One Year Life Expectancy Post Thoraic Surgery Using IBM Watson Machine Learning

Industrial/Practical Training Report

Submitted to the

Department of Electronics and Communication Engineering,

SESHADRI RAO GUDLAVALLERU ENGINEERING COLLEGE

In partial fulfillment of the requirements,

for the award of the Degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

GUDIVADA JANARDHAN

(19481A0470)

Department of Electronics and Communication Engineering

SESHADRIRAO GUDLAVALLERU ENGINEERING COLLEGE

SESHADRI RAO KNOWLEDGE VILLAGE

GUDLAVALLERU – 521356

ANDHRA PRADESH

2022-23

DECLARATION

I G. Janardhan hereby declare that this industrial training report is the record of authentic work carried out by me during the period 03.05.2022 to 07.07.2023 in **Smart Bridge Pvt.Ltd** under the supervision of my training In charge Srikanth Kumar Smart Bridge Pvt.Ltd.

Signature:

Name of the student: G. Janardhan

Acknowledgement

I am very glad to express my deep indebtedness to all the employees of Smart Bridge Pvt.Ltd. Without their support and guidance, it would not have been possible for this training to have materialized and taken a concrete shape. I extend my deep gratitude to my training mentor – Sandeep Doodigani, Divya Nemuri, Gnaneswar Bandari, Ancy Jenifer J, who supported us all the time. I owe my personal thanks to our HOD **Dr.Y.Rama Krishna** and our industrial training coordinator **Mr. N.Samba Murthy** for undergoing training at a reputed company like Smart Bridge Pvt.Ltd.

We would like to take this opportunity to express our profound sense of gratitude to our beloved Principal **Dr.G.V.S.N.R.V.Prasad**, for providing us all the required facilities. We express our thanks to Teaching and Non-Teaching staff of electronics and communication engineering department who helped us directly or indirectly in completion of our internship.

Table of content:

List of Figures

1.Introduction

Overview:

Lung cancer is the leading cause of cancer-related deaths in the world. In the United States, lung cancer claims more lives every year than colon cancer, prostate cancer, and breast cancer combined.

The American Cancer Society's estimates for lung cancer in the United States for 2018 are:

- About 234,030 new cases of lung cancer (121,680 in men and 112,350 in women)
- About 154,050 deaths from lung cancer (83,550 in men and 70,500 in women)

Despite the very serious prognosis (outlook) of lung cancer, some people with earlier-stage cancers are cured. More than 430,000 people alive today have been diagnosed with lung cancer at some point. The data is dedicated to classification problems related to the post-operative life expectancy in lung cancer patients: class 1 - death within one year after surgery, class 2 - survival.

We will be using classification algorithms such as Decision tree, Random forest, KNN, and xgboost. We will train and test the data with these algorithms. From this best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Purpose:

The main aim of this project is to create a model based on one year life expectancy post thoraic surgery. In this project we are using machine learning techniques for the accurate prediction. An application is also build which can be interlinked with the model so as to view the result on Web Page based on the input parameters.

2. literature Survey

Existing problem:

There are just a few systems that attempt to predict survival after pectoral surgery. Existing solutions just provide mil models for matter statements, but no front-end interface for end users. There isn't a lot of information on programming at the United Nations. Also, the types of gift systems that are now available have an accuracy range of 70% to 85%.

Proposed solution:

The system aims at forecasting the survival of lung cancer who are infected patient by post thoracic surgery. The dataset has been included with 17 attributes which are specified in Table1. Among all those attributes, Risk 1YEAR is the target class mentioning zero if the patient survives for at least one-year post thoracic surgery and one for those who died before completing one-year post surgery. The visualization of dataset is been done using the Matplotlib and Seaborn libraries of Python. Furthermore the essential attributes are found based on the Information Gain (IG) attribute evaluation which is been used to find the importance of attribute by using the Information Gain with respect to the target class. IG (Class, Attributes) = E (Class) – E (Class | Attributes). where, E stands for Entropy After the IG Attribute Evaluation on all the 16 independent attributes in the dataset, the essential attributes would be decided which will be used to train the model.

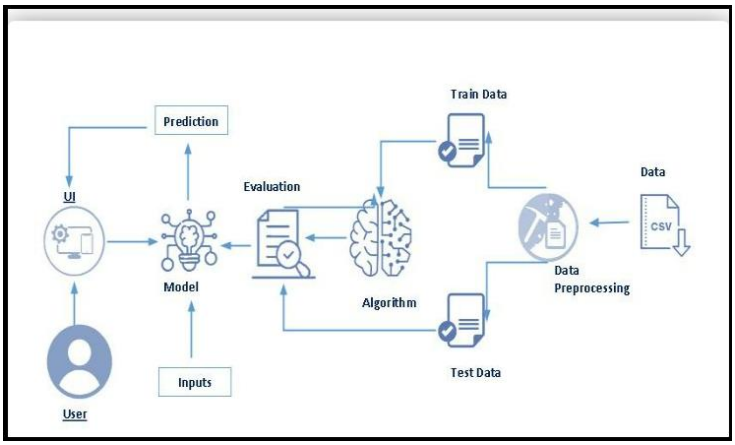


Fig. Proposed system

3. Flow Chart

- The user interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data collection
  - Collect the dataset or create the dataset
- Visualizing and analyzing data
  - Import Libraries
  - Read The dataset
  - Descriptive analysis
  - Exploratory Data Analysis
- Data pre-processing
  - Handling Missing Values
  - Drop the unwanted columns
  - Splitting data into train and test
  - Feature Scaling
- Model building
  - Import the model building libraries
  - Initializing the model
  - Training and testing the model
  - Evaluating the performance of the model
  - Save the model
- Application Building
  - Create an HTML file
  - Build python code

4. Theoretical Analysis

4.1 MACHINE LEARNING

Machine learning is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "Training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks**.**

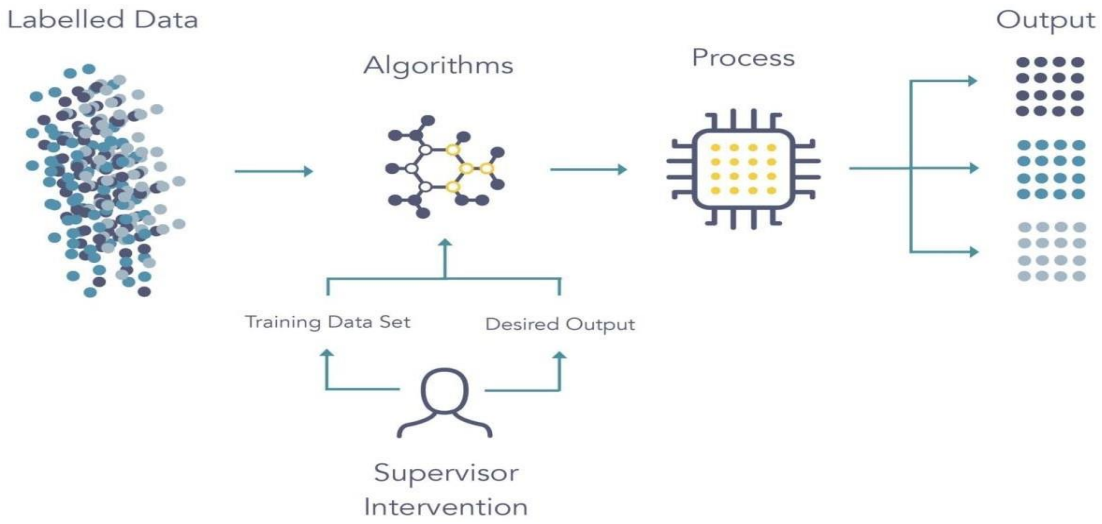


Fig: Machine Learning

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers, but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytic.

4.2 DEEP LEARNING

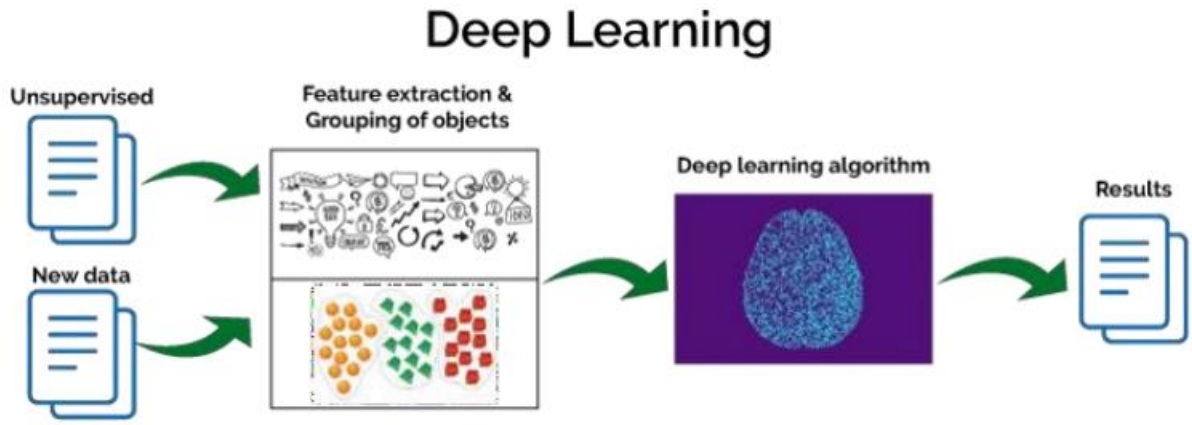


Fig: Deep Learning

A deep learning model is designed to continually analyse data with a logic structure similar to how a human would draw conclusions. To achieve this, deep learning applications use a layered structure of algorithms called an artificial neural network. The design of an artificial neural network is inspired by the biological neural network of the human brain, leading to a process of learning that's far more capable than that of standard machine learning models.

It's a tricky prospect to ensure that a deep learning model doesn't draw incorrect conclusions—like other examples of AI, it requires lots of training to get the learning processes correct. But when it works as it's intended to, functional deep learning is often received as a scientific marvel that many consider being the backbone of true artificial intelligence.

A great example of deep learning is Google's AlphaGo. Google created a computer program with its own neural network that learned to play the abstract board game called Go, which is known for requiring sharp intellect and intuition. By playing against professional Go players, AlphaGo's deep learning model learned how to play at a level never seen before in artificial intelligence, and did without being told when it should make a specific move (as a standard machine learning model would

require). It caused quite a stir when AlphaGo defeated multiple world-renowned "masters" of the game—not only could a machine grasp the complex techniques and abstract aspects of the game, it was becoming one of the greatest players of it as well.

Relation between AI,ML,DL:

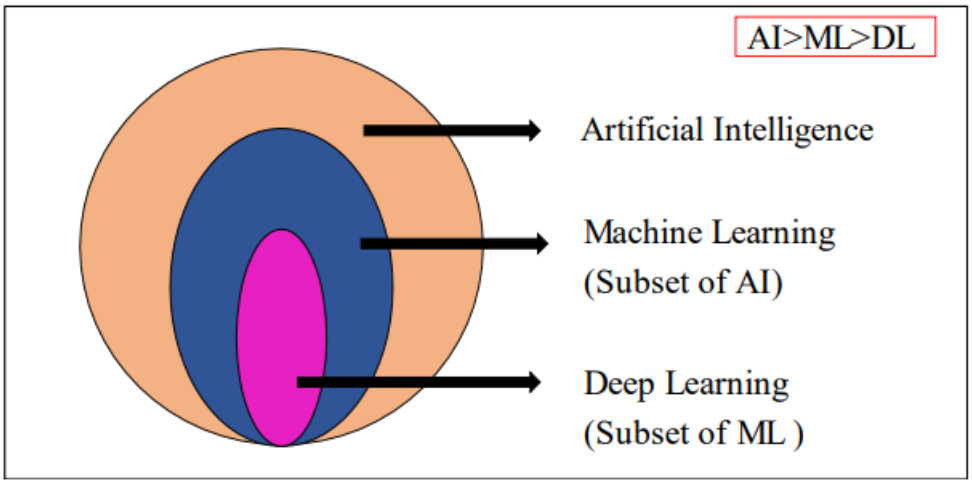


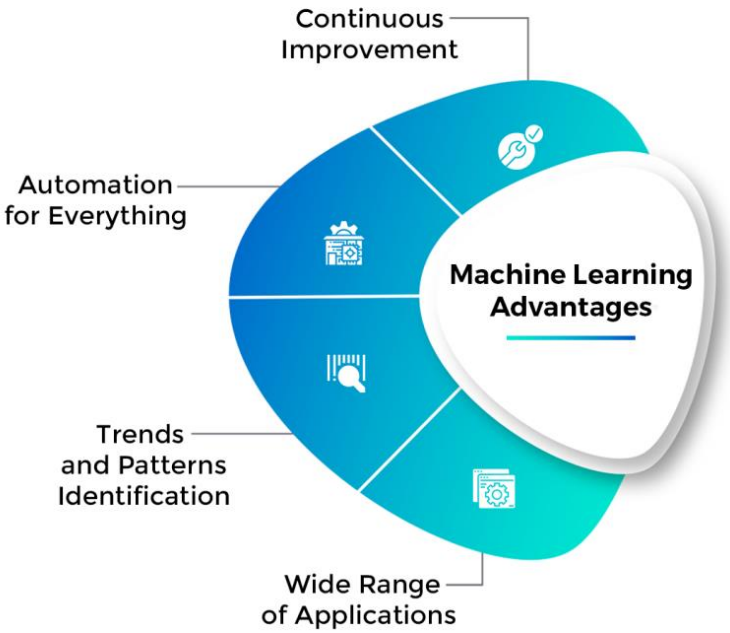
Fig: Relation between AI,ML,DL

ML vs DL

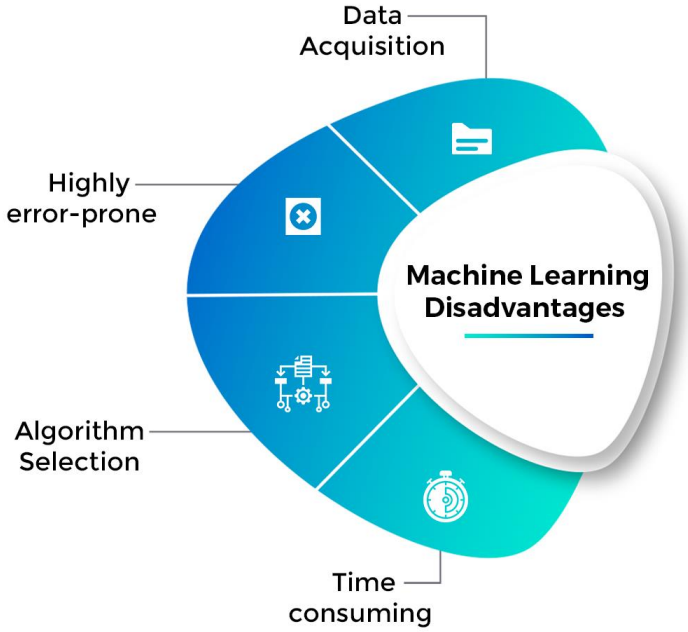
To recap the differences between the two:

- The easiest takeaway for understanding the difference between machine learning and deep learning is to know that **Deep Learning is Machine Learning.**
- More specifically, deep learning is considered an evolution of machine learning. It uses a programmable neural network that enables machines to make accurate decisions without help from humans.
- Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned
- Deep learning is a subfield of machine learning. While both fall under the broad category of artificial intelligence
- Deep learning is what powers the most human-like artificial intelligence

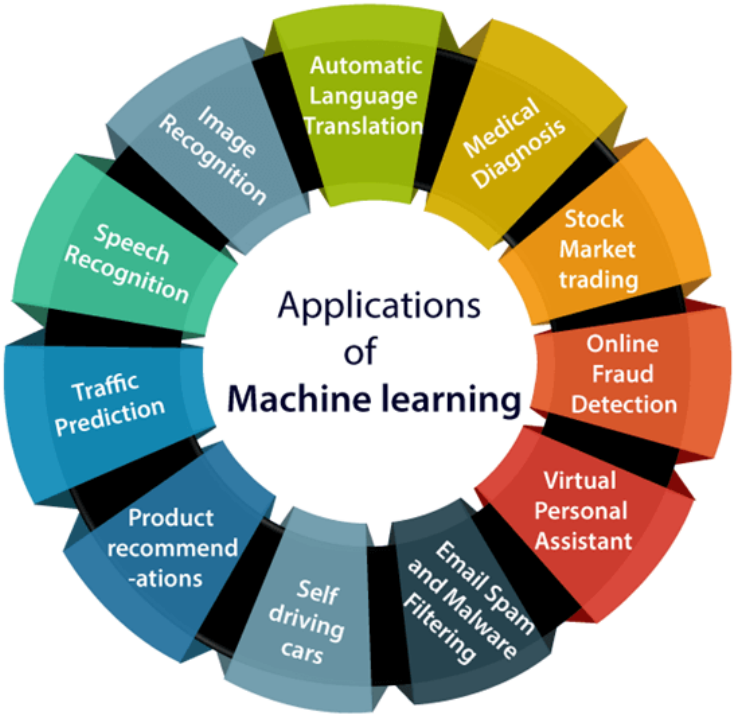
ADVANTAGES OF MACHINE LEARNING



DISADVANTAGES OF MACHINE LEARNING



APPLICATIONS OF MACHINE LEARNING



SYSTEM DESIGN:

A. **IMPLEMENTATION**: Python is typically called interactive programming language that is object-oriented and high level. Guido van Rossum founded Python from 1985-1990. Like Perl, Python's ASCII text file was also preferred under a general public licence ante lope (GPL). Python is typically called interactive programming language that is object-oriented and high level. Guido van Rossum founded Python from 1985- 1990. Like Perl, Python's ASCII text file was also preferred under a general public licence ante lope (GPL).

B. Flash Black: Flask is a small python-language web framework. Flask can be classified as a small framework and requires no specific libraries or tools. Flask supports extensions that are enforced in flasks alone by adding application options. Extensions will exist for relational object mappers, transfers, licenced and many other similar tools connected to authentications. Unit area extensions update more than Core Flask. MongoDB uses the Flask, which provides extra data history management. Flask Framework is used to use Pinterest LinkedIn and the flask itself for community based web content.

Hypertext mark-up language (Front-End) : For establishing sites and net applications, HTML is a normal nomenclature. Language hypertext markup describes the site markup structure. Language labels that include items such as "heading," "paragraph," "table" and so on can be marked in hypertext. Browser does not show the hypertext tags used to render the page contents. The traditional cornerstone technologies for the worldwide network are the cascades of vogue sheets(CSS) and JavaScript. Browser receives hypertext markup language documents which are

stored regularly and rendered to the Web sites by the web server. Semantically express the structure of an internet page originally surrounded for the outline document with the hypertext labelling language.

CSS (Front-End): Cascading mode sheets (CSS) is a piece of paper used to describe the presentation in marking language of a document. Although the visual sites that are continuously visited are the type and user interfaces of sites written in hypertext and XHTML, the language was often used on any XML documents together with XML SVG, XUL and XUL, which are applicable to speech rendering and other media. JavaScript and CSS are the main technology used by most wedding websites for visually stable web pages, user interfaces of networking applications and user interfaces in the different mobile application fields together with the Hypertext marking language. CSS is used primarily to modify display and content separations along with layout direction, colours and fonts. Separation enhances content accessibility flexibility; management within the presentation specifications; and modifies the hypertext multiple markup language pages; share information by inserting relevant CSS files into a separate CSS, and scale quality back, repeating the content within the structure.

JavaScript (Front – End): JavaScript is a prototype based scripting language, dynamic, weekly typewritten, general purposed programming language and has major function. And also multi-paradigm language, supporting object-oriented and imperative and programming designs. JavaScript is formalised in ECMA and is mostly used in the form of clients. JavaScript, Conduct a part of the web browsers to provide more dynamic and more powerful user interfaces. It provides programmers with bunch access to the object. Use JavaScript in external website applications. Specific browsers, desktop widgets, instance in a PDF document. For validation purposes like text nox validation, email validation, signal validation JavaScript is used in this app. JavaScript is an intelligent web application verification tool.

C. SOFTWARE IMPLEMENTATION:

TYPES OF MACHINE LEARNING MODELS

1. SUPERVISED AND UNSUPERVISED LEARNING

Supervised Learning

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically supervised learning is when we teach or train the machine using data that is well labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labelled data.
Supervised learning is classified into two categories of algorithms:

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” , “disease” or “no disease”.

- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.

Types:-

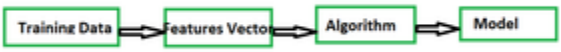
- Regression
- Logistic Regression
- Classification
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

Advantages:-

- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.

Disadvantages:-

- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.



Steps

Unsupervised learning

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.

It allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with unlabelled data.

Unsupervised learning is classified into two categories of algorithms:

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

Types of Unsupervised Learning:-

Clustering

1. Exclusive (partitioning)
2. Agglomerative
3. Overlapping
4. Probabilistic

Clustering Types:-

1. Hierarchical clustering
2. K-means clustering
3. Principal Component Analysis
4. Singular Value Decomposition
5. Independent Component Analysis

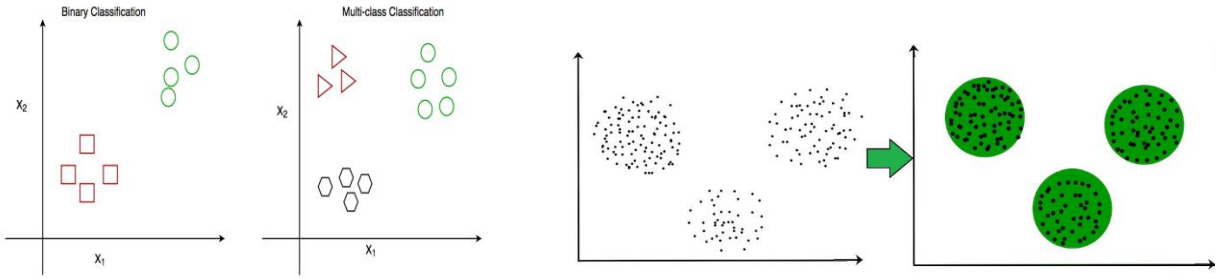
Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data that is not labeled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate

No. of classes	No. of classes is known	No. of classes is not known
Data Analysis	Uses offline analysis	Uses real-time analysis of data
Algorithms used	Linear and Logistics regression, Random forest, Support Vector Machine, Neural Network, etc.	K-Means clustering, Hierarchical clustering, Apriori algorithm, etc.

Classification and clustering

Both Classification and Clustering is used for the categorization of objects into one or more classes based on the features. They appear to be a similar process as the basic difference is minute. In the case of Classification, there are predefined labels assigned to each input instance according to their properties whereas in clustering those labels are missing.



Comparison between Classification and Clustering:

Parameter	CLASSIFICATION	CLUSTERING
Type	used for supervised learning	used for unsupervised learning
Basic	process of classifying the input instances based on their corresponding class labels	grouping the instances based on their similarity without the help of class labels

Parameter	CLASSIFICATION	CLUSTERING
Need	it has labels so there is need of training and testing dataset for verifying the model created	there is no need of training and testing dataset
Complexity	more complex as compared to clustering	less complex as compared to classification
Example Algorithms	Logistic regression, Naive Bayes classifier, Support vector machines, etc.	k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm, etc.

Differences between Classification and Clustering

1. Classification is used for supervised learning whereas clustering is used for unsupervised learning.
2. The process of classifying the input instances based on their corresponding class labels is known as classification whereas grouping the instances based on their similarity without the help of class labels is known as clustering.
3. As Classification have labels so there is need of training and testing dataset for verifying the model created but there is no need for training and testing dataset in clustering.
4. Classification is more complex as compared to clustering as there are many levels in the classification phase whereas only grouping is done in clustering.
5. Classification examples are Logistic regression, Naive Bayes classifier, Support vector machines, etc. Whereas clustering examples are k-means clustering algorithm, Fuzzy c-means clustering algorithm, Gaussian (EM) clustering algorithm, etc.

Types of Classifiers in Machine Learning

1. Decision Tree Classifier:

- Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It

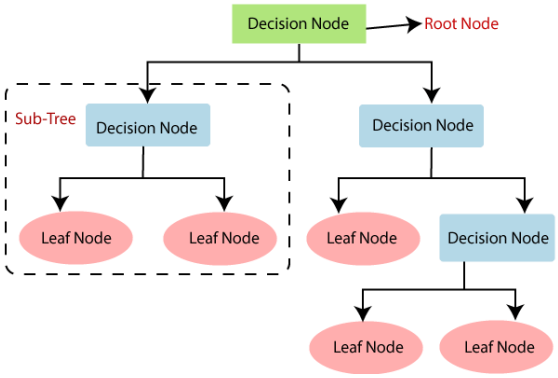
is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- ***It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.***
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the **CART algorithm,** which stands for **Classification and Regression Tree algorithm.**
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Why use Decision Trees?

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree:

- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.



Decision Tree Terminologies

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

How does the Decision Tree algorithm Work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

1. Information Gain:

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.
- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

1. Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

2. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.

- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

Gini Index= $1- \sum_j P_j^2$

Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of overfitting, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Advantages of the Decision Tree

- It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
- It can be very useful for solving decision-related problems.
- It helps to think about all the possible outcomes for a problem.
- There is less requirement of data cleaning compared to other algorithms.

Disadvantages of the Decision Tree

- The decision tree contains lots of layers, which makes it complex.
- It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
- For more class labels, the computational complexity of the decision tree may increase.

Python Implementation of Decision Tree

Steps will also remain the same, which are given below:

- **Data Pre-processing step**
- **Fitting a Decision-Tree algorithm to the Training set**

- **Predicting the test result**
- **Test accuracy of the result(Creation of Confusion matrix)**
- **Visualizing the test set result.**

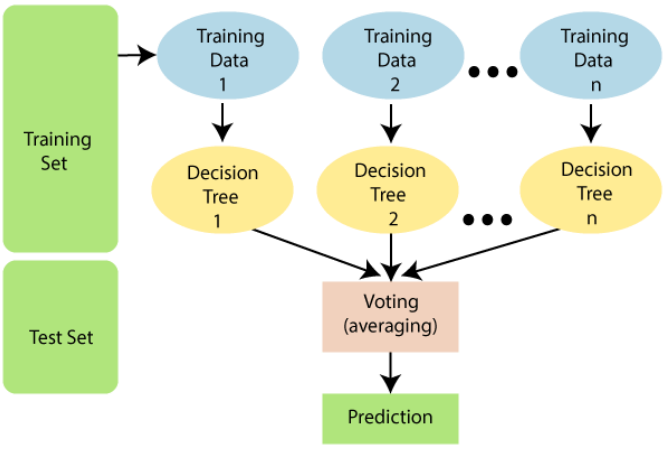
2. RANDOM FOREST CLASSIFIER

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.

- The predictions from each tree must have very low correlations.

Why use Random Forest?

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Applications of Random Forest

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

- Random Forest is capable of performing both Classification and Regression tasks.

- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

Python Implementation of Random Forest Algorithm

Implementation Steps are given below:

- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.

3. KNN CLASSIFIER

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

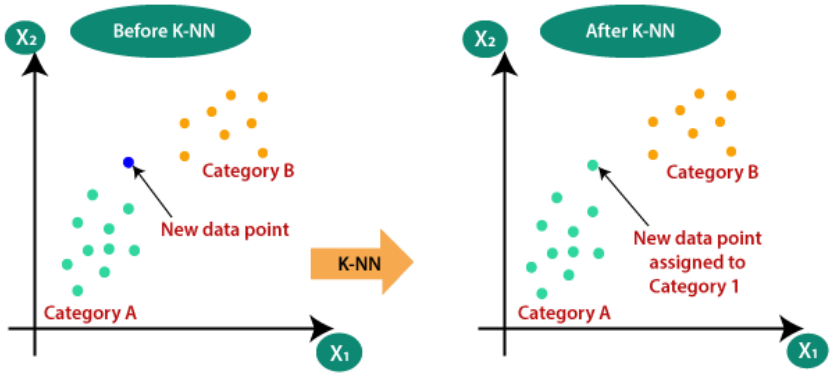
K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Python implementation of the KNN algorithm

Steps to implement the K-NN algorithm:

- Data Pre-processing step
- Fitting the K-NN algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.

4. XG-BOOST CLASSIFIER

Gradient boosted decision trees are implemented by the XGBoost library of Python, intended for speed and execution, which is the most important aspect of ML (machine learning).

XgBoost: XgBoost (Extreme Gradient Boosting) library of Python was introduced at the University of Washington by scholars. It is a module of Python written in C++, which helps ML model algorithms by the training for Gradient Boosting.

Gradient boosting: This is an AI method utilized in classification and regression assignments, among others. It gives an expectation model as a troupe of feeble forecast models, commonly called decision trees.

How does Fundamental Gradient Boosting function?

- A loss function should be improved, which implies bringing down the loss function better than the result.
- To make expectations, weak learners are used in the model

- Decision trees are utilized in this, and they are utilized in a jealous way, which alludes to picking the best-divided focuses in light of Gini Impurity and so forth or to limit the loss function
- The additive model is utilized to gather every one of the frail models, limiting the loss function.
- Trees are added each, ensuring existing trees are not changed in the decision tree. Regularly angle plummet process is utilized to find the best hyper boundaries, post which loads are refreshed further.

In this tutorial, you will find how to introduce and build your most memorable Python XGBoost model.

XGBoost can give improved arrangements than other ML model algorithms. As a matter of fact, since its initiation, it has turned into the "best in class" ML model algorithm to manage organized information.

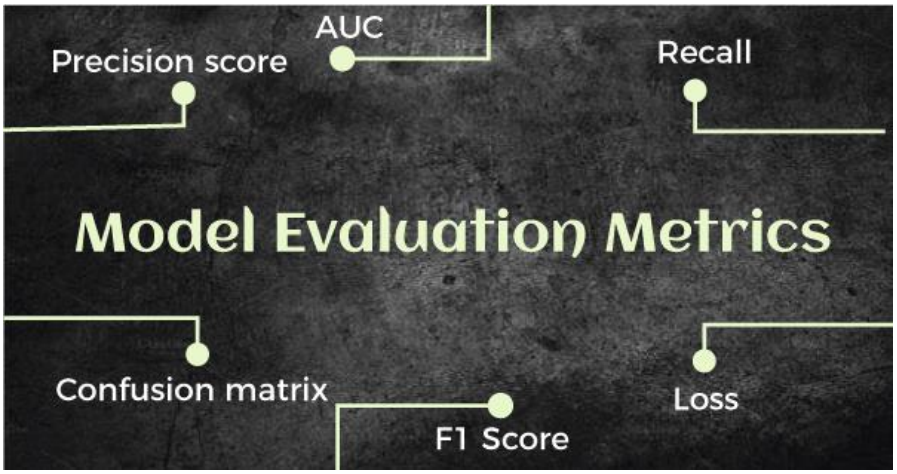
What Makes XGBoost So Famous?

- **Execution and Speed:** Originally built on C++, it is similarly fast to other gathering classifiers.
- **Center calculation is parallelizable**: it can outfit the force of multi-center PCs because the center XGBoost calculation is parallelizable. Moreover, it is parallelizable onto GPUs and across organizations of PCs, making it attainable to prepare on a huge dataset.
- **Reliably outflanks other technique calculations:** It has shown better output on many AI benchmark datasets.
- **Wide assortment of tuning boundaries**: XGBoost inside has boundaries for scikit-learn viable API, missing qualities, regularization, cross-approval, client characterized objective capacities, tree boundaries, etc.

EVALUATION METICS IN MACHINE LEARNING

Evaluating the performance of a Machine learning model is one of the important steps while building an effective ML model. ***To evaluate the performance or quality of the model, different metrics are used, and these metrics are known as performance metrics or evaluation metrics.*** These performance metrics help us understand how well our model has performed for the given data. In this way, we can improve the model's performance by tuning the hyper-parameters.

Each ML model aims to generalize well on unseen/new data, and performance metrics help determine how well the model generalizes on the new dataset.



In machine learning, each task or problem is divided into **classification** and **Regression**. Not all metrics can be used for all types of problems; hence, it is important to know and understand which metrics should be used. Different evaluation metrics are used for both Regression and Classification tasks. In this topic, we will discuss metrics used for classification and regression tasks.

1. Performance Metrics for Classification

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as *Yes or No, 0 or 1, Spam or Not Spam*, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- **Accuracy**
- **Confusion Matrix**
- **Precision**
- **Recall**
- **F-Score**
- **AUC(Area Under the Curve)-ROC**

I. Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

It can be formulated as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ predictions}$$

To implement an accuracy metric, we can compare ground truth and predicted values in a loop, or we can also use the scikit-learn module for this.

Firstly, we need to import the *accuracy_score* function of the scikit-learn library as follows:

```
1. from sklearn.metrics import accuracy_score
2.
3. Here, metrics is a class of sklearn.
4.
5. Then we need to pass the ground truth and predicted values in the function to calculate th
   e accuracy.
6.
7. print(f'Accuracy Score is {accuracy_score(y_test,y_hat)}')
```

Although it is simple to use and implement, it is suitable only for cases where an equal number of samples belong to each class.

When to Use Accuracy?

It is good to use the Accuracy metric when the target variable classes in data are approximately balanced. For example, if 60% of classes in a fruit image dataset are of Apple, 40% are Mango. In this case, if the model is asked to predict whether the image is of Apple or Mango, it will give a prediction with 97% of accuracy.

When not to use Accuracy?

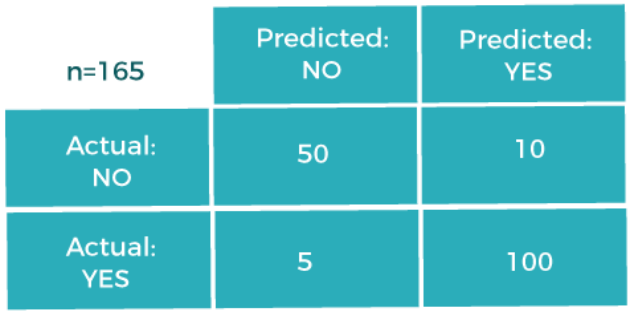
It is recommended not to use the Accuracy measure when the target variable majorly belongs to one class. For example, Suppose there is a model for a disease prediction in which, out of 100 people, only five people have a disease, and 95 people don't have one. In this case, if our model predicts every person with no disease (which means a bad prediction), the Accuracy measure will be 95%, which is not correct.

II. Confusion Matrix

A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of test data when true values are known.

The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.

A typical confusion matrix for a binary classifier looks like the below image(However, it can be extended to use for classifiers with more than two classes).



n=165	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

We can determine the following from the above matrix:

- In the matrix, columns are for the prediction values, and rows specify the Actual values. Here Actual and prediction give two possible classes, Yes or No. So, if we are predicting the presence of a disease in a patient, the Prediction column with Yes means, Patient has the disease, and for NO, the Patient doesn't have the disease.
- In this example, the total number of predictions are 165, out of which 110 time predicted yes, whereas 55 times predicted No.
- However, in reality, 60 cases in which patients don't have the disease, whereas 105 cases in which patients have the disease.

In general, the table is divided into four terminologies, which are as follows:

1. **True Positive(TP):** In this case, the prediction outcome is true, and it is true in reality, also.
2. True Negative(TN): in this case, the prediction outcome is false, and it is false in reality, also.
3. False Positive(FP): In this case, prediction outcomes are true, but they are false in actuality.
4. False Negative(FN): In this case, predictions are false, and they are true in actuality.

III. Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive

or predictions that are actually true to the total positive predictions (True Positive and False Positive).

$$Precision = \frac{TP}{(TP + FP)}$$

IV. Recall or Sensitivity

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as True Positive or predictions that are actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative).

The formula for calculating Recall is given below:

$$\text{Recall} = \frac{TP}{TP+FN}$$

When to use Precision and Recall?

From the above definitions of Precision and Recall, we can say that recall determines the performance of a classifier with respect to a false negative, whereas precision gives information about the performance of a classifier with respect to a false positive.

AD

So, if we want to minimize the false negative, then, Recall should be as near to 100%, and if we want to minimize the false positive, then precision should be close to 100% as possible.

In simple words, *if we maximize precision, it will minimize the FP errors, and if we maximize recall, it will minimize the FN error.*

V. F-Scores

F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, ***the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them.***

The formula for calculating the F1 score is given below:

$$F1 - score = 2 * \frac{precision * recall}{precision + recall}$$

When to use F-Score?

As F-score make use of both precision and recall, so it should be used if both of them are important for evaluation, but one (precision or recall) is slightly more important to consider than the other. For example, when False negatives are comparatively more important than false positives, or vice versa.

VI. AUC-ROC

Sometimes we need to visualize the performance of the classification model on charts; then, we can use the AUC-ROC curve. It is one of the popular and important metrics for evaluating the performance of the classification model.

Firstly, let's understand ROC (Receiver Operating Characteristic curve) curve. ***ROC represents a graph to show the performance of a classification model at different threshold levels***. The curve is plotted between two parameters, which are:

- **True Positive Rate**
- **False Positive Rate**

TPR or true Positive rate is a synonym for Recall, hence can be calculated as:

$$TPR = \frac{TP}{TP + FN}$$

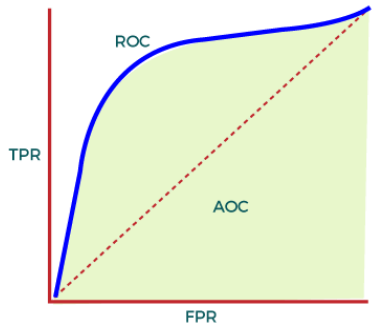
FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

To calculate value at any point in a ROC curve, we can evaluate a logistic regression model multiple times with different classification thresholds, but this would not be much efficient. So, for this, one efficient method is used, which is known as AUC.

AUC: Area Under the ROC curve

AUC is known for **Area Under the ROC curve**. As its name suggests, AUC calculates the two-dimensional area under the entire ROC curve, as shown below image:



AUC calculates the performance across all the thresholds and provides an aggregate measure. The value of AUC ranges from 0 to 1. It means a model with 100% wrong prediction will have an AUC of 0.0, whereas models with 100% correct predictions will have an AUC of 1.0.

When to Use AUC

AUC should be used to measure how well the predictions are ranked rather than their absolute values. Moreover, it measures the quality of predictions of the model without considering the classification threshold.

When not to use AUC

As AUC is scale-invariant, which is not always desirable, and we need calibrating probability outputs, then AUC is not preferable.

Further, AUC is not a useful metric when there are wide disparities in the cost of false negatives vs. false positives, and it is difficult to minimize one type of classification error.

2. Performance Metrics for Regression

Regression is a supervised learning technique that aims to find the relationships between the dependent and independent variables. A predictive regression model predicts a numeric or discrete value. The metrics used for regression are different from the classification metrics. It means we cannot use the Accuracy metric (explained above) to evaluate a regression model; instead, the performance of a Regression model is reported as errors in the prediction. Following are the popular metrics that are used to evaluate the performance of Regression models.

- **Mean Absolute Error**
- **Mean Squared Error**
- **R2 Score**
- **Adjusted R2**

I. Mean Absolute Error (MAE)

Mean Absolute Error or MAE is one of the simplest metrics, which measures the absolute difference between actual and predicted values, where absolute means taking a number as Positive.

To understand MAE, let's take an example of Linear Regression, where the model draws a best fit line between dependent and independent variables. To measure the MAE or error in prediction, we need to calculate the difference between actual values and predicted values. But in order to find the absolute error for the complete dataset, we need to find the mean absolute of the complete dataset.

The below formula is used to calculate MAE:

$$MAE = 1/N \sum |Y - Y'|$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

MAE is much more robust for the outliers. One of the limitations of MAE is that it is not differentiable, so for this, we need to apply different optimizers such as Gradient Descent. However, to overcome this limitation, another metric can be used, which is Mean Squared Error or MSE.

II. Mean Squared Error

Mean Squared error or MSE is one of the most suitable metrics for Regression evaluation. It measures the average of the Squared difference between predicted values and the actual value given by the model.

Since in MSE, errors are squared, therefore it only assumes non-negative values, and it is usually positive and non-zero.

Moreover, due to squared differences, it penalizes small errors also, and hence it leads to over-estimation of how bad the model is.

MSE is a much-preferred metric compared to other regression metrics as it is differentiable and hence optimized better.

The formula for calculating MSE is given below:

$$MSE = 1/N \sum (Y - Y')^2$$

Here,

Y is the Actual outcome, Y' is the predicted outcome, and N is the total number of data points.

III. R Squared Score

R squared error is also known as Coefficient of Determination, which is another popular metric used for Regression model evaluation. The R-squared metric enables us to compare our model with a constant baseline to determine the performance of the model. To select the constant baseline, we need to take the mean of the data and draw the line at the mean.

The R squared score will always be less than or equal to 1 without concerning if the values are too large or small.

$$R^2 = 1 - \frac{MSE(Model)}{MSE(Baseline)}$$

IV. Adjusted R Squared

Adjusted R squared, as the name suggests, is the improved version of R squared error. R square has a limitation of improvement of a score on increasing the terms, even though the model is not improving, and it may mislead the data scientists.

To overcome the issue of R square, adjusted R squared is used, which will always show a lower value than R^2. It is because it adjusts the values of increasing predictors and only shows improvement if there is a real improvement.

We can calculate the adjusted R squared as follows:

$$R_a^2 = 1-\left[\left(\frac{n-1}{n-k-1}\right) \times \left(1 - R^2\right)\right]$$

Here,

n is the number of observations

k denotes the number of independent variables

and R_a^2 denotes the adjusted R^2

NECESSARY LIBRARIES

1. Numpy
2. Pandas
3. Scikit-learn
4. Matplotlib
5. Scipy
6. Seaborn
7. Flask
8. Pickle-mixin

Conclusion

Lung cancer is one of the challenging problems in medical field due to structure of cancer cells. Therefore, the proper medication has to be given to the patient for increasing the survival chances of the patient. Once the cancer is detected, the Thoracic Surgery is one of the best treatment options for the diagnosis of Lung Cancer. The project involves the analysis of the patient's dataset who underwent Thoracic Surgery and an attempt is made to model a classifier that will predict the survival of the patient post the surgery. The dataset will be trained using four Supervised The Algorithms of Machine Learning that are Decision Tree, Random Forest, KNN, Xgboost. Among the four algorithms used, its observed that all the algorithms gives the highest accuracy of 91% compared to the other algorithms data in the future to analyse the system.

FUTURE SCOPE

Ultimately, we were able to improve our results by averaging a series of identical and independently distributed trees, which we would like to contrast with boosting, in which the trees would be grown in an adaptive manner specific to the bias (not I.I.D.). We would like to recursively train on the residuals of each misclassification. A next possible step would be to implement the following algorithm (bumping):

1. Bootstrap n models (with replacement, forcing even ratios), where number of models = number of features.

2. Train n models, with initially one feature per model.

3. Test all n models on original data set. Pick the model with lowest error on original data set, and define a new residual data set on all misclassified examples.

4.Train your next n models on the residual (i.e. boosting) - but NO averaging at this point.

5. Test on the very original data set and pick the best one. Continue process repeatedly.

Hopefully this will further reduce our variance. In addition, we calculated the optimal feature set as shown above, so it would be interesting to compare different results for all of our implementations if we use only those specific features

BIBLOGRAPHY

[Wroclaw University Study] Creators: Marek Lubicz (1), Konrad Pawelczyk (2), Adam Rzechonek (2), Jerzy Kolodziej (2)

– (1) Wroclaw University of Technology, wybrzeze Wyspianskiego 27, 50-370, Wroclaw, Poland.
– (2) Wroclaw Medical University, wybrzeze L. Pasteura 1, 50-367 Wroclaw, Poland.

Boosted SVM for extracting rules from imbalanced data in application to prediction of the post-operative life expectancy in the lung cancer patients. Applied Soft Computing.

PROJECT

APPENDIX

- ### Importing libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix
import itertools
```

- ### Read the data set

```
df=pd.read_csv(r'C:\Users\JANARDHAN\Downloads\ThoracicSurgery.csv')


df.head()
```

	Diagnosis	FVC	FEV1	Performance	Pain	Haemoptysis	Dyspnoea	Cough	Weakness	Tumor_Size	Diabetes_Mellitus	MI_6mo	PAD	Smoking	Asthma	Ag
0	2	2.88	2.16	1	0	0	0	1	1	4	0	0	0	1	0	6
1	3	3.40	1.88	0	0	0	0	0	0	2	0	0	0	1	0	5
2	3	2.76	2.08	1	0	0	0	1	0	1	0	0	0	1	0	5
3	3	3.68	3.04	0	0	0	0	0	0	1	0	0	0	0	0	5
4	3	2.44	0.96	2	0	1	0	1	1	1	0	0	0	1	0	7

df.describe()

	Diagnosis	FVC	FEV1	Performance	Pain	Haemoptysis	Dyspnoea	Cough	Weakness	Tumor_Size	Diabetes_Mellitus	MI_6m
count	454.000000	454.000000	454.00000	454.000000	454.000000	454.000000	454.000000	454.000000	454.000000	454.000000	454.000000	454.0000
mean	3.092511	3.287952	2.51685	0.795154	0.059471	0.136564	0.055066	0.696035	0.171806	1.733480	0.074890	0.0044
std	0.715817	0.872347	0.77189	0.531459	0.236766	0.343765	0.228361	0.460475	0.377628	0.707499	0.263504	0.0662
min	1.000000	1.440000	0.96000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.0000
25%	3.000000	2.600000	1.96000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.0000
50%	3.000000	3.160000	2.36000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000	0.0000
75%	3.000000	3.840000	2.97750	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000	0.0000
max	8.000000	6.300000	5.48000	2.000000	1.000000	1.000000	1.000000	1.000000	1.000000	4.000000	1.000000	1.0000

- **One year death**

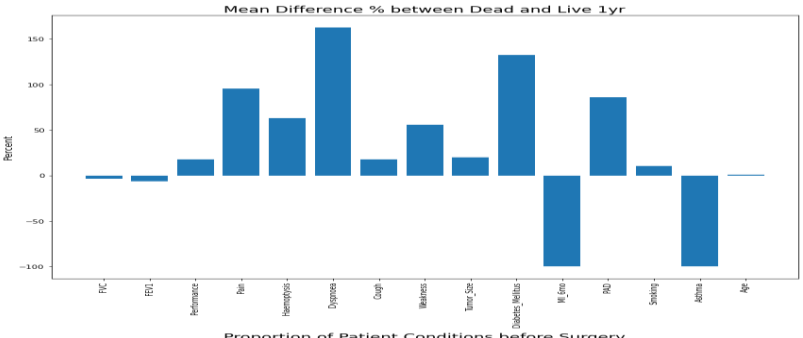
```
live = df[df['Death_1yr'] == 0]
death = df[df['Death_1yr'] == 1]
cond = ['FVC','FEV1','Performance','Pain','Haemoptysis','Dyspnoea','Cough','Weakness',\
    'Tumor_Size','Diabetes_Mellitus','MI_6mo','PAD','Smoking','Asthma','Age']
l = [np.mean(live[c]) for c in cond]
d = [np.mean(death[c]) for c in cond]
ld = pd.DataFrame(data={'Attribute' : cond,'Live 1yr Mean' : l,'Death 1yr Mean' : d})
ld = ld.set_index('Attribute')
print('Death: {:d}, Live: {:d}'.format(len(death), len(live)))
print("1 tear death: {:.2f}% out of 454 patients".format(np.mean(df.Death_1yr)*100))
ld
```

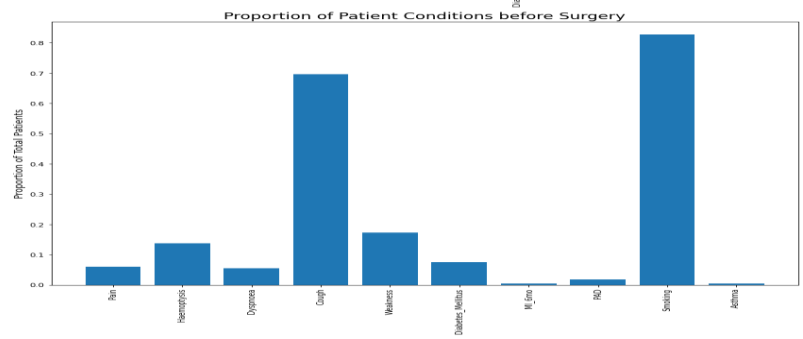
```
Death: 69, Live: 385
1 tear death: 15.20% out of 454 patients
```

Attribute	Live 1yr Mean	Death 1yr Mean
FVC	3.304597	3.195072
FEV1	2.540805	2.383188
Performance	0.774026	0.913043
Pain	0.051948	0.101449
Haemoptysis	0.124675	0.202899
Dyspnoea	0.044156	0.115942
Cough	0.677922	0.797101
Weakness	0.158442	0.246377
Tumor_Size	1.683117	2.014493
Diabetes_Mellitus	0.062338	0.144928
MI_6mo	0.005195	0.000000
PAD	0.015584	0.028986
Smoking	0.815584	0.898551
Asthma	0.005195	0.000000
Age	62.677922	63.333333

- **Mean difference % between Dead and Live 1 year, Proportion of patients before surgery**

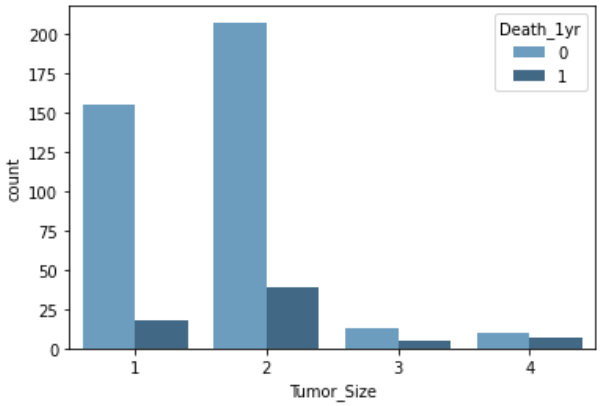
```
d= np.array(d)
l = np.array(l)
p_diff =(d-l)/l*100
fig, axes = plt.subplots(2,1,figsize=(12,18))
axes[0].bar(cond,p_diff)
axes[0].set_title('Mean Difference % between Dead and Live 1yr', fontsize=18)
axes[0].set_xticks(cond)
axes[0].set_xticklabels(cond, rotation=90)
axes[0].set_ylabel('Percent', fontsize=13)
tf_col =
['Pain','Haemoptysis','Dyspnoea','Cough','Weakness','Diabetes_Mellitus','MI_6mo','PAD','
Smoking','Asthma']
tf_sum = [df[col].sum()/454 for col in tf_col]
axes[1].bar(tf_col, tf_sum)
axes[1].set_xticks(tf_col)
axes[1].set_xticklabels(tf_col, rotation = 90)
axes[1].set_ylabel('Proportion of Total Patients', fontsize=13)
axes[1].set_title('Proportion of Patient Conditions before Surgery', fontsize =18)
plt.tight_layout()
```

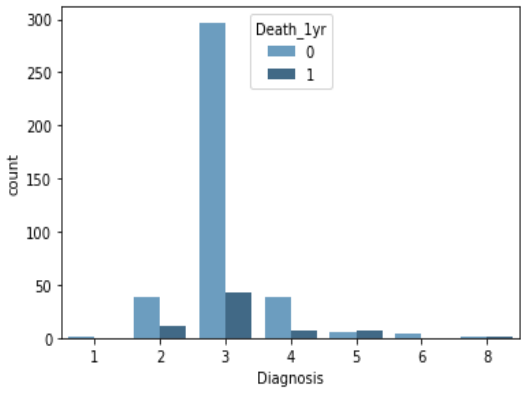


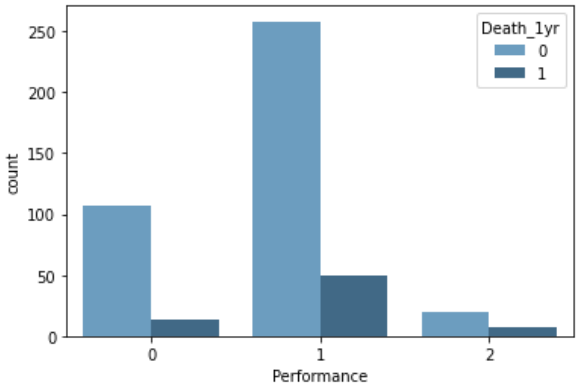


- **Import seaborn**

```
import seaborn as sns
sns.countplot(x='Diagnosis',hue='Death_1yr',data=df,palette='Blues_d')
plt.show()
sns.countplot(x='Tumor_Size',hue='Death_1yr',data=df,palette='Blues_d')
plt.show()
sns.countplot(x='Performance',hue='Death_1yr',data=df,palette='Blues_d')
plt.show()
```







df.shape

(454, 17)

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 454 entries, 0 to 453
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Diagnosis          454 non-null    int64
 1   FVC                454 non-null    float64
 2   FEV1               454 non-null    float64
 3   Performance        454 non-null    int64
 4   Pain               454 non-null    int64
 5   Haemoptysis        454 non-null    int64
 6   Dyspnoea           454 non-null    int64
 7   Cough              454 non-null    int64
 8   Weakness           454 non-null    int64
 9   Tumor_Size         454 non-null    int64
 10  Diabetes_Mellitus  454 non-null    int64
 11  MI_6mo             454 non-null    int64
 12  PAD                454 non-null    int64
 13  Smoking            454 non-null    int64
 14  Asthma             454 non-null    int64
 15  Age                454 non-null    int64
 16  Death_1yr          454 non-null    int64
dtypes: float64(2), int64(15)
memory usage: 60.4 KB
```

- **Checking for null values**

  df.isnull()

	Diagnosis	FVC	FEV1	Performance	Pain	Haemoptysis	Dyspnoea	Cough	Weakness	Tumor_Size	Diabetes_Mellitus	MI_6mo	PAD	Smoking	Asthma
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
449	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
450	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
451	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
452	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
453	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

454 rows × 17 columns

df.isnull().sum()

```
Diagnosis              0
FVC                    0
FEV1                   0
Performance            0
Pain                   0
Haemoptysis            0
Dyspnoea               0
Cough                  0
Weakness               0
Tumor_Size             0
Diabetes_Mellitus      0
MI_6mo                 0
PAD                    0
Smoking                0
Asthma                 0
Age                    0
Death_1yr              0
dtype: int64
```

df.drop(['FVC'],axis=1, inplace=True)

➢ Getting train shape

```
x=df.iloc[:,0:15].values
y=df.iloc[:,15:16].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print('Shape of x_train {}'.format(x_train.shape))
print('Shape of y_train {}'.format(y_train.shape))
print('Shape of x_train {}'.format(x_test.shape))
print('Shape of y_train {}'.format(y_test.shape))
```

```
Shape of x_train (363, 15)
Shape of y_train (363, 1)
Shape of x_train (91, 15)
Shape of y_train (91, 1)
```

➢ Data Preprocessing

from sklearn.preprocessing import StandardScaler

```
sc = StandardScaler()

x_train = sc.fit_transform(x_train)

x_test =  sc.fit_transform(x_test)

def decisionTree(x_train, x_test, y_train, y_test):

    dt= DecisionTreeClassifier()

    dt.fit(x_train,y_train)

    yPred = dt.predict(x_test)

    print('***DecisionTreeClassifier***')

    print('Confusion matrix')

    print (confusion_matrix(y_test,yPred))

    print('Classification report')

    print(classification_report(y_test, yPred))
```

```
***DecisionTreeClassifier***
Confusion matrix
[[58 16]
 [12  5]]
Classification report
              precision    recall  f1-score   support

           0       0.83      0.78      0.81        74
           1       0.24      0.29      0.26        17

    accuracy                           0.69        91
   macro avg       0.53      0.54      0.53        91
weighted avg       0.72      0.69      0.70        91
```

```
def randomForest(x_train, x_test, y_train, y_test):

    rf = RandomForestClassifier()
```

```
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
randomForest(x_train, x_test, y_train, y_test)
```

```
***RandomForestClassifier***
Confusion matrix
[[71  3]
 [17  0]]
Classification report
              precision    recall  f1-score   support

           0       0.81      0.96      0.88        74
           1       0.00      0.00      0.00        17

    accuracy                           0.78        91
   macro avg       0.40      0.48      0.44        91
weighted avg       0.66      0.78      0.71        91
```

```
def KNN(x_train, x_test, y_train, y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train, y_train)
    yPred = knn.predict(x_test)
    print('KNeighboursClassifier')
    print('Confusion matrix')
```

```
    print(confusion_matrix(y_test,yPred))

    print('Classification report')

    print(classification_report(y_test,yPred))

KNN(x_train, x_test, y_train, y_test)
```

```
KNeighboursClassifier
Confusion matrix
[[72  2]
 [16  1]]
Classification report
              precision    recall  f1-score   support

           0       0.82      0.97      0.89        74
           1       0.33      0.06      0.10        17

    accuracy                           0.80        91
   macro avg       0.58      0.52      0.49        91
weighted avg       0.73      0.80      0.74        91
```

```
def xgboost(x_train, x_test, y_train, y_test):

    xg = GradientBoostingClassifier()

    xg.fit(x_train,y_train)

    yPred = xg.predict(x_test)

    print('*** GradientBoostingClassifier***')

    print('Confusion matrix')

    print(confusion_matrix(y_test,yPred))

    print('Classifier report')

    print(classification_report(y_test,yPred))

xgboost(x_train, x_test, y_train, y_test)
```

```
*** GradientBoostingClassifier***
Confusion matrix
[[70  4]
 [17  0]]
Classifier report
              precision    recall  f1-score   support

           0       0.80      0.95      0.87        74
           1       0.00      0.00      0.00        17

    accuracy                           0.77        91
   macro avg       0.40      0.47      0.43        91
weighted avg       0.65      0.77      0.71        91
```

```
def compareModel(x_train, x_test, y_train, y_test):
    decisionTree(x_train, x_test, y_train, y_test)
    print('-'*100)
    randomForest(x_train, x_test, y_train, y_test)
    print('-'*100)
    KNN(x_train, x_test, y_train, y_test)
    print('-'*100)
    xgboost(x_train, x_test, y_train, y_test)
    print('-'*100)
compareModel(x_train, x_test, y_train, y_test)
```

```
***RandomForestClassifier***
Confusion matrix
[[73  1]
 [17  0]]
classification report
              precision    recall  f1-score   support

           0       0.81      0.99      0.89        74
           1       0.00      0.00      0.00        17

    accuracy                           0.80        91
   macro avg       0.41      0.49      0.45        91
weighted avg       0.66      0.80      0.72        91

----------------------------------------------------------------------------------------------------
KNeighboursClassifier
Confusion matrix
[[72  2]
 [16  1]]
classification report
              precision    recall  f1-score   support

           0       0.82      0.97      0.89        74
           1       0.33      0.06      0.10        17

    accuracy                           0.80        91
   macro avg       0.58      0.52      0.49        91
weighted avg       0.73      0.80      0.74        91
```

```
KNeighboursClassifier
Confusion matrix
[[72  2]
 [16  1]]
classification report
              precision    recall  f1-score   support

           0       0.82      0.97      0.89        74
           1       0.33      0.06      0.10        17

    accuracy                           0.80        91
   macro avg       0.58      0.52      0.49        91
weighted avg       0.73      0.80      0.74        91

----------------------------------------------------------------------------------------------------
*** GradientBoostingClassifier***
Confusion matrix
[[70  4]
 [17  0]]
Classifier report
              precision    recall  f1-score   support

           0       0.80      0.95      0.87        74
           1       0.00      0.00      0.00        17

    accuracy                           0.77        91
   macro avg       0.40      0.47      0.43        91
weighted avg       0.65      0.77      0.71        91
```

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

```
rf = RandomForestClassifier()

rf.fit(x_train,y_train)

yPred = rf.predict(x_test)

f1_score(yPred,y_test,average='weighted')
```

0.8640194161666553

```
cv = cross_val_score(rf,x,y,cv=5)

np.mean(cv)
```

0.8392185592185593

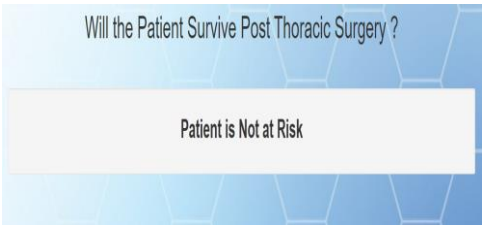
```
import pickle
pickle.dump(rf,open('model.pkl','wb'))
```

- **FLASK CODE**

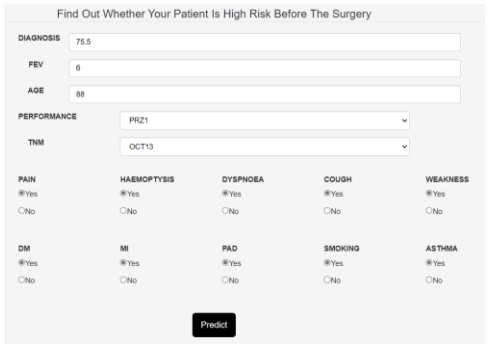
```
import numpy as np
import pandas as pd
import os
import joblib
from flask import Flask, jsonify, request, render_template, url_for, redirect, Markup

app = Flask(__name__)
joblib_file = "model.pkl"
model = joblib.load(joblib_file)


@app.route('/')
def index():
    return render_template('index.html')

@app.route("/form", methods=['GET','POST'])
def getform():
    if request.method == "GET":
        return (render_template("form.html"))

    if request.method == 'POST':
        if 'submit-button' in request.form:
            diagnosis = request.form["diagnosis"]
```

```
        fev = request.form["fev"]
        age = request.form["age"]
        performance = request.form["performance"]
        tnm = request.form["tnm"]
        hae = request.form['hae']
        pain = request.form["pain"]
        dys = request.form["dys"]
        cough = request.form["cough"]
        weakness = request.form["weakness"]
        dm = request.form["dm"]
        mi = request.form["mi"]
        pad = request.form["pad"]
        smoking = request.form["smoking"]
        asthma = request.form["asthma"]
        total = [[diagnosis,fev,age,performance,tnm,hae,pain,dys,cough,weakness,dm,mi,pad,
smoking,asthma]]
        #prediction = model.predict(total)

        #input_variables = pd.DataFrame([[performance, dys, cough, tnm, dm]], columns=['P
erformance', 'Dyspnoea', 'Cough', 'TNM', 'DM'], dtype=float)

        prediction = model.predict(total)[0]

        if int(prediction) == 1:
            prediction = "Patient is at High Risk"


        else:
            prediction = "Patient is Not at Risk"


        return render_template("result.html", prediction = prediction)

    return render_template("result.html")

if __name__=="__main__":
    app.run(debug=False)
```

Input 1 **Output1**



Input 2 **Output 2**



- **IBM DEPLOYMENT**

FLASK CODE

```
import requests
import json
# NOTE: you must manually set API_KEY below using information retrieved from your IB
M Cloud account.
API_KEY = "G-hCBLM-IG6t46oFLPdg5SEh6oXkmn5NUoK_E8nTD3dD"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"field": [["diagnosis","fev","age","performance","tnm","
hae","pain","dys","cough","weakness","dm","mi","pad","smoking","asthma"]],
                                   "values": [[1.22459508,  1,  1, 1, 1,
      0.24147264,  0.64650503, 0.44942937,  1, 1.29466311,
      0.07443229, 1.15011733,  0, 1.05255883,  1]]}]}
```

```
response_scoring = requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/8eda797d-2bfb-454d-a287-de8e8f68de56/predictions?version=2022-08-17', json=payload_scoring,
 headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
predictions = response_scoring.json()
pred = predictions['predictions'][0]['values'][0][0]
if(pred == 0):
    print("Patient is at high risk")
else:
    print("Patient is not at risk")
```

Input

```
ayload_scoring = {"input_data": [{"field": [["diagnosis","fev","age","performance","tnm","hae","pain","dys",
                                 "values": [[1.22459508,  0.00717926,  0.37059679, -0.25400025, -0.40430377,
     -0.24147264,  0.64650503, -0.44942937,  0.38985389, -0.29466311,
     -0.07443229, -0.15011733,  0.46703405, -0.05255883,  1.43058247]]}]}
```

Output

```
In [1]: runfile('C:/Users/JANARDHAN/Desktop/project2/flask/
ibmflask.py', wdir='C:/Users/JANARDHAN/Desktop/project2/flask')
Scoring response
Patient is at high risk
```