1. **INTRODUCTION**

**1.1 Problem Statement**

Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today’s society.

The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumours can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is of utmost importance.

The goal is to first predict if a tumour is malignant or bening, followed by prediction of whether there is a chance of relapse of Cancer in the patient or not.

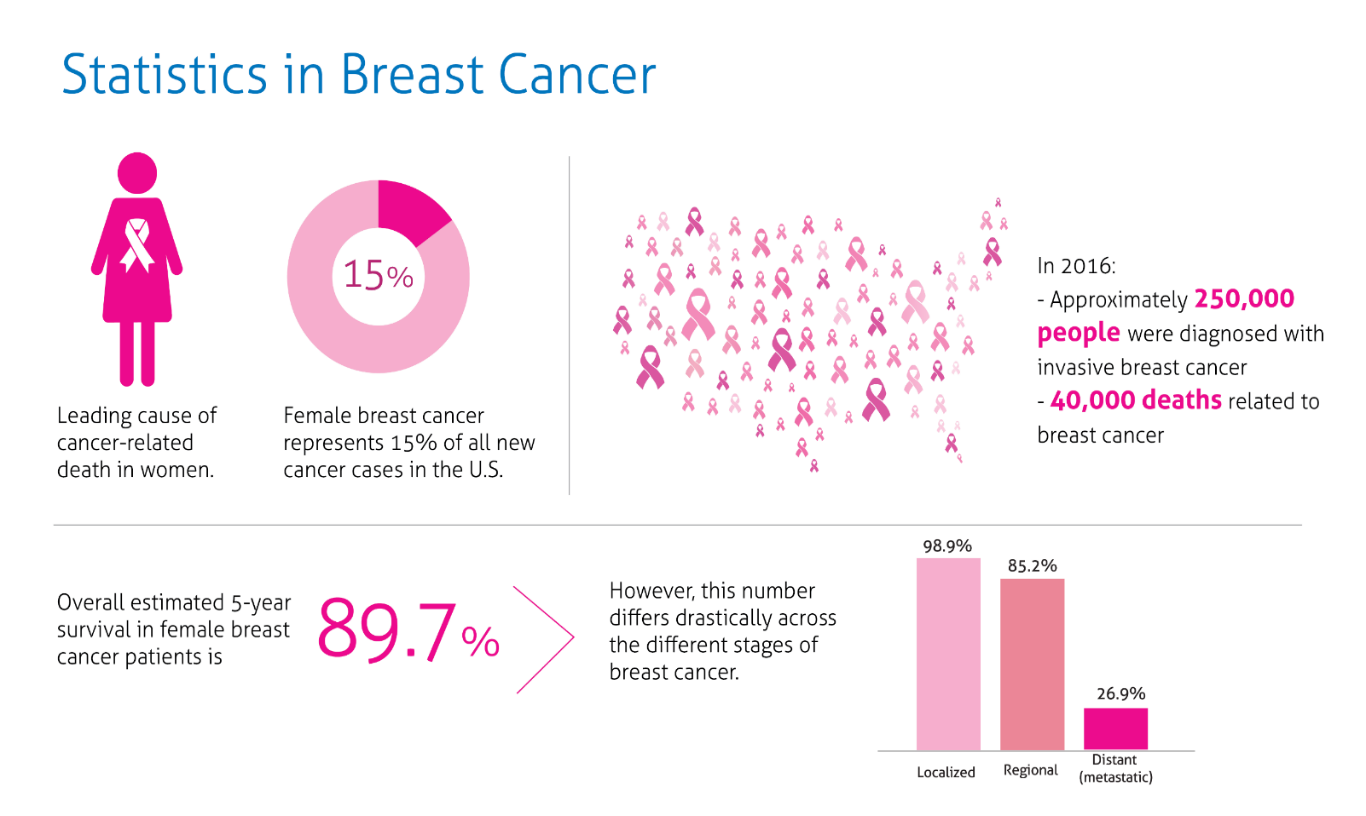


Fig 1.1

**1.2 Scope**

In this project, we use SVM machine learning algorithms to predict the recurrence of breast cancer. Support vector machine (SVM) is an emerging powerful machine learning technique to classify cases. SVM has been used in a range of problems and they have already been successful in pattern recognition in bioinformatics, cancer diagnosis, and more. SVM is a maximum margin classification algorithm rooted in statistical learning theory. It is the method for classifying both linear and non-linear data. It uses a non-linear mapping technique to transform the original training data into a higher dimension. It performs classification tasks by maximizing the margin separating both classes while minimizing the classification errors.

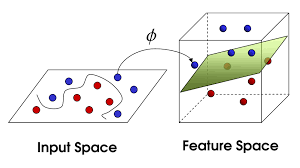


Fig 1.2

* 1. **Applications**

Breast cancer (BC) ﬁgures among the major causes of concern worldwide. According to the latest GLOBOCAN statistics [World Health Organization 2012], it was the second most frequently diagnosed cancer and the ﬁfth cause of cancer mortality worldwide, responsible for 6.4% of all deaths. The goal of breast cancer risk assessment is to personalize management strategies for all women, with the aim of increasing survival in high-risk women while decreasing cost and complications in low-risk women.

**2.DESIGN**

**2.1 Literature Survey**

During the project we went through many research papers out of which we primarily focused on All-CNN network published in the May 2014 IJCSITY paper, " CANCER RECURRENCE PREDICTION USING MACHINE LEARNING ". We also went through popular computer vision websites such as [www.towardsdatascience.com](http://www.towardsdatascience.com) , [www.pyimagesearch.com](http://www.pyimagesearch.com) and read documents related to OpenCv and watched various videos so as to understand the approaches to solve the problem. The main aim in referring the websites was to understand the techniques and select an appropriate and viable method for the project. While searching for a dataset we went through many websites such as www.archive.ics.uci.edu and other dataset websites. We choose Wisconsin Breast Cancer (Diagnostic and Prognostic) dataset due to its contents i.e. it contains about 10 features divided into 3 classes which would surely help in training the model to achieve a high accuracy. For building the neural network we searched for svm libraries and api’s available in Python (as we choose python as the programming language to implement the project). There were many libraries such as sklearn, Deeppy , matplotlib etc. We choose sklearn and matplotlib due to its computational speed.

**2.2 Techniques Used**

**2.2.1 Logistic Regression**

In [statistics](https://en.wikipedia.org/wiki/Statistics), the **logistic model** (or **logit model**) is a widely used [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that, in its basic form, uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable); many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression** (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model; it is a form of [binomial regression](https://en.wikipedia.org/wiki/Binomial_regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, win/lose, alive/dead or healthy/sick; these are represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1". In the logistic model, the [log-odds](https://en.wikipedia.org/wiki/Log-odds) (the [logarithm](https://en.wikipedia.org/wiki/Logarithm) of the [odds](https://en.wikipedia.org/wiki/Odds)) for the value labeled "1" is a [linear combination](https://en.wikipedia.org/wiki/Linear_function_(calculus)) of one or more [independent variables](https://en.wikipedia.org/wiki/Independent_variable) ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a [continuous variable](https://en.wikipedia.org/wiki/Continuous_variable) (any real value). The corresponding [probability](https://en.wikipedia.org/wiki/Probability) of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name.

* + 1. **Nearest Neighbor**

Nearest Neighbour can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

* + 1. **Support Vector Machines**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **support-vector machines** (**SVMs**, also **support-vector networks**) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier) (although methods such as [Platt scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting). A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

**2.2.4 Kernel SVM**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **kernel methods** are a class of algorithms for [pattern analysis](https://en.wikipedia.org/wiki/Pattern_analysis), whose best known member is the [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM). The general task of pattern analysis is to find and study general types of relations (for example [clusters](https://en.wikipedia.org/wiki/Cluster_analysis), [rankings](https://en.wikipedia.org/wiki/Ranking), [principal components](https://en.wikipedia.org/wiki/Principal_components), [correlations](https://en.wikipedia.org/wiki/Correlation), [classifications](https://en.wikipedia.org/wiki/Statistical_classification)) in datasets. In its simplest form, the kernel trick means [transforming data](https://en.wikipedia.org/wiki/Data_transformation) into another dimension that has a clear dividing margin between classes of data. For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into [feature vector](https://en.wikipedia.org/wiki/Feature_vector) representations via a user-specified [*feature map*](https://en.wikipedia.org/w/index.php?title=Feature_map&action=edit&redlink=1): in contrast, kernel methods require only a user-specified *kernel*, i.e., a [similarity function](https://en.wikipedia.org/wiki/Similarity_function) over pairs of data points in raw representation.

* + 1. **Naïve Bayes**

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of [feature](https://en.wikipedia.org/wiki/Feature_vector) values, where the class labels are drawn from some finite set. There is not a single [algorithm](https://en.wikipedia.org/wiki/Algorithm) for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is [independent](https://en.wikipedia.org/wiki/Independence_(probability_theory)) of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible [correlations](https://en.wikipedia.org/wiki/Correlation_and_dependence) between the color, roundness, and diameter features.

* + 1. **Decision Tree Algorithm**

Decision tree learning uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

* + 1. **Random Forest Classification**

**Random forests** or **random decision forests** are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks that operates by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees.[[1]](https://en.wikipedia.org/wiki/Random_forest#cite_note-ho1995-1)[[2]](https://en.wikipedia.org/wiki/Random_forest#cite_note-ho1998-2) Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_set).

**2.3 Functional Requirements**

* The system should train the parameters based on the testing dataset provided.
* System shall recognize the type of cancer ( malignant or benign ) from the testing dataset and place them in the appropriate class.
* The system shall display the predicted result and the actual result for the unknown sample provided.

**2.4 Non-Functional Requirements**

* Performance: The designed system will recognize and classify the cancer in its appropriate class with an accuracy above 85%.
* Reliability: The system should work for all type of values of radius in the dataset.
* Functionality: The system will deliver the functional requirements mentioned in the documents.

**2.5 Software Requirements**

1. Python 3.x
2. PyCharm
3. Sklearn
4. Seaborn
5. Matplotlib
6. Pandas
7. Numpy

**2.6 Hardware Requirements**

1. Intel core i3 +
2. GPU like Nvidia Geforce
3. 4 GB + RAM

**2.7 Description of SVM**

This section includes a brief description of SVM in object recognition.

To illustrate the concept of the Optimal Separating Hyperplane (OSH), we will use an example. Given examples with weight, height and sex of a person, we develop a hypothesis which enables us to determine a persons sex from their weight and height. This can be done by plotting the weights and heights in a 2-D coordinate system and drawing a dividing line or separating hyperplane to separate the weight/height points into male and female regions. A typical plot is shown below.

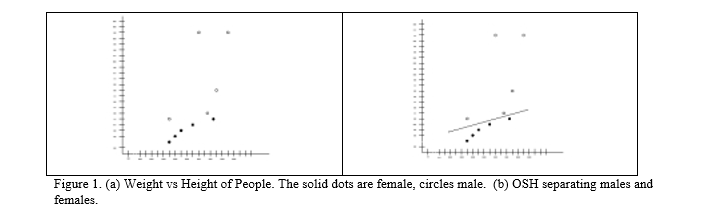


Fig 2.1

Fig 2.1: Weight vs Height of People. The solid dots are female, circles male. (b) OSH separating males and females.

There are a number of possible lines that can separate the data, but there is only one that maximizes the margin (maximizes the distance between it and the nearest data point of each class). This line gives the ‘best’ results, where ‘best’ means that it gives the highest classification rate when new data is used. Intuitively, we would expect this line to generalize well as opposed to other possible ones. This line is called the Optimal Separating Hyperplane. Figure 1 (b) shows the data redrawn with the OSH included.

The OSH algorithm is based on finding a pair of parallel hyperplanes, which separate the data, and which has the largest perpendicular distance between them. It is conjectured that this provides a good approximation to the ‘best’ separating hyperplane. Thus, if we can find an OSH from the training data, as a mathematical function, then only

basic geometry is used to determine which side of the line any given point lies and make a classification of an unseen point. SVMs use geometric properties to exactly calculate the OSH directly from the training data.

Given the following training data:

(x1, y1),….,(xm,ym), x = real, y = {+1,-1} (1)

where each data point is described by a feature vector xi and a truth value yi, the latter of which can take the values of +1 and –1, depending on the class. The two hyperplanes are required to pass through at least one point of each class and there can be no points between them. The boundary between the classes is then defined to be a third parallel hyperplane that is halfway between the other two. The data points that the outer hyperplanes pass through, are called Support Vectors. The two outer hyperplanes are described by the following expressions,

(w • x) + b = +1, (2)

(w • x) + b = -1,

with the first going through a point of class y = +1 and the second going through a point of class y = -1. The constants w and b define the hyperplanes, with w being normal to the hyperplanes and –b/||w|| being the perpendicular distance from the origin to the middle hyperplane. The RHS of Equation (2) will be greater than or equal to +1 for all points of class y = +1 and will be less than or equal to –1 for all points of class y = -1. These can be combined into the following constraint on all the data points,

yi[(w • xi) + b] - 1 >0, i = 1,…m (3)

The perpendicular distance between the two outer hyperplanes (margin) is equal to 2/||w||. Therefore, finding the hyperplanes with the largest margin reduces to finding values for w and b that maximize 2/||w|| or equivalently minimize ½ ||w||2 = ½ (w• w), subject to the constraint in Equation (3).

In other words,

Minimize: f(w) = ½ ||w||2

Subject to: yi[(w • xi) + b] >1, i = 1, … m

A standard method for handling optimization problems with constraints is through the minimization of the Lagrangian. The constraints are taken into account by adding terms involving Lagrange multipliers to the objective function. In this case, this results in the following primal Lagrangian,

C:\Users\SHIVAM\Pictures\Screenshots\Screenshot (12).png

Fig 2.2

Where i α are the Lagrange multipliers associated with each of the constraints in Equation (3). The Lagrangian has to be minimized with respect to the primal variables w and b, and maximized with respect to the dual variables i α (ie a saddle point exists). At the saddle point, the derivatives of Lp with respect to the primal variables must be equal to zero. Doing this results in the following expressions,

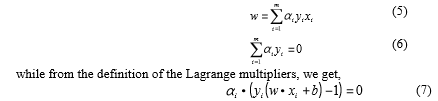


Fig 2.3

Inserting Equations (5) and (6) into (4), removes the primal variables and results in the Wolfe dual Lagrangian where we just have to find the i α which maximize:

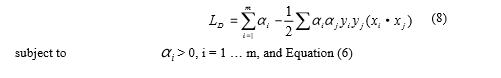


Fig 2.4

This is attractive because the problem is reduced to finding the Lagrange multipliers (the dual variables) that maximize Equation (8) and satisfy both the non-negative constraints and the constraints of Equation (6). Equation (7) means that only those data points which lie on the outer hyperplanes (and hence are active constraints) will have non-zero Lagrange multipliers. These data points are called the support vectors and they are the points that determine the position of the hyperplanes. One can move the other points around the feature space or remove them entirely and the solution will not change, provided one does not move a point across one of the outer hyperplanes.

Equation (8) can be solved using any quadratic programming solver. Once the Lagrange multipliers are known, the solution for w is given by Equation (5), where the sum is over the support vectors, since they are the only ones with non-zero α . One can find b from Equation (7), using any of the support vectors, although one generally averages over all the support vectors for better accuracy. Once these constants are known, the classification of an unknown vector, v, is given by the sign of,

C:\Users\SHIVAM\Pictures\Screenshots\Screenshot (11).png

Fig 2.5

where the sum is over the support vectors. This determines on which side of the boundary (or middle) hyperplane that the data point falls.

**3.Implementation**

**3.1 Data Preprocessing**

**3.1.1 Data Preparation**

We will use the UCI Machine Learning Repository for breast cancer dataset. After extensive research, we have selected the Wisconsin Breast Cancer Dataset. We will use the Diagnostic dataset to predict if a tumour is Malignant or Benign, and the Prognostic dataset to predict if there is a chance of recurrence of BC or not.

*Diagnostic:*[*http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29*](http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29)

*Prognostic:*[*http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28prognostic%29*](http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28prognostic%29)

The dataset used in this project is publicly available and was created by Dr. William H. Wolberg, physician at the University of Wisconsin Hospital at Madison, Wisconsin, USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, then it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector.

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign)

3–32) Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter² / area — 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension (“coastline approximation” — 1)

The mean, standard error and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

* + 1. **Data Exploration**

We will be using ***PyCharm*** to work on this dataset. We will first go with importing the necessary libraries and our dataset to Pycharm:

#importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#importing our cancer dataset

dataset = pd.read\_csv(‘ddata.csv')

X = dataset.iloc[:, 1:31].values

Y = dataset.iloc[:, 31].values

We can examine the data set using the pandas’ head() method:

dataset.head()

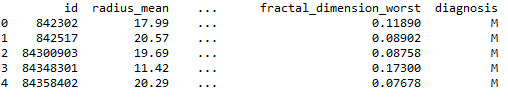


Fig 3.1

We can find the dimensions of the data set using the panda dataset ‘shape’ attribute.

print("Cancer data set dimensions : {}".format(dataset.shape))

Cancer data set dimensions : (569, 32)

We can observe that the data set contain 569 rows and 32 columns. ‘Diagnosis’ is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B (benign) and 212 as M (malignant).

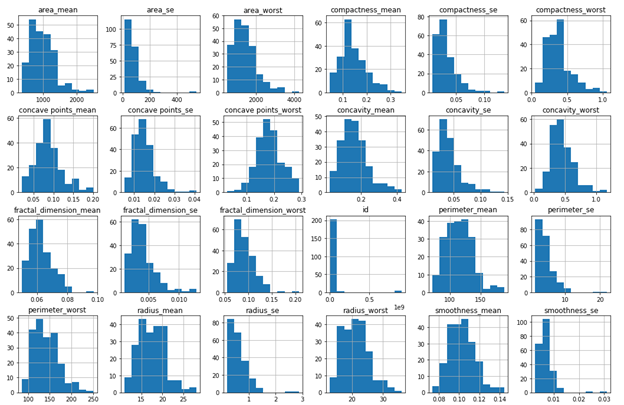


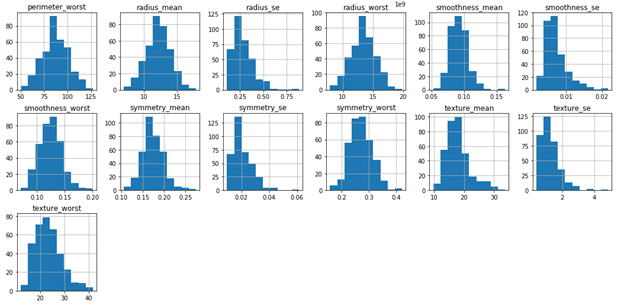
Fig 3.2

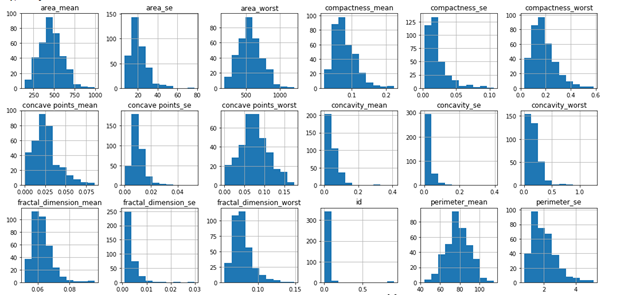
**3.1.3 Data Visualization**

Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person. Python has several interesting visualization libraries such as Matplotlib, Seaborn etc.

In this tutorial we will use pandas’ visualization which is built on top of matplotlib, to find the data distribution of the features.







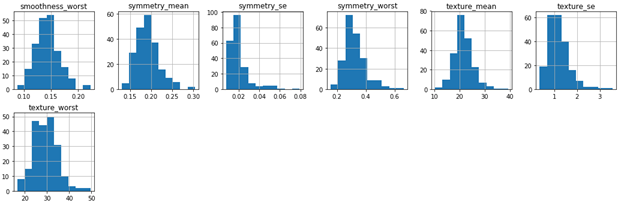


Fig 3.3

**3.2 Algorithm**

**3.2.1 Splitting the Data**

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model’s prediction on this subset.

We will do this using SciKit-Learn library in Python using the train\_test\_split method.

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.25, random\_state = 0)

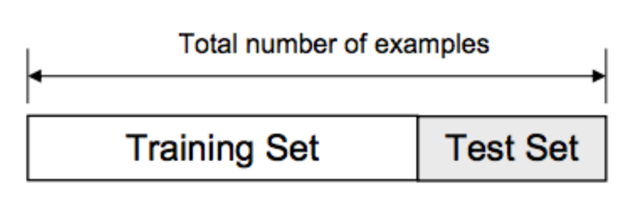


Fig 3.4

**3.2.2 Feature Scaling**

### Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you’re transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

### We will use StandardScaler method from SciKit-Learn library.

#Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**3.2.3 Model Selection**

This is the most exciting phase in Applying Machine Learning to any Dataset. It is also known as Algorithm selection for Predicting the best results.

In our dataset we have the outcome variable or Dependent variable i.e. Y having only two set of values, either M (Malign) or B(Benign). So, we will use Classification algorithm of supervised learning.

We have different types of classification algorithms in Machine Learning:-

1. Logistic Regression

2. Nearest Neighbor

3. Support Vector Machines

4. Kernel SVM

5. Naïve Bayes

6. Decision Tree Algorithm

7. Random Forest Classification

#Using Logistic Regression Algorithm to the Training Set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, Y\_train)

#Using KNeighborsClassifier Method of neighbors class

from sklearn.neighbors import KNeighborsClassifier

classifier=KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, Y\_train)

#Using SVC method of svm class to use Support Vector Machine Algorithm

from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(X\_train, Y\_train)

#Using SVC method of svm class to use Kernel SVM Algorithm

from sklearn.svm import SVC

classifier = SVC(kernel = 'rbf', random\_state = 0)

classifier.fit(X\_train, Y\_train)

#Using GaussianNB method of naïve\_bayes class to use Naïve Bayes Algorithm

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, Y\_train)

#Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, Y\_train)

#Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorithm

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0)

classifier.fit(X\_train, Y\_train)

**3.3 Results**

We will now predict the test set results and check the accuracy with each of our model:

Y\_pred = classifier.predict(X\_test)

To check the accuracy, we need to import confusion matrix method of metrics class. The confusion matrix is a way of tabulating the number of mis-classifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(Y\_test, Y\_pred)

We will use Classification Accuracy method to find the accuracy of our models. Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.



Fig 3.5

To check the correct prediction, we must check confusion matrix object and add the predicted results diagonally which will be number of correct prediction and then divide by total number of predictions.

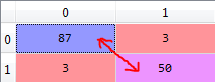


Fig 3.6

After applying the different classification models, we have got below accuracies with different models:

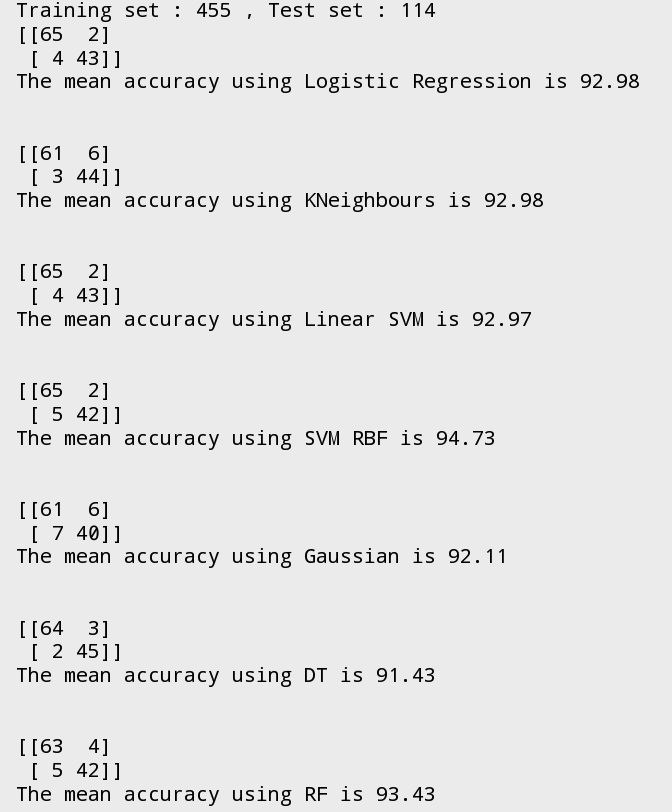


Fig 3.7

**4.Conclusion**

So finally, we have built our classification model and we can see that Non-Linear SVM Classification algorithm gives the best results for our dataset. Well it’s not always applicable to every dataset. To choose our model we always need to analyze our dataset and then apply our machine learning model. We have successfully obtained the highest accuracy of 94.7%.

**5.Future Scope**

This model can be trained further so as to increase the accuracy of the system. It can be used for other datasets as well, with minor changes in the data preprocessing and wrangling.

We can further increase the accuracy and performance of data-sets prediction either by increasing or decreasing data feature or features selection or applying feature engineering in our machine learning model.

**6.References**

1. Dataset: http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29
2. https://www.sciencedirect.com/science/article/pii/S2001037014000464
3. https://www.researchgate.net/publication/309088308\_Predicting\_Breast\_Cancer\_Recurrence\_Using\_Machine\_Learning\_Techniques\_A\_Systematic\_Review
4. https://www.omicsonline.org/using-three-machine-learning-techniques-for-predicting-breast-cancer-2157-7420.1000124.pdf
5. https://medium.com/datadriveninvestor/breast-cancer-detection-using-machine-learning-475d3b63e18e
6. [www.wikipedia.com](http://www.wikipedia.com)
7. https://www.ncbi.nlm.nih.gov/pubmed/12586794