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

Methods and Analysis

2.1 Data Preprocessing and Exploratory Data Analysis

2.1.1 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.

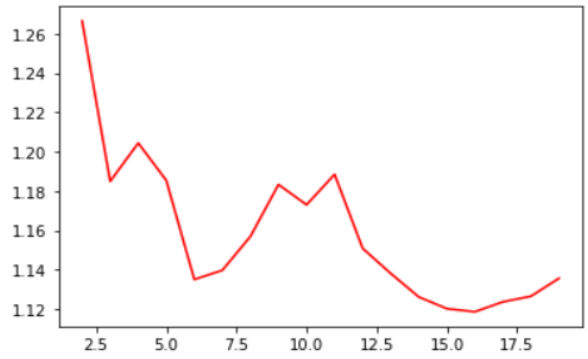
Since the dataset consists of only 699 samples (and not millions), dropping samples with missing feature values will not be a good option as these samples might contain information that can affect the prediction of the machine. Predicting these missing values using an algorithm seems to be a better alternative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

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’. 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

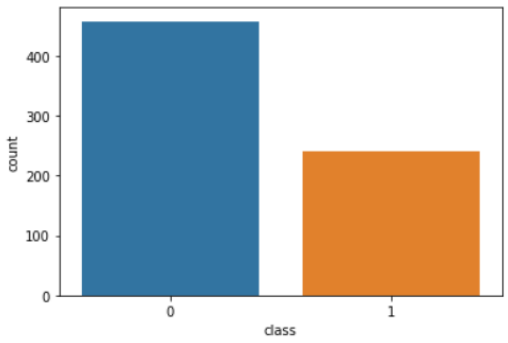
These two unique classes ('2' and '4') are then encoded to 0 and 1. Now, class=0 represents a benign tumor and class=1 represents a malignant tumor.

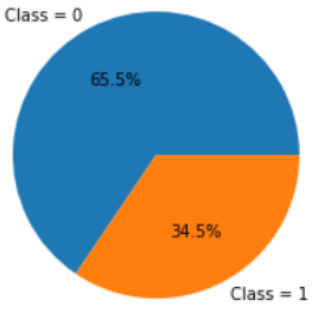
	clump_thickness	size_uniformity	shape_uniformity	marginal_adhesion	epithelial_size	bland_chromatin	normal_nucleoli	mitoses	bare_nucleoli	class
0	8	4	5	1	2	7	3	1	7	1
1	6	6	6	9	6	7	8	1	8	0
2	1	1	1	1	1	2	1	1	1	0
3	1	1	3	1	2	2	1	1	1	0
4	1	1	2	1	3	1	1	1	1	0

2.1.2 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.

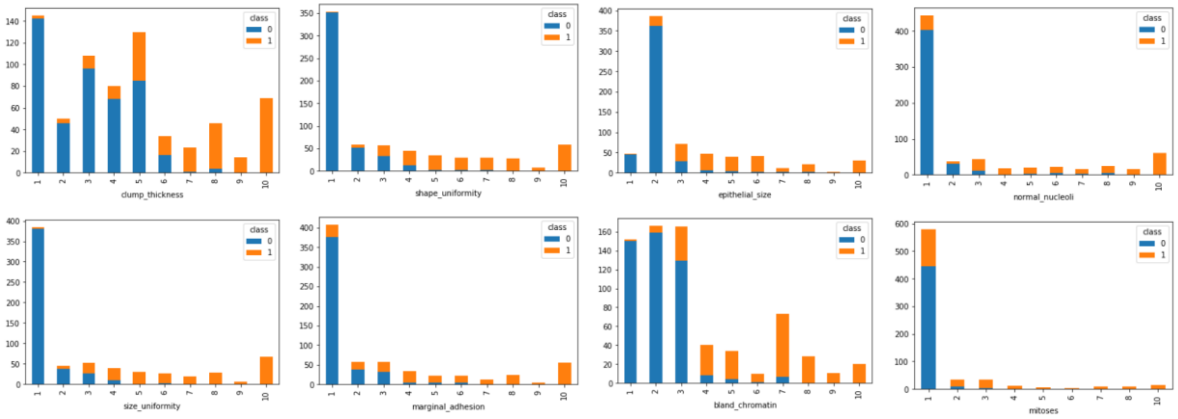
To start with, I plotted both count plot and pie chart for the distribution of class in the given dataset.





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

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

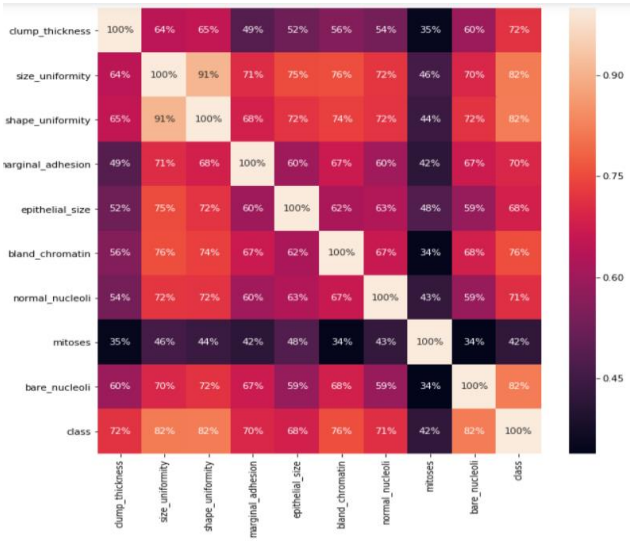


These cross tab plots indicates the number of tumors classified as Benign or Malignant based on the value (1-10) of a certain feature.

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.

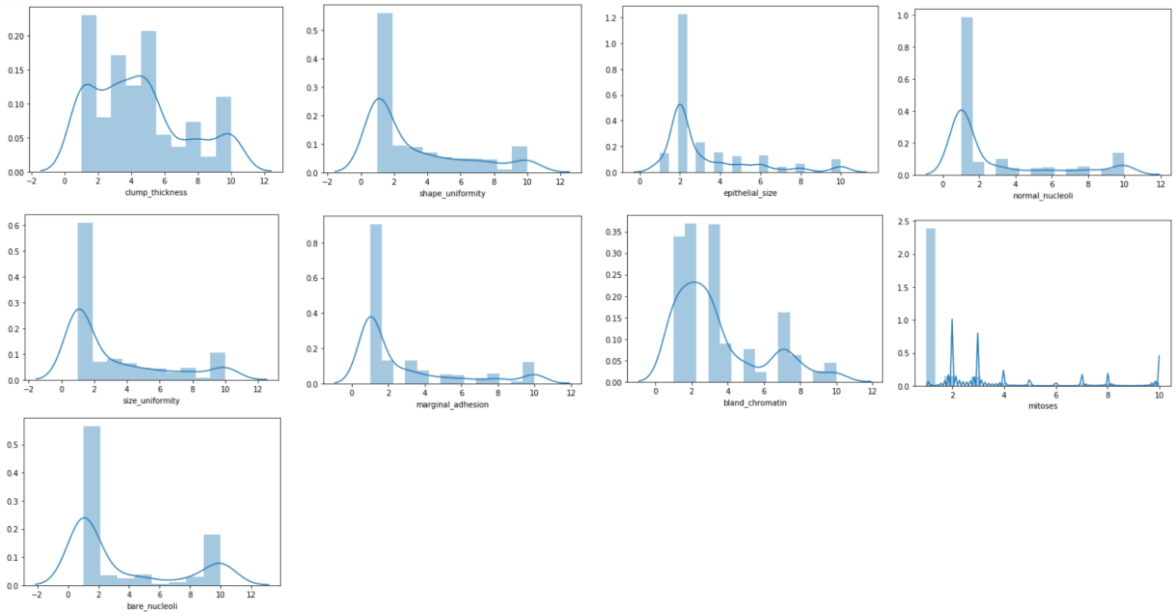
Several features are more sensitive than others in terms of classifying a tumor. For example – If the value of shape uniformity/size uniformity is observed to be greater than 5, the cell is mostly classified as Malignant. In case of mitoses, even if the value reaches 3, then the tumor is more prone to 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.

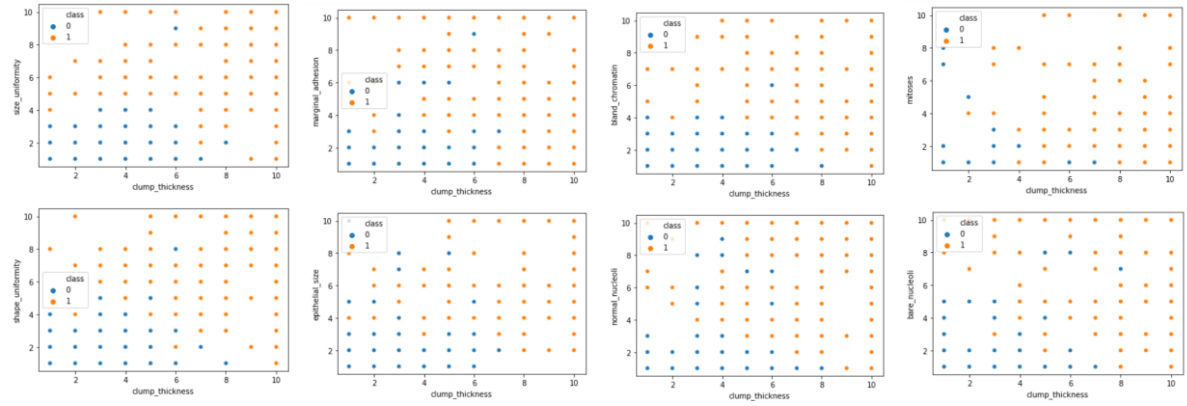


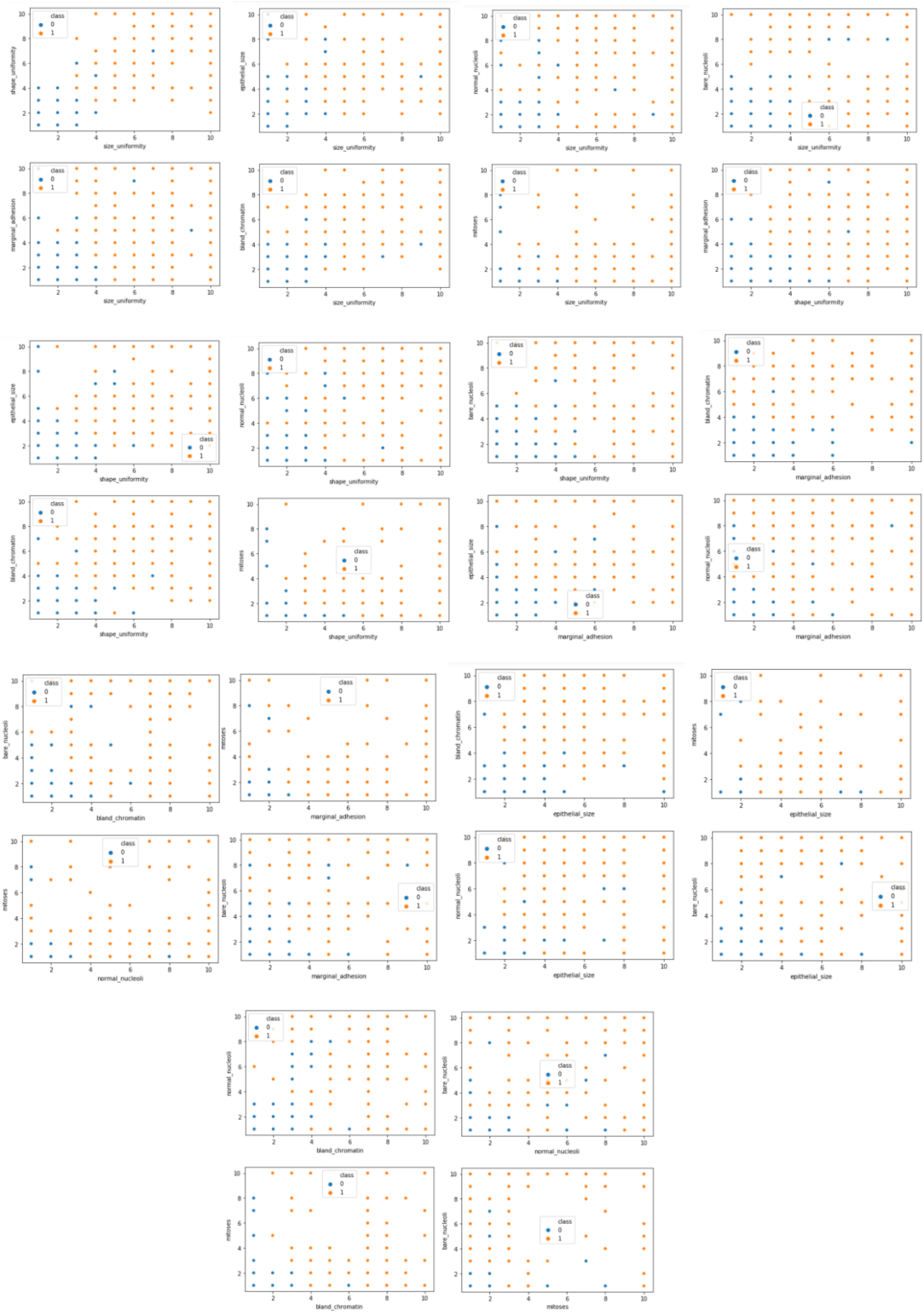
The above heat map shows that the features size uniformity, shape uniformity and bare nucleoli are highly correlated to class while mitoses is the feature that is least correlated to class.

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.

2.2 Modelling

2.2.1 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 [4]. 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).

2.2.2 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.

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

```
# Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
nb_classifier = GaussianNB()
nb_classifier.fit(X_Train, Y_Train)
Y_Pred_nb = nb_classifier.predict(X_Test)
print('Naive Bayes Accuracy on Training Data :',nb_classifier.score(X_Train,Y_Train))
print()
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 : 0.9660107334525939

[[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
```

2.2.3 K-Nearest Neighbors Model

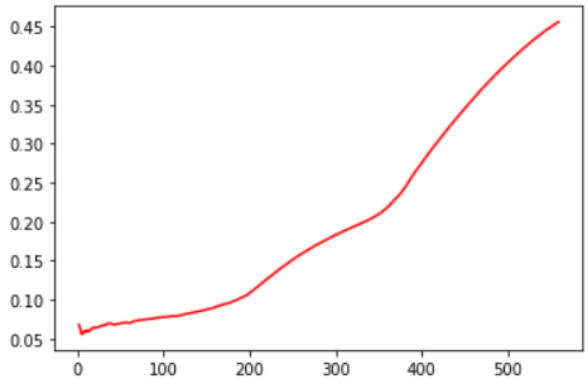
The K-nearest neighbor's algorithm is one of the simplest machine learning algorithms. It has merely supported the concept that objects that are 'near' every alternative can additionally have similar characteristics. So if it can recognize the characteristic options of one of the objects, it will be 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 [5]. 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')
knn_classifier.fit(X_Train, Y_Train)
Y_Pred_knn = knn_classifier.predict(X_Test)
print('K Nearest Neighbors Accuracy on Training Data :',knn_classifier.score(X_Train,Y_Train))
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 : 0.9821109123434705

[[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
```

2.2.4 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 [6]. 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 [7]. 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 Algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_Train_Scaled = sc.fit_transform(X_Train)
X_Test_Scaled = sc.fit_transform(X_Test)
lr_classifier = LogisticRegression()
lr_classifier.fit(X_Train_Scaled, Y_Train)
Y_Pred_lr = lr_classifier.predict(X_Test_Scaled)
print('Logistic Regression Accuracy on Training Data :',lr_classifier.score(X_Train_Scaled,Y_Train))
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))
```

```
Logistic Regression Accuracy on Training Data : 0.9713774597495528

[[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
```

Chapter 3

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.

3.1 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.

3.1.1 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.

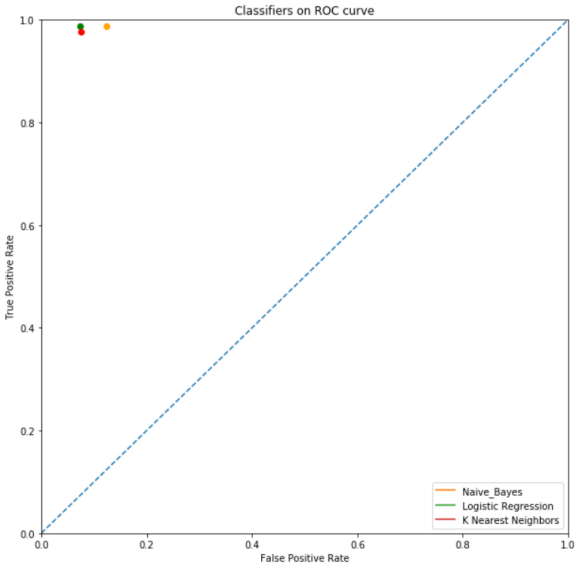
Accuracy/Success Rate: Evaluation of classification models is done by one of the metrics called accuracy. Accuracy is the fraction of prediction. It determines the number of correct predictions over the total number of predictions made by the model. The formula of accuracy is: Accuracy = (TP+TN)/(TP+TN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

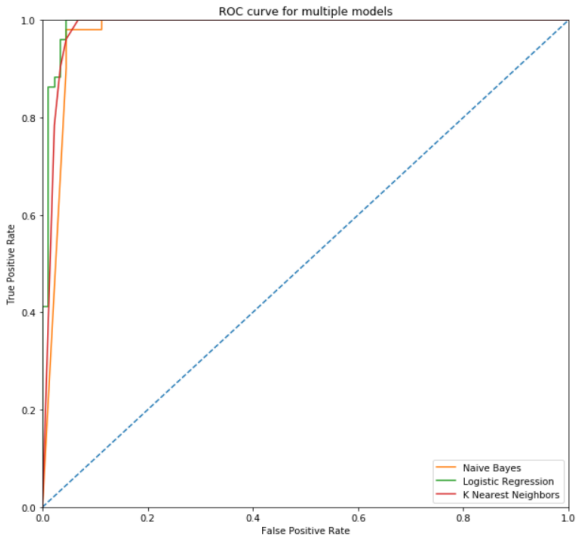
Specificity/TN Rate: Classifier's performance to spot negative results is related by Specificity. It is a measure of the number of patients who are classified as not having complications among those who actually did not have the complications. Specificity is calculated as follows: TN/(TN+FP).

3.1.2 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:



3.2 Model Performances

A total of set of three classification algorithms are used - Naïve Bayes (NB), K-Nearest Neighbors Classifier (KNN) and Logistic Regression (LR) 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

Chapter 4

Conclusion

This paper treats the Wisconsin Madison Breast Cancer diagnosis problem as a pattern classification problem. In this report we investigated several machine learning models and we selected the optimal model by selecting a high accuracy level combined with a low rate of false-negatives (the means that the metric is high sensitivity).

The Logistic Regression model had the optimal results for Accuracy (0.964), Sensitivity (0.988) and Specificity (0.926).

****

For the mid semester report, I've only used 3 learning algorithms namely Naïve Bayes, K Nearest Neighbors and Logistic Regression. As the semester progresses, I intend to use the rest of the algorithms on the dataset. The final report will be in continuation of the same and shall cover the implementation of other classification algorithms on the same dataset. The results and conclusion will change accordingly.

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

[5] Mohammad Bolandraftar and SadeghBafandehImandoust - "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.

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

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