

FUNCTION OF COPD USING MACHINE LEARNING TECHNIQUE

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

M.NIRMAL KUMAR (38110373)

P.THARUN KUMAR (38110599)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600

119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **M.NIRMAL KUMAR (38110373), P.THARUN KUMAR (38110599)** who carried out the project entitled **"PREDICTION OF COPD USING MACHINE LEARNING TECHNIQUE"** under my supervision from November 2021 to March 2022

Internal Guide

Dr. G. Kalaiarasi M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph.D.,

Dr. S. Vigneshwari M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **M.Nirmal kumar** hereby declare that the Project Report entitled **PREDICTION OF COPD USING MACHINE LEARNING TECHNIQUE** done by us under the guidance of **Dr. G. Kalaiaarasi ME. Ph.D.** (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

A handwritten signature in black ink, reading "M. Nirmal" followed by a stylized flourish.

DATE: 05-04-2022

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D. ,** and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G. Kalaiaarasi M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Ultrasound insonation of lungs that are dense with extravascular lung water (EVLW) produces characteristic reverberation artifacts termed B-lines. The number of B-lines present demonstrates reasonable correlation to the amount of EVLW. However, analysis of B-line artifacts generated by this modality is semi-quantitative relying on visual interpretation, and as a result, can be subject to inter-observer variability. The purpose of this study was to translate the use of a novel, quantitative lung ultrasound surface wave elastography technique (LUSWE) into the bedside assessment of pulmonary edema in patients admitted with acute congestive heart failure. To prevent this problem in One of the most interesting (or perhaps most profitable) time series data using machine learning techniques. Hence, pulmonary disease prediction has become an important research area. The aim is to predict machine learning based techniques for pulmonary disease prediction results in best accuracy. The analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. To propose a machine learning-based method to accurately predict the pulmonary diseaseIndex value by prediction results in the form of pulmonary disease classification best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given dataset with evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

TABLE OF CONTENT

CHAPTE R	TITLE	PAGE.NO
01	INTRODUCTION	15
	1.1 Data science	16
	1.1.1 Data Scientist...	17
	1.1.2 Required skills for a data scientist	17
	1.2 Artificial Intelligence	20
	1.3 Natural language processing	20
	1.4 Machine Learning	23
	1.5 Existing system	24
	1.5.1 Drawbacks	24
	1.6 Proposed System	25
	1.6.1 Advantages	
02	LITERATURE SURVEY	26
03	AIM AND SCOPE OF PRESENT INVESTIGATION	31
	3.1 Aim	31
	3.2 Project goals	31
	3.3 Scope of the project	32
	3.4 Feasibility study	32

04	EXPERIMENTAL OR MATERIAL AND METHODS; ALGORITHMS USED	35
	4.1 List of Algorithm	35
	4.2 Project Requirements	35
	4.2.1 Functional Requirements	36
	4.2.2 Non-Functional Requirements	37
	4.3 Environmental Requirements	41
	4.4 Software Description	47
	4.4.1 Anaconda Navigator	59
	4.4.2 Jupyter Notebook	60
	4.5 Python	61
	4.6 System Architecture	62
	4.7 Workflow Diagram	63
	4.8 Use Diagram	64
	4.9 Class Diagram	65
	4.10 Activity Diagram	66
	4.11 Sequence Diagram	66
	4.12 Entity Relationship Diagram	68
	4.13 Module Description	70
	4.13.1 Data preprocessing	72
	4.13.2 Data validation/Cleaning /Preparing process	78
	4.13.3 Comparing Algorithms with prediction in the form of best accuracy result	82
	4.13.4 Deployment	84
05	RESULT AND DISCUSSION, PERFORMANCE ANALYSIS	86
	5.1 Module Results	86
	5.1.1 Logistic Regression	87
	5.1.2 Random Forest Classifier	88
	5.1.3 Decision Tree Classifier	89
	5.1.4 Naïve Bayes Algorithm	90
	5.2 Prediction Result by Accuracy	90
06	SUMMARY AND CONCLUSION	93
	6.1 Coding	93
	6.2 Conclusion	126
	6.3 Future Work	126

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	59
02	WORKFLOW DIAGRAM	60
03	USECASE DIAGRAM	61
04	CLASS DIAGRAM	62
05	ACTIVITY DIAGRAM	63
06	SEQUENCE DIAGRAM	64
07	ER – DIAGRAM	65
08	MODULE DIAGRAM	66

LIST OF SYMBOLS

S.N O	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together.
2.	Association	NAME	Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	<i>Relation</i> (uses)	<i>uses</i>	Used for additional process communication.
6.	Relation (extends)	EXTENDS	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication		Communication between various use cases.

8.	State		State of the process.
9.	Initial State		Initial state of the object
10.	Final state		Final state of the object
11.	Control flow		Represents various control flow between the states.
12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interact ion between the system and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components.
16.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or acion.

17.	External entity		Represents external entities such as keyboard,sensors,etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message	Represents the message exchanged.

CHAPTER 1 : INTRODUCTION

Ultrasound insonation of lungs that are dense with extravascular lung water (EVLW) produces characteristic reverberation artifacts termed B-lines. The number of B-lines present demonstrates reasonable correlation to the amount of EVLW. However, analysis of B-line artifacts generated by this modality is semi-quantitative relying on visual interpretation, and as a result, can be subject to inter-observer variability. The purpose of this study was to translate the use of a novel, quantitative lung ultrasound surface wave elastography technique (LUSWE) into the bedside assessment of pulmonary edema in patients admitted with acute congestive heart failure. To prevent this problem in One of the most interesting (or perhaps most profitable) time series data using machine learning techniques. Hence, pulmonary disease prediction has become an important research area. The aim is to predict machine learning

based techniques for pulmonary disease prediction results in best accuracy. The analysis of dataset by supervised machine learning technique(SMLT) to capture several information's like, variable identification, uni-variate analysis, bi-variate and multi-variate analysis, missing value treatments and analyze the data validation, data cleaning/preparing and data visualization will be done on the entire given dataset. To propose a machine learning-based method to accurately predict the pulmonary diseaseIndex value by prediction results in the form of pulmonary disease classification best accuracy from comparing supervise classification machine learning algorithms. Additionally, to compare and discuss the performance of various machine learning algorithms from the given dataset with evaluation classification report, identify the confusion matrix and to categorizing data from priority and the result shows that the effectiveness of the proposed machine learning algorithm technique can be compared with best accuracy with precision, Recall and F1 Score.

1.1 DATA SCIENCE:

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

1.1.1 Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

1.1.2 Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

1.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by

disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a

foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

1.3 Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy at the page or

paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

1.4 MACHINE LEARNING

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data

points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables (X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output **is $y = f(X)$** . The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include **logistic regression, multi-class classification, Decision Trees and support vector machines etc.** Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into **Classification** problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent

attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

1.5 EXISTING SYSTEM

They are used the surface elastic properties of the lung in patients presenting with symptomatic pulmonary edema. Surface wave speeds of the lung were quantified at bedside using a novel LUSWE technique in combination with standard 2DLUS. We observed significant differences in lung surface wave speed at three frequencies between baseline and follow-up testing. We speculate that these differences are driven by a reduction in lung stiffness due to a decrease in EVLW from diuretic therapy. The calculated reduction in surface wave speed magnitude correlated with other markers of decreased EVLW including a decline in the burden of B-lines and successful fluid removal by diuretics. B-Mode ultrasound insonation of lungs that are dense with extravascular lung water (EVLW) produces characteristic reverberation artifacts termed B-lines. The number of B-lines present demonstrates reasonable correlation to the amount of EVLW. However, analysis of B-line artifacts generated by this modality is semi-quantitative relying on visual interpretation, and as a result, can be subject to inter-observer variability. The purpose of this

study was to translate the use of a novel, quantitative lung ultrasound surface wave elastography technique (LUSWE) into the bedside assessment of pulmonary edema in patients admitted with acute congestive heart failure. B-mode lung ultrasound and LUSWE assessment of the lungs were performed using anterior and lateral intercostal spaces in the supine patient. 14 patients were evaluated at admission with reassessment performed 1-2 days after initiation of diuretic therapy. Each exam recorded the total lung B-lines, lung surface wave speeds (at 100, 150, and 200 Hz) and net fluid balance. The patient cohort experienced effective diuresis (average net fluid balance of negative 2.1 liters) with corresponding decrease in pulmonary edema visualized by B-mode ultrasound (average decrease of 13 B-Lines).

1.5.1 Drawbacks

- They are not classifying pulmonary disease on machine learning classification technique and not mention any accuracy results.
- There are not using any artificial intelligence technique

It can't thereby better determine the regularity of pulmonary disease prediction data and achieve more accurate prediction results.

1.6 PROPOSED SYSTEM:

Exploratory Data Analysis of pulmonary disease Prediction

Pulmonary lung datasets from different sources would be combined to form a generalized dataset, and then different machine learning algorithms would be applied to extract patterns and to obtain results with maximum accuracy.

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using machine learning algorithms are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Building the classification model

The predicting the pulmonary disease problem, ML algorithms prediction model is effective because of the following reasons: It provides better results in classification problem.

It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.

It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Test dataset

Training dataset

Architecture of Proposed model

1.6.1 Advantages:

These reports are to the investigation of applicability of machine learning techniques for pulmonary disease prediction in operational conditions.

CHAPTER 2 : LITERATURE SURVEY

General:

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new

interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Review of Literature Survey

Title : Lung Cancer Prediction using Data Mining Techniques

Author: E. Yatish Venkata Chandra, K. Ravi Teja, M.Hari Chandra Siva Prasad, Mohammed.Ismail.B

Year : 2019

The major cause for death in human beings is because of cancer .Lung cancer is one of the most common and serious types of cancer that severely harms the human body. In order to cure the cancer early cancer detection is required. If lung cancer is diagnosed at early stages many lives will be saved. The other name for lung cancer is lung carcinoma, an uncontrolled malignant tumor distinguished by undisciplined cell growth in lung cells. There are many people suffering from this kind of cancer and confining to death. If this is left untreated, this may grow later than lung by metastasis into other parts of body. Many of the cancers starts from lungs, called as primary lung carcinoma. There are two types of small cell lung carcinoma (SCLC), non small cell lung carcinoma(NSCLC). The main reason for lung cancer is smoking of cigarette. There are many researches targeting on exact approaches for treating cancer. To predict the survival rate for NSCLC patients data mining techniques can be used with selection of algorithms. The algorithms used to detect the lung cancer are Support vector machine (SVM), Decision tree, k-Nearest neighbour, Random forest, Logistic regression. In this paper By implementing 2 different datasets and various packages and libraries in python, it is compared and on

implementation found suitable algorithms have more accuracy on certain data sets for optimum prediction rate of lung cancer.

Title : LUNG DISEASE PREDICTION USING IMAGE PROCESSING AND CNN ALGORITHM

Author: Shivani Kasar¹, Darshan Hujband², Abhijeet Ahire³, Jyoti Rahade⁴

Year : 2020

Lung Cancer could be a Disease of uncontrolled cell growth in tissues of the lung. Discovery of carcinoma in its initial stage is that the key of its cure. All in all, a measure for earlier than schedule stage lung disease determination essentially incorporates those using X-beam midsection movies, CT, MRI so forth. In numerous parts of the planet far reaching screening by CT or MRI isn't yet pragmatic, in order that midsection radiology stays in starting and most elementary system. Firstly, we'll utilize some systems are key to the errand of medicinal picture mining, Lung Field Segmentation, processing, Feature Extraction, Classification utilizing neural system and SVMs. The routines utilized as part of this paper work states to group computerized X-beam midsection movies into two classes: ordinary and weird. Diverse learning examinations were performed on two distinctive information sets, made by method for highlight choice and SVMs prepared with diverse parameters; the outcomes are checked out and reported.

Title : Lung Cancer Prediction using Feed Forward Back Propagation Neural Networks with Optimal Features

Author: Dr. S. Senthil, B. Ayshwarya

Year : 2018

The major cause of deaths in human beings is Lung Cancer, Since the lung cancer symptoms appear in the advanced stages so it is hard to detect

which leads to high mortality rate among other cancer types. Hence the early prediction of lung cancer is mandatory for the diagnosis process and it gives the higher chances for successful treatment. It is the most challenging way to enhance a patient's chance for survival. In this paper a computer aided classification method for lung cancer prediction based an evolutionary system by a combination of architectural evolution with weight learning using neural network and Particle Swarm Optimization is implemented. This method proposed different variants and hybridize it with evolutionary algorithm to improve its performance and uses global searching of PSO and local searching ability of neural network gives better lung cancer prediction as cancerous and non-cancerous. The classification is performed and the results were evaluated with the performance comparison of various algorithms. This prediction system is useful for the doctors to take an appropriate decision based on patient's condition.

Title : A Computer Aided Diagnosis of Lung Disease using Machine Learning Approach

Author: Subapriya V, Jaichandran R, Shunmuganathan K.L, Abhiram Rajan, Akshay T, Shibil Rahman

Year : 2020

Cancer is a disease that is unregulated by cells in the body. Lung nodule is called lung cancer because the disease starts in the lungs. Cancer of the pulmonary system begins in the lungs and may travel to lymph nodes or other body species such as the brain. The lungs can also be impacted by cancer from other bodies. The metastases are named as cancer cells migrate from organ to organ. Lung cancers are normally grouped into two major cell and non-small cell types. In this study we predict a Computer Aided Diagnosis (CAD) for lung

cancer prediction using Convolutional Neural Network (CNN) and ML approach.

Title : Detection and Prediction of Lung Cancer Using Different Algorithms

Author: N. Sudhir Reddy, V Khanaa

Year : 2019

One of the major causes of cancer death is through Lung cancer. The overlapping of cancer cells acts as an impediment for its early detection. Identifying genetic and environmental factors plays a vital role in developing better techniques for its prevention. And in order to discover the anomalies in target images, time factor is paramount. In this work, decision tree algorithm is used for prediction of lung cancer wherein the important pattern with their corresponding weightage and score is studied. Processes such as preprocessing of images and feature extraction are done using Histogram Equalization and using neural network classifier does normal or abnormality check of the patient. If lung cancer is detected early, then it will reduce many treatment options, reduce risk of invasive surgery and increase survival rate. Therefore, an easy, cost effective and time saving method will produce promising result for detection and prediction of lung cancer.

CHAPTER 3 : AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 Aim

The goal is to develop a machine learning model for Pulmonary Disease Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

3.2 Project Goals

Exploration data analysis of variable identification

- Loading the given dataset
- Import required libraries packages

- Analyze the general properties
- Find duplicate and missing values
- Checking unique and count values

Uni-variate data analysis

- Rename, add data and drop the data
- To specify data type

Exploration data analysis of bi-variate and multi-variate

- Plot diagram of pairplot, heatmap, bar chart and Histogram

Method of Outlier detection with feature engineering

- Pre-processing the given dataset
- Splitting the test and training dataset
- Comparing the Decision tree and Logistic regression model and random forest etc.

Comparing algorithm to predict the result

- Based on the best accuracy

3.3 Scope of the Project

Here the scope of the project is that integration of clinical decision support with computer-based patient records could reduce medical errors, enhance patient safety, decrease unwanted practice variation, and improve patient outcome. This suggestion is promising as data modeling and analysis tools, e.g., data mining, have the potential to generate a knowledge-rich environment which can help to significantly improve the quality of clinical decisions.

3.4 FEASIBILITY STUDY:

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Preprocessing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

Building the classification model

The prediction of Heart attack, A high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

It is strong in preprocessing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.

It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and

predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.

CHAPTER 4 : EXPERIMENTAL OR MATERIAL AND METHODS; ALGORITHMS USED

4.1 List of Algorithms

Data Pre-processing

Data Analysis of Visualization

Comparing Algorithm with prediction in the form of best accuracy result

Deployment Using Flask

4.2 PROJECT REQUIREMENTS:

General:

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

4.2.1 Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

4.2.2 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

4.3 ENVIRONMENTAL REQUIREMENTS:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

4.4 SOFTWARE DESCRIPTION:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and

Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

4.4.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window.

You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

JupyterLab

Jupyter Notebook

Spyder

PyCharm

VSCode

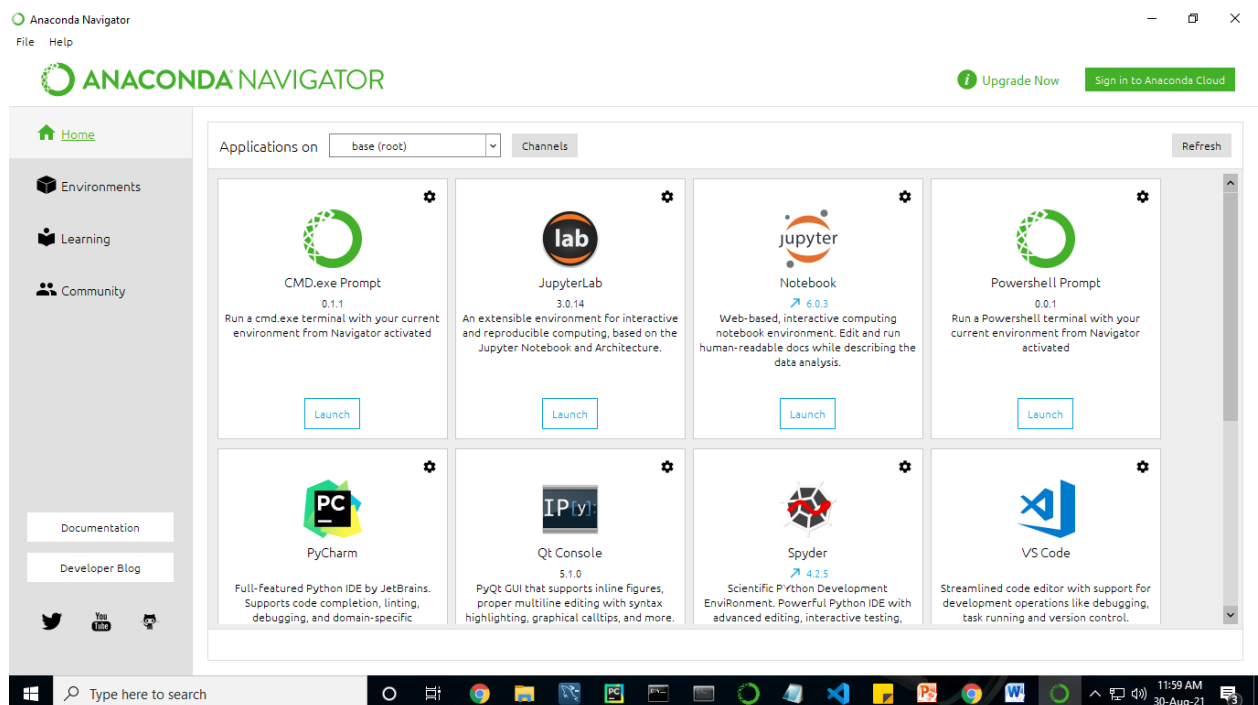
Glueviz

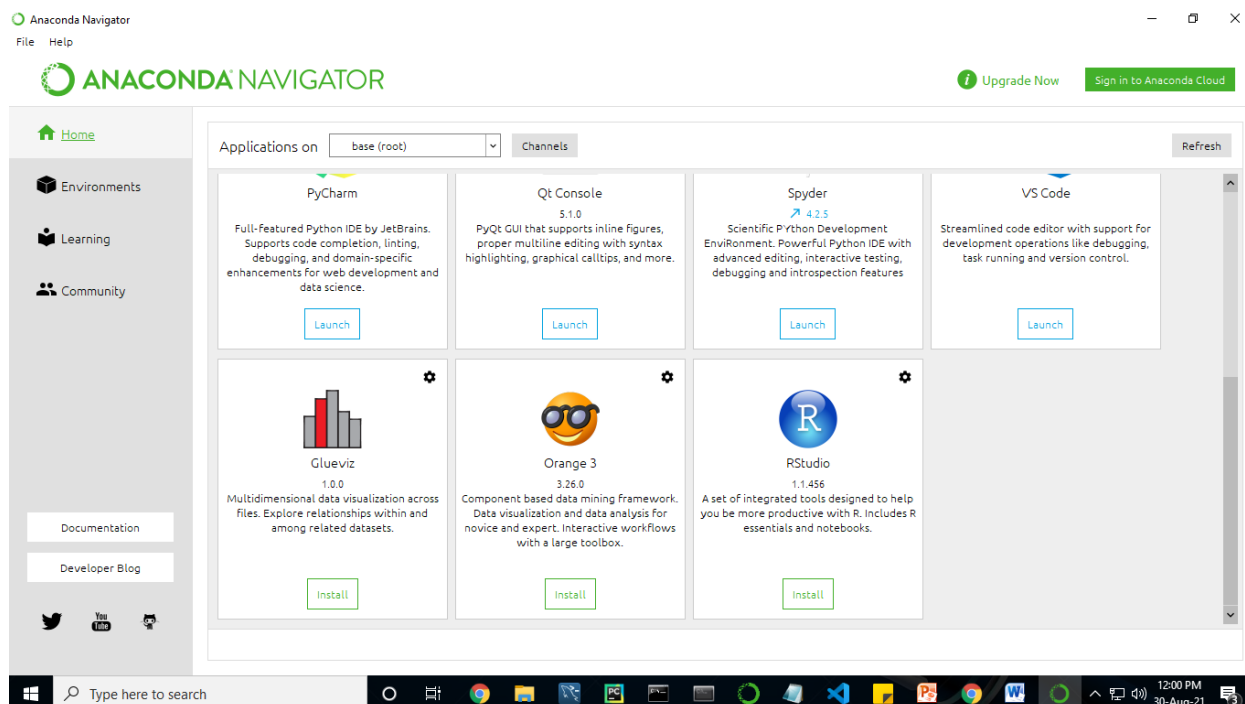
Orange 3 App

RStudio

Anaconda Prompt (Windows only)

Anaconda PowerShell (Windows only)





Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda :

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. these softwares have great graphical user interface and these will make your work easy to do. you can also use it to run your python script. These are the software carried by anaconda navigator.

4.4.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements

(paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

Running the Jupyter Notebook

Launching *Jupyter Notebook App*: The Jupyter Notebook App can be launched by clicking on the *Jupyter Notebook* icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- ❖ Click on the menu *Help -> User Interface Tour* for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing *shift + enter*.
- ❖ You can run the whole notebook in a single step by clicking on the menu *Cell -> Run All*.
- ❖ To restart the kernel (i.e. the computational engine), click on the menu *Kernel -> Restart*. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser.

The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook *kernel* is a “computational engine” that executes the code contained in a Notebook document. The *ipython kernel*, referenced in this guide, executes python code. Kernels for many other languages exist (**official kernels**).

When you open a Notebook document, the associated *kernel* is automatically launched. When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results.

Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down

Notebook Dashboard: The *Notebook Dashboard* is the component which is shown first when you launch Jupyter Notebook App. The *Notebook Dashboard* is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The *Notebook Dashboard* has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

Download and install anaconda and get the most useful package for machine learning in Python.

Load a dataset and understand its structure using statistical summaries and data visualization.

Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

4.5. PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional

programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list

comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one— and preferably only one —obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension

modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*

Syntax and Semantics:

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation:

Main article: [Python syntax and semantics & Indentation](#)

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain

statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow :

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.

- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.^{[80][81]} However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was

passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) `//` and floating-point/division. Python also uses the `**` operator for exponentiation.
- From Python 3.5, the new `@` infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax `:=`, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.
- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`, thereby effectively "modifying the contents" of `t`, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.
- Python has a "string format" operator `%`. This functions analogously to `printf` format strings in C, e.g. `"spam=%s eggs=%d" % ("blah", 2)` evaluates to `"spam=blah eggs=2"`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam={0}"`

`eggs={1}""".format("blah",2). Python 3.6 added "f-strings": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'`

- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. `"spam" + "eggs"` returns `"spameggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"2"`.
- Python has various kinds of string literals:
 - o Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
 - o Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
 - o Raw string varieties, denoted by prefixing the string literal with an `r`. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. Compare "`@-quoting`" in C#.
- Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1: ...` causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

Typing :

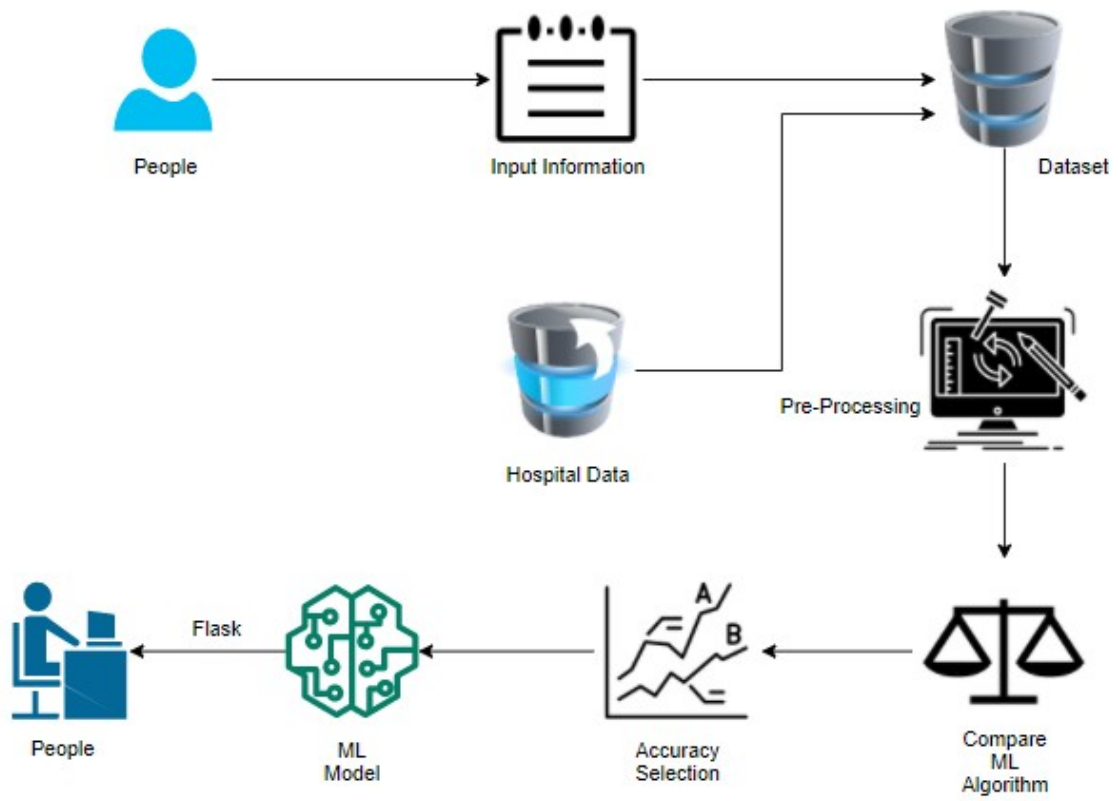
Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

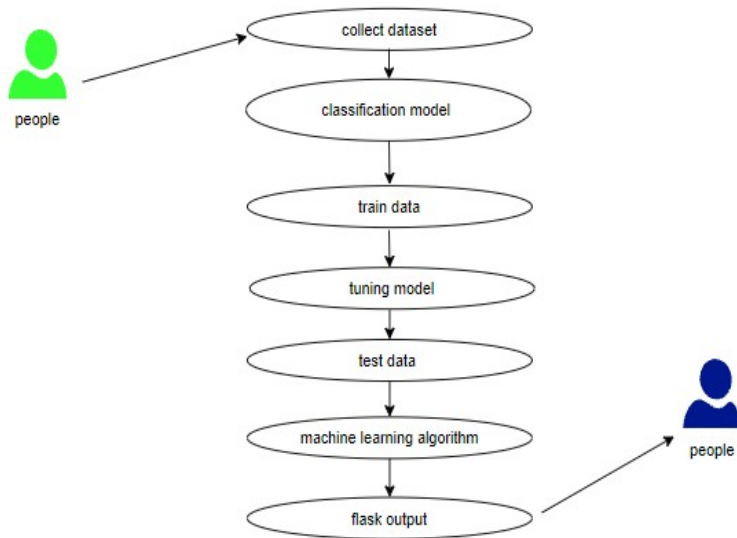
Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from `object` and are instances of `type`). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named `mypy` supports compile-time type checking.

4.6 SYSTEM ARCHITECTURE:

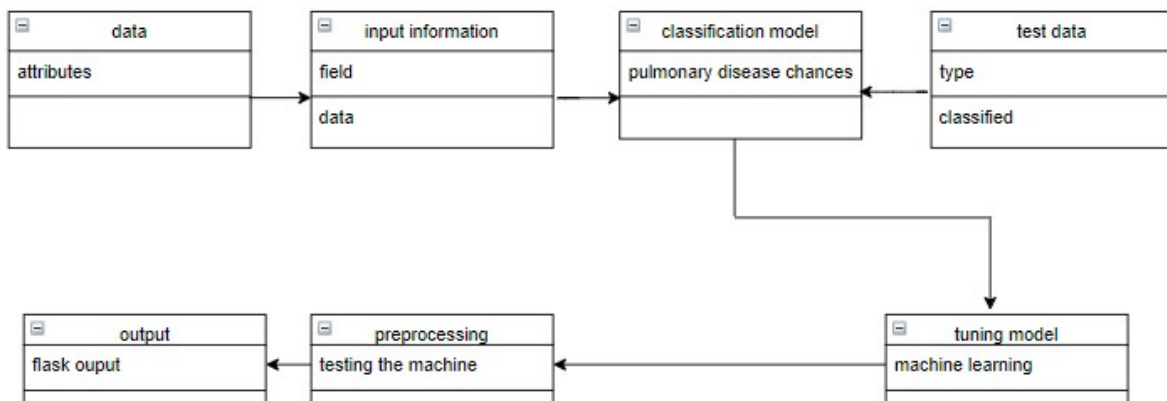


4.8 USE CASE DIAGRAM:



Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

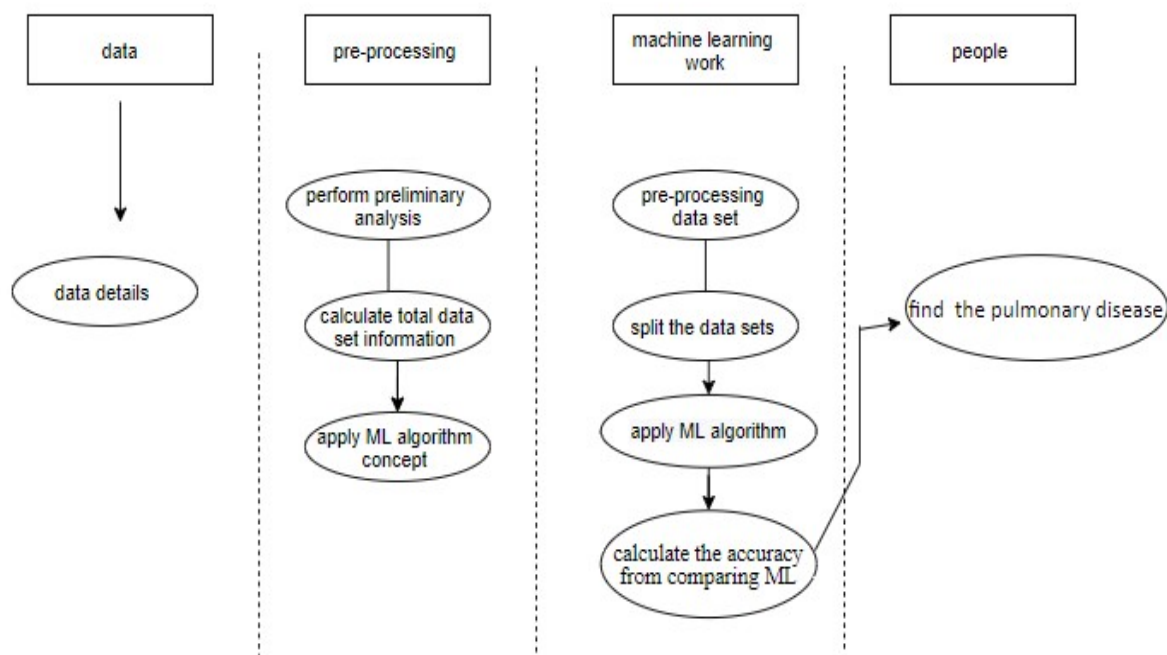
4.9 CLASS DIAGRAM:



Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of

class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

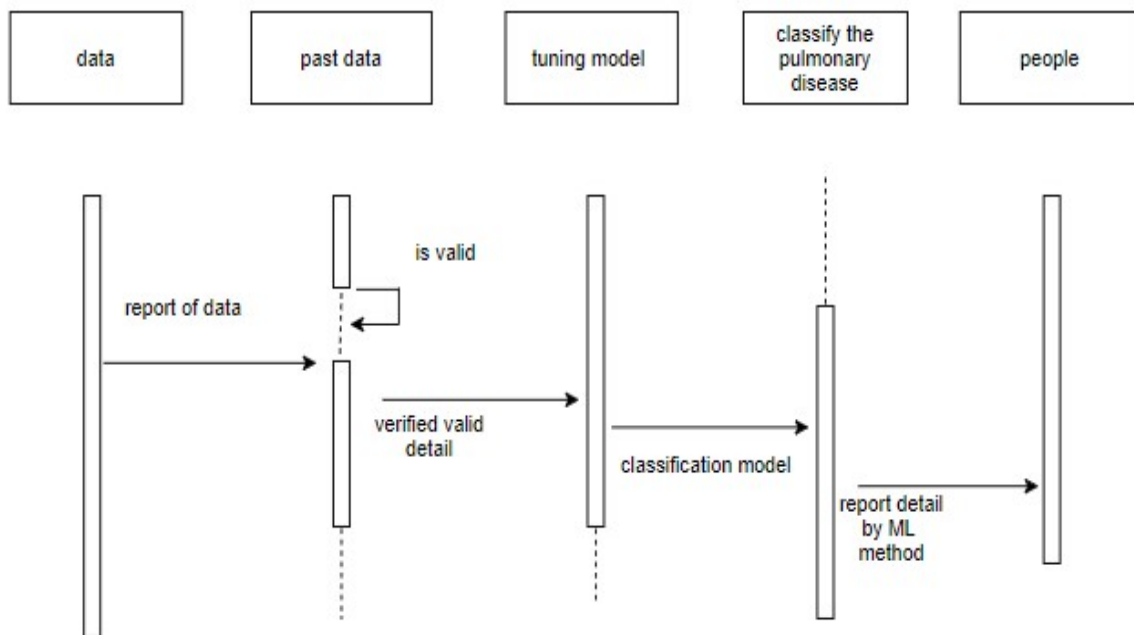
4.10 ACTIVITY DIAGRAM:



Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering

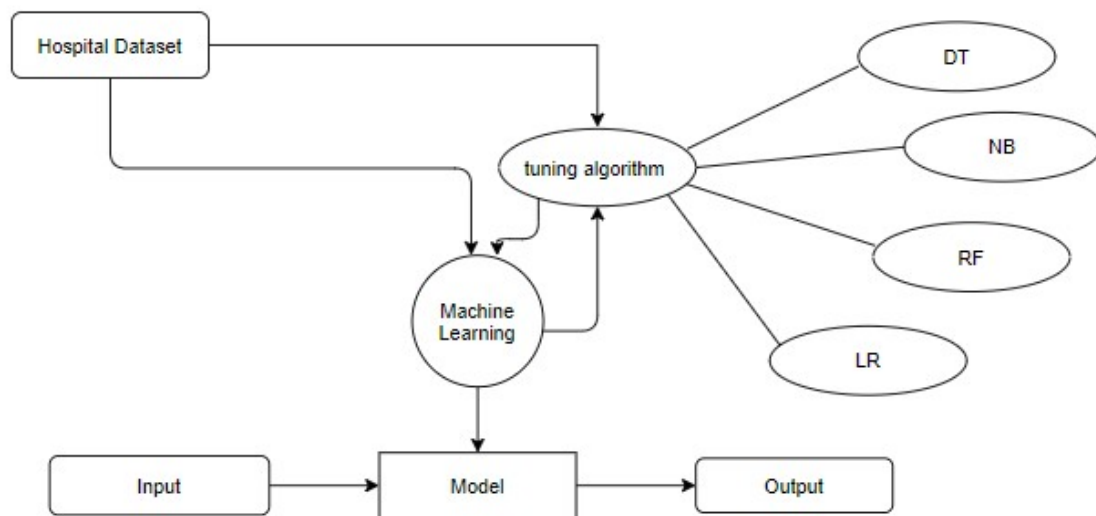
techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

4.11 SEQUENCE DIAGRAM:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Other dynamic modeling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

4.12 ENTITY RELATIONSHIP DIAGRAM (ERD):



An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

4.13 MODULE DESCRIPTION:

4.13.1 Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical

approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

import libraries for access and functional purpose and read the given dataset

General Properties of Analyzing the given dataset

Display the given dataset in the form of data frame

show columns

shape of the data frame

To describe the data frame

Checking data type and information about dataset

Checking for duplicate data

Checking Missing values of data frame

Checking unique values of data frame

Checking count values of data frame

Rename and drop the given data frame

To specify the type of values

To create extra columns

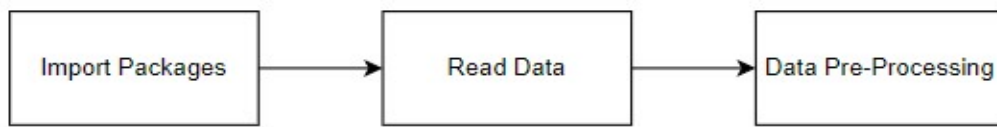
4.13.2 Data Validation/ Cleaning/Preparing Process

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC_DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL_CONSUM
count	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000
mean	61.824074	1.516667	1.444444	1.396148	1.392593	1.466667	1.607407	1.366667	1.394444	1.388889
std	9.603096	0.500185	0.497365	0.489970	0.488780	0.499350	0.488780	0.482341	0.489184	0.487778
min	21.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	62.000000	2.000000	1.000000	1.000000	1.000000	1.000000	2.000000	1.000000	1.000000	1.000000
75%	68.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
max	87.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000

	AGE	SMOKING	YELLOW_FINGERS	ANXIETY	PEER_PRESSURE	CHRONIC_DISEASE	FATIGUE	ALLERGY	WHEEZING	ALCOHOL_CONSUMING
AGE	1.000000	-0.120092	-0.030989	0.024377	0.073636	-0.018442	0.022808	-0.092992	0.090627	0.091025
SMOKING	-0.120092	1.000000	0.014915	0.052361	-0.026813	-0.194615	-0.011130	0.159183	-0.114115	-0.026399
YELLOW_FINGERS	-0.030989	0.014915	1.000000	0.490628	0.189097	-0.104583	-0.250151	-0.007734	0.101672	-0.079986
ANXIETY	0.024377	0.052361	0.490628	1.000000	0.244745	-0.010111	-0.236998	-0.045794	-0.108862	0.186743
PEER_PRESSURE	0.073636	-0.026813	0.189097	0.244745	1.000000	0.053716	-0.044811	0.017837	0.080525	-0.073246
CHRONIC_DISEASE	-0.018442	-0.194615	-0.104583	-0.010111	0.053716	1.000000	0.060304	0.120165	-0.094179	0.287086
FATIGUE	0.022808	-0.011130	-0.250151	-0.236998	-0.044811	0.060304	1.000000	0.108074	0.097940	0.038883
ALLERGY	-0.092992	0.159183	-0.007734	-0.045794	0.017837	0.120165	0.108074	1.000000	0.305868	-0.115045
WHEEZING	0.090627	-0.114115	0.101672	-0.108862	0.080525	-0.094179	0.097940	0.305868	1.000000	0.119532
ALCOHOL_CONSUMING	0.091025	-0.026399	-0.079986	0.186743	-0.073246	0.287086	0.038883	-0.115045	0.119532	1.000000
COUGHING	0.153586	-0.081187	0.186743	-0.073246	0.074419	-0.227705	0.017611	0.292725	0.530859	0.035036
SHORTNESS_OF_BREATH	-0.011143	0.020796	-0.148965	-0.355688	-0.253928	0.024326	0.491875	-0.015807	0.066086	0.020796
SWALLOWING_DIFFICULTY	-0.005605	0.049629	0.429339	0.507547	0.287086	0.038883	-0.115045	0.119532	0.194251	0.000142
CHEST_PAIN	-0.002051	0.151992	0.079742	0.000142	0.109728	-0.131428	-0.032856	0.343792	0.298445	0.000142

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

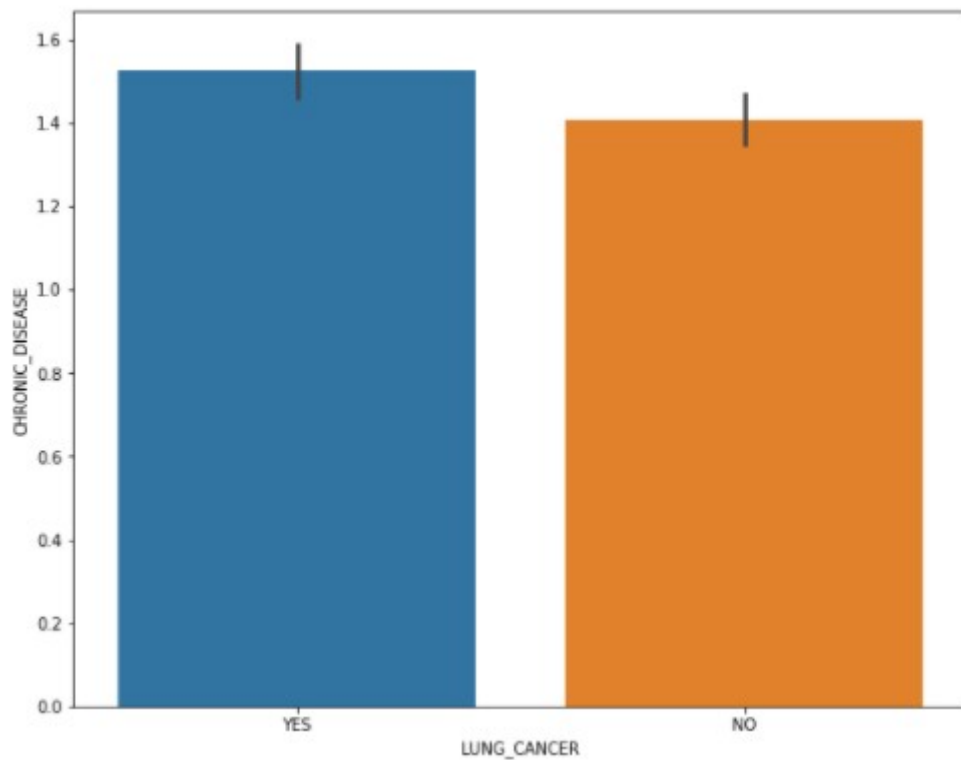
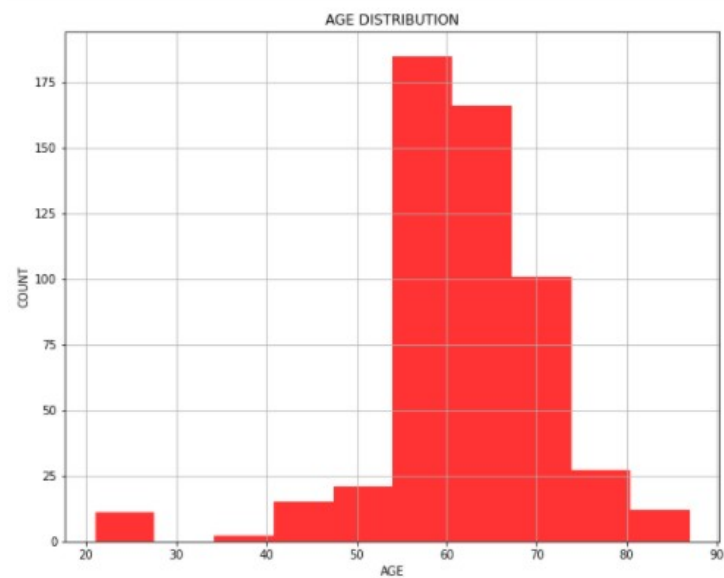
Exploration data analysis of visualization

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

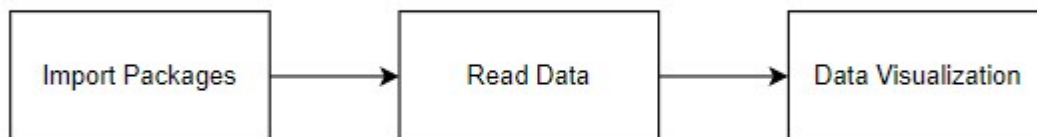
How to chart time series data with line plots and categorical quantities with bar charts.

How to summarize data distributions with histograms and box plots.



Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

4.13.3 Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with

scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

Logistic Regression

Random Forest

Decision Tree Classifier

Naïve Bayes

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in

precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To predicting the result by comparing accuracy.

ALGORITHM AND TECHNIQUES

Algorithm Explanation

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like `train_test_split`, `DecisionTreeClassifier` or `Logistic Regression` and `accuracy_score`.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

Binary logistic regression requires the dependent variable to be binary.

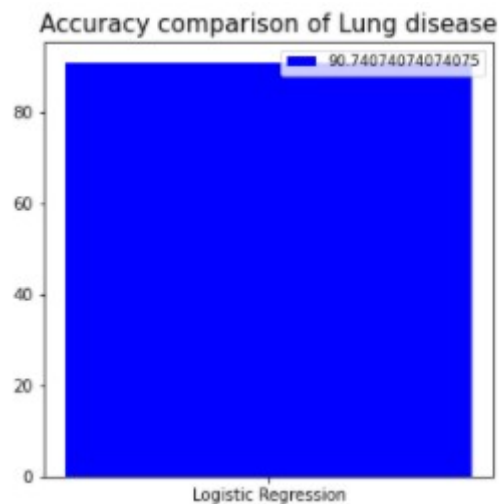
For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.

Only the meaningful variables should be included.

The independent variables should be independent of each other. That is, the model should have little.

The independent variables are linearly related to the log odds.

Logistic regression requires quite large sample sizes.



Classification report of Logistic Regression Results:

	precision	recall	f1-score	support
0	0.89	0.93	0.91	81
1	0.92	0.89	0.91	81
accuracy			0.91	162
macro avg	0.91	0.91	0.91	162
weighted avg	0.91	0.91	0.91	162

Confusion Matrix result of Logistic Regression is:

```
[[75  6]
 [ 9 72]]
```

Sensitivity : 0.9259259259259259

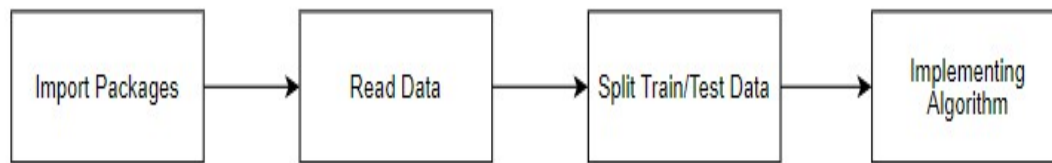
Specificity : 0.8888888888888888

Cross validation test results of accuracy:

```
[0.88888889 0.88888889 0.93518519 0.88888889 0.93518519]
```

Accuracy result of Logistic Regression is: 90.74074074074075

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

Pick N random records from the dataset.

Build a decision tree based on these N records.

Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Classification report of Random Forest Classifier Results:

	precision	recall	f1-score	support
0	0.96	1.00	0.98	81
1	1.00	0.96	0.98	81
accuracy			0.98	162
macro avg	0.98	0.98	0.98	162
weighted avg	0.98	0.98	0.98	162

Confusion Matrix result of Random Forest Classifier is:

```
[[81  0]
 [ 3 78]]
```

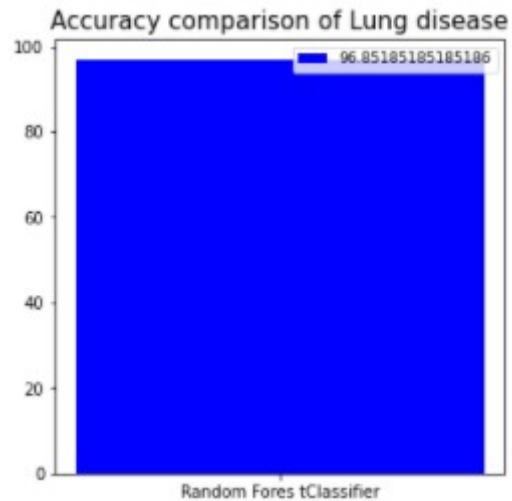
Sensitivity : 1.0

Specificity : 0.9629629629629629

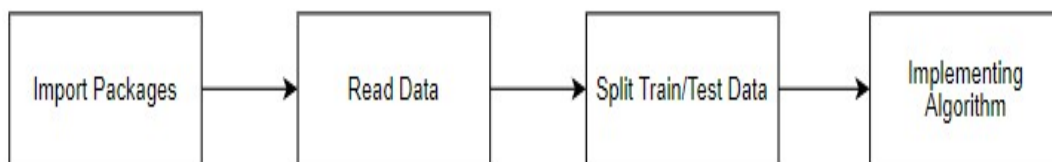
Cross validation test results of accuracy:

```
[0.96296296 0.94444444 1.          0.94444444 0.99074074]
```

Accuracy result of Random Forest Classifier is: 96.85185185185186



MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Decision Tree Classifier

It is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Assumptions of Decision tree:

At the beginning, we consider the whole training set as the root.

Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous.

On the basis of attribute values records are distributed recursively.

We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.

Classification report of Decision Tree Classifier Results:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	81
1	1.00	0.98	0.99	81
accuracy			0.99	162
macro avg	0.99	0.99	0.99	162
weighted avg	0.99	0.99	0.99	162

Confusion Matrix result of Decision Tree Classifier is:

```
[[81  0]
 [ 2 79]]
```

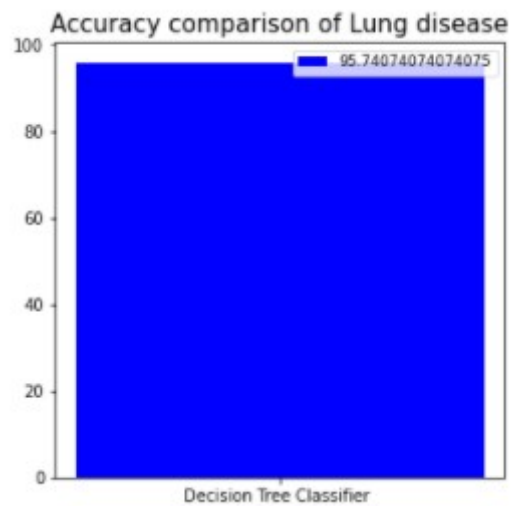
Sensitivity : 1.0

Specificity : 0.9753086419753086

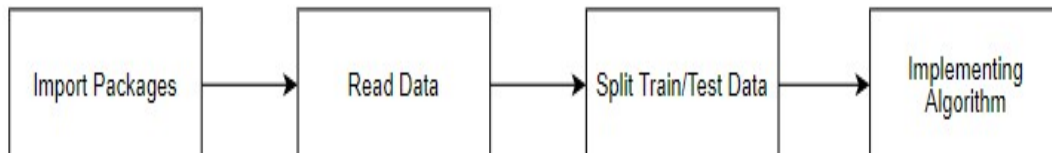
Cross validation test results of accuracy:

```
[0.9537037 0.93518519 0.99074074 0.91666667 0.99074074]
```

Accuracy result of Decision Tree Classifier is: 95.74074074074075



MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Naive Bayes algorithm:

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.

Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.

The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location.

Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive. This assumption is called class conditional independence.

Classification report of Naive Bayes Results:

	precision	recall	f1-score	support
0	0.89	0.86	0.87	81
1	0.87	0.89	0.88	81
accuracy			0.88	162
macro avg	0.88	0.88	0.88	162
weighted avg	0.88	0.88	0.88	162

Confusion Matrix result of Naive Bayes is:

```
[[70 11]
 [ 9 72]]
```

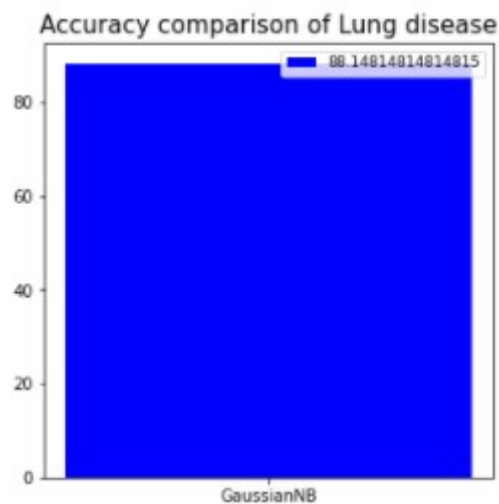
Sensitivity : 0.8641975308641975

Specificity : 0.8888888888888888

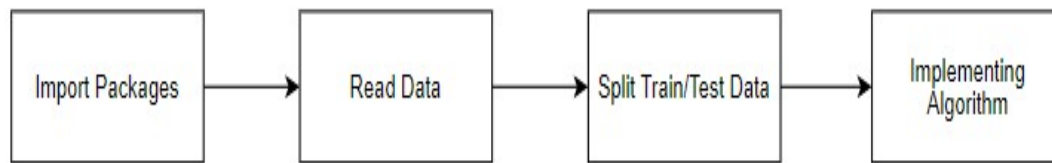
Cross validation test results of accuracy:

```
[0.86111111 0.92592593 0.84259259 0.87037037 0.90740741]
```

Accuracy result of Naive Bayes is: 88.14814814814815



MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

4.13.4 Deployment

Flask (Web FrameWork) :

Flask is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

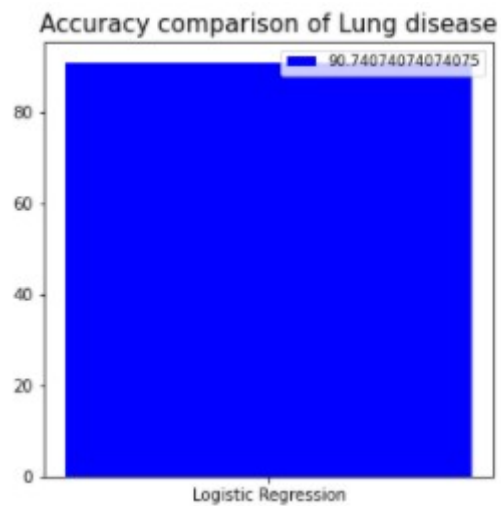
The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

CHAPTER 5 : RESULT AND DISCUSSION , PERFORMANCE ANALYSIS

5.1 Module Results:

5.1.1 Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).



Classification report of Logistic Regression Results:

	precision	recall	f1-score	support
0	0.89	0.93	0.91	81
1	0.92	0.89	0.91	81
accuracy			0.91	162
macro avg	0.91	0.91	0.91	162
weighted avg	0.91	0.91	0.91	162

Confusion Matrix result of Logistic Regression is:

```
[[75  6]
 [ 9 72]]
```

Sensitivity : 0.9259259259259259

Specificity : 0.8888888888888888

Cross validation test results of accuracy:

```
[0.88888889 0.88888889 0.93518519 0.88888889 0.93518519]
```

Accuracy result of Logistic Regression is: 90.74074074074075

5.1.2 Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Classification report of Random Forest Classifier Results:

	precision	recall	f1-score	support
0	0.96	1.00	0.98	81
1	1.00	0.96	0.98	81
accuracy			0.98	162
macro avg	0.98	0.98	0.98	162
weighted avg	0.98	0.98	0.98	162

Confusion Matrix result of Random Forest Classifier is:

```
[[81  0]
 [ 3 78]]
```

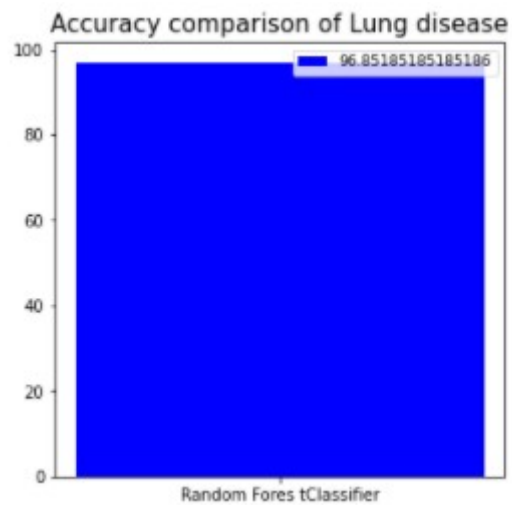
Sensitivity : 1.0

Specificity : 0.9629629629629629

Cross validation test results of accuracy:

```
[0.96296296 0.94444444 1.          0.94444444 0.99074074]
```

Accuracy result of Random Forest Classifier is: 96.85185185185186



5.1.3 Decision Tree Classifier

It is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

Classification report of Decision Tree Classifier Results:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	81
1	1.00	0.98	0.99	81
accuracy			0.99	162
macro avg	0.99	0.99	0.99	162
weighted avg	0.99	0.99	0.99	162

Confusion Matrix result of Decision Tree Classifier is:

```
[[81  0]
 [ 2 79]]
```

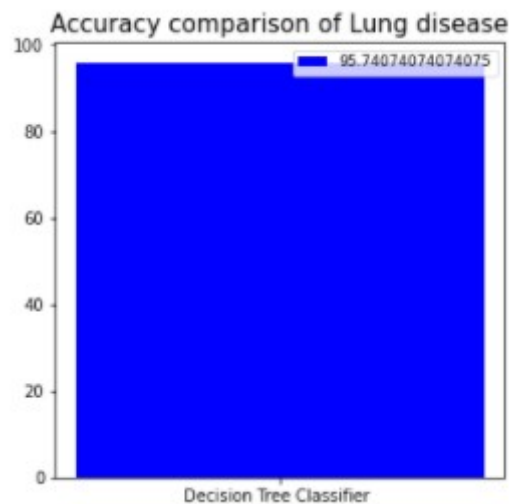
Sensitivity : 1.0

Specificity : 0.9753086419753086

Cross validation test results of accuracy:

```
[0.9537037 0.93518519 0.99074074 0.91666667 0.99074074]
```

Accuracy result of Decision Tree Classifier is: 95.74074074074075



5.1.4 Naive Bayes algorithm:

The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up

with if you wanted to model a predictive modeling problem probabilistically.

Classification report of Naive Bayes Results:

	precision	recall	f1-score	support
0	0.89	0.86	0.87	81
1	0.87	0.89	0.88	81
accuracy			0.88	162
macro avg	0.88	0.88	0.88	162
weighted avg	0.88	0.88	0.88	162

Confusion Matrix result of Naive Bayes is:

```
[[70 11]
 [ 9 72]]
```

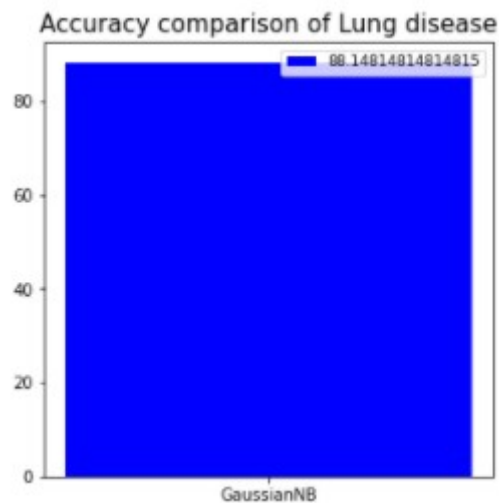
Sensitivity : 0.8641975308641975

Specificity : 0.8888888888888888

Cross validation test results of accuracy:

```
[0.86111111 0.92592593 0.84259259 0.87037037 0.90740741]
```

Accuracy result of Naive Bayes is: 88.14814814814815



5.2 Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm

to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

Accuracy: The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non-defaulters.

Accuracy calculation:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$F\text{- Measure} = 2TP / (2TP + FP + FN)$$

F1-Score Formula:

$$F1 \text{ Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

CHAPTER 6 : SUMMARY AND CONCLUSION

6.1 Coding:

Module – 1

Pre-Processing

#import library packages

import pandas **as** pd

import numpy **as** np

import warnings

warnings.filterwarnings("ignore")

#Load given dataset

data = pd.read_csv("Lung Cancer.csv")

Before drop the given dataset

data.head()

#shape

data.shape

After drop the given dataset

df = data.dropna()

df.head()

df.shape

#show columns

df.columns

#To describe the dataframe

```
df.describe()
```

```
#Checking datatype and information about dataset
```

```
df.info()
```

```
#Checking sum of missing values
```

```
df.isnull().sum()
```

```
df.GENDER.unique()
```

```
df.AGE.unique()
```

```
df.YELLOW_FINGERS.unique()
```

```
df.CHRONIC_DISEASE.unique()
```

```
df.ALCOHOL_CONSUMING.unique()
```

```
df.COUGHING.unique()
```

```
df.SHORTNESS_OF_BREATH.unique()
```

```
df.CHEST_PAIN.unique()
```

```
print("Minimum value of Age of patient is:", df.AGE.min())
```

```
print("Maximum value of Age of patient is:", df.AGE.max())
```

```
print("Age of patient ranges :", sorted(df['AGE'].unique()))
```

```
df.corr()
```

Before Pre-Processing

```
df.head()
```

After Pre-Processing

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod = ['GENDER','LUNG_CANCER']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(int)

df.head()
```

Module – 2

Visualization

```
#import library packages

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

data = pd.read_csv("Lung Cancer.csv")

df=data.dropna()

df.columns

pd.crosstab(df.GENDER,df.LUNG_CANCER)

pd.crosstab(df.CHRONIC_DISEASE,df.LUNG_CANCER)

df["AGE"].hist(figsize=(10,8), color="red", alpha=0.8)
```

```

plt.title("AGE DISTRIBUTION")

plt.xlabel("AGE")

plt.ylabel("COUNT")

plt.show()

#Propagation by variable

def PropByVar(df, variable):

    dataframe_pie = df[variable].value_counts()

    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize =

12)

    ax.set_title(variable + '\n', fontsize = 15)

    return np.round(dataframe_pie/df.shape[0]*100,2)

PropByVar(df, 'AGE')

fig, ax = plt.subplots(figsize=(10,8))

sns.barplot(df.LUNG_CANCER, df.CHRONIC_DISEASE, ax=ax)

plt.show()

fig, ax = plt.subplots(figsize=(15,6))

sns.boxplot(df.AGE, ax =ax)

plt.title("Age distribution")

plt.show()

fig, ax = plt.subplots(figsize=(10,6))

sns.stripplot(df.LUNG_CANCER, df.AGE)

plt.title("Lung Cancer predicted for a age")

```

```

plt.show()

# Heatmap plot diagram

fig, ax = plt.subplots(figsize=(15,10))

sns.heatmap(df.corr(), ax=ax, annot=True)

from sklearn.preprocessing import LabelEncoder

var_mod = ['GENDER','LUNG_CANCER']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(int)

df.head()

```

Module – 3

Logistic Regression Algorithm

```

#import library packages

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import warnings

warnings.filterwarnings('ignore')

```

#Load given dataset

```
data = pd.read_csv("Lung.csv")
```

```
df=data.dropna()
```

```
df.columns
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the test set.

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

```
accuracy_score, roc_auc_score
```

```
from sklearn.preprocessing import LabelEncoder
```

```
var_mod = ['GENDER','LUNG_CANCER']
```

```
le = LabelEncoder()
```

```
for i in var_mod:
```

```
    df[i] = le.fit_transform(df[i]).astype(int)
```

```
X = df.drop(labels='LUNG_CANCER', axis=1)
```

#Response variable

```
y = df.loc[:, 'LUNG_CANCER']
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

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

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

```
random_state=1, stratify=y)
```

Logistic Regression :

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

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

```
from sklearn.model_selection import cross_val_score
```

```
logR= LogisticRegression()
```

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

```
predictR = logR.predict(X_test)
```

```
print("""
```

```
print('Classification report of Logistic Regression Results:')
```

```
print("""
```

```
print(classification_report(y_test,predictR))
```

```
print("""
```

```
cm=confusion_matrix(y_test,predictR)
```

```
print('Confusion Matrix result of Logistic Regression is:\n',cm)
```

```
print("""
```

```
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
```

```
print('Sensitivity : ', sensitivity )
```

```
print("""
```

```

specificity = cm[1,1]/(cm[1,0]+cm[1,1])

print('Specificity : ', specificity)

print("")

accuracy = cross_val_score(logR, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Logistic Regression is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="Logistic Regression"

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

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison of Lung disease",fontsize=15)

    plt.legend(b,data,fontsize=9)

graph()

TP = cm[0][0]

FP = cm[1][0]

```

```

FN = cm[1][1]
TN = cm[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion matrix-LogisticRegression',
cmap=plt.cm.Blues):

```

```

target_names=['Predict','Actual']

plt.imshow(cm2, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick_marks = np.arange(len(target_names))

plt.xticks(tick_marks, target_names, rotation=45)

plt.yticks(tick_marks, target_names)

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')


cm2=confusion_matrix(y_test, predictR)

print('Confusion matrix-LogisticRegression:')

print(cm2)

plot_confusion_matrix(cm2)

```

Module – 4

Random Forest Algorithm

#import library packages

import pandas **as** pd

import matplotlib.pyplot **as** plt

import seaborn **as** sns

```

import numpy as np

import warnings

warnings.filterwarnings('ignore')

#Load given dataset

data = pd.read_csv("Lung.csv")

df=data.dropna()

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, roc_auc_score

from sklearn.preprocessing import LabelEncoder

var_mod = ['GENDER','LUNG_CANCER']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(int)

X = df.drop(labels='LUNG_CANCER', axis=1)

#Response variable

y = df.loc[:, 'LUNG_CANCER']

#We'll use a test size of 30%. We also stratify the split on the response variable,
which is very important to do because there are so few fraudulent transactions.

from sklearn.model_selection import train_test_split

```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
random_state=1, stratify=y)
```

RandomForestClassifier:

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

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import cross_val_score
```

```
rfc = RandomForestClassifier()
```

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

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

```
print("")
```

```
print('Classification report of Random Forest Classifier Results:')
```

```
print("")
```

```
print(classification_report(y_test,predictR))
```

```
print("")
```

```
cm=confusion_matrix(y_test,predictR)
```

```
print('Confusion Matrix result of Random Forest Classifier is:\n',cm)
```

```
print("")
```

```
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
```

```
print('Sensitivity : ', sensitivity )
```

```
print("")
```

```
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
```

```

print('Specificity : ', specificity)

print("")

accuracy = cross_val_score(rfc, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Random Forest Classifier is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="Random Forest Classifier"

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

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison of Lung disease",fontsize=15)

    plt.legend(b,data,fontsize=9)

graph()

TP = cm[0][0]

FP = cm[1][0]

FN = cm[1][1]

TN = cm[0][1]

```

```

print("True Positive :",TP)

print("True Negative :",TN)

print("False Positive :",FP)

print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)

TNR = TN/(TN+FP)

FPR = FP/(FP+TN)

FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)

print("True Negative Rate :",TNR)

print("False Positive Rate :",FPR)

print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)

print("Negative predictive value :",NPV)

def plot_confusion_matrix(cm2, title='Confusion
matrix-RandomForestClassifier', cmap=plt.cm.Blues):

    target_names=['Predict','Actual']

    plt.imshow(cm2, interpolation='nearest', cmap=cmap)

```

```

plt.title(title)

plt.colorbar()

tick_marks = np.arange(len(target_names))

plt.xticks(tick_marks, target_names, rotation=45)

plt.yticks(tick_marks, target_names)

plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')


cm2=confusion_matrix(y_test, predictR)

print('Confusion matrix-RandomForestClassifier:')

print(cm2)

plot_confusion_matrix(cm2)

import joblib

joblib.dump(rfc,"model.pkl")

['model.pkl']

```

Module – 5

Decision Tree Algorithm

#import library packages

```

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import warnings

warnings.filterwarnings('ignore')

#Load given dataset

data = pd.read_csv("Lung.csv")

df=data.dropna()

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, roc_auc_score

from sklearn.preprocessing import LabelEncoder

var_mod = ['GENDER','LUNG_CANCER']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(int)

X = df.drop(labels='LUNG_CANCER', axis=1)

#Response variable

y = df.loc[:, 'LUNG_CANCER']

```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
random_state=1, stratify=y)
```

Decision Tree Classifier:

```
from sklearn.metrics import accuracy_score, confusion_matrix  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.model_selection import cross_val_score  
dtc = DecisionTreeClassifier()  
dtc.fit(X_train,y_train)  
predictR = dtc.predict(X_test)  
print("")  
print('Classification report of Decision Tree Classifier Results:')  
print("")  
print(classification_report(y_test,predictR))  
  
print("")  
cm=confusion_matrix(y_test,predictR)  
print('Confusion Matrix result of Decision Tree Classifier is:\n',cm)  
print("")
```

```

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])

print('Sensitivity : ', sensitivity )

print("")

specificity = cm[1,1]/(cm[1,0]+cm[1,1])

print('Specificity : ', specificity)

print("")

accuracy = cross_val_score(dtc, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Decision Tree Classifier is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="Decision Tree Classifier "

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

    b=plt.bar(alg,data,color=("b"))

    plt.title("Accuracy comparison of Lung disease",fontsize=15)

    plt.legend(b,data,fontsize=9)

graph()

```

```

TP = cm[0][0]
FP = cm[1][0]
FN = cm[1][1]
TN = cm[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)
NPV = TN/(TN+FN)

print("Positive Predictive Value :",PPV)
print("Negative predictive value :",NPV)

```

```

def plot_confusion_matrix(cm2, title='Confusion
matrix-DecisionTreeClassifier', cmap=plt.cm.Blues):

    target_names=['Predict','Actual']

    plt.imshow(cm2, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick_marks = np.arange(len(target_names))

    plt.xticks(tick_marks, target_names, rotation=45)

    plt.yticks(tick_marks, target_names)

    plt.tight_layout()

    plt.ylabel('True label')

    plt.xlabel('Predicted label')

cm2=confusion_matrix(y_test, predictR)

print('Confusion matrix-DecisionTreeClassifier:')

print(cm2)

plot_confusion_matrix(cm2)

```

Module – 6

Naïve Bayes Algorithm

#import library packages

```

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import warnings

warnings.filterwarnings('ignore')

#Load given dataset

data = pd.read_csv("Lung.csv")

df=data.dropna()

df.columns

#According to the cross-validated MCC scores, the random forest is the
best-performing model, so now let's evaluate its performance on the test set.

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, roc_auc_score

from sklearn.preprocessing import LabelEncoder

var_mod = ['GENDER','LUNG_CANCER']

le = LabelEncoder()

for i in var_mod:

    df[i] = le.fit_transform(df[i]).astype(int)

X = df.drop(labels='LUNG_CANCER', axis=1)

#Response variable

y = df.loc[:, 'LUNG_CANCER']

```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
random_state=1, stratify=y)
```

Naive Bayes:

```
from sklearn.metrics import accuracy_score, confusion_matrix  
from sklearn.naive_bayes import GaussianNB  
from sklearn.model_selection import cross_val_score
```

```
nb = GaussianNB()
```

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

```
predictR = nb.predict(X_test)
```

```
print("")
```

```
print('Classification report of Naive Bayes Results:')
```

```
print("")
```

```
print(classification_report(y_test,predictR))
```

```
print("")
```

```

cm=confusion_matrix(y_test,predictR)

print('Confusion Matrix result of Naive Bayes is:\n',cm)

print("")

sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])

print('Sensitivity : ', sensitivity )

print("")

specificity = cm[1,1]/(cm[1,0]+cm[1,1])

print('Specificity : ', specificity)

print("")

accuracy = cross_val_score(nb, X, y, scoring='accuracy')

print('Cross validation test results of accuracy:')

print(accuracy)

#get the mean of each fold

print("")

print("Accuracy result of Naive Bayes is:",accuracy.mean() * 100)

LR=accuracy.mean() * 100

def graph():

    import matplotlib.pyplot as plt

    data=[LR]

    alg="GaussianNB"

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

    b=plt.bar(alg,data,color=("b"))

```

```

plt.title("Accuracy comparison of Lung disease",fontsize=15)

plt.legend(b,data,fontsize=9)

graph()

TP = cm[0][0]
FP = cm[1][0]
FN = cm[1][1]
TN = cm[0][1]

print("True Positive :",TP)
print("True Negative :",TN)
print("False Positive :",FP)
print("False Negative :",FN)

print("")

TPR = TP/(TP+FN)
TNR = TN/(TN+FP)
FPR = FP/(FP+TN)
FNR = FN/(TP+FN)

print("True Positive Rate :",TPR)
print("True Negative Rate :",TNR)
print("False Positive Rate :",FPR)
print("False Negative Rate :",FNR)

print("")

PPV = TP/(TP+FP)

```

$NPV = TN/(TN+FN)$

```
print("Positive Predictive Value :",PPV)
```

```
print("Negative predictive value :",NPV)
```

```
def plot_confusion_matrix(cm2, title='Confusion matrix-Naive Bayes',
```

```
cmap=plt.cm.Blues):
```

```
    target_names=['Predict','Actual']
```

```
    plt.imshow(cm2, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title)
```

```
    plt.colorbar()
```

```
    tick_marks = np.arange(len(target_names))
```

```
    plt.xticks(tick_marks, target_names, rotation=45)
```

```
    plt.yticks(tick_marks, target_names)
```

```
    plt.tight_layout()
```

```
    plt.ylabel('True label')
```

```
    plt.xlabel('Predicted label')
```

```
cm2=confusion_matrix(y_test, predictR)
```

```
print('Confusion matrix-Naive Bayes:')
```

```
print(cm2)
```

```
plot_confusion_matrix(cm2)
```

HTML Code:

```
<!DOCTYPE html>
```

```
<html >
```

```

<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>TITLE</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css')
  }}">
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'
  type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
  type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'
  type='text/css'>
  <link
  href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
  rel='stylesheet' type='text/css'>
  <style>
  .back{
    background-image: url("{{ url_for('static', filename='image/img.png') }}" );
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-size: 100% 100%;
  }
  .white{
    color:white;
  }
  .space{
    margin:10px 30px;
    padding:10px 10px;
    background: palegreen;

```

```

width:500px
}
.gap{
padding:10px 20px;
}
</style>

</head>

<body class="back">
<div>
    <div class="jumbotron">
        <h1 style="text-align:center">DIOGNOSIS OF LUNG DISEASE</h1>

    </div>

    <!-- Main Input For Receiving Query to our ML -->
    <form class="form-group" action="{{ url_for('predict')}}"method="post">

        <div class="row">
            <div class="gap col-md-6 ">

                <label class="white" for="">GENDER</label>
                <select class="space form-control" name="GENDER"
id="GENDER">

                    <option value=0>FEMALE</option>
                    <option value=1>MALE</option>
                </select>

                <label class="white" for="">AGE</label>

```

```
<input type="number" class="space form-control" step="0.01"
name="AGE" placeholder="AGE" required="required" /><br>
```

```
<label class="white" for="">SMOKING</label>
<input type="number" class="space form-control" step="0.01"
name="SMOKING" placeholder="SMOKING" required="required" /><br>
```

```
<label class="white" for="">YELLOW FINGERS</label>
<input type="number" class="space form-control" step="0.01"
name="YELLOW FINGERS" placeholder="YELLOW FINGERS"
required="required" /><br>
```

```
<label class="white" for="">ANXIETY</label>
<input type="number" class="space form-control" step="0.01"
name="ANXIETY" placeholder="ANXIETY" required="required" /><br>
```

```
<label class="white" for="">PEER PRESSURE</label>
<input type="number" class="space form-control" step="0.01"
name="PEER PRESSURE" placeholder="PEER PRESSURE"
required="required" /><br>
```

```
<label class="white" for="">CHRONIC DISEASE</label>
<input type="number" class="space form-control" step="0.01"
name="CHRONIC DISEASE" placeholder="CHRONIC DISEASE"
required="required" /><br>
```

```
<label class="white" for="">FATIGUE</label>
<input type="number" class="space form-control" step="0.01"
name="FATIGUE" placeholder="FATIGUE" required="required" /><br>
```

```

</div>

<div class="gap col-md-6">

  <label class="white" for="">ALLERGY</label>
  <input type="number" class="space form-control" step="0.01"
name="ALLERGY" placeholder="ALLERGY" required="required" /><br>

  <label class="white" for="">WHEEZING</label>
  <input type="number" class="space form-control" step="0.01"
name="WHEEZING" placeholder="WHEEZING" required="required" /><br>

  <label class="white" for="">ALCOHOL
CONSUMING</label>
  <input type="number" class="space form-control" step="0.01"
name="ALCOHOL CONSUMING" placeholder="ALCOHOL CONSUMING"
required="required" /><br>

  <label class="white" for="">COUGHING</label>
  <input type="number" class="space form-control" step="0.01"
name="COUGHING" placeholder="COUGHING" required="required" /><br>

  <label class="white" for="">SHORTNESS OF
BREATH</label>
  <input type="number" class="space form-control" step="0.01"
name="SHORTNESS OF BREATH" placeholder="SHORTNESS OF
BREATH" required="required" /><br>

  <label class="white" for="">SWALLOWING
DIFFICULTY</label>

```

```

        <input type="number" class="space form-control" step="0.01"
name=SWALLOWING DIFFICULTY" placeholder="SWALLOWING
DIFFICULTY" required="required" /><br>
        <label class="white" for="">CHEST PAIN</label>
        <input type="number" class="space form-control" step="0.01"
name="CHEST PAIN" placeholder="CHEST PAIN" required="required"
/><br>
</div>
</div>
<div style="padding:2% 35%">
    <button type="submit" class="btn btn-success btn-block"
style="width:350px;padding:20px">Predict</button>
</div>
    </form>
    <br>
    <br>
<div style="background:skyblue;padding:2% 40%">
    {{ prediction_text }}
</div>
</div>
</body>
</html>

```

Flask Deploy:

```

import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

import joblib

```

```

app = Flask(__name__)

model = joblib.load('model.pkl')

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/predict',methods=['POST'])

def predict():

    """
    For rendering results on HTML GUI
    """
    int_features = [(x) for x in request.form.values()]

    final_features = [np.array(int_features)]

    print(final_features)

    prediction = model.predict(final_features)

    print(prediction)

    output = prediction[0]

    if output == 1:

        return render_template('index.html', prediction_text='LUNG
AFFECTED')

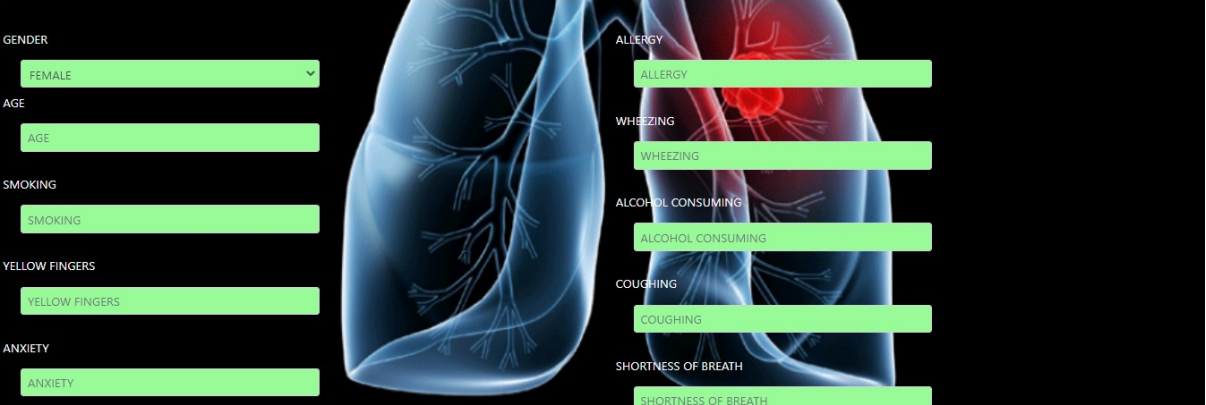
    else:

        return render_template('index.html', prediction_text='LUNG NOT
AFFECTED')

    print(output)

```

```
if __name__ == "__main__":  
    app.run(host="localhost", port=8000)
```



6.2 Conclusion

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This application can help to find the Prediction of Pulmonary Disease.

6.3 Future Work

Pulmonary Disease prediction to connect with cloud.
To optimize the work to implement in Artificial Intelligence environment.

References