Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

IoT DOMAIN ANALYST

ECE3502

PROJECT REPORT FILE

Under the Guidance of

Prof. Dr. SITHARTHAN R

BREAST CANCER PREDICTION USING MACHINE LEARNING ALGORITHMS

TEAM MEMBERS :-

HARSH KUMAR (20BEI0088)

ROHAN JHA(20BEE0147)

AMIT KUMAR (20BEE0092)

HIMANSHU SINGH(20BEE0116)

CERTIFICATE

We hereby declare that the report file titled " BREAST CANCER PREDICTION USING MACHINE LEARNING ALGORITHMS" submitted by our team for the course – IoT DOMAIN ANALYST to VIT university, Vellore as a record of bonafide work carried out by us under the guidance of Prof. Dr. SITHARTHAN R We further declare that the work done for this project work has not been used earlier and shall not be used further for any other courses in this or any other university.

SIGNATURE

(Dr. SITHARTHAN R)

Abstract

Breast cancer is a disease which we hear about a lot nowadays. It is one of the most widespread diseases. It affects women all around the world. The National Cancer Institute says that breast cancer is the second most common cancer for women in the United States. There are around 2000+ new cases of breast cancer in men each year and about 2,30,000 new cases in women every year. Diagnosis of this disease is crucial so that woman can get it treated faster. It is best for a correct and early diagnosis. This is an important step in rehabilitation and treatment. Breast cancer detection is done with the help of mammograms, which are basically X-rays of the breasts. It’s a tool which is used to detect and help diagnose breast cancer. But, detection is not easy due to different kinds of uncertainties in using these mammograms. Machine Learning (ML) techniques can help in the detection of breast cancer. We can use these techniques to make tools for doctors that can be used as an effective mechanism for early detection and diagnosis of breast cancer which could greatly enhance the survival rate of patients.

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

Our team from the SELECT School have great pleasure in expressing our deep sense of gratitude to our faculty, Prof. Dr. SITHARTHAN R, SELECT for providing the necessary infrastructure, information and creating good environment as well as guiding us with his immense knowledge throughout the course of this project work. Finally, we would extend our note of gratitude towards the other non-teaching staff of SELECT for cooperating with us during the completion of this project.

- **HARSH KUMAR 20BEI0088**
- **ROHAN JHA 20BEE0147**
- **AMIT KUMAR 20BEE0092**
- **HIMANSHU SINGH 20BEE0116**

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 DOMAIN SPECIFIC

Cancer is a disease that occurs when there are changes or mutations that take place in genes that help in cell growth. These mutations allow the cells to divide and multiply in a very uncontrolled and chaotic manner. These cells keep increasing and start making replicas which end up becoming more and more abnormal. These abnormal cells later on form a tumor. Tumors, unlike other cells, don't die even though the body doesn't need them.

The cancer that develops in the breast cells is called breast cancer. This type of cancer can be seen in the breast ducts or the lobules. Cancer can also occur in the fatty tissue or the fibrous connective tissue within the breast. These cancer cells become uncontrollable and end up invading other healthy breast tissues and can travel to the lymph nodes under the arms.

There are two types of cancers. Malignant and Benign. Malignant cancers are cancerous. These cells keep dividing uncontrollably and start affecting other cells and tissues in the body. They spread to all other parts of the body and it is hard to cure this type of cancer. Chemotherapy, radiation therapy and immunotherapy are types of treatments that can be given for these types of tumors. Benign cancer is non-cancerous. Unlike malignant, this tumor does not spread to other parts of the body and hence is much less risky that malignant. In many cases, such tumors don't really require any treatment.

Breast cancer is most commonly diagnosed in women of ages above 40. But this disease can affect men and woman of any age. It can also occur when there's a family history of breast cancer. Breast Cancer has always had a high mortality rate and according statistics, it alone accounts for about 25% of all new cancer diagnoses and 15% of all cancer deaths among women worldwide. Scientists know about the dangers of it from very early on, and hence there's been a lot of research put into finding the right treatment for it.

Breast cancer detection is done with the help of mammograms, which are basically X-rays of the breasts. It’s a tool which can help detect and diagnose breast cancer. But, detection is not easy due to different kinds of uncertainties in using these mammograms. The result of a mammogram are images that can show any calcifications or deposits of calcium in the breasts. These don’t always have to be cancerous. These test can also find cysts which are fluid-filled sacs that are very normal during some women’s menstrual cycles — and any cancerous or noncancerous lumps.

Mammograms can cost around ₹15,000 - ₹40,000 or more based on the hospital, location, and area of body to be covered. This is very expensive and not many can afford it.

It is always best for an early diagnosis so that the treatment process can also be started early on.

1.2 OBJECTIVES OF THE PROJECT

Breast cancer is a disease which we hear about a lot nowadays. It is one of the most widespread diseases. There are around 2000+ new cases of breast cancer in men each year, and about 2,30,000 new cases in women every year. Diagnosis of this disease is crucial so that woman can get it treated faster. It is best for a correct and early diagnosis.
The main objective of this project is to help doctors analyze the huge datasets of cancer data and find patterns with the patient’s data and that cancer data available. With this analysis we can predict whether the patient might have breast cancer or not.
Machine learning algorithms will help with this analysis of the datasets. These techniques will be used to predict the outcome. The outcome can be either that the cancer is benign or malignant. Benign cancer is the cancer which doesn’t spread whereas malignant cancer cells spread across the body making it very dangerous.
This prediction can help doctors prescribe different medical examinations for the patients based on the cancer type. This helps save a lot of time as well as money for the patient.

1.3 PROBLEM DEFINITION

Over the years, a continuous evolution related to cancer research has been performed. Scientists used various methods, like early stage screening, so that they could find different types of cancer before it could do any damage. With this research, they were able to develop new strategies to help predict early cancer treatment outcome.

With the arrival of new technology in the medical field, huge amount of data related to cancer has been collected and is available for medical research. But, physicians find the accurate prediction of the cancer outcome as the most interesting yet challenging part.

For this reason, machine learning techniques have become popular among researchers. These tools can help discover and identify patterns and relationships between the cancer data, from huge datasets, while they are able to effectively predict future outcomes of a cancer type. Patients have to spend a lot of money on different tests and treatments to check whether they have breast cancer or not. These tests can take a long time and the results can be delayed. Also, after confirmation that the patient has cancer, more tests need to be done to check whether the cancer is benign or malignant.

In this project, I will be using different machine learning techniques to analyze the data given in the datasets. This analysis will help us predict whether the cancer is benign or malignant. Benign cancer is the cancer which doesn’t spread whereas malignant cancer cells spread across the body making it very dangerous.

Machine learning algorithms will help with this analysis of the datasets. These techniques will be used to predict the outcome. The outcome can be either that the cancer is benign or malignant. Benign cancer is the cancer which doesn’t spread whereas malignant cancer cells spread across the body making it very dangerous.

This prediction can help doctors prescribe different medical examinations for the patients based on the cancer type. This helps save a lot of time as well as money for the patient.

1.4 PROJECT FEATURES

This project scheme was developed to reduce some amount of work for the physicians and other doctors so that they don’t have to conduct many tests on the patients. It also helps minimize the amount of time and money spent by the patients undergo these tests. As everything is digitalized and based on data analysis, it takes less amount of time to get results. Based on the results, further action can be taken. It also helps researchers in the medical as well as IT sector to understand how different algorithms can predict different outcomes.

This scheme normally requires a huge amount of data about different patient history and the cancer details. This data has been collected by many doctors for a long period of time and will be used to do the analysis. This reduces the computational time required to gather all the data necessary.

CHAPTER 2

LITERATURE SURVEY

2.1 DOMAIN SPECIFIC

Literature survey is a very important step in the software development process. Before building any new tool, we need to check the time factor, economy and company strength. When these things are fulfilled, at that point following stages is to figure out which working framework and language can be utilized for building up the device. A lot of help is required for building the tool, internal as well as external. Senior programmers can help and provide this support to the developers from various sources like research papers, books or online websites. Before building the framework the above thought are considered for building up the proposed framework.

Machine Learning

Machine Learning is a sub category of Artificial Intelligence which allows systems to automatically learn and understand data from experience without the system being programmed to do so. It helps software applications become better at predicting outcomes for various types of problems. The basic idea of ML is to take in input data and use different algorithms to help it predict outcomes and also update these outcomes when new data is available as input.

The procedures used with machine learning are like that of data mining and predictive modeling. Both require scanning through huge amounts of data to search for any type of pattern in the data and then modify the program accordingly.

Machine Learning has been seen many a times by individuals while shopping on the internet. They are then shown ads based on what they were searching earlier on. This happens because many of these websites use machine learning to customize the ads based on user searches and this is done in real time. Machine learning has also been used in other various places like detecting fraud, filtering of spam, network security threat detection, predictive maintenance and building news feeds.

Machine Learning methods:

- Supervised learning – Here both the input and output is known. The training dataset also contains the answer the algorithm should come up with on its own. So, a labeled dataset of fruit images would tell the model which photos were of apples, bananas and oranges. When a new image is given to the model, it compares it to the training set to predict the correct outcome.
- Unsupervised learning – Here input dataset is known but output is not known. A deep learning model is given a dataset without any instructions on what to do with it. The training data contains information without any correct result. The network tries to automatically understand the structure of the model.
- Semi-supervised learning – This type comes somewhere between supervised and unsupervised learning. It contains both labelled and un-labelled data.
- Reinforcement learning – In this type, AI agents are trying to find the best way to accomplish a particular goal. It tries to predict the next step which could possibly give the model the best result at the end.

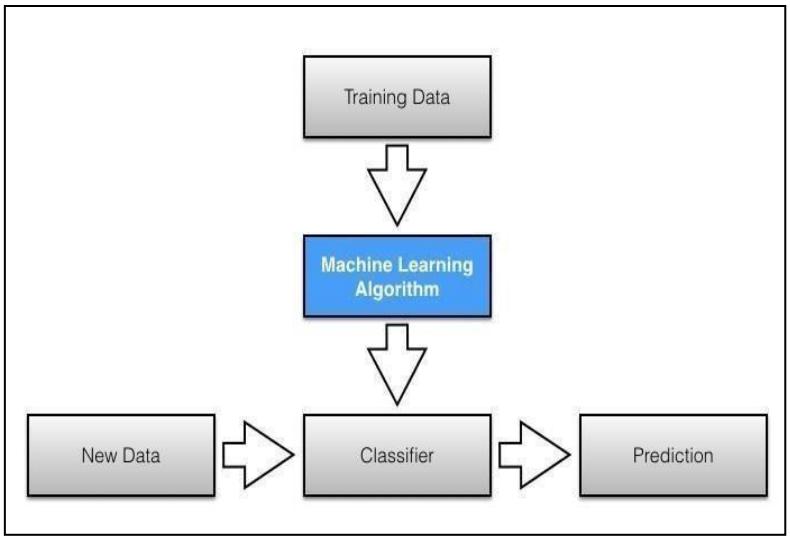


Fig 2.1 ML Architecture

Advantages of ML:

- It gives fast and real time predictions to problems.
- Efficiently utilizes the resources.
- Helps in automation of different tasks.
- It is used a lot in different sectors of life like business, medicine, sports etc.
- Helps interpret previous behavior of model.

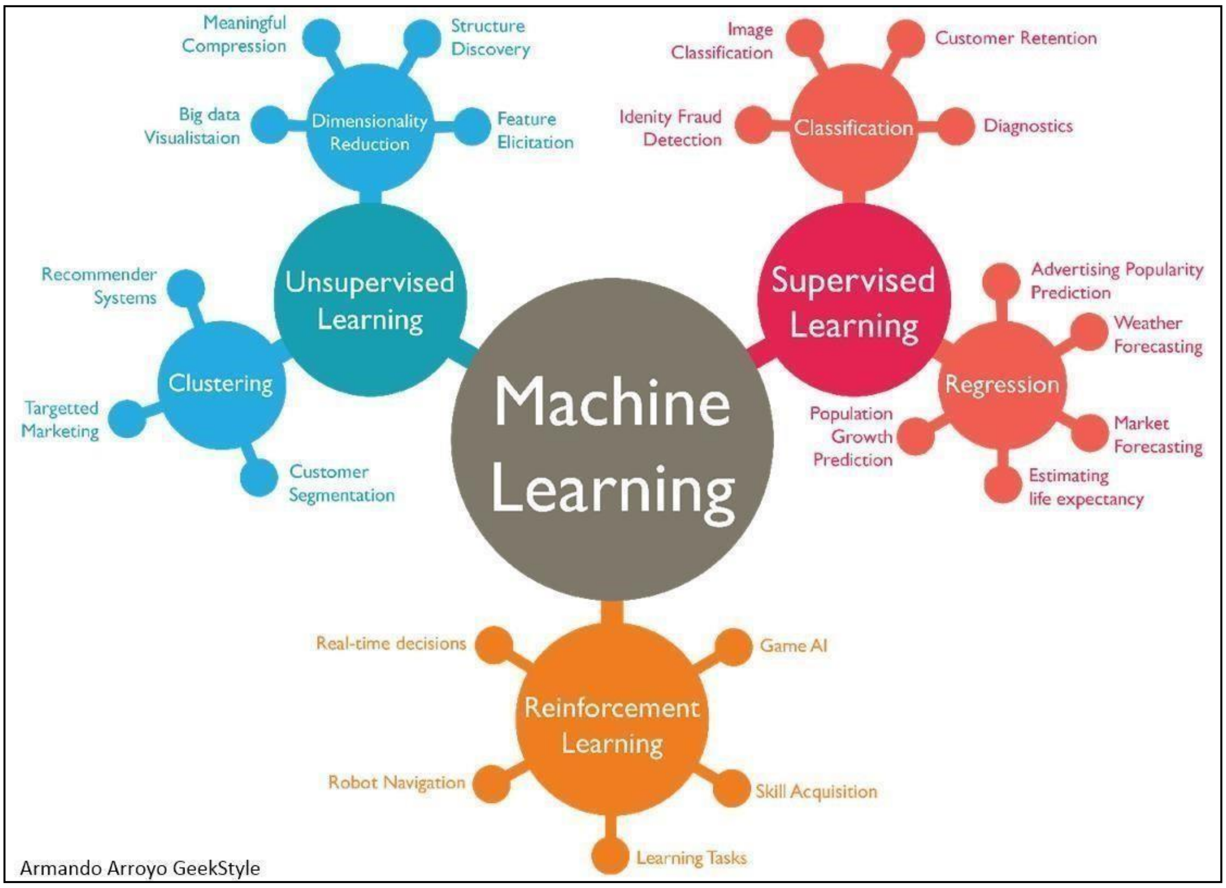


Fig 2.2 Places ML is used

2.2 EXISTING SYSTEM

Traditionally, the diagnosis of breast cancer and the classification of the cancer as malignant or benign was done by various medical procedures like:

- **Breast exam –** The doctor would check the breasts and lymph nodes in the armpits to check if there are any lumps or abnormalities.
- **Mammogram –** These are like X-ray of the breast. They are used to check whether there is breast cancer or not. If any issues are found, the doctor may ask the patient to take a diagnostic mammogram to check for further abnormalities.
- **Breast ultrasound -** Ultrasound uses sound waves to produce images to determine whether a new breast lump is a solid mass or a fluid-filled cyst.
- **Removing a sample of breast cells for testing (biopsy) –** This is probably the only definite way of checking if a patient has breast cancer. The doctor uses a specialized needle device guided by the X-ray or any other test to take samples of tissues from the area to be checked.
- **MRI of the breasts -** An MRI machine uses a magnet and radio waves to create pictures to see the interiors of the breast tissues.

Blood tests, CT scans and PET scans are also done to check for breast cancer.

Disadvantages:

- Time consuming.
- Not completely accurate.
- Very expensive.

2.3 PROPOSED SYSTEM

In the proposed system we plan on using existing data of breast cancer patients which has been collected for a number of years and run different machine learning algorithms on them. These algorithms will analyze the data from the datasets to predict whether the patient has breast cancer or not and it will also tell us if the cancer is malignant or benign.

It is done by taking the patient's data and mapping it with the dataset and checking whether there are any patterns found with the data. If a patient has breast cancer, then instead of taking more tests to check whether the cancer is malignant or benign, ML can be used to predict the case based on the huge amount of data on breast cancer. This proposed system helps the patients as it reduces the amount of money they need to spend just for the diagnosis.

Also, if the tumor is benign, then it is not cancerous, and the patient doesn't need to go through any of the other tests. This saves a lot of time as well.

Advantages:

- Reduces costs for medical tests.
- Does not take huge amount of time.
- Accurate.
- Intelligent way of using available data.

MODULE DESCRIPTION

- **DATASET**

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset which can be found in

University of California, Irvin’s Machine Learning Dataset Repository will be used for this project. All ML algorithms will be performed on this dataset. The features which are in the dataset were computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe different characteristics of the cell nuclei found in the image. There are 569 data points in the dataset: 212 for Malignant and 357 for Benign.

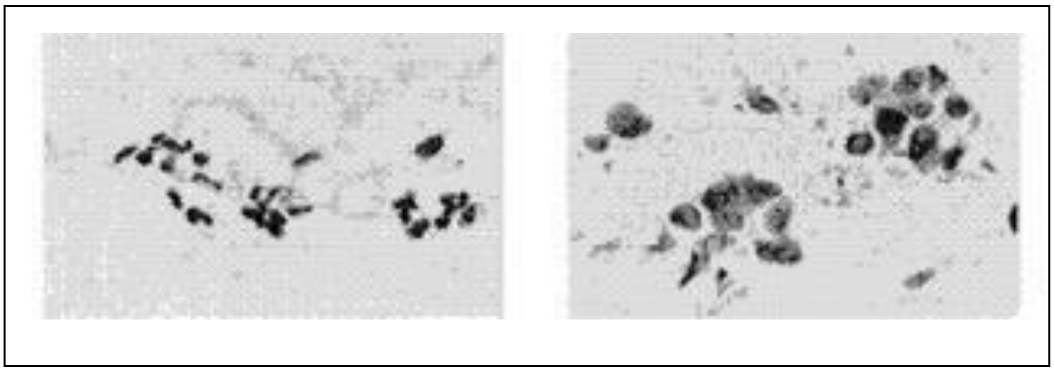


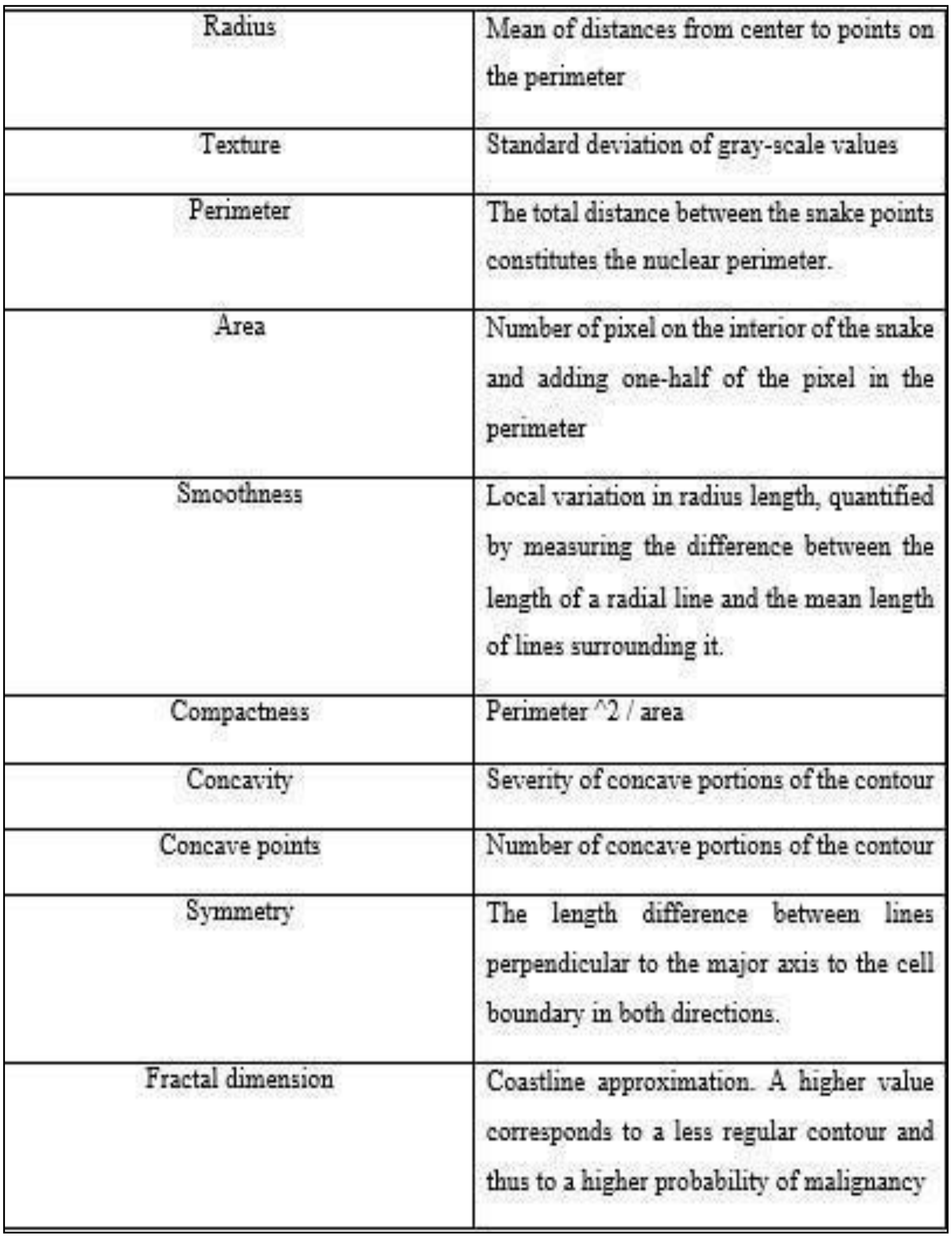
Fig 2.3 Digitized images of FNA (a) Benign (b) Malignant

The dataset contains these features:

- radius
- texture
- perimeter
- area
- smoothness
- compactness
- concavity
- concave points
- symmetry

- fractal dimension

Each of these features have information on mean, standard error, and “worst” or largest (mean of the three largest values) computed. Hence, the dataset has a total of 30 features.



Radius	Mean of distances from center to points on the perimeter
Texture	Standard deviation of gray-scale values
Perimeter	The total distance between the snake points constitutes the nuclear perimeter.
Area	Number of pixel on the interior of the snake and adding one-half of the pixel in the perimeter
Smoothness	Local variation in radius length, quantified by measuring the difference between the length of a radial line and the mean length of lines surrounding it.
Compactness	Perimeter ^2 / area
Concavity	Severity of concave portions of the contour
Concave points	Number of concave portions of the contour
Symmetry	The length difference between lines perpendicular to the major axis to the cell boundary in both directions.
Fractal dimension	Coastline approximation. A higher value corresponds to a less regular contour and thus to a higher probability of malignancy

Table 2.1 Description of features used in the dataset

- **ARTIFICIAL NEURAL NETWORKS**

Artificial neural networks is a very important tool used in machine learning. This technique, as the name suggests, is inspired by the brain and its activities. They were designed in a way to replicate the way humans learn. Neural networks normally consist of input as well as output layers, and sometimes a hidden layer consisting of units that change the input into something that the output layer can use. These tools help in finding different patterns which are too hard for a human to find himself. A programmer programs the machine to recognize it.

- **ML ALGORITHMS**

1. DECISION TREES

This algorithm is used to predict the value of an output or target variable based on many input variables. They are a collection of divide and conquer problem solving strategies. It takes the shape of a tree like structure. It starts with root nodes and this splits into sub-nodes or child nodes. These branches keep spitting until the outcome isn't reached. It is used mainly for classification problems.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.

- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

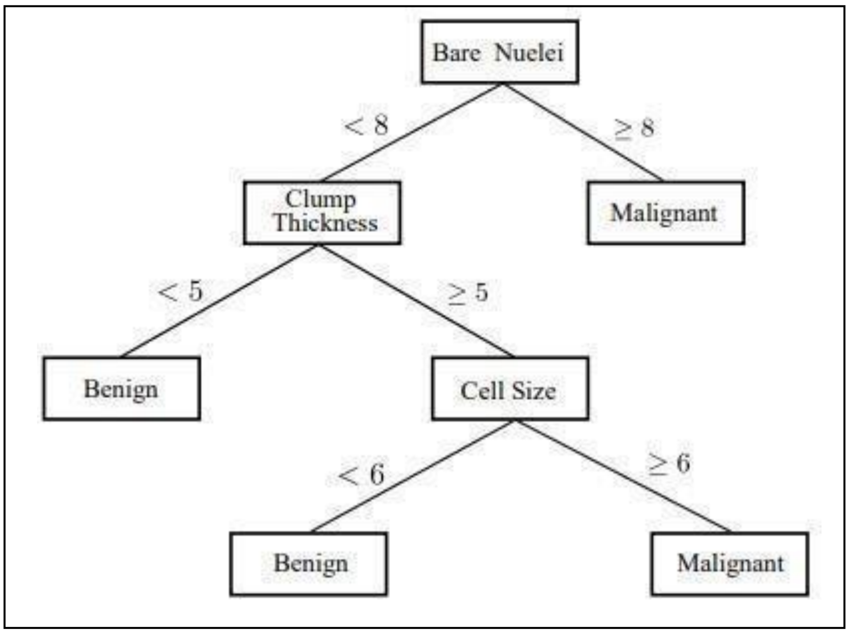


Fig 2.4 Decision Trees

2. K-NEAREST NEIGHBOUR

This algorithm is one of the simplest Machine learning techniques. It is a lazy learning algorithm used for regression and classification. It classifies the objects using their "k" nearest neighbors. k-NN only considers the neighbors around the object, not the underlying data distribution. If k = 1, it basically assigns the unknown to the class of the nearest neighbor. If k > 1, the classification is decided by majority vote based on the k nearest neighbor prediction result.

Both for classification and regression, a useful technique can be used to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.

When KNN is used for classification, the output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes is taken as the prediction.

If you are using K and you have an even number of classes (e.g. 2) it is a good idea to choose a K value with an odd number to avoid a tie. And the inverse, use an even number for K when you have an odd number of classes.

Ties can be broken consistently by expanding K by 1 and looking at the class of the next most similar instance in the training dataset.

Here are some things to keep in mind:

- As we decrease the value of K to 1, our predictions become less stable.
- Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.

- In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

Advantages:

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

Disadvantage

- The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

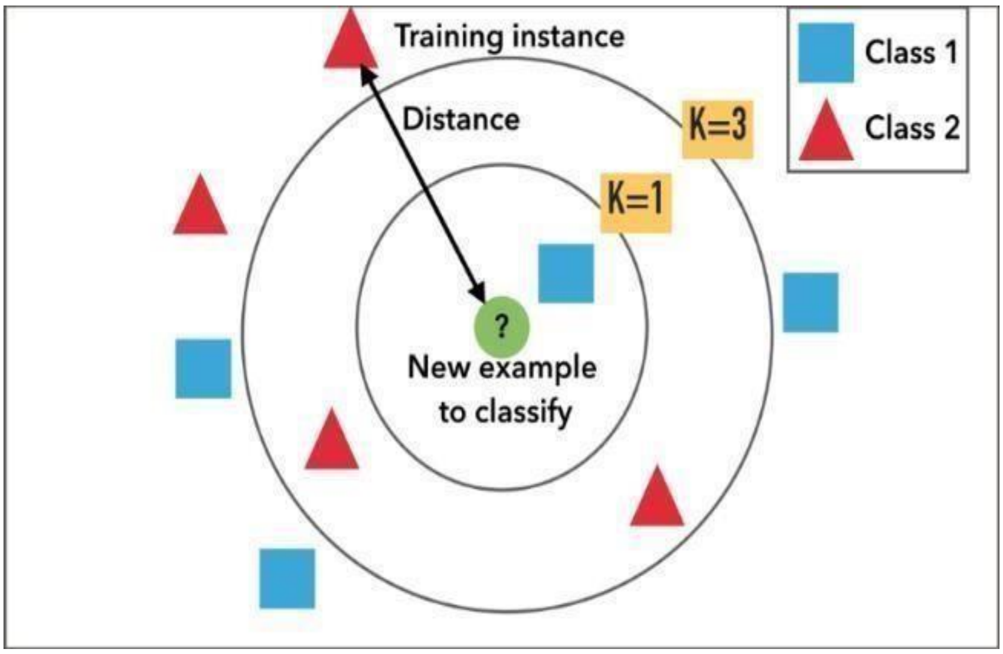


Fig 2.5 K-NN

3. NAÏVE BAYES

Naive Bayes classifier is based on Bayes' theorem and is one of the oldest approaches for classification problems. The formula is:

$$\boxed{P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}}$$

The objective here is to determine the likelihood of an event A happening given B happens. The naive Bayes classifier combines Bayes' model with decision rules like the hypothesis which is the most probable outcomes. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. It was initially introduced for text categorization tasks and still is used as a benchmark.

Pros:

- It is easy and fast to predict class of test data set. It also performs well in multi class prediction
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It performs well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Applications of Naive Bayes Algorithms

- **Real time Prediction:** Naive Bayes is an eager learning classifier and it is sure fast. Thus, it could be used for making predictions in real time.
- **Multi class Prediction:** This algorithm is also well known for multi class prediction feature. Here we can predict the probability of multiple classes of target variable.

- **Text classification/ Spam Filtering/ Sentiment Analysis:** Naive Bayes classifiers mostly used in text classification (due to better result in multi class problems and independence rule) have higher success rate as compared to other algorithms. As a result, it is widely used in Spam filtering (identify spam e-mail) and Sentiment Analysis (in social media analysis, to identify positive and negative customer sentiments)
- **Recommendation System:** Naive Bayes Classifier and Collaborative Filtering together builds a Recommendation System that uses machine learning and data mining techniques to filter unseen information and predict whether a user would like a given resource or not

How to build a basic model using Naive Bayes in Python?

Scikit learn (python library) will help build a Naive Bayes model in Python. There are three types of Naive Bayes model under scikit learn library:

- **Gaussian:** It is used in classification and it assumes that features follow a normal distribution.
- **Multinomial:** It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".
- **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

4. FOREST AND TREE METHODS

Random forests, also known as random decision forests, are a popular ensemble method that can be used to build predictive models for both classification and regression problems. Ensemble methods use multiple learning models to gain better predictive results — in the case of a random forest, the model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer. Random forest aims to reduce correlation issue by choosing only a subsample of the feature space at each split. Essentially, it aims to make the trees de-correlated and prune the trees by setting a stopping criterion for node splits.

- The same **random forest algorithm** or the random forest classifier can use for both classification and the regression task.
- Random forest classifier will **handle the missing** values.
- When we have more trees in the forest, random forest classifier won't **overfit** the model.
    - Can model the random forest classifier for **categorical values** also.

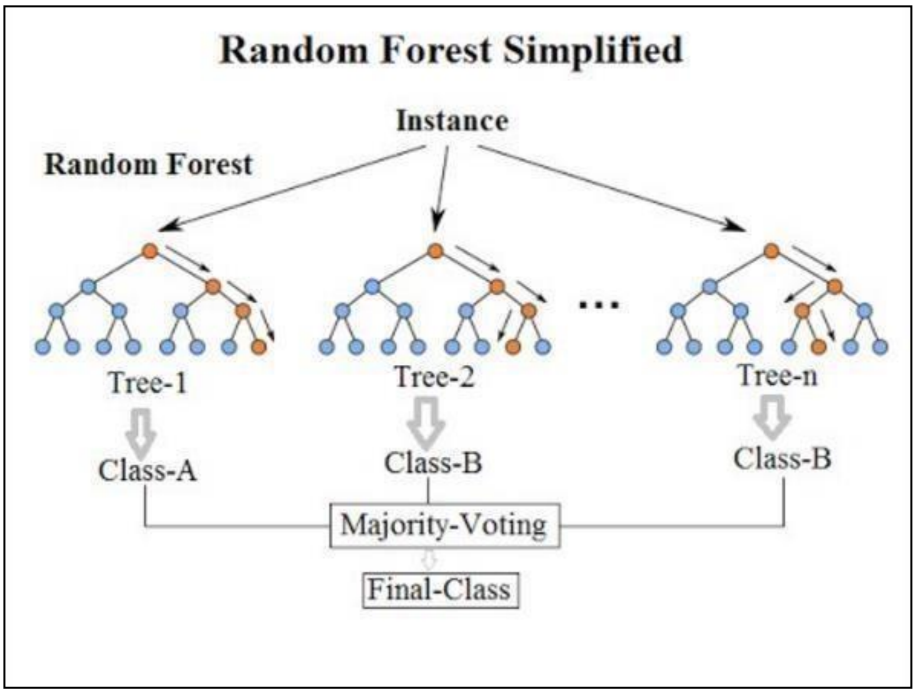


Fig 2.6 Random Forest

SOFTWARE DESCRIPTION

- **PYTHON**

Python is a translated, question situated, abnormal state programming dialect with dynamic semantics. Its abnormal state worked in information structures, joined with dynamic composing and dynamic authoritative, make it exceptionally appealing for Rapid Application Development, and for use as a scripting or paste dialect to interface existing parts together. Python's straightforward, simple to learn sentence structure accentuates intelligibility and subsequently decreases the expense of program upkeep. Python underpins modules and bundles, which energizes program seclusion and code reuse. The Python mediator and the broad standard library are accessible in source or parallel frame without charge for every single significant stage and can be openly disseminated.

- **R PROGRAMMING**

R is a dialect and condition for measurable registering and illustrations. R gives a wide assortment of measurable (straight and nonlinear displaying, established factual tests, time arrangement investigation, characterization, bunching) and graphical procedures, and is profoundly extensible. The S dialect is frequently the vehicle of decision for research in factual philosophy, and R gives an Open Source course to support in that action.

R's quality is the straightforwardness with which all around structured distribution quality plots can be created, including numerical images and formulae where required. Extraordinary consideration has been assumed control over the defaults for the minor structure decisions in illustrations, yet the client holds full control.

- **SCIKIT-LEARN**

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012. As of 2018, scikit-learn is under active development.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 METHODOLOGY FOLLOWED

The steps followed to do this project are:

1. Collection of dataset.
2. Understanding features of dataset.
3. Pre-processing the data.
4. Split data into training dataset and testing dataset.
5. Apply ML algorithms to dataset to predict the breast cancer.
6. Improving results

3.2 FUNCTIONAL AND NON FUNCTIONALREQUIREMENTS

FUNCTIONAL:

The functional requirement define the system or the components of the system. A function is basically inputs, behaviors and outputs. Stuff that can be called functional requirements are: calculations, technical details, data manipulation and processing. It tells us what a system is supposed to do.

Here, the system has to perform the following tasks:

- Understand all the features as well as the data provided in the dataset.

- Map the data in the dataset with the given input data. Find patterns, if any, with both the dataset as well as input data.
- Check whether the input data of a patient will result in the diagnosis of breast cancer or not.
- If breast cancer is diagnosed, provide information on the type of breast cancer, i.e, benign or malignant.
- Provide the percentage accuracy of the proposed prediction.

NON FUNCTIONAL:

A non-functional requirement is a requirement gives the criteria that can be used to judge how well a system can function. It comes under system/requirements engineering. It gives a judgement on the overall unlike functional requirements which define specific behavior or functions. Functional requirements are implemented by using the system design whereas system architecture is what is used for implementing the non-functional requirements.

Non-functional requirements are also called constraints.

Some of the quality attributes are as follows:

3.2.1 ACCESSIBILITY:

Accessibility is a term that is used to describe if a product or software is accessible to the public and how easily can it be accessed.

It is easy to access as the dataset is open source and can be found on the University of California, Irvin’s ML dataset repository. Unlike breast cancer diagnosis tests in hospitals which cost a lot, anyone can access this dataset for free.

3.2.2 MAINTAINABILITY:

Maintainability tells us how easily a software or tool or system can be modified in order to:

- Correct defects

- Meet new requirements

Different programming languages can be used to make the predictive model based on the programmer’s wishes. The datasets can also be modified and new data can be added as and when the data is updated by doctors. Different ML algorithms can also be used to check which algorithm will give the best result.

As python and R are both programming languages that can adapt to new changes easily, it is easy to maintain this type of system.

3.2.3 SCALABILITY:

The system can work normally under situations such as low bandwidth and huge datasets. The

R studio as well as Excel can take care of these data and can perform the algorithms with ease.

3.2.4 PORTABILITY:

Portability is a feature which tells us about the ease at which we can reuse an existing piece of code when we move from one location or environment to some other.

This system uses python and R programming languages and they can be executed under different operation conditions provided it meet its minimum configurations. Only system files and dependant assemblies would have to be configured in such case.

3.3 HARDWARE REQUIREMENTS

Processor	: Any Processor above 500 MHz
RAM	: 512Mb
Hard Disk	: 10 GB
Input device	: Standard Keyboard and Mouse

Output device	: VGA and High Resolution Monitor

3.4 SOFTWARE REQUIREMENTS

- Operating system : Windows 10
- IDE : R studio
- Microsoft Excel with Data Analysis
- Rattle
- Anaconda with Python3
- Spyder
- Jupyter Notebook
- Scikit-learn library

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

Under our model, the goal of our project is to create a design to achieve the following:

4.1.1 ACCURACY

Only accurate outcomes can help make this model a good one. It can be reliable only when all the outcomes are correct and can be trusted. As this data is required for healthcare purposes, it is important that no errors occur.

4.1.2 EFFICIENCY

The model should be efficient as there is no requirement of manual data entry work or any work by doctors. It takes less time to predict outcomes after all the ML algorithms have been used on the data.

4.2 SYSTEM ARCHITECTURE

As this project does not have any UI, the architecture is basically the dataset and the features of the dataset. It is trying to understand the dataset and try making the system as simple and easy as possible.

The dataset is first split into training and testing set. The training set if first exposed to the machine learning algorithms so that the system understands what data gives what type of outcome.

After the system is trained, the testing data is used to test whether the system can correctly predict the class of the data. It checks the percentage accuracy of the model.

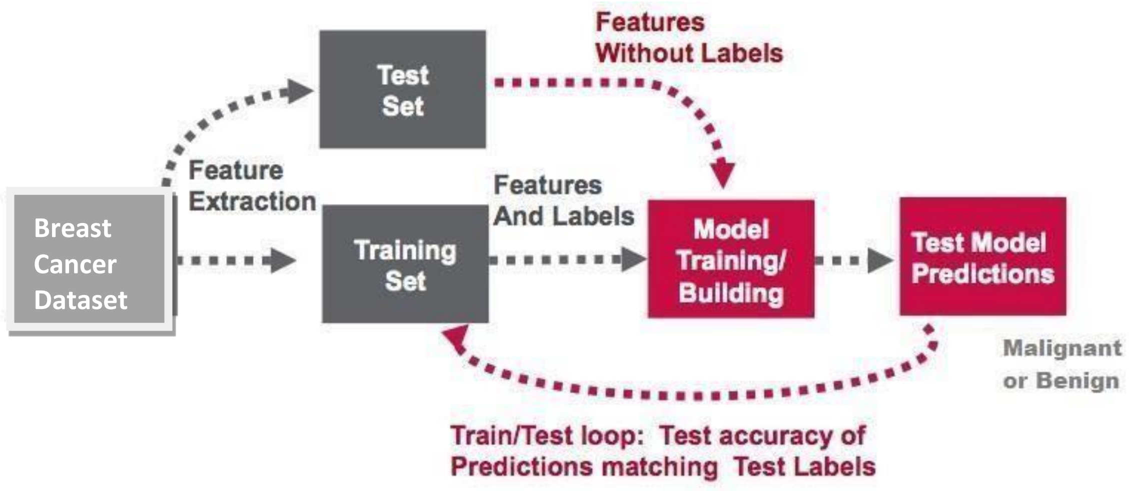


Fig 4.1 System Architecture

4.3 DATA FLOW DIAGRAM

The dataflow diagram shows the way in which the data from the dataset moves.

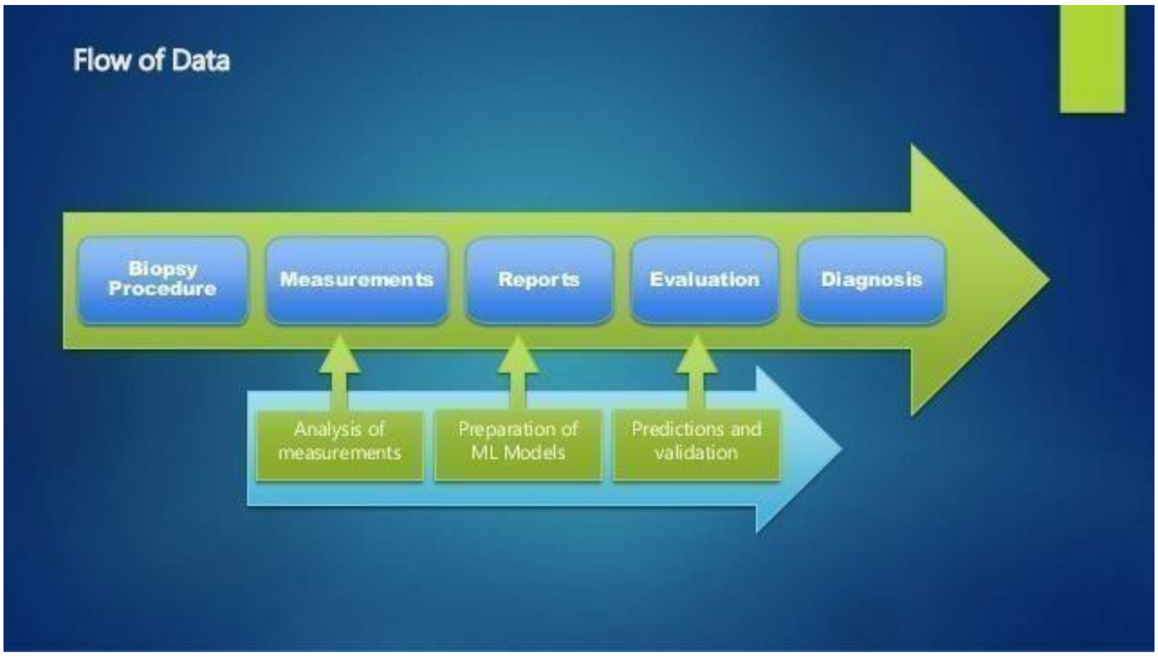


Fig 4.2 Data Flow Diagram

4.4 EXPECTED OUTCOME

The outcome of this model is to correctly check and predict whether a patient has breast cancer or not. If yes, the model should also be able to tell if the patient has malignant or benign type of cancer.

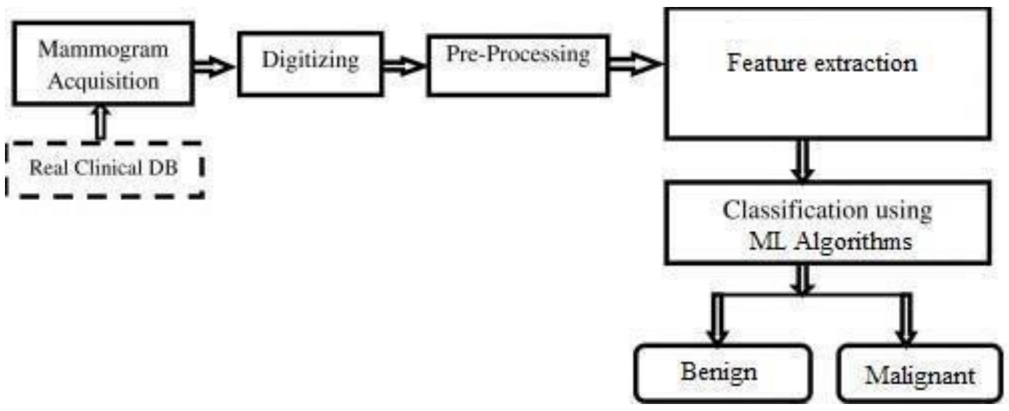


Fig 4.3 Expected Outcome

CHAPTER 5

IMPLEMENTAION AND OUTPUT

5.1 PREPARING THE DATA

Step 1: The first step in the machine learning process is to prepare the data. This includes importing all the packages that will help us organize and visualize the data. The packages used are as follows:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2: After importing all the necessary packages, we need to load the dataset. We use the help of Pandas to load the data set.

```
data=pd.read_csv('../input/data.csv');
```

Step 3: We need to drop the first column of the dataset which consists of IDs as this field will not help us in the classification process. This is done as follows:

```
data.drop(data.columns[[-1, 0]],axis=1, inplace=True)
```

Step 4: To check how many data points are Malignant and benign.

```
diagnosis_all = list(data.shape)[0]
diagnosis_categories = list(data['diagnosis'].value_counts())


print("\n \t The data has {} diagnosis, {} malignant and {} benign.".format(diagnosis_all, diagn
osis_categories[0], diagnosis_categories[1]))
```

The data has 569 diagnosis, 357 malignant and 212 benign.

5.2 VISUALIZING THE DATA

We need to build visualizations of the data in order to decide how to proceed with the machine learning tools. The Seaborn and the Matplotlib packages will be used for this purpose. We use the mean values of the features. So first we will have to separate those features in the list to make some work easier and the code more readable.

```
features_mean= list(data.columns[1:11])
```

The first method that can be used for visualization is heat map. A heat map is a two-dimensional representation of data in which values are represented by colors. A simple heat map provides an immediate visual summary of information. More elaborate heat maps allow the viewer to understand complex data sets.

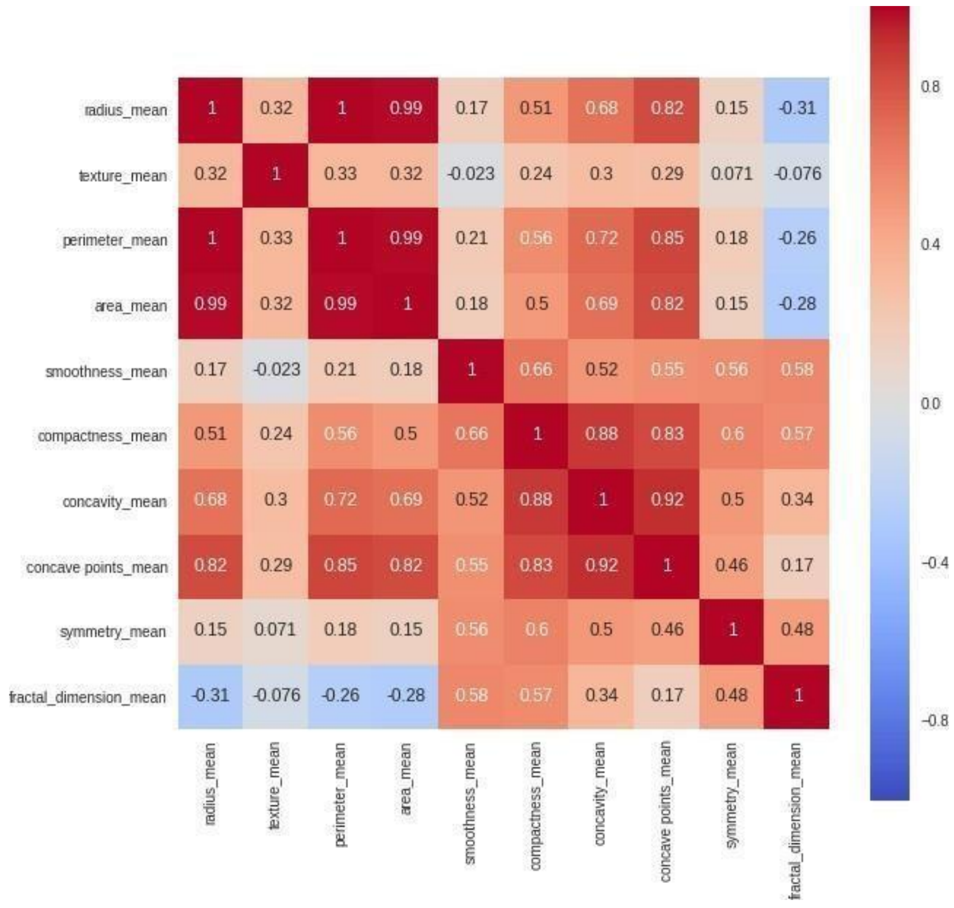


Fig 5.1 Heat Map

We can also see how the malignant or benign tumors cells can have (or not) different values for the features plotting the distribution of each type of diagnosis for each of the mean features.

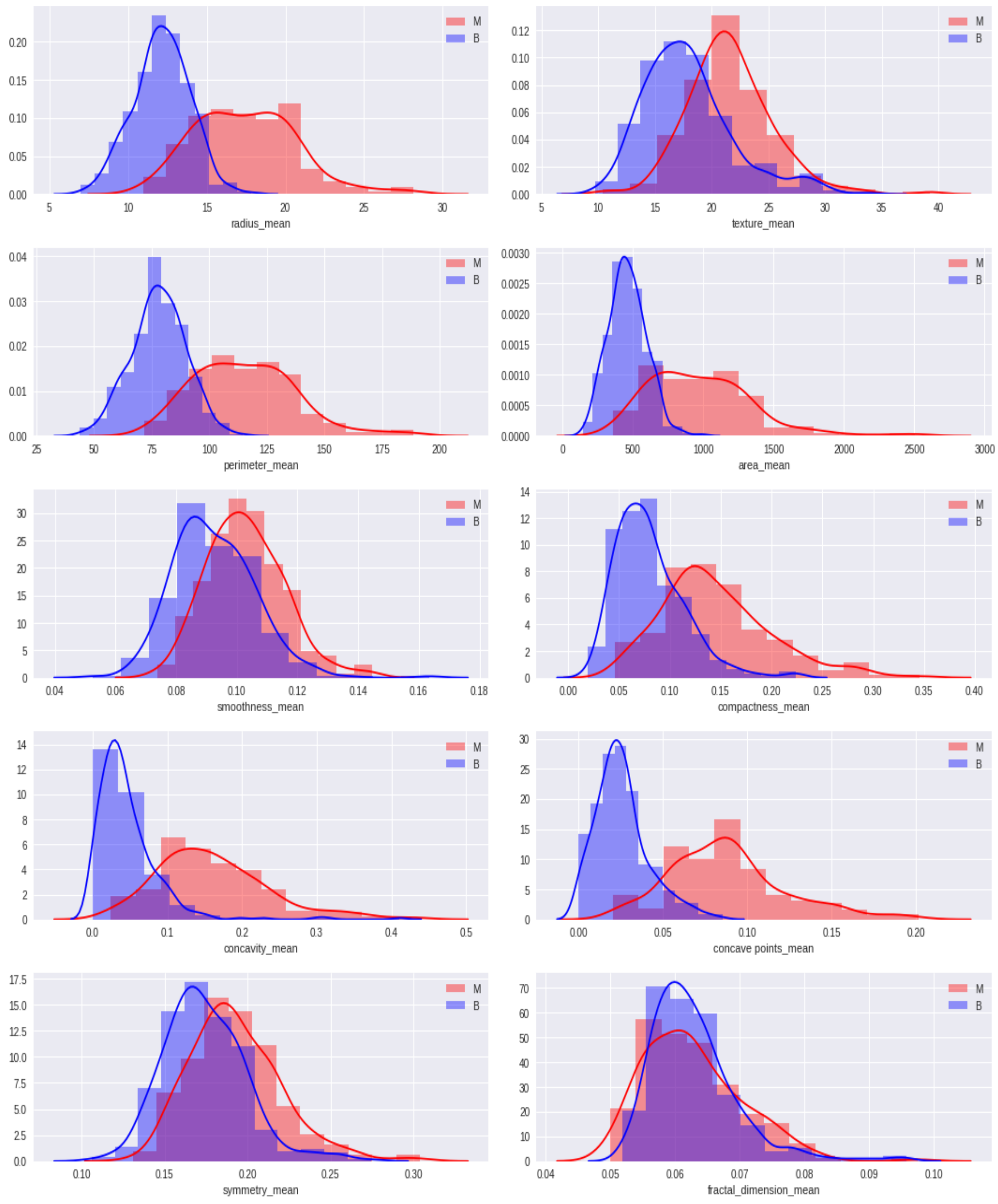


Fig 5.2 Feature Plotting

We can make use of box plots for visualization. In descriptive statistics, a box plot or boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot and box-and-whisker diagram. Outliers may be plotted as individual points. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. The spacings between the different parts of the box indicate the degree of dispersion (spread) and skewness in the data and showoutliers.

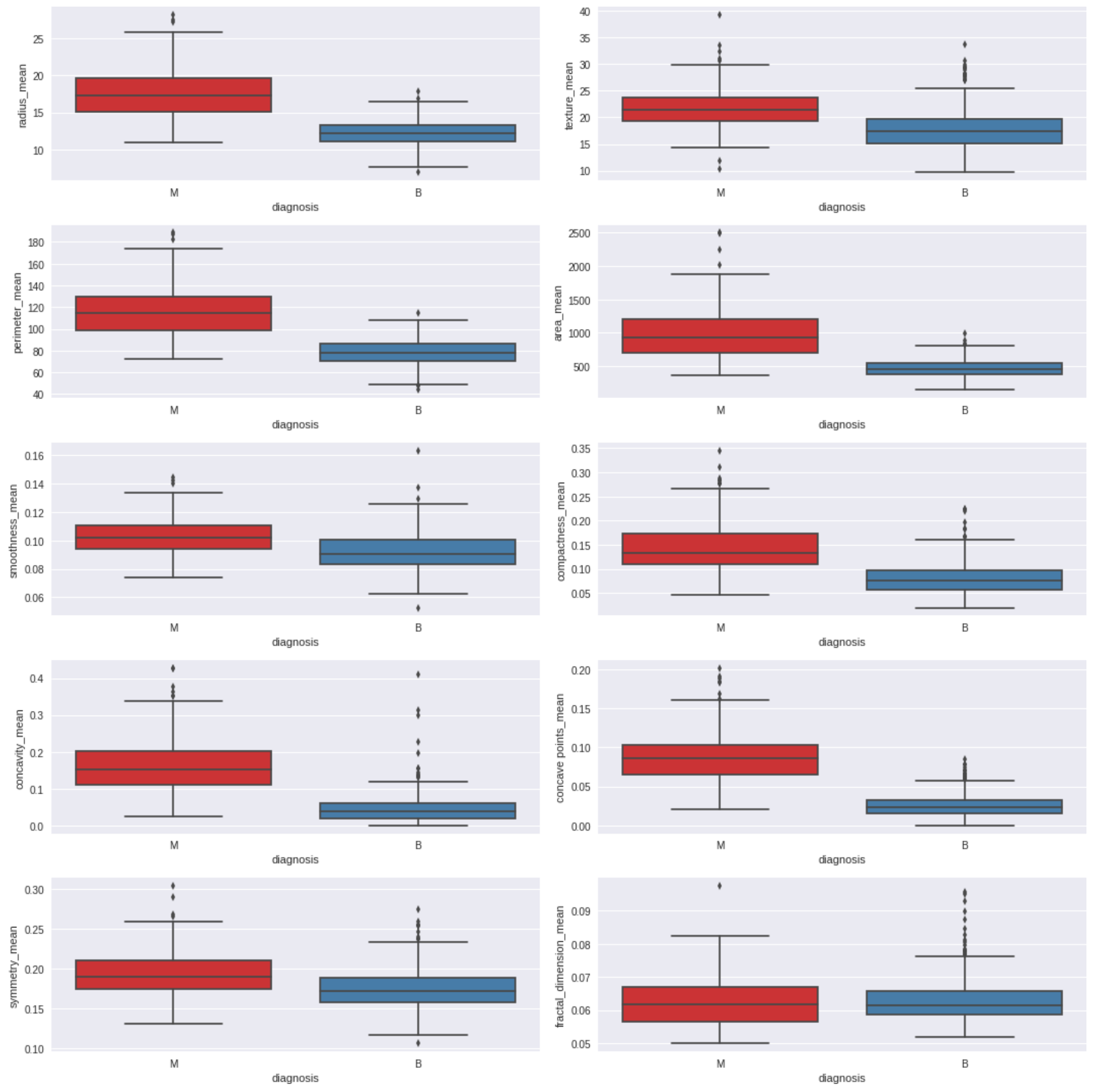


Fig 5.3 Box plots

5.3 APPLYING THE ML ALGORITHMS

1. K-NEAREST NEIGHBORS

Steps to use in R Studio:

a. Convert your data set to .csv file

b. Import dataset into R studio using:

c. Remove the unnecessary columns or columns with nominal data.

d. Normalize the data as all data use different scales.

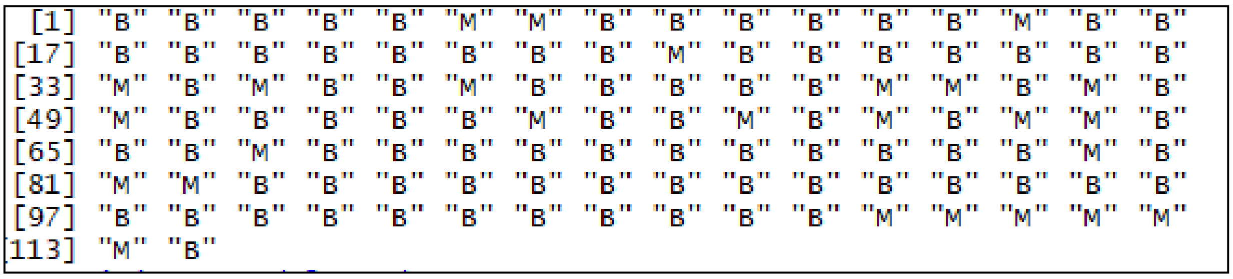
e. Split data into training and testing data (70-30).

f. Calculate K value.

g. Using the training data and KNN algorithm, get the predicted values for the testing data.

h. Compare the predicted values to actual values and calculate accuracy of model.

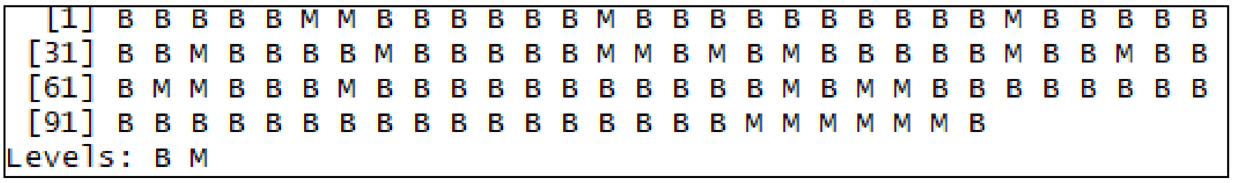
The data from the dataset looks like this:



```
  [1] "B" "B" "B" "B" "B" "M" "M" "B" "B" "B" "B" "B" "B" "M" "B" "B"
 [17] "B" "B" "B" "B" "B" "B" "B" "B" "M" "B" "B" "B" "B" "B" "B" "B"
 [33] "M" "B" "M" "B" "B" "M" "B" "B" "B" "B" "B" "M" "M" "B" "M" "B"
 [49] "M" "B" "B" "B" "B" "B" "M" "B" "B" "M" "B" "M" "B" "M" "M" "B"
 [65] "B" "B" "M" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "M" "B"
 [81] "M" "M" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B"
 [97] "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "B" "M" "M" "M" "M" "M"
[113] "M" "B"
```

Fig 5.4 KNN Actual Data

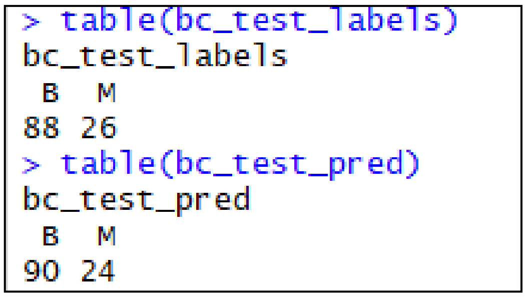
The data after using the algorithm on the testing set of data looks like this:



```
  [1] B B B B B M M B B B B B B M B B B B B B B B B B M B B B B B
 [31] B B M B B B B M B B B B B M M B M B M B B B B B M B B M B B
 [61] B M M B B B M B B B B B B B B B B B M B M M B B B B B B B B
 [91] B B B B B B B B B B B B B B B B B M M M M M M B
Levels: B M
```

Fig 5.5 KNN Predicted Data

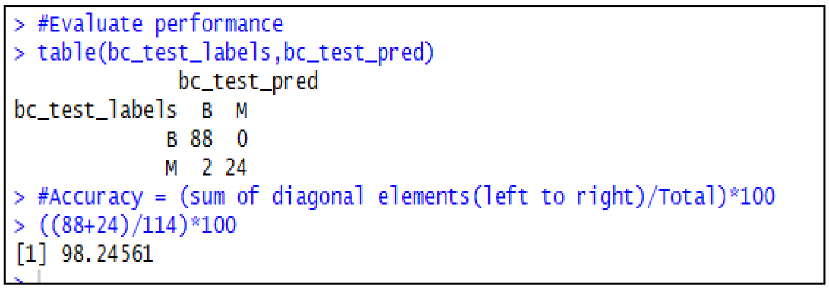
Data previously and predicted:



```
> table(bc_test_labels)
bc_test_labels
 B  M
88 26
> table(bc_test_pred)
bc_test_pred
 B  M
90 24
```

Fig 5.6 Previous and predicted data

Confusion matrix between the data and accuracy:



```
> #Evaluate performance
> table(bc_test_labels,bc_test_pred)
              bc_test_pred
bc_test_labels  B  M
             B 88  0
             M  2 24
> #Accuracy = (sum of diagonal elements(left to right)/Total)*100
> ((88+24)/114)*100
[1] 98.24561
```

Fig 5.7 Confusion matrix and accuracy

Steps to use in Python:

Using all mean values features from the dataset and the scikit-learn libraries, we can run the code to find the accuracy in prediction of breast cancer data.

```
from sklearn.neighbors import KNeighborsClassifier
start = time.time()
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))
```

Accuracy: 92.11%

Execution time: 0.020711 seconds

2. NAÏVE BAYES

Steps to use in Python:

Loading the dataset:

Scikit-learn comes with a few small standard datasets that do not require downloading any file from any external website. The dataset includes several data about the breast cancer tumors along with the classification labels, viz., malignant or benign. It can be loaded using the following function:

The data set has 569 instances or data of 569 tumors and includes data on 30 attributes or features like the radius, texture, perimeter, area, etc. of a tumor. We will be using these features to train our model.

Installing the necessary modules:

For this machine learning project, we will be needing the 'Scikit-learn' Python module by running the following command in the command prompt:

```
pip install scikit-learn
```

Step #1: Importing the necessary module and dataset.

We will be needing the 'Scikit-learn' module and the Breast cancer wisconsin (diagnostic) dataset.

Step #2: Loading the dataset to a variable.

The important attributes that we must consider from that dataset are 'target-names'(the meaning of the labels), 'target'(the classification labels), 'feature_names'(the meaning of the features) and 'data'(the data to learn).

Step #3: Organizing the data and looking at it

To get a better understanding of what the dataset contains and how we can use the data to train our model, let us first organize the data and then see what it contains by using the print() function.

Step #4: Organizing the data into Sets.

For testing the accuracy of our classifier, we must test the model on unseen data. So, before building the model, we will split our data into two sets, viz., training set and test set. We will be using the training set to train and evaluate the model and then use the trained model to make predictions on the unseen test set.
The sklearn modlue has a built-in function called the train_test_split(), which automatically divides the data into these sets. We will be using this function two split the data.

The train_test_split() function randomly splits the data using the parameter test_size. What we have done here is that, we have split 33% of the original data into test data (test). The remaining data (train) is the training data. Also, we have respective labels for both the train variables and test variables, i.e. train_labels and test_labels.

Step #5: Building the Model.

There are many machine learning models to choose from. All of them have their own advantages and disadvantages. For this model, we will be using the Naive Bayes algorithm that usually performs well in binary classification tasks. Firstly, import the GaussianNB module and initialize it using the GaussianNB() function. Then train the model by fitting it to the data in the dataset using the fit() method.

Step #6: Evaluating the trained model's accuracy.

As we have predicted values now, we can evaluate our model's accuracy by comparing it with the actual labels of the test set, i.e., comparing predictions with test_labels. For this purpose, we will be using the built-in **accuracy_score()** function in the sklearn module.

Accuracy Output: 0.941489

3. FOREST AND TREE METHODS

Using all mean values features from the dataset and the scikit-learn libraries, we can run the code to find the accuracy in prediction of breast cancer data.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
start = time.time()
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Random Forest Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))

start = time.time()
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_all.append(accuracy_score(prediction, y_test))
cvs_all.append(np.mean(scores))
print("Decision Tree Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: {0:.5} seconds \n".format(end-start))
```

```
Random Forest Accuracy: 93.86%
Execution time: 0.055068 seconds
Decision Tree Accuracy: 92.11%
Execution time: 0.025084 seconds
```

USING THE SELECTED FEATURES

Features are: radius_mean, perimeter_mean, area_mean, concavity_mean, concave points_mean.

```
X = data.loc[:,features_selection]
y = data.loc[:, 'diagnosis']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

accuracy_selection = []
```

```
cvs_selection = []
```

1. K-NEAREST NEIGHBORS

```
from sklearn.neighbors import KNeighborsClassifier
start = time.time()
clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))
```

Accuracy: 92.11%

Execution time: 0.020711 seconds

2. NAÏVE BAYES

```
from sklearn.naive_bayes import GaussianNB
start = time.time()
clf = GaussianNB()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))
```

Accuracy: 94.74%

Execution time: 0.019591 seconds

3. FOREST AND TREE METHODS

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
start = time.time()
clf = RandomForestClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Random Forest Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))
start = time.time()
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
prediction = clf.predict(X_test)
scores = cross_val_score(clf, X, y, cv=5)
end = time.time()
accuracy_selection.append(accuracy_score(prediction, y_test))
cvs_selection.append(np.mean(scores))
print("Decision Tree Accuracy: {0:.2%}".format(accuracy_score(prediction, y_test)))
print("Execution time: %s seconds \n" % "{0:.5}".format(end-start))
```

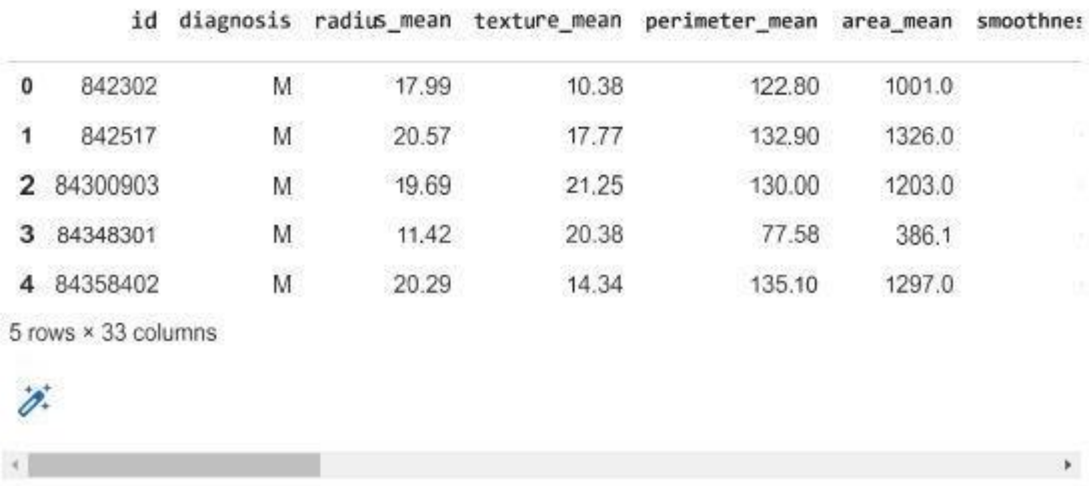
```
Random Forest Accuracy: 92.11%
Execution time: 0.15038 seconds
Decision Tree Accuracy: 88.60%
Execution time: 0.024996 seconds
```

5.4 PROJECT CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('/content/data.csv')
```

```
df.head()
```



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

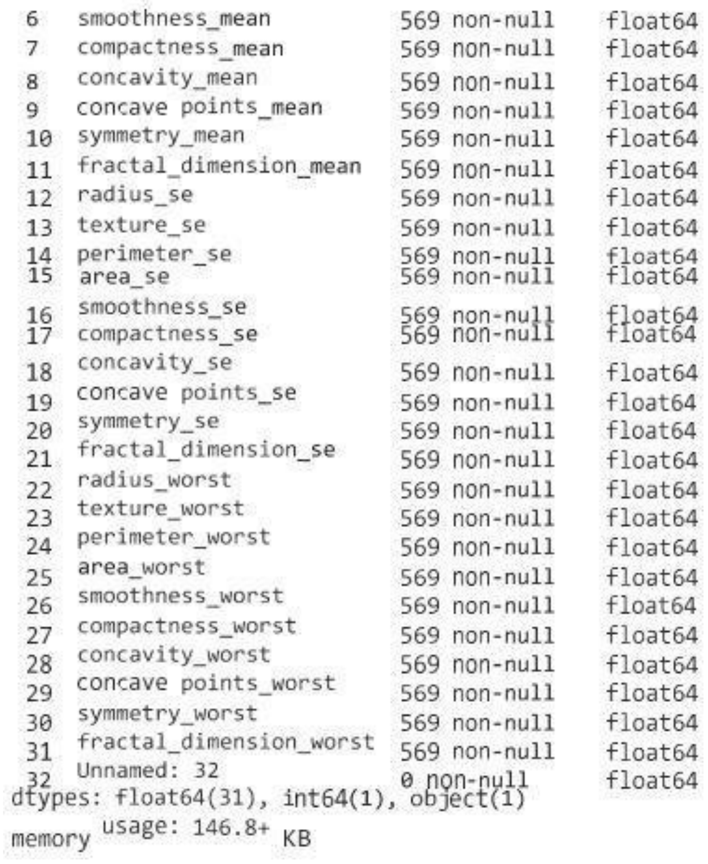
5 rows × 33 columns

```
#EDA
#cheacking the total no of rows and columns
df.shape
```

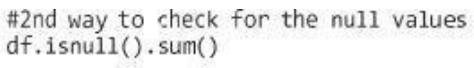
```
(569, 33)
```

```
#checking the columns and their corresponding data types
# the properties of the data = summary statistics
df.info()
```

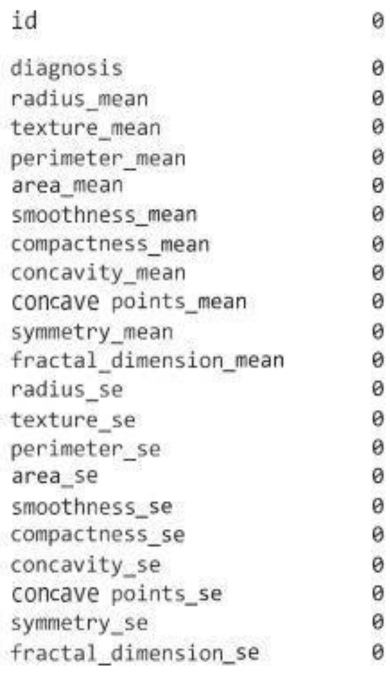
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    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
 32  Unnamed: 32              0 non-null     float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```



```
#2nd way to check for the null values
df.isnull().sum()
```



```
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
```

```
#drop the column with all missing values.
df= df.dropna(axis=1)
```

```
df.shape
```

```
    (569, 32)
```

```
#checking the datatypes
df.dtypes
```

```
    id                          int64
    diagnosis                  object
    radius_mean               float64
    texture_mean              float64
    perimeter_mean            float64
    area_mean                 float64
    smoothness_mean           float64
    compactness_mean          float64
    concavity_mean            float64
    concave points_mean       float64
    symmetry_mean             float64
    fractal_dimension_mean    float64
    radius_se                 float64
    texture_se                float64
    perimeter_se              float64
    area_se                   float64
    smoothness_se             float64
    compactness_se            float64
    concavity_se              float64
    concave points_se         float64
    symmetry_se               float64
    fractal_dimension_se      float64
    radius_worst              float64
    texture_worst             float64
    perimeter_worst           float64
    area_worst                float64
    smoothness_worst          float64
    compactness_worst         float64
    concavity_worst           float64
    concave points_worst      float64
```

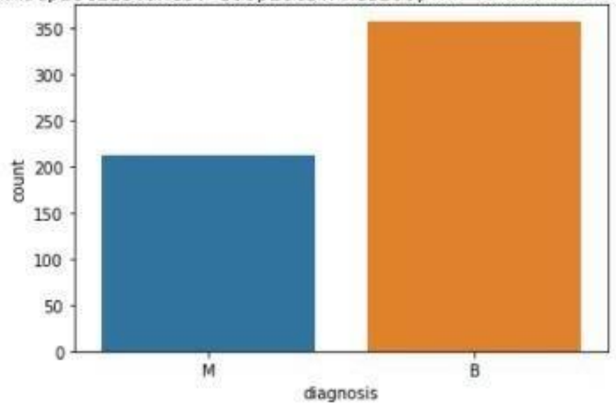
```
symmetry_worst             float64
fractal_dimension_worst    float64
dtype: object
```

```
df['diagnosis'].value_counts()
```

```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
sns.countplot(df['diagnosis'], label="count")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf5b4cdcd0>
```



```
from sklearn.preprocessing import LabelEncoder
labelencoder_Y=LabelEncoder()
```

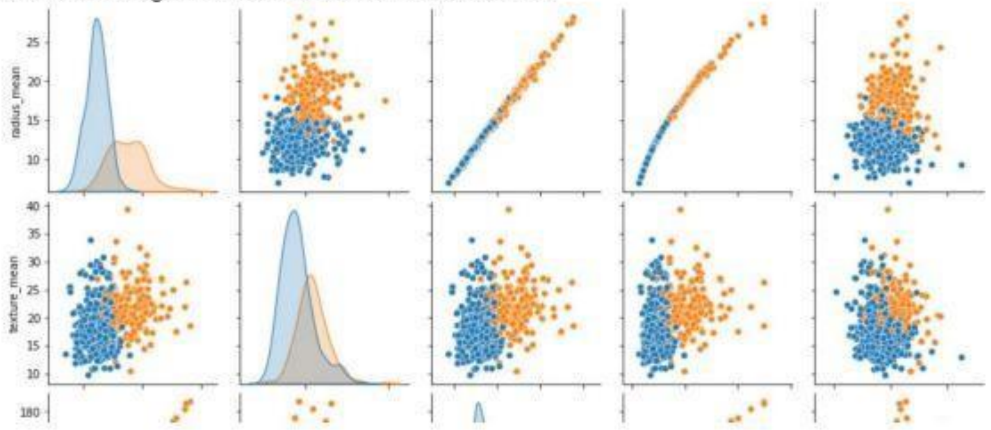
```
#transforming catagorical data to numerical
df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)
```

```
df.iloc[:,1].values
```

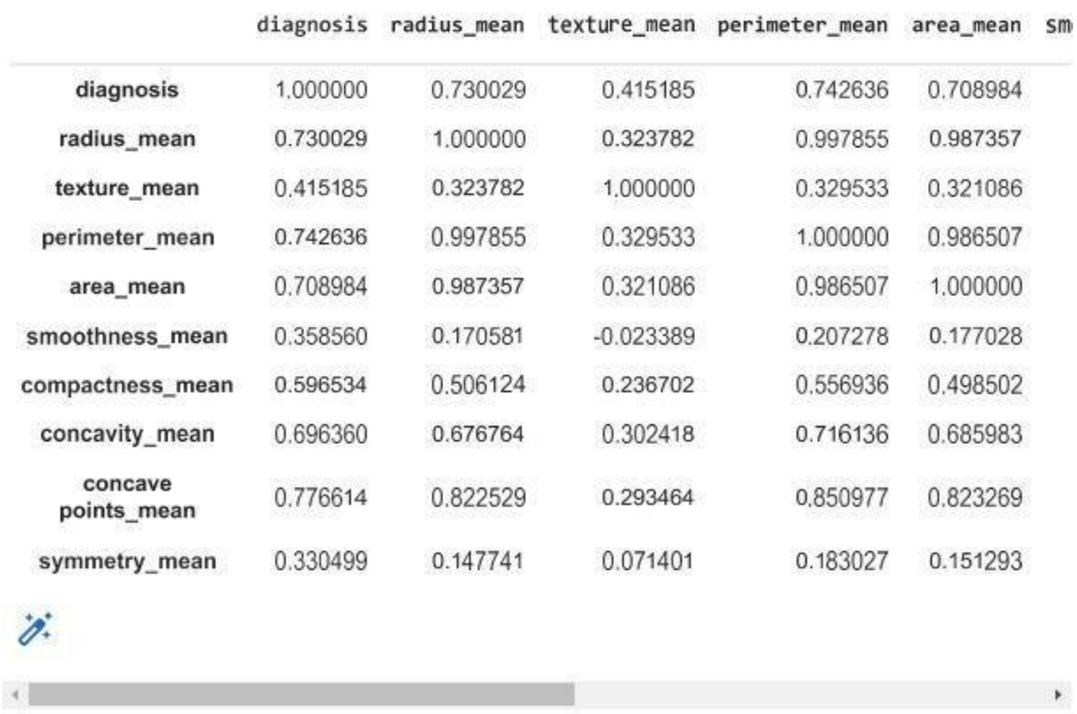
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       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, 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, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 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, 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])
```

```
sns.pairplot(df.iloc[:,1:7], hue="diagnosis")
```

```
<seaborn.axisgrid.PairGrid at 0x7fdf5a8a7550>
```



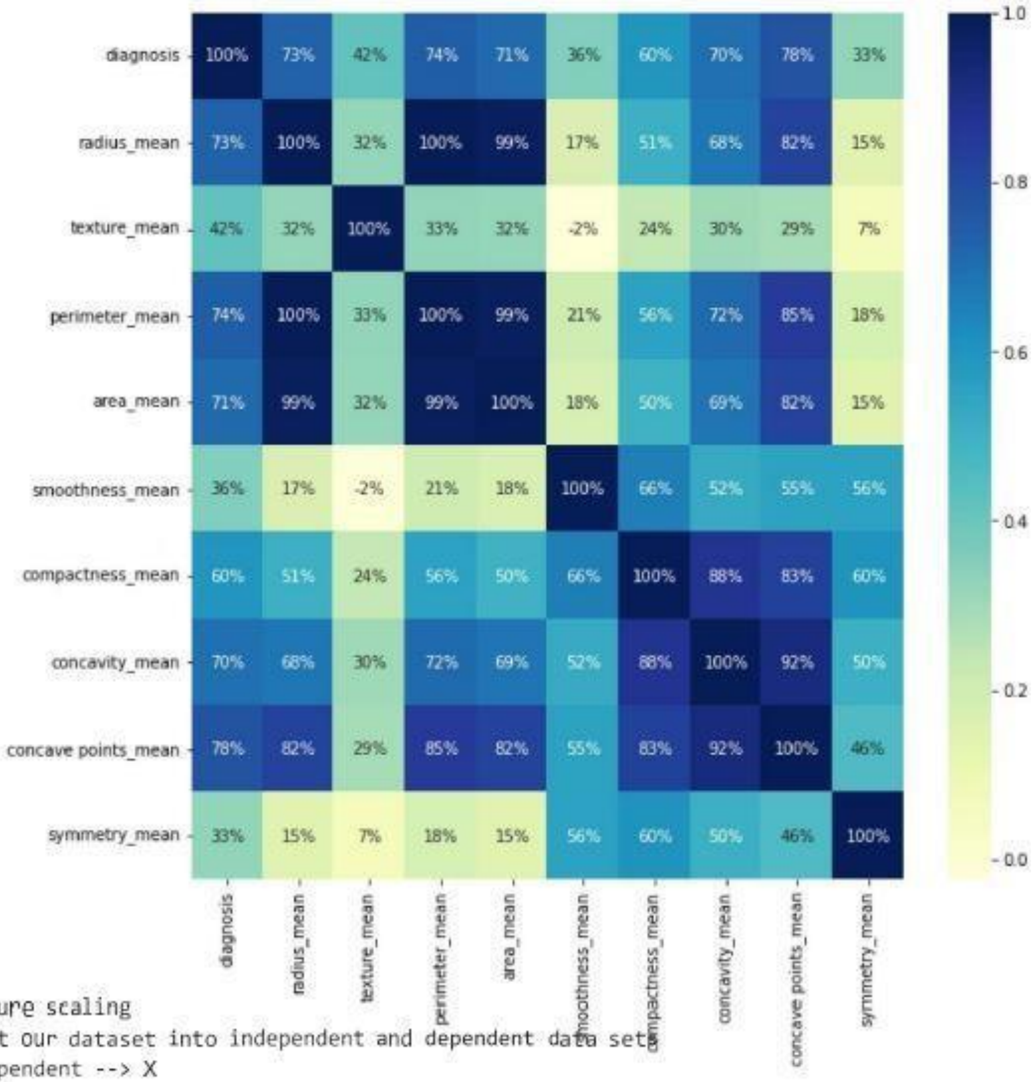
```
#corilation between columns
df.iloc[:,1:11].corr()
```



	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	sm
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	

```
#heatmap
plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:11].corr(),cmap='YlGnBu',annot=True,fmt='.0%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf5b45afd0>
```

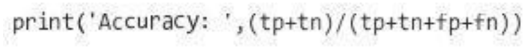


```
#feature scaling
#split our dataset into independent and dependent data sets
#independent --> X
#dependent --> Y
X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values


#80:20 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size =0.20, random_state = 0)


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

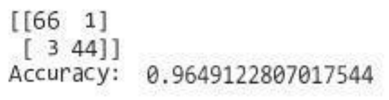
```
X_train

    array([[-1.15036482, -0.39064196, -1.12855021, ..., -0.81232053,
            -0.75798367, -0.01614761],
           [-0.93798972,  0.68051405, -0.94820146, ..., -0.37504806,
            -0.60687023,  0.09669004],
           [ 0.574121  , -1.03333557,  0.51394098, ..., -0.18298917,
            -0.02371948, -0.20050207],
           ...,
           [-1.32422924, -0.20048168, -1.31754581, ..., -0.76769066,
            -0.97974953, -0.71542314],
           [-1.24380987, -0.2245526 , -1.28007609, ..., -1.34136004,
            -1.75401433, -1.58157125],
           [-0.73694129,  1.14989702, -0.71226578, ...,  0.47893704,
            -0.27460457, -1.25895095]])
```

```
def models(X_train,Y_train):
  from sklearn.linear_model import LogisticRegression
  log = LogisticRegression(random_state=0)
  log.fit(X_train,Y_train)

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

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



    #print the accuracy of each model on the training dataset
    print('The accuracy of Logestic Regresion : ',log.score(X_train,Y_train))
    print('The accuracy of Decision Tree : ',tree.score(X_train,Y_train))
    print('The accuracy of Random Forest : ',forest.score(X_train,Y_train))

    return log, tree, forest

  model = models(X_train,Y_train)

      The accuracy of Logestic Regresion :  0.9912087912087912
      The accuracy of Decision Tree :  1.0
      The accuracy of Random Forest :  0.9978021978021978
```

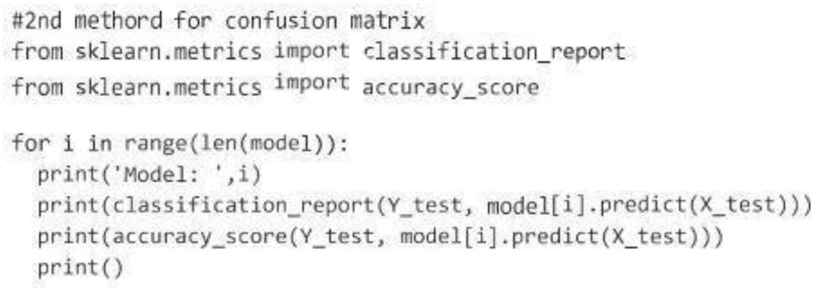
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, model[0].predict(X_test))
tp=cm[0][0]
tn=cm[1][1]
fn=cm[1][0]
fp=cm[0][1]
print(cm)
```



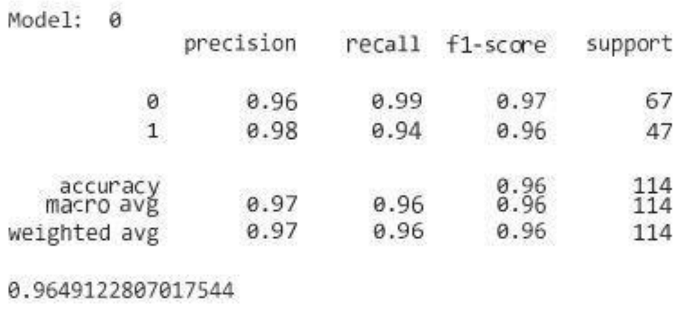
```
print('Accuracy: ',(tp+tn)/(tp+tn+fp+fn))
```



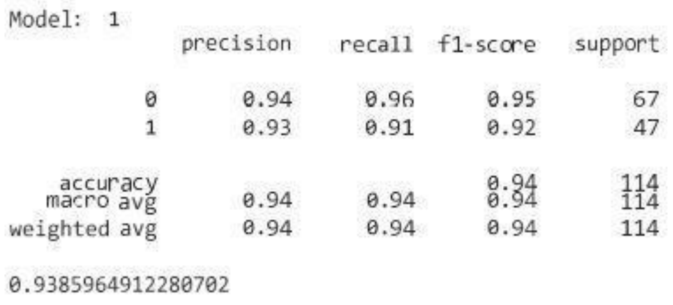
```
[[66  1]
 [ 3 44]]
Accuracy:  0.9649122807017544
```



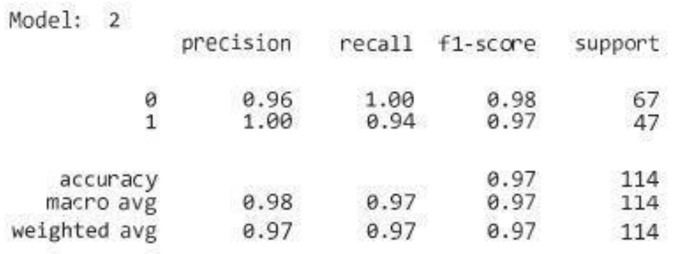
```
#2nd methord for confusion matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

for i in range(len(model)):
  print('Model: ',i)
  print(classification_report(Y_test, model[i].predict(X_test)))
  print(accuracy_score(Y_test, model[i].predict(X_test)))
  print()
```



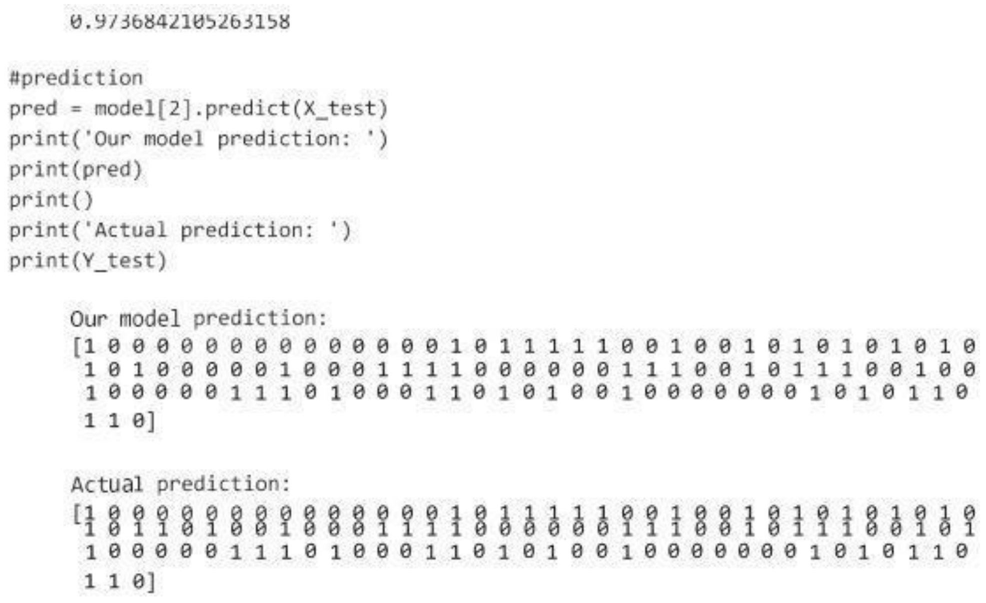
```
Model:  0
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        67
           1       0.98      0.94      0.96        47

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

0.9649122807017544
```



```
Model:  1
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        67
           1       0.93      0.91      0.92        47

    accuracy                           0.94       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.94      0.94      0.94       114

0.9385964912280702
```



```
Model:  2
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        67
           1       1.00      0.94      0.97        47

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```



```
    0.9736842105263158

#prediction
pred = model[2].predict(X_test)
print('Our model prediction: ')
print(pred)
print()
print('Actual prediction: ')
print(Y_test)

    Our model prediction:
    [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 0 0 0 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 0
     1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0
     1 1 0]

    Actual prediction:
    [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 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0
     1 1 0]
```

CHAPTER 6

CONCLUSION

In this paper we have worked to collect the suitable dataset needed to help in this predictive analysis. This dataset is then processed to remove all the junk data. The predictive analysis method is being used in many different fields and is slowly picking up pace. It is helping us by using smarter ways to solve or predict a problem’s outcome. Our scheme was developed to reduce the time and cost factors of the patients as well as to minimize the work of a doctor. We have tried to use a very simple and understandable model to do this job. Next, machine learning algorithms should be used on the training data and the testing data should be used to check if the outcomes are accurate enough.

In the future, we can also use a dataset to predict the re-occurrence of breast cancer after a surgery or chemotherapy session. Artificial Neural Networks can be applied to make the prediction better and smarter. Accuracy can be increased by selecting better features.

REFERENCES

[1] Wolberg, Street and Mangasarian, "Wisconsin Diagnostic Breast Cancer Dataset" http://archive.ics.uci.edu/ml

[2] Yu, M., 2017. *Breast cancer prediction using machine learning algorithm* (Doctoral dissertation).

[3] Yue, W., Wang, Z., Chen, H., Payne, A. and Liu, X., 2018. Machine learning with applications in breast cancer diagnosis and prognosis. *Designs*, *2*(2), p.13.

[4] Palaniappan, S. and Pushparaj, T., 2013. A novel prediction on breast cancer from the basis of association rules and neural network. *International Journal of Computer Science and Mobile Computing*, *2*(4), pp.269-77.

[5] Cruz, J.A. and Wishart, D.S., 2006. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, *2*, p.117693510600200030.