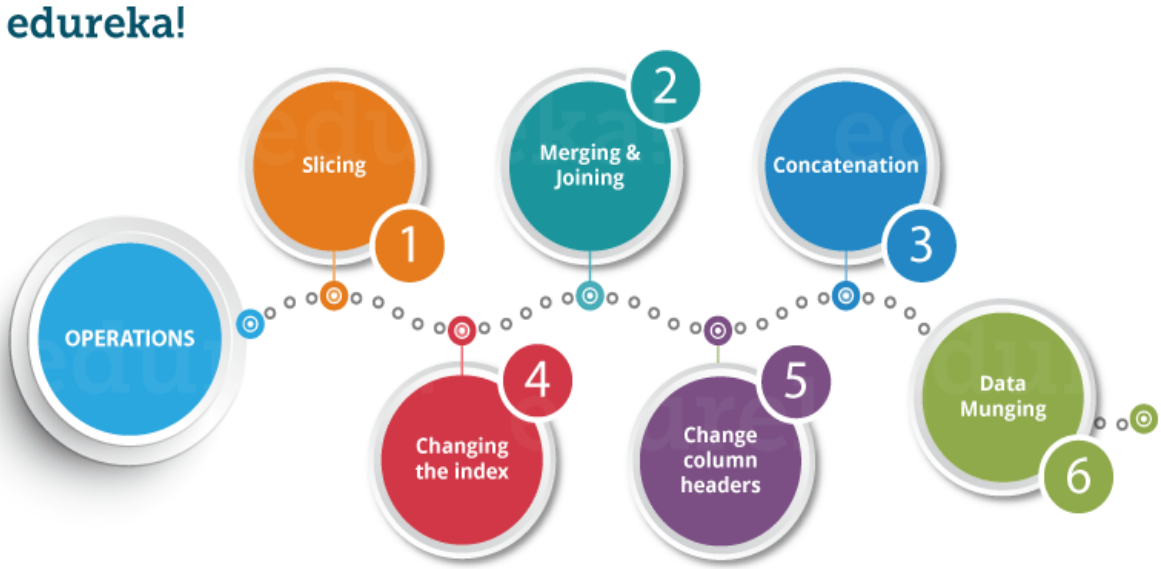


3.2 Pandas

=>Pandas is a Python library used for working with data sets.

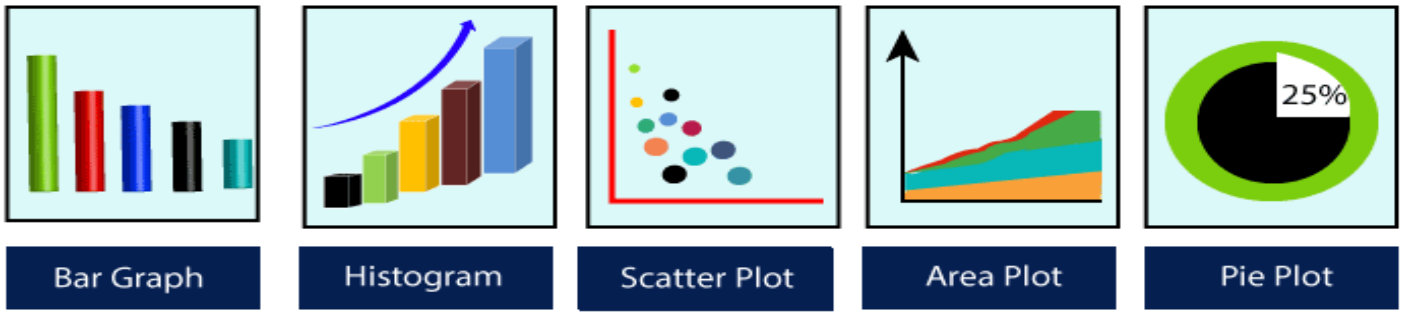
=> It has functions for analyzing, cleaning, exploring, and manipulating data.

=> The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.



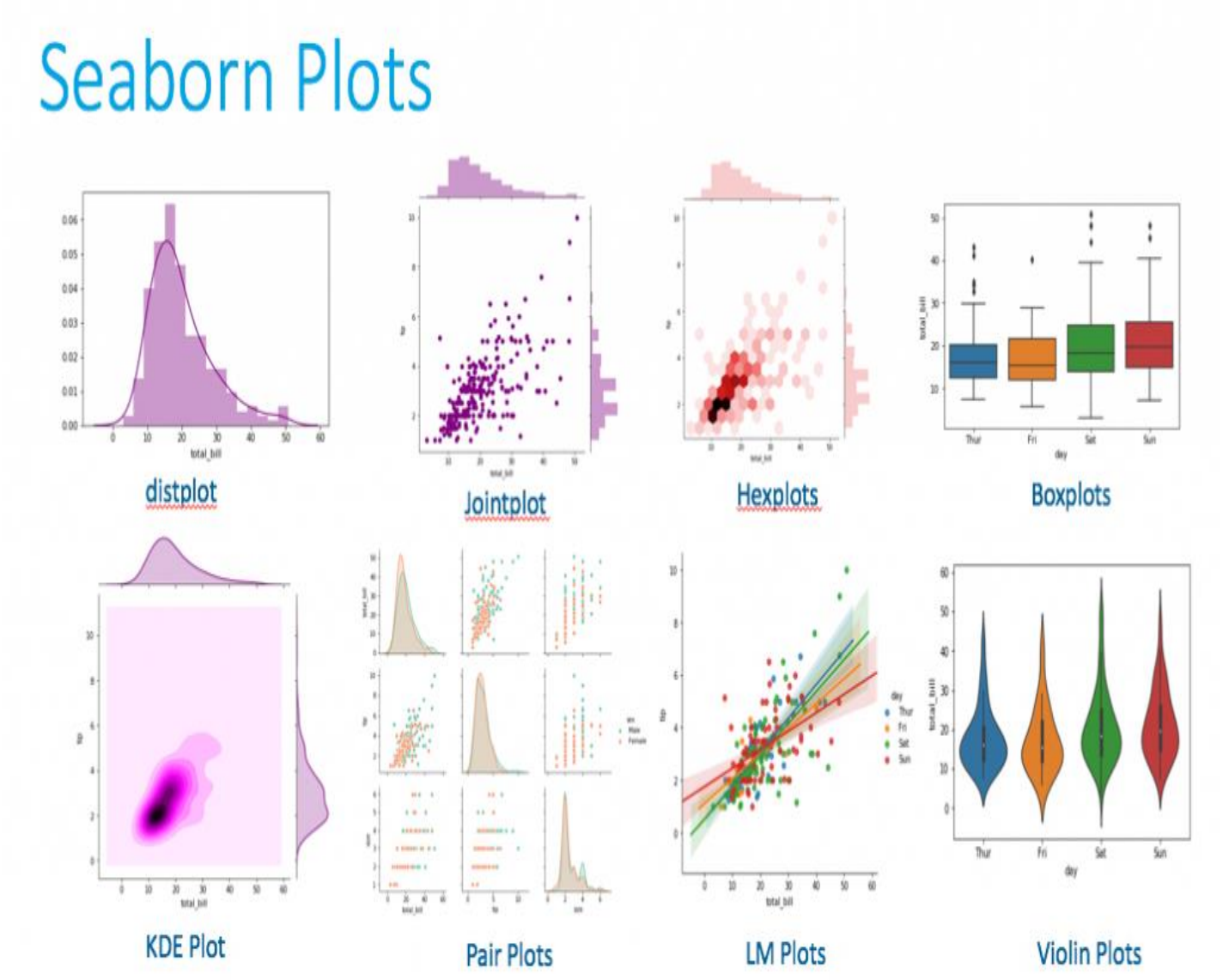
3.3 Matplotlib

=>Matplotlib is a multi-platform data visualization library built on NumPy arrays, and designedto work with the broader SciPy stack. It was conceived by John Hunter in 2002, originally as a patch to IPython for enabling interactive MATLAB-style plotting via gnuplot from the IPython command line.



3.4 Seaborn

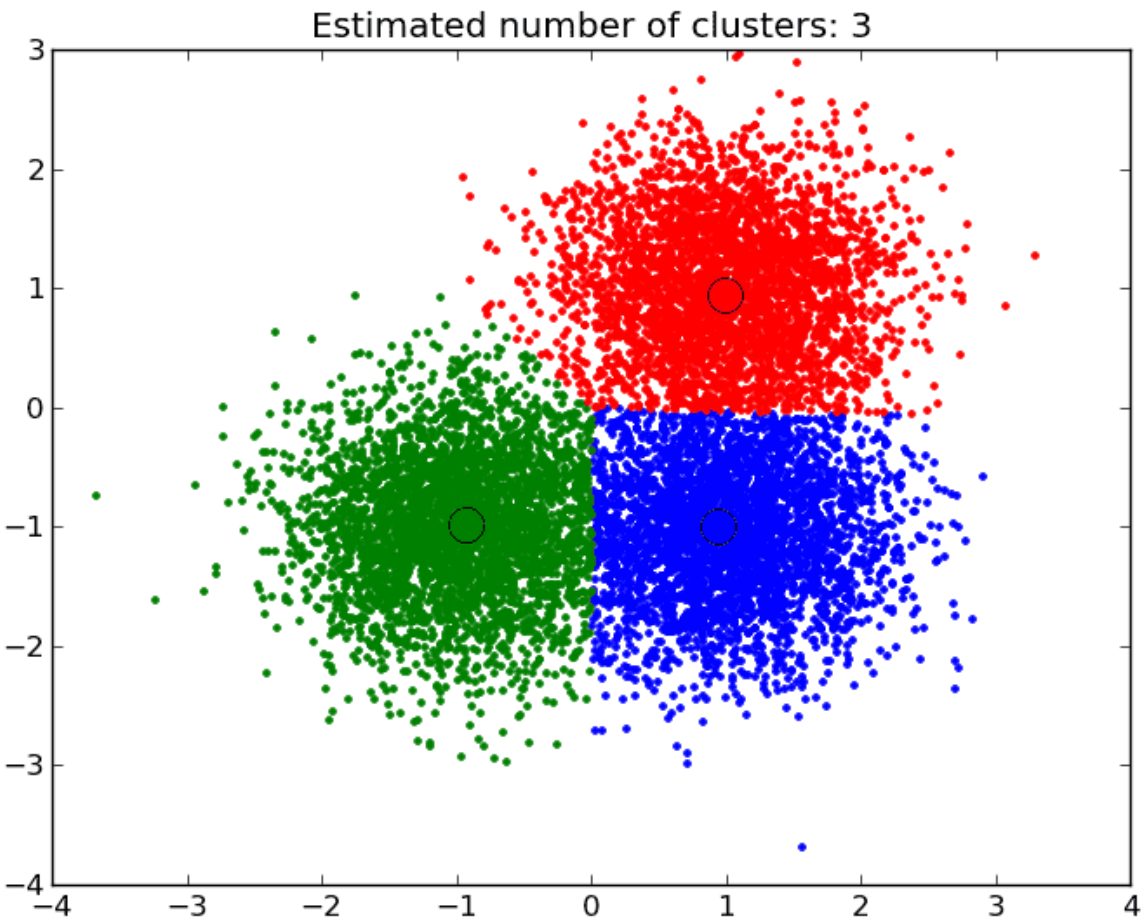
Seaborn helps to visualize the statistical relationships, To understand how variables in a dataset are related to one another and how that relationship is dependent on other variables, we perform statistical analysis. This Statistical analysis helps to visualize the trends and identify various patterns in the dataset.



3.5 Sklearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

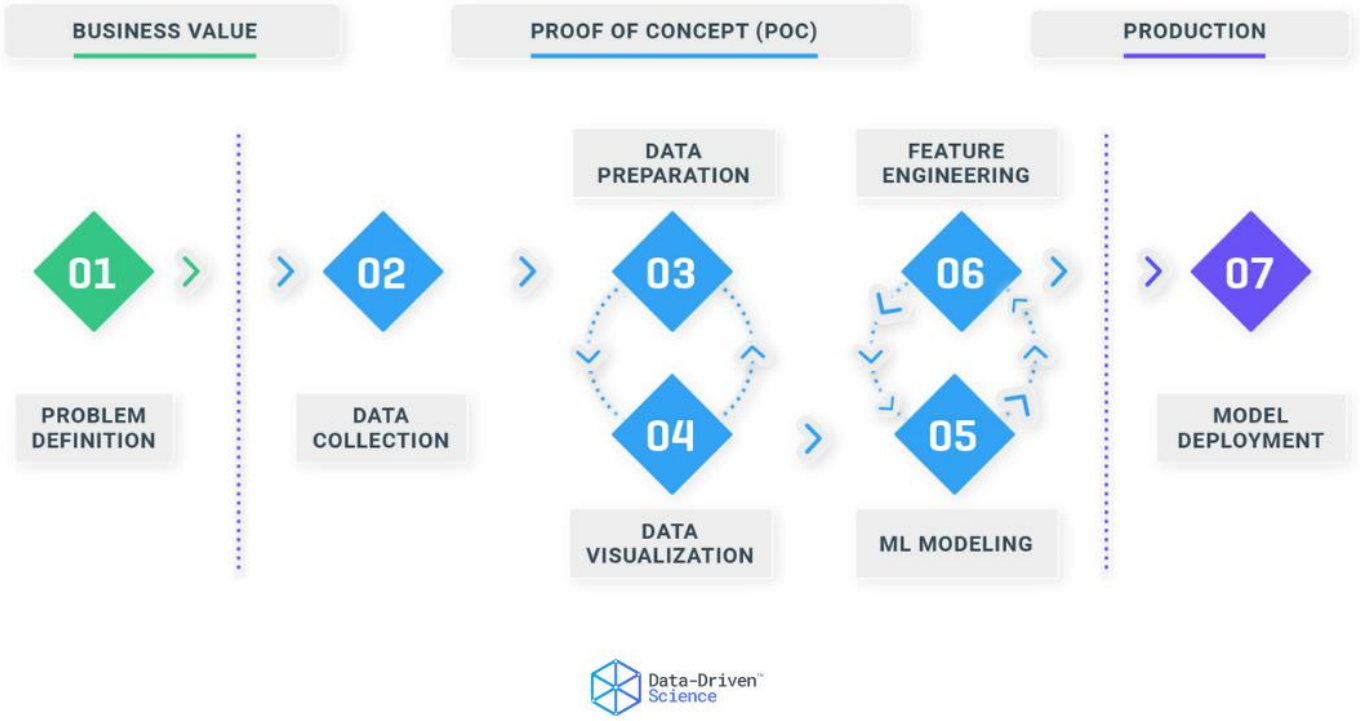
For example-



CHAPTER 4

STAGES OF OUR MODEL

1. Problem Definition
2. Data Collection
3. Data Preparation
4. Data Visualization
5. ML Modeling
6. Feature Engineering



These 7 stages are the key steps in our framework. We have categorized them additionally into groups to get a better understanding of the larger picture.

The stages are grouped into 3 phases:

1. Business Value
2. Proof of Concept (POC)
3. Production

Phase 1 — Business Value

It is absolutely crucial to adopt a business mindset when thinking about a problem that should be solved with Machine Learning — defining customer benefits and creating business impact is top priority. Domain expertise and knowledge is also essential as the true power of data can only be harnessed if the domain is well known and understood.

Phase 2 — Proof of Concept (POC)

Proof of Concept (POC) is the most comprehensive part of our framework. From Data Collection to Feature Engineering, 5 stages of our ML framework are included here. Core of any POC to test an idea in terms of its feasibility and value to the business. Also, questions around performance and evaluation metrics are answered in that phase. Only a strong POC that delivers business value and is feasible allows one putting the ML Model into production.

Phase 3 — Production

In the third phase, one is taking the ML model and scaling it. The goal is to integrate Machine Learning into a business process solving a problem with a superior solution compared to, for example, traditional programming. The process of taking a trained ML model and making its predictions available to users or other systems is known as model deployment. Lastly, it is also essential to iterate on the ML model over time to improve it.

4.1 Problem Definition

The first stage in the DDS Machine Learning Framework is to define and understand the problem that someone is going to solve. Start by analyzing the goals and the *why* behind a particular problem statement. Understand the power of data and how one can use it to make a change and drive results. And asking the right questions is always a great start.

Few possible questions:

- What is the business?
- Why does the problem need to be solved?
- Is a traditional solution available to solve the problem?
- If probabilistic in nature, then does available data allow to model it?
- What is a measurable business goal?

4.2 Data Collection

Once the goal is clearly defined, one has to start getting the data that is needed from various available data sources.

At this stage, some of the questions worth considering are:

- What data do I need for my project?
- Where is that data available?
- How can I obtain it?
- What is the most efficient way to store and access all of it?

There are many different ways to collect data that is used for Machine Learning. For example, focus groups, interviews, surveys, and internal usage & user data. Also, public data can be another source and is usually free. These include research and trade associations such as banks, publicly-traded corporations, and others. If data isn't publicly available, one could also use web scraping to get it (however, there are some legal restrictions).

	GENDER	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHING	SHORTN OF BRE
0	MALE	69	0	1	1	0	0	1	0	1	1	1	
1	MALE	74	1	0	0	0	1	1	1	0	0	0	
2	FEMALE	59	0	0	0	1	0	1	0	1	0	1	
3	MALE	63	1	1	1	0	0	0	0	0	1	0	
4	FEMALE	63	0	1	0	0	0	0	0	1	0	1	

4.3 Data Preparation

The third stage is the most time-consuming and labor-intensive. Data Preparation can take up to 70% and sometimes even 90% of the overall project time. But what is the purpose of this stage?

Well, the type and quality of data that is used in a Machine Learning model affects the output considerably. In Data Preparation one explores, pre-processes, conditions, and transforms data prior to modeling and analysis. It is absolutely essential to understand the data, learn about it, and become familiar before moving on to the next stage.

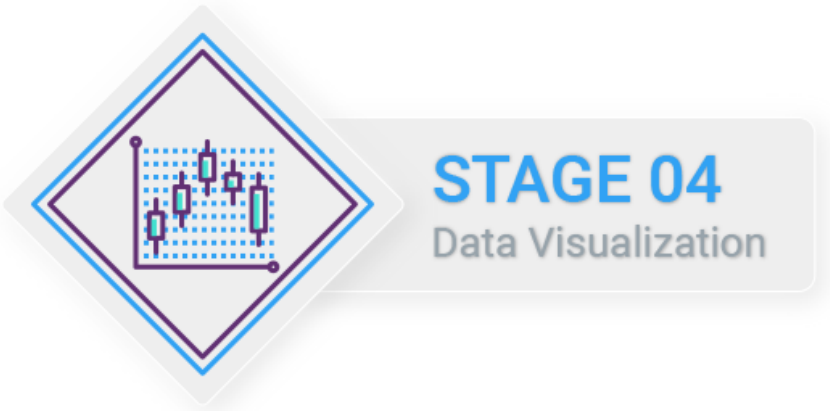
Some of the steps involved in this stage are:

- Data Filtering
- Data Validation & Cleansing
- Data Formatting
- Data Aggregation & Reconciliation

```
data.info()
```

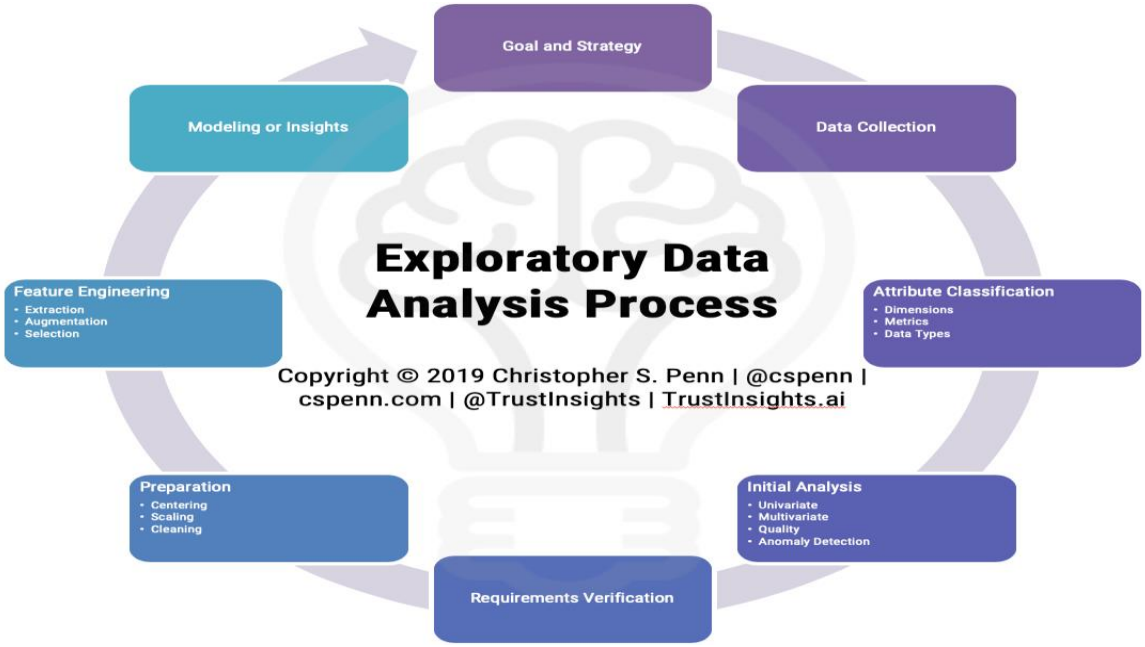
```
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   GENDER                 276 non-null    object
 1   AGE                    276 non-null    int64
 2   SMOKING                276 non-null    object
 3   YELLOW_FINGERS         276 non-null    object
 4   ANXIETY                276 non-null    object
 5   PEER_PRESSURE          276 non-null    object
 6   CHRONIC DISEASE        276 non-null    object
 7   FATIGUE                276 non-null    object
 8   ALLERGY                276 non-null    object
 9   WHEEZING               276 non-null    object
 10  ALCOHOL CONSUMING      276 non-null    object
 11  COUGHING               276 non-null    object
 12  SHORTNESS OF BREATH    276 non-null    object
 13  SWALLOWING DIFFICULTY  276 non-null    object
 14  CHEST PAIN             276 non-null    object
 15  LUNG_CANCER            276 non-null    object
```

4.4 Data Visualization



Data Visualization is used to perform Exploratory Data Analysis (EDA). When one is dealing with large volumes of data, building graphs is the best way to explore and communicate findings. Visualization is an incredibly helpful tool to identify patterns and trends in data, which leads to clearer understanding and reveals important insights. Data Visualization also helps for faster decision making through the graphical illustration.

4.4.1 Exploratory Data Analysis



The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Data scientists can use exploratory analysis to ensure the results they produce are valid and applicable to any desired business outcomes and goals. EDA also helps stakeholders by confirming they are asking the right questions. EDA can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modeling, including machine learning.

Exploratory data analysis tools

Specific statistical functions and techniques you can perform with EDA tools include:

- Clustering and dimension reduction techniques, which help create graphical displays of high-dimensional data containing many variables.
- Univariate visualization of each field in the raw dataset, with summary statistics.
- Bivariate visualizations and summary statistics that allow you to assess the relationship between each variable in the dataset and the target variable you're looking at.
- Multivariate visualizations, for mapping and understanding interactions between different fields in the data.
- K-means Clustering is a clustering method in unsupervised learning where data points are assigned into K groups, i.e. the number of clusters, based on the distance from each group's centroid. The data points closest to a particular centroid will be clustered under the same category. K-means Clustering is commonly used in market segmentation, pattern recognition, and image compression.
- Predictive models, such as linear regression, use statistics and data to predict outcomes.

Types of exploratory data analysis

There are four primary types of EDA:

- **Univariate non-graphical.** This is simplest form of data analysis, where the data being analyzed consists of just one variable. Since it's a single variable, it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.
- **Univariate graphical.** Non-graphical methods don't provide a full picture of the data. Graphical methods are therefore required. Common types of univariate graphics include:
    - Stem-and-leaf plots, which show all data values and the shape of the distribution.
    - Histograms, a bar plot in which each bar represents the frequency (count) or proportion (count/total count) of cases for a range of values.
    - Box plots, which graphically depict the five-number summary of minimum, first quartile, median, third quartile, and maximum.

- **Multivariate nongraphical:** Multivariate data arises from more than one variable. Multivariate non-graphical EDA techniques generally show the relationship between two or more variables of the data through cross-tabulation or statistics.
- **Multivariate graphical:** Multivariate data uses graphics to display relationships between two or more sets of data. The most used graphic is a grouped bar plot or bar chart with each group representing one level of one of the variables and each bar within a group representing the levels of the other variable.

4.4.2 Finding Missing Values: The real-world data often has a lot of missing values.

The cause of missing values can be data corruption or failure to record data. The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.

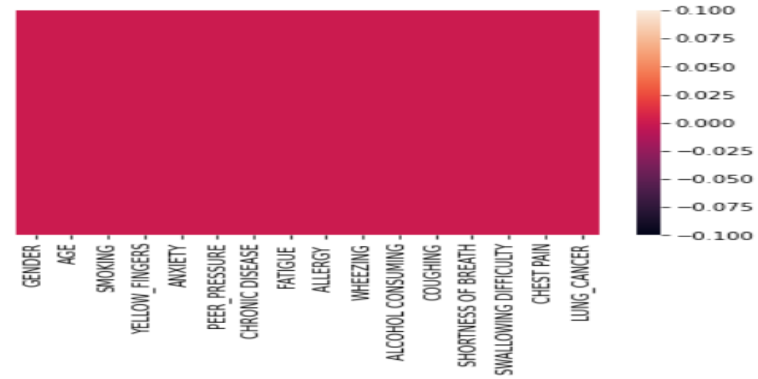
Finding Missing Values:

```
data.isnull().sum()
```

```
GENDER                    0
AGE                       0
SMOKING                   0
YELLOW_FINGERS            0
ANXIETY                   0
PEER_PRESSURE             0
CHRONIC DISEASE           0
FATIGUE                   0
ALLERGY                   0
WHEEZING                  0
ALCOHOL CONSUMING         0
COUGHING                  0
SHORTNESS OF BREATH       0
SWALLOWING DIFFICULTY     0
CHEST PAIN                0
LUNG_CANCER               0
dtype: int64
```

```
#Also using heatmap find null values
sns.heatmap(data.isnull(),yticklabels = False)
```

```
<AxesSubplot:>
```



Not Found! Any null values.So,no need to clean the data.

4.4.3 <u>Relationship Analysis</u>: There are two kinds of relationship of analysis of correlation : 1. Positive correlation A positive correlation is a relationship between 2 variables which the increase of one variable causes an increase for another variable. Or it can also be defined otherwise, the lower a variable, the more it moves down as well as other variables.

	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL CONSUMING	COUGHINC
AGE	1.000000	-0.073410	0.025773	0.050605	0.037848	-0.003431	0.021606	0.037139	0.052803	0.052049	0.16865
SMOKING	-0.073410	1.000000	-0.020799	0.153389	-0.030364	-0.149415	-0.037803	-0.030179	-0.147081	-0.052771	-0.13855
YELLOW_FINGERS	0.025773	-0.020799	1.000000	0.558344	0.313067	0.015316	-0.099644	-0.147130	-0.058756	-0.273643	0.02080
ANXIETY	0.050605	0.153389	0.558344	1.000000	0.210278	-0.006938	-0.181474	-0.159451	-0.174009	-0.152228	-0.21884
PEER_PRESSURE	0.037848	-0.030364	0.313067	0.210278	1.000000	0.042893	0.094661	-0.066887	-0.037769	-0.132603	-0.06822
CHRONIC DISEASE	-0.003431	-0.149415	0.015316	-0.006938	0.042893	1.000000	-0.099411	0.134309	-0.040546	0.010144	-0.16081
FATIGUE	0.021606	-0.037803	-0.099644	-0.181474	0.094661	-0.099411	1.000000	-0.001841	0.152151	-0.181573	0.14853
ALLERGY	0.037139	-0.030179	-0.147130	-0.159451	-0.066887	0.134309	-0.001841	1.000000	0.166517	0.378125	0.20636
WHEEZING	0.052803	-0.147081	-0.058756	-0.174009	-0.037769	-0.040546	0.152151	0.166517	1.000000	0.261061	0.35365
ALCOHOL CONSUMING	0.052049	-0.052771	-0.273643	-0.152228	-0.132603	0.010144	-0.181573	0.378125	0.261061	1.000000	0.19802
COUGHING	0.168654	-0.138553	0.020803	-0.218843	-0.068224	-0.160813	0.148538	0.206367	0.353657	0.198023	1.00000
SHORTNESS OF BREATH	-0.009189	0.051761	-0.109959	-0.155678	-0.214115	-0.011760	0.407027	-0.018030	0.042289	-0.163370	0.28496
SWALLOWING DIFFICULTY	0.003199	0.042152	0.333349	0.478820	0.327764	0.068263	-0.115727	-0.037581	0.108304	-0.000635	-0.13688
CHEST PAIN	-0.035806	0.106984	-0.099169	-0.123182	-0.074655	-0.048895	0.013757	0.245440	0.142846	0.310767	0.07798

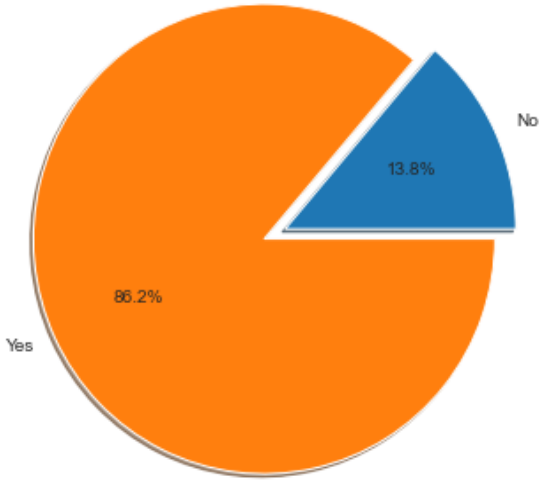
Other common types of multivariate graphics include:

- Scatter plot, which is used to plot data points on a horizontal and a vertical axis to show how much one variable is affected by another.
- Multivariate chart, which is a graphical representation of the relationships between factors and a response.
- Run chart, which is a line graph of data plotted over time.
- Bubble chart, which is a data visualization that displays multiple circles (bubbles) in a two-dimensional plot.
- Heat map, which is a graphical representation of data where values are depicted by color.

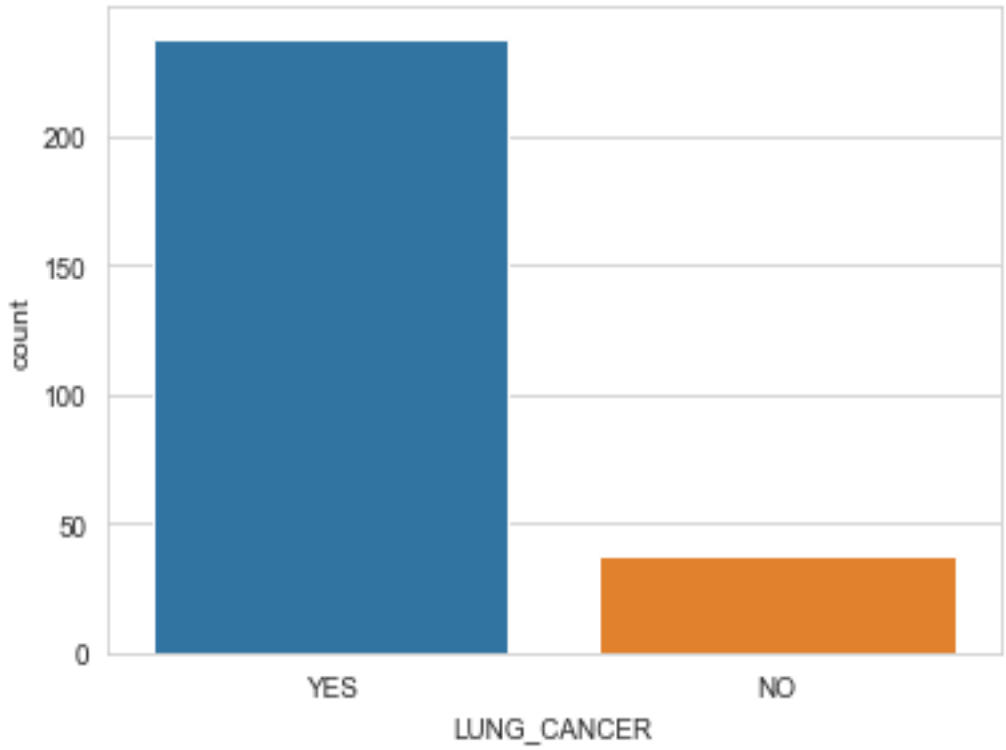
Here are some common ways of visualization:

- Area Chart
- Bar Chart
- Box-and-whisker Plots

- Bubble Cloud
- Dot Distribution Map
- Heat Map
- Histogram

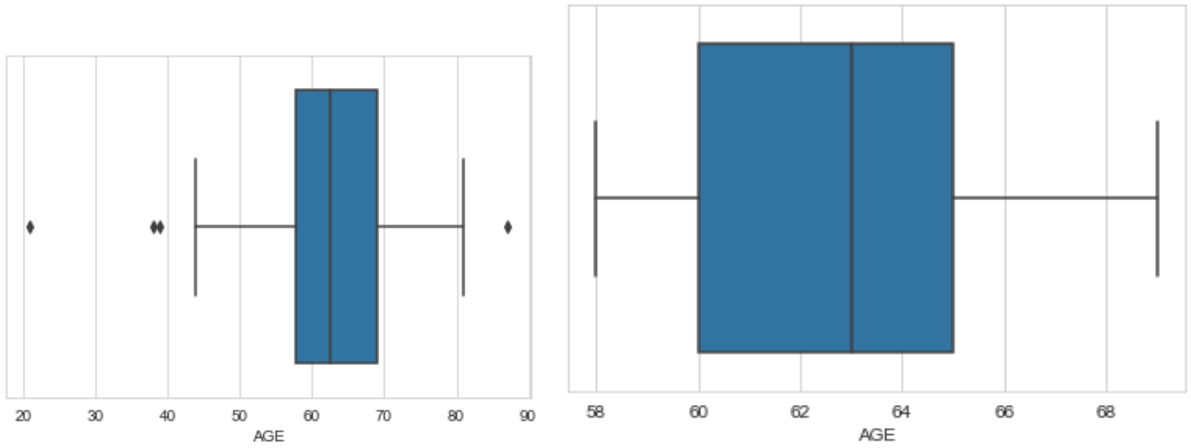


- Network Diagram

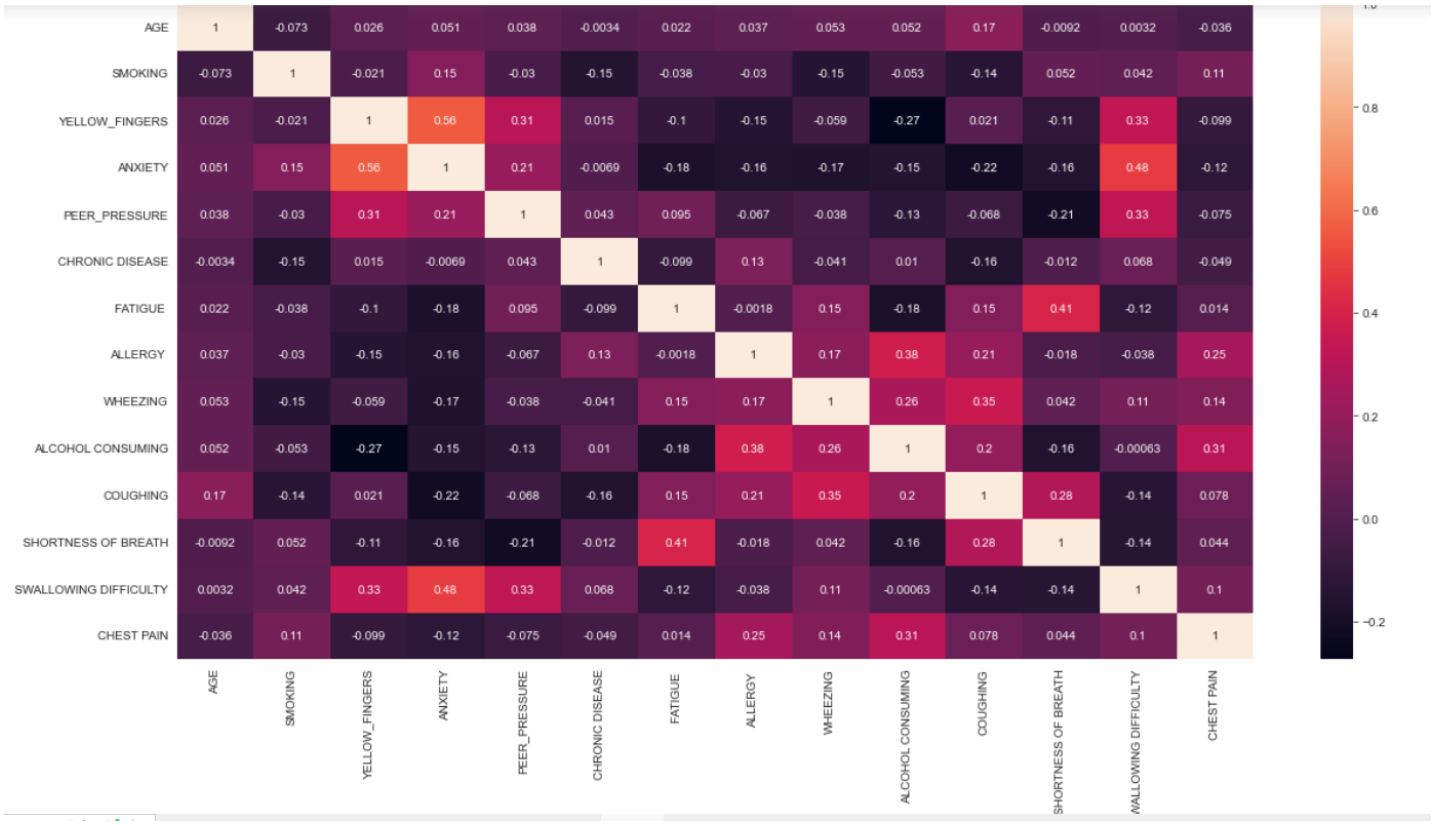


Outliers are an important part of a dataset. They can hold useful information about your data. Outliers can give helpful insights into the data you're studying, and they can have an effect on statistical results. This can potentially help you discover inconsistencies and detect any errors in your statistical processes.

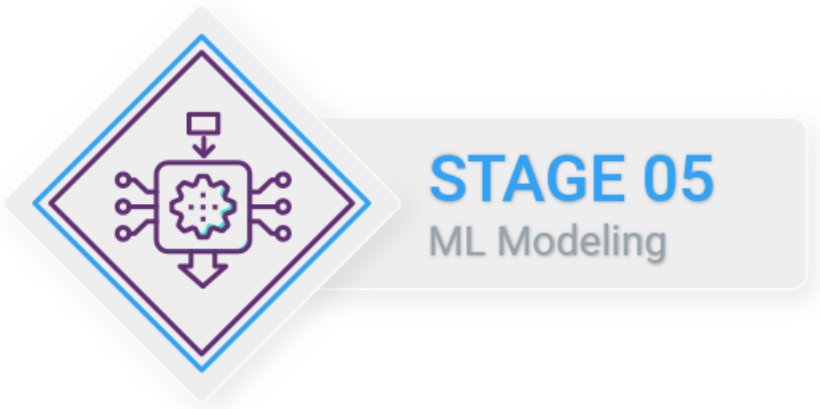
So, knowing how to find outliers in a dataset will help you better understand your data.



4.4.4 <u>Correlation Matrix</u> : Correlation Matrix is a statistical method of showing the relationship between two or more variables and the interrelation in their movements etc. In short, it helps in defining the relationship and dependence among the variables. It is a very commonly used mechanism and finds its application in the field of Investment Management, Risk Management, Statistics as well as Economics to name a few.



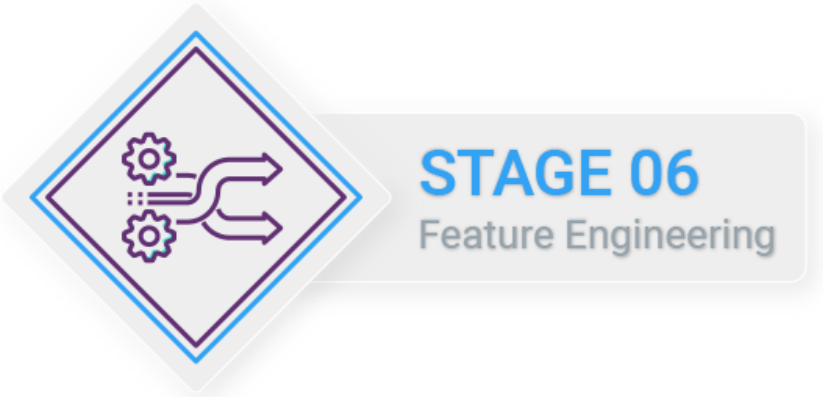
4.5 ML Modeling



Finally, this is where *'the magic happens'*. Machine Learning is finding patterns in data, and one can perform either supervised or unsupervised learning. ML tasks include regression, classification, forecasting, and clustering.

In this stage of the process one has to apply mathematical, computer science, and business knowledge to train a Machine Learning algorithm that will make predictions based on the provided data. It is a crucial step that will determine the quality and accuracy of future predictions in new situations. Additionally, ML algorithms help to identify key features with high predictive value.

4.6 Feature Engineering



Machine Learning algorithms learn recurring patterns from data. Carefully engineered features are a robust representation of those patterns.

Feature Engineering is a process to achieve a set of features by performing mathematical, statistical, and heuristic procedures. It is a collection of methods for identifying an optimal set of inputs to the Machine Learning algorithm. Feature Engineering is extremely important because well-engineered features make learning possible with simple models.

Following are the characteristics of good features:

- Represents data in an unambiguous way
- Ability to captures linear and non-linear relationships among data points
- Capable of capturing the precise meaning of input data
- Capturing contextual details

CHAPTER 5

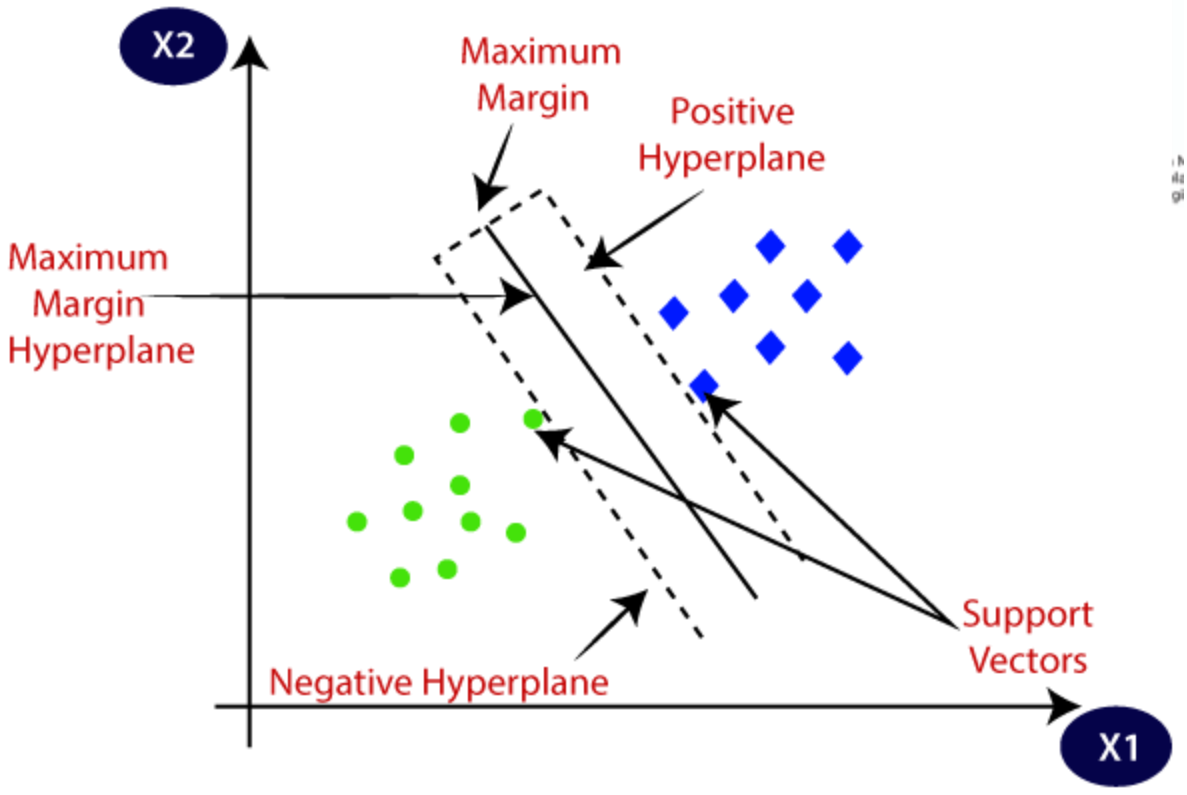
IMPLEMENTATION OF ALGORITHM

5.1 SVM (SUPPORT VECTOR MACHINE)

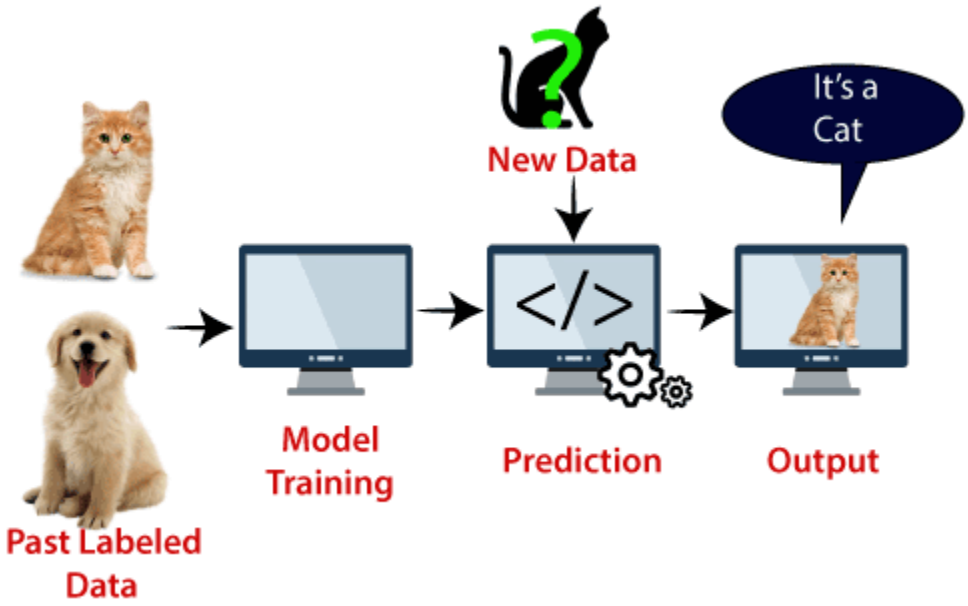
Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Example: SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



SVM algorithm can be used for **Face detection, image classification, text categorization,** etc.

Types of SVM

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Hyperplane and Support Vectors in the SVM algorithm:

Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

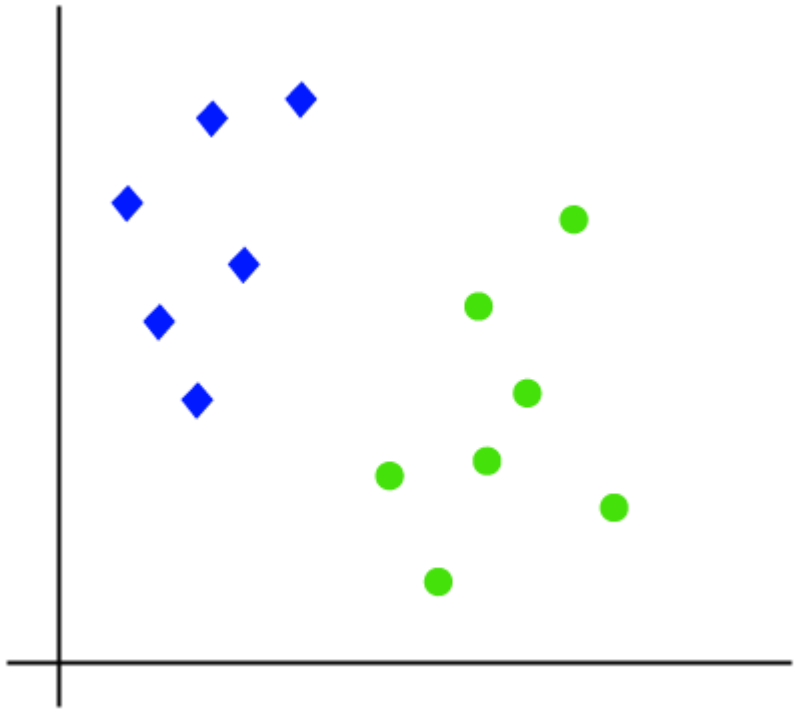
Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

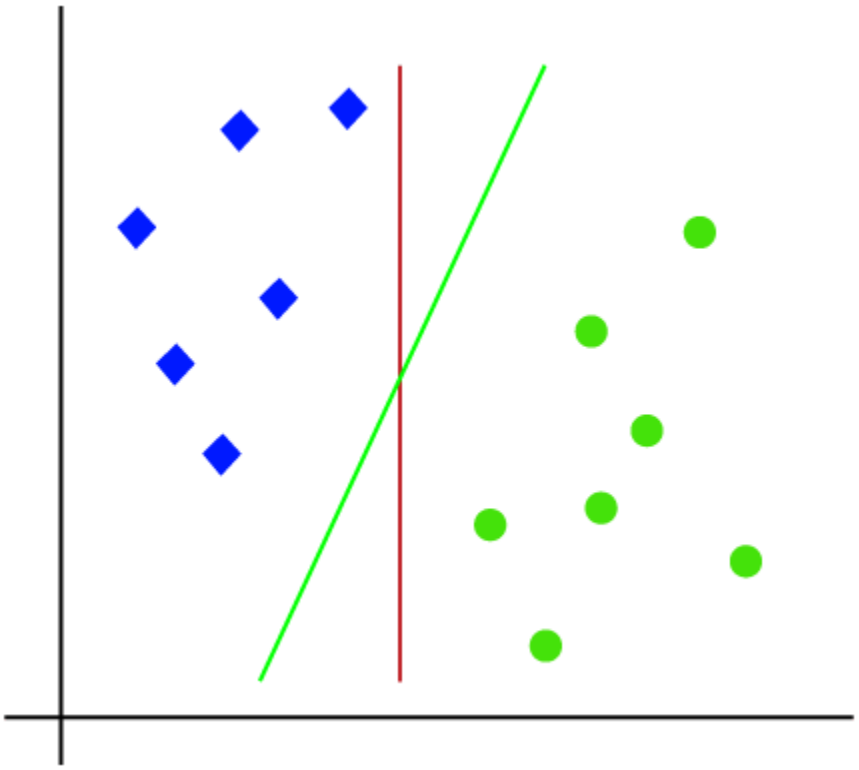
How does SVM works?

Linear SVM:

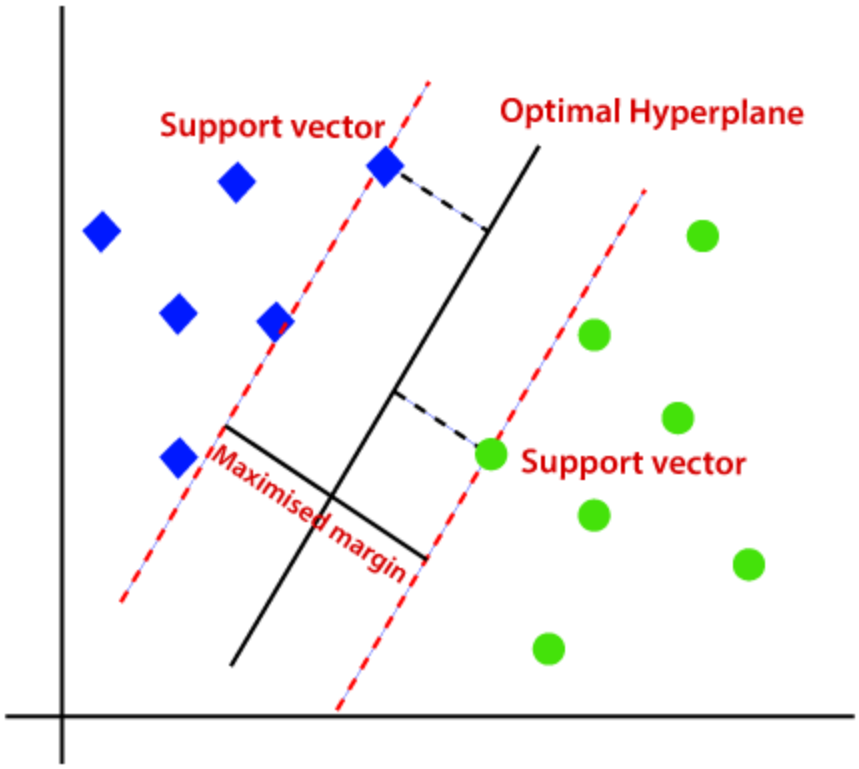
The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair(x1, x2) of coordinates in either green or blue. Consider the below image:



So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:

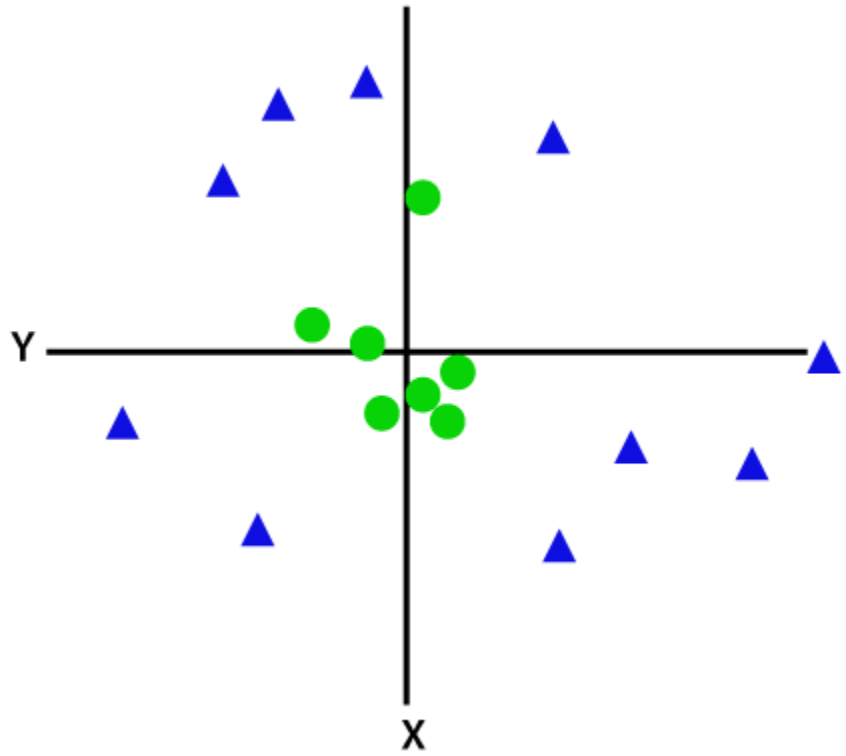


Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a **hyperplane**. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as **margin**. And the goal of SVM is to maximize this margin. The **hyperplane** with maximum margin is called the **optimal hyperplane**.



Non-Linear SVM:

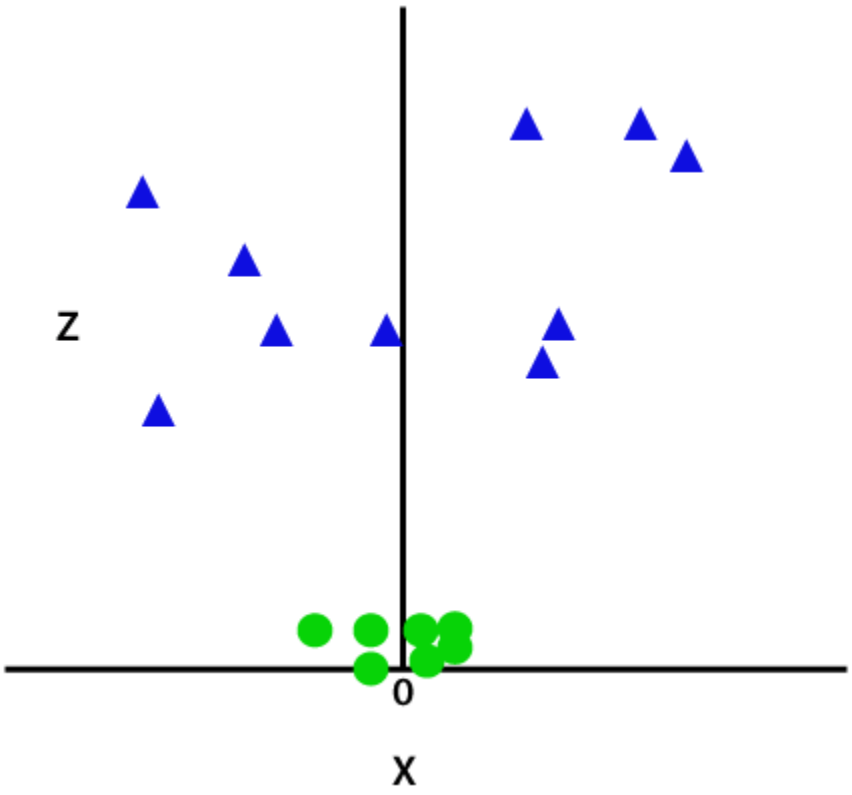
If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



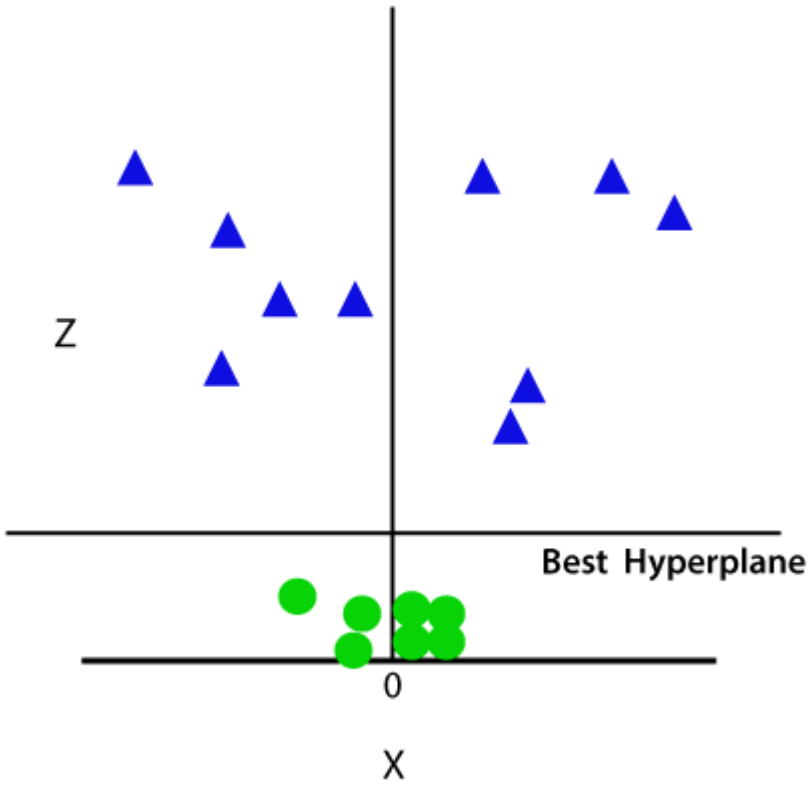
So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

$z=x^2 +y^2$

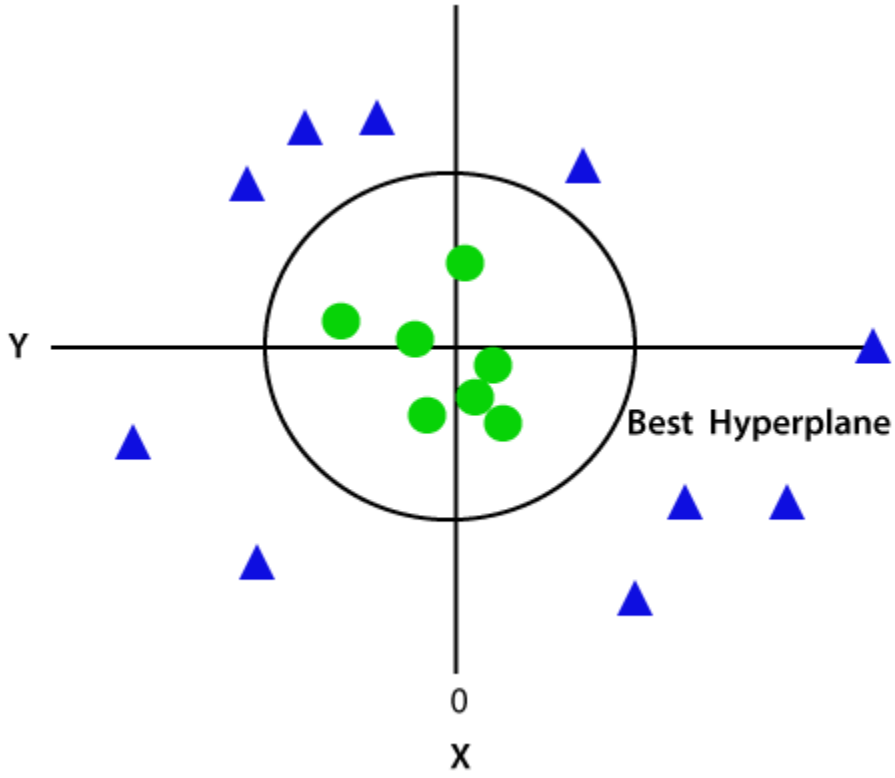
By adding the third dimension, the sample space will become as below image:



So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:



Hence we get a circumference of radius 1 in case of non-linear data.

Python Implementation of Support Vector Machine

Now we will implement the SVM algorithm using Python. Here we will use the same dataset **user_data**, which we have used in Logistic regression and KNN classification.

- ### Data Pre-processing step

Till the Data pre-processing step, the code will remain the same. Below is the code:

```
1. #Data Pre-processing Step
2. # importing libraries
3. import numpy as nm
4. import matplotlib.pyplot as mtp
5. import pandas as pd
6.
7. #importing datasets
```

```
8.  data_set= pd.read_csv('user_data.csv')
9.
10. #Extracting Independent and dependent Variable
11. x= data_set.iloc[:, [2,3]].values
12. y= data_set.iloc[:, 4].values
13.
14. # Splitting the dataset into training and test set.
15. from sklearn.model_selection import train_test_split
16. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
17. #feature Scaling
18. from sklearn.preprocessing import StandardScaler
19. st_x= StandardScaler()
20. x_train= st_x.fit_transform(x_train)
21. x_test= st_x.transform(x_test)
```

After executing the above code, we will pre-process the data. The code will give the dataset as:

data_set - DataFrame

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0

Format | Resize | Background color | Column min/max | Save and Close | Close

The scaled output for the test set will be:

x_test - NumPy array

	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

Format | Resize | Background color | Save and Close | Close

y_test - NumPy array

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Format | Resize | Background color | Save and Close | Close

Fitting the SVM classifier to the training set:

Now the training set will be fitted to the SVM classifier. To create the SVM classifier, we will import **SVC** class from **Sklearn.svm** library. Below is the code for it:

```
1. from sklearn.svm import SVC # "Support vector classifier"
2. classifier = SVC(kernel='linear', random_state=0)
3. classifier.fit(x_train, y_train)
```

In the above code, we have used **kernel='linear'**, as here we are creating SVM for linearly separable data. However, we can change it for non-linear data. And then we fitted the classifier to the training dataset(x_train, y_train)

Output:

```
Out[8]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=0,
    shrinking=True, tol=0.001, verbose=False)
```

The model performance can be altered by changing the value of **C(Regularization factor), gamma, and kernel**.

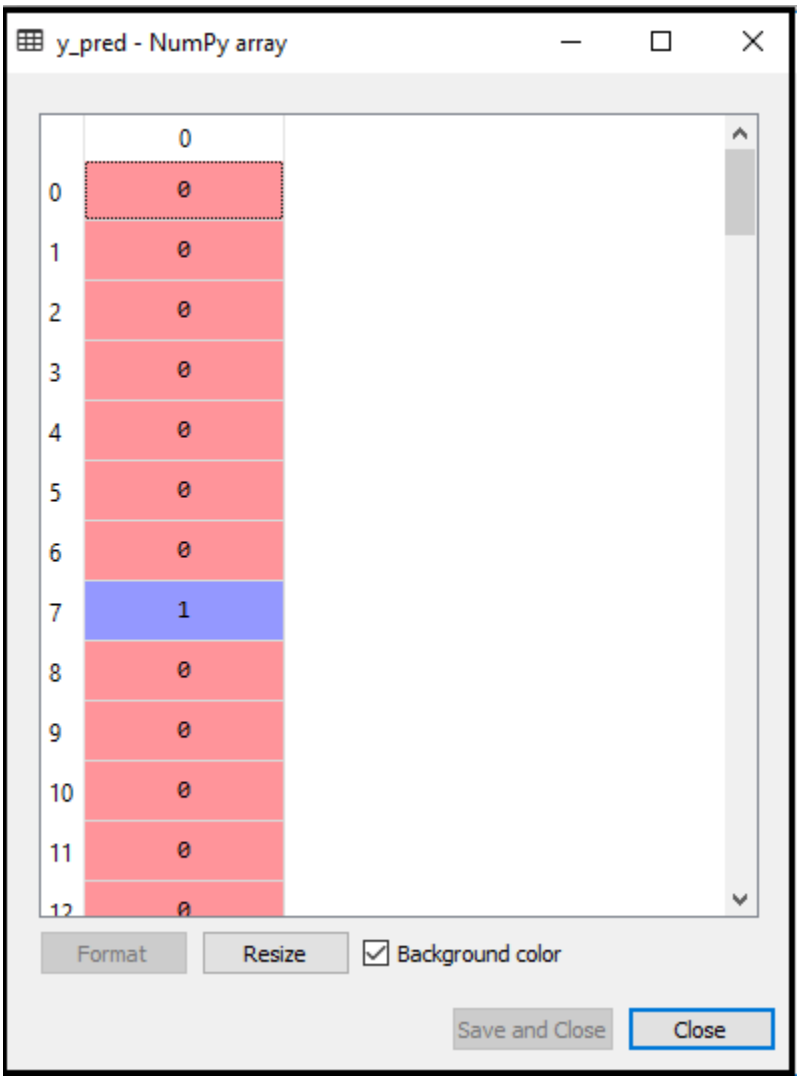
- **Predicting the test set result:**

  Now, we will predict the output for test set. For this, we will create a new vector y_pred. Below is the code for it:

```
1. #Predicting the test set result
2. y_pred= classifier.predict(x_test)
```

After getting the y_pred vector, we can compare the result of **y_pred** and **y_test** to check the difference between the actual value and predicted value.

Output: Below is the output for the prediction of the test set:



- **Creating the confusion matrix:**

  Now we will see the performance of the SVM classifier that how many incorrect predictions are there as compared to the Logistic regression classifier. To create the confusion matrix, we need to import the **confusion_matrix** function of the sklearn library. After importing the function, we will call it using a new variable **cm**. The function takes two parameters, mainly **y_true**(the actual values) and **y_pred** (the targeted value return by the classifier). Below is the code for it:

```
1. #Creating the Confusion matrix
2. from sklearn.metrics import confusion_matrix
3. cm= confusion_matrix(y_test, y_pred)
```

Output:

As we can see in the above output image, there are 66+24= 90 correct predictions and 8+2= 10 correct predictions. Therefore we can say that our SVM model improved as compared to the Logistic regression model.

- **Visualizing the training set result:**

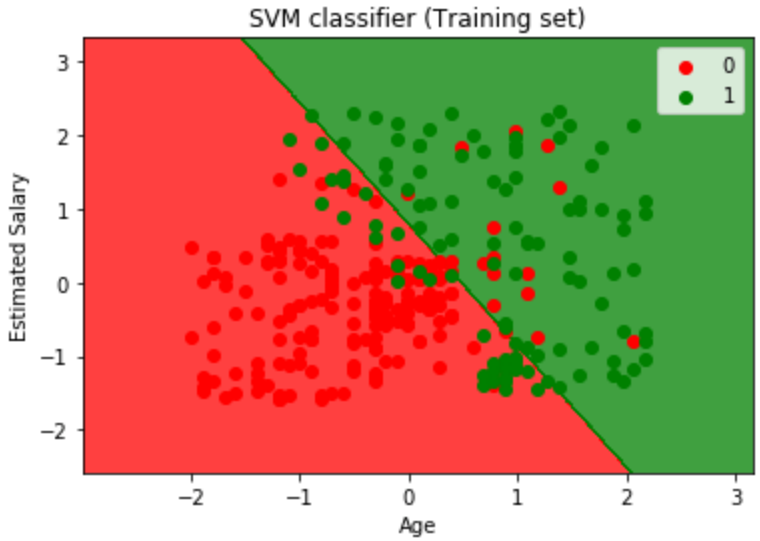
  Now we will visualize the training set result, below is the code for it:

```
1. from matplotlib.colors import ListedColormap
2. x_set, y_set = x_train, y_train
3. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
4. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
5. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
6. alpha = 0.75, cmap = ListedColormap(('red', 'green')))
7. mtp.xlim(x1.min(), x1.max())
8. mtp.ylim(x2.min(), x2.max())
9. for i, j in enumerate(nm.unique(y_set)):
10.     mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
11.         c = ListedColormap(('red', 'green'))(i), label = j)
12. mtp.title('SVM classifier (Training set)')
13. mtp.xlabel('Age')
14. mtp.ylabel('Estimated Salary')
15. mtp.legend()
```

16. mtp.show()

Output:

By executing the above code, we will get the output as:



As we can see, the above output is appearing similar to the Logistic regression output. In the output, we got the straight line as hyperplane because we have **used a linear kernel in the classifier**. And we have also discussed above that for the 2d space, the hyperplane in SVM is a straight line.

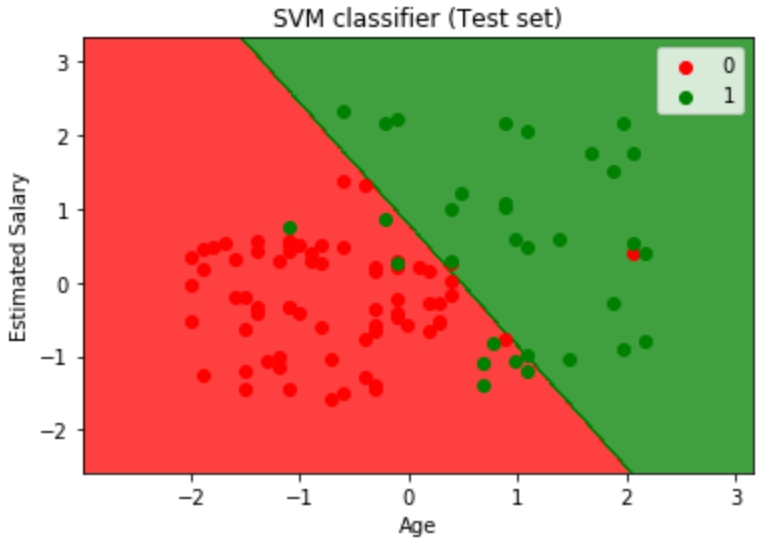
- ## Visualizing the test set result:

1. #Visulaizing the test set result
2. from matplotlib.colors **import** ListedColormap
3. x_set, y_set = x_test, y_test
4. x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step =0.01),
5. nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
6. mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
7. alpha = 0.75, cmap = ListedColormap(('red','green')))
8. mtp.xlim(x1.min(), x1.max())
9. mtp.ylim(x2.min(), x2.max())
10. **for** i, j in enumerate(nm.unique(y_set)):
11. mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
12. c = ListedColormap(('red', 'green'))(i), label = j)
13. mtp.title('SVM classifier (Test set)')

14. mtp.xlabel('Age')
15. mtp.ylabel('Estimated Salary')
16. mtp.legend()
17. mtp.show()

Output:

By executing the above code, we will get the output as:



As we can see in the above output image, the SVM classifier has divided the users into two regions (Purchased or Not purchased). Users who purchased the SUV are in the red region with the red scatter points. And users who did not purchase the SUV are in the green region with green scatter points. The hyperplane has divided the two classes into Purchased and not purchased variable.

5.2 KNN(K NEAREST NEIGHBOUR)

Step 1: Calculate Euclidean Distance

The first step is to calculate the distance between two rows in a dataset.

Rows of data are mostly made up of numbers and an easy way to calculate the distance between two rows or vectors of numbers is to draw a straight line. This makes sense in 2D or 3D and scales nicely to higher dimensions.

We can calculate the straight line distance between two vectors using the Euclidean distance measure. It is calculated as the square root of the sum of the squared differences between the two vectors.

- Euclidean Distance = sqrt(sum i to N (x1_i – x2_i)^2)

Where *x1* is the first row of data, *x2* is the second row of data and *i* is the index to a specific column as we sum across all columns.

With Euclidean distance, the smaller the value, the more similar two records will be. A value of 0 means that there is no difference between two records.

Below is a function named *euclidean_distance()* that implements this in Python.

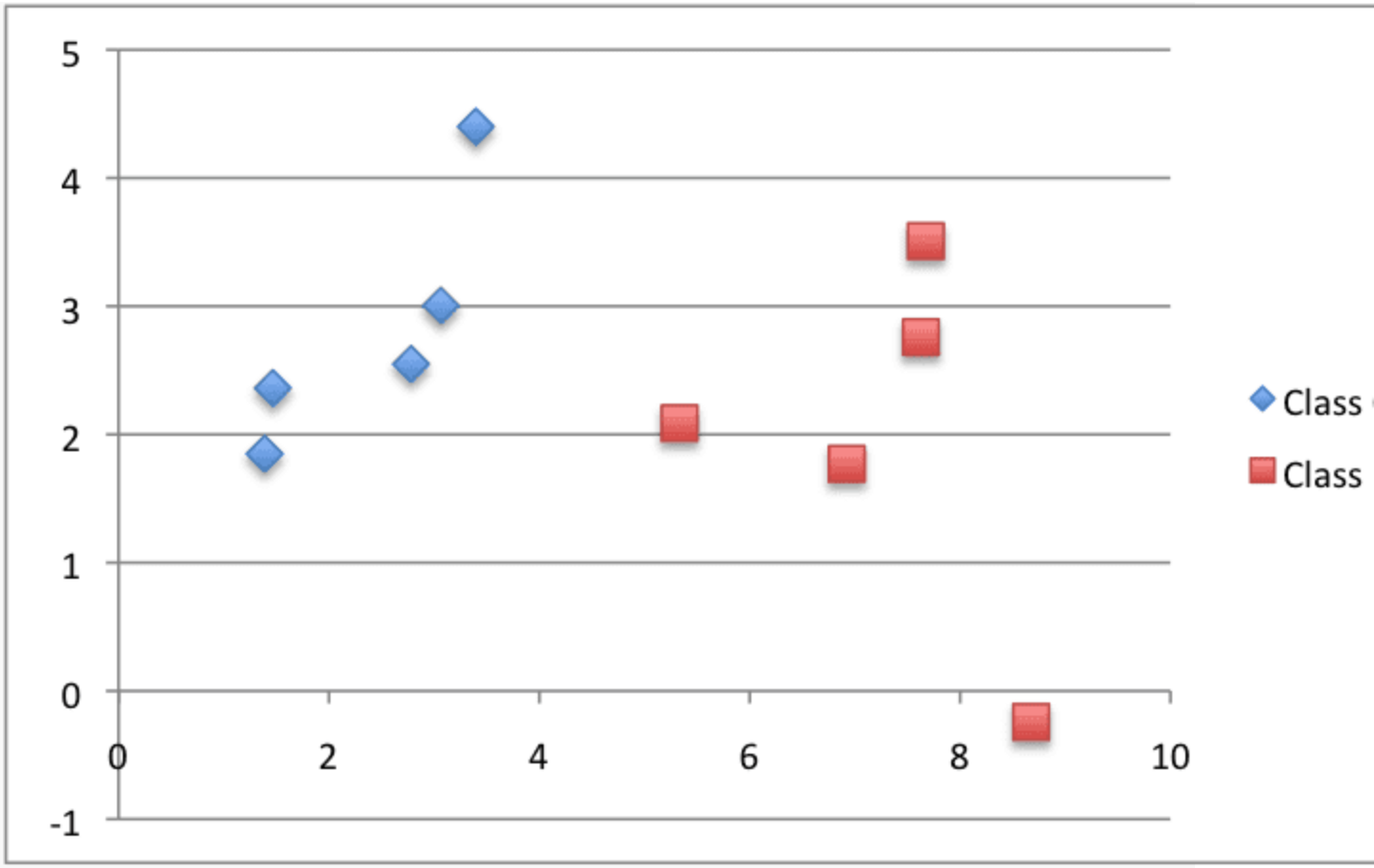
```
# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
distance = 0.0
for i in range(len(row1)-1):
 distance += (row1[i] - row2[i])**2
return sqrt(distance)
```

You can see that the function assumes that the last column in each row is an output value which is ignored from the distance calculation.

We can test this distance function with a small contrived classification dataset. We will use this sample dataset a few times as we construct the elements needed for the KNN algorithm.

	X1	X2	Y
2	2.7810836	2.550537003	0
3	1.465489372	2.362125076	0
4	3.396561688	4.400293529	0
5	1.38807019	1.850220317	0
6	3.06407232	3.005305973	0
7	7.627531214	2.759262235	1
8	5.332441248	2.088626775	1
9	6.922596716	1.77106367	1
10	8.675418651	-0.242068655	1
11	7.673756466	3.508563011	1

Below is a plot of the dataset using different colors to show the different classes for each point.



Scatter Plot of the Small Contrived Dataset for Testing the KNN Algorithm

Putting this all together, we can write a small example to test our distance function by printing the distance between the first row and all other rows. We would expect the distance between the first row and itself to be 0, a good thing to look out for.

The full example is listed below.

```
# Example of calculating Euclidean distance
from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
distance = 0.0
for i in range(len(row1)-1):
 distance += (row1[i] - row2[i])**2
return sqrt(distance)

# Test distance function
dataset = [[2.7810836,2.550537003,0],
```

```
13 [1.465489372,2.362125076,0],
14 [3.396561688,4.400293529,0],
15 [1.38807019,1.850220317,0],
16 [3.06407232,3.005305973,0],
17 [7.627531214,2.759262235,1],
18 [5.332441248,2.088626775,1],
19 [6.922596716,1.77106367,1],
20 [8.675418651,-0.242068655,1],
21 [7.673756466,3.508563011,1]]
22 row0 = dataset[0]
23 for row in dataset:
24 distance = euclidean_distance(row0, row)
25 print(distance)
```

Running this example prints the distances between the first row and every row in the dataset, including itself.

```
1 0.0
2 1.3290173915275787
3 1.9494646655653247
4 1.5591439385540549
5 0.5356280721938492
6 4.850940186986411
7 2.592833759950511
8 4.214227042632867
9 6.522409988228337
10 4.985585382449795
```

Now it is time to use the distance calculation to locate neighbors within a dataset.

Step 2: Get Nearest Neighbors

Neighbors for a new piece of data in the dataset are the *k* closest instances, as defined by our distance measure.

To locate the neighbors for a new piece of data within a dataset we must first calculate the distance between each record in the dataset to the new piece of data. We can do this using our distance function prepared above.

Once distances are calculated, we must sort all of the records in the training dataset by their distance to the new data. We can then select the top *k* to return as the most similar neighbors.

We can do this by keeping track of the distance for each record in the dataset as a tuple, sort the list of tuples by the distance (in descending order) and then retrieve the neighbors.

Below is a function named *get_neighbors()* that implements this.

```
# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
distances = list()
for train_row in train:
 dist = euclidean_distance(test_row, train_row)
 distances.append((train_row, dist))
distances.sort(key=lambda tup: tup[1])
neighbors = list()
for i in range(num_neighbors):
 neighbors.append(distances[i][0])
return neighbors
```

You can see that the *euclidean_distance()* function developed in the previous step is used to calculate the distance between each *train_row* and the new *test_row*.

The list of *train_row* and distance tuples is sorted where a custom key is used ensuring that the second item in the tuple (*tup[1]*) is used in the sorting operation.

Finally, a list of the *num_neighbors* most similar neighbors to *test_row* is returned.

We can test this function with the small contrived dataset prepared in the previous section.

The complete example is listed below.

```
# Example of getting neighbors for an instance
from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
distance = 0.0
for i in range(len(row1)-1):
 distance += (row1[i] - row2[i])**2
return sqrt(distance)

# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
distances = list()
for train_row in train:
 dist = euclidean_distance(test_row, train_row)
 distances.append((train_row, dist))
distances.sort(key=lambda tup: tup[1])
neighbors = list()
for i in range(num_neighbors):
 neighbors.append(distances[i][0])
```

```
21          return neighbors
22
23          # Test distance function
24          dataset = [[2.7810836,2.550537003,0],
25          [1.465489372,2.362125076,0],
26          [3.396561688,4.400293529,0],
27          [1.38807019,1.850220317,0],
28          [3.06407232,3.005305973,0],
29          [7.627531214,2.759262235,1],
30          [5.332441248,2.088626775,1],
31          [6.922596716,1.77106367,1],
32          [8.675418651,-0.242068655,1],
33          [7.673756466,3.508563011,1]]
34          neighbors = get_neighbors(dataset, dataset[0], 3)
35          for neighbor in neighbors:
36          print(neighbor)
```

Running this example prints the 3 most similar records in the dataset to the first record, in order of similarity.

As expected, the first record is the most similar to itself and is at the top of the list.

```
1       [2.7810836, 2.550537003, 0]
2       [3.06407232, 3.005305973, 0]
3       [1.465489372, 2.362125076, 0]
```

Now that we know how to get neighbors from the dataset, we can use them to make predictions.

Step 3: Make Predictions

The most similar neighbors collected from the training dataset can be used to make predictions.

In the case of classification, we can return the most represented class among the neighbors.

We can achieve this by performing the *max()* function on the list of output values from the neighbors. Given a list of class values observed in the neighbors, the *max()* function takes a set of unique class values and calls the count on the list of class values for each class value in the set.
Below is the function named *predict_classification()* that implements this.

```
1       # Make a classification prediction with neighbors
2       def predict_classification(train, test_row, num_neighbors):
3       neighbors = get_neighbors(train, test_row, num_neighbors)
4       output_values = [row[-1] for row in neighbors]
5       prediction = max(set(output_values), key=output_values.count)
```

```
6 return prediction
```

We can test this function on the above contrived dataset.

Below is a complete example.

```
1  # Example of making predictions
2  from math import sqrt
3
4  # calculate the Euclidean distance between two vectors
5  def euclidean_distance(row1, row2):
6  distance = 0.0
7  for i in range(len(row1)-1):
8   distance += (row1[i] - row2[i])**2
9  return sqrt(distance)
10
11 # Locate the most similar neighbors
12 def get_neighbors(train, test_row, num_neighbors):
13 distances = list()
14 for train_row in train:
15  dist = euclidean_distance(test_row, train_row)
16  distances.append((train_row, dist))
17 distances.sort(key=lambda tup: tup[1])
18 neighbors = list()
19 for i in range(num_neighbors):
20  neighbors.append(distances[i][0])
21 return neighbors
22
23 # Make a classification prediction with neighbors
24 def predict_classification(train, test_row, num_neighbors):
25 neighbors = get_neighbors(train, test_row, num_neighbors)
26 output_values = [row[-1] for row in neighbors]
27 prediction = max(set(output_values), key=output_values.count)
28 return prediction
29
30 # Test distance function
31 dataset = [[2.7810836,2.550537003,0],
32 [1.465489372,2.362125076,0],
33 [3.396561688,4.400293529,0],
34 [1.38807019,1.850220317,0],
35 [3.06407232,3.005305973,0],
36 [7.627531214,2.759262235,1],
37 [5.332441248,2.088626775,1],
```

```
38          [6.922596716,1.77106367,1],
39          [8.675418651,-0.242068655,1],
40          [7.673756466,3.508563011,1]]
41          prediction = predict_classification(dataset, dataset[0], 3)
42          print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

Running this example prints the expected classification of 0 and the actual classification predicted from the 3 most similar neighbors in the dataset.

```
1           Expected 0, Got 0.
```

We can imagine how the *predict_classification()* function can be changed to calculate the mean value of the outcome values.

We now have all of the pieces to make predictions with KNN. We can apply it to a real dataset.

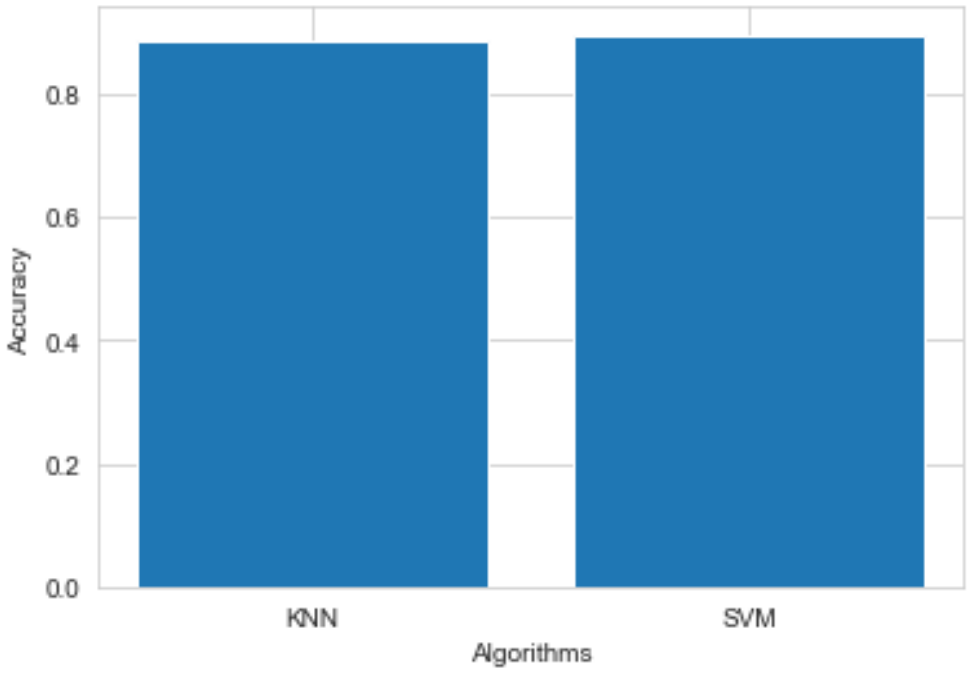
CHAPTER 6

RESULTS

- When we applied SVM (support vector machine) to our dataset we got **Accuracy -82%**

- When we applied KNN (k nearest neighbour) to our dataset we got **Accuray-88.3%**

Hence , on comparing both we came to know that accuracy of KNN is better and good than SVM.



- **So, K Nearest Neighbour (KNN) will classify lung cancer in early stage more accurately than SVM so that many lives can be saving.**

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

The objective of our model is to predict and classify lung cancer in early stages. For this we have applied two classification algorithm SVM and KNN on Common platform or common dataset to compare their accuracy ratio.

We have found that accuracy of KNN is more than the SVM , so we can say that KNN will classify more precisely and accurately lung cancer in early stages. The models were able to retain accurate prediction even with a very small number of features selected.

This is the most effective model to predict patients with lung cancer disease. This project could answer complex queries, each with its own strength with respect to ease of model interpretation, access to detailed information and accuracy.

7.2 Future scope

In future the advanced level of algorithm is used to increase the level of prediction while we are in process to include two more Classification Algorithm the to use the data set more effectively.

We are also planning to make our model efficient and more accurate by using feature selection method

References:

- P. Chaudhari, H. Agarwal, and V. Bhateja, “Data augmentation for cancer classification in oncogenomics: an improved KNN based approach,” Evol. Intell., pp. 1–10, 2019.
- [2] S. F. Khorshid and A. M. Abdulazeez, “BREAST CANCER DIAGNOSIS BASED ON K-NEAREST NEIGHBORS: A REVIEW,” PalArch’s J. Archaeol. Egypt/Egyptology, vol. 18, no. 4, pp. 1927–1951, 2021.
- [3] F. Q. Kareem and A. M. Abdulazeez, “Ultrasound Medical Images Classification Based on Deep Learning Algorithms: A Review.”
- [4] D. Q. Zeebaree, A. M. Abdulazeez, D. A. Zebari, H. Haron, and H. N. A. Hamed, “Multi-Level Fusion in Ultrasound for Cancer Detection Based on Uniform LBP Features.”
- [5] J. R. F. Junior, M. Koenigkam-Santos, F. E. G. Cipriano, A. T. Fabro, and P. M. de Azevedo-Marques, “Radiomics-based features for pattern recognition of lung cancer histopathology and
- metastases,” Comput. Methods Programs Biomed., vol. 159, pp. 23–30, 2018.
- [6] I. Ibrahim and A. Abdulazeez, “The Role of Machine Learning Algorithms for Diagnosing Diseases,” J. Appl. Sci. Technol. Trends, vol. 2, no. 01, pp. 10–19, 2021.
- [7] P. Das, B. Das, and H. S. Dutta, “Prediction of Lungs Cancer Using Machine Learning,” EasyChair, 2020.
- [8] G. A. P. Singh and P. K. Gupta, “Performance analysis of various machine learning-based approaches for detection and classification of lung cancer in humans,” Neural Comput. Appl., vol. 31, no. 10, pp. 6863–6877, 2019.
- [9] B. Charbuty and A. Abdulazeez, “Classification Based on Decision Tree Algorithm for Machine Learning,” J. Appl. Sci. Technol. Trends, vol. 2, no. 01, pp. 20–28, 2021.
- [10] H. A. Hussein and A. M. Abdulazeez, “COVID-19 PANDEMIC DATASETS BASED ON MACHINE L

- N.Maleki , Y.Zeinali , S.T.A.Niaki (2021) “A K-NN method for lung cancer prognosis with the use of a genetic algorithm for feature selection ”, Expert Systems with application ,vol. 164, pp:1-7.

- Dr. S. Senthil , B. Ayshwarya (2018) “Lung Cancer Prediction using Feed Forward Back Propagation Neural Networks with Optimal Features”, International Journal of Applied Engineering Research ISSN 0973-4562, Volume 13, pp : 1-8.
- P.Chaturvedi , A.Jhamb , M.Vanani1 and V.Nemade1 (2021) “Prediction and Classification of Lung Cancer Using Machine Learning Techniques”, Materials Science and Engineering , pp : 1-20.
- P.Bhuvaneswari , Dr.A.Brintha Therese (2015) “Detection of Cancer in Lung with K-NN Classification Using Genetic Algorithm”, Procedia Materials Science 10:433-440 , pp: 1-10
- D.M.Abdullah , A.M.Abdulazeez , A.B.Sallow (2021) “Lung cancer Prediction and Classification based on Correlation Selection method Using Machine Learning Techniques” , pp : 1-9.