Predictive Model to determine life expectancy post-surgery

Submitted in partial fulfillment of the requirements of the degree of

B.E. Computer Engineering

By

Palash Anjania	**60004160005**
Krina Bhimani	**60004160010**
Shrushti Gangar	**60004160030**

Guide:
Prof. Sindhu Nair
Assistant Professor
Department of Computer Engineering

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

University of Mumbai
2019-2020

CERTIFICATE

This is to certify that the project entitled **“Predictive Model to determine life expectancy post surgery”** is a bonafide work of “**Palash Anjania” (60004160005) , “Krina Bhimani” (60004160010) and “Shrushti Gangar” (60004160030)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of B.E. in Computer Engineering.

Prof. Sindhu Nair

Assistant Professor

Dr. Meera Narvekar	**Dr. Hari Vasudevan**
Head of Department	**Principal**

Project Report Approval for B. E.

This project report entitled '**Predictive Model to determine life expectancy post surgery'** by **Palash Anjania, Krina Bhimani** and **Shrushti Gangar** is approved for the partial fulfilment of the degree of ***B.E. in Computer Engineering.***

Examiners

1. ________________

2. ________________

Date:

Place:

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

____

Palash Anjania (60004160005)

____

Krina Bhimani (60004160010)

____

Shrushti Gangar (60004160030)

Date: ____________

Abstract

Monitoring health outcomes is essential to enhance quality initiatives, healthcare management and consumer education. These days cancer is one of the major causes of death in most countries. Currently, lung cancer is the most frequent augury for thoracic surgery. Thoracic Surgery is the data collected of patients who underwent major lung resections for primary lung cancer. Postoperative respiratory complications are the most common fatal following any type of thoracic surgery. One particular metric that has been used to quantify mortality rates in the past has been the thirty-day mortality rate, which has been implemented traditionally using basic Machine Learning algorithms. This metric, however, may not be entirely comprehensive because many patients die shortly after this time period or become very weak, or have to be taken to another facility before passing away there. As a result, many of these deaths are severely underreported. This project aims at using the ensemble machine learning algorithms for discovery and identification of models and relationships between them, from large datasets, the data is analysed to extract useful information by reducing the dimensionality that supports disease augury, and to improve models that predict patient's health more precisely and accurately. Initially, the dataset consists of continuous data as well as classification data. The classification predicts whether the patient survived the following year-long period or not. If the patient survives for more than one year it is a true condition and if the patient dies within a span of one year then it proves to be a false condition as per the model. Initially, we use feature selection methods like Univariate Feature Selection and Recursive Feature Elimination techniques to select the best feature inorder to train the data. The next step consists of applying various ensemble machine learning algorithms like Gradient Boosting Method and Random Forest Method in order to train the data. Later analysis accuracy testing is performed. Once the outcomes are obtained, these results play a crucial role and might have large implications in the medical field. The scope of our project is to examine the mortality of patients within a full year after the surgery after considering various attributes. More specifically, we are examining the underlying health factors of patients that could potentially prove to be a powerful predictor for surgically related deaths post operations.

Contents

List of Figures

List of Tables

List of Abbreviations

Sr. No.	Abbreviation	Expanded form
i	XGB	Extreme Gradient Boosting
ii	RFE	Recursive Feature Elimination
iii	SVM	Support Vector Machine
iv	SU	Symmetrical Uncertainty
v	IG	Information Gain
vi	RF	Relief Factor
vii	DFD	Data Flow Diagram
viii	EDA	Exploratory Data Analysis
viii	GUI	Graphic User Interface

` Chapter 1 Introduction

1.1. Description

Initially, the field of medicine was not very advanced, it was not easy to treat everything that the patient was diagnosed with. The hospitals had a limited amount of resources to understand and treat the illness. The medical field has developed a lot in recent years, from predicting the trickiest diseases to treating them in the most critical stages, all is now possible. Due to the facilities available in today's technically advanced world the death rate has been reduced by a significant margin.

Even in today's world with all the technological developments, one of the most difficult parts in this field of life-saving is to predict the after-effects of surgeries and other medical procedures. The doctors can never be sure of the complications that can take place after the operation. These complications can sometimes be fatal. To predict these postoperative problems has been a topic of great interest for doctors and scientists.

The project focuses on the repercussions of thoracic surgery. The main focus of the project is to find the life expectancy of a patient post-surgery. By using various parameters the model is trained to predict whether the patient will live for a year after the surgery. We use machine learning to train the model to make decisions on whether there will be any problems post-operation or not.

1.2. Problem Formulation

India gets a lot of surgeries every day. Looking after the patients while their treatment is going on when they are admitted to the hospital is a complex task as it is. On top of that doctors have to make sure of the problems that might occur after an operation. Preventing the occurrence of these problems is not an easy task.

Every patient's risk is assessed based on the problems that might occur within 30 days of the surgery. This time period is very little time to observe any big changes that may happen post-surgery in the patient's body. These changes may be fatal to the patient. If within the 30 day time period there are no complications then the patient is declared free of risk. But the problem is that the patient may develop complications after the 30-day time period.

This can lead to false promises and bad faith among people. This study aims to provide a solution to alleviate these problems by replacing this 30-day mortality rate by a year-long mortality rate.

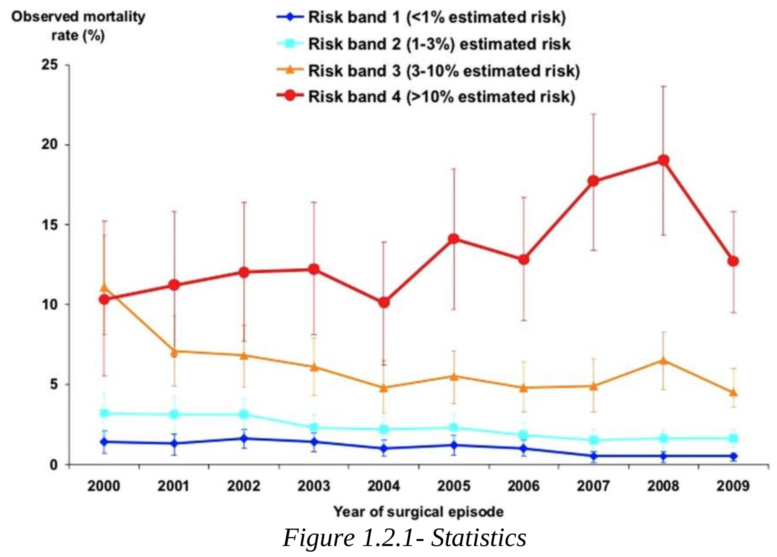


Figure 1.2.1- Statistics

1.3. Motivation

We deeply feel the plight of the people who have faced consequences due to the drawbacks of the currently used method.

There have been people who have been assessed with no risk within the 30-day period but have faced deaths after a month or so. Such things develop bad faith towards doctors and hospitals. This also affects the mental health of the patient's families.

There are situations where the system may show risk but it actually may not be there. Such anomalies cause a surge of tension and panic among people and also make them spend money where it is not needed. Such issues must be resolved. Sometimes a doctor's mistake or misjudgment would cause trouble or panic to the patient and his/her family. Such troubles in a life or death situation could be nerve-racking and such pressure would eventually end up deteriorating the patient's health.

The 30-day time period is not sufficient enough to make a correct prediction of the patients' lives. Complex diseases may show anomalous symptoms a little late and hence more than 30 days are required to understand the situation better.

Due to a small period of consideration i.e. only 30 days even the doctor is tensed that the symptoms may come back. To prevent this, the doctors give medicine to patients which may have a few side-effects. These side-effects not only trouble the patients and their families but the doctors are troubled too.

1.4. Proposed Solution

We aim to develop a system to provide ease in predicting the life expectancy of the patient and reduce the pressure on doctors and hospitals. Our solution will not only help the doctors but also prepare the patients for the upcoming issues.

This model will help the patients as they are the victims of these inconsistencies. The aim is to provide an efficient system to reduce inconsistency and help make logical decisions.

Our solution uses machine learning algorithms to predict the output. The dataset contains a number of tuples with different factors where the last factor is the desired output. After pre-processing of the dataset a couple of feature selection techniques are used to pick the most relevant factors contributing to accurate prediction. Then various machine learning algorithms are used to train the model which is then tested to find out which algorithm gives the most accurate solution.

The steps of the flow diagram are explained below.

A. Data Collection

Data has been collected from various sources to create a wider range of cases to work with and increase the efficiency of the model. An initial dataset was scraped from the University of California, Irvine repository. More data will be collected by contacting hospitals across Mumbai where thoracic surgery is treated.

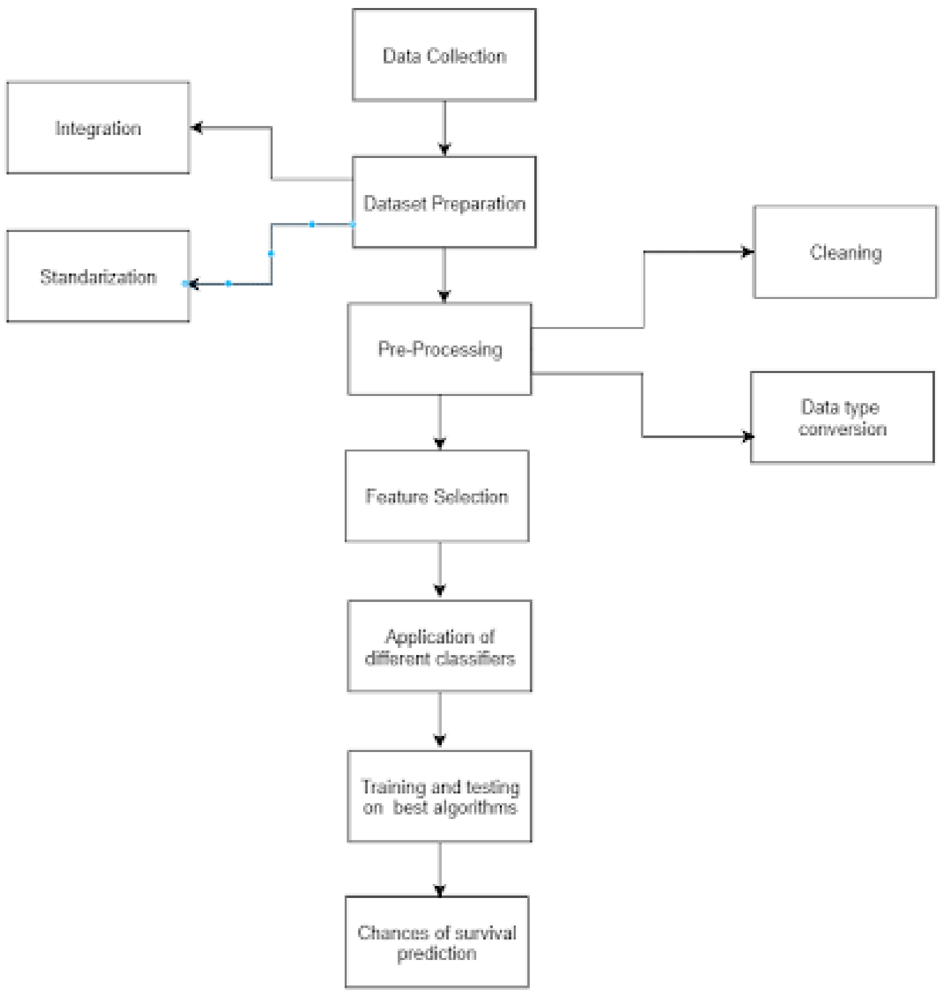


Fig. 1.4.1 - Flow Diagram

B. Data set Preparation

As the data has been collected from various sources, it is not in a general or standard format. This can create issues that will compute the data. Non-standardized data can lead to faulty results. So the first step in data preparation will be to convert data into a more general form.

The second step will include integrating the data properly from these different sources. Combining data into one set not only helps in better visualization of the data but also helps in pre-processing and metadata management.

Metadata contains where the data was taken from i.e. the source of data and the amount of data taken from a particular source. Although metadata may not play a role in prediction but it is good to keep track of data.

C. Pre-Processing

The data collected is not clean. it requires cleaning. In the cleaning process, the data is made complete and fit for use. The collected data has missing values which are filled or removed so as they do not cause problems in the training and testing stage.

Next, the data is converted in the required form like binary, boolean, varchar and numeric. This helps in processing data and evaluates results better and faster.

D. Feature Extraction

Feature extraction is a very important step in the development of a machine learning model. This helps to reduce irrelevant factors and create a compact yet powerful dataset.

Recursive Feature selection

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's coef_ or feature_importances_ attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model[8].

RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. To find the optimal number of features cross-validation is used with RFE to score different feature subsets and select the best scoring collection of features. The RFECV visualizer plots the number of features in the model along with their cross-validated test score and variability and visualizes the selected number of features[9].

E. Machine Learning Algorithms

Extreme Gradient Boosting(XGB)

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function[6].
XGBoost is one of the implementations of Gradient Boosting concept, but what makes XGBoost unique is that it uses "a more regularized model formalization to control over-fitting, which gives it better performance".

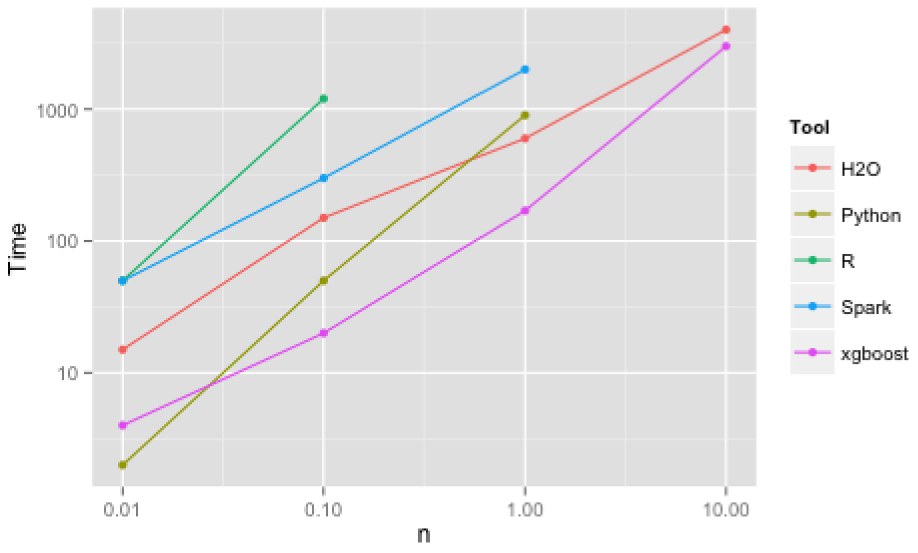


Figure 1.4.2-XGboost

It is built on the principles of gradient boosting framework and designed to “push the extreme of the computation limits of machines to provide a scalable, portable and accurate library[7].”

Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space, this hyperplane is a line dividing a plane into two parts wherein each class lay in either side[4].

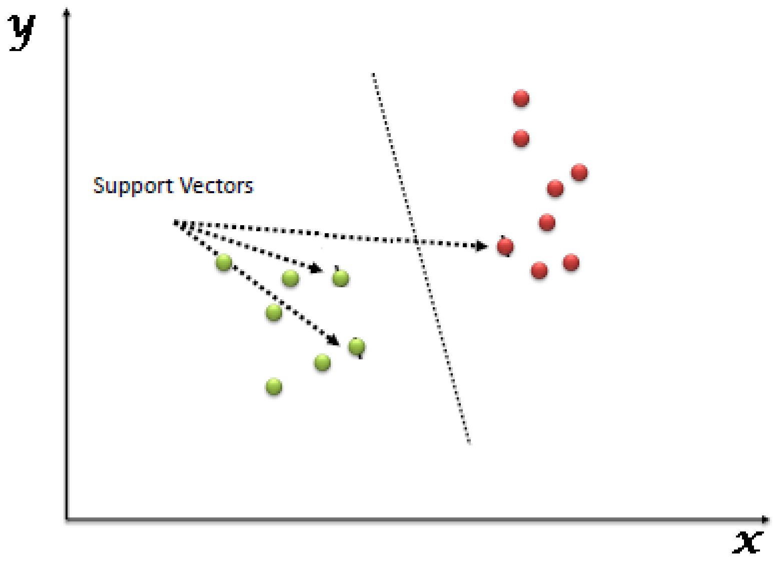


Figure 1.4.3- SVM

The objective of the support vector machine algorithm is to find a hyperplane in N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence[5].

Random Forest

Random Forests or Random decision forests are an ensemble learning method for classification, regression and other tasks that work by developing a huge number of choice trees at preparing time and yielding the class that is the method of the classes (order) or mean expectation (relapse) of the individual trees. Random decision forests improve for decision trees' propensity for overfitting to their preparation set.

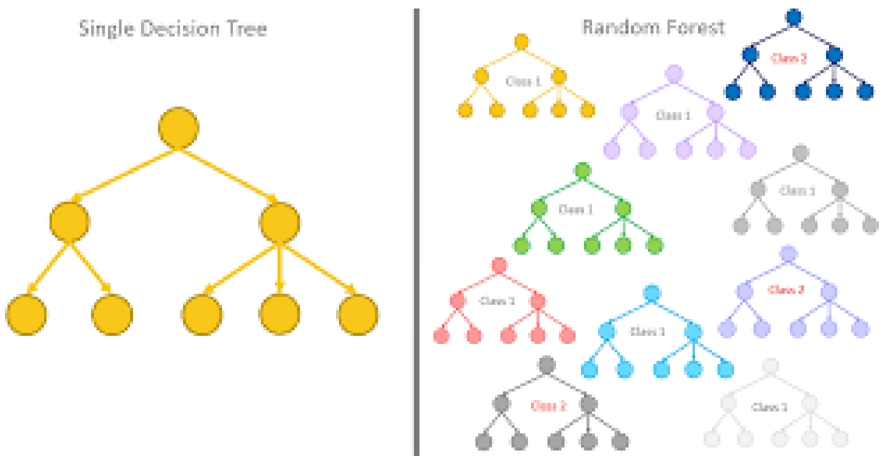


Fig 1.4.4 Random Forest

Data science gives plenty of ordered algorithms, for example, Logistic Regression, Support vector machine, Naive Bayes classifier, and Decision trees. In any case, close to the highest point of the classifier chain of command is the Random forest classifier.
To comprehend and utilize the different alternatives, additional data about how they are figured is helpful. The vast majority of the choices rely upon two information objects created by arbitrary woods. At the point when the preparation set for the present tree is drawn by testing with substitution, around 33% of the cases are kept separate from the example. This oob (out-of-bag) information is utilized to get a running fair gauge of the order blunder as trees are added to the woodland. It is additionally used to get assessments of variable significance. After each tree is assembled, the entirety of the information are run down the tree, and vicinities are processed for each pair of cases. In the event that two cases possess a similar terminal hub, their vicinity is expanded by one. Toward the finish of the run, the vicinities are standardized by isolating by the quantity of trees. Vicinities are utilized in supplanting missing information, finding exceptions, and creating lighting up low-dimensional perspectives on the information.

1.5. Scope of the project

The current scope of the project is to create an efficient system to predict the life expectancy of a patient post thoracic surgery.
Even though this system may not be able to completely solve the problem of predicting the post-surgery complications, it will make the prediction a lot easier. We aim to create a Machine learning model to predict the life expectancy of a patient after a year of thoracic surgery.
Future scope includes a system that can predict life expectancy for various diseases. It will be a system that will work in a similar way, that is, it will use a number of different factors for different diseases in the model to predict the life expectancy of the respective patients and diseases.

Chapter 2
Review of Literature

The following paragraphs describe the literature survey, covering the ideas in the different papers examined for the project.

Adam Abdulhamid, Ivaylo Bahtchevanov and Peng Jia[1] in their paper titled **"Life Expectancy Post Thoracic Surgery"** adopted various supervised machine learning techniques including Naive Bayes, SVM, and Logistic Regression in order to train the dataset and obtain testing values using k-folds cross-validation considering k equals to 10. In their research, they later applied the bootstrapping method on randomly sampled data in order to obtain a split in the dataset. Later, they trained 40 models and obtained 40 different predictions and then the average of them was taken in order to get the final prediction. Finally, datasets of different sizes were tried and tested in order to check its effect on accuracy and recall.

Abeer S. Desuky and Lamiaa M. El Bakrawy [2] in their paper titled **"Improved Prediction of Post-operative Life Expectancy after Thoracic Surgery "** have used three attribute ranking methods to reduce the number of attributes such as Symmetrical Uncertainty(SU), Information Gain(IG), Relief Factor(RF) and examined the quality of Machine learning algorithms like Naive Bayes, Simple Logistic, Multilayer Perceptron and J48 by comparing their performance with or without attribute ranking methods along with the boosted versions of the machine learning algorithms. The researchers have used ten-fold cross-validation technique to validate the results. Ten rounds of cross-validation are performed and in each round 2 through 10 are used as testing dataset.The validation dataset is averaged over the ten rounds in the final phase.

Information Gain (IG) Attribute Evaluation is used to evaluate the importance of an attribute by measuring the Information Gain with regard to the class. Symmetrical Uncertainty (SU) is used to compensate for the inherent bias in Information Gain. Relief Factor (RF) attribute evaluation is used to rank the quality of features depending on how well their values differ from the cases that are close to each other. Using these attribute ranking techniques the number of attributes was reduced from 16 to 13 attributes. The obtained results were later compared with Boosted versions of Naive Bayes, J48, Simple Logistic Regression, Perceptron, and SVM algorithms. The final experimental results showed that boosting is not always the better choice where attribute ranking and selection can perform better in predicting accuracy.

Kwetise Joro Danjuma [3] has adopted a different technique Synthetic Minority Over-Sampling Technique (SMOTE) in the research paper titled **"Performance Evaluation of Machine Learning Algorithms in Post-Operative Life Expectancy in the Lung Cancer Patients "** to generate synthetic minority samples to oversample the minority class and smoothen the sample distribution. It operates by interpolating between several minority class examples that lie together, using the concept of k-nearest neighbour to avoid overfitting causing the decision boundaries for the minority class to spread further into the majority class space. In this research paper, the thoracic datasets is dedicated to classification problem related to the post-operative life expectancy in lung cancer patients : class1 - death within one year after surgery, class 2 - survival; recoded as Risk1Y: 1 year survival period -(T)rue value if died (T,F), where the class is (Risk1Y) is binary-valued. Multilayer Perceptron, J48 and

Naive Bayes machine learning algorithms were calibrated to optimize the baseline performance accuracy of each classifier.

Also, stratified 10-fold cross-validation was used to measure the unbiased predictive accuracy of each classifier. The experimental results for stratified 10-fold cross-validation comparative analysis showed that Multilayer Perceptron achieved a classification accuracy of 82.3 %, J48 classifier achieved showed a classification accuracy of 81.8% and Naive Bayes classifier displayed a classification accuracy of 74.4%.

Chapter 3
System Analysis

3.1. Functional Requirements

- **Maintaining records**
  The system should be able to maintain records and data of the patients with all the factors and tuples.
- **Pre-processing**
  The data received has to be modified for effective use. The process begins with filling the empty spaces if any. The next step is to convert the data into the required form. the data is converted to binary, boolean, numerical as required. The third and final step is to integrate the data collected.
- **Managing authenticity, reliability, and integrity**
  The system should be capable enough to maintain the authenticity, reliability, and integrity of the dataset. it should be able to do this regardless of the maintenance activities or other user actions.
- **Feature selection**
  The system should be efficient enough to select the required feature and eliminate irrelevant or useless features.
  The system uses a recursive feature selection technique to achieve the goal of reducing the factors to relevant ones.
- **Displaying results**
  The system is expected to display results efficiently without any difficulty or error in the display. The final output should be clear and precise, not ambiguous.
  The display should be clear to read and understand.

3.2. Non-Functional Requirement

- **Performance**

In the above-mentioned system, functionality plays a very important role. It tells whether the patient will survive post-surgery or not within a span of a year. The criticality of these functions tells us the performance and accuracy of the model. The model should perform well when the patient’s history is huge and a large number of features are given. Thus, the system should be such that it works efficiently in load conditions.

- **Accuracy**

The features selected must be able to predict accurate or nearly accurate results which might prove to be reliable in the future.

- **Reliability**

The model should be able to produce reliable output even in the conditions where the data is not complete or a little faulty.

3.3. Specific Requirements

The application reads categorical as well as boolean values. Hence, to process the information provided there are certain system requirements to be met which shall ensure smooth and efficient performance.

1. Software Requirements

- Python
- Anaconda
- Google Colab
- Operating System: Windows 10
- IDE: Jupyter/Spyder

2. Hardware Requirements

- RAM: 4GB and Above
- Intel Core i5 processor
- Laptop

Chapter 4
Analysis Modelling

4.1. Functional Modelling

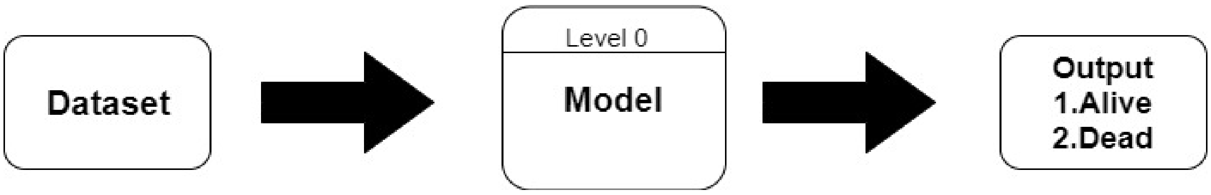


Figure 4.1.1- DFD-level 0

Level 0:

The context diagram shows a basic data flow.
In our project, the main goal is, by providing dataset as input to the model, the output will be alive or dead.

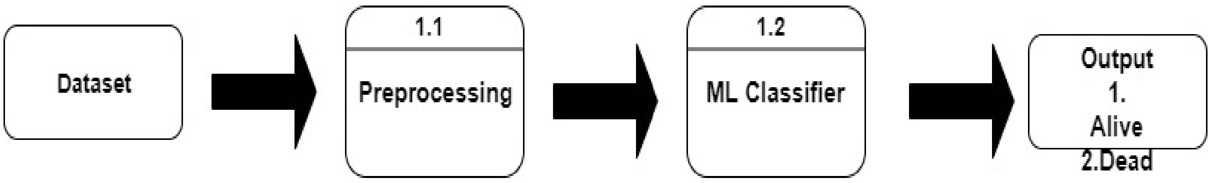


Figure 4.1.2- DFD-level 1

Level 1:

A general overview but may into more detail than context level. The particular level includes the preprocessing and decision phase as more detailed content.
Preprocessing uses different data standardization and normalization steps and ML is used to make a judgment as alive or dead.

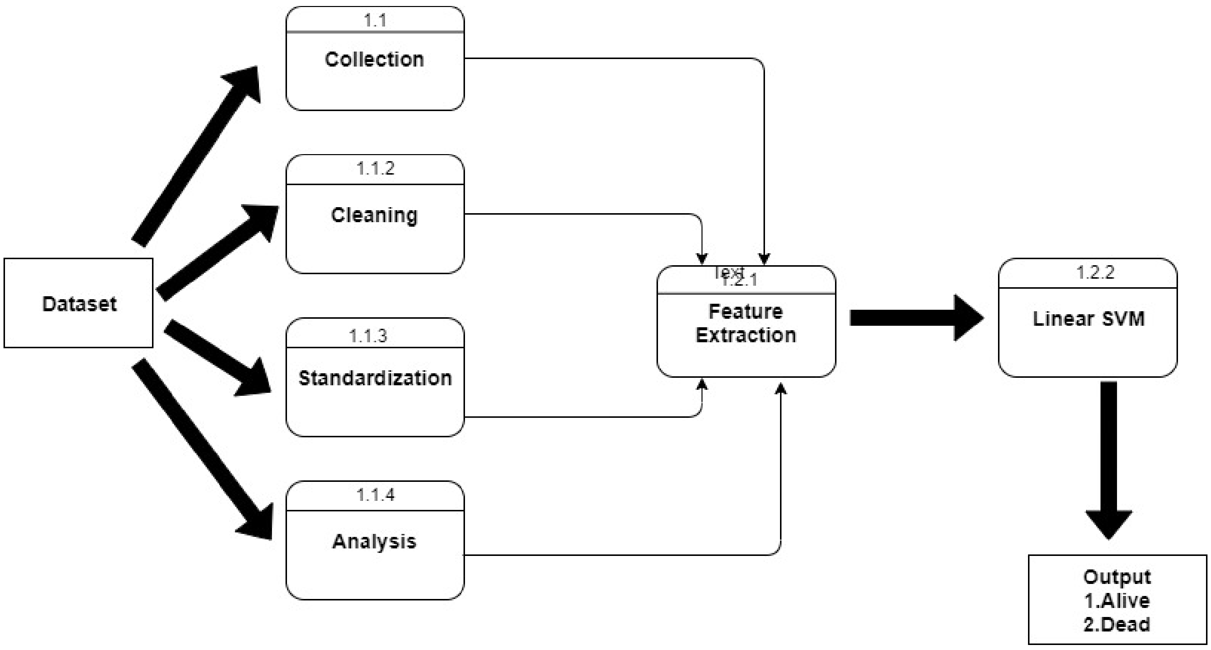


Figure 4.1.3- DFD-level 2

Level 2:

Simply break processes into more detailed sub-processes.

Data Preprocessing:

- The gathering of a dataset is done with the respective features required for training the model.
- Cleaning is getting rid of less useful parts that are not required for further process.
- Standardization is basically normalizing data, converting boolean values to an integer, extracting an integer from string values and having a range for a feature called age.
- The analysis is statistical probing, analyzing the data set by plotting a scatter plot and a different hypothesis.

Once the analysis of input is done, the feature selection method is applied to extract features and ML algorithms are applied to determine output as alive or dead.

4.2. Timeline Chart

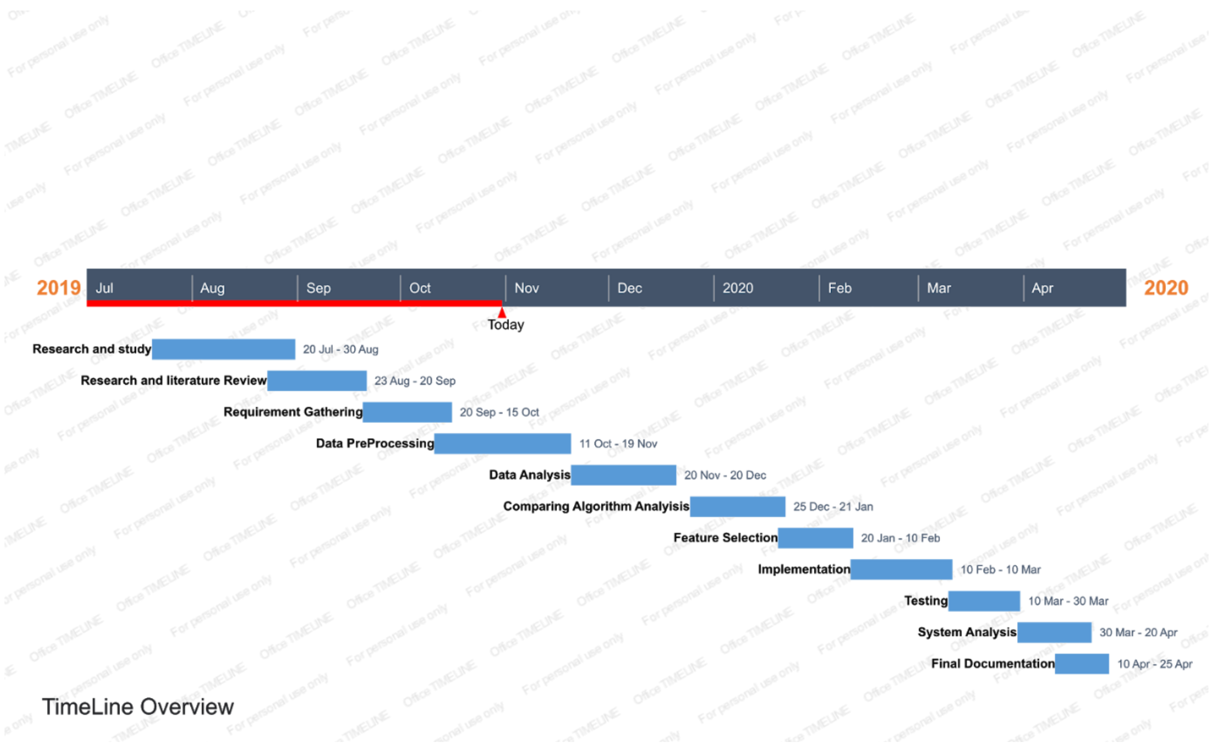


Figure 4.2.1- Timeline

Chapter 5

Design

5.1. Architectural Design

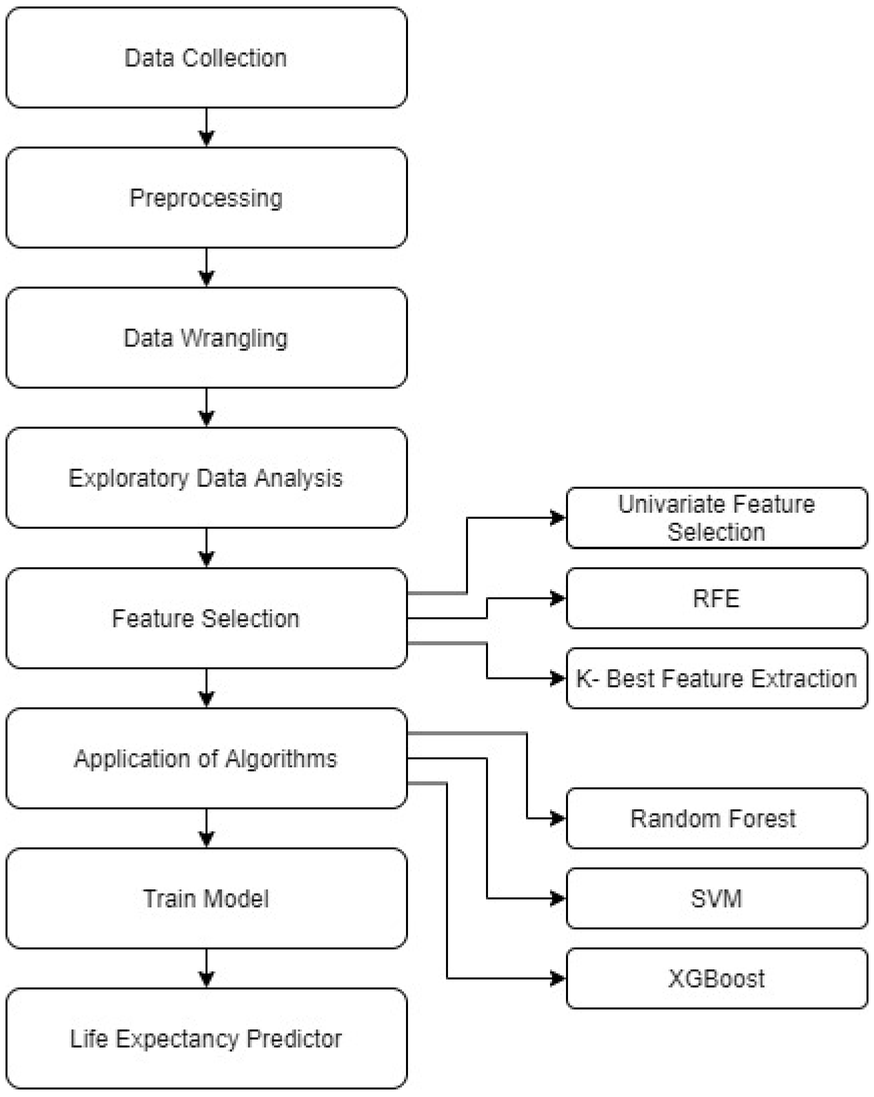
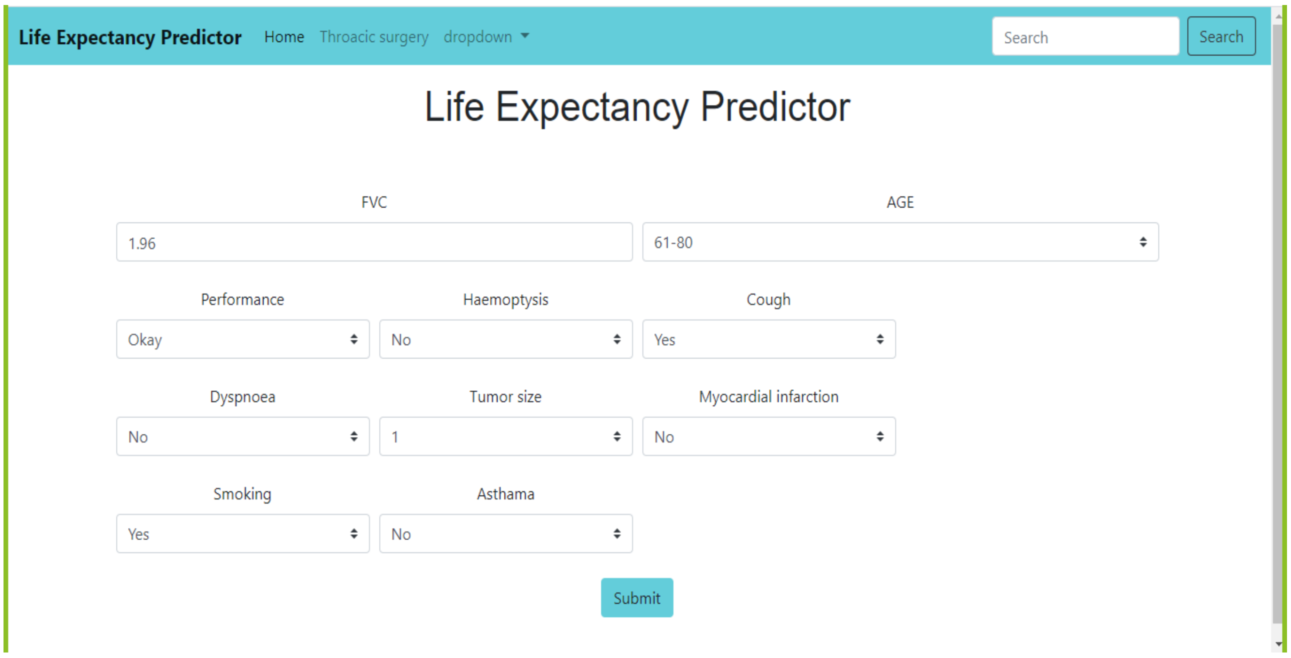

Data Collection
Preprocessing
Data Wrangling
Exploratory Data Analysis
Feature Selection
Univariate Feature Selection
RFE
K- Best Feature Extraction
Application of Algorithms
Random Forest
SVM
XGBoost
Train Model
Life Expectancy Predictor

Fig 5.1 - Architecture Design

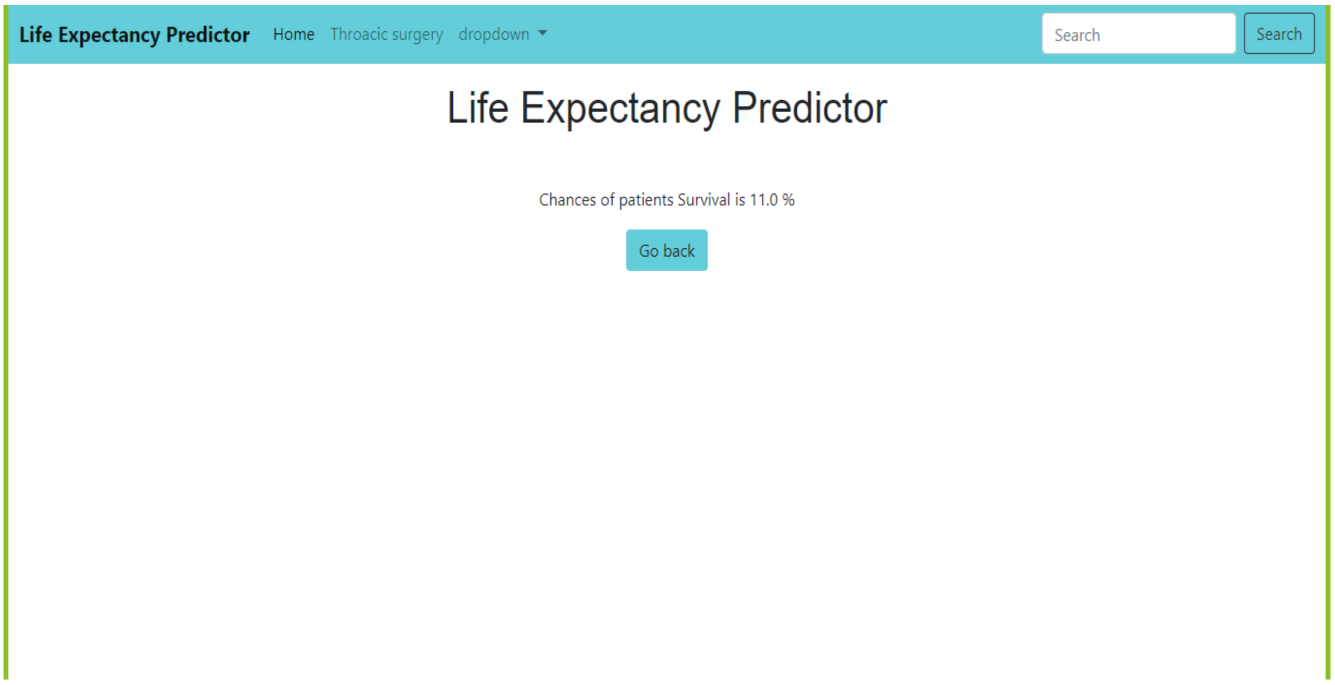
5.2. User Interface Design



Life Expectancy Predictor

FVC	1.96
AGE	61-80
Performance	Okay
Haemoptysis	No
Cough	Yes
Dyspnoea	No
Tumor size	1
Myocardial infarction	No
Smoking	Yes
Asthama	No

Fig 5.2.1 - GUI for Attribute Selection



Life Expectancy Predictor

Chances of patients Survival is 11.0 %

Fig 5.2.2 - GUI of Expectancy Predictor

Chapter 6
Implementation

6.1. Data Preprocessing

- Initially, the dataset from the respective repository was WEKA ARFF format, so the first step was to convert in CSV format.
- There were no missing values or NaN values present in the dataset.
- For numerical columns, we need to consider outliers and thus plot scatter plots.

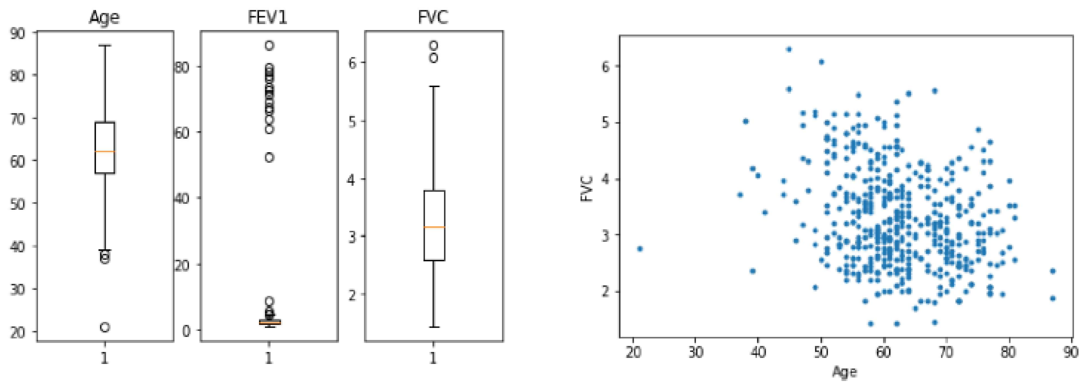


Fig 6.1 - Data Preprocessing

6.2. Data Wrangling

- As the dataset has a mixture of categorical, numerical and boolean values our step was to standardized them.
- By observation, there are 11 columns namely PRE7, PRE8, PRE9, PRE10, PRE11,PRE17, PRE19, PRE25, PRE30, PRE32 and Risk1Yr in the dataset that has boolean values T or F. So convert these values into 1 and 0 respectively.
- Further the diagnosis column has specific combinations of ICD-10 codes for multiple tumours which is represented in the form string with numerical values attached. So to make it an easy evaluation, consider only numeric values of these ICD-10 codes.
- The tumour size column has values from OC11 (smallest) to OC14 (largest) and considering only numeric values.
- Moreover, consider only numeric values for the performance status follows zubriod scale which ranges from PRZ0(good) to PRZ2 (poor).
- Lastly, the Age column includes varied values so for further ease, different ranges are specified like 0-20, 21-40, 41-60, 61-80 and 80 above.

6.3. Exploratory Data Analysis

In this section we explain the dataset and understand its complexity. We plot graphs to understand the data better and use it more efficiently.

1. Dataset Information

Fifty percent of the data was collected retrospectively at Wroclaw Thoracic Surgery Centre for patients who underwent major lung resections for lung cancer. The Centre is related with the Department of Thoracic Surgery of the Medical University of Wroclaw and Lower-Silesian Centre for Pulmonary Diseases, Poland, while the research database consists a section of the National Lung Cancer Registry, administered by the Institute of Tuberculosis and Pulmonary Diseases in Warsaw, Poland. Additional sources of the data include the Kaggle repository and some dummy data created for understanding the process.

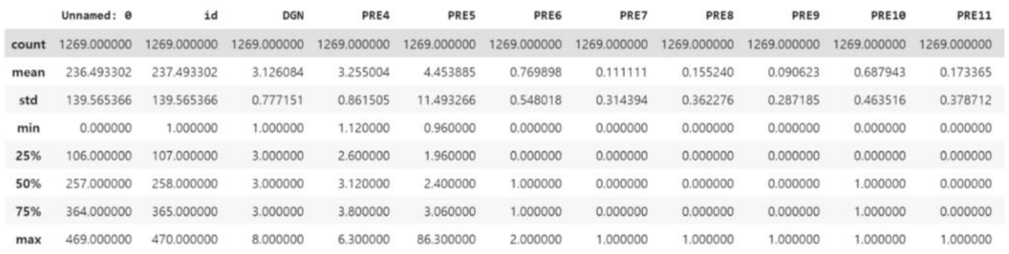
Attribute	Description
Diagnosis	ICD-10 codes for primary and secondary as well multiple tumors if any
FVC	Amount of air which can be forcibly exhaled from the lungs after taking the deepest breath possible
FEV1	Volume that has been exhaled at the end of the first second of forced expiration
Performance	Performance status on Zubrod scale, Good (0) to Poor (2)
Pain	Pain before surgery (T = 1, F = 0)
Haemoptysis	Coughing up blood, before surgery (T = 1, F = 0)
Dyspnoea	Difficulty or labored breathing, before surgery (T = 1, F = 0)
Cough	Symptoms of Coughing, before surgery (T = 1, F = 0)
Weakness	Weakness, before surgery (T = 1, F = 0)
Tumor_Size	T in clinical TNM - size of the original tumor, 1 (smallest) to 4 (largest)
Diabetes_Mellitus	Type 2 diabetes mellitus (T = 1, F = 0)
MI_6mo	Myocardial infarction (Heart Attack), up to 6 months prior(T = 1, F = 0)
PAD	Peripheral arterial diseases (T = 1, F = 0)

Smoking	Patient smoked (T = 1, F = 0)
Asthma	Patient has asthma (T = 1, F = 0)
Age	Age at surgery
Death_1yr	1 year survival period - (T) value if died (T = 1, F = 0)

Table - 1: Attributes

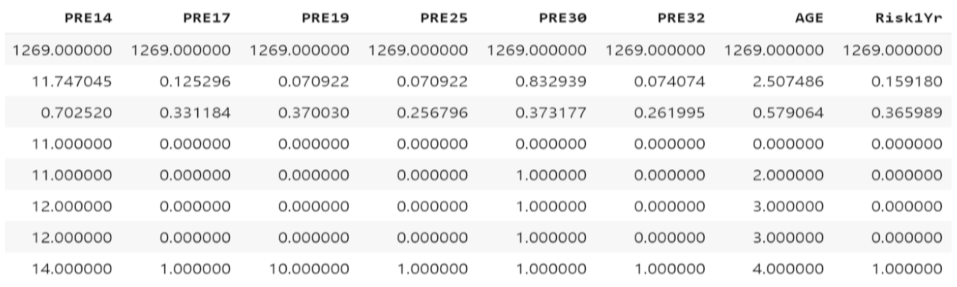
2. Data description

The dataset contains 16 attributes Which contribute to the final decision of the model. Each of these attributes have been explained above. For ease of handling the data, the headings have been changed. This helps to code and manage the data efficiently. The description of the data can be seen using the code 'df.describe'. The output has been displayed below:



	Unnamed: 0	id	DGN	PRE4	PRE5	PRE6	PRE7	PRE8	PRE9	PRE10	PRE11
count	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000
mean	236.493302	237.493302	3.126084	3.255004	4.453885	0.769898	0.111111	0.155240	0.090623	0.687943	0.173365
std	139.565366	139.565366	0.777151	0.861505	11.493266	0.548018	0.314394	0.362276	0.287185	0.463516	0.378712
min	0.000000	1.000000	1.000000	1.120000	0.960000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	106.000000	107.000000	3.000000	2.600000	1.960000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	257.000000	258.000000	3.000000	3.120000	2.400000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000
75%	364.000000	365.000000	3.000000	3.800000	3.060000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000
max	469.000000	470.000000	8.000000	6.300000	86.300000	2.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Fig 6.3.1 - df.describe() function



PRE14	PRE17	PRE19	PRE25	PRE30	PRE32	AGE	Risk1Yr
1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000	1269.000000
11.747045	0.125296	0.070922	0.070922	0.832939	0.074074	2.507486	0.159180
0.702520	0.331184	0.370030	0.256796	0.373177	0.261995	0.579064	0.365989
11.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
11.000000	0.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
12.000000	0.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
12.000000	0.000000	0.000000	0.000000	1.000000	0.000000	3.000000	0.000000
14.000000	1.000000	10.000000	1.000000	1.000000	1.000000	4.000000	1.000000

Fig 6.3.2 - df.head() function

3. How many patients died in a year ?

Out of the 1270 tuples of data, 203 people have died within a period of 1 year, this accounts for 16% of the total number of patients approximately. Here is the average of attribute values for a patient that lived and died within a year.

Attribute	**Live 1yr Mean**	**Death 1yr Mean**
PRE4	3.265370	3.200248
PRE5	4.712362	3.088564
PRE6	0.741331	0.920792
PRE7	0.101218	0.163366
PRE8	0.150890	0.178218
PRE9	0.083411	0.128713
PRE10	0.666354	0.801980
PRE11	0.153702	0.277228
PRE14	11.701031	11.990099
PRE17	0.106842	0.222772
PRE19	0.073102	0.059406
PRE25	0.064667	0.103960
PRE30	0.820993	0.896040
PRE32	0.074039	0.074257
AGE	2.492971	2.584158

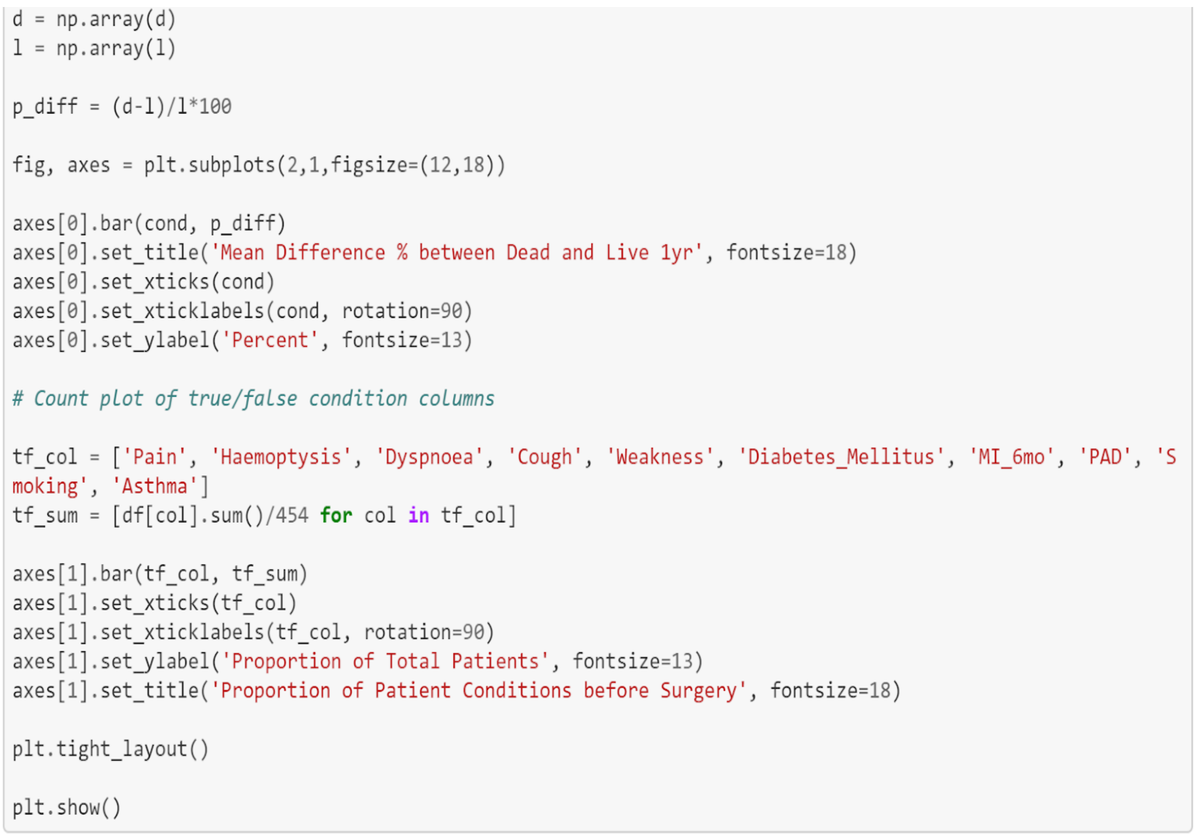
Table - 2: Live and Death 1yr Mean

4. What are the differences between 1 year death patients?

The differences between the attributes of the patients that lived post a year and that did not give us vital information on how and what affects these deaths. The following code and output snippets display various graphs and charts to understand the data better.

A. Mean Difference % between Dead and Live 1yr and Proportion of Patient Conditions before Surgery.

Just looking at the numbers without scaling them appropriately to each other makes comparison difficult. So, let's do an approximate normalization of each value for convenient comparison.



```
d = np.array(d)
l = np.array(l)

p_diff = (d-l)/l*100

fig, axes = plt.subplots(2,1,figsize=(12,18))

axes[0].bar(cond, p_diff)
axes[0].set_title('Mean Difference % between Dead and Live 1yr', fontsize=18)
axes[0].set_xticks(cond)
axes[0].set_xticklabels(cond, rotation=90)
axes[0].set_ylabel('Percent', fontsize=13)

# Count plot of true/false condition columns

tf_col = ['Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'S
moking', 'Asthma']
tf_sum = [df[col].sum()/454 for col in tf_col]

axes[1].bar(tf_col, tf_sum)
axes[1].set_xticks(tf_col)
axes[1].set_xticklabels(tf_col, rotation=90)
axes[1].set_ylabel('Proportion of Total Patients', fontsize=13)
axes[1].set_title('Proportion of Patient Conditions before Surgery', fontsize=18)

plt.tight_layout()

plt.show()
```

Fig 6.3.3 - Mean Difference % between dead and live & Proportion of Patient Conditions

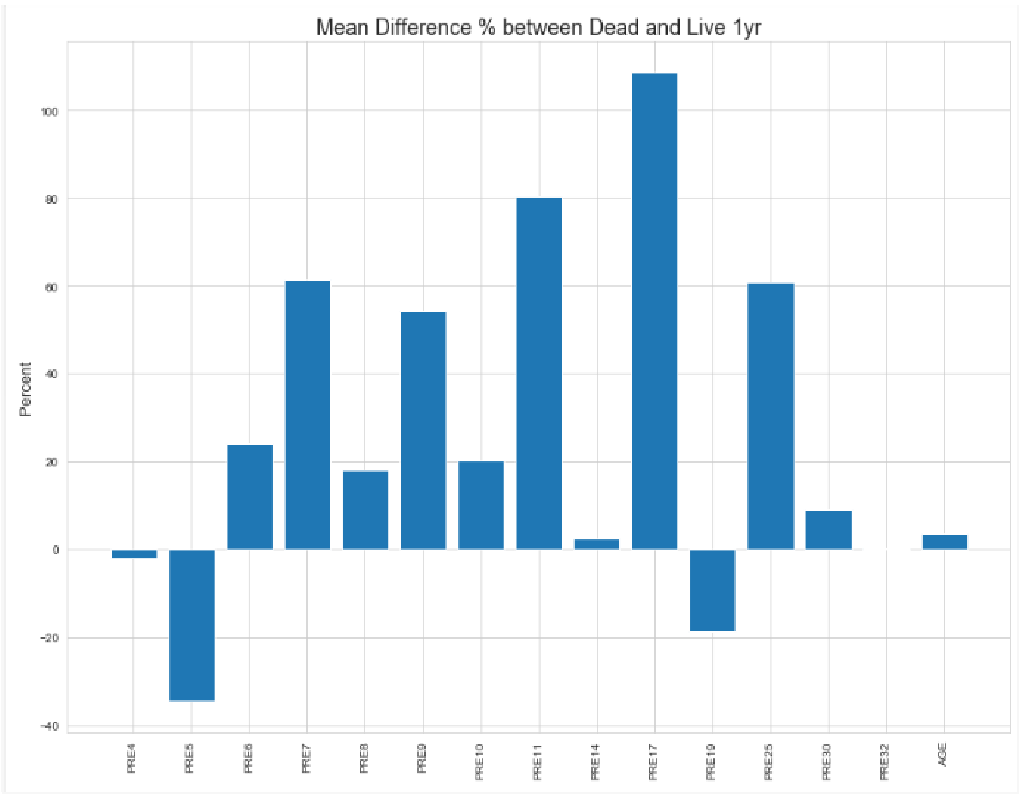


Fig 6.3.4 Mean Difference % between Dead and Live (Bar Graph)

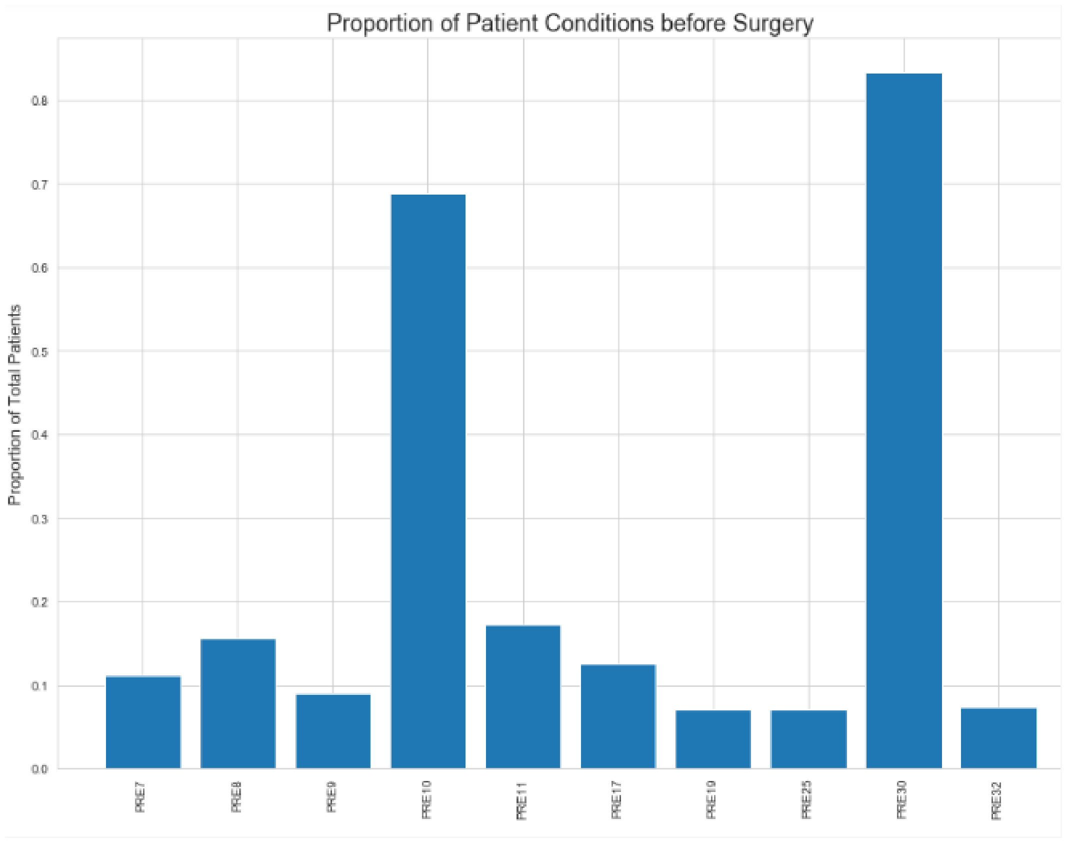
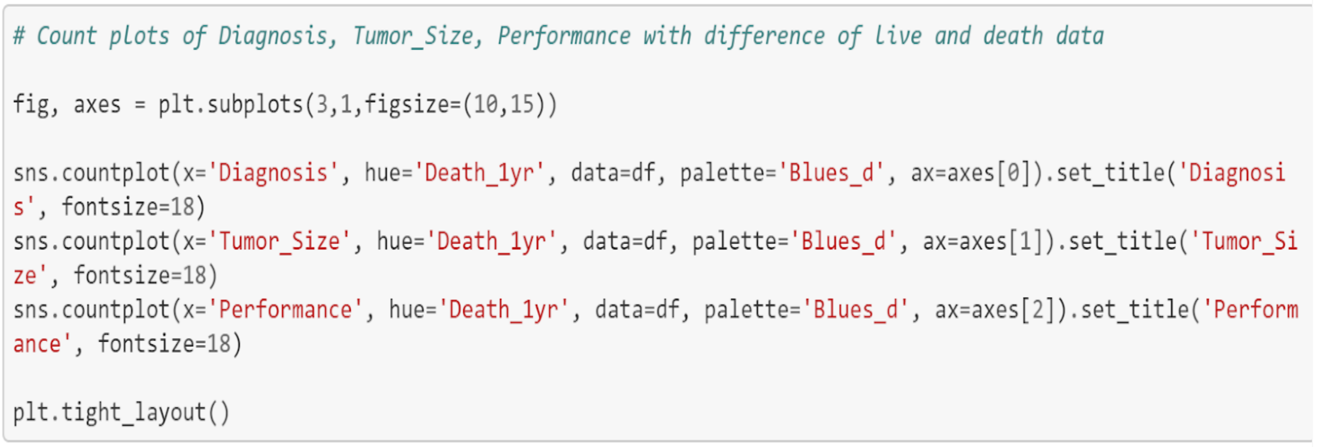


Fig 6.3.5 - Proportion of Patient Conditions before 1 year (Bar Graph)

Looking at the graph, one can easily compare the attributes to determine features of significance. The most notable attributes for those who died are Dyspnoea, Diabetes Mellitus, Pain, PAD, and Haemoptysis (in decreasing order), indicating that for those who died, these features were strongly presented. Asthma and MI of 6 months have negative 40% values, and looking at the numerical values reveals that those who died did not completely exhibit asthma or MI. Although the mean differences are useful, further investigation of the number of instances of each attribute in combination with the mean differences will improve our decision on what features to focus on.

The overall count should be considered when comparing mean differences, because the lower count numbers will have larger fluctuations to small differences. The count of Cough and Smoking are most noteworthy indicating these conditions are strongly correlated to those patients who are to receive thoracic surgery for lung cancer, but the mean differences are a small positive value indicating more representation in the dead patients. Looking at the count graph for noteworthy features from the mean differences graph, most of the values are low in count. To best assess if these attributes' mean differences are of significance, a hypothesis test of whether the mean difference is significant will help narrow down the features of interest.

B. Count plots of Diagnosis, Tumor_Size, Performance with difference of live and death data.



```
# Count plots of Diagnosis, Tumor_Size, Performance with difference of live and death data

fig, axes = plt.subplots(3,1,figsize=(10,15))

sns.countplot(x='Diagnosis', hue='Death_1yr', data=df, palette='Blues_d', ax=axes[0]).set_title('Diagnosi
s', fontsize=18)
sns.countplot(x='Tumor_Size', hue='Death_1yr', data=df, palette='Blues_d', ax=axes[1]).set_title('Tumor_Si
ze', fontsize=18)
sns.countplot(x='Performance', hue='Death_1yr', data=df, palette='Blues_d', ax=axes[2]).set_title('Perform
ance', fontsize=18)

plt.tight_layout()
```

Fig 6.3.6 - Code for diagnosis, Tumor_Size, Performance with difference of live and death data

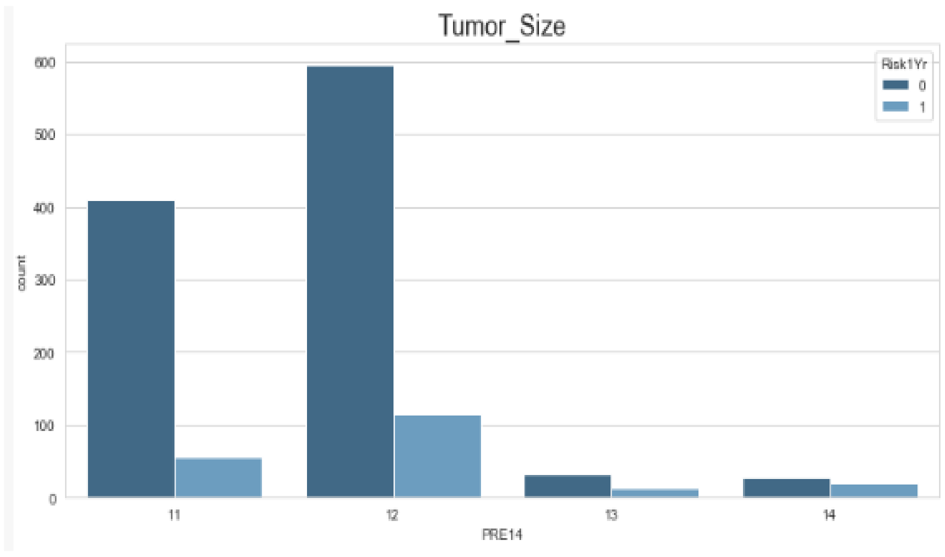


Fig 6.3.7 - Tumor_Size (Bar Graph)

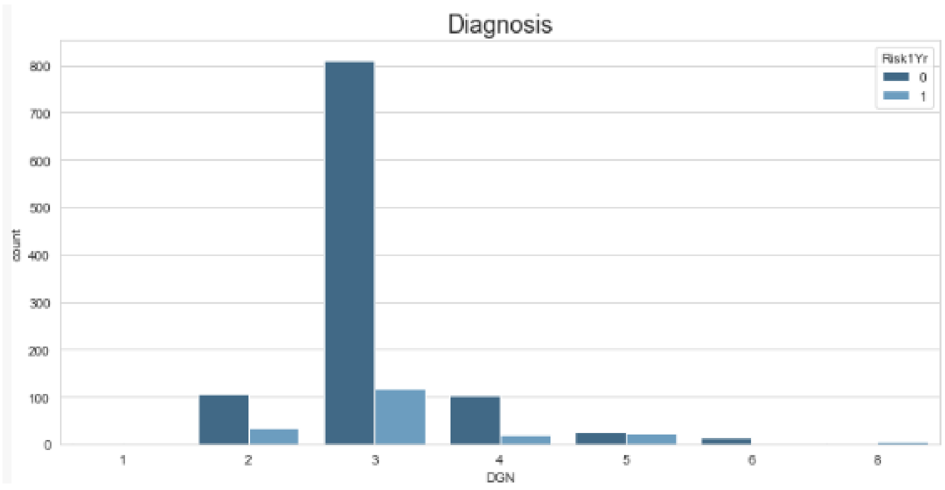


Fig 6.3.8 - Diagnosis (Bar Graph)

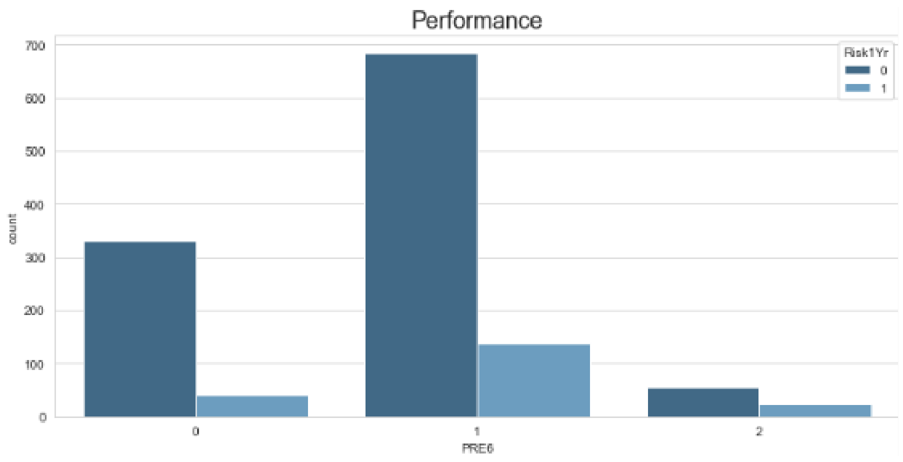


Fig 6.3.9 - Performance (Bar Graph)

For Diagnosis, the large majority of patients are in category 3. The other categories are relatively small while category 4,2 and 5 should be considered for their counts in that order. The proportion of live to dead at a glance seems to be similar for the diagnosis categories except for 5, where the death count is higher than the live count, which indicates this diagnosis is more fatal than the others even with surgery.

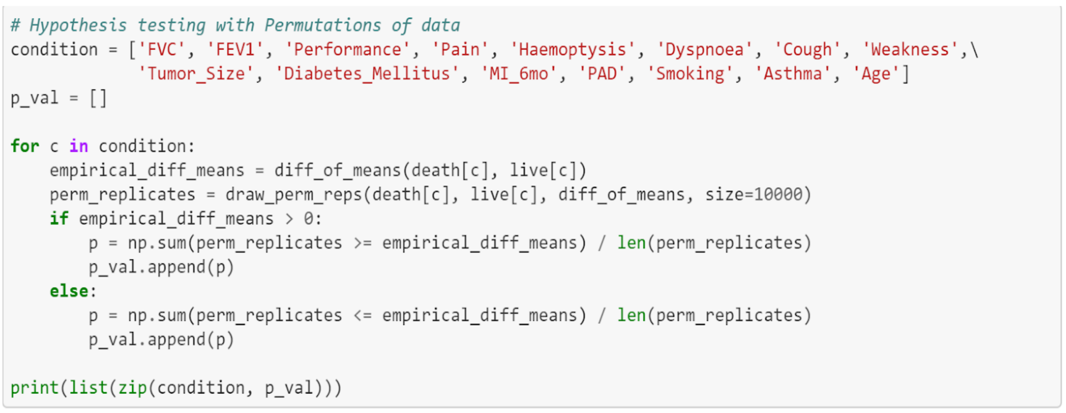
For Tumor Size, categories 1 and 2 are the majority. At a glance, the proportion of the dead to live generally increases with tumor size ranging from 1 to 4, indicating the higher tumor size correlates to higher chance of death even with surgery. Category 4 tumor size is most even in its split between death and live patient data. Also looking at the dead to live mean difference graph, the dead had higher means indicating larger tumor sizes overall.

For Performance, categories are 1,0,2 in decreasing order of count. Performance 0 category reveals low death count and good proportion to live data, which makes sense since on the Zubrod scale 0 is good and 2 is poor. Category 1 and 2 display similar proportions to live and dead patients, but with category 1 having a majority of the count. Referring to the dead to live mean difference graph, the dead had higher means indicating the dead on average had poor performance with a higher Zubrod score than the live.

C. Hypothesis test of mean differences between live and death patients

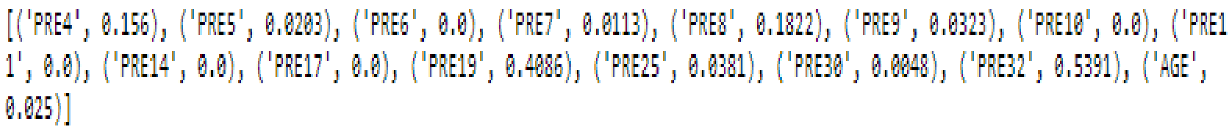
All the observations above highlighted the trends and patterns in the attributes. However, to ascertain their significance, a hypothesis test will be useful to see if these patterns and trends are not just by chance.

- **Null Hypothesis:** The 1 year live and death patients have the same distribution and mean. (Tested for each attribute.)
- **Test Statistic:** Mean difference between death and live patients.



```
# Hypothesis testing with Permutations of data
condition = ['FVC', 'FEV1', 'Performance', 'Pain', 'Haemoptysis', 'Dyspnoea', 'Cough', 'Weakness',\
             'Tumor_Size', 'Diabetes_Mellitus', 'MI_6mo', 'PAD', 'Smoking', 'Asthma', 'Age']
p_val = []

for c in condition:
    empirical_diff_means = diff_of_means(death[c], live[c])
    perm_replicates = draw_perm_reps(death[c], live[c], diff_of_means, size=10000)
    if empirical_diff_means > 0:
        p = np.sum(perm_replicates >= empirical_diff_means) / len(perm_replicates)
        p_val.append(p)
    else:
        p = np.sum(perm_replicates <= empirical_diff_means) / len(perm_replicates)
        p_val.append(p)

print(list(zip(condition, p_val)))
```

Fig 6.3.10 - Hypothesis Testing Code

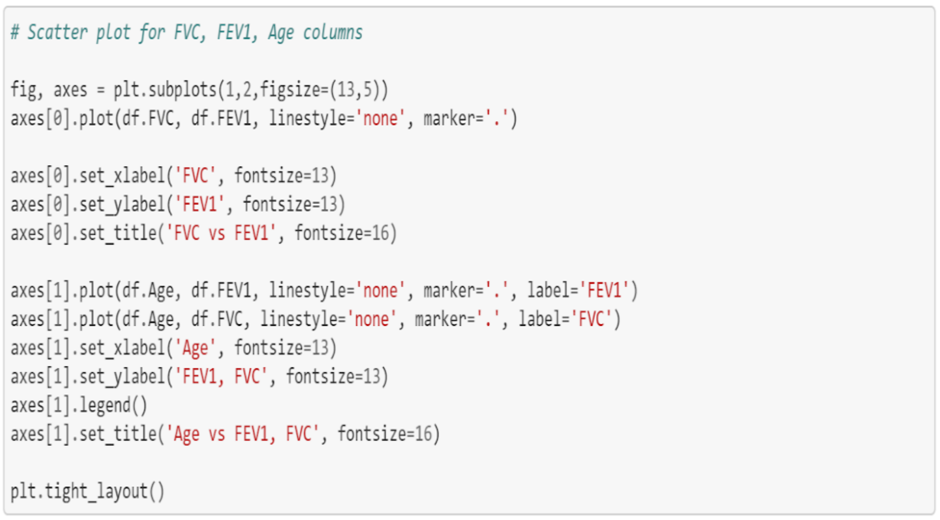


```
[('PRE4', 0.156), ('PRE5', 0.0203), ('PRE6', 0.0), ('PRE7', 0.0113), ('PRE8', 0.1822), ('PRE9', 0.0323), ('PRE10', 0.0), ('PRE1
1', 0.0), ('PRE14', 0.0), ('PRE17', 0.0), ('PRE19', 0.4086), ('PRE25', 0.0381), ('PRE30', 0.0048), ('PRE32', 0.5391), ('AGE',
0.025)]
```

Fig 6.3.11 - Hypothesis testing Output

D. Numerical Data (Age, FVC, FEV1)

The mean difference graph reveals little difference in Age while there is a small negative difference for the dead compared to the live. So this indicates the dead patients on average performed worse for lung capacity compared to the live patients. The plots below will further investigate the relationship between these three numerical data columns.



```
# Scatter plot for FVC, FEV1, Age columns

fig, axes = plt.subplots(1,2,figsize=(13,5))
axes[0].plot(df.FVC, df.FEV1, linestyle='none', marker='.')

axes[0].set_xlabel('FVC', fontsize=13)
axes[0].set_ylabel('FEV1', fontsize=13)
axes[0].set_title('FVC vs FEV1', fontsize=16)

axes[1].plot(df.Age, df.FEV1, linestyle='none', marker='.', label='FEV1')
axes[1].plot(df.Age, df.FVC, linestyle='none', marker='.', label='FVC')
axes[1].set_xlabel('Age', fontsize=13)
axes[1].set_ylabel('FEV1, FVC', fontsize=13)
axes[1].legend()
axes[1].set_title('Age vs FEV1, FVC', fontsize=16)

plt.tight_layout()
```

Fig 6.3.12 - Scatter Plot Code for Age,FVC,FEV1

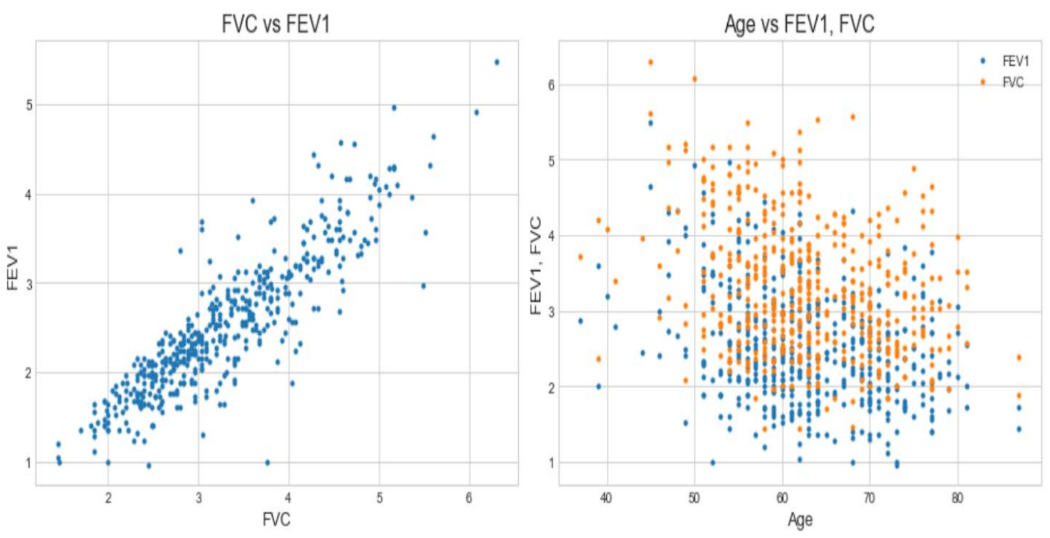


Fig 6.3.13 - Scatter Plot of FVC vs FEV1 and Age vs FEV1,FVC

E. Correlations of FVC, FEV1, and Age

```
In: np.corrcoef(df.FVC, df.FEV1)[0,1] ,  Out: 0.88754527338290001
In: np.corrcoef(df.FVC, df.FEV1)[0,1] ,  Out: 0.88754527338290001
In: np.corrcoef(df.FVC, df.FEV1)[0,1] ,  Out: 0.88754527338290001
```

From looking at the graphs, one can see a strong positive correlation between FVC and FEV1, while Age has a slight negative trend in the graphs. The correlation coefficient calculated for FVC and FEV1 is *0.89*, which is very strong on top of the fact that the data points are grouped together to show a visible linear trend. On the other hand, Age's correlation with FVC and FEV1 are about *-0.3* for both, but the data points are more spread out. The mild negative trend for age against the other two features makes intuitive sense as it would be expected that as you get older, your lung capacity decreases.

F. Summary of EDA

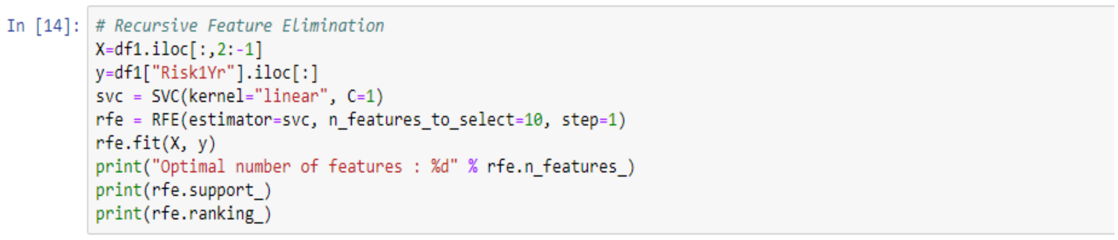
- Out of the 1270 patients, 203 died and 1067 lived the 1 year period after surgery. So, 16.0% of patients died.
- Several features presented strongly for those who died: Dyspnoea, Diabetes Mellitus, Pain, PAD, and Haemoptysis (Top 5 in decreasing order).
- Overall, most patients who received surgery smoked (~80%) and presented with symptoms of coughing (~70%), while the rest of the attributes presented under 20% of the total patients.

- Majority of patients were categorized as diagnosis code 3. The proportion of live to dead is similar for the diagnosis codes. For tumor size, most of the patients are in category 1 and 2, and the data presents a trend of more proportion of dead as tumor size increases. For performance, the trend observed is more proportion of dead as performance zubrod score increases, which means patient performance decreases.
- Hypothesis testing reveals attributes of significance: Performance, Dyspnoea, Cough, Tumor_Size, Diabetes_Mellitus.
- As Age increases, the FEV1 and FVC decrease with correlations of -0.31 and -0.30, respectively. FEV1 and FVC are highly correlated with a value of 0.89.

6.4. Methodology

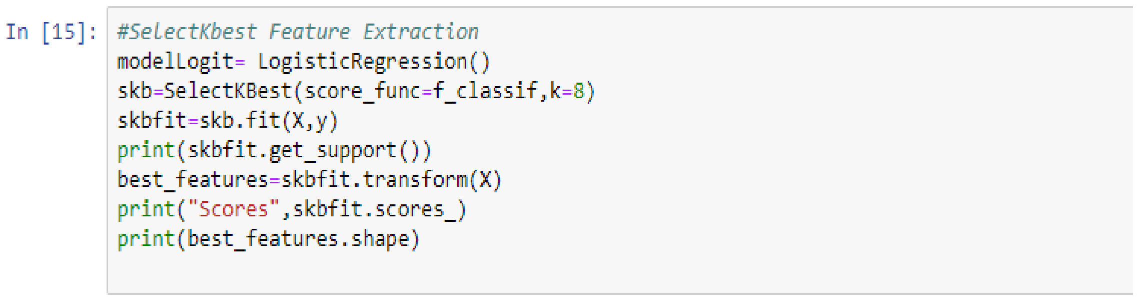
So now that we have thoroughly examined the patterns and trends in the data set, the next step is to use machine learning models to see if the target variable, Death 1yr, with the function variables can be predicted as well. Because the data set is imbalanced and mostly live patients (85%), only predicting all live patients would yield a high precision score of ~85%. So for the model, accuracy is not a reasonable method of score and instead looks at the confusion matrix.

Before proceeding for modelling, not every attribute of the dataset contributes towards the result. So in order to evaluate which attributes affect the result, we performed feature extraction. All the feature extraction by default provides with half attributes of the total. So considering EDA, we considered 10 attributes for modelling. First step was to split the dataset into training and test data where 30 percent is test data using sklearn functions. Once this is done, feature scaling is done for standardization purposes. Starting with the first model Support Vector Machine, with the help of sklearn we imported an SVM classifier which helps to train and fit the dataset and further helps to predict the accuracy and confusion matrix. Further same procedure was followed for XGboost.



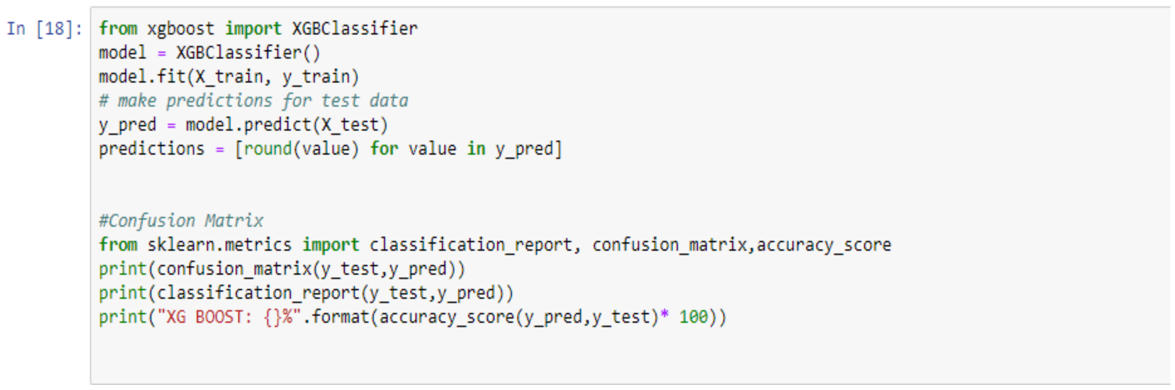
```
In [14]: # Recursive Feature Elimination
         X=df1.iloc[:,2:-1]
         y=df1["Risk1Yr"].iloc[:]
         svc = SVC(kernel="linear", C=1)
         rfe = RFE(estimator=svc, n_features_to_select=10, step=1)
         rfe.fit(X, y)
         print("Optimal number of features : %d" % rfe.n_features_)
         print(rfe.support_)
         print(rfe.ranking_)
```

Fig 6.4.1 - RFE Code



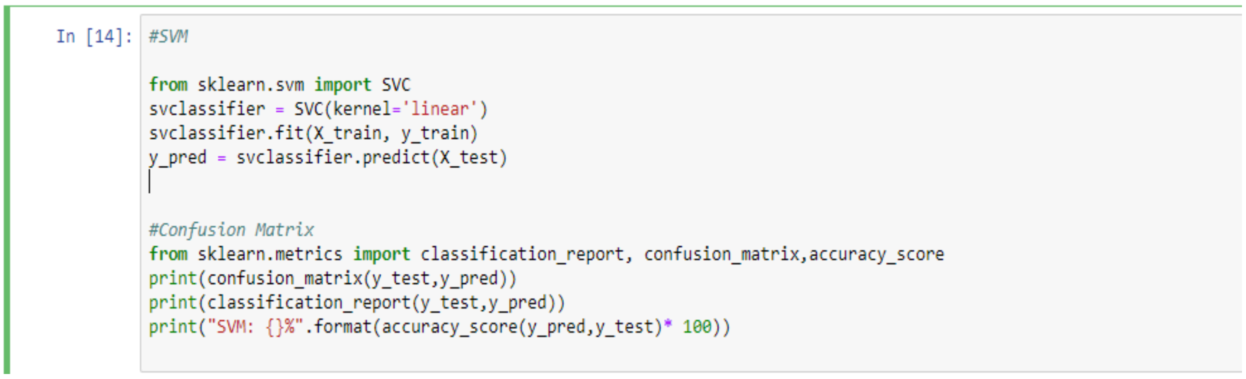
```
In [15]: #SelectKbest Feature Extraction
         modelLogit= LogisticRegression()
         skb=SelectKBest(score_func=f_classif,k=8)
         skbfit=skb.fit(X,y)
         print(skbfit.get_support())
         best_features=skbfit.transform(X)
         print("Scores",skbfit.scores_)
         print(best_features.shape)
```

Fig 6.4.2 - Select k-best Code



```
In [18]: from xgboost import XGBClassifier
         model = XGBClassifier()
         model.fit(X_train, y_train)
         # make predictions for test data
         y_pred = model.predict(X_test)
         predictions = [round(value) for value in y_pred]


         #Confusion Matrix
         from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
         print(confusion_matrix(y_test,y_pred))
         print(classification_report(y_test,y_pred))
         print("XG BOOST: {}%".format(accuracy_score(y_pred,y_test)* 100))
```

Fig 6.4.3 - XGBoost Code



```
In [14]: #SVM

         from sklearn.svm import SVC
         svclassifier = SVC(kernel='linear')
         svclassifier.fit(X_train, y_train)
         y_pred = svclassifier.predict(X_test)


         #Confusion Matrix
         from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
         print(confusion_matrix(y_test,y_pred))
         print(classification_report(y_test,y_pred))
         print("SVM: {}%".format(accuracy_score(y_pred,y_test)* 100))
```

Fig 6.4.4 - SVM Code

6.5. Working of the Project GUI

The project consists of a Machine Learning model which is trained over the dataset to predict the probability of patients. The project is presented as a web app which can be used by the doctors to predict the chances of patients' survival. This enables doctors to take the necessary precaution before the surgery. The web application is flask website which consists of 5 main files wiz. main.py, training.py, model.pkl and 2 html files. The flow diagram is shown below:

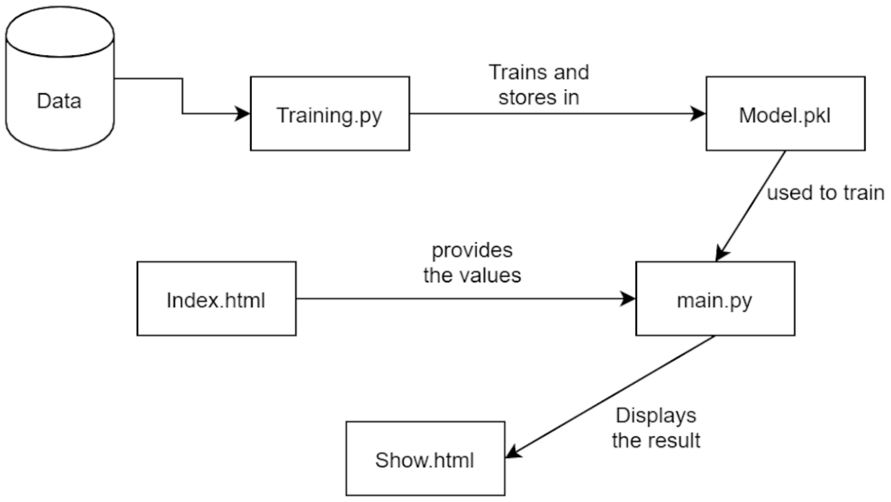


Fig 6.5.1 - Working of GUI

The dataset is used by the *training.py* to train and test the model. This trained model is then stored in a file called *model.pkl*. This *model.pkl* is stored remotely and called by the *main.py* to predict the output. The trained model gets inputs required for prediction from the *index.html* file which is entered by the user. These inputs are used by *main.py* to predict the probability of survival. This prediction is displayed in *show.html* file

Chapter 7

Testing

7.1. Test Cases

a) **Pipeline Flow Testing:**

End to End Testing of the System, right from attribute collection to prediction of the life expectancy of the patient.

b) **Interface Testing:**

User interaction and interaction ability testing to look out for glitches and loopholes in the system.

7.2. Types of Testing Used

a) **Component Testing:**

It is mostly performed by the developers after the completion of unit testing. Component Testing consists of testing multiple functionalities as a single code and its objective is to identify if any defect exists after connecting those multipleu functionalities with each other.

b) **End-to-End Testing:**

Just like system testing, End-to-end testing implies testing of the entire application environment in a situation that mimicsqreal-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

c) **Functional Testing:**

Functional testing ignores the internal parts and focuses only on the output to check if it is according to the requirement given or not. It is a Black box testing method. It is adapted to get the functional requirements of an application.

d) Graphical User Interface (GUI) Testing:

The main aim to test the GUI is to validate the GUI as per the business requirement given. Generally, the GUI of the application that is expected by the developer is mentioned in the Detailed Design Document and GUI mockup screens. The GUI testing consists of the size of the buttons and input field present on the screen, alignment of all tables, text and content in the tables. It also validates the menu of the application, after choosing multiple menu and menu items, it validates that the page does not shift and the alignment remains the constant after moving the mouse on the menu.

Chapter 8

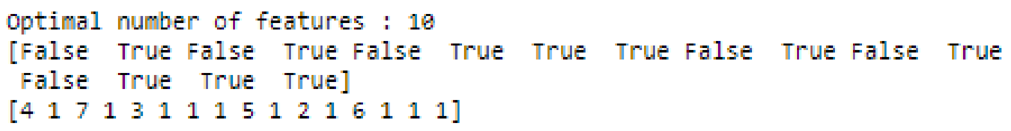
Results and Discussion

1. Feature selection

The following feature selection techniques were used and their results are displayed below

a. Recursive Feature extraction

The recursive feature extraction selected 10 top features that will make the model better and avoid overfitting. The selected features are FVC, Performance, Haemoptysis, Dyspnoea, Cough, Tumor_Size, Myocardial infarction, Smoking, Asthma and Age.

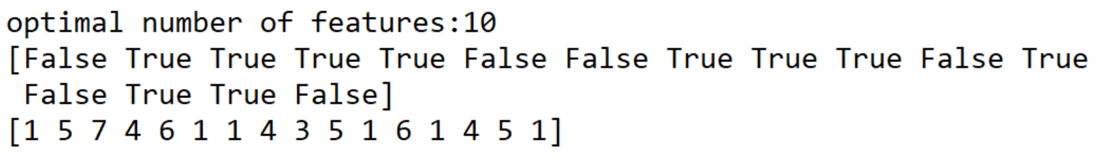


```
Optimal number of features : 10
[False  True False  True False  True  True  True False  True False  True
 False  True  True  True]
[4 1 7 1 3 1 1 1 5 1 2 1 6 1 1 1]
```

Fig 8.1.1 - RFE Output

b. Univariate Feature extraction

The recursive feature extraction selected 10 top features that will make the model better and avoid overfitting. The selected features are FVC, FEV 1, Performance, Pain, Weakness, Cough, Tumor_Size, Myocardial infarction, Smoking and Asthma.

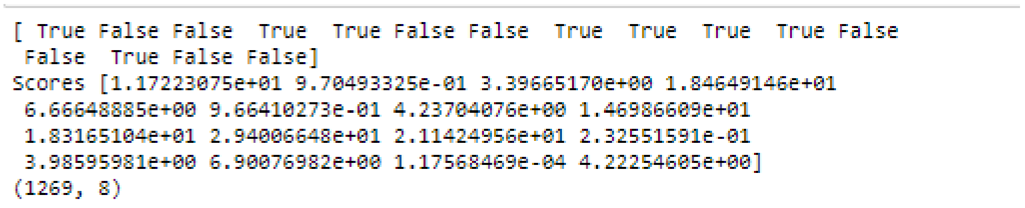


```
optimal number of features:10
[False True True True True False False True True True False True
 False True True False]
[1 5 7 4 6 1 1 4 3 5 1 6 1 4 5 1]
```

Fig 8.1.2 - Univariate Feature Selection

c. Select K best Feature extraction

The select K best selected 10 top features that will make the model better and avoid overfitting. The selected features are Diagnosis, Performance, Pain, Cough, Weakness, Tumour size, diabetes mellitus and smoking.

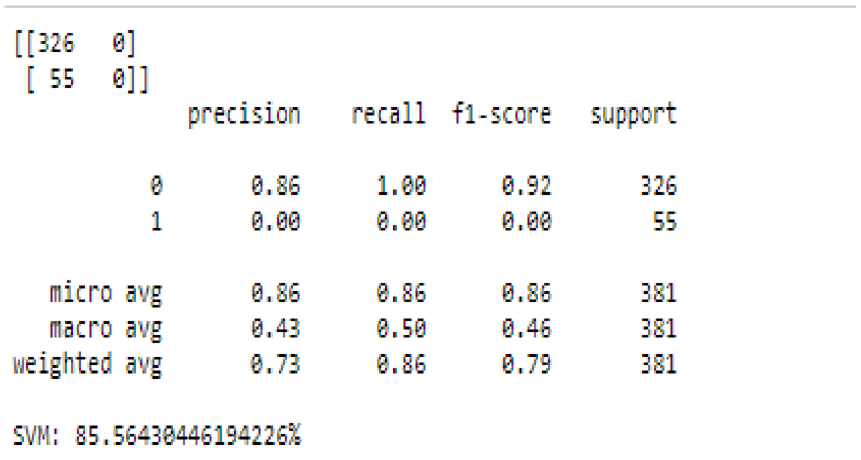


```
[ True False False  True  True False False  True  True  True  True False
 False  True False False]
Scores [1.17223075e+01 9.70493325e-01 3.39665170e+00 1.84649146e+01
 6.66648885e+00 9.66410273e-01 4.23704076e+00 1.46986609e+01
 1.83165104e+01 2.94006648e+01 2.11424956e+01 2.32551591e-01
 3.98595981e+00 6.90076982e+00 1.17568469e-04 4.22254605e+00]
(1269, 8)
```

Figure 8.1.3 - K-best Feature Extraction

2. SVM

We applied the SVM algorithm on the output of all the feature extraction. Out of those the best accuracy was obtained in RFE. The SVM algorithm on classification of the data from RFE gave an accuracy of 85.56%. This is a good accuracy but it could be further improved. To improve the accuracy we used Random Forest algorithm.

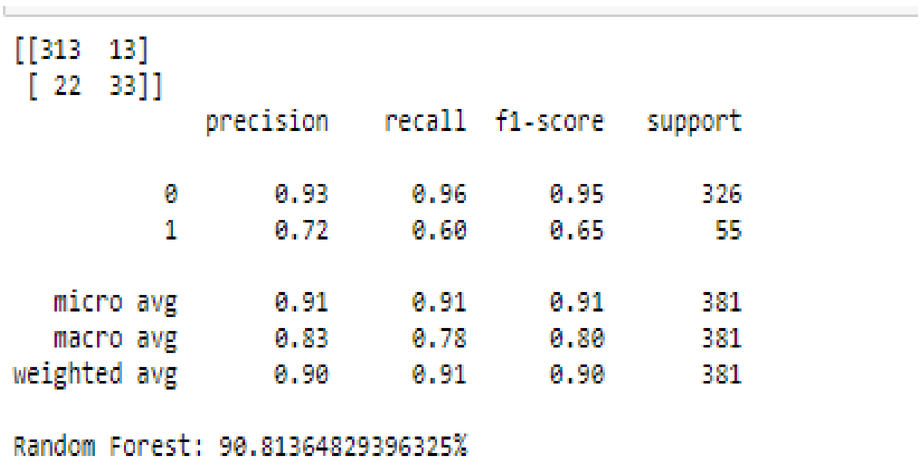


```
[[326   0]
 [ 55   0]]
              precision    recall  f1-score   support

           0       0.86      1.00      0.92       326
           1       0.00      0.00      0.00        55

   micro avg       0.86      0.86      0.86       381
   macro avg       0.43      0.50      0.46       381
weighted avg       0.73      0.86      0.79       381

SVM: 85.56430446194226%
```

Fig 8.1.4 - SVM Output

3. Random Forest

We applied the Random Forest algorithm on the output of all the feature extraction. Out of those the best accuracy was obtained in RFE. The Random Forest algorithm on classification of the data from RFE gave an accuracy of 90.81%. This is a good accuracy but it could be further improved. To improve the accuracy we used Extreme Boosting algorithm.

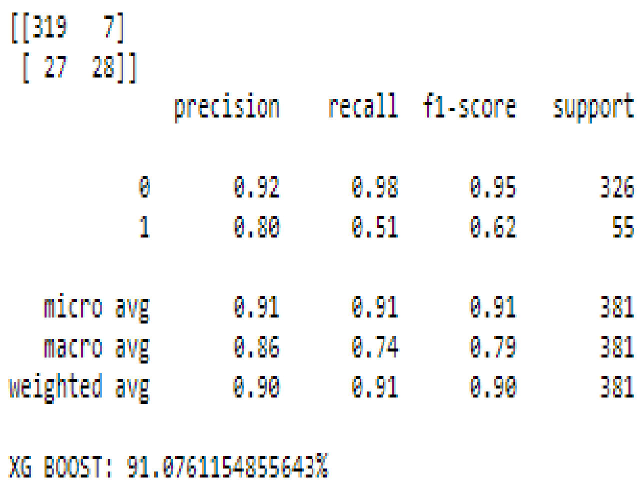


```
[[313  13]
 [ 22  33]]
              precision    recall  f1-score   support

           0       0.93      0.96      0.95       326
           1       0.72      0.60      0.65        55

   micro avg       0.91      0.91      0.91       381
   macro avg       0.83      0.78      0.80       381
weighted avg       0.90      0.91      0.90       381

Random Forest: 90.81364829396325%
```

Fig 8.1.5 - Random Forest Output

4. XGB

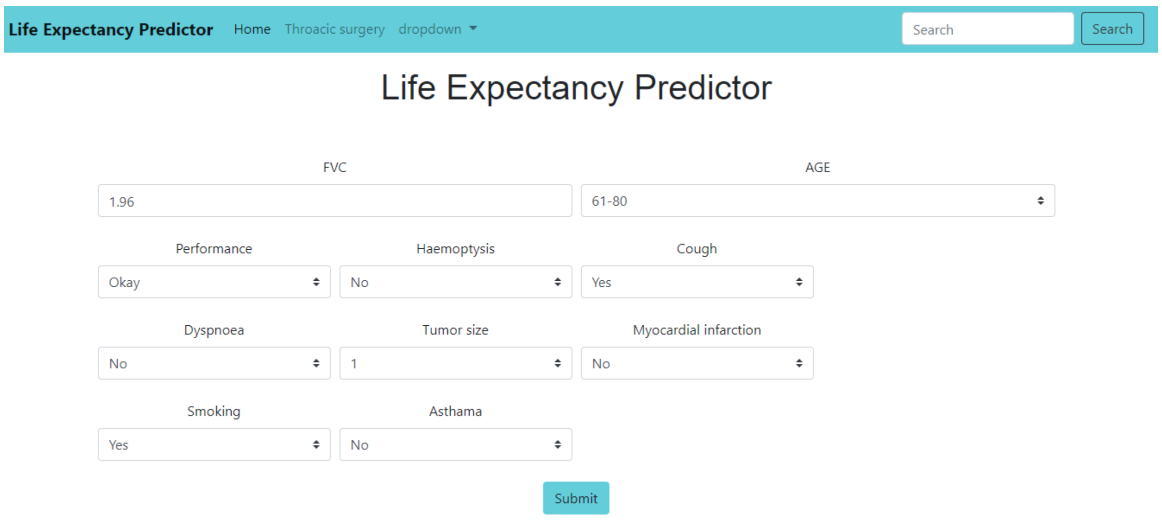
We applied the Extreme Boosting (xgb) algorithm on the output of all the feature extraction. Out of those the best accuracy was obtained in RFE. The xgb algorithm on classification of the data from RFE gave an accuracy of 91.07%. This is only 1% increase in the accuracy but it makes a load of difference in the prediction capabilities. The xgb algorithm boosts the model to make clear and accurate prediction.



```
[[319   7]
 [ 27  28]]
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       326
           1       0.80      0.51      0.62        55

   micro avg       0.91      0.91      0.91       381
   macro avg       0.86      0.74      0.79       381
weighted avg       0.90      0.91      0.90       381

XG BOOST: 91.0761154855643%
```

Fig 8.1.6 - XGB Output

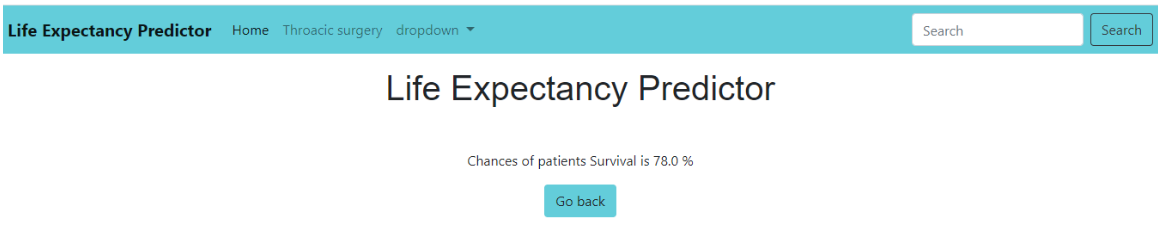
5. GUI Working



Life Expectancy Predictor

FVC: 1.96	AGE: 61-80	
Performance: Okay	Haemoptysis: No	Cough: Yes
Dyspnoea: No	Tumor size: 1	Myocardial infarction: No
Smoking: Yes	Asthama: No	

Fig 8.1.7 - GUI (Attribute)



Life Expectancy Predictor

Chances of patients Survival is 78.0 %

Fig 8.1.8 - GUI (Predictor)

Chapter 9
Conclusions and Future Scope

9.1. Conclusion

The main aim of the model is to ease the decision-making process of whether the patient will have any post-surgery complications. As opposed to the currently used 30-day mortality rate, our solution aims to expand this time period to up to a year thus increasing the effectiveness The model proposed in this project serves the purpose of selecting appropriate features using the recursive feature elimination method and then apply machine learning algorithms like SVM and ensemble technique like XGB (Extreme Gradient Boosting) in order to approximately predict the survival rate of a patient after a year post-surgery.
This model is machine learning model specifically designed for thoracic surgery and fulfils its purpose with good results. The project successfully predicts the survival rate of the patient using Extreme Boosting algorithm with an accuracy of 91.07%.

9.2. Future Scope

The project predicts whether the patient has survived for a year or not and hence this project has many applications for the future. The project primarily focuses on thoracic surgery, but it can be applied to various other surgeries and medical practices.

The model's accuracy could be increased by bumping or boosting methods to produce reliable results. Also the models could be applied without ranking or feature selection with good accuracy and recall. We can also apply other feature selection and data mining techniques and obtain much better accuracy with reduced number of features. Bootstrapping could be another useful method to apply for improving the results.

References

[1] Adam Abdul Hamid, Ivaylo Bahtchevanov, Peng Jia, “Life Expectancy Post Thoracic Surgery”, Stanford University - CS 229
[2] Abeer S. Desuky and Lamiaa M. El Bakrawy, “Improved Prediction of Post-operative Life Expectancy after Thoracic Surgery”, Al-Azhar University, Cairo, Egypt.
[3] Kwetise Joro Danjuma, “Performance Evaluation of Machine Learning Algorithms in Post-Operative Life Expectancy in the Lung Cancer Patients ”**,** Department of Computer Science, Modibbo Adama University of Technology, Yola, Adamawa State, Nigeria.
[4]https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
[5]https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
[6]https://blog.exploratory.io/introduction-to-extreme-gradient-boosting-in-exploratory-7bbec554ac7
[7]https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
[8]https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15
[9] https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html

Acknowledgements

We thank the almighty for giving us the courage and perseverance in completing the mini project. This mini project itself is acknowledgement for all those people who gave their heartfelt co-operation in making this project.

We extend our sincere thanks to Dr. Meera Narvekar, HOD of our department, for providing constant encouragement and good environment in the department to complete our course. We specially thank our Project mentors Mrs. Sindhu Nairr, Assistant Professor, Computer engineering and Mrs. Aruna Gawde, Assistant Professor, Computer engineering, for providing valuable guidance at every stage of this project work. We are profoundly grateful towards the unmatched help rendered by them. Our special thanks to all the faculties and peers for their valuable advice which has helped us in finally compiling it. Last but not the least, we would like to express our deep sense of gratitude and earnest thanks giving to our dear parents for their moral support and heartfelt cooperation.