Annexure-I

Project On Machine Learning

Breast Cancer Detection

A training report

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

B.Tech in Computer Science And Engineering

(CSE Honours)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 06/01/21 to 07/01/21

SUBMITTED BY

Gundu Lalithendranath

11904466

gundu lalithendranath

TABLE OF CONTENTS

Annexure-II: Student Declaration

To whom so ever it may concern

I, **Gundu Lalithendranath, 11904466,** hereby declare that the work done by me on "**Breast Cancer Detection Using Machine Learning**" from **June, 2021** to **July, 2021**, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **B.Tech in Computer Science And Engineering.**

Gundu Lalithendranath(11904466)

Signature

Date:9th July, 2021.


CERTIFICATE OF COMPLETION
THIS CERTIFICATE IS AWARDED TO
Gundu Lalithendranath
for successfully completing
Machine Learning Course
09th Jul, 2021
ISSUED DATE
CEO, Board Infinity
Sumesh Nair
BI31ML35479040
CERTIFICATE NO.
BOARD

INTRODUCTION OF THE PROJECT UNDERTAKEN

- **Objectives of the work undertaken**

  The main Objective of the project work that I have undertaken is to predict whether the patient has breast cancer or not. Malignant means the person is affected with breast cancer Benign means the person is not affected with cancer. To predict in form of binary in my project 1 means Malignant. 0 means Benign.

- **Scope of the Project work**

  The scope of the work is as long as human life exists on earth. As it is useful to caution the person about their disease and help them to take precaution and do the needful to get cure as early as possible.

- **Importance and Applicability**

  It is important because many females are suffering for this problem because of their no knowledge on these applications, they are unable to decide whether they are affected or not. And also if affected they can know and get precautions. It is applicable to all females with having inputs that are required to determine whether they are affected or not.

- **Role and Profile**

  My role here is I have collected a dataset from Kaggle and made a detailed analysis of how to predict whether a person is suffering from breast cancer or not. And using Machine Learning model I have feed the data to Logistic Regression model and train it to make predicts for new data.

INTRODUCTION OF THE COMPANY BOARD INFINITY

- **BI's Vision and Mission**

  BI is a platform where you can learn the courses with experts and get your dream job.
  BI's vision and mission is to personalize your career journey, help you realize true potential and meet your career dreams.

- **Origin and Growth**

  The platform was launched in year 2017 by the Chief Executing Officer and Co-founder **Sumesh Nair.** Since then, the company has seen massive growth of nearly 400% in the recent pandemic times by providing massive courser at a reasonable rate and help aspirants to achieve their dream careers and dream jobs.

- **Various departments and their functions**

  Call centre department where they give info and insights about the courses and discounts and such things.
  Help centre department where they connect to social media platforms and provide help regarding details and such thing.
  Coaches who teach each and everything to the students of that registered course.
  Mentors and Moderators, they help in resolving queries of students regarding subject and help students in connecting with coaches.

Chapter-1 Introduction to Machine Learning

➢ Machine Learning

Machine Learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and predictive maintenance.

➢ Importance Of Machine Learning

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

Classical machine learning is often categorized by how an algorithm learns to become more accurate in its predictions. There are four basic approaches:
supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning. The type of algorithm data scientists chooses to use depends on what type of data they want to predict.

- **Supervised learning:** In this type of machine learning, data scientists supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified.

- **Unsupervised learning:** This type of machine learning involves algorithms that train on unlabeled data. The algorithm scans through data sets looking for any meaningful connection. The data that algorithms train on as well as the predictions or recommendations they output are predetermined.

- **Semi-supervised learning:** This approach to machine learning involves a mix of the two preceding types. Data scientists may feed an algorithm mostly labeled training data, but the model is free to explore the data on its own and develop its own understanding of the data set.

- **Reinforcement learning:** Data scientists typically use reinforcement learning to teach a machine to complete a multi-step process for which there are clearly defined rules. Data scientists program an algorithm to complete a task and give it positive or negative cues as it works out how to complete a task. But for the most part, the algorithm decides on its own what steps to take along the way.

➢ Working of Supervised Learning

Supervised machine learning requires the data scientist to train the algorithm with both labelled inputs and desired outputs. Supervised learning algorithms are good for the following tasks:

- **Binary classification:** Dividing data into two categories.
- **Multi-class classification:** Choosing between more than two types of answers.
- **Regression modelling:** Predicting continuous values.
- **Ensembling:** Combining the predictions of multiple machine learning models to produce an accurate prediction.

➢ Working of Unsupervised Leaning

Unsupervised machine learning algorithms do not require data to be labelled. They sift through unlabelled data to look for patterns that can be used to group data points into subsets. Most types of deep learning, including neural networks, are unsupervised algorithms. Unsupervised learning algorithms are good for the following tasks:

- **Clustering:** Splitting the dataset into groups based on similarity.
- **Anomaly detection:** Identifying unusual data points in a data set.
- **Association mining:** Identifying sets of items in a data set that frequently occur together.
- **Dimensionality reduction:** Reducing the number of variables in a data set.

➢ Working Of Semi-supervised learning

Semi-supervised learning works by data scientists feeding a small amount of labeled training data to an algorithm. From this, the algorithm learns the dimensions of the data set, which it can then apply to new, unlabeled data. The performance of algorithms typically improves when they train on labeled data sets. But labeling data can be time consuming and expensive. Semi-supervised learning strikes a middle ground between the performance of supervised learning and the efficiency of unsupervised learning. Some areas where semi-supervised learning is used include:

- **Machine translation:** Teaching algorithms to translate language based on less than a full dictionary of words.
- **Fraud detection:** Identifying cases of fraud when you only have a few positive examples.
- **Labelling data:** Algorithms trained on small data sets can learn to apply data labels to larger sets automatically.

➢ Working of Reinforcement Learning

Reinforcement learning works by programming an algorithm with a distinct goal and a prescribed set of rules for accomplishing that goal. Data scientists also program the algorithm to seek positive rewards -- which it receives when it performs an action that is beneficial toward the ultimate goal -- and avoid punishments -- which it receives when it performs an action that gets it farther away from its ultimate goal. Reinforcement learning is often used in areas such as:

- **Robotics:** Robots can learn to perform tasks the physical world using this technique.
- **Video gameplay:** Reinforcement learning has been used to teach bots to play a number of video games.
- **Resource management:** Given finite resources and a defined goal, reinforcement learning can help enterprises plan out how to allocate resources.

➢ Uses of Machine Learning

- **Business intelligence:** BI and analytics vendors use machine learning in their software to identify potentially important data points, patterns of data points and anomalies.

- **Human resource information systems.**: HRIS systems can use machine learning models to filter through applications and identify the best candidates for an open position.

- **Self-driving cars:** Machine learning algorithms can even make it possible for a semi-autonomous car to recognize a partially visible object and alert the driver.

- **Virtual assistants:** Smart assistants typically combine supervised and unsupervised machine learning models to interpret natural speech and supply context.

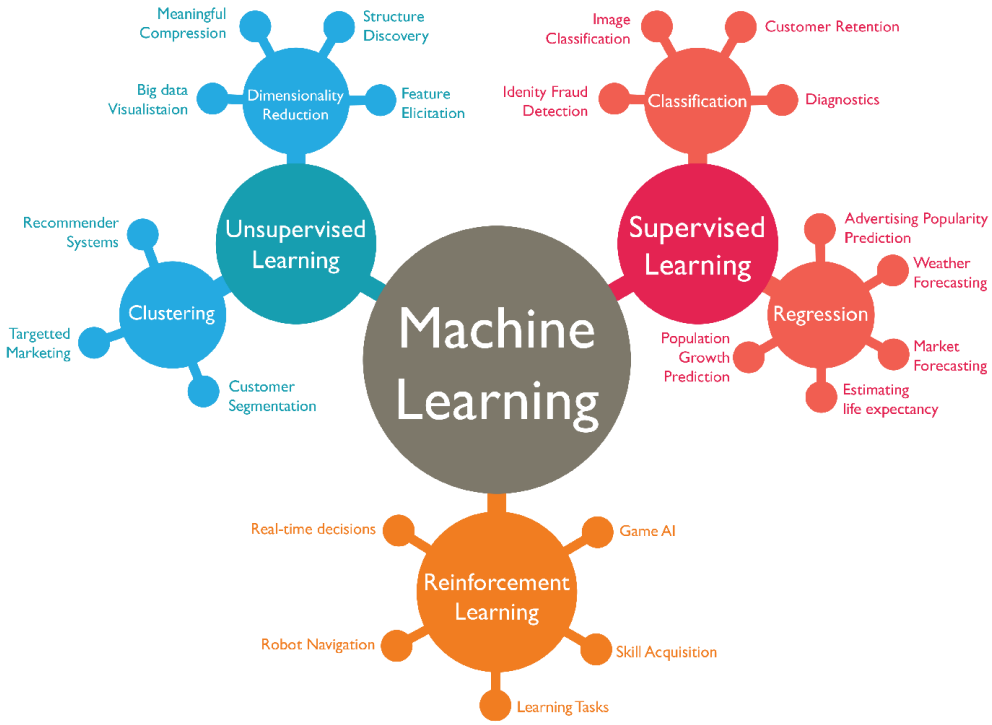

Fig 1.1

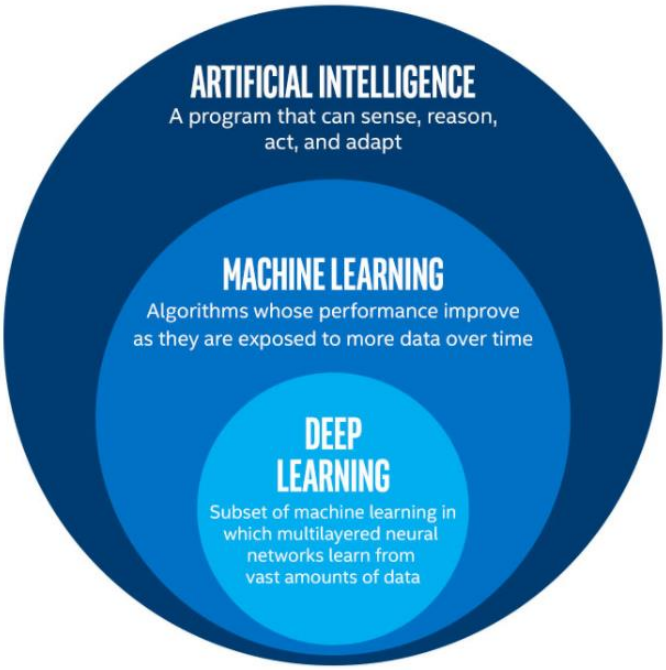

Fig1.2

Chapter-2 Introduction to Breast Cancer

- **Breast Cancer**

Cells in the body normally divide (reproduce) only when new cells are needed. Sometimes, cells in a part of the body grow and divide out of control, which creates a mass of tissue called a tumor. If the cells that are growing out of control are normal cells, the tumor is called benign (not cancerous). If, however, the cells that are growing out of control are abnormal and don't function like the body's normal cells, the tumor is called malignant (cancerous).

Cancers are named after the part of the body from which they originate. Breast cancer originates in the breast tissue. Like other cancers, breast cancer can invade and grow into the tissue surrounding the breast. It can also travel to other parts of the body and form new tumors, a process called metastasis.

- **To Whom it is Affected**

Breast cancer is the most common cancer among women other than skin cancer. Increasing age is the most common risk factor for developing breast cancer, with 66% of breast cancer patients being diagnosed after the age of 55.

In the US, breast cancer is the second-leading cause of cancer death in women after lung cancer, and it's the leading cause of cancer death among women ages 35 to 54. Only 5 to 10% of breast cancers occur in women with a clearly defined genetic predisposition for the disease. The majority of breast cancer cases are "sporadic," meaning there is no definitive gene mutation.

- **Cause of Breast Cancer**

We do not know what causes breast cancer, although we do know that certain risk factors may put you at higher risk of developing it. A woman's age, genetic factors, family history, personal health history, and diet all contribute to breast cancer risk.

- **Risk factors of Breast Cancer**

Like many conditions, risk factors for breast cancer fall into the categories of things you can control and things that you cannot control. Risk factors affect your chances of getting a disease, but having a risk factor does not mean that you are guaranteed to get a certain disease.

Controllable risk factors for breast cancer

- **Alcohol consumption:** The risk of breast cancer increases with the amount of alcohol consumed. For instance, women who consume two or three alcoholic beverages daily have an approximately 20% higher risk of getting breast cancer than women who do not drink at all.
- **Body weight:** Being obese is a risk factor for breast cancer. It is important to eat a healthy diet and exercise regularly.

- **Breast implants:** Having silicone breast implants and resulting scar tissue make it harder to distinguish problems on regular mammograms. It is best to have a few more images (called implant displacement views) to improve the examination. There is also a rare cancer called anaplastic large cell lymphoma (ALCL) that is associated with the implants.
- **Choosing not to breastfeed:** Not breastfeeding can raise the risk.

Non-controllable risk factors for breast cancer

- **Being a woman:** Although men do get breast cancer, it is far more common in women.
- **Breast density:** You are at higher risk of breast cancer if you have dense breasts. It can also make it harder to see tumors during mammograms.
- **Getting older:** Aging is a factor. A majority of new breast cancer diagnoses come after the age of 55.
- **Reproductive factors:** These include getting your period before age 12, entering menopause after age 55, having no children, or having your first child after 30.
- **Exposure to radiation:** This type of exposure could result from having many fluoroscopy X-rays or from being treated with radiation to the chest area.
- **Having a family history of breast cancer, or having genetic mutations:** Family history that includes having a first degree relative (mother, sister, daughter, father, brother, son) with breast cancer poses a higher risk for you. If you have more than one relative on either side of your family with breast cancer, you have a higher risk. In terms of genetic mutations, these include changes to genes like BRCA1 and BRCA2.
- **Having already had breast cancer:** The risk is higher for you if you have already had breast cancer and/or certain types of benign breast conditions such as lobular carcinoma in situ, ductal carcinoma in situ, or atypical hyperplasia.

➢ **Warning Signs of Breast Cancer**

- A lump or thickening in or near the breast or in the underarm that persists through the menstrual cycle.
- A mass or lump, which may feel as small as a pea.
- A change in the size, shape, or contour of the breast.
- A blood-stained or clear fluid discharge from the nipple.
- A change in the look or feel of the skin on the breast or nipple (dimpled, puckered, scaly, or inflamed).
- Redness of the skin on the breast or nipple.

- An area that is distinctly different from any other area on either breast.
- A marble-like hardened area under the skin.

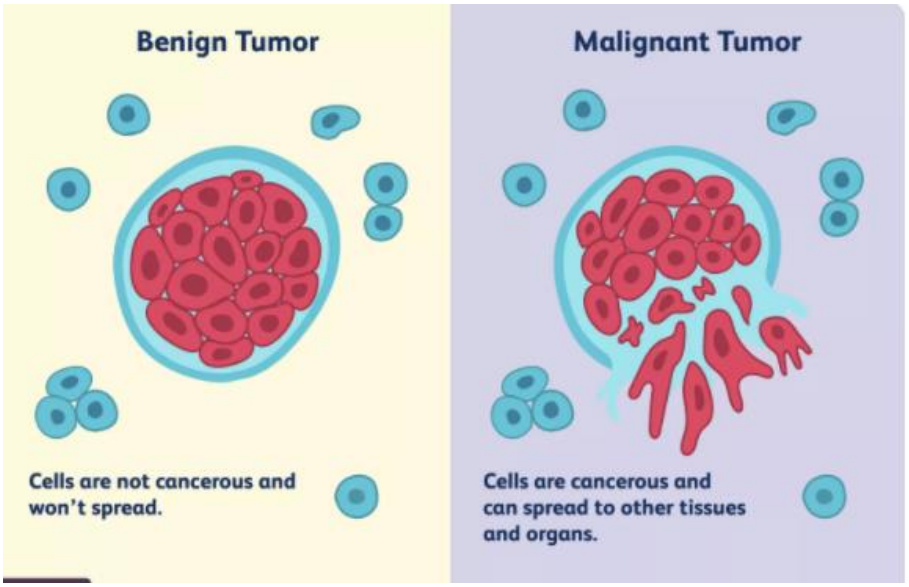

Fig 2.1

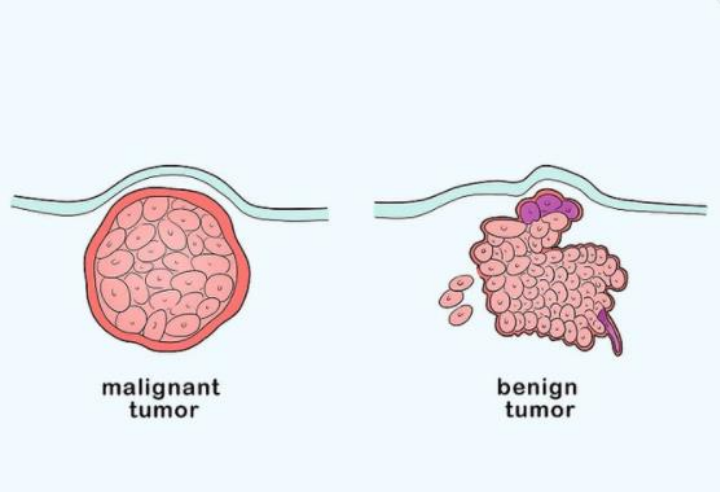

Fig 2.2

Chapter-3 Methods used for Training & Prediction

➢ Classification Algorithms

- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog,** etc. Classes can be called as targets/labels or categories.
- Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.
- In classification algorithm, a discrete output function(y) is mapped to input variable(x).
- y=f(x), where y = categorical output
- The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.
- Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.

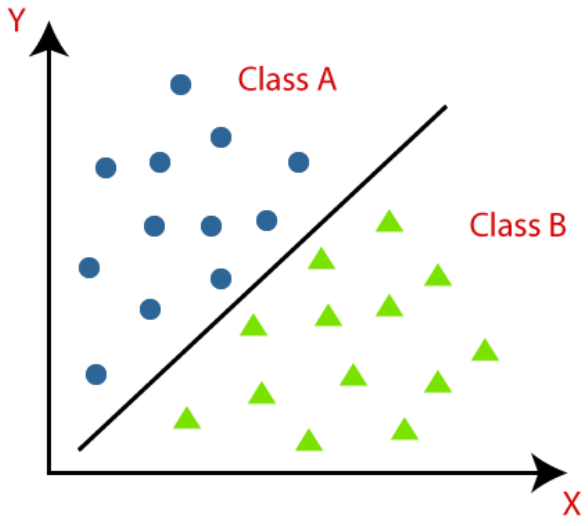

Fig 3.1(Linear Separable data)

The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications:

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier.

  Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier.

  Example: Classifications of types of crops, Classification of types of music.

➢ **Linear Models**

- Logistic Regression

➢ **Non-Linear Models**

- Decision Tress Classification
- Random Forest Classification

➢ **Logistic Regression**

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

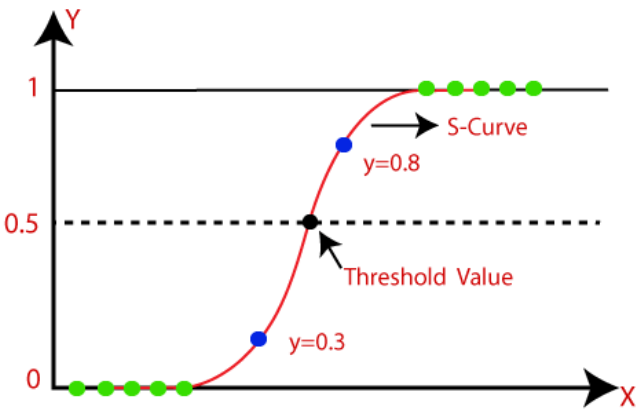

Fig3.2

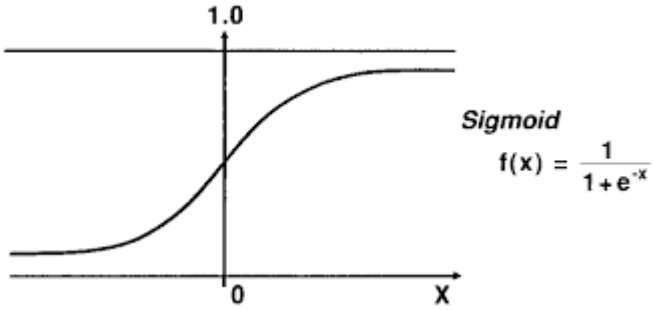

Fig 3.3 (Logistic Regression)

- **Decision Tree Classification**

  - 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.
  - A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.
  - Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
  - **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.

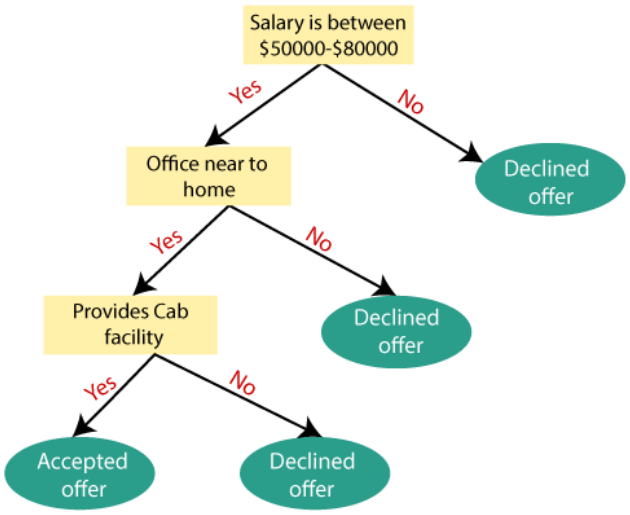

Fig 3.4 (Decision Tree Classification)

➢ **Random Forest Classification**

- 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.**
- 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.

- 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.

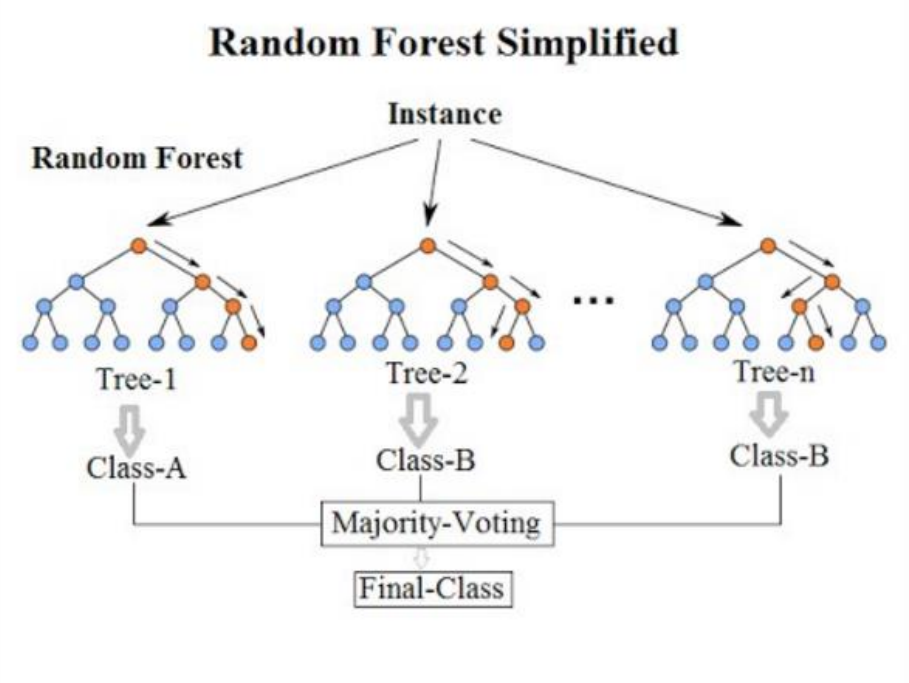

Fig 3.5 (Random Forest Classification)

- **Regression Analysis**

- Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables.
- More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc
- Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables.

- It is mainly used for **prediction, forecasting, time series modeling, and determining the causal-effect relationship between variables**.
- Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum

- **Dependent Variable:** The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called **target variable**.
- **Independent Variable:** The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a **predictor**.
- **Outliers:** Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.
- **Multicollinearity:** If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.
- **Underfitting and Overfitting:** If our algorithm works well with the training dataset but not well with test dataset, then such problem is called **Overfitting**. And if our algorithm does not perform well even with training dataset, then such problem is called **underfitting**.

Advertisement	Sales
$90	$1000
$120	$1300
$150	$1800
$100	$1200
$130	$1380
$200	??

Fig 3.6(Regression Sales Example)

- **Linear Regression**

  - Linear regression is a statistical regression method which is used for predictive analysis.
  - It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables.
  - It is used for solving the regression problem in machine learning.
  - Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.
  - If there is only one input variable (x), then such linear regression is called **simple linear regression**. And if there is more than one input variable, then such linear regression is called **multiple linear regression**.
  - The relationship between variables in the linear regression model can be explained using the below image. Here we are predicting the salary of an employee on the basis of **the year of experience**.

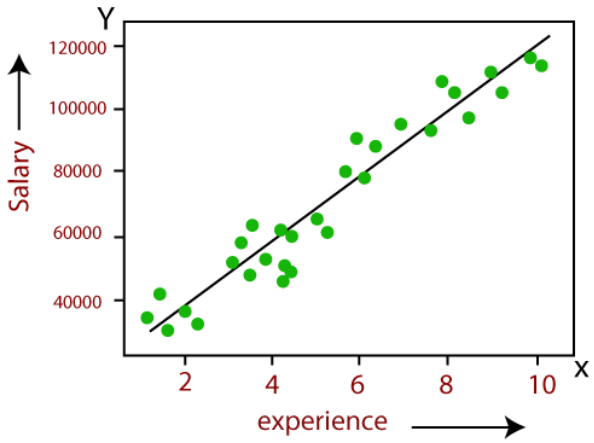

Fig 3.6(Linear Regression)

- Like Y=aX+b equation.
- Y = dependent variables (target variables).
- X= Independent variables (predictor variables), a and b are the linear coefficients.

➢ **Difference Between Linear Regression and Logistic Regression**

Linear Regression	**Logistic Regression**
Linear Regression is a supervised regression model.	Logistic Regression is a supervised classification model.
In Linear Regression, we predict the value by an integer number.	In Logistic Regression, we predict the value by 1 or 0.
Here no activation function is used.	Here activation function is used to convert a linear regression equation to the logistic regression equation

Here no threshold value is needed.	Here a threshold value is added.
Here we calculate Root Mean Square Error(RMSE) to predict the next weight value.	Here we use precision to predict the next weight value.
Here dependent variable should be numeric and the response variable is continuous to value.	Here the dependent variable consists of only two categories. Logistic regression estimates the odds outcome of the dependent variable given a set of quantitative or categorical independent variables.
It is based on the least square estimation.	It is based on maximum likelihood estimation.
Here when we plot the training datasets, a straight line can be drawn that touches maximum plots.	Any change in the coefficient leads to a change in both the direction and the steepness of the logistic function. It means positive slopes result in an S-shaped curve and negative slopes result in a Z-shaped curve.
Linear regression is used to estimate the dependent variable in case of a change in independent variables. For example, predict the price of houses.	Whereas logistic regression is used to calculate the probability of an event. For example, classify if tissue is benign or malignant.
Linear regression assumes the normal or gaussian distribution of the dependent variable.	Logistic regression assumes the binomial distribution of the dependent variable.

Chapter-4 Source Code Explanation and Implementation

➢ Complete Code Explanation with Snapshots

```
In [1]: #importing libraries
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Fig 4.1

- In this cell 1, I have initialized all the libraries like numpy for data manipulation, Pandas for dataframes, matplotlib and seaborn for data visualization.

```
In [2]: #getting the data
        data = pd.read_csv(r'C:\Users\lalli\OneDrive\Desktop\ML project\breast cancer data.csv')
```

Fig 4.2

- In cell 2, we can see that I have loaded the data of breast cancer prediction that I got it from Kaggle datacenter site. And I load it in program in a pandas dataframe variable “data”. And I can access this by calling data. Pandas and numpy are the poweful features provided by python for data manipulation and visualization.

```
In [3]: data.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns

Fig 4.3

- By data.head( ) command we can view the first 5 rows of our dataset.
- In this data we are interested in “diagnosis” column because it is the target variable to us. That means we need to predict that variable “M” – malignant cells (harmful) and “B” – benign cells (not harmful)
- In this data we have 569 rows of data and 33 columns.

```
In [26]: data.columns

Out[26]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

Fig 4.4

- Here by data.columns command we can see the names of the all the columns that are in our data.

```
In [4]: #counting no.of rows and columns of data(shape)
        data.shape

Out[4]: (569, 33)
```

Fig 4.5

- In this cell, by the command data.shape we can see the (no.of rows, no.of columns) that are in our data

```
In [5]: #counting no.of empty values in each column
        data.isna().sum()
```

```
Out[5]: id                         0
        diagnosis                  0
        radius_mean                0
        texture_mean               0
        perimeter_mean             0
        area_mean                  0
        smoothness_mean            0
        compactness_mean           0
        concavity_mean             0
        concave points_mean        0
        symmetry_mean              0
        fractal_dimension_mean     0
        radius_se                  0
        texture_se                 0
        perimeter_se               0
        area_se                    0
        smoothness_se              0
        compactness_se             0
        concavity_se               0
        concave points_se          0
        symmetry_se                0
        fractal_dimension_se       0
        radius_worst               0
        texture_worst              0
        perimeter_worst            0
        area_worst                 0
        smoothness_worst           0
        compactness_worst          0
        concavity_worst            0
        concave points_worst       0
        symmetry_worst             0
        fractal_dimension_worst    0
        Unnamed: 32              569
        dtype: int64
```

Fig 4.6

- By the command data.isna().sum() , this will return you output with no.of null values in each column of our data. As a part of data cleaning, it is important for us to remove the unwanted and unnecessary data. Here in our data, you can see Unnamed: 32 is having all null values so better to remove it.

```
In [12]: data.iloc[:,0:34].nunique()

Out[12]: id                         569
         diagnosis                    2
         radius_mean                456
         texture_mean               479
         perimeter_mean             522
         area_mean                  539
         smoothness_mean            474
         compactness_mean           537
         concavity_mean             537
         concave points_mean        542
         symmetry_mean              432
         fractal_dimension_mean     499
         radius_se                  540
         texture_se                 519
         perimeter_se               533
         area_se                    528
         smoothness_se              547
         compactness_se             541
         concavity_se               533
         concave points_se          507
         symmetry_se                498
         fractal_dimension_se       545
         radius_worst               457
         texture_worst              511
         perimeter_worst            514
         area_worst                 544
         smoothness_worst           411
         compactness_worst          529
         concavity_worst            539
         concave points_worst       492
         symmetry_worst             500
         fractal_dimension_worst    535
         Unnamed: 32                  0
         dtype: int64
```

Fig 4.7

- data.iloc[:,0:34].nunique() , you can see above by using this command we can get how many unique value are there in each column.

```
In [13]: data['radius_mean'].unique()

Out[13]: array([17.99 , 20.57 , 19.69 , 11.42 , 20.29 , 12.45 , 18.25 , 13.71 ,
                13.   , 12.46 , 16.02 , 15.78 , 19.17 , 15.85 , 13.73 , 14.54 ,
                14.68 , 16.13 , 19.81 , 13.54 , 13.08 ,  9.504, 15.34 , 21.16 ,
                16.65 , 17.14 , 14.58 , 18.61 , 15.3  , 17.57 , 18.63 , 11.84 ,
                17.02 , 19.27 , 16.74 , 14.25 , 13.03 , 14.99 , 13.48 , 13.44 ,
                10.95 , 19.07 , 13.28 , 13.17 , 18.65 ,  8.196, 12.05 , 13.49 ,
                11.76 , 13.64 , 11.94 , 18.22 , 15.1  , 11.52 , 19.21 , 14.71 ,
                13.05 ,  8.618, 10.17 ,  8.598,  9.173, 12.68 , 14.78 ,  9.465,
                11.31 ,  9.029, 12.78 , 18.94 ,  8.888, 17.2  , 13.8  , 12.31 ,
                16.07 , 13.53 , 18.05 , 20.18 , 12.86 , 11.45 , 13.34 , 25.22 ,
                19.1  , 12.   , 18.46 , 14.48 , 19.02 , 12.36 , 14.64 , 14.62 ,
                15.37 , 13.27 , 13.45 , 15.06 , 20.26 , 12.18 ,  9.787, 11.6  ,
                14.42 , 13.61 ,  6.981,  9.876, 10.49 , 13.11 , 11.64 , 22.27 ,
                11.34 ,  9.777, 12.63 , 14.26 , 10.51 ,  8.726, 11.93 ,  8.95 ,
                14.87 , 17.95 , 11.41 , 18.66 , 24.25 , 14.5  , 13.37 , 13.85 ,
                19.   , 19.79 , 12.19 , 15.46 , 16.16 , 15.71 , 18.45 , 12.77 ,
                11.71 , 11.43 , 14.95 , 11.28 ,  9.738, 16.11 , 12.9  , 10.75 ,
                11.9  , 11.8  , 14.44 , 13.74 ,  8.219,  9.731, 11.15 , 13.15 ,
                12.25 , 17.68 , 16.84 , 12.06 , 10.9  , 11.75 , 19.19 , 19.59 ,
                12.34 , 23.27 , 14.97 , 10.8  , 16.78 , 17.47 , 12.32 , 13.43 ,
                11.08 , 10.66 ,  8.671,  9.904, 16.46 , 13.01 , 12.81 , 27.22 ,
                21.09 , 15.7  , 15.28 , 10.08 , 18.31 , 11.81 , 12.3  , 14.22 ,
                 9.72 , 14.86 , 12.91 , 13.77 , 18.08 , 19.18 , 14.45 , 12.23 ,
                17.54 , 23.29 , 13.81 , 12.47 , 15.12 , 17.01 , 15.27 , 20.58 ,
                28.11 , 17.42 , 14.19 , 13.86 , 11.89 , 10.2  , 19.8  , 19.53 ,
                13.65 , 13.56 , 10.18 , 15.75 , 14.34 , 10.44 , 15.   , 12.62 ,
                12.83 , 17.05 , 11.32 , 11.22 , 20.51 ,  9.567, 14.03 , 23.21 ,
                20.48 , 17.46 , 12.42 , 11.3  , 13.75 , 19.4  , 10.48 , 13.2  ,
                12.89 , 10.65 , 20.94 , 11.5  , 19.73 , 17.3  , 19.45 , 13.96 ,
                19.55 , 15.32 , 15.66 , 15.53 , 20.31 , 17.35 , 17.29 , 15.61 ,
                17.19 , 20.73 , 10.6  , 13.59 , 12.87 , 10.71 , 14.29 , 11.29 ,
                21.75 ,  9.742, 17.93 , 11.33 , 18.81 , 19.16 , 11.74 , 16.24 ,
                12.58 , 11.26 , 11.37 , 14.41 , 14.96 , 12.95 , 11.85 , 12.72 ,
                10.91 , 20.09 , 11.46 ,  9.   , 13.5  , 11.7  , 14.61 , 12.76 ,
                11.54 ,  8.597, 12.49 ,  9.042, 12.43 , 10.25 , 20.16 , 20.34 ,
```

Fig 4.8

- In this we can see the unique values or words in each row by using the command data[‘columnname’].unique()

```
In [14]: #droping column with all missing values
         data = data.dropna(axis=1)

In [15]: #checking the no.of rows and column after droping empty
         data.shape

Out[15]: (569, 32)
```

Fig 4.9

- By this command used in cell 14 data.dropna(axis=1). Here we are droping nullvalues and axis=1 this attribute indicates drop the column that have all null values in it and we are update back the content to that dataframe variable “data” back.
- And by using data.shape we can actually see the difference because earlier we were having 33 columns and now we are having 32. We have actually dropped the Unnamed: 32 column.

```
In [17]: data.describe()
```

Out[17]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetr
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0

8 rows × 31 columns

Fig 4.10

- data.describe() this function returns statistical analysis like the count, mean, std, min, 25%(data), 50%(data), 75%(data), maximum value of each columns are returned by this function.

```
In [18]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

Fig 4.11

- info() that is applied on the dataframe data, returns columns names with no.of non-null records and type of data (“dtype”) that is in that column.(information of 31 columns are returned)

```
In [8]: #get count of malignant(M) or benign(B) cells
        data['diagnosis'].value_counts()

Out[8]: B    357
        M    212
        Name: diagnosis, dtype: int64
```

```
In [9]: #visualization of data count
        sns.countplot(data['diagnosis'],label='count')

        C:\Users\lalli\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following va
        rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments w
        yword will result in an error or misinterpretation.
          warnings.warn(

Out[9]: <AxesSubplot:xlabel='diagnosis', ylabel='count'>
```

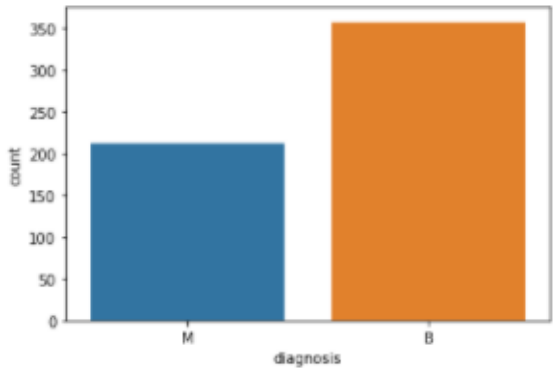

Fig 4.12

- In cell 8, we are about to find the record of no.of Malignant cells(M) in data (i.e. 212), and no.of Benign cells(B) (i.e.357) as per the data we have.
- In cell 9, we can see the same data of no.of malignant and no.of benign cells in the from of visualization using seaborne by typing that particular command.

```
In [63]: #encoding
         from sklearn.preprocessing import LabelEncoder
         le_Y = LabelEncoder()
         data.iloc[:,1]=le_Y.fit_transform(data.iloc[:,1].values)
         print(le_Y.fit_transform(data.iloc[:,1].values))
         #data.iloc[:,1].values
         #data.iloc[:,1]

         [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
          0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
          0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
          0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
          0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
          0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
          0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
          1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
          0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
          0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
          1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
          0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0
          1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
          0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
          0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 1 1 1 1 1 1 0]
```

Fig 4.13

- Now here if you see in cell 63, we have imported LabelEncoder from sklearn.preprocessing library that is used to convert the categorical data to binary format to train the model and make prediction in easier way. (Like in our data M

and B → 1 and 0.) those lines of code enter there transform all M to 1 and all B to 0 in our dataframe.

```
In [113]: #a pair plot
          sns.pairplot(data.iloc[:,1:5],hue='diagnosis')
Out[113]: <seaborn.axisgrid.PairGrid at 0x1688b521f70>
```

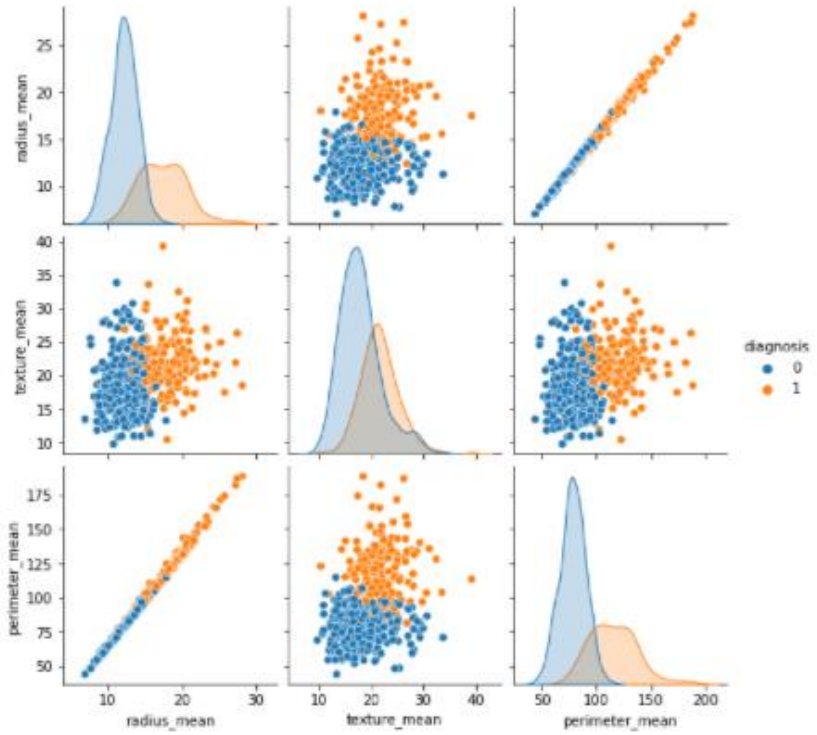

Fig 4.14

- I have used pairplot() to determine the multiple pairwise bivariate distributions here I have selected first 4 columns excluding "id" column and I have used hue attribute "data" to map plot aspects to different colors with respect to the "diagnosis" column values .
- Here in the graphs 0 represents Benign cells and 1 represents Malignant cells.

```
In [17]: data.head(10)
```

Out[17]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	ra
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	
5	843786	1	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	...	
6	844359	1	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	...	
7	84458202	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	...	
8	844981	1	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	...	
9	84501001	1	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	...	

10 rows × 32 columns

Fig 4.15

- Now, here we can see that the “diagnosis” columns values in our dataframe “data” are update in binary variable format.

```
In [18]: #correlation of columns
         data.iloc[:,1:12].corr()
```

Out[18]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	conc points_m
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	0.776
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166

Fig 4.16

- By using this corr() function we can find correlation between variable of the dataframe.
- Data.iloc[:,1:12].corr(), this command return the correlation between the variable of first 11 columns excluding “id” column.
- It also gives whether there is +ve correlation or -ve correlation between column variables.

```
In [28]: #visualize the correlation
         plt.figure(figsize=(10,10))
         sns.heatmap(data.iloc[:,1:12].corr(),annot=True, fmt='.0%')
```

Out[28]: <AxesSubplot:>

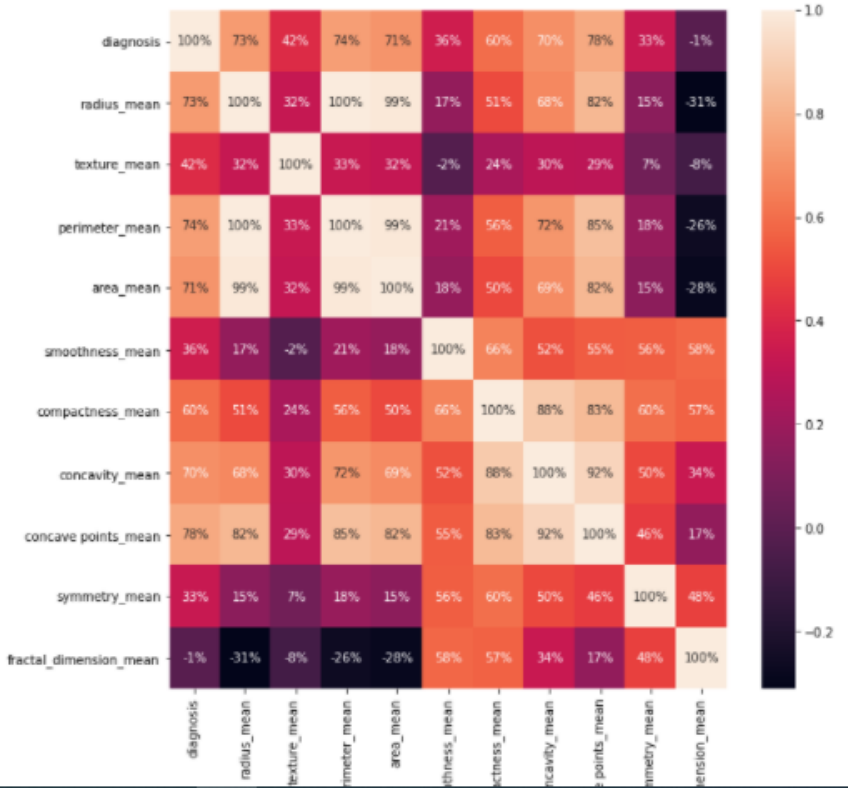

Fig 4.17

- Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix.
- **annot:** If True, write the data value in each cell.
- **fmt:** String formatting code to use when adding annotations.(basically used to return the relational values in the form of percentage.)
- In heatmap, by default darker color represent less correlation and lighter colors represent more correlation.

```
In [100]: #spliting data into independent(X) and dependent(Y) datasets
          X = data.iloc[:,2:32].values  #changing from pd to nparray(array) type bcoz of parameter that we taking for model
          Y = data.iloc[:,1].values
          type(X)

Out[100]: numpy.ndarray

In [101]: #spliting dataset into 75% traning and 25% testing
          from sklearn.model_selection import train_test_split
          X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.4, random_state = 0)

In [114]: #Feature scaling
          from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()
          X_train = sc.fit_transform(X_train)
          X_test = sc.transform(X_test)

          X_train

Out[114]: array([[-0.84952106,  2.37711987, -0.89596908, ..., -1.79184446,
                  -2.03197496, -1.36800648],
                 [ 1.56737211,  0.66649792,  1.52471723, ...,  1.88619234,
                  -0.18519113, -0.42361718],
                 [ 0.58208813, -0.86168928,  0.49246772, ...,  0.29822921,
                  -0.90423841, -0.63624284],
                 ...,
                 [-1.34468942, -0.24093727, -1.33865316, ..., -1.01372713,
                  -0.66762883, -0.13588219],
                 [-1.26496844, -0.26463009, -1.30152482, ..., -1.79184446,
                  -1.5049809 , -1.00681899],
                 [-0.76250168,  1.0882302 , -0.73888763, ..., -0.30507356,
                  -1.19308646,  0.19327338]])
```

Fig 4.18

- In cell 100, we have take a variable X and stored all the independent variable values. In the form of numpy n-dimensional array.
- And in another variable Y, we have stored the dependent variable “diagnosis” values.
- Now we come to the main part of the program “Train Test Split”
- Here we import train_test_split function from sklearn.model_selection library
- To the variable X_train, Y_train we are storing the data that is for training purpose and in X_test, Y_test we will store the rest of the data for testing purpose like to match the perdictions by the model and check the test accuracy of the model
- The ideal state to divide the data in 70 : 30 proportion like 70% of data for training and 30% of data for testing purpose.

- But actually after checking the “Testing Score” of the model, I have changed the “test_size” variable from 0.3 to 0.25 the score was so good again the I have change to 0.35 even then it was so accurate. Then I have set it to 0.4 which actually gave pretty good accuracy as compared to rest of the values. So, I have set the “test_size=0.4”.
- As this prediction application is used for medical purposes. I need to make sure that it gives accuracy near to 100%. Now we got around 98.2% accuracy at 0.4 test size which is actually a great achievement.
- Random_state can **be 0 or 1 or any other integer.** Random_state is **basically used for reproducing your problem the same every time it is run**. If you do not use a random_state in train_test_split, every time you make the split you might get a different set of train and test data points and will not help you in debugging in case you get an issue.
- StandardScaler **removes the mean and scales each feature/variable to unit variance**. This operation is performed feature-wise in an independent way. StandardScaler can be influenced by outliers (if they exist in the dataset) since it involves the estimation of the empirical mean and standard deviation of each feature.

In [103]:
```
#create a function for the models
def Models(X_train, Y_train):

    #Logistic regression
    from sklearn.linear_model import LogisticRegression
    log = LogisticRegression(random_state=0)
    log.fit(X_train, Y_train)

    #Descision Tree
    from sklearn.tree import DecisionTreeClassifier
    tree = DecisionTreeClassifier(criterion='entropy',random_state=0)
    tree.fit(X_train, Y_train)

    #Randomforest Classifier
    from sklearn.ensemble import RandomForestClassifier
    forest = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=0)
    forest.fit(X_train, Y_train)

    #accuracy printing
    print("[0]logistic regression traning accuracy:",log.score(X_train, Y_train))
    print("[1]decision tree classifier traning accuracy:",tree.score(X_train, Y_train))
    print("[2]random forest classifier traning accuracy:",forest.score(X_train, Y_train))

    return log,tree,forest
```

In [104]:
```
#all models accuracy values
model = Models(X_train, Y_train)
```

```
[0]logistic regression traning accuracy: 0.9882697947214076
[1]decision tree classifier traning accuracy: 1.0
[2]random forest classifier traning accuracy: 0.9970674486803519
```

Fig 4.19

- Here in this cell 103, I have created 3 training models and choosen the best one out of them. To make the coding easy and simple I have put them in a module(function) called “Models” with parameters (X_train, Y_train).

- From sklearn.linear_model library we have imported LogisticRegression function model. And given it to a variable “log”.
- log.fit( ) this function helps us to train the logistic regression model with the X_train and Y_train variables(values).
- From sklearn.tree library we have imported DecisionTreeClassifier function model. And given it to a variable “tree”.
- ‘**Criterion is a standard by which the values are decided**’.**Entropy** refers to disorder or uncertainty. Entropy controls how a Decision Tree decides to **split** the data. It actually effects how a **Decision Tree** draws its boundaries all non-empty classed $p(i \mid t) \neq 0$, where $p(i \mid t)$ is the proportion (or frequency or probability) of the samples that belong to class i for a particular node t; C is the number of unique class labels.
- Entropy(t) = - Σp(i|t) log2 p(i | t) (from i=1 to C)
- tree.fit( ) this function helps us to train the DecisionTreeClassifier model with the X_train and Y_train variables(values).
- From sklearn.ensemble library we have imported RandomForestClassifier function model. And given it to a variable “forest”.
- n_estimators this the no.of trees you want to build before taking maximum voting or average predictions. Higher number of trees give you better performance but makes your code slow.
- You can choose as high value as your processor can handle because this makes your prediction stronger and more stable.
- forest.fit( ) this function helps us to train the RandomForestClassifier model with the X_train and Y_train variables(values).
- And after training of all the 3 models is completed, we have printed the Training score of each model with the function score( ) by passing X_train and Y_train variables(values) as arguments in this function.
- From the output that we have got we can see that DecisionTreeClassifier model has really performed well with Test score of 100% followed by RandomForestClassifier model with 99.7 % score and at last we have LogisticRegression model with 98.89% test score.

```
In [105]: from sklearn.metrics import accuracy_score
          from sklearn.metrics import classification_report

          for i in range(len(model)):
              print('Model ',i)
            #Check precision, recall, f1-score
              print( classification_report(Y_test, model[i].predict(X_test)) )
            #Another way to get the models accuracy on the test data
              print( accuracy_score(Y_test, model[i].predict(X_test)))
              print()#Print a new line

Model  0
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       145
           1       0.98      0.98      0.98        83

    accuracy                           0.98       228
   macro avg       0.98      0.98      0.98       228
weighted avg       0.98      0.98      0.98       228

0.9824561403508771

Model  1
              precision    recall  f1-score   support

           0       0.95      0.96      0.95       145
           1       0.93      0.90      0.91        83

    accuracy                           0.94       228
   macro avg       0.94      0.93      0.93       228
weighted avg       0.94      0.94      0.94       228

0.9385964912280702

Model  2
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       145
           1       0.94      0.87      0.90        83

    accuracy                           0.93       228
   macro avg       0.93      0.92      0.92       228
```

Fig 4.20

- There are four ways to check if the predictions are right or wrong:

1. **TN / True Negative**: the case was negative and predicted negative
2. **TP / True Positive**: the case was positive and predicted positive
3. **FN / False Negative**: the case was positive but predicted negative
4. **FP / False Positive**: the case was negative but predicted positive

- **Precision — What percent of your predictions were correct?**

- Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive.

- Precision:- Accuracy of positive predictions.

- Precision = TP/(TP + FP)

- **Recall —** ***What percent of the positive cases did you catch?***

- Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives.

- Recall:- Fraction of positives that were correctly identified.

- Recall = TP/(TP+FN)

- **F1 score —** ***What percent of positive predictions were correct?***

- The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

- **Support**

- Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process.

- "Accuracy score". It is the ratio of number of correct predictions to the total number of input samples.

- A **macro-average** will compute the metric independently for each class and then take the **average** (hence treating all classes equally).

- Weighted average or weighted sum ensemble is an ensemble machine learning approach that **combines the predictions from multiple models**, where the contribution of each model is weighted proportionally to its capability or skill. The weighted average ensemble is related to the voting ensemble

```
In [106]: pred = model[0].predict(X_test)
          print(pred)

          #Print a space
          print()

          #Print the actual values
          print(Y_test)
```

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1
 0 0 0 0 1 0 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0
 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 0
 1 0 0 0 0 1]

[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 0 1
 0 0 0 0 0 0 0 1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0
 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 0
 1 0 0 0 0 1]
```

Fig 4.21

- Here in this cell, I have printed first the values that are in Y_test and secondly I had printed values that the logistic regression model has predicted based on the input X_test.
- Here we can see the expected outcome and predicted outcome in the form of binary.
- We can see that is no marginal difference between the test data and predicted data that means we have successfully trained our model to give at most accurate data.

CONCLUSION AND FUTURE PRESPECTIVE

- Machine learning is on a demand at this particular moment.
- Every industry looking to apply ML expert in their domain, studying machine learning opens world of opportunities to develop cutting edge machine learning applications in various aspects – such as cyber security, image recognition, medical fields and face recognition.
- Several machine learning companies on the verge of hiring skilled ML engineers as it is becoming the master of business intelligence.
- By taking this course I have gained good amount of understanding on the fundamental issues and challenges of machine learning: data, model selection, model complexity, training of model and etc.
- I got to know the mathematical relationships within Machine Learning algorithms and the paradigms of different types of learnings models.
- I am able to understand the implementation of various machine learning algorithms in a range of real-world applications.
- I improved my ability to integrate machine learning libraries and mathematical, statistical tools with modern technologies.
- Now I have an idea that how we encode the real-world data in the Binary variables (0s and 1s).
- Board infinity has helped me to gain knowledge on all the topics and how to apply them to real life applications and use them to develop projects.
- The knowledge that I have gained here has helped me to code better in python and also helped me in competitive coding.
- The experience that I have gained by building and training the Machine Learning models is very effective for my project Breast cancer prediction.
- In terms of future scope and job opportunities in the present market for machine learning engineers is not just hot but it’s sizzling.
- All the top tech companies are investing into Artificial intelligence, Machine Leaning and Deep Learning as the see that the world future is closely integrated with this niche technologies.
- Having a certification like this is precious and adds value to my resume

REFERENCES

- https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
- https://www.youtube.com
- https://my.clevelandclinic.org/health/diseases/3986-breast-cancer
- https://scikit-learn.org
- https://www.geeksforgeeks.org
- https://seaborn.pydata.org
- https://www.javatpoint.com
- https://medium.com/analytics-vidhya
- https://stackoverflow.com

All Images from

- https://www.google.co.in/imghp?hl=en&authuser=0&ogbl
- https://www.javatpoint.com