Breast Cancer Prediction Using Machine Learning

Vaibhav Sachdeva

1710110369

Prof. Madan Gopal

Abstract

Affecting roughly around 10 percent of the women across the globe in some stage of their lives, Breast Cancer has stood out to be one of the most feared and frequently occurring cancers at present among women [1]. While the cure for this cancer is now available in almost all first world and some of the third world nations, the main dilemma takes place when the cancer cannot be correctly identified at the very initial stages. Machine Learning, in this field has proved to play a vital role in predicting diseases such as cancers alike. Classification and data mining methods so far have been reliant and an effective way to classify data. Especially in medical field, these methods have been used to predict and to make decisions. In this report, I have successfully used three classification techniques in the form of Naïve Bayes, K-Nearest Neighbors and Logistic Regression on the Wisconsin Breast Cancer datasets. The main objective is to assess the correctness in classifying data with respect to the efficiency and effectiveness of each algorithm in terms of accuracy, sensitivity and specificity and false positive rate. These techniques are coded in Python and executed in Jupyter Notebook. Experimental results have shown that Logistic Regression performs best among the used algorithms when it comes to Breast Cancer Prediction for this dataset.

Contents

Chapter 1

Overview

This project is related to EED-363 Applied Machine Learning course. The present report starts with a general idea of the project and by representing its objectives.

Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict whether a breast cancer cell is benign or malignant until a final model. Results will be explained. Finally, the report will end with some concluding remarks.

1.1 Introduction

Breast cancer is the most common cancer in women, affecting about 10 percent of all women at some stages of their life. In modern times, the rate keeps increasing and data show that the survival rate is 88 percent after five years from diagnosis and 80 percent after 10 years from diagnosis. Early prediction of breast cancer so far have made heaps of improvement, death rate of breast cancer by 39 percent, starting from 1989. Due to varying nature of breast cancers symptoms, patients are often subjected to a barrage of tests, including but not limited to mammography, ultrasound and biopsy, to check their likelihoods of being diagnosed with breast cancer. Biopsy, is the most indicative among these procedures, which involves extraction of sample cells or tissues for examination. The sample of cells is obtained from a breast fine needle aspiration (FNA) procedure and then sent to a pathology laboratory to be examined under a microscope [2]. Numerical features, such as bland chromatin, mitoses and bare nucleoli, can be observed from microscopic images. Data, later on, obtained from FNA are analyzed in combination with various imaging data to predict probability of the patient having malignant breast cancer tumor. An automated system here would be hugely beneficial in this scenario. It will likely expedite the process and enhance the accuracy of the doctor's predictions. In addition, if supported by abundance dataset and the automated system consistently performs well, it will potentially eliminate the needs for patients to go through various other tests, such as mammography, ultrasound, and MRI, which subject patients to significant amount of pain and radiation. In all, early prediction remains is one of the vital aspects in the follow-up process. Data mining methods or classification can help to reduce the number of false positive and false negative decision. The major models used and tested will be supervised learning models (algorithms that learn from labelled data), which are most used in these kinds of data analysis. In this report, using three classification models; Naïve Bayes, K- Nearest Neighbors and Logistic Regression have been run on the Wisconsin Breast Cancer Dataset. The results obtained are than measured using various performance metrics to compare among the algorithms in order to find out the best suited model for cancer prediction.

1.2 Aim of the project

The objective of this report is to train machine learning models to predict whether a breast cancer cell is Benign or Malignant. Data will be transformed to reveal patterns in the dataset and create a more robust analysis. As previously said, the optimal model will be selected following the resulting accuracy, sensitivity, specificity, amongst other factors. We will later define these metrics. We can use machine learning method to extract the features of cancer cell nuclei image and classify them. It would be helpful to determine whether a given sample appears to be Benign ("B") or Malignant ("M").

The machine learning models that we will applicate in this report try to create a classifier that provides a high accuracy level combined with a low rate of false-negatives (high sensitivity) and false positives (high specificity).

1.3 Dataset

The present report covers the Breast Cancer Wisconsin Dataset (https://www.kaggle.com/roustekbio/breast-cancer-csv) created by Dr. William H. Wolberg, physician at the University of Wisconsin Hospital at Madison, Wisconsin, USA. The data used for this project was collected in 1992 by the University of Wisconsin and it is composed by the biopsy result of 699 patients in Wisconsin Hospital [3].

The dataset's features describe characteristics of the cell nuclei on the image. The features information are specified below:

1) Sample code number id number
2) Clump Thickness 1 - 10
3) Uniformity of Cell Size 1 - 10
4) Uniformity of Cell Shape 1 - 10
5) Marginal Adhesion 1 - 10
6) Single Epithelial Cell Size 1 - 10
7) Bare Nuclei 1 - 10
8) Bland Chromatin 1 - 10
9) Normal Nucleoli 1 – 10
10) Mitoses 1-10

Chapter 2

Literature Review

Throughout the decades, ML algorithms have been widely used in BC diagnosis and prognosis to gain different insights from data samples. ML is a form of Artificial Intelligence (AI) that uses a variety of statistical, probabilistic and optimization tools to learn and improve performance automatically from new data and past experiences, without explicitly programmed instructions. Both statistics and ML are used for analyzing data. In dealing with complex, large, high dimensional data, ML approaches are typically capable of extracting key features and potential rules which might be difficult to be discovered using traditional statistics. As discussed in [4-7] for BC datasets, ML approaches have proved to be more suitable than statistics ones. Traditional statistical methods of data analysis to predict medical outcomes include but are not limited to **logistic regression**, linear regression and discriminant analysis [8-12]. Much work comparing different traditional statistical methods with traditional ML classification methods have been published to demonstrate the advantages of ML and its potential [13]. The early findings showed results to the contrary; in 1998, Fogel et al. compared **ANN (Artificial Neural Network)** with linear discriminant analysis of the radiographic features of masses and patient's age in 139 suspicious masses to detect whether they developed breast cancer and demonstrated the superiority of latter [14]. Lately however with the maturity and improvement of the ML algorithms, and the increasing quantity and complexity of the data, results show ML approaches have better classification accuracy [4, 6, 7, 15-17]. In 2004, two ML classification methods (DT and ANN) were compared with a statistical method (linear regression) to predict the breast cancer survival using a large dataset which has more than 200,000 cases and demonstrated that ML methods could be a promising classification method for practical use. The results showed that DT was the best predictor with the accuracy of 93.6% with ANN achieving 91.2% and both were better than linear regression achieving 89.2% accuracy [4]. ML algorithms can be classified into two general categories based on the way they "learn" about data; supervised learning and unsupervised learning. Choosing whether to use a supervised or an unsupervised ML algorithm generally depends on the data types and structures. A review of current research reveals that almost all the ML algorithms employed in the BC diagnosis and prognosis are supervised [18]. An artificial neural network (ANN) can be defined as a model of reasoning based on the human brain [19]. Over the past few decades, ANNs have been employed increasingly by more and more researchers, and become an active research area [20-23]. ANNs have afforded numerous successes with great progress in BC classification and diagnosis in the very early stages [14, 24-30]. As early as 1993, a three-layer feedforward network with BP algorithm was utilized to predict whether patients had malignant or benign lesions by analyzing a mammography atlas of breast cancer from a database of 133 cases [30]. The mammography atlas database was compiled by Taber and Dean in 1985 [31]. This paper proved ANN can be helpful in the mammography analysis at early on. A year after, Floyd et al. proposed an ANN method for predicting breast cancer from mammographic findings [24]. At that time, Floyd et al. used the basic BP algorithm with a three-layer architecture. The use of the ANN proved to give better diagnostic performance than the radiologists when the network output was compared to the radiologists' categorical assessment. Both [24-30] utilized ANN to predict malignancy using different mammographic elements as inputs. The accuracies were significant and improved by 3–5% compared with conventional expert's judgment. Thus, more than twenty years ago, ANN has been proved excellent in BC diagnosis and prognosis. The concept of **SVM (Support Vector Machines)**, which was proposed by Vapnik on the basis of the statistical learning theory [32], also become an essential component in ML techniques. A few decades ago, a basic SVM approach was used to interrogate the WBCD [33] and 97.2% classification accuracy was obtained. The result has proved the basic SVM approach achieved a high accuracy given quality data in BC. Then in 2007, a least square SVM (LS-SVM) [34] was used to improve the accuracy up to 98.53% [35]. An SVM-based algorithm combined with feature selection was proposed to achieve 99.51% when classifying the WBCD data [36].

This method chose the best feature combination (of five features) by calculating the F-score of different feature combinations. With the same database, a rough set (RS)-based SVM classifier (RS-SVM) was proposed, in which a different feature selection method using the rough set attribute reduction algorithm was applied to generate 20 subsets of attributes [37]. The further reduction of subsets (from 20 to 7) was achieved by calculating the correlation between each condition attribute with a decision attribute. The condition attribute with the weakest relevancy was kept, since the prediction accuracy can be improved by a weakly relevant feature [38]. The highest classification accuracies were 99.41%, 100% and 100%, simultaneously the averages were 96.55%, 96.72% and 96.87% in 50–50, 70–30 and 80–20 training-testing datasets. **K-Nearest Neighbors** is one of the most central ML techniques in classification [39]. K-NN is a non-parametric lazy learning algorithm used for classification, which classifies the objects using their "K" nearest neighbors. K-NN only considers the neighbors around the object, not the underlying data distribution. K-NN related algorithms have a number of applications in BC diagnosis and prognosis. The quality of the classification depends on the selection of K. In 2000, the K-NN and fuzzy K-NN algorithms were implemented to classify the WBCD [40]. The different values of K from 1 to 15 were considered, and the best performance was when K equaled 1. The K-NN algorithm achieved the accuracy of 98.25% in the testing set, and fuzzy K-NN acquired 98.83%. In 2003, rank K-NN has been used to achieve 98.10% [41]. In 2013, four methods of calculating distance were used for K-NN to determine the distance between each WBCD of the data points [42]. The results showed that the highest classification accuracy was 98.70% when using the Euclidean distance with k=1. Another classification algorithm that gained a very high importance was the **Decision Tree Algorithm**. In ML, DT is a predictive model that represents a mapping between object attributes and object values [43,44]. In 1996, Quinlan applied a modified C4.5 decision tree which has the benefits of having a smaller construction and higher predictive accuracy, to twenty UCI databases, which included the WBCD [45]. In 2005, 3 features were extracted from the WBCD using the C4.5 decision tree with expert suggestions. The 6 remaining features were used by an artificial immune recognition system (AIRS) [46] to do the classification. The new feature selection AIRS (FS-AIRS) could achieve 98.51% accuracy [47]. In2014, a J48 decision tree method, originally developed by the WEKA project team [48], was used to classify WBCD with a classification accuracy of 94.56% [49]. In 2016, an accuracy of 99.9% was achieved using three steps [50]: clustering using farthest first clustering (FFC), pre-processing using outlier detection (OD) and classification using the J48 decision tree. By removing 120 extreme values from malignant data, the new dataset only contained 578 instances. Although 99.9% classification accuracy has been attained, this algorithm has the limitation of not considering all instances. In the same year, a hybrid intelligent system for medical data classification was proposed [51] consisting of a combination of the fuzzy min-max (FMM) ANN [52, 53], the classification and regression tree (CART) and **random forest** (FF) model [54]. This hybrid intelligent system combined the advantages of these three techniques. This novel system was able to learn incrementally from data samples (owing to the fuzzy min-max ANN), explain its predicted outputs (owing to the classification and regression tree) and achieve high classification performances (owing to random forest). By analyzing the benchmark dataset WBCD, 98.84% accuracy was obtained.

With due respect to all related work referred above, this report compares the performance of the algorithms; Naïve Bayes, K Nearest Neighbors, Logistic Regression, Support Vector Machine (SVM), Decision Trees, Random Forest Classifier and Multi-Layer Perceptron (ANN) using Wisconsin Breast Cancer (original) datasets in both diagnosis and analysis to make decisions. The goal is to achieve the most efficient algorithm to help us predict breast cancer at the very initial stages. To do so, we compare efficiency and effectiveness of those approaches in terms of certain criteria such as testing accuracy, true positive rate and false positive rate.

Chapter 3

Methods and Analysis

3.1 Data Preprocessing and Exploratory Data Analysis

3.1.1 Data Preprocessing

By observing the given dataset, we found that it contains 699 observations (samples) with 11 variables (features).

```
df.shape
```

```
(699, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
id                   699 non-null int64
clump_thickness      699 non-null int64
size_uniformity      699 non-null int64
shape_uniformity     699 non-null int64
marginal_adhesion    699 non-null int64
epithelial_size      699 non-null int64
bare_nucleoli        699 non-null object
bland_chromatin      699 non-null int64
normal_nucleoli      699 non-null int64
mitoses              699 non-null int64
class                699 non-null int64
dtypes: int64(10), object(1)
memory usage: 60.1+ KB
```

The feature ‘id’ does not contribute in the learning of machine as it doesn’t provide any information based on which benign and malignant cells can be differentiated. Hence, the column ‘id’ is removed/dropped.

The above data information indicates that the datatype for ‘bare nucleoli’ is ‘object’ while other features have the datatype ‘int64’ (indicating numeric values). This difference in datatype indicates that the column ‘bare nucleoli’ consists of values other than numeric values (ideally, all the columns should be having numeric values only).

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

```
1     402
10    132
5      30
2      30
3      28
8      21
4      19
?      16
9       9
7       8
6       4
Name: bare_nucleoli, dtype: int64
```

Upon further analysis it is seen that the column 'bare nucleoli' consists of 16 '?' indicating missing values.

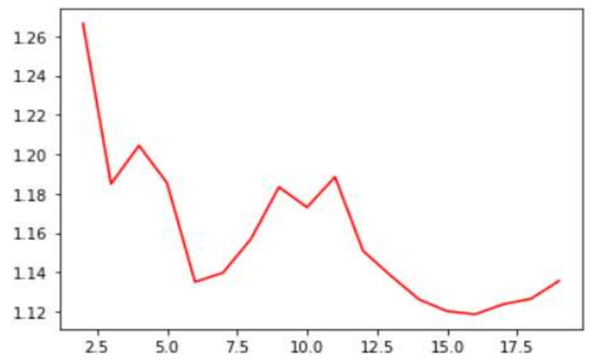
Since the dataset consists of only 699 samples (and not millions), dropping samples with missing feature values will not be a good option as these samples might contain information that can affect the prediction of the machine. Predicting these missing values using an algorithm seems to be a better alternative than dropping these samples.

The missing values were predicted using **K-Nearest Neighbors algorithm**.

Initially, all the 16 samples consisting of '?' in the bare nucleoli feature were extracted from the dataset. The remaining 683 samples were then treated as a separate dataset. This dataset was then used to train the machine using the K-Nearest Neighbors algorithm. Finally, the trained machine was used to predict the values for the missing features.

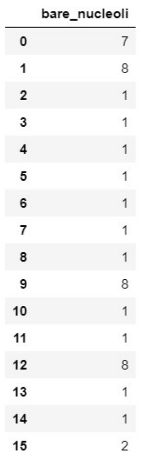
Using a trained machine for predicting the missing values is a better choice than replacing these missing values with mean/median/mode values of bare nucleoli because a trained machine takes into consideration all the features corresponding to all the samples and then predicts the output. This predicted output will take into account not only the values present in one particular feature (as in the case of mean/media/mode), but will take the entire dataset and hence the distribution.

Here, on the basis of the minimum Mean Absolute Error, the number of neighbors were taken to be 16 while the distance/similarity function was taken to be 'Minkowski'(to take care of outliers (if any) present in the dataset). The selection of appropriate number of neighbors and the distance function will be discussed in detail later.



```
Best k parameter is  16
```

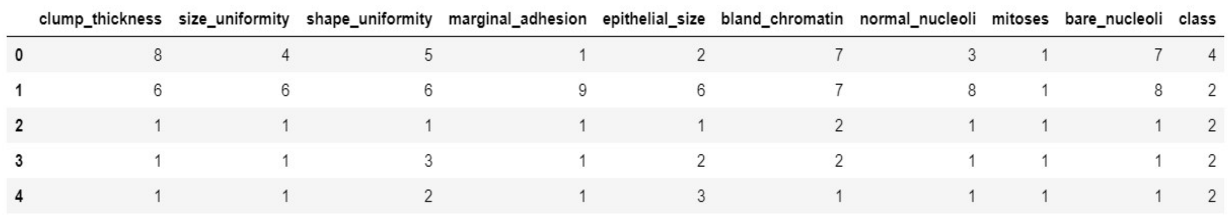
The predicted values were then rounded off and appended to the extracted samples. These extracted samples were then concatenated to the remaining 683 samples in order to obtain the previous dataset with the new predicted values.



	bare_nucleoli
0	7
1	8
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	8
10	1
11	1
12	8
13	1
14	1
15	2

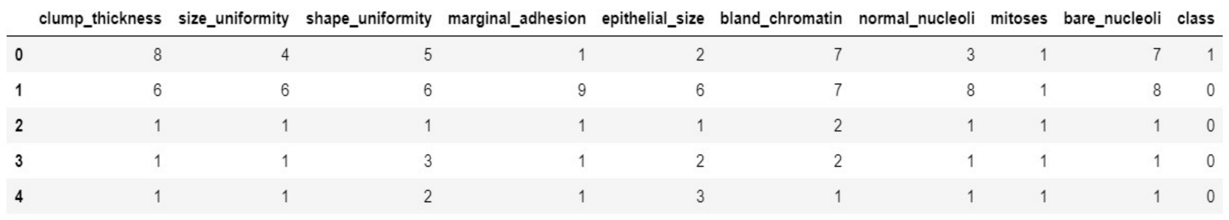
Thus, the dataset now has no missing values and we can move on to further analyses.

The column ‘class’ of the dataset consists of two unique classes (binary classification) namely ‘2’ and ‘4’ where class=2 represents a benign tumor (not cancerous) and class=4 represents a malignant tumor (dangerous).



	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bland_chromatin	normal_nucleoli	mitoses	bare_nucleoli	class
0	8	4	5	1	2	7	3	1	7	4
1	6	6	6	9	6	7	8	1	8	2
2	1	1	1	1	1	2	1	1	1	2
3	1	1	3	1	2	2	1	1	1	2
4	1	1	2	1	3	1	1	1	1	2

These two unique classes (‘2’ and ‘4’) are then encoded to 0 and 1. Now, class=0 represents a benign tumor and class=1 represents a malignant tumor.

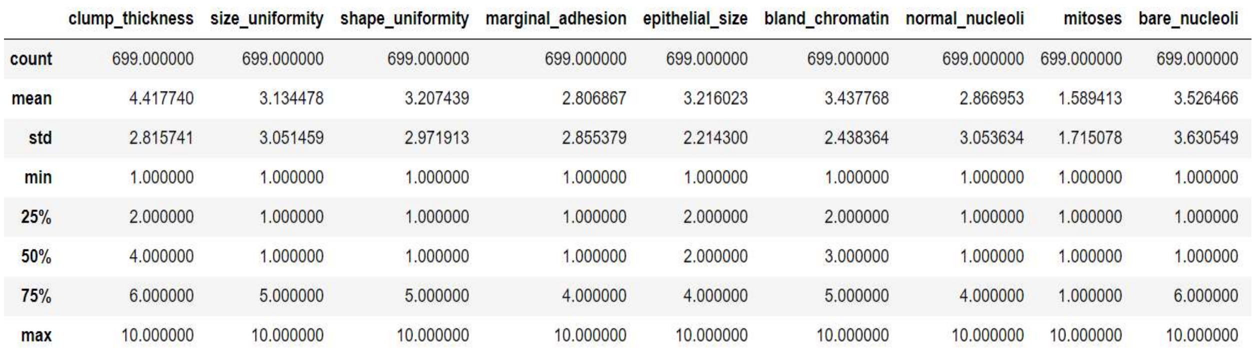


	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bland_chromatin	normal_nucleoli	mitoses	bare_nucleoli	class
0	8	4	5	1	2	7	3	1	7	1
1	6	6	6	9	6	7	8	1	8	0
2	1	1	1	1	1	2	1	1	1	0
3	1	1	3	1	2	2	1	1	1	0
4	1	1	2	1	3	1	1	1	1	0

3.1.2 Exploratory Data Analysis

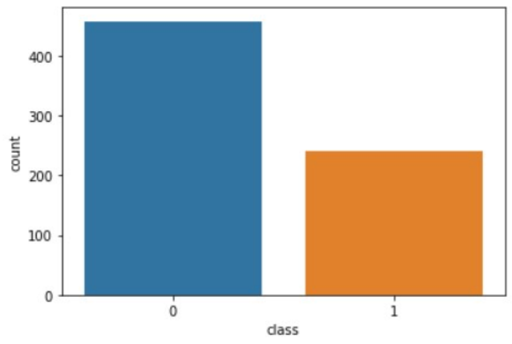
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

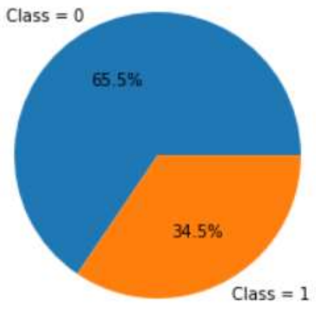
To start with, I calculated the mean, variance, standard deviation, maximum and minimum values for each feature present in the dataset.



	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bland_chromatin	normal_nucleoli	mitoses	bare_nucleoli
count	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000
mean	4.417740	3.134478	3.207439	2.806867	3.216023	3.437768	2.866953	1.589413	3.526466
std	2.815741	3.051459	2.971913	2.855379	2.214300	2.438364	3.053634	1.715078	3.630549
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	2.000000	1.000000	1.000000	1.000000	2.000000	2.000000	1.000000	1.000000	1.000000
50%	4.000000	1.000000	1.000000	1.000000	2.000000	3.000000	1.000000	1.000000	1.000000
75%	6.000000	5.000000	5.000000	4.000000	4.000000	5.000000	4.000000	1.000000	6.000000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000

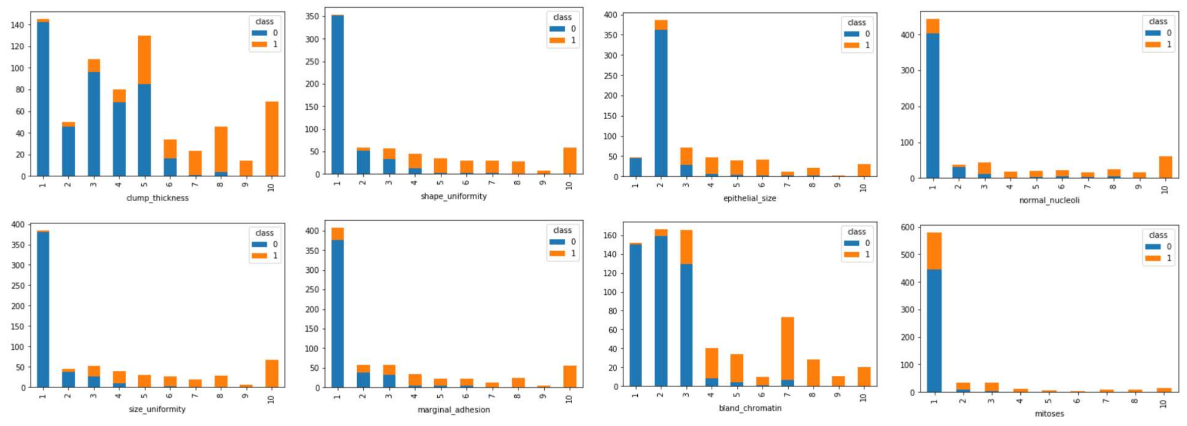
Further, I plotted the count plot and pie chart for the feature ‘class’ so as to decide whether the given data is balanced or not. Data is considered to be balanced if the distribution of the two classes is about 50% each whereas the data is considered to be imbalanced/skewed if the distribution of the two classes is around 90-10.





Both count plot and the pie chart give a quantitative comparison of the binary classes present in the dataset. 65.5% of the given samples are classified as Benign while the remaining 34.5% are classified as Malignant. Since the distribution is 65-35 (slightly imbalanced), it is assumed to be balanced and hence no up-sampling/down-sampling/cost sensitive learning is performed. All the further analysis performed in this report assume the data to be balanced.

For a detailed quantitative analysis, cross tab plot for the all the features of the given dataset were plotted.

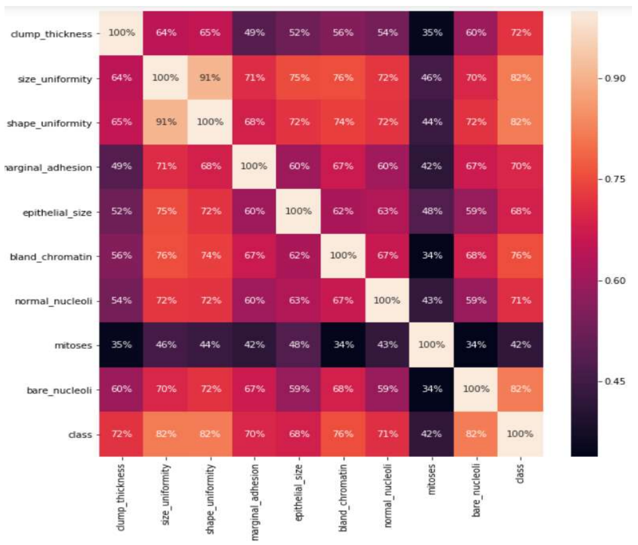


These cross tab plots indicates the number of tumors classified as Benign or Malignant based on the value (1-10) of a certain feature.

In every graph it can be clearly seen that if the feature has a value greater than a certain threshold, the cell is classified as Malignant. Different thresholds for different feature values can also be observed.

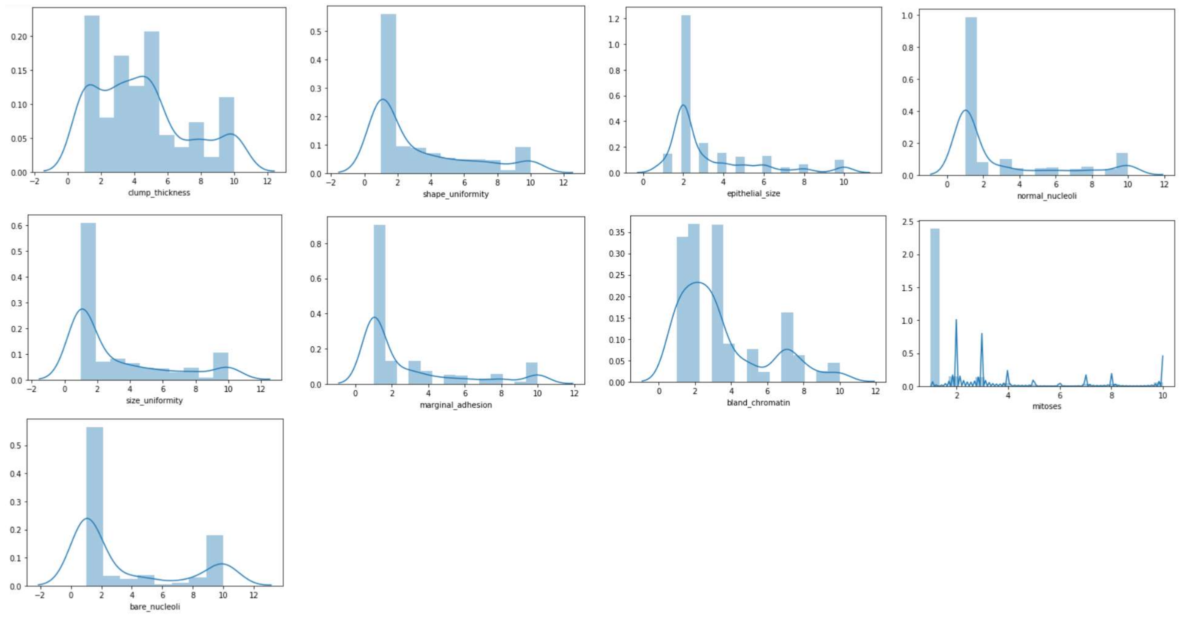
Several features are more sensitive than others in terms of classifying a tumor. For example – If the value of shape uniformity/size uniformity is observed to be greater than 5, the cell is mostly classified as Malignant. In case of mitoses, even if the value reaches 3, then the tumor is more prone to be Malignant while in case of clump thickness, having a value as high as 8 still has a slight chance the cell might not be classified as Malignant. Different features have different impacts on classification and this impact can thus be related to Correlation. Each feature has different correlation with other features and with the class each sample is being mapped to.

Correlation analysis is a statistical method used to evaluate the strength of relationship between two quantitative variables. A high **correlation** means that two or more variables have a strong relationship with each other, while a weak **correlation** means that the variables are hardly related. In other words, it is the process of studying the strength of that relationship with available statistical data. To what extent a feature can affect the other or affect the class can thus be determined with the help of a correlation matrix. The same has been drawn using a heat map.

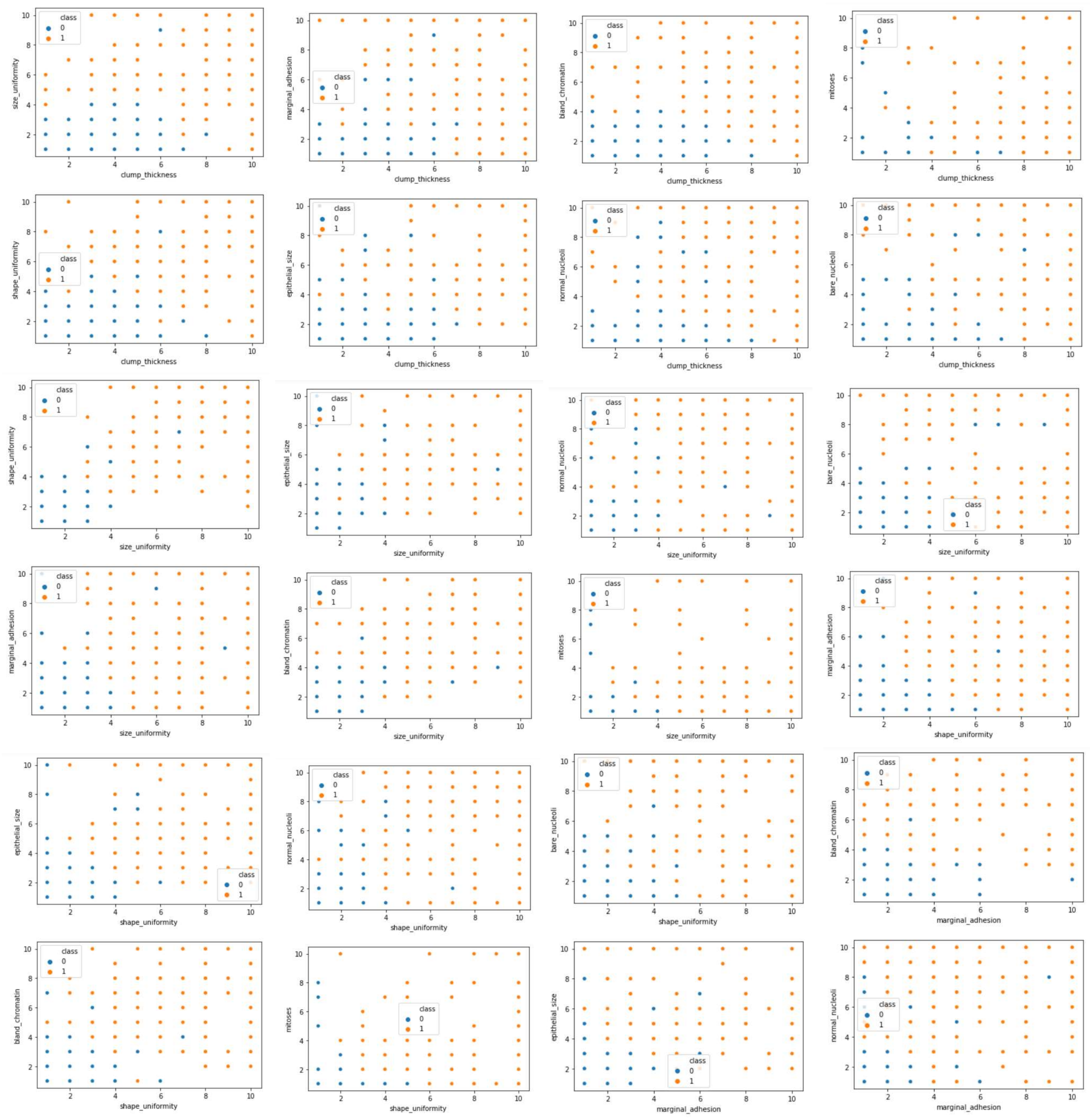


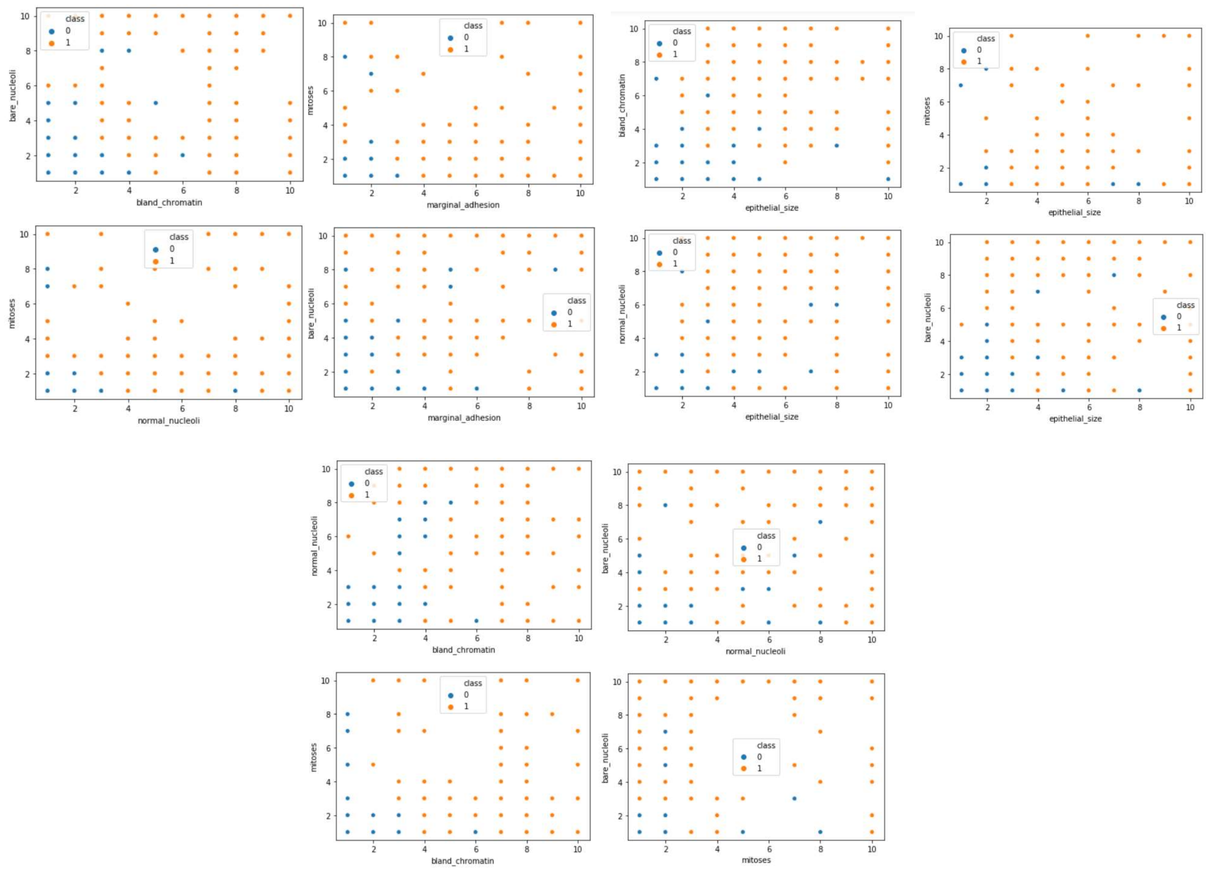
The above heat map shows that the features size uniformity, shape uniformity and bare nucleoli are highly correlated to class while mitoses is the feature that is least correlated to class.

From a practical perspective, we can think of plotting the distribution of feature values for each feature. The distribution provides a parameterized mathematical function that can be used to calculate the probability for any individual observation from the sample space. This distribution describes the grouping or the density of the observations, called the probability density function. We can also calculate the likelihood of an observation having a value equal to or lesser than a given value. A summary of these relationships between observations is called a cumulative density function. Once a distribution function is known, it can be used as a shorthand for describing and calculating related quantities, such as likelihoods of observations, and plotting the relationship between observations in the domain. The distribution plots of all the features are thus plotted. The most variables of the dataset are normally distributed as show with the below plot:



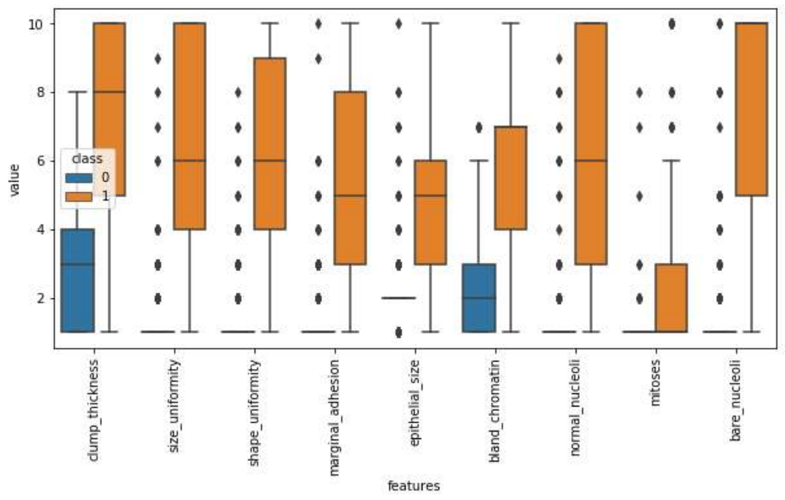
The next task to be accomplished is to analyze the data for defining a classifier. Scatter plots are useful for spotting structured relationships between variables, like whether you could summarize the relationship between two variables with a line. Attributes with structured relationships may also be correlated and good candidates for removal from the dataset. Presence of outliers in the data can easily be spotted with the help of scatter plots and hence can be removed to improve the accuracy of the machine.





The scatter plots between two of the features in the dataset are generated. Blue points represent "Malignant" class whereas orange points represent "Benign" class. In many of the plots plotted here, we can see a distinction between classes or existence of a boundary of sort between malignant and benign classes. This directs to also use a classification algorithm which decides classes based on some diving boundary.

Another thing to be taken care of was to determine if any outliers were present in the dataset. The same was achieved using box plot given below-



As per the box plot, there were few outliers present in the dataset. But since the dataset is too small and every feature has a value between 1 and 10 only, no operations were performed on the dataset for these outliers.

3.2 Modelling

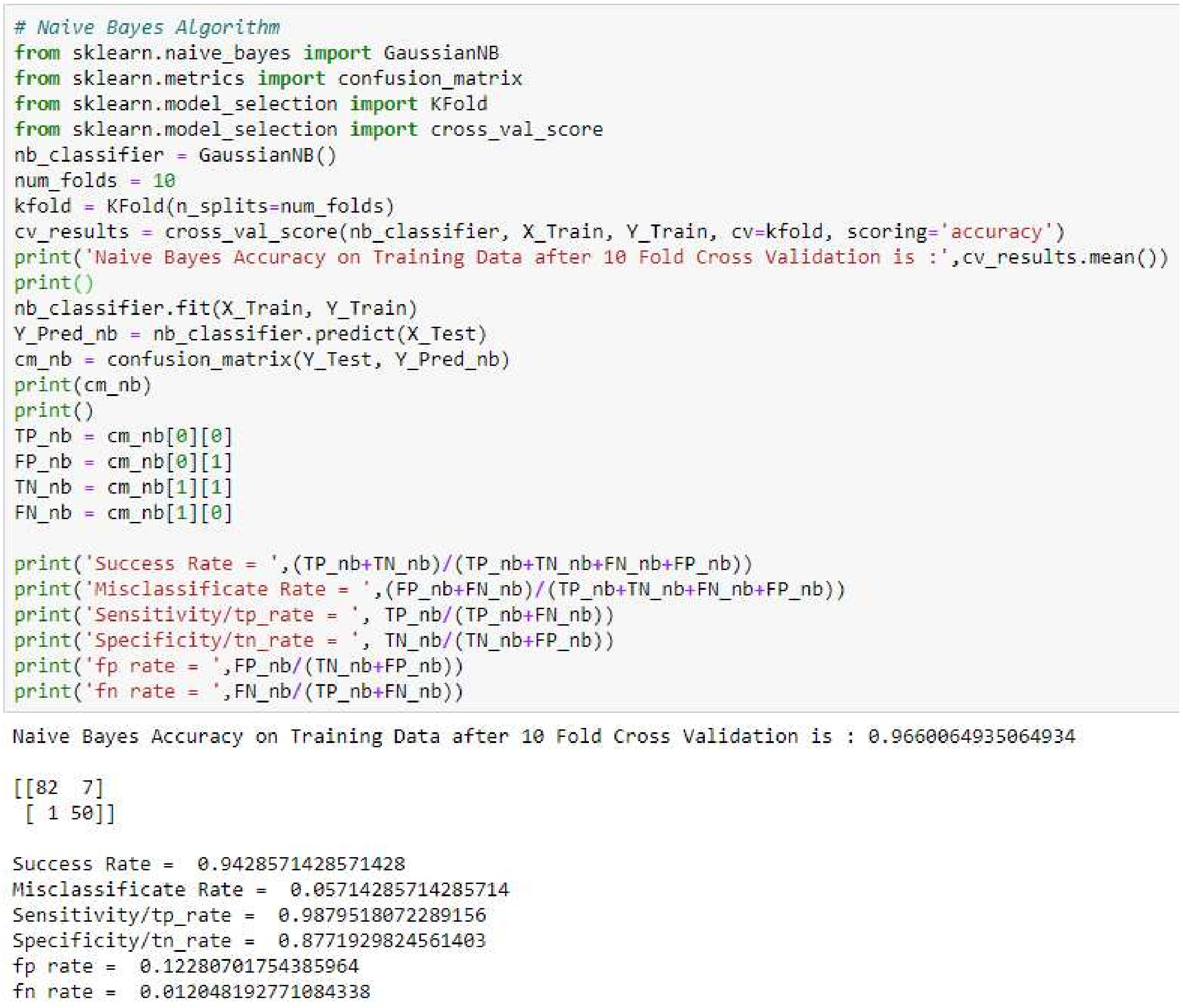
3.2.1 Model creation

Data, in machine learning, in most scenarios are split into training data and testing data (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. Training dataset is a part of the actual dataset that we use to train the model. The model sees and learns from this data. Test data, on the other hand, is the sample of data used to provide an unbiased analysis of a final model fit on the training dataset. The Test dataset provides the ideal standard used to evaluate the model. It is used once the model is completely trained [55]. Splitting the dataset into training, validation testing sets can be determined on two categories. Firstly, it depends on how much the total number of samples in the data and second, on the actual model the user is training. Some models need efficient or large data to train upon, so in that case one could optimize for the larger training sets. Models with very few hyper parameters are estimated to be easy to validate and tune, so one can possibly reduce the size of your validation set. However, given the model has many hyper parameters, the user would want to have a large validation set as well. In this report, I have split the dataset into 80%-20% ratio for training and test respectively (the first 559 instances for training while the next 140 instances for testing the model).

3.2.2 Naïve Bayes Model

Naive Bayes classifiers are a group of classification algorithms supported Bayes' Theorem. Bayes theorem uses the contingent probability that successively uses previous information to calculate the probability that a future event can happen. In Naive Bayes classifier, it's assumed that the input variables are independent of every alternative which all options can separately contribute to the chance of target variable. So, the existence of one feature variable doesn't have an effect on the opposite feature variables. This can be why it's known as Naive. However, in real knowledge sets, the feature variables are dependent on one another therefore this can be one among the drawbacks of Naive Bayes classifier. Naive Bayes classifier though, works fine for giant knowledge sets and generally per-form higher than the difficult classifiers. The formula for Naive Bayes theorem is: Here, P (C|A) is the posterior probability, the probability that a hypothesis (C) is true given some evidence (A). P (C) is the prior probability, the probability of the hypothesis being true. P (A) is the probability of the evidence, irrespective of the hypothesis. P (A|C) is the probability of the evidence when hypothesis is true Naive Bayes algorithmic program is employed for binary and multi category classification and might even be trained on small low information set that could be a huge advantage.

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$



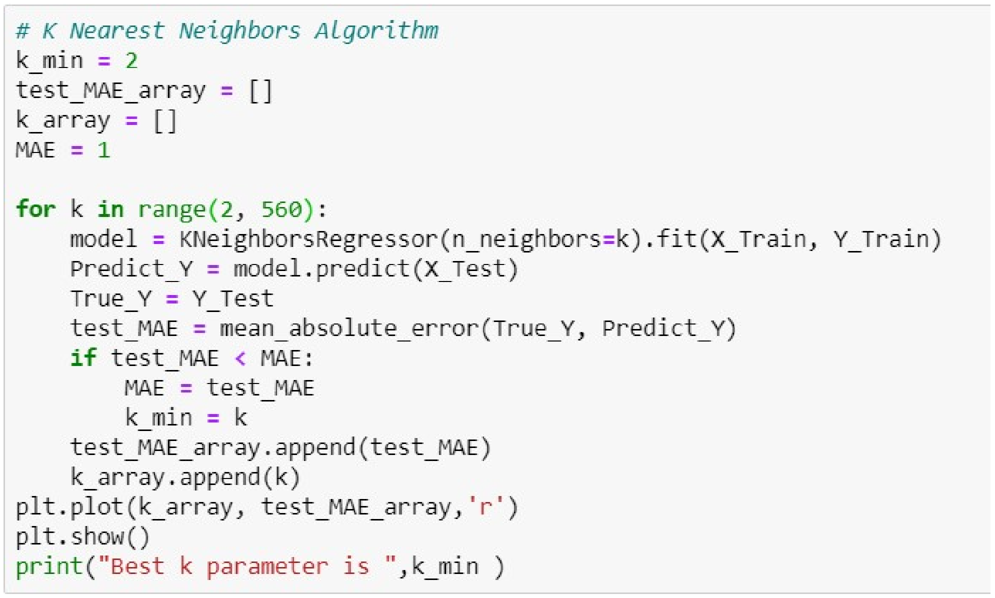
```python
# Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
nb_classifier = GaussianNB()
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(nb_classifier, X_Train, Y_Train, cv=kfold, scoring='accuracy')
print('Naive Bayes Accuracy on Training Data after 10 Fold Cross Validation is :',cv_results.mean())
print()
nb_classifier.fit(X_Train, Y_Train)
Y_Pred_nb = nb_classifier.predict(X_Test)
cm_nb = confusion_matrix(Y_Test, Y_Pred_nb)
print(cm_nb)
print()
TP_nb = cm_nb[0][0]
FP_nb = cm_nb[0][1]
TN_nb = cm_nb[1][1]
FN_nb = cm_nb[1][0]

print('Success Rate = ',(TP_nb+TN_nb)/(TP_nb+TN_nb+FN_nb+FP_nb))
print('Misclassificate Rate = ',(FP_nb+FN_nb)/(TP_nb+TN_nb+FN_nb+FP_nb))
print('Sensitivity/tp_rate = ', TP_nb/(TP_nb+FN_nb))
print('Specificity/tn_rate = ', TN_nb/(TN_nb+FP_nb))
print('fp rate = ',FP_nb/(TN_nb+FP_nb))
print('fn rate = ',FN_nb/(TP_nb+FN_nb))
```

```
Naive Bayes Accuracy on Training Data after 10 Fold Cross Validation is : 0.9660064935064934

[[82  7]
 [ 1 50]]

Success Rate =  0.9428571428571428
Misclassificate Rate =  0.05714285714285714
Sensitivity/tp_rate =  0.9879518072289156
Specificity/tn_rate =  0.8771929824561403
fp rate =  0.12280701754385964
fn rate =  0.012048192771084338
```

3.2.3 K-Nearest Neighbors Model

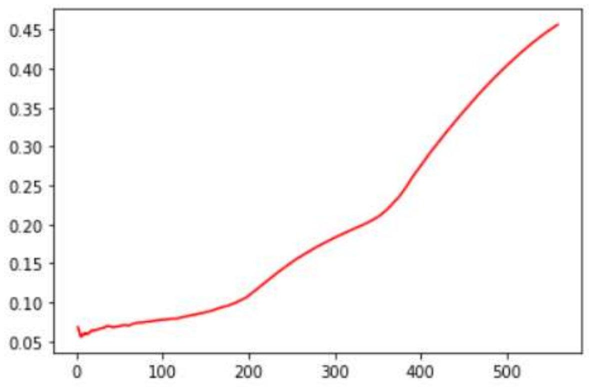
The K-nearest neighbor’s algorithm is one of the simplest machine learning algorithms. It has merely supported the concept that objects that are ‘near’ every alternative can additionally have similar characteristics. So if it can recognize the characteristic options of one of the objects, it will be additionally predicted for its nearest neighbor. KNN is associate improvisation over the nearest neighbor technique. It is based mostly on the plan that any new instance will be classified by the majority vote of its ‘k’ neighbors, - wherever k is a positive number, sometimes a little variety. KNN is one amongst the foremost easy and simple data processing techniques. It is known as Memory-Based Classification as the coaching examples have to be in the memory at run-time. Once handling continuous attributes the distinction between the attributes is calculated using a similarity/distance function. The key issue of KNN algorithm is the distance/similarity function, which is selected on the basis of applications and nature of data. The cautious selection of an appropriate distance function is crucial step in the use of KNN. Another question is the number of neighbors to select. Investigation of varying number of neighbors with the help of validation set can facilitate establishing the optimal number of neighbors. When KNN is employed for classification, the output is calculated because the category with the very best frequency from the K-most similar instances. Every instance in essence votes for their class and therefore the class with the foremost votes is taken for the prediction.

Thus the classification of the test data point hinges on the classification of its nearest neighbors [56]. In the KNN model for this dataset, the value of number of neighbors is derived with the help of MAE (Mean Absolute Error). Given any test data-set, Mean Absolute Error of the model refers to the mean of the absolute values of each prediction error on all instances of the test data-set. Prediction error is the difference between the actual value and the predicted value for that instance. For different number of neighbors the machine is trained and the mean absolute error is calculated and compared. The number of neighbors for which the least mean absolute error is observed is used for training the model.



```
# K Nearest Neighbors Algorithm
k_min = 2
test_MAE_array = []
k_array = []
MAE = 1

for k in range(2, 560):
    model = KNeighborsRegressor(n_neighbors=k).fit(X_Train, Y_Train)
    Predict_Y = model.predict(X_Test)
    True_Y = Y_Test
    test_MAE = mean_absolute_error(True_Y, Predict_Y)
    if test_MAE < MAE:
        MAE = test_MAE
        k_min = k
    test_MAE_array.append(test_MAE)
    k_array.append(k)
plt.plot(k_array, test_MAE_array,'r')
plt.show()
print("Best k parameter is ",k_min )
```

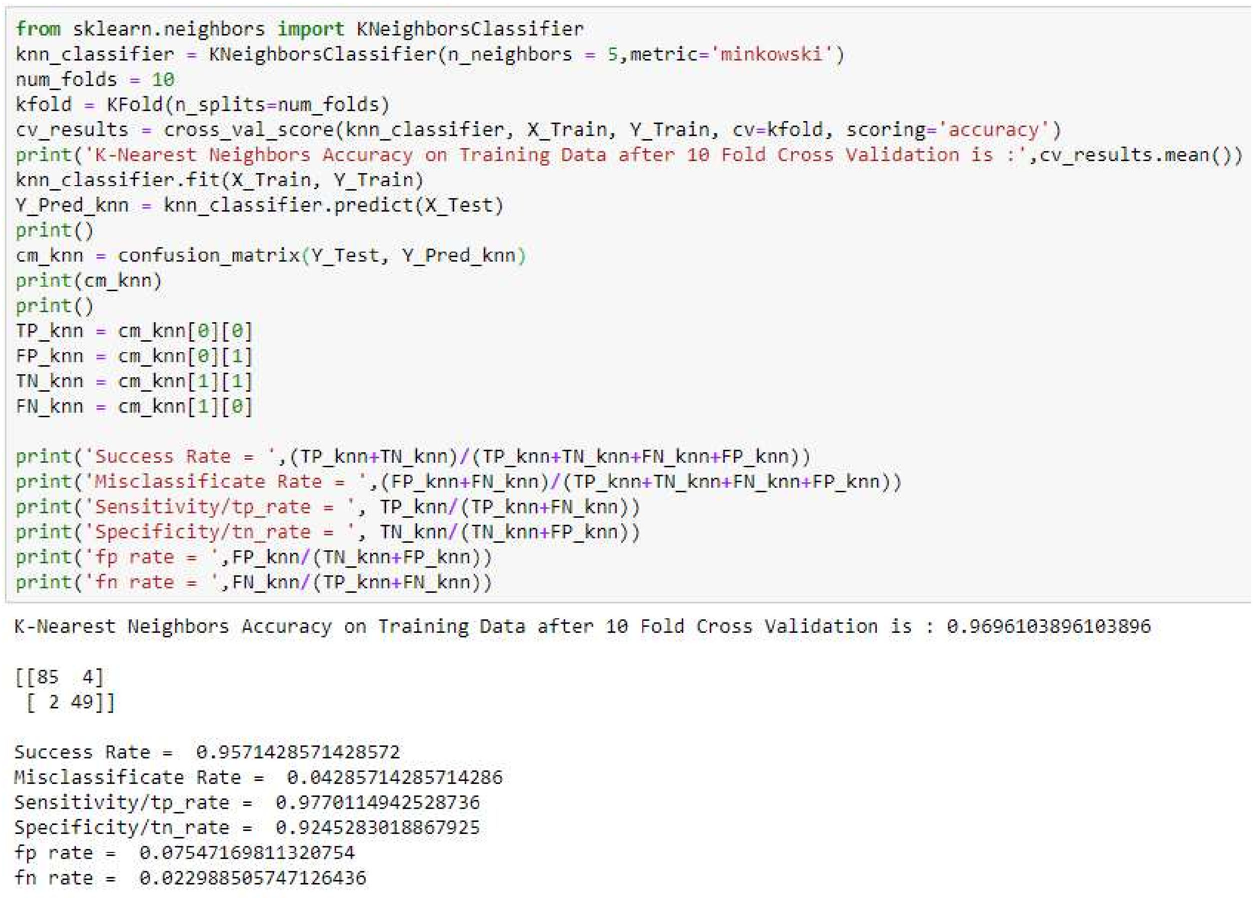
The above code snippet plots the value of Mean Absolute Error obtained each time the machine is trained with a different value of number of neighbors. The number of neighbors is varied from 2 to 560 (2 being the value just greater than 1 and 560 being the maximum number of samples present in training data). Following is the graph obtained-



```
Best k parameter is  5
```

The graph clearly shows that for number of neighbors = 5, least MAE is observed. Hence we use 5 as the number of neighbors in training the model. Euclidean distance is sensitive to outliers and if data comprises of outliers, a preferred choice is the use of robust distance like Minkowski distance.

The similarity/distance function hence being used in this case is the Minkowski Distance because of the presence of several outliers as seen in the scatter plots.



```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors = 5,metric='minkowski')
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(knn_classifier, X_Train, Y_Train, cv=kfold, scoring='accuracy')
print('K-Nearest Neighbors Accuracy on Training Data after 10 Fold Cross Validation is :',cv_results.mean())
knn_classifier.fit(X_Train, Y_Train)
Y_Pred_knn = knn_classifier.predict(X_Test)
print()
cm_knn = confusion_matrix(Y_Test, Y_Pred_knn)
print(cm_knn)
print()
TP_knn = cm_knn[0][0]
FP_knn = cm_knn[0][1]
TN_knn = cm_knn[1][1]
FN_knn = cm_knn[1][0]

print('Success Rate = ',(TP_knn+TN_knn)/(TP_knn+TN_knn+FN_knn+FP_knn))
print('Misclassificate Rate = ',(FP_knn+FN_knn)/(TP_knn+TN_knn+FN_knn+FP_knn))
print('Sensitivity/tp_rate = ', TP_knn/(TP_knn+FN_knn))
print('Specificity/tn_rate = ', TN_knn/(TN_knn+FP_knn))
print('fp rate = ',FP_knn/(TN_knn+FP_knn))
print('fn rate = ',FN_knn/(TP_knn+FN_knn))
```

```
K-Nearest Neighbors Accuracy on Training Data after 10 Fold Cross Validation is : 0.9696103896103896

[[85  4]
 [ 2 49]]

Success Rate =  0.9571428571428572
Misclassificate Rate =  0.04285714285714286
Sensitivity/tp_rate =  0.9770114942528736
Specificity/tn_rate =  0.9245283018867925
fp rate =  0.07547169811320754
fn rate =  0.022988505747126436
```

3.2.4 Logistic Regression Model

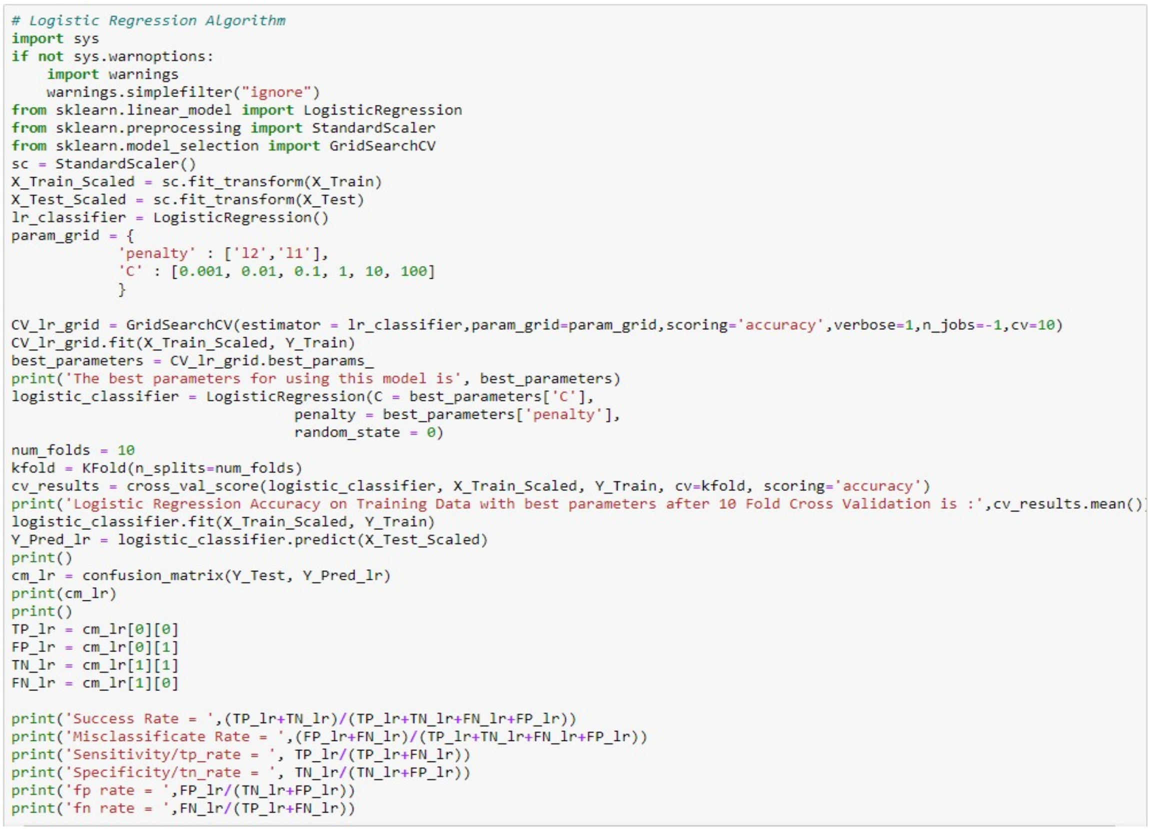
Logistic Regression is a supervised machine learning technique, employed in classification jobs (for predictions based on training data). Logistic Regression uses an equation similar to Linear Regression but the outcome of logistic regression is a categorical variable whereas it is a value for other regression models. Binary outcomes can be predicted from the independent variables. The outcome of dependent variable is discrete. Logistic Regression uses a simple equation which shows the linear relation between the independent variables. These independent variables along with their coefficients are united linearly to form a linear equation that is used to predict the output [57]. This algorithm is entitled as logistic regression as the key method behind it is logistic function. The output can be predicted from the independent variables, which form a linear equation. The output predicted has no restrictions, it can be any value from negative infinity to positive infinity. But the output required is a class variable (i.e., yes or no, 1 or 0). So, the outcome of the linear equation should be flattened into a small range (i.e [0,1]). Logistic function is used here to suppress the outcome value between 0 and 1 [58]. Logistic function can also be called cost function. Logistic regression measures the link between the variable quantity, the output, and therefore the freelance variables, the input.

One of the observed features of Logistic regression is that, it gives better results when data is standardized. Feature standardization makes the values of each feature in the data have zero-mean (when subtracting the mean in the numerator) and unit-variance. The general method of calculation is to determine the distribution mean and standard deviation for each feature. Next we subtract the mean from each feature. Then we divide the values (mean is already subtracted) of each feature by its standard deviation. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data.

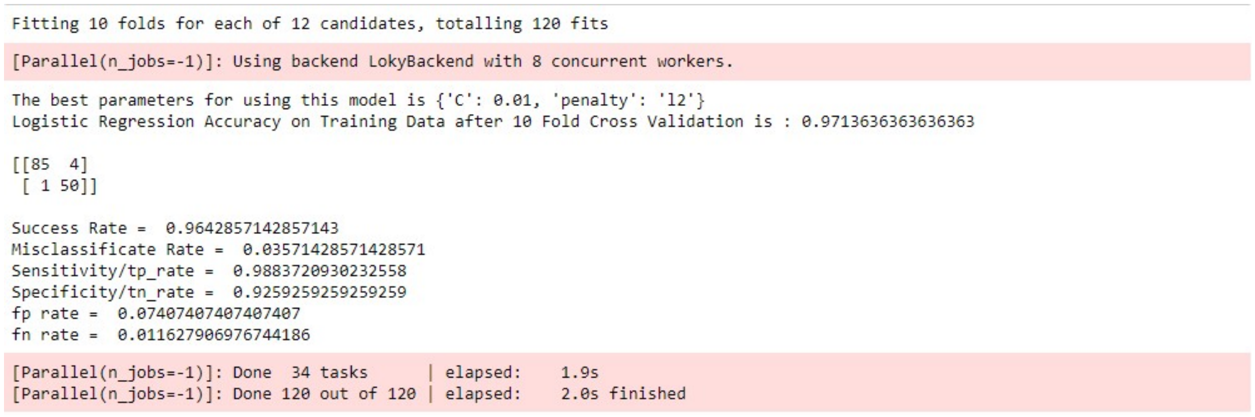
Logistic regression has several parameters that can be tuned (optimized) in order to achieve the maximum testing accuracy. The parameters used in this report to achieve the maximum testing accuracy are –

- **Parameter C** – Regularization Parameter - Defined as the inverse of regularization strength (or lambda). Lambda (λ) controls the trade-off between allowing the model to increase its complexity as much as it wants with trying to keep it simple. For example, if λ is very low or 0, the model will have enough power to increase its complexity (over fit) by assigning big values to the weights for each parameter. If, in the other hand, we increase the value of λ, the model will tend to under fit, as the model will become too simple. Parameter C will work the other way around. For small values of C, we increase the regularization strength which will create simple models which under fit the data. For big values of C, we low the power of regularization which implies the model is allowed to increase its complexity, and therefore, over fit the data. In order to achieve the best fit, parameter C has to be tuned/optimized.
- **Penalty** – Attribute 'Penalty' when used with Logistic regression specifies the norm used in penalization. The two types of penalties used in this report are l1 and l2. l1 signifies **Lasso Regression** whereas l2 signifies **Ridge Regression**. Ridge regression adds "squared magnitude" of coefficient as penalty term to the loss function whereas Lasso Regression adds "absolute magnitude" of coefficient as penalty term to the loss function. The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether and hence works well for **feature selection** in case we have a huge number of features.

The above two parameters were optimized using GridSearchCV and the best parameters were obtained. The testing accuracy of the model with the best parameters was then calculated.



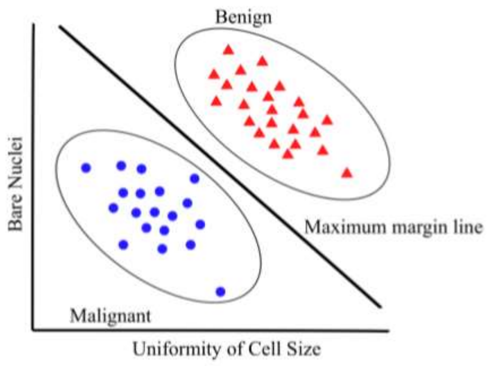
```python
# Logistic Regression Algorithm
import sys
if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
sc = StandardScaler()
X_Train_Scaled = sc.fit_transform(X_Train)
X_Test_Scaled = sc.fit_transform(X_Test)
lr_classifier = LogisticRegression()
param_grid = {
            'penalty' : ['l2','l1'],
            'C' : [0.001, 0.01, 0.1, 1, 10, 100]
            }

CV_lr_grid = GridSearchCV(estimator = lr_classifier,param_grid=param_grid,scoring='accuracy',verbose=1,n_jobs=-1,cv=10)
CV_lr_grid.fit(X_Train_Scaled, Y_Train)
best_parameters = CV_lr_grid.best_params_
print('The best parameters for using this model is', best_parameters)
logistic_classifier = LogisticRegression(C = best_parameters['C'],
                                penalty = best_parameters['penalty'],
                                random_state = 0)
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(logistic_classifier, X_Train_Scaled, Y_Train, cv=kfold, scoring='accuracy')
print('Logistic Regression Accuracy on Training Data with best parameters after 10 Fold Cross Validation is :',cv_results.mean()
logistic_classifier.fit(X_Train_Scaled, Y_Train)
Y_Pred_lr = logistic_classifier.predict(X_Test_Scaled)
print()
cm_lr = confusion_matrix(Y_Test, Y_Pred_lr)
print(cm_lr)
print()
TP_lr = cm_lr[0][0]
FP_lr = cm_lr[0][1]
TN_lr = cm_lr[1][1]
FN_lr = cm_lr[1][0]

print('Success Rate = ',(TP_lr+TN_lr)/(TP_lr+TN_lr+FN_lr+FP_lr))
print('Misclassificate Rate = ',(FP_lr+FN_lr)/(TP_lr+TN_lr+FN_lr+FP_lr))
print('Sensitivity/tp_rate = ', TP_lr/(TP_lr+FN_lr))
print('Specificity/tn_rate = ', TN_lr/(TN_lr+FP_lr))
print('fp rate = ',FP_lr/(TN_lr+FP_lr))
print('fn rate = ',FN_lr/(TP_lr+FN_lr))
```



```
Fitting 10 folds for each of 12 candidates, totalling 120 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
The best parameters for using this model is {'C': 0.01, 'penalty': 'l2'}
Logistic Regression Accuracy on Training Data after 10 Fold Cross Validation is : 0.9713636363636363

[[85  4]
 [ 1 50]]

Success Rate =  0.9642857142857143
Misclassificate Rate =  0.03571428571428571
Sensitivity/tp_rate =  0.9883720930232558
Specificity/tn_rate =  0.9259259259259259
fp rate =  0.07407407407407407
fn rate =  0.011627906976744186
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:    1.9s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:    2.0s finished
```

3.2.5 Support Vector Machines Model

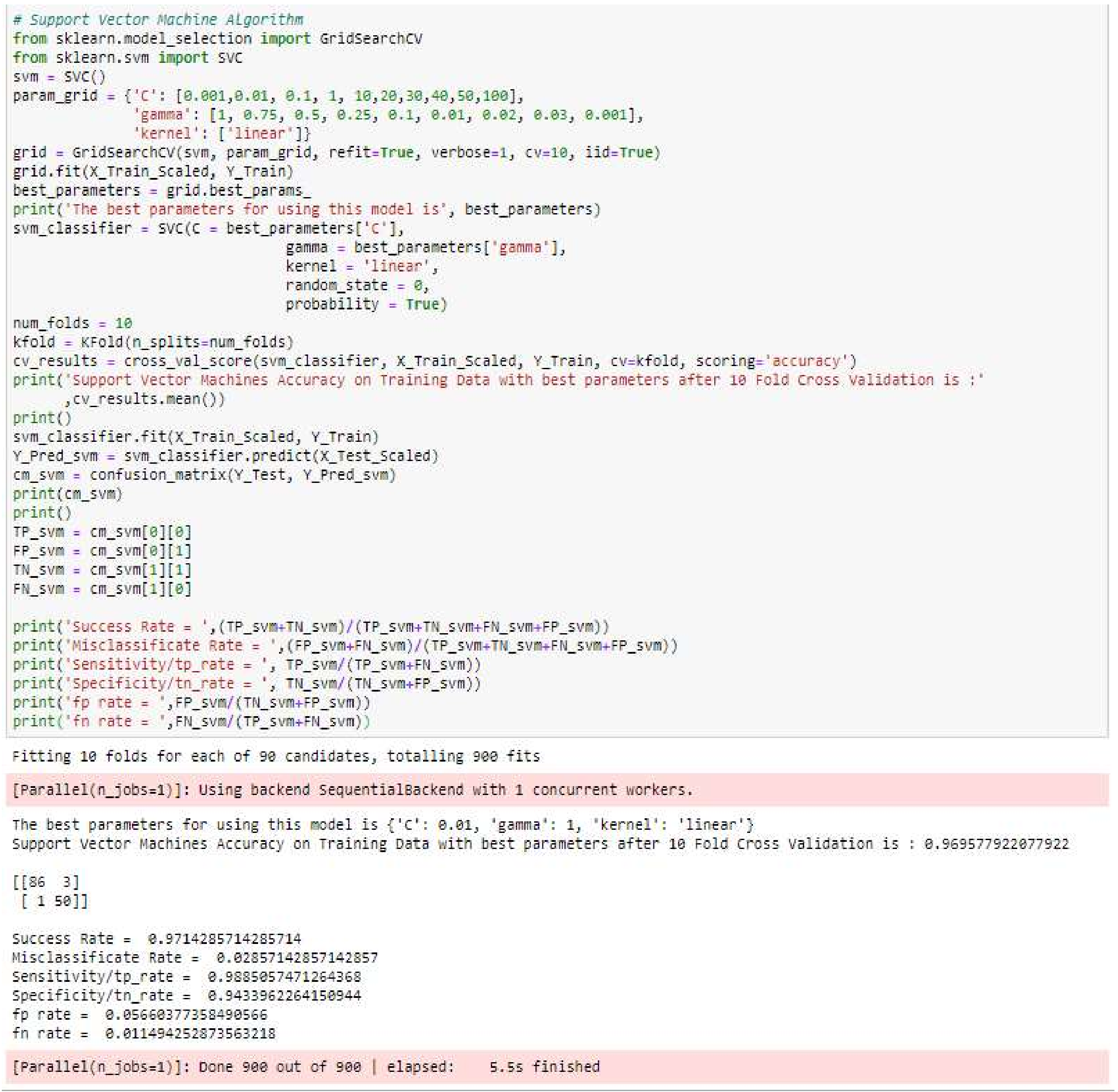
Support Vector Machine (SVM) is a supervised machine learning algorithmic rule which might be used for each classification or regression challenges. However, it's principally utilized in classification issues. In this algorithmic rule, we plot each data item as a point in n-dimensional space where n is number of features one has with the value of each feature being the value of a particular coordinate [59]. Then, we perform classification by finding the hyper-plane that differentiates the two classes. Often researchers tend to plot every knowledge item as some extent in n-dimensional area with the worth of every feature being the worth of a selected coordinate. Then, to perform classification by finding the hyper-plane that differentiate the 2 categories fine. It is a non-probabilistic binary linear classifier, however are often manipulated during a manner that it will perform non-linear and probabilistic classification also, creating it versatile algorithmic program. AN SVM model could be an illustration of the instances as points in area mapped, so they will be categorized and divided by a transparent gap. New instances are then mapped into the identical area and foreseen that within which class it would be supported which aspect of the gap they fall in. the most advantages of SVM is that the indisputable fact that it's effective in high dimensional areas [60-61]. When data are not labeled, supervised learning is not possible, and an unsupervised learning approach is required, which attempts to find natural clustering of the data to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering [62]. The main idea in SVM is to identify the optimal separating hyperplane which maximizes the margin of the training data.



Even in case of Support Vector Machines there are several parameters whose tuning/optimization can greatly affect the testing accuracy. Some of the parameters of Support Vector Machines that have been tuned/ optimized in this project are –

- **Parameter C – Regularization Parameter** – Same as defined for Logistic Regression.

- **Sigma** - The gamma parameter in the RBF kernel determines the reach of a single training instance. If the value of Gamma is low, then every training instance will have a far reach. Conversely, high values of gamma mean that training instances will have a close reach. So, with a high value of gamma, the SVM decision boundary will simply be dependent on just the points that are closest to the decision boundary, effectively ignoring points that are farther away. In comparison, a low value of gamma will result in a decision boundary that will consider points that are further from it. As a result, high values of gamma typically produce highly flexed decision boundaries, and low values of gamma often results in a decision boundary that is more linear.
- **Kernel** - SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid. The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces**.** Linear Kernel is used when the data is linearly separable, that is, it can be separated using a single line. It is one of the most common kernels to be used. In this project only 'Linear' kernel has been used.



```
# Support Vector Machine Algorithm
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
svm = SVC()
param_grid = {'C': [0.001,0.01, 0.1, 1, 10,20,30,40,50,100],
              'gamma': [1, 0.75, 0.5, 0.25, 0.1, 0.01, 0.02, 0.03, 0.001],
              'kernel': ['linear']}
grid = GridSearchCV(svm, param_grid, refit=True, verbose=1, cv=10, iid=True)
grid.fit(X_Train_Scaled, Y_Train)
best_parameters = grid.best_params_
print('The best parameters for using this model is', best_parameters)
svm_classifier = SVC(C = best_parameters['C'],
                              gamma = best_parameters['gamma'],
                              kernel = 'linear',
                              random_state = 0,
                              probability = True)
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(svm_classifier, X_Train_Scaled, Y_Train, cv=kfold, scoring='accuracy')
print('Support Vector Machines Accuracy on Training Data with best parameters after 10 Fold Cross Validation is :'
      ,cv_results.mean())
print()
svm_classifier.fit(X_Train_Scaled, Y_Train)
Y_Pred_svm = svm_classifier.predict(X_Test_Scaled)
cm_svm = confusion_matrix(Y_Test, Y_Pred_svm)
print(cm_svm)
print()
TP_svm = cm_svm[0][0]
FP_svm = cm_svm[0][1]
TN_svm = cm_svm[1][1]
FN_svm = cm_svm[1][0]

print('Success Rate = ',(TP_svm+TN_svm)/(TP_svm+TN_svm+FN_svm+FP_svm))
print('Misclassificate Rate = ',(FP_svm+FN_svm)/(TP_svm+TN_svm+FN_svm+FP_svm))
print('Sensitivity/tp_rate = ', TP_svm/(TP_svm+FN_svm))
print('Specificity/tn_rate = ', TN_svm/(TN_svm+FP_svm))
print('fp rate = ',FP_svm/(TN_svm+FP_svm))
print('fn rate = ',FN_svm/(TP_svm+FN_svm))
```

```
Fitting 10 folds for each of 90 candidates, totalling 900 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

The best parameters for using this model is {'C': 0.01, 'gamma': 1, 'kernel': 'linear'}
Support Vector Machines Accuracy on Training Data with best parameters after 10 Fold Cross Validation is : 0.969577922077922

[[86  3]
 [ 1 50]]

Success Rate =  0.9714285714285714
Misclassificate Rate =  0.02857142857142857
Sensitivity/tp_rate =  0.9885057471264368
Specificity/tn_rate =  0.9433962264150944
fp rate =  0.05660377358490566
fn rate =  0.011494252873563218

[Parallel(n_jobs=1)]: Done 900 out of 900 | elapsed:    5.5s finished
```

3.2.6 Decision Tree Model

In Machine Learning, Decision Trees is a predictive model that represents a mapping between object attributes and object values [43, 44]. A Decision Tree is a tree-like classifier that partitions every possible outcome of data recursively into classes. Decision Trees is similar to the flowchart, in which every non-leaf node indicates a test on a particular attribute, every branch represents an outcome of that test and every leaf node expresses a classification or decision. The node at the topmost label in the tree is called root node, which corresponds to the best predictor. By using Decision Trees, both numerical and categorical data can be processed. Based on maximum information gain, decision-makers can choose best alternative and traversal from root node to leaf nodes denoting unique class separation [63]. The most popular Decision Trees methods are Iterative Dichotomiser 3 (ID3) [64], C4.5 [65], C5.0 and classification and regression tree (CART) which use entropy-based measures to grow the tree. Some of the common terms associated with Decision Trees are-

Information Gain - When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy.

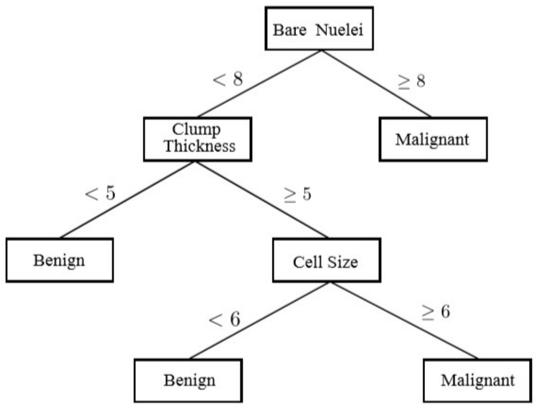
Entropy - Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.

Gini Index - Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified. It means an attribute with lower Gini Index should be preferred.

The most notable types of decision tree algorithms are thus explained:-

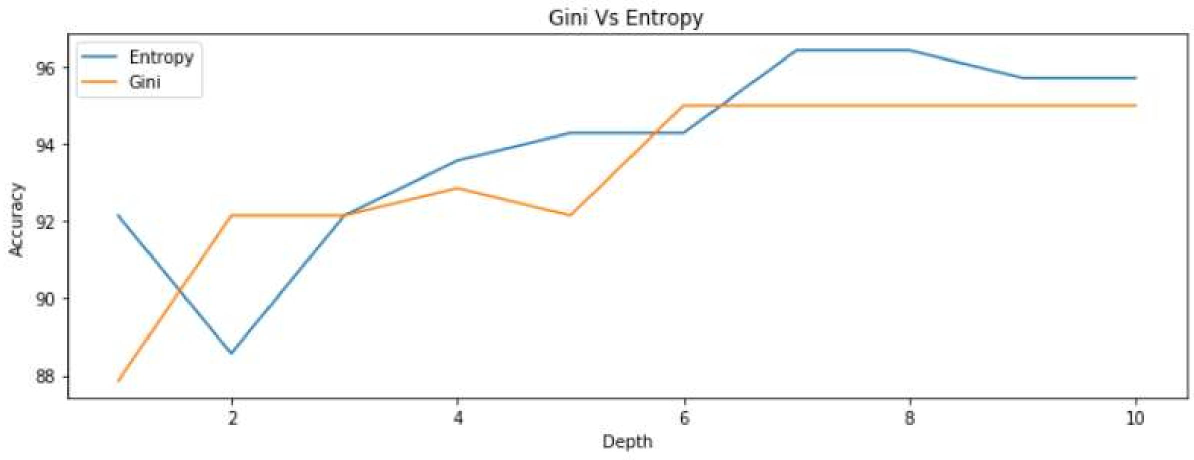
1. **ID3 Decision Tree** - This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.
2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
3. **Classification and Regression Tree (CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

To understand how Decision Trees works in the context of Breast Cancer diagnosis and prognosis, an example of a BC diagnosis and prognosis decision tree structure is given which is used to predict whether a person has malignant or benign BC.



In this project the default Decision Tree Classifier (CART) has been used for which the maximum depth and criterion (Entropy or Gini) were tuned in order to achieve the best testing accuracy.

The following graph shows the variation in accuracy as and when the maximum depth of decision tree is increased, for both Entropy and Gini -

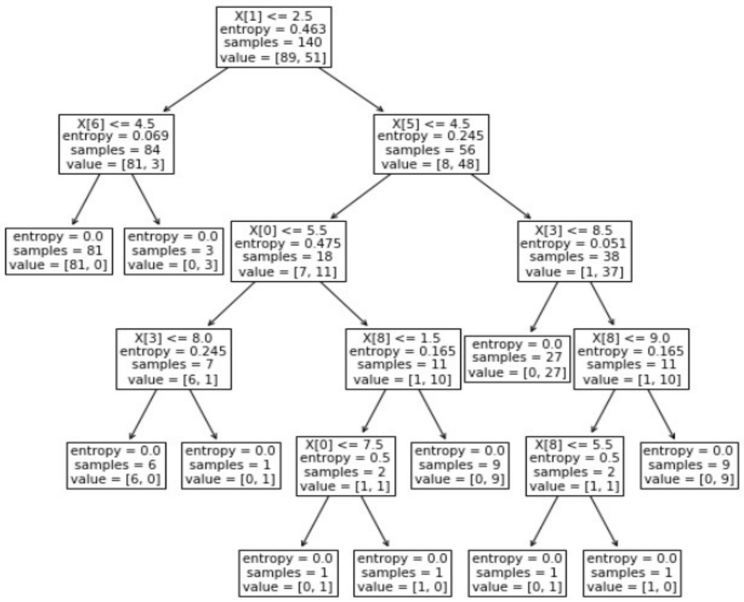


```
Best Criterion: Entropy, Accuracy 96.43% at depth = 7
```

From the graph it can be clearly seen that Entropy for a Maximum Depth of 7 gives the best training accuracy. The classifier is then trained with these best parameters and testing accuracy is calculated.

```
Decision Trees Accuracy on Training Data with best parameters after 10 Fold Cross Validation is : 0.9516883116883117

[[86  3]
 [ 2 49]]

Success Rate =  0.9642857142857143
Misclassificate Rate =  0.03571428571428571
Sensitivity/tp_rate =  0.9772727272727273
Specificity/tn_rate =  0.9423076923076923
fp rate =  0.057692307692307696
fn rate =  0.022727272727272728
```

Decision tree for the test data is as follows –



3.2.7 Random Forests Model

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees using a clever idea. The goal is to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). Random forest is another ensemble method based on decision trees. It split data into sub-samples, trains decision tree classifiers on each sub-sample and averages prediction of each classifier. Splitting dataset causes higher bias but it is compensated by large decrease in variance. Random Forest is a supervised learning algorithm and it is flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and the fact that it can be used for both classification and regression tasks. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process. It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases. The algorithm can be used in both classification and regression problems. Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values. You can get the relative feature importance, which helps in selecting the most contributing features for the classifier. Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming. The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Random Forest vs Decision Trees –

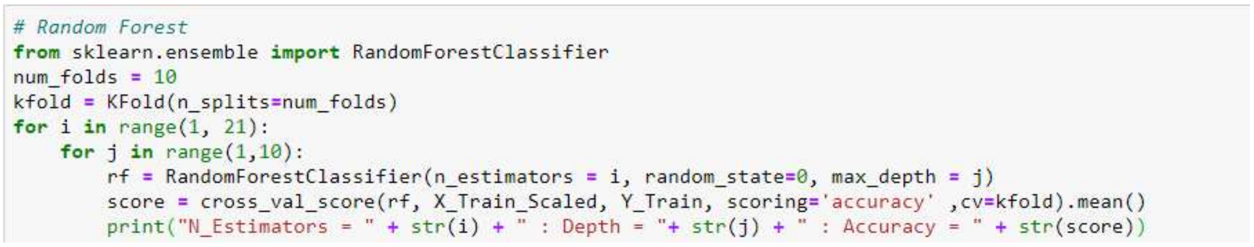
- Random forests is a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

Random forests also offers a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1. This score helps in choosing the most important features and drop the least important ones for model building. Random forest uses Gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

Even without parameter tuning, the Random Forest Classifier is capable of giving good results. In order to achieve the best results the Number of Estimators and Maximum Depth of a tree were optimized.

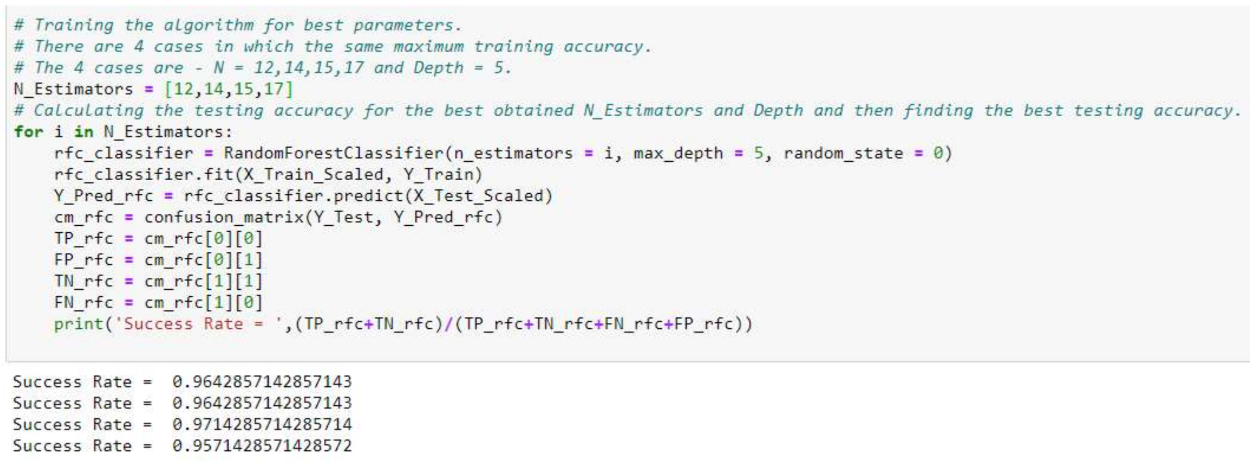
- Number of Estimators defines the number of trees in the forest.
- Maximum Depth defines the maximum depth each tree can go to or it can also be described as the length of the longest path from the tree root to a leaf. The root node is considered to have a depth of 0.

For each tree in the forest, the depth was varied and along with it, the number of trees in the forest are also varied so as to achieve the maximum training accuracy and best parameters.
The model is then trained according to the best parameter and the testing accuracy is calculated.



```python
# Random Forest
from sklearn.ensemble import RandomForestClassifier
num_folds = 10
kfold = KFold(n_splits=num_folds)
for i in range(1, 21):
    for j in range(1,10):
        rf = RandomForestClassifier(n_estimators = i, random_state=0, max_depth = j)
        score = cross_val_score(rf, X_Train_Scaled, Y_Train, scoring='accuracy' ,cv=kfold).mean()
        print("N_Estimators = " + str(i) + " : Depth = "+ str(j) + " : Accuracy = " + str(score))
```

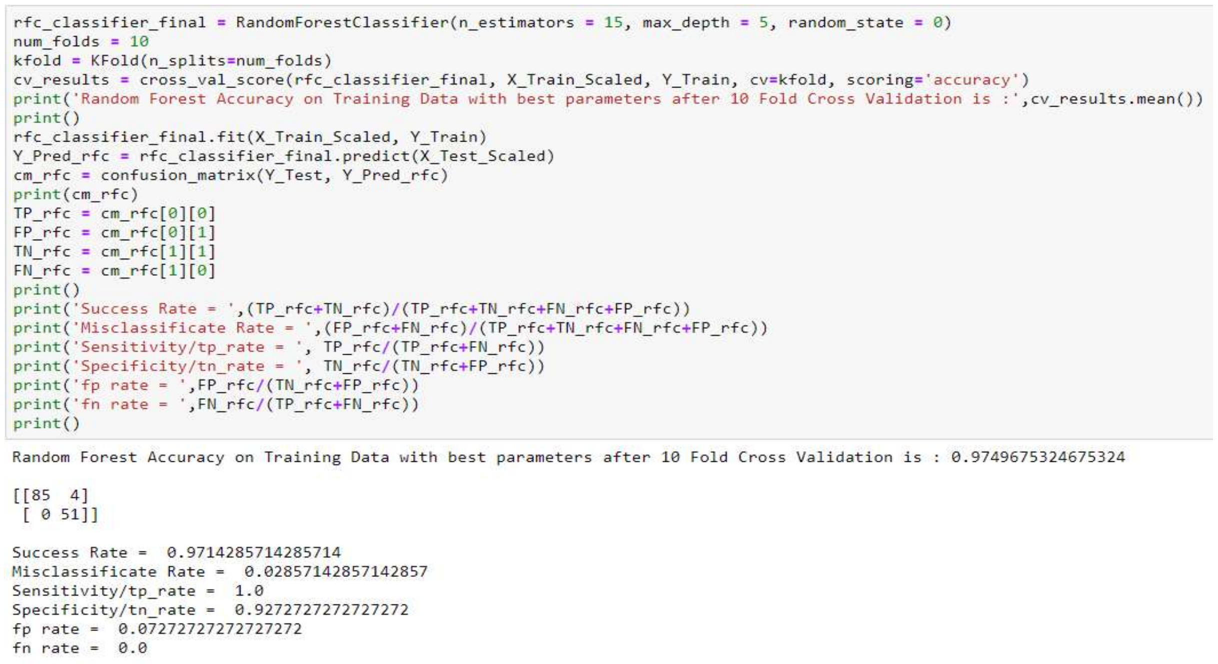
The above code snippet calculated the training accuracy for different values of n_estimators and max_depth. By eyeballing, the best training accuracies obtained were chosen. It was observed that best training accuracy was obtained for max_depth = 5. But for the same depth, there were 4 different values of n_estimators which gave the same training accuracy. For n_estimators = 12, 14, 15 and 17 and max_depth = 5, same training accuracies were obtained. The model then was trained four times for different n_estimators and constant max_depth and their testing accuracies were obtained.



```python
# Training the algorithm for best parameters.
# There are 4 cases in which the same maximum training accuracy.
# The 4 cases are - N = 12,14,15,17 and Depth = 5.
N_Estimators = [12,14,15,17]
# Calculating the testing accuracy for the best obtained N_Estimators and Depth and then finding the best testing accuracy.
for i in N_Estimators:
    rfc_classifier = RandomForestClassifier(n_estimators = i, max_depth = 5, random_state = 0)
    rfc_classifier.fit(X_Train_Scaled, Y_Train)
    Y_Pred_rfc = rfc_classifier.predict(X_Test_Scaled)
    cm_rfc = confusion_matrix(Y_Test, Y_Pred_rfc)
    TP_rfc = cm_rfc[0][0]
    FP_rfc = cm_rfc[0][1]
    TN_rfc = cm_rfc[1][1]
    FN_rfc = cm_rfc[1][0]
    print('Success Rate = ',(TP_rfc+TN_rfc)/(TP_rfc+TN_rfc+FN_rfc+FP_rfc))
```

```
Success Rate =  0.9642857142857143
Success Rate =  0.9642857142857143
Success Rate =  0.9714285714285714
Success Rate =  0.9571428571428572
```

It was observed that for n_estimators = 15 and max_depth = 5, the best testing accuracy was obtained.

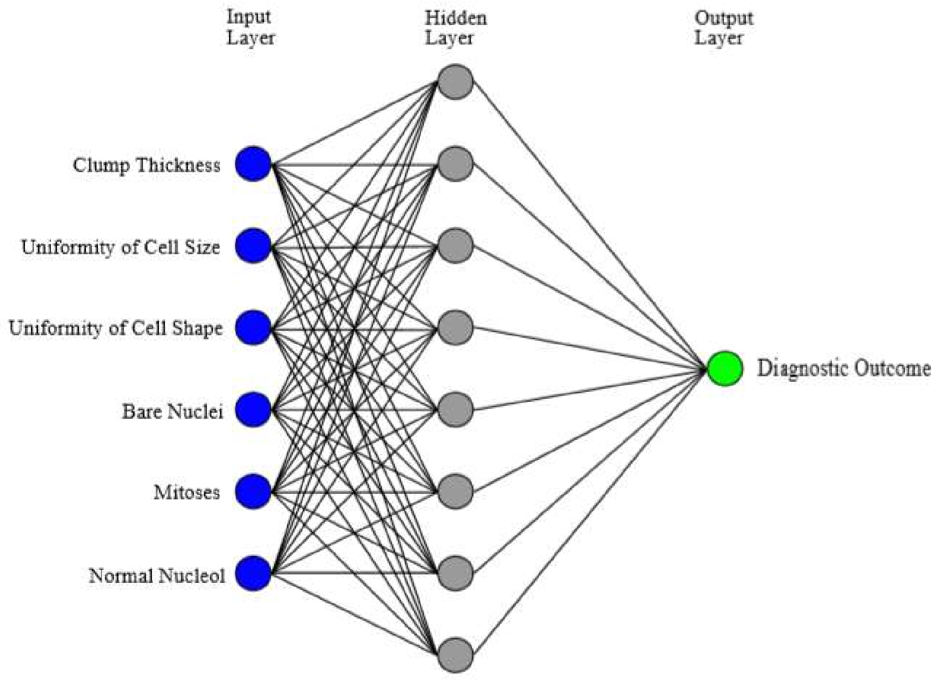


```python
rfc_classifier_final = RandomForestClassifier(n_estimators = 15, max_depth = 5, random_state = 0)
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(rfc_classifier_final, X_Train_Scaled, Y_Train, cv=kfold, scoring='accuracy')
print('Random Forest Accuracy on Training Data with best parameters after 10 Fold Cross Validation is :',cv_results.mean())
print()
rfc_classifier_final.fit(X_Train_Scaled, Y_Train)
Y_Pred_rfc = rfc_classifier_final.predict(X_Test_Scaled)
cm_rfc = confusion_matrix(Y_Test, Y_Pred_rfc)
print(cm_rfc)
TP_rfc = cm_rfc[0][0]
FP_rfc = cm_rfc[0][1]
TN_rfc = cm_rfc[1][1]
FN_rfc = cm_rfc[1][0]
print()
print('Success Rate = ',(TP_rfc+TN_rfc)/(TP_rfc+TN_rfc+FN_rfc+FP_rfc))
print('Misclassificate Rate = ',(FP_rfc+FN_rfc)/(TP_rfc+TN_rfc+FN_rfc+FP_rfc))
print('Sensitivity/tp_rate = ', TP_rfc/(TP_rfc+FN_rfc))
print('Specificity/tn_rate = ', TN_rfc/(TN_rfc+FP_rfc))
print('fp rate = ',FP_rfc/(TN_rfc+FP_rfc))
print('fn rate = ',FN_rfc/(TP_rfc+FN_rfc))
print()
```

```
Random Forest Accuracy on Training Data with best parameters after 10 Fold Cross Validation is : 0.9749675324675324

[[85  4]
 [ 0 51]]

Success Rate =  0.9714285714285714
Misclassificate Rate =  0.02857142857142857
Sensitivity/tp_rate =  1.0
Specificity/tn_rate =  0.9272727272727272
fp rate =  0.07272727272727272
fn rate =  0.0
```

3.2.8 Artificial Neural Networks Model

Artificial Neural Networks (NN) are a type of mathematical algorithms originating in the simulation of networks of biological neurons. An artificial Neural Network consists of nodes (called neurons) and edges (called synapses). Input data is transmitted through the weighted synapses to the neurons where calculations are processed and then either sent to further neurons or represent the output. Neural Networks take in the weights of connections between neurons. The weights are balanced, learning data point in the wake of learning data point. When all weights are trained, the neural network can be utilized to predict the class or a quantity, if there should arise an occurrence of regression of a new input data point. With Neural networks, extremely complex models can be trained and they can be utilized as a kind of black box, without playing out an unpredictable complex feature engineering before training the model. Joined with the "deep approach" even more unpredictable models can be picked up to realize new possibilities. Neural networks normally have two at least two layer of neurons, with first layer neurons having nonlinear and differentiable activation functions. Such networks can approximate any non- linear function. In real life, we are faced with nonlinear problems, and multilayer neural network structures have the capability of providing solutions to these problems. Neural Networks possessing several layers are known as Multi-Layer Perceptron (MLP); their computing power is meaningfully improved over the one layer NN. A typical neural network consist of an input layer, an output layer and hidden layers. A Multi-Layer Perceptron (MLP) can also be regarded as a feedforward neural network with one or more hidden layers. Each hidden layer has its own specific function. Input terminals accept input signal from outside world and redistribute these signals to all the neurons in a hidden layer. The output layer accepts a stimulus pattern from a hidden layer and establishes the output pattern on the entire network. Neurons in the hidden layers perform transformations of input attributes; the weights of the neurons represent the features in the transformed domain. These features are then used by the output layer in determining the output pattern [*] A simple example of how ANN is trained to predict the diagnostic outcome from six inputs and one hidden layer with 8 neurons is as follows.



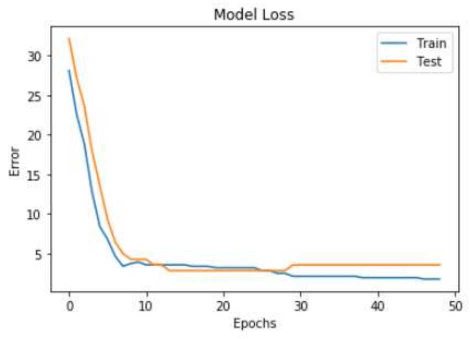
In this project, ANN is trained to predict the outcome from nine inputs (equaling 9 features), no hidden layers and 2 outputs (binary classification). Further, the learning rate and maximum iterations (epochs) are optimized to obtain the best training accuracy.

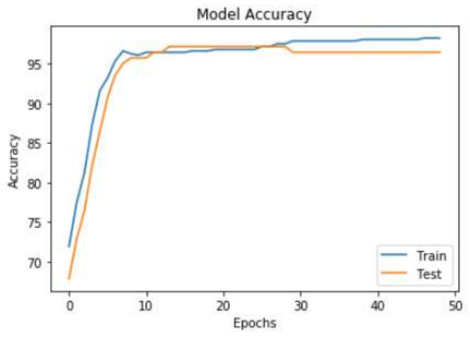
The learning rate is how quickly a network abandons old beliefs for new ones. A higher learning rate means the network changes its mind more quickly. That can be good in some cases, but can also be bad. If the learning rate is too high, it might give abrupt results. In general, we find a learning rate that is low enough that the network converges to something useful, but high enough that we don't have to spend years training it.

Epochs/Maximum Iterations is defined as the number of times the algorithm sees the entire data set. One forward pass and one backward pass of all the training examples is defined as an epoch.

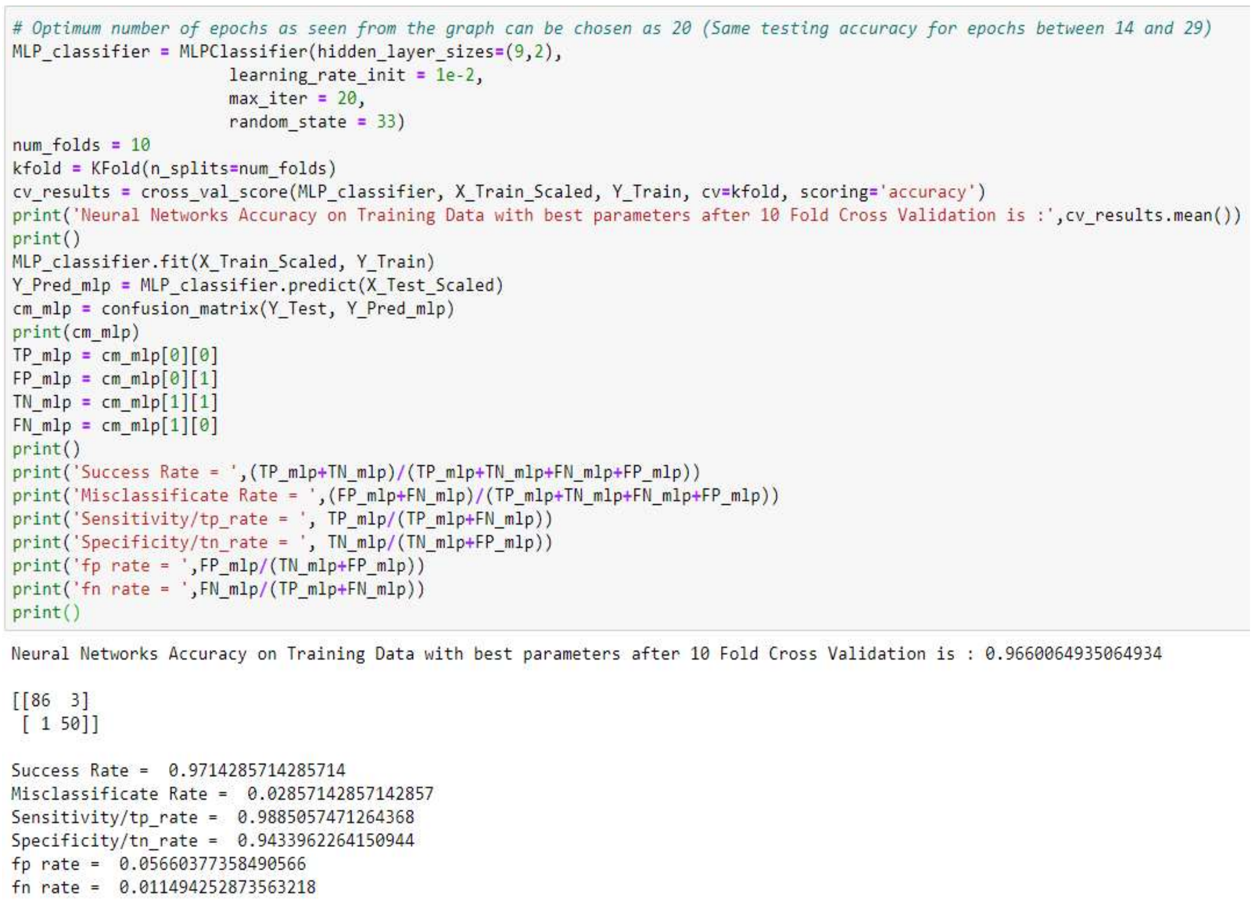
```python
# Neural Networks
from sklearn.neural_network import MLPClassifier
acc = []
learning_rate = [1e-5,1e-4,1e-3,1e-2,1e-1,1]
for i in range(10,90,10):
    for j in learning_rate:
        nn = MLPClassifier(hidden_layer_sizes=(9,2),
                      learning_rate_init = j,
                      max_iter = i,
                      random_state = 33)
        score = cross_val_score(nn, X_Train_Scaled, Y_Train, scoring='accuracy' ,cv=kfold).mean()
        print("Epochs/Number_Of_Iterations = " + str(i) + " : Learning_Rate = "+ str(j) + " : Accuracy = " + str(score))
```

The above code snippet prints training accuracy for various values of learning rate and number of epochs. After eyeballing, it was observed that for learning rate = 1e-2, the best training accuracy was obtained. The number of epochs which gave the best accuracy with learning rate being 1e-2 were 40 and 50. But since there is a possibility that with a constant learning rate, increasing number of epochs might lead to overfitting of data which can lead to poor generalization of the model (poor testing accuracy). Hence, keeping the learning rate at 1e-2, the number of epochs were varied within a certain range (up to 50 epochs) so as to determine the best testing accuracy. The graphs displaying the variation of training and testing accuracy and training and testing error thus obtained are as follows –





The above graphs clearly indicate that as the number of epochs increase, the training accuracy and the testing accuracy keeps on increasing. But after a certain number of epochs, the testing accuracy starts to degrade but the training accuracy keeps on increasing. This clearly implies that with increasing number of epochs, the model over fits the data giving a higher training accuracy but a lower testing accuracy. The optimal number of epochs is that value of epochs after which the training accuracy starts to degrade. Hence, for this model any value of number of epochs between 14 and 29 give the same testing accuracy.



```
# Optimum number of epochs as seen from the graph can be chosen as 20 (Same testing accuracy for epochs between 14 and 29)
MLP_classifier = MLPClassifier(hidden_layer_sizes=(9,2),
                      learning_rate_init = 1e-2,
                      max_iter = 20,
                      random_state = 33)
num_folds = 10
kfold = KFold(n_splits=num_folds)
cv_results = cross_val_score(MLP_classifier, X_Train_Scaled, Y_Train, cv=kfold, scoring='accuracy')
print('Neural Networks Accuracy on Training Data with best parameters after 10 Fold Cross Validation is :',cv_results.mean())
print()
MLP_classifier.fit(X_Train_Scaled, Y_Train)
Y_Pred_mlp = MLP_classifier.predict(X_Test_Scaled)
cm_mlp = confusion_matrix(Y_Test, Y_Pred_mlp)
print(cm_mlp)
TP_mlp = cm_mlp[0][0]
FP_mlp = cm_mlp[0][1]
TN_mlp = cm_mlp[1][1]
FN_mlp = cm_mlp[1][0]
print()
print('Success Rate = ',(TP_mlp+TN_mlp)/(TP_mlp+TN_mlp+FN_mlp+FP_mlp))
print('Misclassificate Rate = ',(FP_mlp+FN_mlp)/(TP_mlp+TN_mlp+FN_mlp+FP_mlp))
print('Sensitivity/tp_rate = ', TP_mlp/(TP_mlp+FN_mlp))
print('Specificity/tn_rate = ', TN_mlp/(TN_mlp+FP_mlp))
print('fp rate = ',FP_mlp/(TN_mlp+FP_mlp))
print('fn rate = ',FN_mlp/(TP_mlp+FN_mlp))
print()
```

```
Neural Networks Accuracy on Training Data with best parameters after 10 Fold Cross Validation is : 0.9660064935064934

[[86  3]
 [ 1 50]]

Success Rate =  0.9714285714285714
Misclassificate Rate =  0.02857142857142857
Sensitivity/tp_rate =  0.9885057471264368
Specificity/tn_rate =  0.9433962264150944
fp rate =  0.05660377358490566
fn rate =  0.011494252873563218
```

Chapter 4

Result Analysis

The next step after applying implementing machine learning models is to seek out how effective is that the model, i.e. how the models performed on the datasets. This is carried out by running the models on the test dataset which was set earlier. The test dataset comprised of 20% of the dataset for Breast Cancer prediction. In order to determine and compare the performances of the different algorithms, several metrics have been used.

4.1 Performance Metrics

Several performance metrics have been used to figure out the performance of the Machine Learning algorithms. As the report sincerely deals with classification problems, performance metrics relating to classifications are discussed here. For Breast Cancer prediction, if the target variable is 1(malignant), then it is a positive instance, meaning the patient has Breast cancer. And if the target variable is 0 (benign), then it is a negative instance, stating that the patient does not have the cancer.

4.1.1 Confusion Matrix

Summarization the performance of a classification algorithm is based on a technique which is known as confusion matrix. It is arguably the easiest way to regulate the performance of a classification model by comparing how many positive instances are correctly/incorrectly classified and how many negative instances are correctly/incorrectly classified. In a confusion matrix, as shown here, the rows represent the actual labels while the columns represent the predicted labels.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

True Positives (TP): These are the occurrences where both the predictive and actual class is true (1), i.e., when the patient has complications (breast cancer in this case) and is also classified by the model to have complications.

True Negatives (TN): True negatives are the occurrences where both the predicted class and actual class is False (0), i.e., when a patient does not have complications and is also classified by the model as not having complications.

False Negative (FN): These are occurrences where the predicted class is False (0) but actual class is True (1), i.e., case of a patient being classified by the model as not having complications even though in reality, they do.

False Positive (FP): False positives are the occurrences where the predicted class is True (1) while the actual class is False (0), i.e., when a patient is classified by the model as having complications even though in reality, they do not.

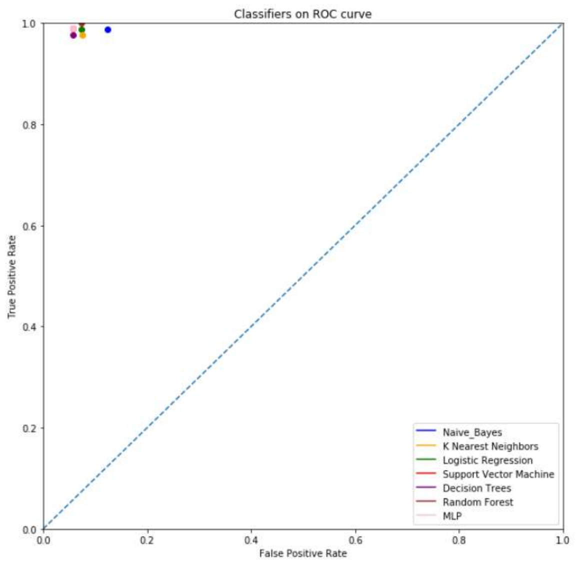
Accuracy/Success Rate: Evaluation of classification models is done by one of the metrics called accuracy. Accuracy is the fraction of prediction. It determines the number of correct predictions over the total number of predictions made by the model. The formula of accuracy is: Accuracy = (TP+TN)/(TP+TN +FP+FN).

Sensitivity/TP Rate: Classifier's performance to spot positive results is related by Sensitivity. It is a measure of the number of patients who are classified as having complications among those who actually have the complications. Specificity is calculated as follows Precision = TP/(TP+FN).

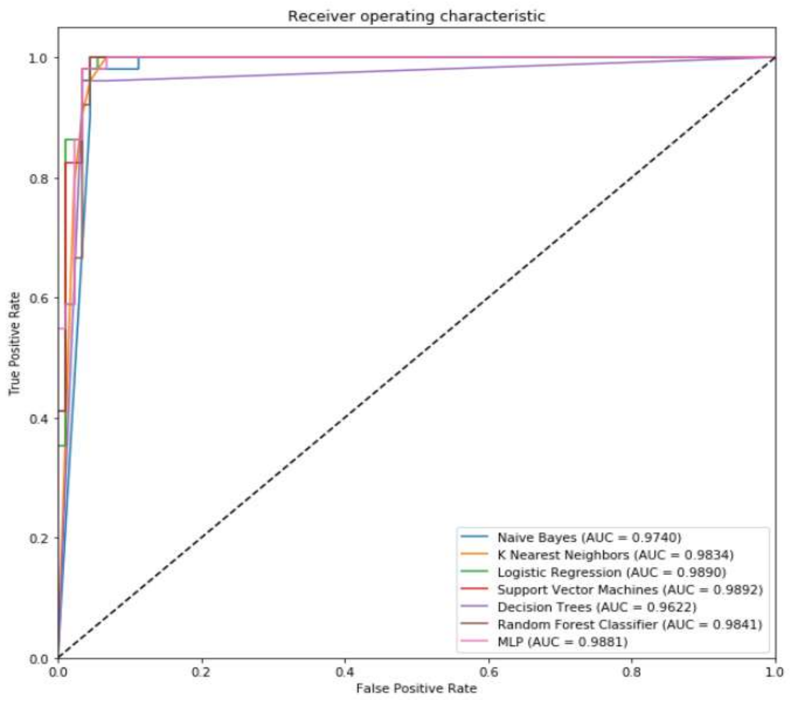
Specificity/TN Rate: Classifier's performance to spot negative results is related by Specificity. It is a measure of the number of patients who are classified as not having complications among those who actually did not have the complications. Specificity is calculated as follows: TN/(TN+FP).

4.1.2 Comparing Classifiers based on ROC Curves

ROC stands for Receiver Operating Characteristics. The ROC Graph plots TP rate (sensitivity) on the y axis and FP rate (1-specificity) on the x axis. An ROC curve hence shows relative tradeoffs between advantages and costs. ROC analysis provides tools to select possibly optimal models and to discard suboptimal ones independently from (and prior to specifying) the cost context or the class distribution. ROC analysis is related in a direct and natural way to cost/benefit analysis of diagnostic decision making. Classifiers in the upper left corner of ROC graph are preferred as both their sensitivity and specificity are high (classifier having the highest sensitivity and specificity is the best model for that dataset and hence is used for prediction). ROC graph for the above classifiers is as follows:



When a classifier algorithm is applied to test set, it yields a single confusion matrix, which in turn corresponds to one ROC point. We can create an ROC curve by thresholding the classifier with respect to its complexity. Each level of complexity in the space of a hypothesis class produces a different point in ROC space. Comparison of two different learning schemes on the same domain may be done by analyzing ROC curves in the same ROC space for the learning schemes. ROC curves for different classifiers are as follows:



Another metric that can be considered to evaluate the performances of various algorithms is the Area Under the Curve (ROC). But area under the curve is used when the given dataset is imbalanced. For imbalanced data, area under the ROC is considered as an evaluation parameter and not the testing accuracy. The above graph tells the AUC for all the classifiers used in this project. Support Vector Machines have the maximum AUC followed by Logistic Regression and MLP. Since the dataset has a 65-35 class distribution and as we assumed the data to be balanced, we use testing accuracy as the performance metric and not the area under the curve.

In Breast Cancer Prediction, the false negative rate should be given a higher priority over the false positive rate. This is because, a person not having cancer, but getting a positive test result might undergo various other tests to confirm the results, but a person having cancer getting a negative test result is at a greater risk of not taking any other test and hence his/her cancer would be left undetected. Hence, the trained machine should have an extremely low or zero false negative rate. Although both sensitivity and specificity of the trained machine should be high, in cancer prediction, the false negative rate can be given an edge above the false positive rate.

Interpreting the ROC Curve – It can be clearly seen from the ROC curve that all the trained classifiers lie in the upper left corner and hence it can be said that all the trained classifier give good results for the given dataset. But because all the classifiers lie in the upper left corner, their relative positions are to be taken into account. Random forest classifier has the topmost position in the ROC curve, indicating the highest true positive rate whereas support vector machine, MLP and Decision trees are the leftmost classifiers on the ROC curve, indicating the minimum false positive rate. Considering both true positive and false negative rate, there is no classifier that has the maximum of both w.r.t. other classifiers. Hence, there is a tradeoff between TP and FP rate while choosing the best model. In breast cancer prediction false negative rate should be as low as possible. This means that the true positive rate should be as high as possible. Random Forest classifier clearly shows the highest Sensitivity (= 1) and hence the least FN rate (= 0). But the false positive rate of Random Forest classifier is a little higher than MLP, SVM and Decision Tree classifier. No concrete conclusion can be made looking at the ROC curve. Hence we need to consider the numerical values of sensitivity and specificity to determine the best classifier for the given dataset.

4.2 Model Performances

A total of set of seven classification algorithms are used - Naïve Bayes (NB), K-Nearest Neighbors Classifier (KNN) and Logistic Regression (LR), Support Vector Machines (SVM), Decision Trees (DT), Random Forests and Multi-Layer Perceptron (MLP) have been applied on the dataset. For each experiment, the performance of the algorithms are measured using Accuracy, Sensitivity and Specificity. The table below demonstrates the results of different metrics for the algorithms to predict Breast Cancer:

Algorithm	Accuracy	Misclassification Rate	Sensitivity	Specificity	FP rate	FN Rate
Naïve Bayes	0.943	0.057	0.988	0.877	0.123	0.012
K Nearest Neighbors	0.957	0.043	0.977	0.924	0.075	0.022
Logistic Regression	0.964	0.036	0.988	0.926	0.074	0.012
SVM	0.971	0.029	0.989	0.943	0.057	0.011
Decision Tree	0.964	0.036	0.977	0.942	0.058	0.023
Random Forest	0.971	0.029	1.0	0.927	0.073	0
ANN (MLP)	0.971	0.029	0.989	0.943	0.057	0.011

Chapter 5

Conclusion

This paper treats the Wisconsin Madison Breast Cancer diagnosis problem as a pattern classification problem. In this report, several machine learning models were investigated and the optimal model was selected by considering the one that had the maximum sensitivity and specificity.

Both SVM and MLP gave the same accuracy, sensitivity and specificity of 97.1%, 98.9% and 94.3% respectively.

On the other hand, Random Forest Classifier gave no false negative results and hence a sensitivity of 100% but a specificity of 92.7% which was a little lower than that of MLP and SVM.

Considering the average of sensitivity and specificity, MLP and SVM gave better results by achieving an average of 96.60% than Random Forest Classifier which gave an average of 96.35%.

Hence, it can be concluded that both SVM and MLP gave the optimal (best) results when both sensitivity and specificity are taken into account. But it cannot be ignored that Random Forests gave a sensitivity of 100% which is remarkable and also an average of 96.35% which was a bit lower than that of MLP and SVM.

Bibliography

[1] Breast cancer facts and figures 2003-2004 (2003). American Cancer Society.

[2] Fine Needle Aspiration Biopsy of the Breast. American Cancer Society.

[3] William H Wolberg, W Nick Street, and Olvi L Mangasarian. Breast cancer Wisconsin (diagnostic) data set.

[4] Delen, D.; Walker, G.; Kadam, A. Predicting breast cancer survivability: A comparison of three data mining methods. Artif. Intell. Med. 2005, 34, 113–127.

[5]. Funahashi, K.; Nakamura, Y. Approximation of dynamical systems by continuous time recurrent neural networks. Neural Netw. 1993, 6, 801–806.

[6]. Razi, M.A.; Athappilly, K. A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models. Expert Syst. Appl. 2005, 29, 65–74.

[7]. Subasi, A.; Ercelebi, E. Classification of EEG signals using neural network and logistic regression. Comput. Methods Prog. Biomed. 2005, 78, 87–99.

[8]. Breiman,L.;Friedman,J.H.;Olshen,R.A.;Stone,C.J. Classification and Regression Trees; CRC Press: Boca Raton, FL, USA, 1984.

[9]. Howell, A.; Cuzick, J.; Baum, M.; Buzdar, A.; Dowsett, M.; Forbes, J.F.; Hoctin-Boes, G.; Houghton, J.; Locker, G.Y.; Tobias, J.S.; et al. Results of the ATAC (Arimidex, Tamoxifen, Alone or in Combination) trial after completion of 5 years' adjuvant treatment for breast cancer. Lancet 2004, 365, 60–62.

[10]. Pincus, S.M.; Gladstone, I.M.; Ehrenkranz, R.A. A regularity statistic for medical data analysis. J. Clin. Monit. Comput. 1991, 7, 335–345.

[11]. Wasson, J.H.; Sox, H.C.; Neff, R.K.; Goldman, L. Clinical prediction rules: Application and methodological standards. N. Engl. J. Med. 1985, 313, 793–799.

[12]. West, M.; Blanchette, C.; Dressman, H.; Huang, E.; Ishida, S.; Spang, R.; Zuzan, H.; Olson, J.A.; Marks, J.R.; Nevins, J.R.; et al. Predicting the clinical status of human breast cancer by using gene expression profiles. Proc. Natl. Acad. Sci. USA 2001, 98, 11462–11467.

[13]. Paliwal,M.;Kumar,U.A. Neural networks and statistical techniques: A review of applications. Expert Syst. Appl. 2009, 36, 2–17.

[14]. Furundzic, D.; Djordjevic, M.; Bekic, A.J. Neural networks approach to early breast cancer detection. J. Syst. Archit. 1998, 44, 617–633.

[15]. Behrman, M.; Linder, R.; Assadi, A.H.; Stacey, B.R.; Backonja, M.-M. Classification of patients with pain based on neuropathic pain symptoms: Comparison of an artificial neural network against an established scoring system. Eur. J. Pain 2007, 11, 370–376.

[16]. Ture, M.; Kurt, I.; Kurum, A.T.; Ozdamar, K. Comparing classification techniques for predicting essential hypertension. Expert Syst. Appl. 2005, 29, 583–588.

[17]. Pérez-Ortiz, M.; Gutiérrez, P.A.; Hervás-Martínez, C.; Yao, X. Graph-Based Approaches for over-Sampling in the context of ordinal regression. IEEE Trans. Knowl. Data Eng. 2015, 27, 1233–1245.

[18]. Cruz, J.A.; Wishart, D.S. Applications of machine learning in cancer prediction and prognosis. Cancer Inform. 2006, 2, 59–77.

[19]. Hinton, G.E. How neural networks learn from experience. Sci. Am. 1992, 267, 144–151.

[20]. Fu, Q.; Luo, Y.; Liu, J.; Bi, J.; Qiu, S.; Cao, Y.; Ding, X. Improving learning algorithm performance for spiking neural networks. In Proceedings of the 17th IEEE International Conference on Communication Technology, Chengdu, China, 27–30 October 2017.

[21]. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. Neuro computing 2017, 234, 11–26.

[22]. Wang, L.; Wang, Z.; Wei, G.; Alsaadi, F.E. Finite-time state estimation for recurrent delayed neural networks with component-based event-triggering protocol. IEEE Trans. Neural Netw. Learn. Syst. 2018, 29, 1046–1057.

[23]. Yang, F.; Dong, H.; Wang, Z.; Ren, W.; Alsaadi, F.E. A new approach to non-fragile state estimation for continuous neural networks with time-delays. Neuro computing 2016, 197, 205–211.

[24]. Floyd, C.E.; Lo, J.Y.; Yun, A.J.; Sullivan, D.C.; Kornguth, P.J. Prediction of breast cancer malignancy using an artificial neural network. Cancer 1994, 74, 2944–2948.

[25]. Fogel, D.B.; Wasson, E.C.; Boughton, E.M. Evolving neural networks for detecting breast cancer. Cancer Lett. 1995, 96, 49–53.

[26].Fogel,D.B.;Wasson,E.C.;Boughton,E.M.;Porto,V.W.;Angeline,P.J. Linear and neural models for classifying breast masses. IEEE Trans. Med. Imaging 1998, 17, 485–488.

[27].Pendharkar,P.C.;Rodger,J.A.;Yaverbaum,G.J.;Herman,N.;Benner,M.Association,statistical,mathematical and neural approaches for mining breast cancer patterns. Expert Syst. Appl. 1999, 17, 223–232.

[28]. Setiono, R. Extracting rules from pruned neural networks for breast cancer diagnosis. Artif. Intell. Med. 1996, 8, 37–51.

[29]. Wilding, P.; Morgan, M.A.; Grygotis, A.E.; Shoffner, M.A.; Rosato, E.F. Application of backpropagation neural networks to diagnosis of breast and ovarian cancer. Cancer Lett. 1994, 77, 145–153.

[30].Wu,Y.;Giger,M.L.;Doi,K.;Vyborny,C.J.;Schmidt,R.A.;Metz,C.E.Artificial neural networks in mammography: Application to decision making in the diagnosis of breast cancer. Radiology 1993, 187, 81-87.

[31]. Tabár, L.; Dean, P.B. Teaching Atlas of Mammography; Thieme: Stuttgart, Germany, 1985.

[32]. Vapnik,V.N.An overview of statistical learning theory. IEEE Trans. Neural Netw. 1999, 10, 988–999.

[33]. Bennett, K.P.; Blue, J.A. A support vector machine approach to decision trees. In Proceedings of the IEEE International Joint Conference on Neural Networks, Anchorage, AK, USA, 4–9 May 1998; Volume 3, pp. 2396–2401.

[34].Suykens,J.A.K.;Vandewalle,J. Least squares support vector machine classifiers. Neural Proccess.Lett.1999,9,293–300.

[35]. Polat, K.; Günes¸, S. Breast cancer diagnosis using least square support vector machine. Digit. Signal Process. 2007, 17, 694–701.

[36]. Akay, M.F. Support vector machines combined with feature selection for breast cancer diagnosis. Expert Syst. Appl. 2009, 36, 3240–3247.

[37]. Chen, H.-L.; Yang, B.; Liu, J.; Liu, D.-Y. A support vector machine classifier with rough set-based feature selection for breast cancer diagnosis. Expert Syst. Appl. 2011, 38, 9014–9022.

[38]. John, G. H.; Kohavi, R.; Pfleger, K. Irrelevant features and the subset selection problem. In Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ, USA, 10–13 July 1994; pp. 121–129.

[39]. Moreno-Seco, F.; Micó, L.; Oncina, J. A modification of the LAESA algorithm for approximated k-NN classification. Pattern Recognit. Lett. 2003, 24, 47–53.

[40]. Sarkar, M.; Leong, T.Y. Application of k-nearest neighbors algorithm on breast cancer diagnosis problem. In Proceedings of the AMIA Symposium, Los Angeles, CA, USA, 4–8 November 2000; pp. 759–763.

[41]. Bagui, S.C.; Bagui, S.; Pal, K.; Pal, N.R. Breast cancer detection using rank nearest neighbor classification rules. Pattern Recognit. 2003, 36, 25–34.

[42]. Medjahed, S.A.; Ait Saadi, T.; Benyettou, A. Breast cancer diagnosis by using k-nearest neighbor with different distances and classification rules. Int. J. Comput. Appl. 2013, 62, 1–5.

[43]. De Mántaras, R.L. A distance-based attribute selection measure for decision tree induction. Mach. Learn. 1991, 6, 81–92.

[44]. Mingers, J. An empirical comparison of selection measures for decision-tree induction. Mach.Learn.1989,3,319–342.

[45]. Quinlan, J.R. Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. 1996, 4, 77–90.

[46]. Watkins, A.B.; Boggess, L.C. A resource limited artificial immune classifier. In Proceedings of the 2002 Congress on Evolutionary Computation, Honolulu, HI, USA, 12–17 May 2002; pp. 926–931.

[47]. Polat, K.; Sahan, S.; Kodaz, H.; Gnes, S. A new classification method for breast cancer diagnosis: Feature selection artificial immune recognition system (FS-AIRS). In Proceedings of the International Conference on Natural Computation, Changsha, China, 27–29 August 2005; Volume 3611, pp. 830–838.

[48] Witten, I.H.; Frank, E.; Trigg, L.E.; Hall, M.A.; Holmes, G.; Cunningham, S.J. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Available online: https://researchcommons. waikato.ac.nz/handle/10289/1040 (accessed on 10 January 2018).

[49].Sumbaly,R.;Vishnusri,N.;Jeyalatha,S.Diagnosis of breast cancer using decision tree data mining technique. Int. J. Comput. Appl. 2014, 98, 16–24.

[50].Devi,R.D.H.;Devi,M.I. Outlier detection algorithm combined with decision tree classifier for early diagnosis of breast cancer. Int. J. Adv. Eng. Technol. 2016, 12, 93–98.

[51]. Seera,M.;Lim,C.P. A hybrid intelligent system for medical data classification.ExpertSyst. Appl.2014, 41, 2239–2249.

[52]. Frayman, Y.; Wang, L. Data mining using dynamically constructed recurrent fuzzy neural networks. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Melbourne, Australia, 15–17 April 1998; pp. 122–131.

[53]. Wang,L.;Chu,F.;Xie,W.Accuratecancerclassificationusingexpressionsofveryfewgenes. IEEE/ACM Trans. Comput. Biol. Bioinform. 2007, 4, 40–53.

[54]. Zhang,J.;Zulkernine,M.;Haque,A. Random-forests-based network intrusion detection systems. IEEE Trans. Syst. Man Cybern. 2008, 38, 649–659.

[55] Adi Bronshtein. Train/Test Split and Cross Validation in Python. Understanding Machine Learning (2017).

[56] Mohammad Bolandraftar and Sadegh Bafandeh Imandoust - "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background"- International Journal of Engineering Research and Applications Vol. 3, Issue 5, Sep-Oct 2013.

[57] Chao-Ying, Joanne, Peng Kuk Lida Lee, Gary M. Ingersoll –"An Introduction to Logistic Regression Analysis and Reporting ", September/October 2002 [Vol. 96(No. 1)]

[58] Logistic Regression for Machine Learning - Machine Learning Mastery https://machinelearningmastery.com/logistic-regression-formachine-learning/

[59] Puneet Yadav, Rajat Varshney, Vishan Kumar Gupta. Diagnosis of Breast Cancer using Decision Tree Models and SVM (2016)

[60] Cristianini N, Shawe-taylor J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, (2000) London: Cambridge University Press.

[61] Joachims T. Making large-scale support vector machine learning practical. Advances in Kernel Methods: Support Vector Learning. (1998) MIT Press, Cambridge, MA, 169- 184.

[62] 9 Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". The American Statistician. 46 (3): 175–185.

[63]. Quinlan, J.R. Induction of decision trees. Mach. Learn. 1986, 1, 81–106.

[64]. Apté,C.;Weiss,S. Data mining with decision trees and decision rules .Future Gener. Comput.Syst.1997, 13, 197–210.

[65]. Quinlan, J.R. C4.5: Programs for Machine Learning; Elsevier: New York, NY, USA, 1993.

[66]. Applied Machine Learning by Madan Gopal - Book