A project report on

LUNG CANCER PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment for the award of the degree of

B.TECH-INFORMATION TECHNOLOGY

by

SATYA AAKASH OBELLANENI (18BIT0128)

SITE- VIT,VELLORE

UNDER THE GUIDANCE

OF

Dr. KURUVA LAKSHAMANNA

Lung Cancer Prediction Using Machine Learning Techniques

Submitted in partial fulfillment of the requirements for the degree of

B.Tech – Information Technology and Engineering

by

SATYA AAKASH CHOWDARY OBELLANENI

(18BIT0128)

SITE-VIT

April,2022

UNDER GUIDANCE

OF

DR. KURUVA LAKSHMANNA

DECLARATION

I hereby declare that the thesis entitled "Lung cancer prediction using machine learning techniques" submitted by me, for the award of the degree of *Bachelor of Technology in Information Technology-ITE4999* to VIT is a record of bonafide work carried out by me under the supervision of Dr.Kuruva Lakshmanna.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date:22/04/2022

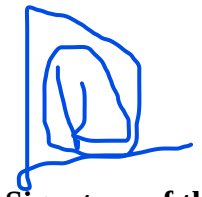
Aakash

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Lung cancer prediction using machine learning techniques" submitted by **Satya Aakash chowdary obellaneni & 18BIT0128**, **SITE**, VIT, for the award of the degree of *Bachelor of Technology in Information technology*, is a record of bonafide work carried out by him under my supervision during the period, 10. 01. 2022 to 27.04.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.



Signature of the Guide **Signature of the HoD**

Internal Examiner **External Examiner**

ABSTRACT

Lung cancer is the prime cause of cancer related deaths for both men and women. Lung cancer is the rapid growth of abnormal cells in one or both Lung. Lung cancer is a type of cancer that starts in the lungs. Your lungs are two spongy organs in your chest that take in oxygen when you inhale and release carbon dioxide when you exhale. Lung cancer is the leading cause of cancer deaths in the United States, among both men and women. There is various type of the cancer like lung, breast, prostrate, carnival etc. Each type of cancer has specific symptoms. Based on the symptoms the type of the cancer is predicted. In this project we are mainly focusing on the lung cancer prediction as 1 in 4 cancer deaths are from lung cancer. Lung cancer claims more lives each year than do colon, prostate, ovarian, and breast cancers combined. In India, more than 1 million cases were reported per annum, and it is only handled by professionals in those sectors. Lung cancer needs medical diagnosis, so prediction and detection of lung cancer is much needed beforehand. In this research, we gave a thought to predict lung cancer using some famous and strong machine learning algorithms to stand against lung cancer. WHO also reported that if lung cancer is recognized earlier so that we can diagnose and also reduce the death rate.

The information related to patients is collected from the standard dataset from Kaggle then it is pre-processed with the traditional technique. We analyze lung cancer prediction using classification algorithms such as Naive Bayes, logistic regression, SVM (Support Vector Machine), KNN, Decision Tree, Random Forest and few advanced techniques like hyper parameter tuning for least accuracy algorithms and also max voting ensemble technique. Cancer and non-cancer patient's data is gathered from the dataset, The dataset that we have collected has 1000 instances and 25 attributes. The noisy, irrelevant, missing data is eliminated. Then we are going to use classification algorithms like Naïve Bayes, Support Vector Machine, Logistic regression, etc to build a cancer risk prediction system is proposed here which predicts cancers and is also user-friendly, time and cost-saving.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Kuruva Lakshmanna, Associate Professor, SITE, Vellore Institute of Technology, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of area.

I would like to express my gratitude to Chancellor, VPs, VC, PRO-VC, and Dr. Sumathy, SITE, for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Usha Devi, HoD, SITE all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Vellore

Name of the student

Date:22-04-2022

AAKASH

CONTENTS

::

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

UCI - UC Irvine Machine Learning Repository

NLST - National Lung Screening Trial

NSCLC - Non-Small Cell Lung Cancer

LIDC-IDRI - Lung Image Database Consortium image collection

BNN - Binarized Neural Network

PCA - Principal Component Analysis

CT Scan- Computerized Tomography

SVM - Support Vector Machine

KNN - K -Nearest Neighbors

DT - Decision Tree

HPLG - Hyper parameter tuning for logistic regression

HPGNB - Hyper parameter tuning for Naïve Bayes

HPSVM - Hyper parameter tuning for Support Vector machine

Chapter 1

Introduction

Lung cancer is the most common cause of cancer death worldwide. If the original lung cancer has spread, a person may feel symptoms in other places in the body. The lung cancer symptom is used to predict risk level of disease. We are analyzing very popular algorithms and stimulating them by training with an approved dataset which is the dataset of people who are diagnosed. When the same data set is trained with different algorithms, we can identify which one of the algorithms are giving out better results and also try to improve the algorithms accuracy.

1.1 PROBLEM STATEMENT

There is various type of the cancer like lung, breast, prostrate, carnival etc. Each type of cancer has specific symptoms. Based on the symptoms the type of the cancer is predicted. In this project we are mainly focusing on the lung cancer prediction as 1 in 4 cancer deaths are from lung cancer. To compare the machine learning model's accuracy for the given dataset and improve the least accuracy models to compete with higher performance models. The problem is to find the solution for lung cancer patients and give them the result by predicting far superior. This enhances the new opportunities and new findings in cross over of both medical and software industry.

1.2 OBJECTIVE

We analyze the lung cancer prediction using classification algorithm such as Naive Bayes, Decision tree and SVM (Support Vector Machine), KNN, Decision Tree, Random Forest and few advanced techniques like hyper parameter tuning for least accuracy algorithms and also max voting ensemble technique. Cancer and non-cancer patient's data is gathered

from the dataset, pre-processing will be done and will be analyzed using a classification algorithm for predicting lung cancer. The dataset that we have collected has 1000 instances and 25 attributes. These 25 attributes are most commonly observed in lung cancer patients and we perform feature selection for highly effective features for better accuracy model .

1.3 OVERVIEW

Lung cancer has high death rate among all the cancers. When I decided to do machine learning project for my capstone, I thought of completing it for cause to society. Then I researched about cancers and its death rates, Lung cancer is phenomenal to do a project. There was plenty of research was going on with highly advanced technical aspects, so I was more excited to bring results with this project and make this a stepping stone for my future endeavors.

I have researched about 15 2021 research papers and gave a architecture to my problem and then started about process and implementation, evaluation and results to it. This model consists of 7-8 techniques of predicting lung cancer and data analysis, visualization, feature selection has higher priority for the results obtained. In this research, we gave a thought to predict lung cancer using some famous and strong machine learning algorithms to stand against lung cancer. WHO also reported that if lung cancer is recognized earlier so that we can diagnose and reduce the death rate.

All the traditional process of machine learning models were observed in This system with designing, implementation, validation and improving Them with advanced techniques like hyper parameter tuning and ensemble methods for strong machine learning algorithms. Providing a great comparison for strong algorithms to implement models for lung cancer disease in future.

CHAPTER 2

Background and Survey

2.1 LITERATURE REVIEW

Zhiyu Wang et al [1]; proposed that "Machine Learning Algorithm Guiding Local Treatment Decisions to Reduce Pain for Lung Cancer Patients with Bone Metastases" describes that they intake 746 patients and conducted the study on bone metastases which they classified into four groups according to surgery and other constraints. They evaluated on basis of cost valuation, performance evaluation, and model development.

The paper also states that this was the first time to experiment for a decision model in lung cancer with bone metastasis. The treatment brings pain relief without any side effects. The study concludes that a decision tree with 90.06% was better than the support vector machine and BNN model.

B R Manju et al [2]; proposed "Efficient multi-level lung cancer prediction model using support vector machine classifier" describes that model gives high accuracy in early detection of lung cancer. They collected a dataset from the UCI repository with 600 instances and with strong attributes. The study used PCA for feature selection in this and reduced unnecessary attributes from the dataset. They also performed feature correlation for the contribution of each feature. The results of the study are 87% accuracy with a 0.3 error rate with a support vector classifier.

Marjolein A. Heuvelmans et al [3]; proposed "Lung cancer prediction by Deep Learning to identify benign lung nodules" describes the study with 10368 participants (NLST DATASET) from CT image using supervised learning approach. They used three different benign nodules like Groningen, Heidelberg, and Oxford. the model identified 18.5% of patients correctly in benign nodules with 99% sensitivity and they used the LCP-CNN method for this study and it shows excellent performance on lung nodules containing patients.

Sebastien Benzekry et al [4]; proposed "Machine Learning for Prediction of Immunotherapy Efficacy in Non-Small Cell Lung Cancer from Simple Clinical and Biological Data" describes about Immune checkpoint inhibitors in NSCLC(non-small cell lung cancer) . Machine learning models were used for this with tenfold cross validation and the results show random forest is best for the system. The data collected from 298 patients was approved by Institutional Review Board of French Society of Respiratory Diseases and performed with different algorithms(logistic regression, random forest, single layer neural network, naive bayes, k-nearest neighbors and support vector machines) for high accuracy. Best accuracy was 68% is achieved by random forest.

R. Sujitha et al [5]; proposed "Classification of lung cancer stages with machine learning over big data healthcare framework" describes the competition between machine learning and big data health care frame work and it achieves 86%accuracy with support vector machine. The data is collected from microscope lab with 500 sputum color images to classify stages of lung cancer and they processed data with mapreduce framework over apache spark.SVM multi class had edge over binary class in this study and also study proves that both frameworks produces the cancer stage and also how much effect it is .

Ping-Hsien Tsou et al [6]; proposed "Exploring Volatile Organic Compounds in Breath for High-Accuracy Prediction of Lung Cancer" describes the human exhaled volatile organic compounds (VOC) with 316 patients data of 116 kinds and performed a study with predictive model (XGBoost). The data collected from NCTU and NTUH and they observed major differences between both of the groups like ethanol, formic acid, ethanedoal, methanol, etc concentrations are differ from lung cancer and healthy patients .the model identified lung cancer patients perfectly with exhaled breath. it was a milestone in this era because new method of detecting lung cancer with using machine learning techniques.

Pankaj Nanglia et al[7]; proposed "A hybrid algorithm for lung cancer classification using svm and neural networks" describes the study of lung cancer prediction with the combination of support vector machine and feed-forward back propagation neural network and this proposed method is named as KASC . The dataset used in this study was developed by Cornell University which had 500 CT scan images .The results shows that the proposed method had 98% accuracy and also a rise of 3% with rise in number of samples from 100 to 500.F-measure for KASC shows an average value of 97% and recall value of 96.5%. The integration of svm and neural network had made the model hybrid with high accuracy scores.

S.Shanthi et al [8]; proposed " Lung Cancer Prediction Using Stochastic Difusion Search (SDS) Based Feature Selection and Machine Learning Methods" describes the lung cancer predicition through grey level co-occurence matrix combined with gabor filter feature extraction are performed in this study. Method used stochastic diffusion search algorithm for optimal features and also neural networks like Naive bayes, decision tree were used. In this study, Radiomics feature is also used for detailed characterisation of tumour. The dataset of 270 patients were recorded and it was produced by TCGA. Results proved that neural networks are used for the diagnosis of lung cancer (SDS-NN by 2.51% for SDS-decision tree and by 1.25% for SDS-naive bayes).

Naresh Cherukuri et al [9]; proposed "Deep Learning for Lung Cancer Prediction using NSCLS patients CT Information" describes the lung cancer prediction using deep learning by using three-dimensional CONVOLUTION NEURAL NETWORK. the data of 1183 patients was collected for study by NSCLC. The results shows 95% CI with AUC and Surgical treatment and also it has a 0.91 correlation factor. The method defines the patience for 2years clinical trails had proved the high accuracy with typical set of records.

Meraj Begum Shaikh Ismail et al [10]; proposed "Lung Cancer Detection and Classification using Machine Learning Algorithm" describes lung cancer detection using CT scan images due to less noise and those images were

segmented by Kmeans and feature selection was performed. Then many algorithms were performed for achieving accuracy. Three datasets were used for this study 1)TCIA 2)LIDC-IDRI 3)kaggle 1595 patients. The convolutional network is used for image segmentation and then logistic regression, naive Bayes, support vector machine, random forest, Gradient Boosting are used for training. the results showed that classifiers are performing better than neural networks in this study.

TABLE 1. COMPARISON OF RELATED WORK

S.NO	Paper Title	Method used	Dataset	Pros/Cons	Measure
1.	Machine Learning Algorithm Guiding Local Treatment Decisions to Reduce Pain for Lung Cancer Patients with Bone Metastases, a Prospective Cohort Study	Decision tree, Support vector machine, Binarized neural network	from local clinical patients (on their own)	The prediction rate is high but the data is not adequate because it was the first attempt in this sector.	The decision tree had 89.44% sensitivity, 90.06% accuracy, test set 86.11% accuracy
2.	Efficient multi-level lung cancer prediction model using support vector machine classifier	support vector classifier	UCI repository with 600 instances	results show average accuracy rate but with less error rate	accuracy 87% error rate 0.3
3.	Lung cancer prediction by deep learning to identify benign lung nodules	Lcp-Convolution neutral network	NLST-10368 patient records	the system used ct scan images with AI tool and	sensitivity for lung nodules identification

				predicts extremely well with lung nodules patients , but in remaining cases it has low accuracy	is 99%
4	Machine Learning for Prediction of Immunotherapy Efficacy in Non-Small Cell Lung Cancer from Simple Clinical and Biological Data	logistic regression, random forest,single layer neural network, naive bayes, k-nearest neighbours and support vector machines	Institutional Review Board of French Society of Respiratory Diseases	The study performs average with accuracy but it produces a quality report of best algorithm among 6 top algorithm as random forest	Random forest achieves 68% and high in specificity
5	Classifcation of lung cancer stages with machine learning over big data healthcare framework	a combination of binary classifcation (SVM-non linear SVM with Radial Basis Function RBF) and Multiclass classifcation	microscope lab with 500 sputum color images	SVM multi class had edge over binary class in this study and also study proves that both frameworks produces the cancer stage and also how	86% ACCURACY with support vector machine

		(WTA-SVM winner-takes-all with support vector machine) with threshold technique (T-BMSVM) to classify nodules into malignant or benign nodules and also their malignancy levels respectively		much effect it is .	
6	Exploring Volatile Organic Compounds in Breath for High-Accuracy Prediction of Lung Cancer	eXtreme Gradient Boosting (XGBoost) algorithm	NCTU and NTUH -316 patients	It was the new technique to predict lung cancer using machine learning techniques	92% accuracy was achieved
7	A hybrid algorithm for lung cancer classification using SVM and Neural Networks	combination of support vector machine and feed-forward	dataset produced by Cornell University which had	The integration of svm and neural network had	The results shows that the proposed method had 98%

		back propagation neural network (KASC)	500 CT scan images .	made the model hybrid with high accuracy scores.	accuracy and also a rise of 3% with rise in number of samples from 100 to 500.F-measure for KASC shows an average value of 97% and recall value of 96.5%
8	Lung Cancer Prediction Using Stochastic Difusion Search (SDS) Based Feature Selection and Machine Learning Methods	Method used stochastic diffusion search algorithm for optimal features and also neural networks like Naive bayes, decision tree were used	dataset of 270 patients were recorded and it was produced by TCGA	BY using feature extraction , the classification of lung cancer had high score of prediction and SDS played a major role in this method for high accuracy	SDS-NN by 2.51% for SDS-decision tree and by 1.25% for SDS-naive bayes
9	Deep Learning for Lung Cancer	three-dimensional	1183 patients were	The method defines the	5% CI with AUC and

	Prediction using NSCLS patients CT Information	CONVOLUTION NEURAL NETWORK	collected for study by NSCLC	patience for 2years clinical trials had proved the high accuracy.	Surgical treatment and it has 0.91 correlation facto
10	Lung Cancer Detection and Classification using Machine Learning Algorithm	logistic regression, naive Bayes, support vector machine, random forest, Gradient Boosting	1)TCIA 2) LIDC-IDRI 3)kaggle 1595 patients	System used to compare the insights between both neural networks and classifiers with radiologists.	41% high precision for classifiers than 1-2% by radiologists

CHAPTER 3

DESIGN AND REQUIREMENTS

3.1 REQUIREMENTS

3.1.1 Hardware Requirements:

- Operating system: windows, Linux
- Processor: minimum intel i3
- Ram: Minimum 4gb
- Hard disk: 250gb and more or SSD

.

3.1.2 Software requirements:

We have used various modules and

libraries to make this process easy:

- SciPy
- NumPy
- Matplotlib
- Pandas
- Seaborn
- Python IDE
- Jupyter Notebook/ Collaboratory

3.2DETAILED DESIGN:

3.2.1 System Architecture:

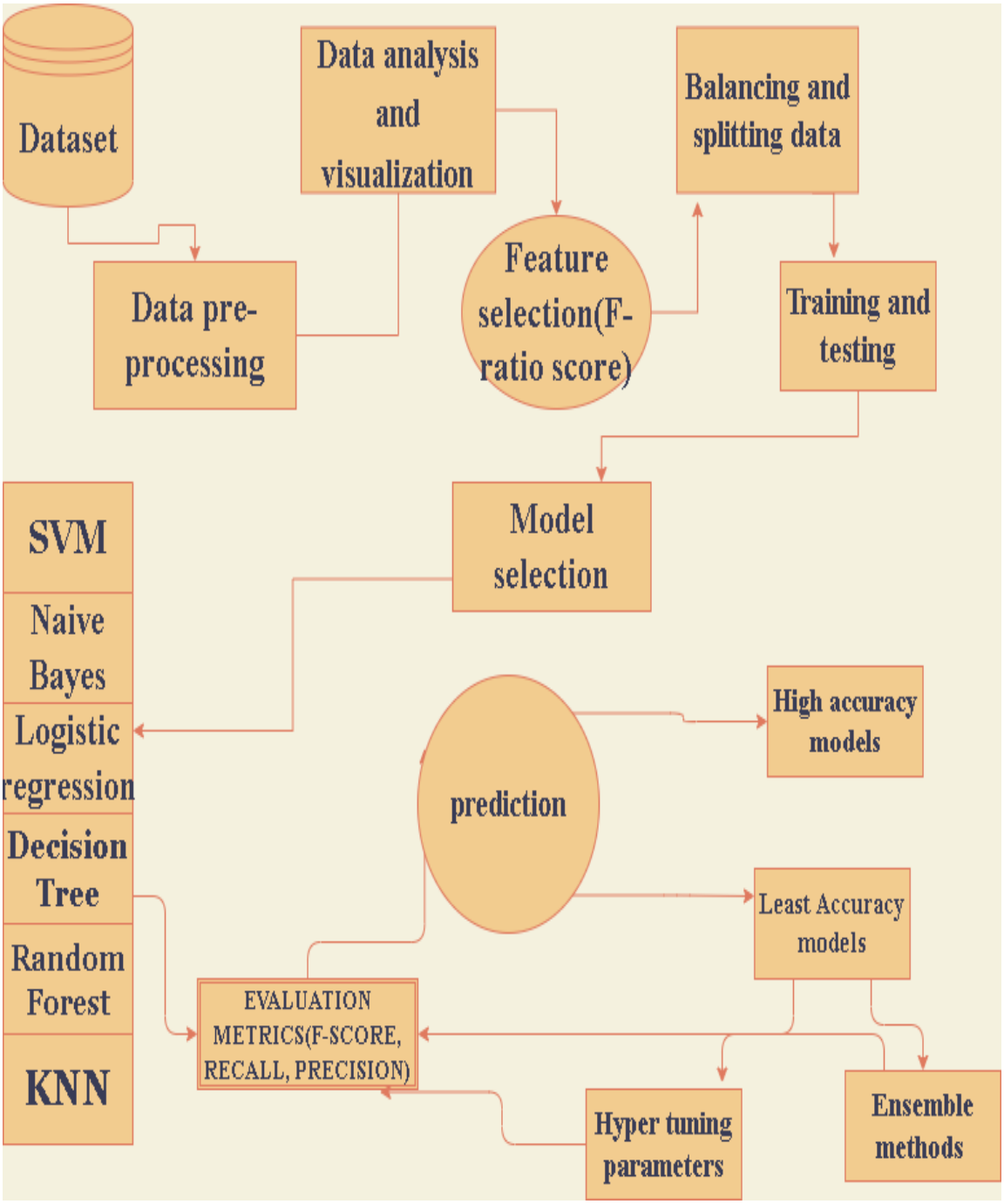


Fig 1 System architecture

3.2.2 USE CASE

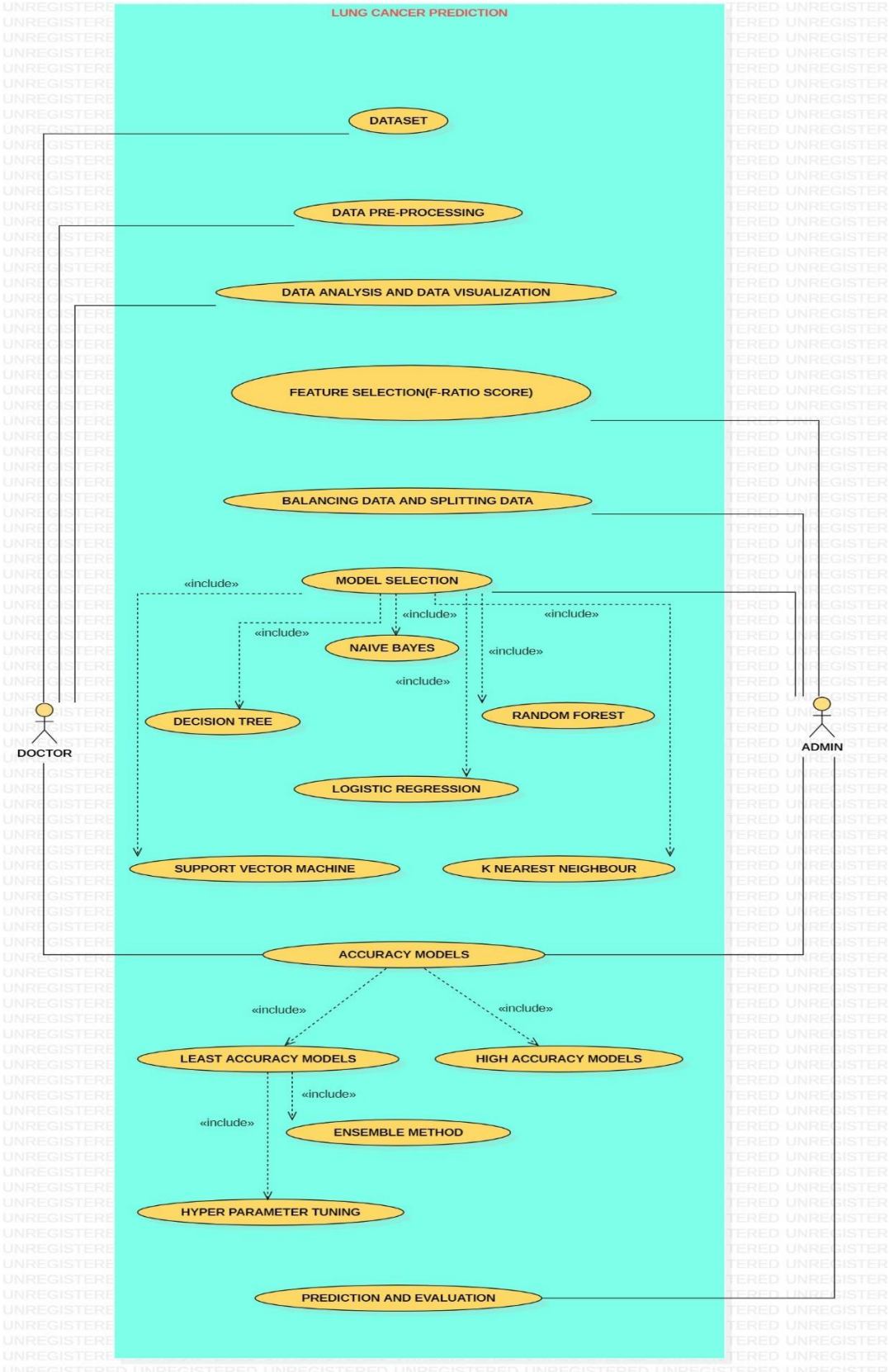


Fig 2 Use case diagram

3.2.3 ACTIVITY

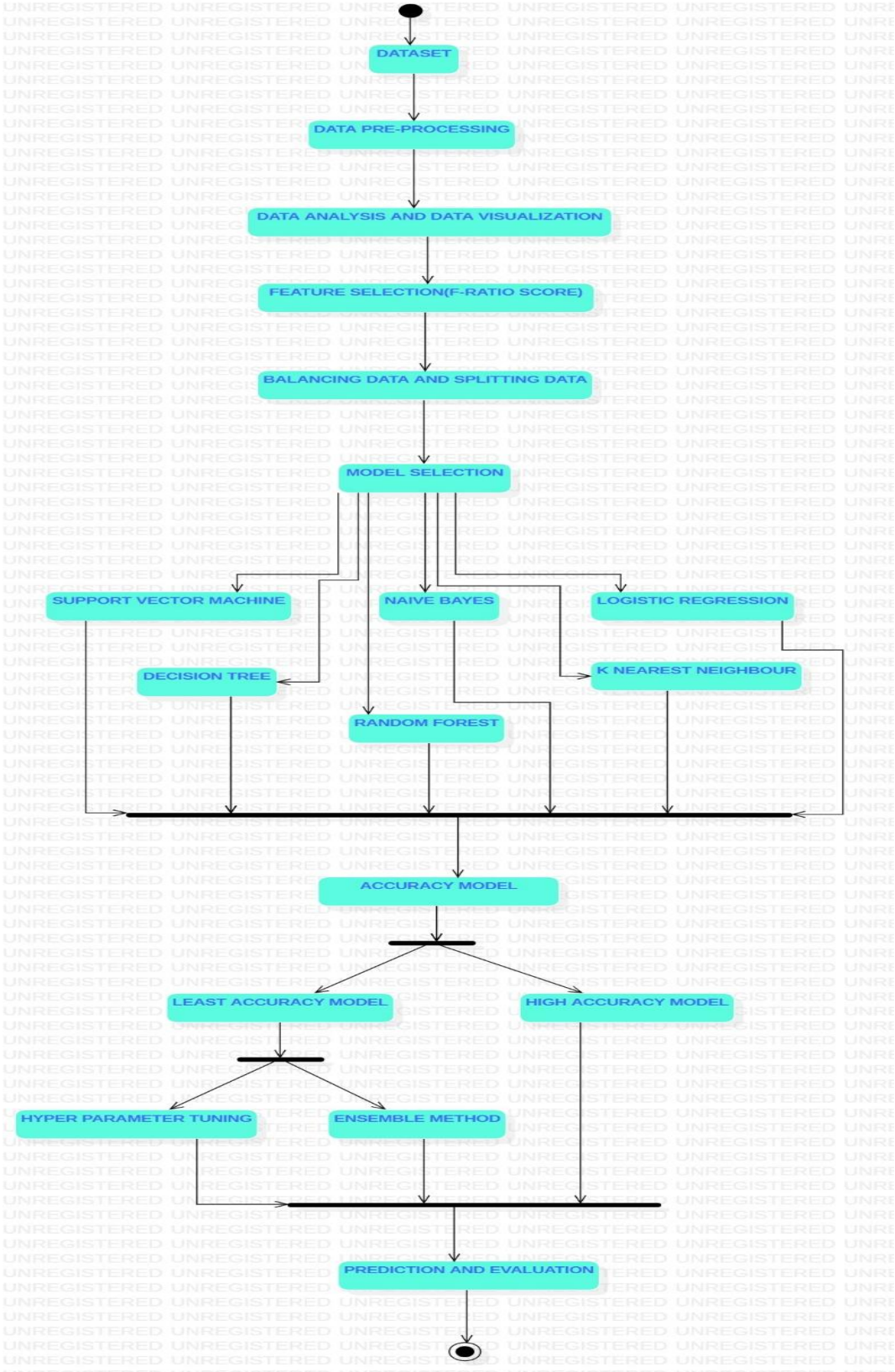


Fig 3 Activity diagram

3.2.4 DATA FLOW DIAGRAM

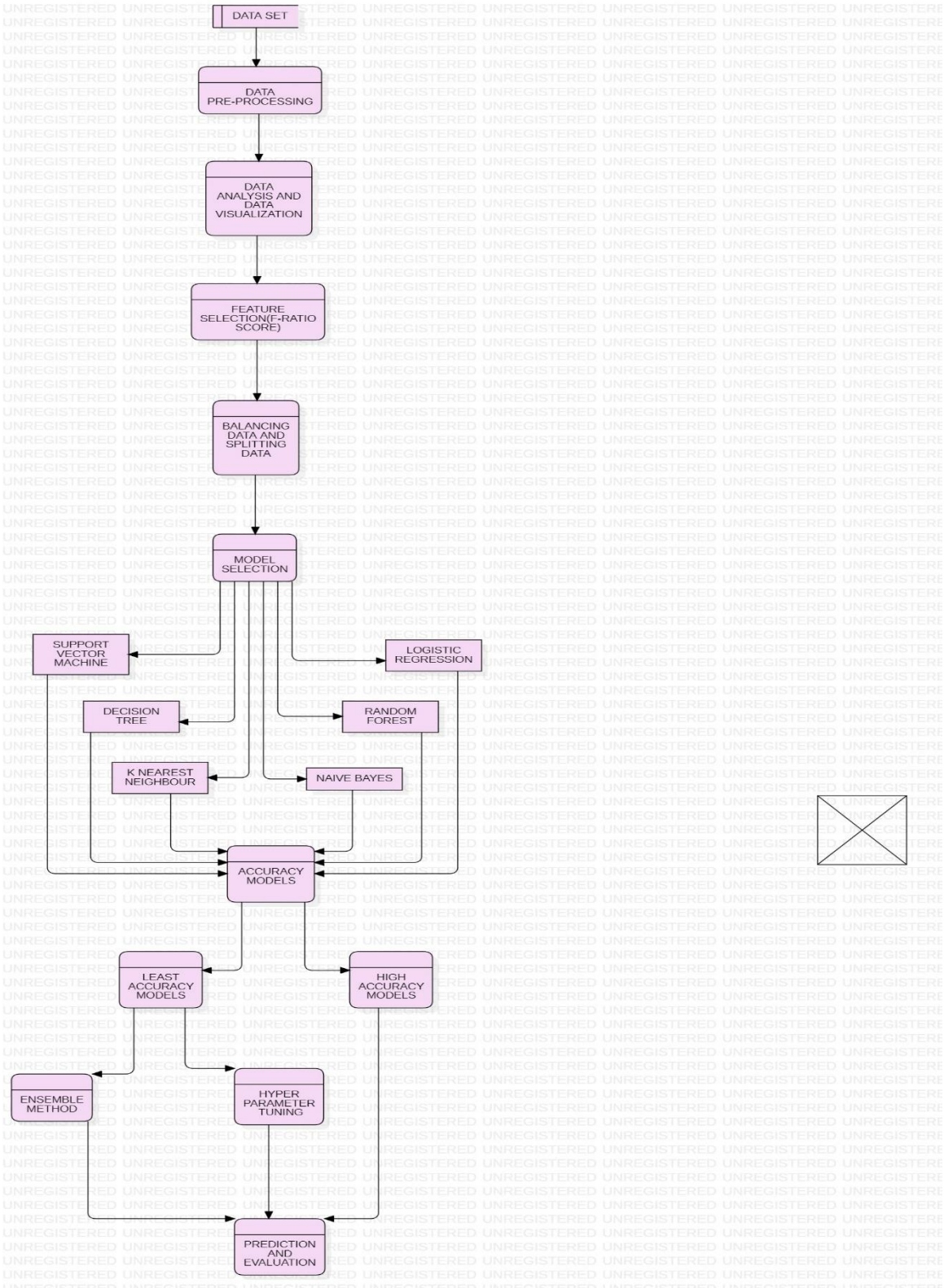


Fig 4 Data flow diagram

CHAPTER 4

DESCRIPTION OF PROPOSED METHODOLOGIES

4.1 Data collection

There are various types of the cancer like lung, breast, prostrate, carnival etc. Each type of cancer has specific symptoms. Based on the symptoms the type of the cancer is predicted. Dataset used should be more precise and accurate in order to improve the predictive accuracy of machine learning algorithms. We have taken our datasets from Kaggle (Cancer Patients Dataset). This dataset contains the attributes that are taken into consideration for lung cancer prediction.

4.2 Description of Attributes

Symptoms are considered as the attributes and diagnosis is done using Machine Learning techniques. Here we consider 25 attributes with 1000 instances Or 1000 patients.

The attributes taken into consideration are:

- Age: age is a effective attribute to the system and dataset consists of all age groups from 16-70 and the mean age group is 37 from the dataset.

- Gender: gender has role in lung cancer where the research shows women has high chances. Dataset consists of both gender where male was 598 and female were 402.

- Air Pollution: air pollution has high contribution in lung cancer where research shows that air pollution is directly linked to risk of death and increase of each unit of air pollution leads to 8% increase in risk of lung cancer.[14]

- Dust Allergy: Dust allergy is an attribute where it is related to lung cancer for few people like breath in dust causes a lung disease called hypersensitivity

pneumonitis, it has symptoms like coughing and shortness of breath.[15]

- Alcohol: Alcohol is an attribute which is linked to all types of cancer but there is no perfect research to prove it is directly linked to lung cancer and also few studies show that if a man consumes 21 alcohol drinks per week then it may increase risk for lung cancer.[16]

- Occupational Hazards: International agency for research on lung cancer has identified 12 occupational factors to human lung(aluminium production, arsenic, asbestos, bis-chloromethyl ether, beryllium, cadmium, hexavalent chromium, coke and coal gasification fumes, crystalline silica, nickel, radon and soot).[17]

- Genetic Risk: The research shows 8% of lung cancer cases are genetic predisposition [18], genetic risk shows a direct relation-ship to lung cancer.

- Chronic Lung Disease: Chronic disease had a linear relation ship with lung cancer where research shows that chronic obstructive pulmonary disease is an important risk factor for lung cancer [19].

- Coughing of blood: it is not the symptom for identifying a lung cancer , but according to American cancer society coughing of blood occurs at advance stage of lung cancer.[20].

- Fatigue: Research shows that it is most frequently reported symptom in lung cancer according to CRF and fatigue can belongs to cluster pain, depression and also insomnia [21].

- Weight loss: normal weight loss doesn't indicate a symptom for lung cancer but unexplained weight loss is the first noticeable symptom for lung cancer according to American cancer society [22].

- Shortness of breath: it is an early symptom for lung cancer patients and as it advances patient may cause inflammation.

- Wheezing: is a symptom of lung cancer only it represents with a combination of other symptoms like shortness of breath.

- Swallowing difficulty: is also called dysphagia, according to national cancer reports 1-2% of lung cancers had dysphagia [23].

- Balanced Diet: there is no evidence or direct relationship that a diet causes lung cancer.

- Obesity: it causes compression of lungs which leads to pulmonary damage for lung cancer [24].

- Smoking and passive smoker: smoking plays avital role in lung cancer and also according to reports 80-90% of lung cancer deaths are linked to smoking.

- Chest pain: is one of the important symptoms for lung cancer, the discomfort of chest pain may result enlarged lymph nodes and it spreads to your bones and results pain all over body.[25]

- Clubbing of fingernails: it is the most often symptom in both lung and heart disease and also it reduces oxygen in blood.

- Frequent cold: This attribute is not a significant but it had presence for lung cancer.

- Dry cough: is the common symptom for lung cancer and it is an early warning and also as cancer increases dry cough increases with it.

- Snoring: is also a significant symptom to lung cancer with daytime sleepiness and it was high in lung cancer patients.

4.3 **Data Analysis and Visualization:**

We have used various modules and libraries in order to make this process easy:

- SciPy
- NumPy
- matplotlib
- pandas
- sklearn
- seaborn

And then we performed Data Visualization to classify patients with respect to the attributes we have considered for lung cancer prediction.

4.3.1 Data visualization:

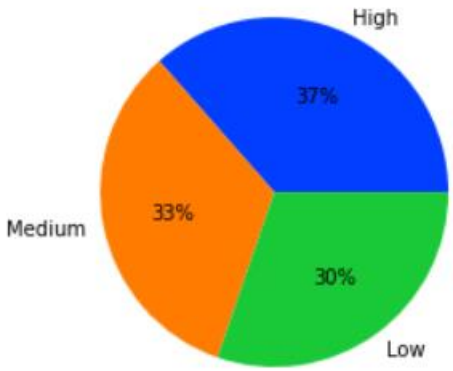


Fig 5 representing the level of risk of patients

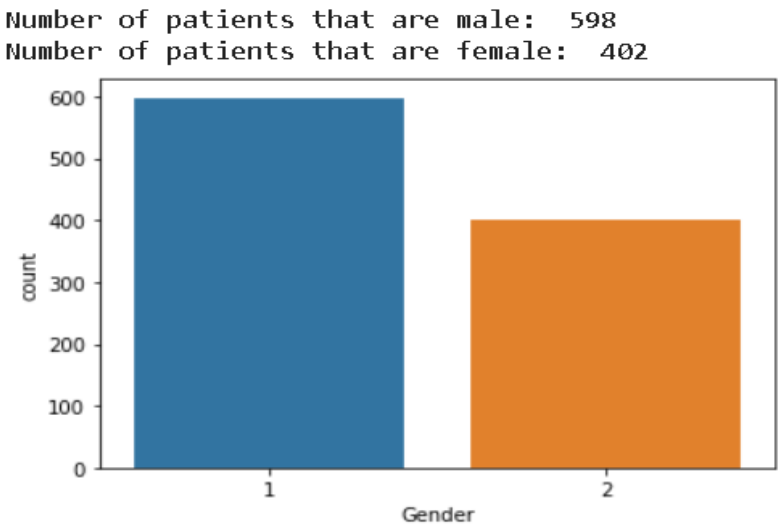


Fig 6 Male vs Female in cancer data set

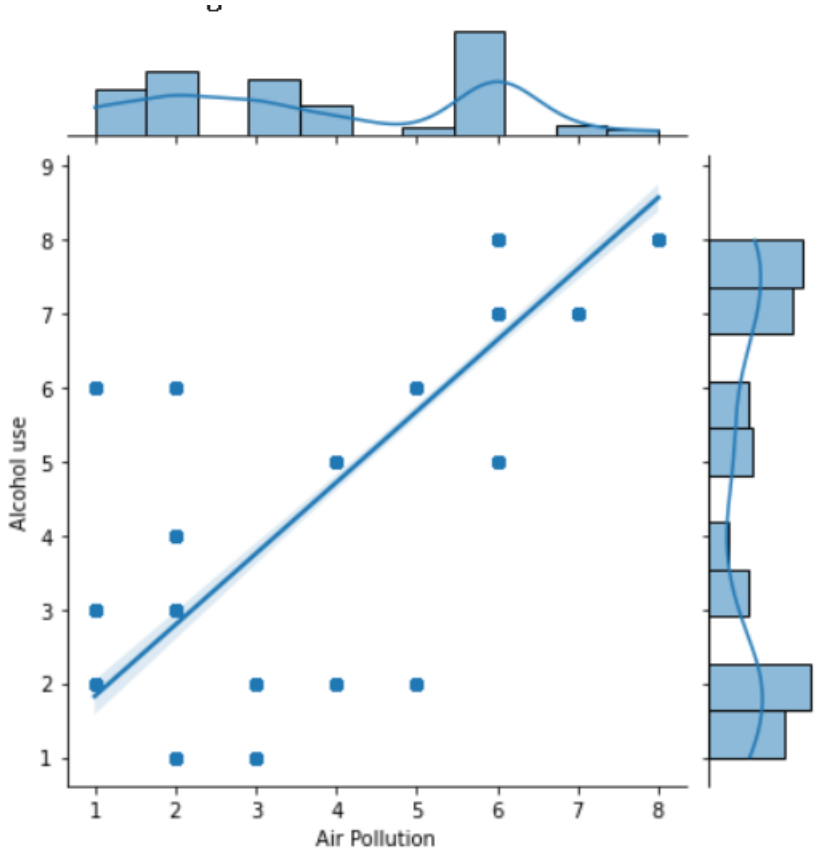


Fig 7 Alcohol vs air pollution

Here It states that there is no linear relationship between air pollution and alcohol

4.4 Feature Selection:

Though there are a lot of variables to look at we can just find the most important ones by using the Select K Best Algorithm with ANOVA F-ratio statistics

Feature Selection is a method through which we can generate the F-ratio scores of all features and we can determine which ones to use for machine learning. Hence using feature selection, we have confined our dataset to certain attributes which obtained from the results of this process.

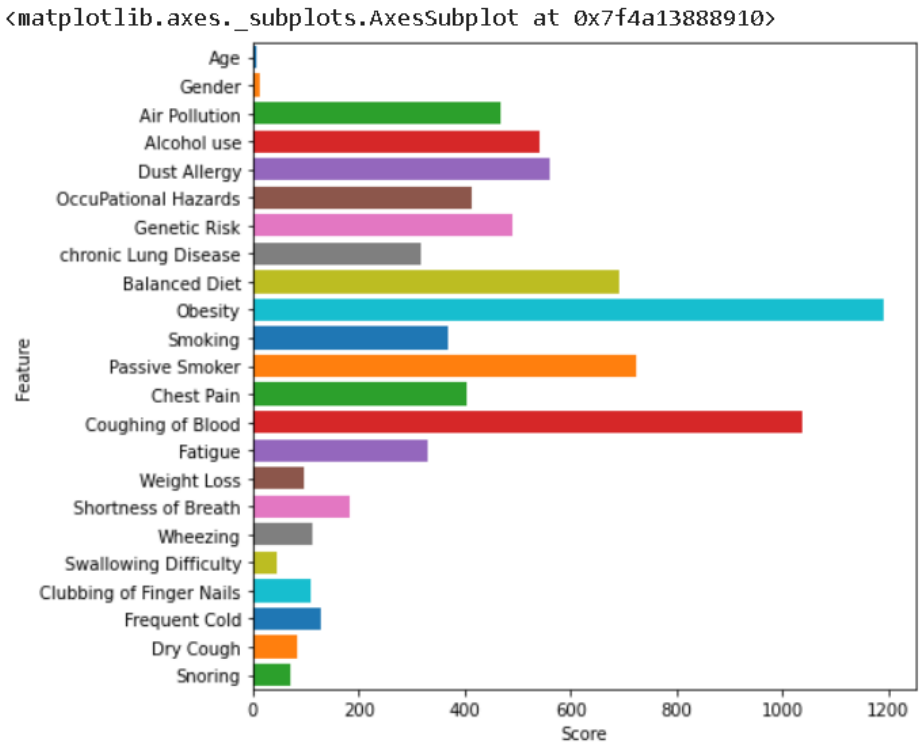


Fig 8 feature selection

4.5 Model Selection:

Since this is a Classification problem various Supervised Machine learning algorithm can be used. For this project the algorithms which we chose were Logistic Regression, Support Vector Machine and Naïve Bayes, decision tree , random forest, KNN, Hyper parameter optimization and ensemble method techniques.

SVM: In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting) [11]

Naïve Bayes: In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian

network models. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. [12]

Logistic Regression: In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one. [13]

KNN: *K*-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.[26]

Decision Tree: A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). [27]

Random Forest : Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.[28]

Hyper parameter tuning: In machine learning, hyperparameter optimization[1] or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters and have to be tuned so that the model can optimally solve the

machine learning problem.[29]

Max-voting ensemble: Every model makes a prediction (votes) for each test instance and the final output prediction is the one that receives more than half of the votes. If none of the predictions get more than half of the votes, we may say that the ensemble method could not make a stable prediction for this instance. Although this is a widely used technique, you may try the most voted prediction (even if that is less than half of the votes) as the final prediction.[30]

4.6 Evaluation of Models:

In the project every model is implemented using Python's Scikit-Learn library.
All the models are evaluated in the following procedure:

- Importing all the libraries necessary for each model
- Importing the Dataset
- Data preprocessing:

This step involves the following two steps:

i. Dividing the data into attributes and labels
ii. Dividing the data into training and testing sets.

We have train test split method in the Scikit-Learn library which allows us to comfortably divide data into training and test sets.

Training the Algorithm:

Once we have divided the data into training and testing sets, it's time to train our models on the training sets. The Scikit-Learn has a certain libraries like svm, sklearn.neighbors. These libraries have certain built-in classes to make our tasks easy

Making predictions:

We now finally come to the end of the process where we make prediction. This becomes possible using predict method from the sklearn libraries.

Evaluating the Algorithm:

For evaluating the algorithms, we obtain a confusion matrix. The Scikit-Learns metrics has certain methods like classification report and confusion matrix using which we can obtain the values of these metrics(F1-call, recall, precision, confusion matrix.

CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

```
# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
import warnings
warnings.filterwarnings("ignore")
```

```
Python: 3.7.13 (default, Mar 16 2022, 17:37:17)
[GCC 7.5.0]
scipy: 1.4.1
numpy: 1.21.6
matplotlib: 3.2.2
pandas: 1.3.5
sklearn: 1.0.2
```

```
#Import all required libraries for reading data, analysing and visualizing data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
```

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving cancer patient data sets.xlsx to cancer patient data sets.xlsx
```

```
#Read the training & test data
import io
lung_df = pd.read_excel('cancer patient data sets.xlsx')

#Pandas head() method is used to return top n (5 by default) rows of a data frame or series.
lung_df.head()
```

```
  Patient Id  Age  Gender  Air Pollution  Alcohol use  Dust Allergy  \
0         P1   33       1              2            4             5
1        P10   17       1              3            1             5
2       P100   35       1              4            5             6
3      P1000   37       1              7            7             7
4       P101   46       1              6            8             7

   OccuPational Hazards  Genetic Risk  chronic Lung Disease  Balanced Diet  \
0                     4             3                     2              2
1                     3             4                     2              2
2                     5             5                     4              6
3                     7             6                     7              7
4                     7             7                     6              7

   ...  Fatigue  Weight Loss  Shortness of Breath  Wheezing  \
0  ...        3            4                    2         2
1  ...        1            3                    7         8
2  ...        8            7                    9         2
3  ...        4            2                    3         1
4  ...        3            2                    4         1

   Swallowing Difficulty  Clubbing of Finger Nails  Frequent Cold  Dry Cough  \
0                      3                         1              2          3
1                      6                         2              1          7
2                      1                         4              6          7
3                      4                         5              6          7
4                      4                         2              4          2

   Snoring   Level
0        4     Low
1        2  Medium
2        2    High
3        5    High
4        3    High

[5 rows x 25 columns]
```

```
#This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.
lung_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 25 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Patient Id             1000 non-null   object
 1   Age                    1000 non-null   int64
 2   Gender                 1000 non-null   int64
 3   Air Pollution          1000 non-null   int64
 4   Alcohol use            1000 non-null   int64
 5   Dust Allergy           1000 non-null   int64
 6   OccuPational Hazards   1000 non-null   int64
 7   Genetic Risk           1000 non-null   int64
 8   chronic Lung Disease   1000 non-null   int64
 9   Balanced Diet          1000 non-null   int64
 10  Obesity                1000 non-null   int64
 11  Smoking                1000 non-null   int64
 12  Passive Smoker         1000 non-null   int64
 13  Chest Pain             1000 non-null   int64
 14  Coughing of Blood      1000 non-null   int64
 15  Fatigue                1000 non-null   int64
 16  Weight Loss            1000 non-null   int64
 17  Shortness of Breath    1000 non-null   int64
 18  Wheezing               1000 non-null   int64
 19  Swallowing Difficulty  1000 non-null   int64
 20  Clubbing of Finger Nails  1000 non-null   int64
 21  Frequent Cold          1000 non-null   int64
 22  Dry Cough              1000 non-null   int64
 23  Snoring                1000 non-null   int64
 24  Level                  1000 non-null   object
dtypes: int64(23), object(2)
memory usage: 195.4+ KB
```

```
#Describe gives statistical information about NUMERICAL columns in the dataset
lung_df.describe(include='all')
```

```
       Patient Id          Age       Gender  Air Pollution  Alcohol use  \
count        1000  1000.000000  1000.000000      1000.0000  1000.000000
unique       1000          NaN          NaN            NaN          NaN
top            P1          NaN          NaN            NaN          NaN
freq            1          NaN          NaN            NaN          NaN
mean          NaN    37.174000     1.402000         3.8400     4.563000
std           NaN    12.005493     0.490547         2.0304     2.620477
min           NaN    14.000000     1.000000         1.0000     1.000000
25%           NaN    27.750000     1.000000         2.0000     2.000000
50%           NaN    36.000000     1.000000         3.0000     5.000000
75%           NaN    45.000000     2.000000         6.0000     7.000000
max           NaN    73.000000     2.000000         8.0000     8.000000

       Dust Allergy  OccuPational Hazards  Genetic Risk  \
```

count	1000.000000	1000.000000	1000.000000
unique	NaN	NaN	NaN
top	NaN	NaN	NaN
freq	NaN	NaN	NaN
mean	5.165000	4.840000	4.580000
std	1.980833	2.107805	2.126999
min	1.000000	1.000000	1.000000
25%	4.000000	3.000000	2.000000
50%	6.000000	5.000000	5.000000
75%	7.000000	7.000000	7.000000
max	8.000000	8.000000	7.000000

	chronic Lung Disease	Balanced Diet	...	Fatigue	Weight Loss	\
count	1000.000000	1000.000000	...	1000.000000	1000.000000	
unique	NaN	NaN	...	NaN	NaN	
top	NaN	NaN	...	NaN	NaN	
freq	NaN	NaN	...	NaN	NaN	
mean	4.380000	4.491000	...	3.856000	3.855000	
std	1.848518	2.135528	...	2.244616	2.206546	
min	1.000000	1.000000	...	1.000000	1.000000	
25%	3.000000	2.000000	...	2.000000	2.000000	
50%	4.000000	4.000000	...	3.000000	3.000000	
75%	6.000000	7.000000	...	5.000000	6.000000	
max	7.000000	7.000000	...	9.000000	8.000000	

	Shortness of Breath	Wheezing	Swallowing Difficulty	\
count	1000.000000	1000.000000	1000.000000	
unique	NaN	NaN	NaN	
top	NaN	NaN	NaN	
freq	NaN	NaN	NaN	
mean	4.240000	3.777000	3.746000	
std	2.285087	2.041921	2.270383	
min	1.000000	1.000000	1.000000	
25%	2.000000	2.000000	2.000000	
50%	4.000000	4.000000	4.000000	
75%	6.000000	5.000000	5.000000	
max	9.000000	8.000000	8.000000	

	Clubbing of Finger Nails	Frequent Cold	Dry Cough	Snoring	\
count	1000.000000	1000.000000	1000.000000	1000.000000	
unique	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	
mean	3.923000	3.536000	3.853000	2.926000	
std	2.388048	1.832502	2.039007	1.474686	
min	1.000000	1.000000	1.000000	1.000000	
25%	2.000000	2.000000	2.000000	2.000000	
50%	4.000000	3.000000	4.000000	3.000000	
75%	5.000000	5.000000	6.000000	4.000000	
max	9.000000	7.000000	7.000000	7.000000	

```
       Level
count   1000
unique     3
top     High
freq     365
mean     NaN
std      NaN
min      NaN
25%      NaN
50%      NaN
75%      NaN
max      NaN

[11 rows x 25 columns]
```

```
#Which features are available in the dataset?
lung_df.columns
```

```
Index(['Patient Id', 'Age', 'Gender', 'Air Pollution', 'Alcohol use',
       'Dust Allergy', 'OccuPational Hazards', 'Genetic Risk',
       'chronic Lung Disease', 'Balanced Diet', 'Obesity', 'Smoking',
       'Passive Smoker', 'Chest Pain', 'Coughing of Blood', 'Fatigue',
       'Weight Loss', 'Shortness of Breath', 'Wheezing',
       'Swallowing Difficulty', 'Clubbing of Finger Nails', 'Frequent Cold',
       'Dry Cough', 'Snoring', 'Level'],
      dtype='object')
```

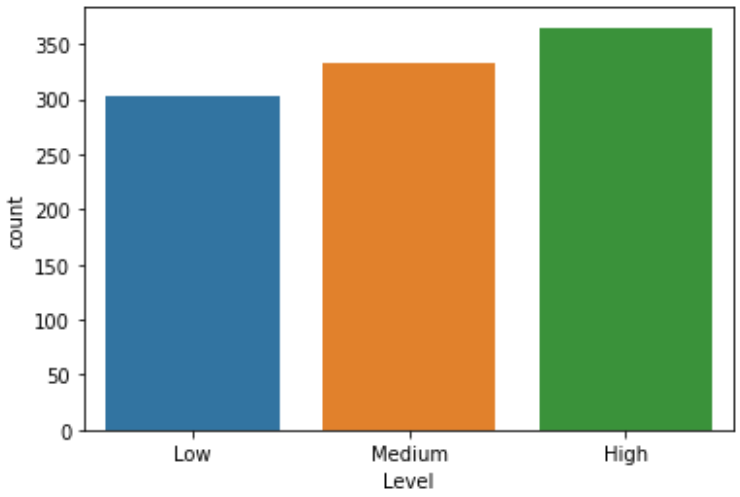
```
#Check for any null values
lung_df.isnull().sum()
```

```
Patient Id                0
Age                       0
Gender                    0
Air Pollution             0
Alcohol use               0
Dust Allergy              0
OccuPational Hazards      0
Genetic Risk              0
chronic Lung Disease      0
Balanced Diet             0
Obesity                   0
Smoking                   0
Passive Smoker            0
Chest Pain                0
Coughing of Blood         0
Fatigue                   0
Weight Loss               0
Shortness of Breath       0
Wheezing                  0
Swallowing Difficulty     0
```

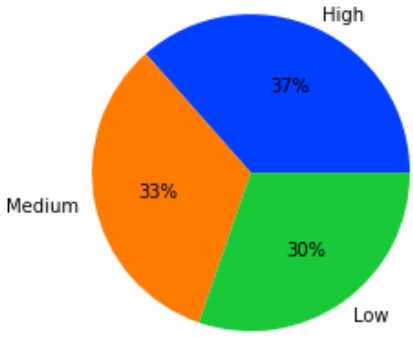
```
Clubbing of Finger Nails    0
Frequent Cold               0
Dry Cough                   0
Snoring                     0
Level                       0
dtype: int64
```

```
#Data Visualization with classification of people with lung disease with all levels of risk.
sns.countplot(data=lung_df, x = 'Level', label='Count')

LD, NLD,hld = lung_df['Level'].value_counts()
print('Number of patients diagnosed with lung disease with low risk: ',LD)
print('Number of patients not diagnosed with lung disease with average risk: ',NLD)
print('Number of patients not diagnosed with lung disease with high risk: ',hld)
```

```
Number of patients diagnosed with lung disease with low risk:  365
Number of patients not diagnosed with lung disease with average risk:  332
Number of patients not diagnosed with lung disease with high risk:  303
```

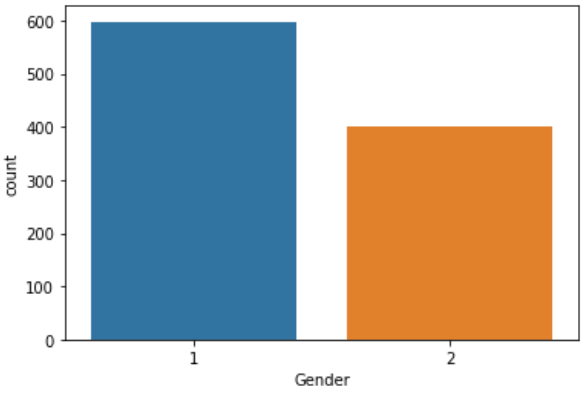



```
values = lung_df['Level'].value_counts().tolist()
names = list(dict(lung_df['Level'].value_counts()).keys())

colors = sns.color_palette('bright')
plt.figure()
plt.pie(values, labels=names,colors = colors, autopct = '%0.0f%%')
plt.show()
```

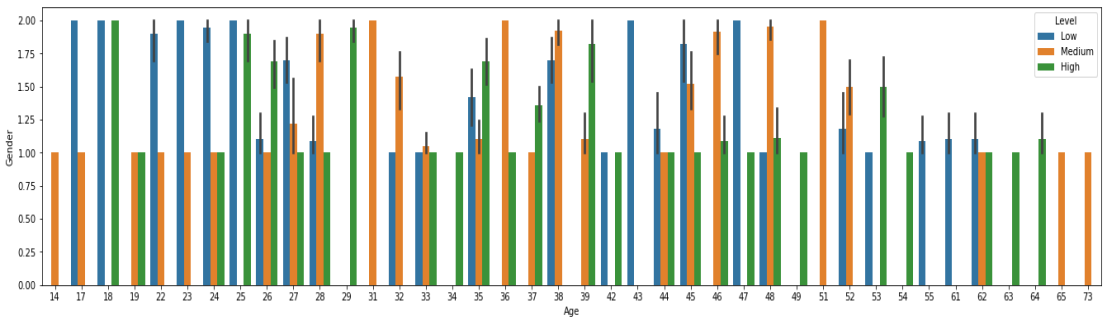



```
#datavisualized with male and female classification
sns.countplot(data=lung_df, x = 'Gender', label='Count')

M, F = lung_df['Gender'].value_counts()
print('Number of patients that are male: ',M)
print('Number of patients that are female: ',F)
```

Number of patients that are male: 598
Number of patients that are female: 402




```
#graph shown is   classifaction of age using male and female dataset
plt.figure(figsize=(23, 5))
sns.barplot(x="Age", y="Gender", hue="Level", data=lung_df);
#Age seems to be a factor for lung disease for both male and female genders
```



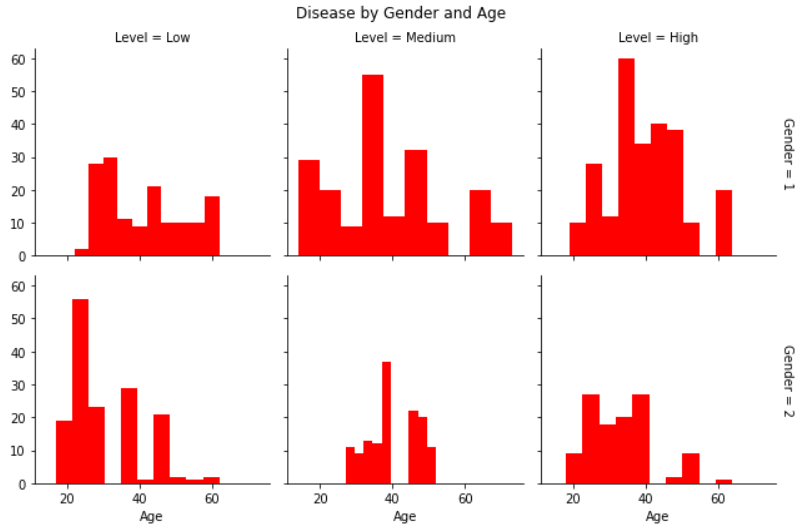

```
#data groups with lung patients of all levels of stages in cancer with gender
lung_df[['Gender', 'Level','Age']].groupby(['Level','Gender'],
as_index=False).count().sort_values(by='Level', ascending=False)
```

	Level	Gender	Age
4	Medium	1	197
5	Medium	2	135
2	Low	1	149
3	Low	2	154
0	High	1	252
1	High	2	113

```
lung_df[['Gender', 'Level','Age']].groupby(['Level','Gender'],
as_index=False).mean().sort_values(by='Level', ascending=False)
```

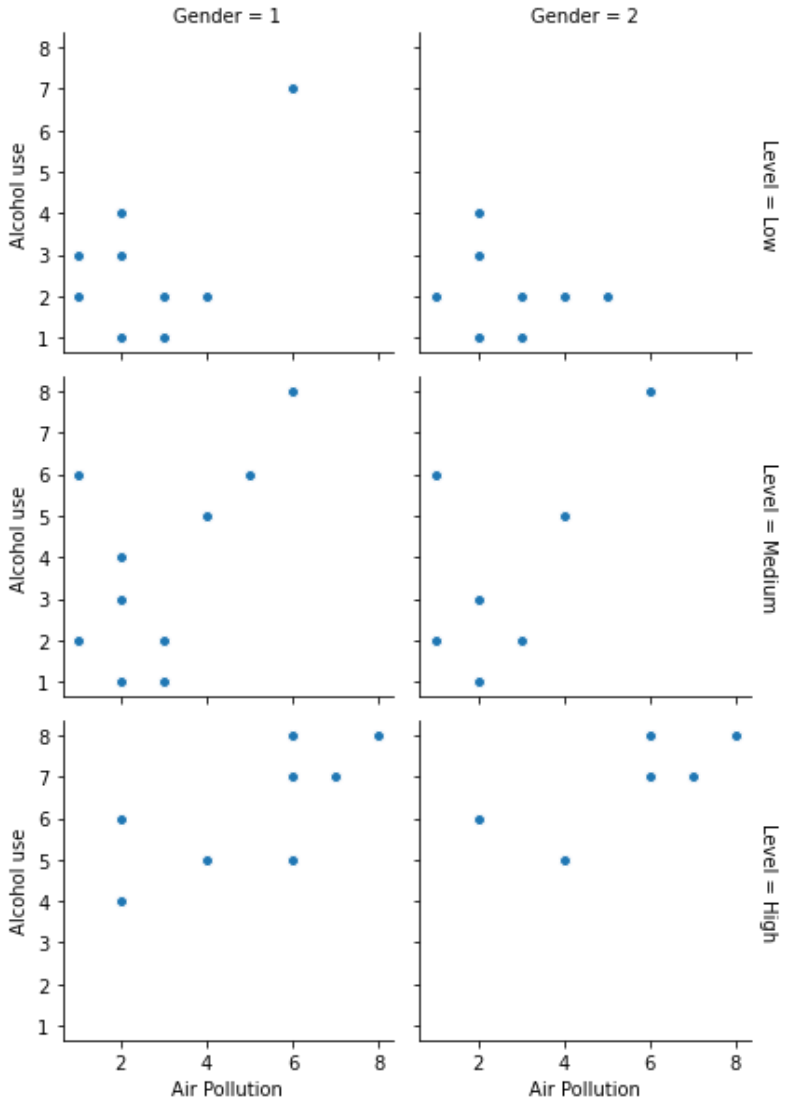
	Level	Gender	Age
4	Medium	1	37.827411
5	Medium	2	39.777778
2	Low	1	40.765101
3	Low	2	30.233766
0	High	1	39.257937
1	High	2	33.000000

```
g = sns.FacetGrid(lung_df, col="Level", row="Gender", margin_titles=True)
g.map(plt.hist, "Age", color="red")
plt.subplots_adjust(top=0.9)
g.fig.suptitle('Disease by Gender and Age');
```



#age is not to important on level of cancer.

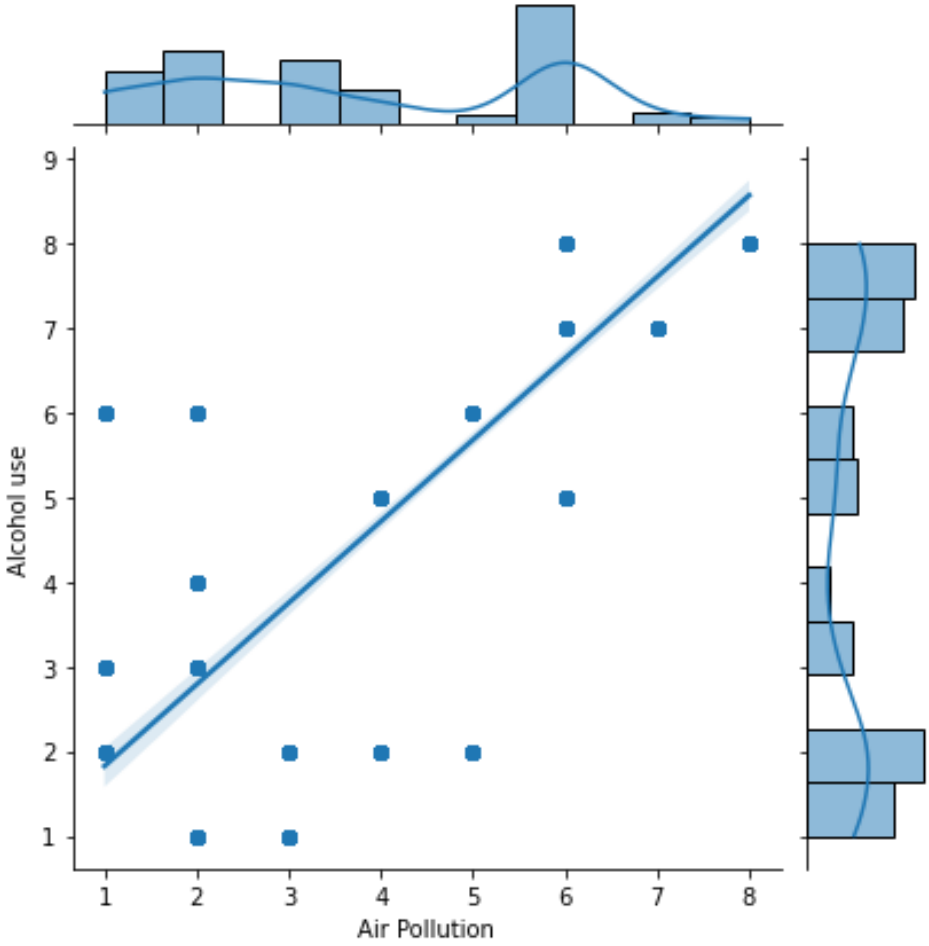
```
g = sns.FacetGrid(lung_df, col="Gender", row="Level", margin_titles=True)
g.map(plt.scatter,"Air Pollution", "Alcohol use", edgecolor="w")
plt.subplots_adjust(top=0.9)
```



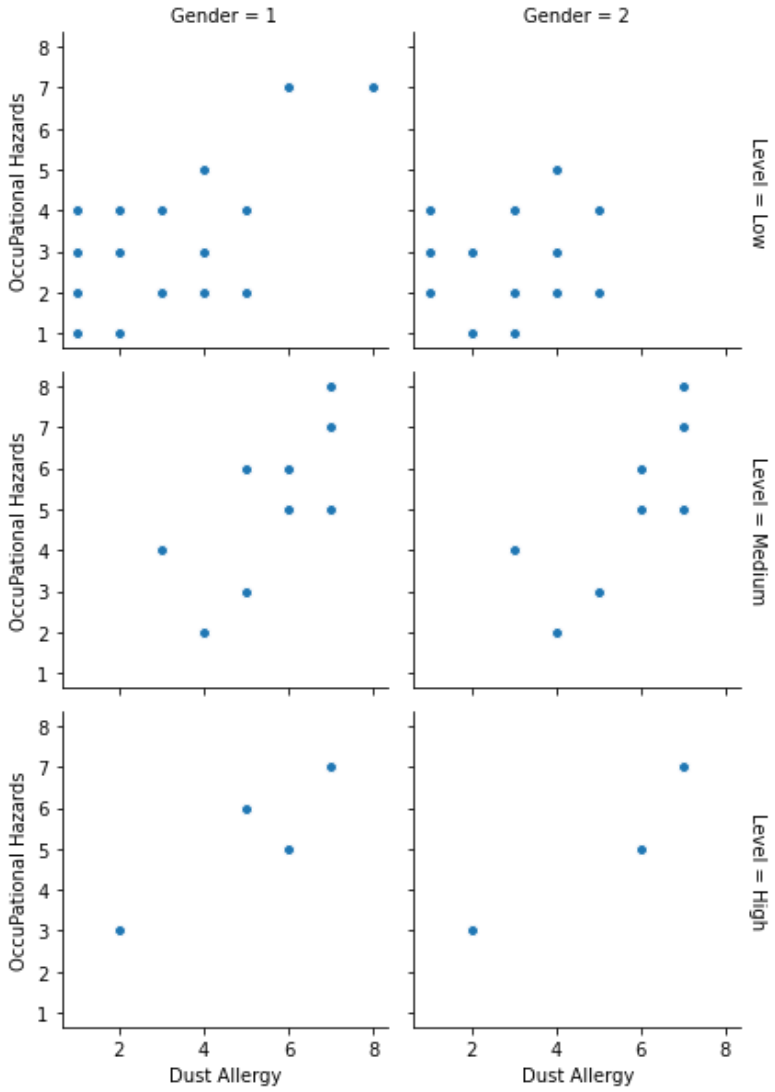
#There seems to be no direct relationship between Air Pollution and alcohol use.

```
sns.jointplot("Air Pollution", "Alcohol use", data=lung_df, kind="reg")
```

```
<seaborn.axisgrid.JointGrid at 0x7f8d9e0cced0>
```



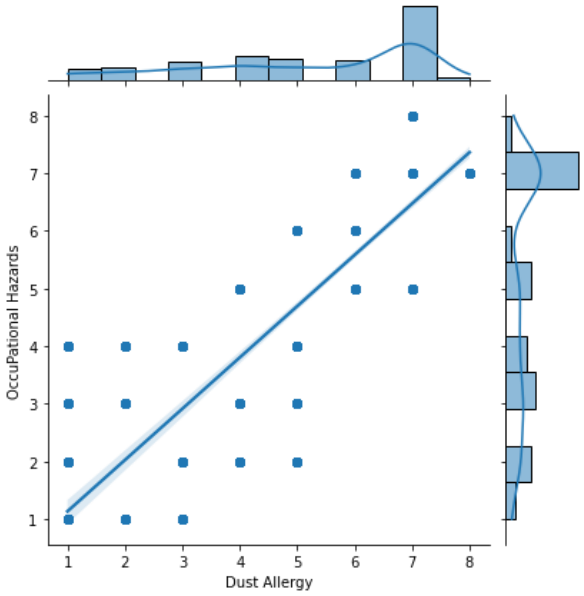

```
g = sns.FacetGrid(lung_df, col="Gender", row="Level", margin_titles=True)
g.map(plt.scatter,"Dust Allergy", "OccuPational Hazards",  edgecolor="w")
plt.subplots_adjust(top=0.9)
```



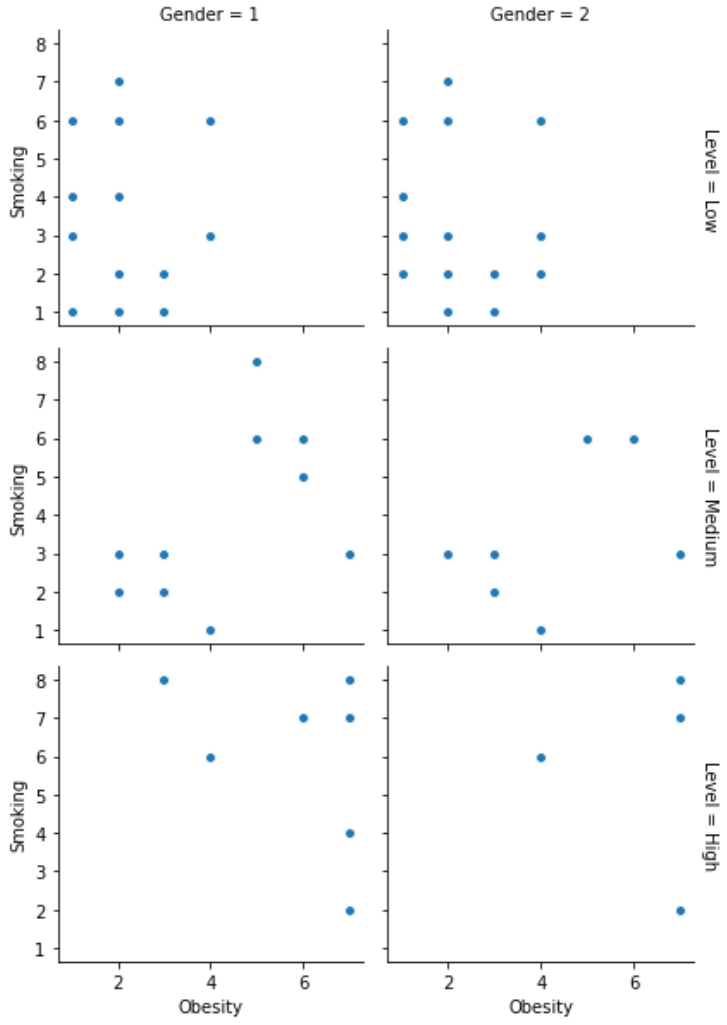
#There is no linear relationship between occupationhazards and dustallergy and the gender. .

```
sns.jointplot("Dust Allergy", "OccuPational Hazards", data=lung_df, kind="reg")
```

```
<seaborn.axisgrid.JointGrid at 0x7f8d9e1e96d0>
```

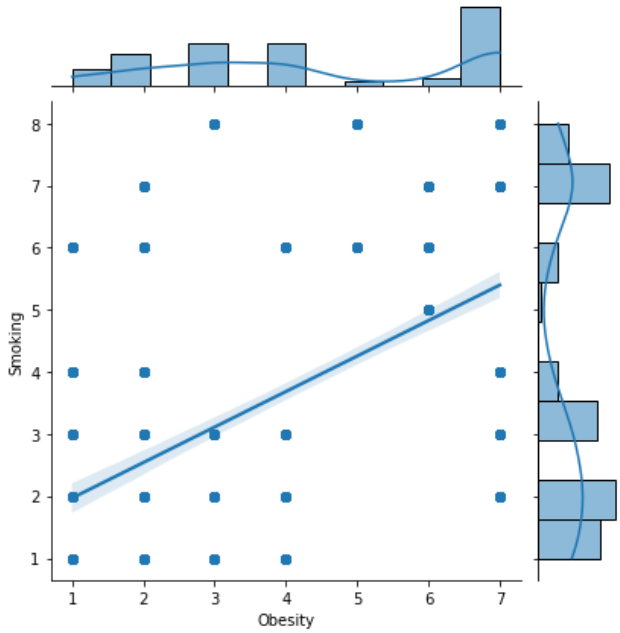



```
g = sns.FacetGrid(lung_df, col="Gender", row="Level", margin_titles=True)
g.map(plt.scatter,"Obesity", "Smoking",  edgecolor="w")
plt.subplots_adjust(top=0.9)
```




```
sns.jointplot("Obesity", "Smoking", data=lung_df, kind="reg")
```

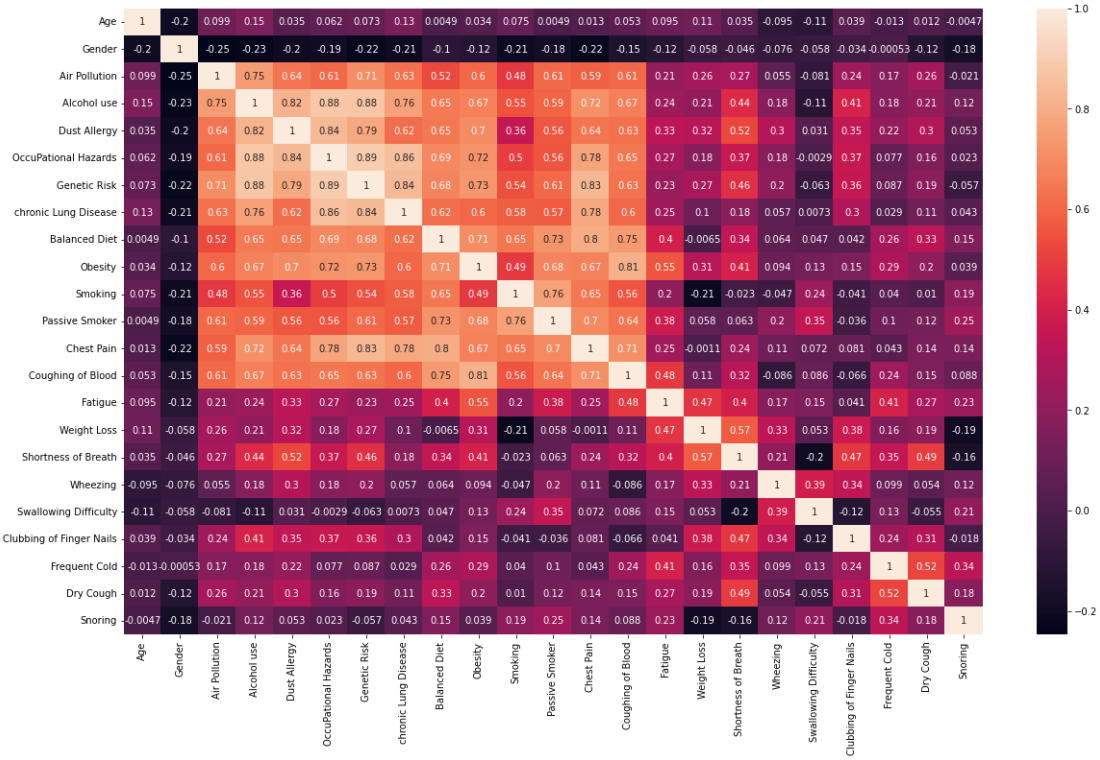
<seaborn.axisgrid.JointGrid at 0x7f8d9b701410>



#No linear correlation between obesity and Smoking

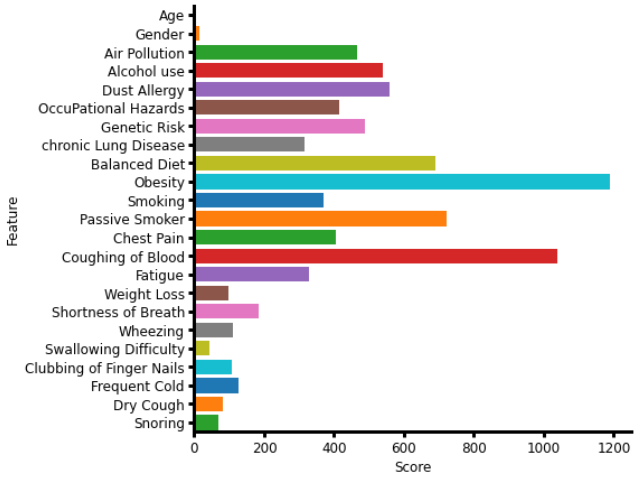
```
plt.figure(figsize=(20,12))
sns.heatmap(lung_df.corr(),annot= True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8d9b6b7cd0>



```
#Though there are a lot of variables to look at we can we can just find the most important ones by using the SelectKBest Algorithm with ANOVA F-ratio statistic

#Feature Selection
#This method will generate the F-ratio scores of all features and we can determine which ones to use for machine learning.

from sklearn.feature_selection import SelectKBest #Feature Selector
from sklearn.feature_selection import f_classif #F-ratio statistic for categorical values


#Feature Selection
X=lung_df.drop(['Level','Patient Id'], axis=1)
Y=lung_df['Level']
bestfeatures = SelectKBest(score_func=f_classif, k='all')
fit = bestfeatures.fit(X,Y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Feature','Score']  #naming the dataframe columns

#Visualize the feature scores
fig, ax=plt.subplots(figsize=(7,7))
plot=sns.barplot(data=featureScores, x='Score', y='Feature', palette='tab10',linewidth=0.5, saturation=2, orient='h')
Plotter(plot, 'Score', 'Feature', legend=False, save=True, save_name='Feature Importance.png')#Plotter function for aesthetics
plot
```

```
No handles with labels found to put in legend.

<matplotlib.axes._subplots.AxesSubplot at 0x7f4a135d0390>
```



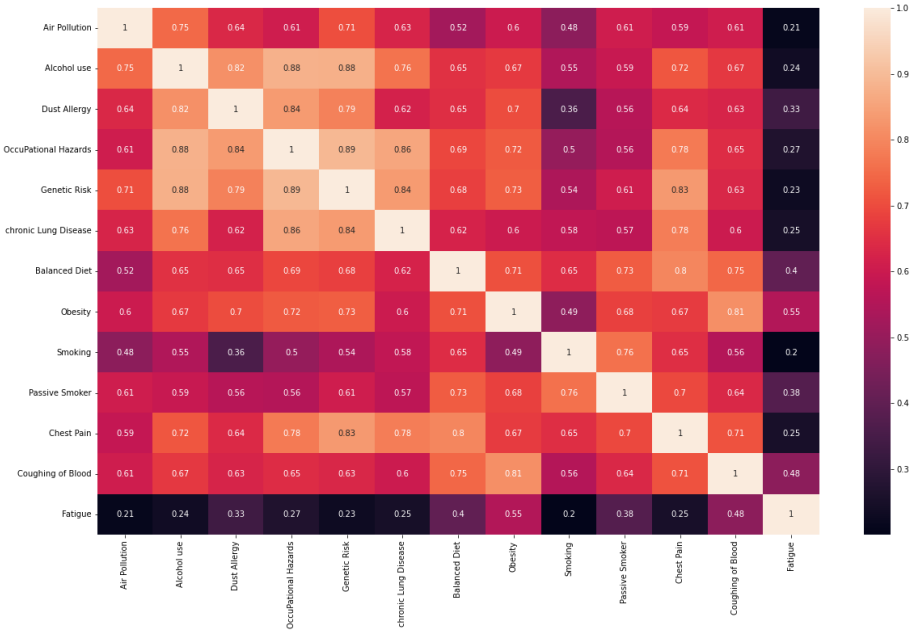

```
#We will take all the features that scored more than 200 as they show the least redundancy.

#Selection method
selection=featureScores[featureScores['Score']>=200]#Selects features that scored more
```

```
than 200
selection=list(selection['Feature'])#Generates the features into a list
selection.append('Level')#Adding the Level string to be used to make new data frame
new_data=lung_df[selection] #New dataframe with selected features.
new_data.head(1000) #Lets take a look at the first 1000.
```

```
     Air Pollution  Alcohol use  Dust Allergy  OccuPational Hazards  \
0              2         4          5                 4
1              3         1          5                 3
2              4         5          6                 5
3              7         7          7                 7
4              6         8          7                 7
..           ...       ...        ...               ...
995            6         7          7                 7
996            6         8          7                 7
997            4         5          6                 5
998            6         8          7                 7
999            6         5          6                 5

     Genetic Risk  chronic Lung Disease  Balanced Diet  Obesity  Smoking  \
0              3                     2              2        4        3
1              4                     2              2        2        2
2              5                     4              6        7        2
3              6                     7              7        7        7
4              7                     6              7        7        8
..           ...                   ...            ...      ...      ...
995            7                     6              7        7        7
996            7                     6              7        7        7
997            5                     4              6        7        2
998            7                     6              7        7        8
999            5                     4              6        7        2

     Passive Smoker  Chest Pain  Coughing of Blood  Fatigue   Level
0                 2           2                  4        3     Low
1                 4           2                  3        1  Medium
2                 3           4                  8        8    High
3                 7           7                  8        4    High
4                 7           7                  9        3    High
..              ...         ...                ...      ...     ...
995               8           7                  7        5    High
996               8           7                  7        9    High
997               3           4                  8        8    High
998               7           7                  9        3    High
999               3           4                  8        8    High

[1000 rows x 14 columns]
```

```
plt.figure(figsize=(20,12))
sns.heatmap(new_data.corr(),annot= True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d98ee2250>
```




```
X = new_data.drop(['Level'], axis=1)
X.head()
```

	Air Pollution	Alcohol use	Dust Allergy	OccuPational Hazards	\
0	2	4	5	4	
1	3	1	5	3	
2	4	5	6	5	
3	7	7	7	7	
4	6	8	7	7	

	Genetic Risk	chronic Lung Disease	Balanced Diet	Obesity	Smoking	\
0	3	2	2	4	3	
1	4	2	2	2	2	
2	5	4	6	7	2	
3	6	7	7	7	7	
4	7	6	7	7	8	

	Passive Smoker	Chest Pain	Coughing of Blood	Fatigue
0	2	2	4	3
1	4	2	3	1
2	3	4	8	8
3	7	7	8	4
4	7	7	9	3

```
y = new_data['Level']
y.head()
```

```
0       Low
1    Medium
2      High
3      High
4      High
Name: Level, dtype: object
```

Machine learning

```
# Importing modules
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
BaggingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

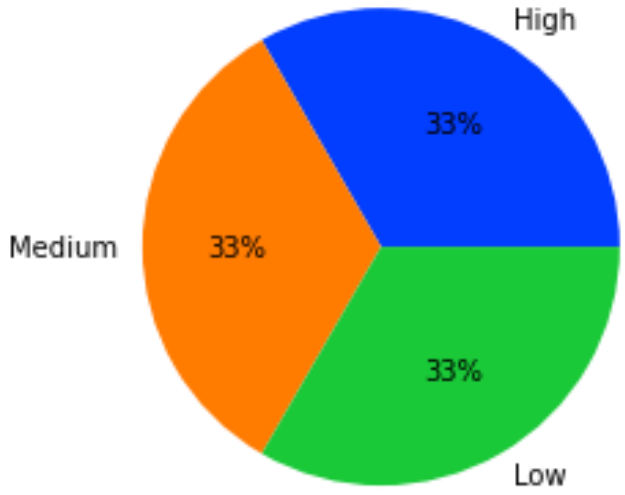
Balancing Data

Adding randomized samples to the data as the data is imbalanced

```
# Adding randomized samples to the data as the data is imbalanced
from imblearn.over_sampling import RandomOverSampler

over_samp =  RandomOverSampler(random_state=0)
X_train_res, y_train_res = over_samp.fit_resample(X, y)
X_train_res.shape, y_train_res.shape
```

```
((1095, 13), (1095,))
```

```
values = y_train_res.value_counts().tolist()
colors = sns.color_palette('bright')
plt.figure()
plt.pie(values, labels=names,colors = colors, autopct = '%0.0f%%')
plt.show()
```



```
X_train, X_test, y_train, y_test = train_test_split(X_train_res, y_train_res, test_size=0.25,
random_state=101)
print (X_train.shape)
print (y_train.shape)
print (X_test.shape)
print (y_test.shape)
```

```
(821, 13)
(821,)
(274, 13)
(274,)
```

Scaling Preprocessing

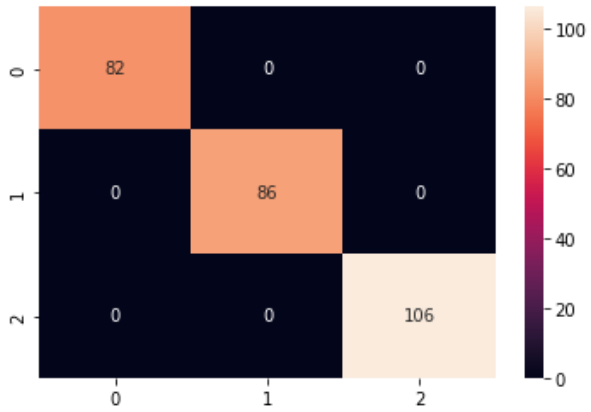
the data will be scaled by the standard scaler function in the sklearn package using the formula $z=Xo-\mu/\sigma$. This can help reduce the effect of outliers when modeling later.

```
from sklearn import preprocessing
scaler=preprocessing.StandardScaler()

X_train_scaled=scaler.fit_transform(X_train) #Scaling and fitting the training set to a
model
X_test_scaled=scaler.transform(X_test) #Transformation of testing set based off of
trained scaler model

# KNeighborsClassifier


classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
#Predict Output
classifier_predicted = classifier.predict(X_test)


classifier_score_test = round(classifier.score(X_test, y_test) * 100, 2)

print('knnclassifier Test Score: \n', classifier_score_test)
print('Accuracy: \n', accuracy_score(y_test, classifier_predicted))
print(confusion_matrix(y_test,classifier_predicted))
print(classification_report(y_test,classifier_predicted))

sns.heatmap(confusion_matrix(y_test,classifier_predicted),annot=True,fmt="d")
```

```
knnclassifier Test Score:
 100.0
Accuracy:
 1.0
[[ 82   0   0]
 [  0  86   0]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
```

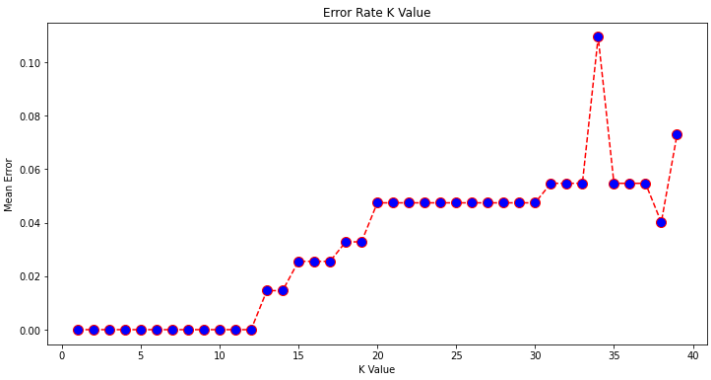
```
         Low       1.00      1.00      1.00        86
      Medium       1.00      1.00      1.00       106

    accuracy                           1.00       274
   macro avg       1.00      1.00      1.00       274
weighted avg       1.00      1.00      1.00       274
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95f2fcd0>
```



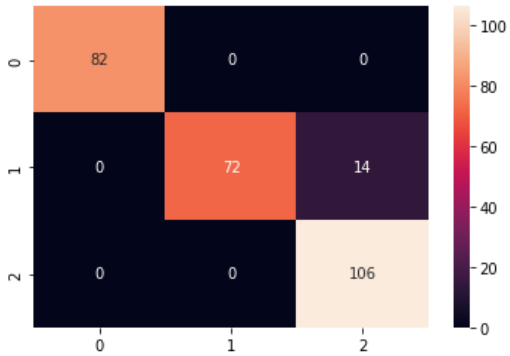

```
#Comparing Error Rate with the K Value
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

```
Text(0, 0.5, 'Mean Error')
```



```
#SVM
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)


#Predict Output
svclassifier_predicted = svclassifier.predict(X_test)


svclassifier_score_test = round(svclassifier.score(X_test, y_test) * 100, 2)

print('svclassifier Test Score: \n', svclassifier_score_test)
print('Accuracy: \n', accuracy_score(y_test, svclassifier_predicted))
print(confusion_matrix(y_test,svclassifier_predicted))
print(classification_report(y_test,svclassifier_predicted))

sns.heatmap(confusion_matrix(y_test,svclassifier_predicted),annot=True,fmt="d")
```

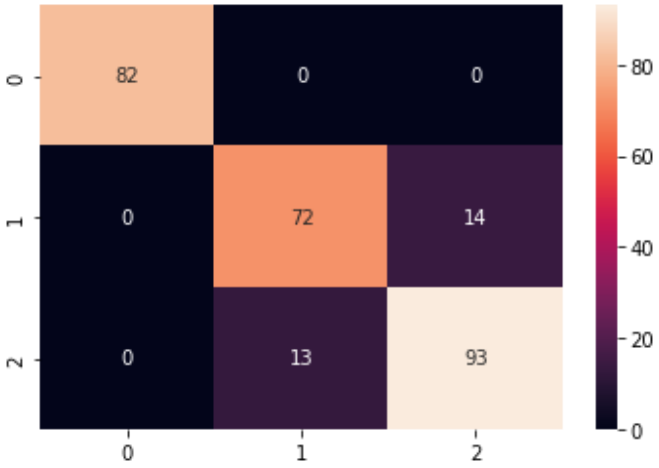
```
svclassifier Test Score:
 94.89
Accuracy:
 0.948905109489051
[[ 82   0   0]
 [  0  72  14]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      0.84      0.91        86
      Medium       0.88      1.00      0.94       106

    accuracy                           0.95       274
   macro avg       0.96      0.95      0.95       274
weighted avg       0.95      0.95      0.95       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95d7a650>
```




```
#Logistic regresion
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
#Predict Output
logreg_predicted = logreg.predict(X_test)


logreg_score_test = round(logreg.score(X_test, y_test) * 100, 2)

print('logistic regression Test Score: \n', logreg_score_test)
print('Accuracy: \n', accuracy_score(y_test, logreg_predicted))
print(confusion_matrix(y_test,logreg_predicted))
print(classification_report(y_test,logreg_predicted))

sns.heatmap(confusion_matrix(y_test,logreg_predicted),annot=True,fmt="d")
```

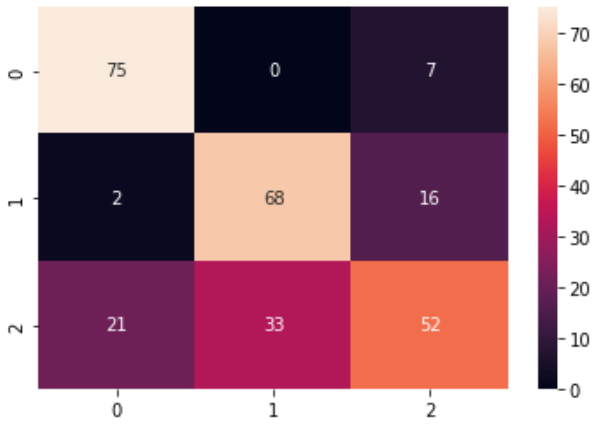
```
logistic regression Test Score:
 90.15
Accuracy:
 0.9014598540145985
[[82  0  0]
 [ 0 72 14]
 [ 0 13 93]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       0.85      0.84      0.84        86
      Medium       0.87      0.88      0.87       106

    accuracy                           0.90       274
   macro avg       0.91      0.90      0.91       274
weighted avg       0.90      0.90      0.90       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95d2efd0>
```




```
# Gaussian Naive Bayes

gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
#Predict Output
```

```
gauss_predicted = gaussian.predict(X_test)


gauss_score_test = round(gaussian.score(X_test, y_test) * 100, 2)

print('Gaussian Test Score: \n', gauss_score_test)
print('Accuracy: \n', accuracy_score(y_test, gauss_predicted))
print(confusion_matrix(y_test,gauss_predicted))
print(classification_report(y_test,gauss_predicted))

sns.heatmap(confusion_matrix(y_test,gauss_predicted),annot=True,fmt="d")
```

```
Gaussian Test Score:
 71.17
Accuracy:
 0.7116788321167883
[[75  0  7]
 [ 2 68 16]
 [21 33 52]]
              precision    recall  f1-score   support

        High       0.77      0.91      0.83        82
         Low       0.67      0.79      0.73        86
      Medium       0.69      0.49      0.57       106

    accuracy                           0.71       274
   macro avg       0.71      0.73      0.71       274
weighted avg       0.71      0.71      0.70       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95c65410>
```

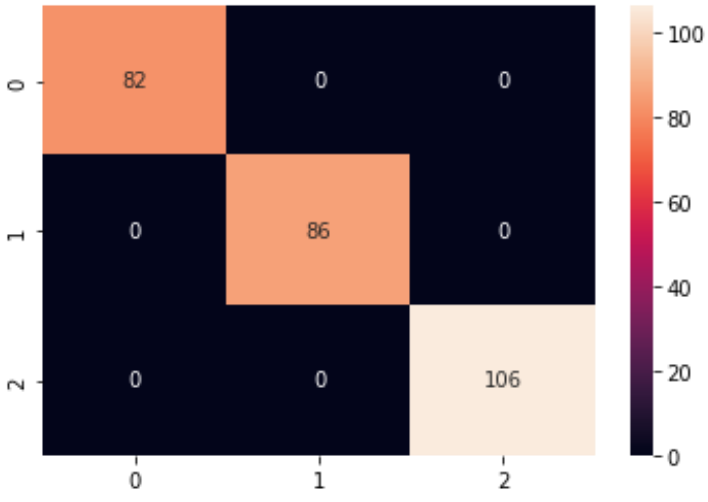



```
#Decision tree
dtree=DecisionTreeClassifier()

dtree.fit(X_train, y_train)
#Predict Output
dtree_predicted = dtree.predict(X_test)
```

```
dtree_score_test = round(dtree.score(X_test, y_test) * 100, 2)

print('decision tree Test Score: \n', dtree_score_test)
print('Accuracy: \n', accuracy_score(y_test, dtree_predicted))
print(confusion_matrix(y_test,dtree_predicted))
print(classification_report(y_test,dtree_predicted))

sns.heatmap(confusion_matrix(y_test,dtree_predicted),annot=True,fmt="d")
```

```
decision tree Test Score:
 100.0
Accuracy:
 1.0
[[ 82   0   0]
 [  0  86   0]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        86
      Medium       1.00      1.00      1.00       106

    accuracy                           1.00       274
   macro avg       1.00      1.00      1.00       274
weighted avg       1.00      1.00      1.00       274
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95ba2d50>
```

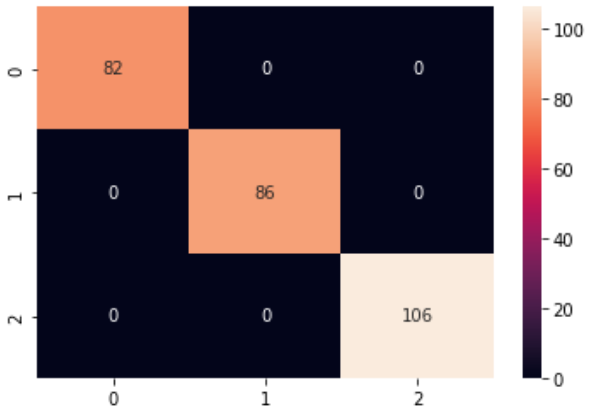



```
#Random Forest

rf = RandomForestClassifier(n_estimators = 100)

rf.fit(X_train, y_train)
#Predict Output
rf_predicted = rf.predict(X_test)
```

```
rf_score_test = round(rf.score(X_test, y_test) * 100, 2)

print('random forest Test Score: \n', rf_score_test)
print('Accuracy: \n', accuracy_score(y_test, rf_predicted))
print(confusion_matrix(y_test,rf_predicted))
print(classification_report(y_test,rf_predicted))

sns.heatmap(confusion_matrix(y_test,rf_predicted),annot=True,fmt="d")
```

```
random forest Test Score:
 100.0
Accuracy:
 1.0
[[ 82   0   0]
 [  0  86   0]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        86
      Medium       1.00      1.00      1.00       106

    accuracy                           1.00       274
   macro avg       1.00      1.00      1.00       274
weighted avg       1.00      1.00      1.00       274
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95cfbad0>
```



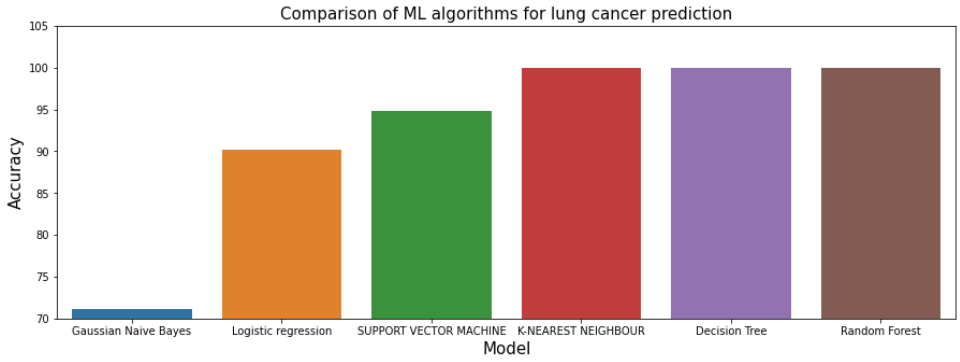

```
###Model evaluation
#We can now rank our evaluation of all the models to choose the best one for our problem.
models = pd.DataFrame({
    'Model': [ 'K-NEAREST NEIGHBOUR', 'Gaussian Naive Bayes','SUPPORT VECTOR MACHINE','Logistic regression','Decision Tree','Random Forest'],

    'Test Score': [ classifier_score_test, gauss_score_test,
```

```
svclassifier_score_test,logreg_score_test,dtree_score_test,rf_score_test]})
models.sort_values(by='Test Score', ascending=False)
```

	Model	Test Score
0	K-NEAREST NEIGHBOUR	100.00
4	Decision Tree	100.00
5	Random Forest	100.00
2	SUPPORT VECTOR MACHINE	94.89
3	Logistic regression	90.15
1	Gaussian Naive Bayes	71.17

```
plt.figure(figsize=(15, 5))

t=sns.barplot(x='Model',y='Test Score', data=models,order=models.sort_values('Test Score').Model)
t.set_ylim(70,105)
plt.title('Comparison of ML algorithms for lung cancer prediction',SIZE=15)
plt.xlabel('Model',size=15)
plt.ylabel('Accuracy', size=15)
```

```
Text(0, 0.5, 'Accuracy')
```



Hyper parameter tuning to improve accuracy

1.Hyper parameter tuning for Gaussian Naive Bayes

2.Hyper parameter tuning for Logistic Regression

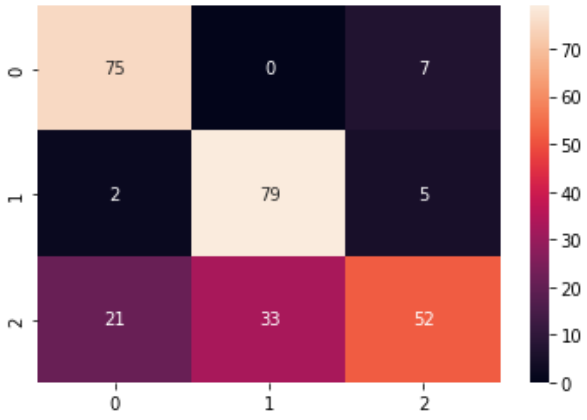
3.Hyper parameter tuning for Support Vector Machine

```
#Hyper parameter tuning for Gaussian Naive Bayes
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import PowerTransformer
param_grid_nb = {'var_smoothing': np.logspace(0,-9, num=100)}
Model_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb,
verbose=1, cv=10, n_jobs=-1)
Data_transformed = PowerTransformer().fit_transform(X_test)

Model_grid.fit(Data_transformed, y_test);

print(Model_grid.best_estimator_)
print(Model_grid.best_params_)
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
GaussianNB(var_smoothing=0.0657933224657568)
{'var_smoothing': 0.0657933224657568}
```

```
#Hyper parameter tuning for Gaussian Naive Bayes
# predict the target on the test dataset
hpnb_predicted = Model_grid.predict(Data_transformed)

# Accuracy Score on test dataset
HPGNB_SCORE= (accuracy_score(y_test,hpnb_predicted)*100)
print('accuracy_score on test dataset : ', HPGNB_SCORE)
print(confusion_matrix(y_test,hpnb_predicted))
print(classification_report(y_test,hpnb_predicted))

sns.heatmap(confusion_matrix(y_test,hpnb_predicted),annot=True,fmt="d")
```

```
accuracy_score on test dataset :  75.18248175182481
[[75  0  7]
 [ 2 79  5]
 [21 33 52]]
              precision    recall  f1-score   support

        High       0.77      0.91      0.83        82
         Low       0.71      0.92      0.80        86
      Medium       0.81      0.49      0.61       106

    accuracy                           0.75       274
   macro avg       0.76      0.77      0.75       274
weighted avg       0.76      0.75      0.74       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d959ee410>
```



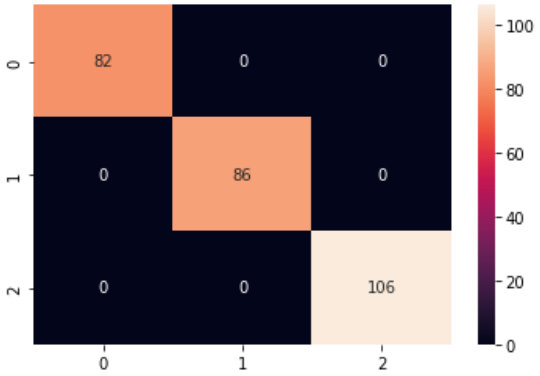

```
#Hyper parameter tuning for Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Creating the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
```

```
param_grid = {'C': c_space}

# Instantiating logistic regression classifier
logreg = LogisticRegression()

# Instantiating the GridSearchCV object
logreg_cv = GridSearchCV(logreg, param_grid, cv = 5)

logreg_cv.fit(X, y)

# Print the tuned parameters and score
print("Tuned Logistic Regression Parameters: {}".format(logreg_cv.best_params_))
print("Best score is {}".format(logreg_cv.best_score_))
print(logreg_cv.best_params_)
```

```
Tuned Logistic Regression Parameters: {'C': 2275.845926074791}
Best score is 1.0
{'C': 2275.845926074791}
```

```
#Hyper parameter tuning for Logistic Regression
hplg_predicted = logreg_cv.predict(X_test)

# Accuracy Score on test dataset
HPLG_SCORE = (accuracy_score(y_test,hplg_predicted)*100)
print('accuracy_score on test dataset : ', HPLG_SCORE)
print(confusion_matrix(y_test,hplg_predicted))
print(classification_report(y_test,hplg_predicted))

sns.heatmap(confusion_matrix(y_test,hplg_predicted),annot=True,fmt="d")
```

```
accuracy_score on test dataset :  100.0
[[ 82   0   0]
 [  0  86   0]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        86
      Medium       1.00      1.00      1.00       106

    accuracy                           1.00       274
   macro avg       1.00      1.00      1.00       274
weighted avg       1.00      1.00      1.00       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95a0be90>
```




```
#Hyper parameter tuning for Support Vector Machine
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, y_train)
# print best parameter after tuning
print(grid.best_params_)

# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.964 total time=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.988 total time=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.957 total time=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.957 total time=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.994 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.970 total time=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.982 total time=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.957 total time=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.970 total time=   0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.988 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.848 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.921 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.848 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.890 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.902 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.715 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.726 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.750 total time=   0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.732 total time=   0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.738 total time=   0.0s
[CV 1/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.467 total time=   0.0s
[CV 2/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.445 total time=   0.0s
[CV 3/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.463 total time=   0.0s
```

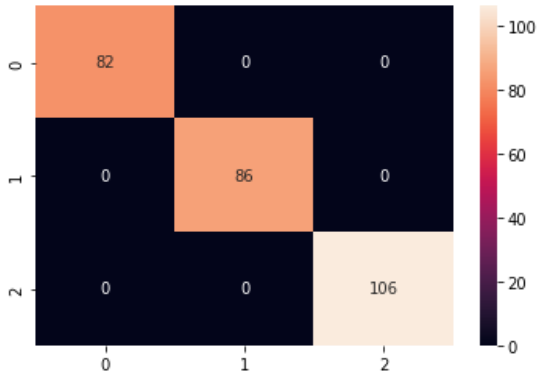
[CV 4/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.470 total time= 0.0s
[CV 5/5] END ...C=0.1, gamma=0.0001, kernel=rbf;, score=0.445 total time= 0.0s
[CV 1/5] END .....C=1, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=1, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=1, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=1, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=1, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=1, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=1, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=1, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.01, kernel=rbf;, score=0.982 total time= 0.0s
[CV 2/5] END .....C=1, gamma=0.01, kernel=rbf;, score=0.982 total time= 0.0s
[CV 3/5] END .....C=1, gamma=0.01, kernel=rbf;, score=0.957 total time= 0.0s
[CV 4/5] END .....C=1, gamma=0.01, kernel=rbf;, score=0.951 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.01, kernel=rbf;, score=0.982 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.879 total time= 0.0s
[CV 2/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.921 total time= 0.0s
[CV 3/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.848 total time= 0.0s
[CV 4/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.878 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.001, kernel=rbf;, score=0.921 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.727 total time= 0.0s
[CV 2/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.726 total time= 0.0s
[CV 3/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.750 total time= 0.0s
[CV 4/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.732 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.0001, kernel=rbf;, score=0.738 total time= 0.0s
[CV 1/5] END .....C=10, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=10, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=10, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=10, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=10, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=10, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=10, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=10, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=10, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=10, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=10, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=10, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=10, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=10, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=10, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.897 total time= 0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.957 total time= 0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.909 total time= 0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.933 total time= 0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.970 total time= 0.0s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.879 total time= 0.0s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.921 total time= 0.0s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.829 total time= 0.0s

[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.878 total time= 0.0s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;, score=0.921 total time= 0.0s
[CV 1/5] END .....C=100, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=100, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=100, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=100, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=100, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=100, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=100, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=100, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=100, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=100, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END ....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END ....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END ....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END ....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ...C=100, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END ...C=100, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END ...C=100, gamma=0.001, kernel=rbf;, score=0.970 total time= 0.0s
[CV 4/5] END ...C=100, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END ...C=100, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.879 total time= 0.0s
[CV 2/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.939 total time= 0.0s
[CV 3/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.854 total time= 0.0s
[CV 4/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.890 total time= 0.0s
[CV 5/5] END ...C=100, gamma=0.0001, kernel=rbf;, score=0.939 total time= 0.0s
[CV 1/5] END ......C=1000, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END ......C=1000, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END ......C=1000, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END ......C=1000, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END ......C=1000, gamma=1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END .....C=1000, gamma=0.1, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END ....C=1000, gamma=0.01, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 2/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 3/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 4/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 5/5] END ...C=1000, gamma=0.001, kernel=rbf;, score=1.000 total time= 0.0s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.921 total time= 0.0s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.970 total time= 0.0s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.884 total time= 0.0s

```
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.963 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.957 total time=   0.0s
{'C': 1, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=1, gamma=1)
```

```
#Hyper parameter tuning for Support Vector Machine

hpsvm_predicted = grid.predict(X_test)

# Accuracy Score on test dataset
HPSVM_SCORE = (accuracy_score(y_test,hpsvm_predicted)*100)
print('accuracy_score on test dataset : ', HPSVM_SCORE)
print(confusion_matrix(y_test,hpsvm_predicted))
print(classification_report(y_test,hpsvm_predicted))

sns.heatmap(confusion_matrix(y_test,hpsvm_predicted),annot=True,fmt="d")
```

```
accuracy_score on test dataset :  100.0
[[ 82   0   0]
 [  0  86   0]
 [  0   0 106]]
              precision    recall  f1-score   support

        High       1.00      1.00      1.00        82
         Low       1.00      1.00      1.00        86
      Medium       1.00      1.00      1.00       106

    accuracy                           1.00       274
   macro avg       1.00      1.00      1.00       274
weighted avg       1.00      1.00      1.00       274


<matplotlib.axes._subplots.AxesSubplot at 0x7f8d95053350>
```



MAX VOTING ENSEMBLE TECHNIQUE FOR SVM, LOGISTIC AND NAIVE BAYES FOR BETTER ACCURACY

```
from sklearn.ensemble import VotingClassifier
model1 = LogisticRegression(random_state=1)
model2 = SVC(gamma ='auto', probability = True)
model3= GaussianNB()
```

```
ENSEMBLE1 = VotingClassifier(estimators=[('lr', model1), ('svc', model2),('gnb',model3)],
voting='hard')
ENSEMBLE1.fit(X_train,y_train)
ENSEMBLE1.score(X_test,y_test)
```

```
0.9124087591240876
```

```
# predict the target on the test dataset
predict_test = ENSEMBLE1.predict(X_test)

# Accuracy Score on test dataset
ENSEMBLE_hard_test = (accuracy_score(y_test,predict_test)*100)
print('accuracy_score on test dataset : ', ENSEMBLE_hard_test)
```

```
accuracy_score on test dataset :  91.24087591240875
```

```
from sklearn.ensemble import VotingClassifier
model1 = LogisticRegression(random_state=1)
model2 = SVC(gamma ='auto', probability = True)
model3= GaussianNB()
ENSEMBLE2 = VotingClassifier(estimators=[('lr', model1), ('svc', model2),('gnb',model3)],
voting='soft')
ENSEMBLE2.fit(X_train,y_train)
ENSEMBLE2.score(X_test,y_test)
```

```
1.0
```

```
# predict the target on the test dataset
predict_test = ENSEMBLE2.predict(X_test)

# Accuracy Score on test dataset
ENSEMBLE_soft_test = (accuracy_score(y_test,predict_test)*100)
print('accuracy_score on test dataset : ', ENSEMBLE_soft_test)
```

```
accuracy_score on test dataset :  100.0
```

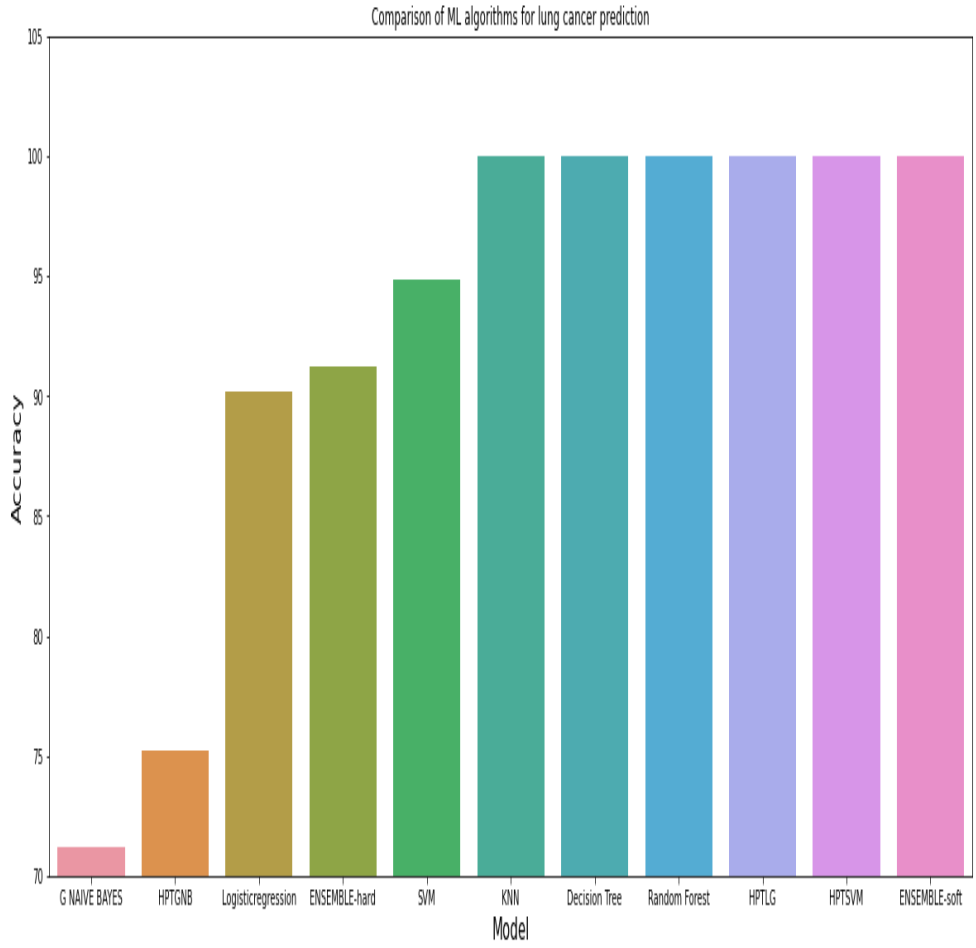
```
###Model evaluation
#We can now rank our evaluation of all the models to choose the best one for our
problem.
models = pd.DataFrame({
   'Model': [ 'KNN', 'G NAIVE BAYES','SVM','Logisticregression','Decision
Tree','Random Forest','HPTGNB','HPTLG','HPTSVM','ENSEMBLE-
soft','ENSEMBLE-hard'],

   'Test Score': [ classifier_score_test, gauss_score_test,
svclassifier_score_test,logreg_score_test,dtree_score_test,rf_score_test,HPGNB_SCORE,HP
LG_SCORE,HPSVM_SCORE,ENSEMBLE_soft_test,ENSEMBLE_hard_test]})
models.sort_values(by='Test Score', ascending=False)
```

```
              Model  Test Score
0               KNN  100.000000
4     Decision Tree  100.000000
5     Random Forest  100.000000
```

```
7              HPTLG  100.000000
8            HPTSVM  100.000000
9       ENSEMBLE-soft  100.000000
2               SVM   94.890000
10      ENSEMBLE-hard   91.240876
3   Logisticregression   90.150000
6              HPTGNB   75.182482
1       G NAIVE BAYES   71.170000
```

```
plt.figure(figsize=(20, 8))

t=sns.barplot(x='Model',y='Test Score', data=models,order=models.sort_values('Test Score').Model)
t.set_ylim(70,105)
plt.title('Comparison of ML algorithms for lung cancer prediction')
plt.xlabel('Model',size=15)
plt.ylabel('Accuracy', size=15)
```

```
Text(0, 0.5, 'Accuracy')
```



CHAPTER 6

RESULTS AND CONCLUSION

6.1 Results

Accuracy of algorithms

	Model	Test Score
0	K-NEAREST NEIGHBOUR	100.00
4	Decision Tree	100.00
5	Random Forest	100.00
2	SUPPORT VECTOR MACHINE	94.89
3	Logistic regression	90.15
1	Gaussian Naive Bayes	71.17

Fig 9 accuracy of algorithms

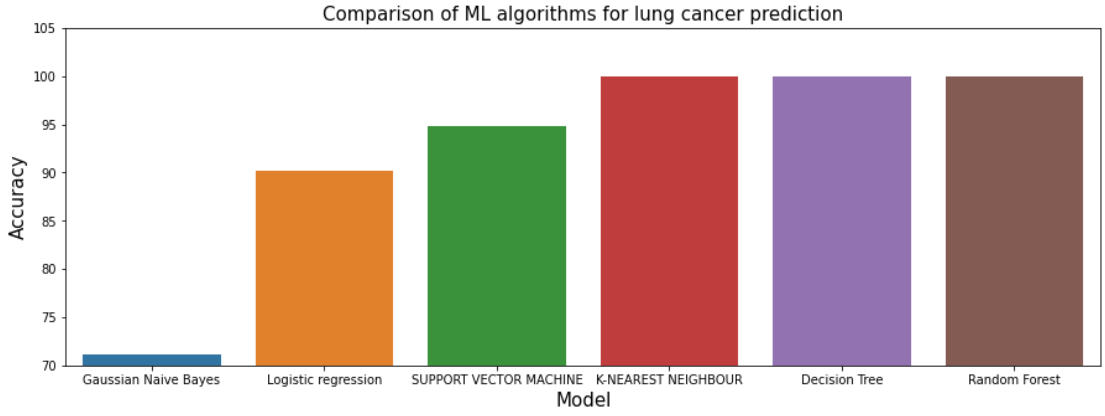


Fig 10 accuracy comparison

The least predicted accuracy models were increased with advanced techniques then the accuracies are:

Final comparison of all algorithms for lung cancer prediction

[ ]

	Model	Test Score
0	KNN	100.000000
4	Decision Tree	100.000000
5	Random Forest	100.000000
7	HPTLG	100.000000
8	HPTSVM	100.000000
9	ENSEMBLE-soft	100.000000
2	SVM	94.890000
10	ENSEMBLE-hard	91.240876
3	Logisticregression	90.150000
6	HPTGNB	75.182482
1	G NAIVE BAYES	71.170000

Fig 11 Final accuracies of all models for lung cancer

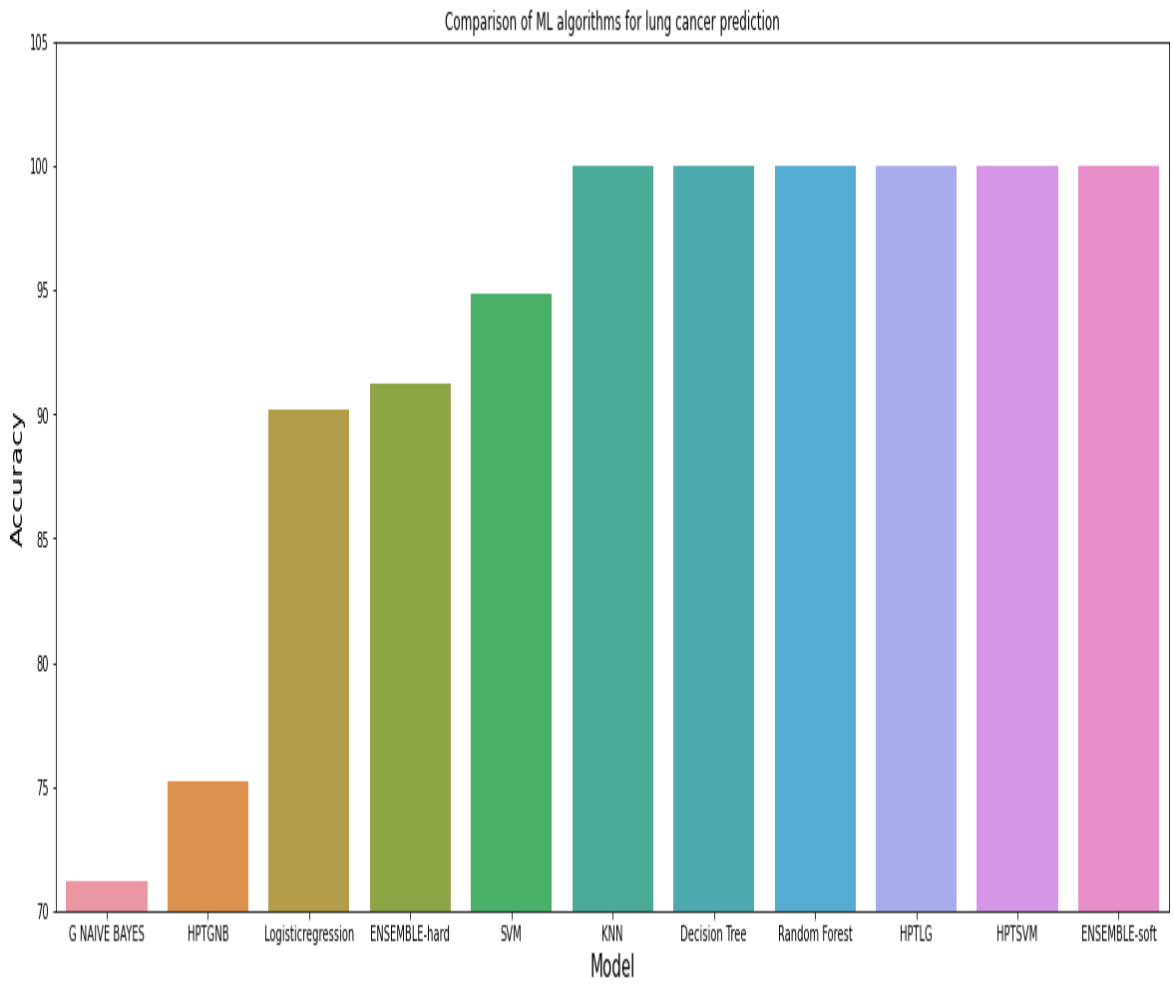


Fig 12 Comparison of algorithms

6.2 CONCLUSION AND FUTURE WORK

Lung cancer prediction using various machine learning algorithms has been researched with 100% accuracy with 6 models and mainly the objective is to compare and review the algorithms for further study. The model predicted with atmost accuracy and few algorithms were least reacted so they were upgraded with hyper parameter and ensemble technique to achieve the accuracy. The future work is to implement a software using image dataset and high advanced computing algorithms to predict with atmost accuracy and also a reliable software which is used for the society and medical team. A few drawbacks from the system is data set which is perfect , so the prediction was achieved but considering a world wide problem we need to focus on real image ct-scan data set to implement and to prove the results . so, Focusing on above mentioned steps for the future work .

CHAPTER 7

REFERENCES

7.1 References:

1. Wang, Z., Sun, J., Sun, Y. *et al.* Machine Learning Algorithm Guiding Local Treatment Decisions to Reduce Pain for Lung Cancer Patients with Bone Metastases, a Prospective Cohort Study. *Pain Ther* 10, 619–633 (2021). https://doi.org/10.1007/s40122-021-00251-2
2. Manju, B. R., Athira, V., & Rajendran, A. (2021). Efficient multi-level lung cancer prediction model using support vector machine classifier. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1012, No. 1, p. 012034). IOP Publishing.
3. Heuvelmans, M. A., van Ooijen, P. M., Ather, S., Silva, C. F., Han, D., Heussel, C. P., ... & Oudkerk, M. (2021). Lung cancer prediction by Deep Learning to identify benign lung nodules. *Lung cancer*, *154*, 1-4.
4. Benzekry, S., Grangeon, M., Karlsen, M., Alexa, M., Bicalho-Frazeto, I., Chaleat, S., ... & Greillier, L. (2021). Machine learning for prediction of immunotherapy efficacy in non-small cell lung cancer from simple clinical and biological data. *Cancers*, *13*(24), 6210.
5. Sujitha, R., Seenivasagam, V. Classification of lung cancer stages with machine learning over big data healthcare framework. *J Ambient Intell Human Comput* 12, 5639–5649 (2021). https://doi.org/10.1007/s12652-020-02071-2

6. Tsou, P. H., Lin, Z. L., Pan, Y. C., Yang, H. C., Chang, C. J., Liang, S. K., ... & Li, Y. K. (2021). Exploring volatile organic compounds in breath for high-accuracy prediction of lung cancer. *Cancers*, *13*(6), 1431.
7. Nanglia, P., Kumar, S., Mahajan, A. N., Singh, P., & Rathee, D. (2021). A hybrid algorithm for lung cancer classification using SVM and Neural Networks. *ICT Express*, *7*(3), 335-341.
8. Shanthi, S., Rajkumar, N. Lung Cancer Prediction Using Stochastic Diffusion Search (SDS) Based Feature Selection and Machine Learning Methods. *Neural Process Lett* 53, 2617–2630 (2021). https://doi.org/10.1007/s11063-020-10192-0
9. Cherukuri, N., Bethapudi, N. R., Thotakura, V. S. K., Chitturi, P., Basha, C. Z., & Mummidi, R. M. (2021, March). Deep Learning for Lung Cancer Prediction using NSCLS patients CT Information. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 325-330). IEEE.
10. Ismail, M. B. S. (2021). Lung Cancer Detection and Classification using Machine Learning Algorithm. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(13), 7048-7054.
11. https://en.wikipedia.org/wiki/Support-vector_machine
12. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
13. https://en.wikipedia.org/wiki/Logistic_regression
14. https://news.cancerconnect.com/lung-cancer/does-air-pollution-cause-lung-cancer#:~:text=Exposure%20to%20air%20pollution%20was,and%204%25%20for%20other%20illnesses .
15. https://www.webmd.com/lung/hypersensitivity-

pneumonitis-overview

16. https://academic.oup.com/jnci/article/94/24/1877/2906810
17. https://academic.oup.com/aje/article/152/1/32/139163
18. https://www.verywellhealth.com/is-lung-cancer-inherited-2248975#:~:text=Although%20smoking%20remains%20the%20predominant,linked%20to%20a%20genetic%20predisposition.
19. https://erj.ersjournals.com/content/39/5/1230
20. https://www.healthline.com/health/lung-cancer-coughing-up-blood#:~:text=Coughing%20up%20blood%20is%20not,already%20reached%20an%20advanced%20stage.
21. https://pubmed.ncbi.nlm.nih.gov/28210163/
22. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2409471/#:~:text=In%20patients%20with%20lung%20cancer,energy%20intake%20and%20energy%20expenditure.
23. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1282241/#:~:text=LeRoux3%20reported%20that%201,According%20to%20Stankey%20et%20al.
24. https://mrmjournal.biomedcentral.com/articles/10.1186/s40248-016-0066-z
25. https://www.healthline.com/health/lung-cancer/early-signs#:~:text=When%20lung%20cancer%20causes%20chest,other%20areas%20of%20your%20body.
26. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
27. https://en.wikipedia.org/wiki/Decision_tree#:~:text=A%20decision%20tree%20is%20a%20flowchart%2Dlike%20structure%20in%20which,taken%20after%20computing%20all%20attributes).
28. https://en.wikipedia.org/wiki/Random_forest
29. https://en.wikipedia.org/wiki/Hyperparameter_optimization
30. https://www.toptal.com/machine-learning/ensemble-methods-machine-learning

T H E END