Wisconsin Breast Cancer Diagnostic Classification using KNN and Random Forest

Josh Lee, Steve Ray, Stanley Ng

Abstract

Breast cancer is a disease that in which early diagnosis is extremely important as it can facilitate the subsequent clinical management of patients. In the past a lot of people are misdiagnosed or diagnosed for breast cancer at a later stage due to insufficient tools to identify the patient's situation. We have taken a dataset from Dr. William H. Wolberg who is from the University of Wisconsin Hospitals, Madison, and with this dataset, have performed Machine learning classification method on. Therefore, we are applying machine learning (ML) classification methods to classify breast cancer patients into benign or malignant risk groups for this datasets by using K-Nearest Neighbor and Random Forest classifier methods. The predictive models we are using here are based on supervised machine learning techniques with different input attributes. In this report we look at the statistical distribution of the data along with the different performances of the two classifier algorithms and which of them performs better.

Contents

1. Introduction

In this data set, we are looking at 10 different variables and their relation to whether a cancer cell is considered malignant or benign. In this paper, we will use several methods of data analysis to predict how accurate our dataset is for determining which cancer cells are malignant or benign. In the second section, we will go over the data description, where the variables are explained in greater detail. In the third section, we are looking at the data distribution and Statistical Analysis which will point out some of the statistical relationships within the data. The fourth section mentions K-Nearest Neighbor Classification, which is a method that splits the data and uses Euclidean Distance to assess how accurate of a predictor the data is. The fifth section is Random Forest Classification which also splits the data and uses decision trees to assess the predictive strength of the data. The sixth section is the Conclusion, where we will sum up our findings and provide suggestions for future projects on this topic.

2. Data Description

The data for this project was taken from Dr. William H. Wolberg who is from the University of Wisconsin Hospitals in Madison. The data consists 700 rows of just females each with 11 attributes, 10 of which specify the different characteristics of a tumor that was found within the breast of each individual. This table shows what each variable is and the values they can contain.

#	Attribute	Domain
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
11.	Class:	(2 for benign, 4 for malignant)

We used this data to train and classify already existing observations to produce an accuracy using two supervised machine learning algorithms, Random Forest Classifier and K-Nearest Neighbor Classifier. We also look at each attribute and their correlation to other existing attributes within the data set.

3. Data Distribution and Statistical Analysis using SAS

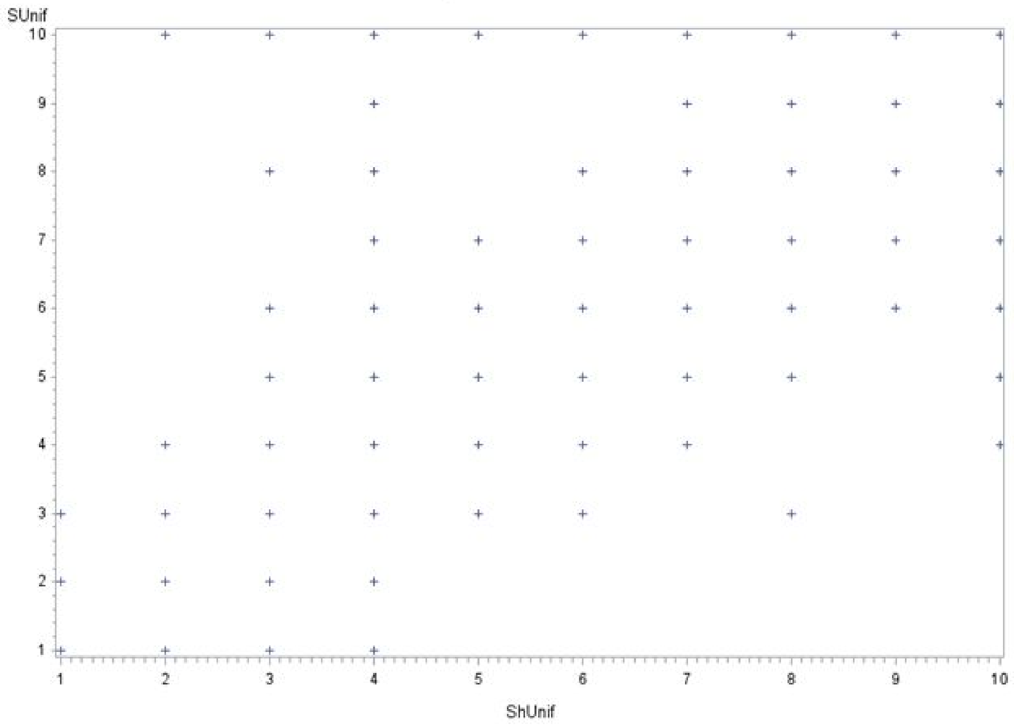
The 10 variables we are looking at in this data set all vary between 1 and 10. The variables we found the strongest correlation between is SUnif and ShUnif with a correlation of .90688. Some other strong correlations in the data set are those between SUnif and BChrom, SUnif and EpSize, SUnif and BChrom, and SUnif and NNuc, all of which have correlation coefficients between .7 and .8 . The chart below shows the more important portion of the correlation coefficient chart.

Pearson Correlation Coefficients
Prob > |r| under H0: Rho=0
Number of Observations

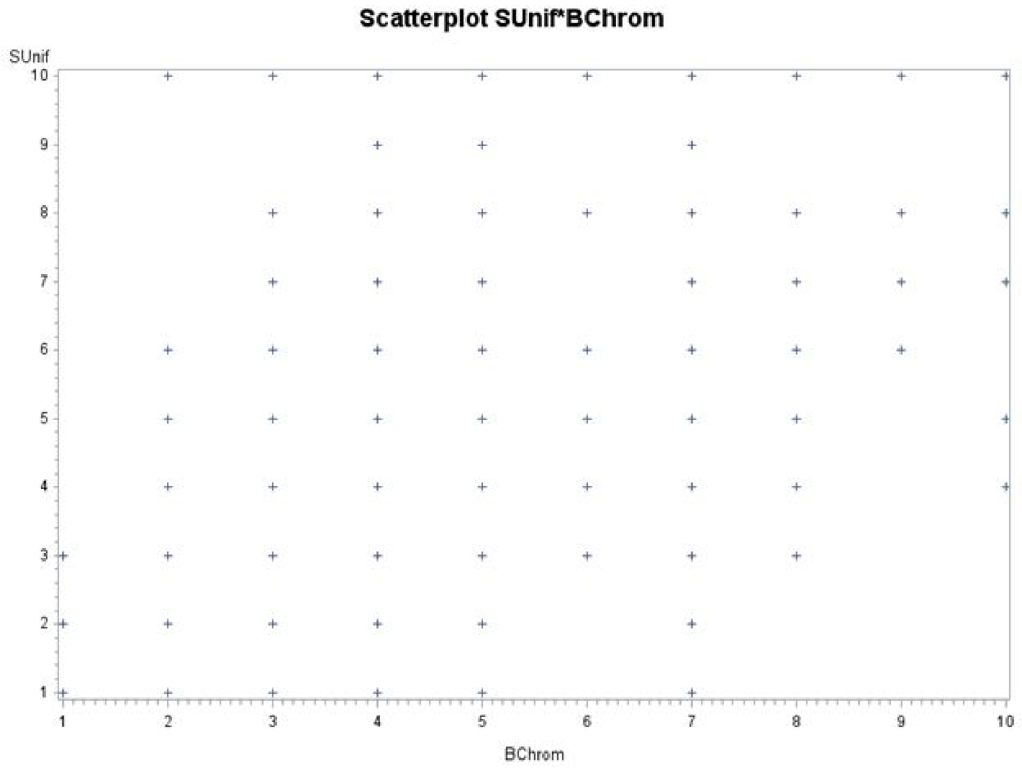
	CThick	SUnif	ShUnif	MAdhes	EpSize	BNuc	BChrom	NNuc	Mit
CThick	1.00000	0.64491	0.65459	0.48636	0.52182	0.59309	0.55843	0.53583	0.35003
		<.0001	<.0001	<.0001	<.0001	<.0001	<.0001	<.0001	<.0001
	699	699	699	699	699	683	699	699	699
SUnif	0.64491	1.00000	0.90688	0.70558	0.75180	0.69171	0.75572	0.72286	0.45869
	<.0001		<.0001	<.0001	<.0001	<.0001	<.0001	<.0001	<.0001
	699	699	699	699	699	683	699	699	699
ShUnif	0.65459	0.90688	1.00000	0.68308	0.71967	0.71388	0.73595	0.71945	0.43891
	<.0001	<.0001		<.0001	<.0001	<.0001	<.0001	<.0001	<.0001
	699	699	699	699	699	683	699	699	699
MAdhes	0.48636	0.70558	0.68308	1.00000	0.59960	0.67065	0.66672	0.60335	0.41763
	<.0001	<.0001	<.0001		<.0001	<.0001	<.0001	<.0001	<.0001
	699	699	699	699	699	683	699	699	699

On the next several pages, there are several scatterplots which show the data for the variables which are most strongly correlated.

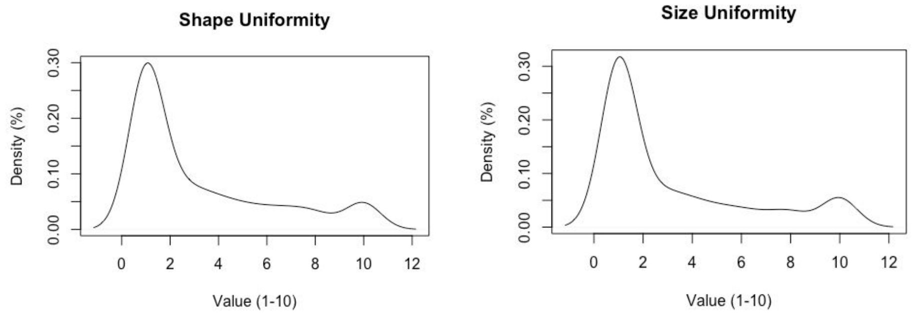
Scatterplot SUnif*ShUnif



Above is the scatterplot between SUnif and ShUnif, which contain the highest correlation between variables in the data set.



This scatterplot shows the correlation between SUnif and BChrom which has the second strongest correlation in the data set.



Here are the density functions for shape uniformity and size uniformity, which are extremely similar as would be expected from the strong correlation between them.

4. K-Nearest Neighbor Classification

4.1 The Basics of K-Nearest Neighbor Classification

K-Nearest Neighbor is a machine learning classification algorithm. It uses previous observations to train it's algorithm into classifying new observations by looking at already classified examples. How does it use these other observations to classify new ones? One way this can be done is through the Euclidean distance formula. The Euclidean distance is a modified version of the pythagorean theorem[2]. The pythagorean theorem states if you know the distance of a and b, then you can find the distance c using this formula[3].

$$a^2 + b^2 = c^2.$$

Knowing this formula we can apply it to the points in a 2 dimensional plane. If we know of points q and p, then using the coordinates for those points we can calculate the distance using the euclidean distance[2]

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}.$$

If q1 and p1 denote the x position for each observation in the coordinate plane, we can apply the formula above to find the distance between those two points[2]. Knowing this, we can take the formula a step further and look at observations with n number of variables.

$$\begin{aligned} \mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) &= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \end{aligned}$$

This formula calculates the distance of 2 points with n number of variables in an n-dimensional space. K-Nearest Neighbor uses the Euclidean Distance Formula to calculate the K number of nearest neighbors, that are already classified, for a new observation. The algorithm then takes the majority vote of the number of neighbors to classify the observation. Regression lines can show where the classification splits when the number of neighbors changes to a certain amount. To visually represent this we can look at an example. This is a 2-dimensional dataset, meaning it consists of 2 variables, used to calculate the Euclidean Distance.

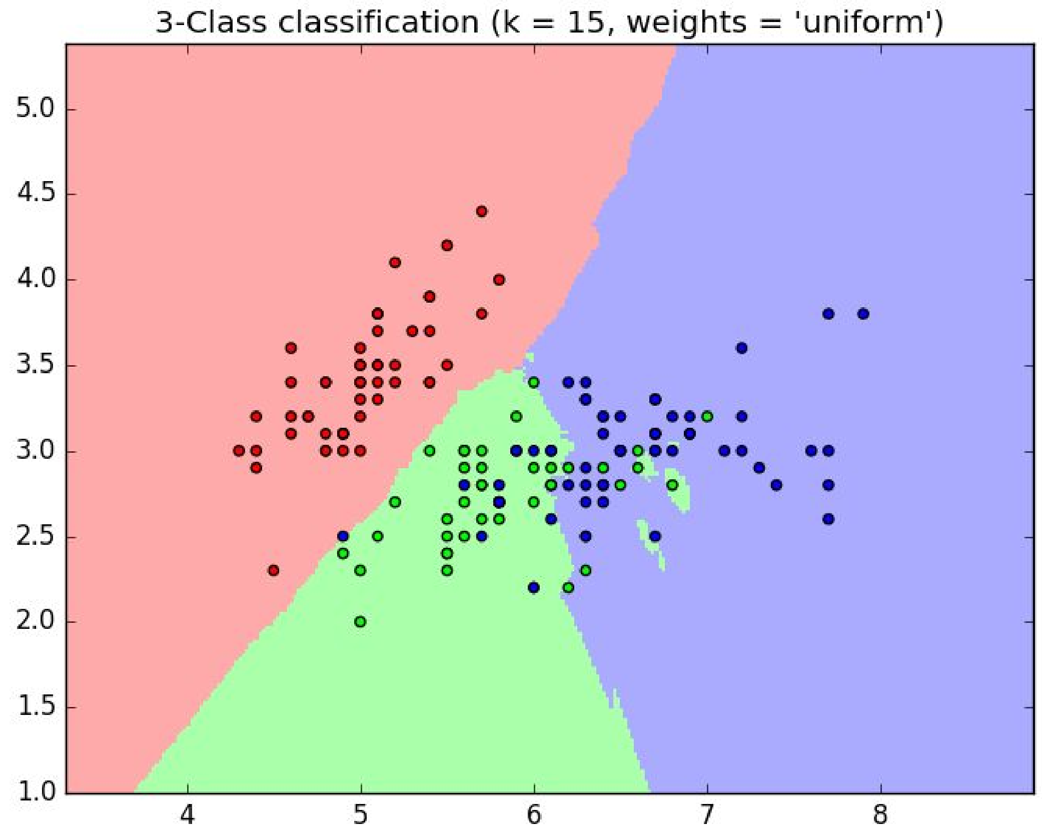
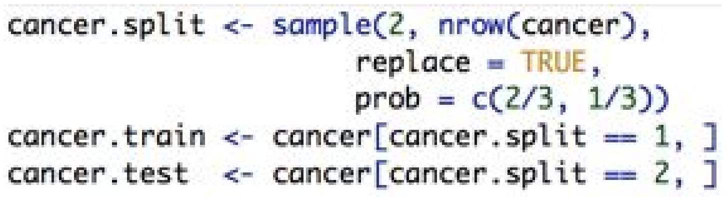


Figure: 4.1 K-NN Classification Regions [4]

In this example the number of neighbors used to classify a new observation is 15. We can see where the regression lines are drawn to split the data into 3 different classifications.

4.2 Applying the K-Nearest Neighbor Algorithm

Our Data set consists of 10 different useable variables to classify a patient's tumor as benign (2) or malignant (4). We use the K-Nearest Neighbor library provided by R to calculate the distances between points and to do the majority vote classification of any new observation. To test the accuracy of our algorithm, we split up our data into 2 sets,



```
cancer.split <- sample(2, nrow(cancer),
                        replace = TRUE,
                        prob = c(2/3, 1/3))
cancer.train <- cancer[cancer.split == 1, ]
cancer.test  <- cancer[cancer.split == 2, ]
```

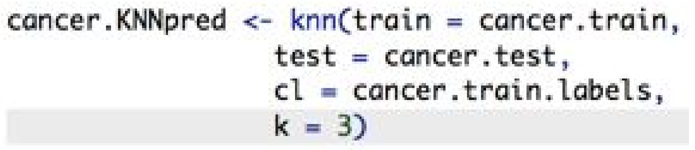
[1]

a training set (⅔), and a test set (⅓). We do this because we want to test how well our K-Nearest Neighbor algorithm works on other data points which we already know the classification for. After running K-Nearest Neighbor multiple times on the data we found that our accuracy was the same every iteration.

```
K:3, Accuracy: 0.9832636
K:3, Accuracy: 0.9832636
K:3, Accuracy: 0.9832636
K: 3, Accuracy: 0.9832636
```

The reason for this, is every time we split the data it's always going to split into the same training set and test set. We need to randomize the data split to get an accurate prediction overall. We do this by splitting the data within a loop.

The next step was to find which number of neighbors (K) fits best for our algorithm.



```
cancer.KNNpred <- knn(train = cancer.train,
                      test = cancer.test,
                      cl = cancer.train.labels,
                      k = 3)
```

[1]

We know that having too many K can misclassify outlier groupings, so we need to find a number that, overall, gives us the best accuracy. Since we know that the data is split up into two thirds and one third, with 700 observations total, then we know that we have around 450 observations that our test set can use to calculate the neighbors. Obviously 450 neighbors is too many so we cut that down to half since we are only doing two classification groups. After testing the number of neighbors K could be we found the lower results for K, the better the algorithm performed.

```
K: 1, Accuracy: 0.9581395
K: 2, Accuracy: 0.972093
K: 3, Accuracy: 0.972093
K: 4, Accuracy: 0.972093
K: 5, Accuracy: 0.9627907
K: 6, Accuracy: 0.9674419
K: 7, Accuracy: 0.9627907
```

after running rerunning the algorithm with random splits for the training set and test set 100 times, we found that only 4 neighbors are needed to optimally classify a new observation, giving us an accuracy prediction of around 97.7%.

```
K: 3, Overall Accuracy: 0.9774771
```

5. Random Forest Classification

5.1 Random Forest Background

Random Forests is an effective machine learning (ML) algorithm that uses an ensemble learning of decision trees. In this context, ensemble learning uses multiple learners or hypothesis for coming up with predictions, often times it outperforms classifier that only use one algorithm alone.

5.2 The Basics of Random Forest Algorithm

Suppose matrix S is a matrix of training sample as input to the Random Forest algorithm to create a classification model.

$$S = \begin{bmatrix} f_{A1} & f_{B1} & f_{C1} & C_1 \\ \vdots & & \vdots & \\ f_{AN} & f_{BN} & f_{CN} & C_N \end{bmatrix}$$

In this case, fA1,fA2...fAn, are the feature A of the 1st sample, followed by feature B to the Nth sample, and feature C to the Nth sample. Last column C1-Cn represents the training class (a.k.a Target column). The goal is to create a random forest to classify this sample set.

Create random subsets from the sample set:

$$S_1 = \begin{bmatrix} f_{A12} & f_{B12} & f_{C12} & C_{12} \\ f_{A15} & f_{B15} & f_{C15} & C_{15} \\ \vdots & & \vdots & \\ f_{A35} & f_{B35} & f_{C35} & C_{35} \end{bmatrix} S_2 = \begin{bmatrix} f_{A2} & f_{B2} & f_{C2} & C_2 \\ f_{A6} & f_{B6} & f_{C6} & C_6 \\ \vdots & & \vdots & \\ f_{A20} & f_{B20} & f_{C20} & C_{20} \end{bmatrix}$$

$$S_M = \begin{bmatrix} f_{A4} & f_{B4} & f_{C4} & C_4 \\ f_{A9} & f_{B9} & f_{C9} & C_9 \\ \vdots & & \vdots & \\ f_{A12} & f_{B12} & f_{C12} & C_{12} \end{bmatrix}$$

Next, we create a lot of subsets with random values from the sample set. For example, for S1 subset, we use the line row 12 to row 35. Then we create Decision Tree #1, for S2 subset, we use the line row 2 to 20, forming Decision Tree #2. From this M sample, we can create the Decision Tree #M.

Use all the decision trees to create a ranking of classifiers:

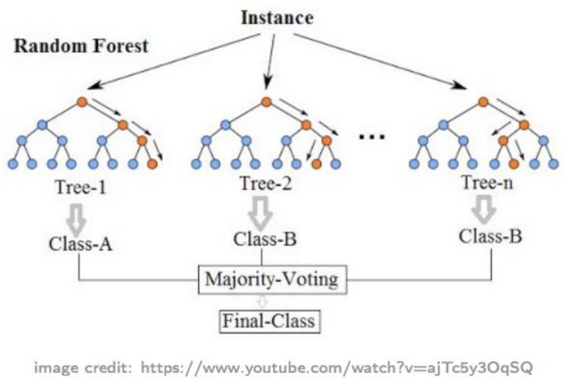


Figure 5.1 Random Forest Tree Illustration

The general thumb rules for random forest is to build a bunch of trees (a forest) and have them vote on the prediction. From the graph above, we import S1,S2..Sn Decision Trees to RF model to make the class prediction. Suppose Tree-1 is Class A, Tree-2 is Class B, Tree-N is Class B, the majority voting account number of votes for each classes and predicted the Final-Class as Class B.

Measure of Node Impurity to get the best split in Decision Tree:

In this project, we applied the default random forest GINI Index [5] as the measure of node impurity. Other measures that can be implemented are Entropy, Misclassification error, etc.

$$G(t) = 1 - \sum_{k=1}^{Q} p^2\left(k \middle| t\right)$$

Reference: 2013 SciRes.com

Given a node t and estimated class probabilities p(k|t) k = 1,...Q, where Q is the number of classes. To calculate the node impurity, at each node the decrease in the GINI index G(t) is calculated for variable tj used to make the split, where the variable xj is used to split a node.

5.3 Implementing Random Forest & Experimental Results

The Wisconsin Breast Cancer Diagnosis Datasets problem has two class outcomes: “Malignant” or “Benign”. In this section we present experimental results of the Random Forest Classification method through 20 times of trials. This is done in R program language with RF installed package [6]. The datasets were cleaned and divided randomly into training set and test set in the ration of (⅔,⅓).

Parameters Proposed to fit RF model:
Target Column: Class
Data: Breast Cancer Wisconsin Training Set
Number of Trees in RF:500

The RF method was then experimented 20 times. The graph figure 1. below shows the result of 20 trials of proposed method on training set and validation set. [File: Result.xls]

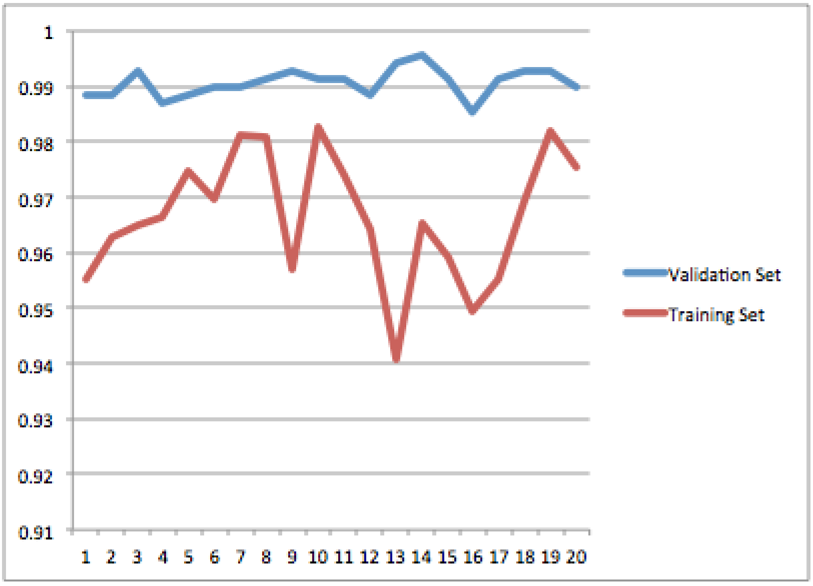


Figure 5.2. Result of 20 trials of RF method on training set and Validation set of Wisconsin Breast Cancer Dataset.

Next, we analyze how the numbers of trees in Random Forest would affect classification error. In Figure 2. We can clearly see there's a prominent decrease after increasing the number of trees until 120, after that the results shows a stagnant trend. Classification error included in this plot are: err.rate,oob.times error,confusion error.

The oob error is called the Out-of-Bag error, which is an internal error estimate of a random forest as it is being constructed, also known as Misclassification rate.[6]

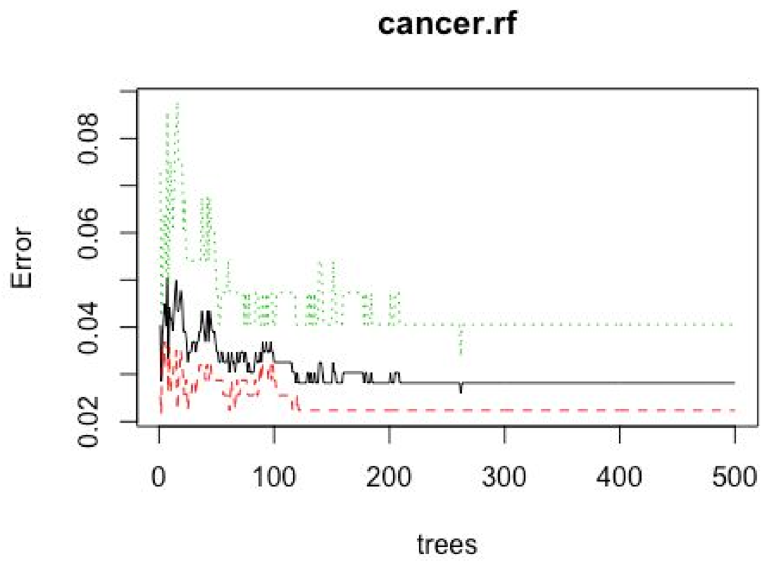
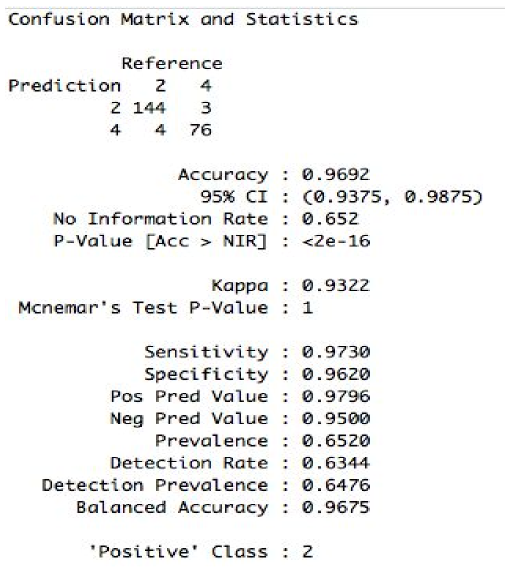


Figure 5.3. The increase in number of trees for Random Forest Classification[7] Results in a smaller classification error.

5.4 Evaluating Random Forest Classifier Result with R

In this section, I used library package (“caret”) and (“pROC”) to further analyze the dataset result study. Documentation for “Caret”[8] and “pROC”[9] can be found under reference.

Analysing with Caret:
With the results above, We can now run it to produce a confusion matrix with caret; Summary of confusion.matrix:



```
Confusion Matrix and Statistics

          Reference
Prediction   2   4
         2 144   3
         4   4  76

               Accuracy : 0.9692
                 95% CI : (0.9375, 0.9875)
    No Information Rate : 0.652
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9322
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9730
            Specificity : 0.9620
         Pos Pred Value : 0.9796
         Neg Pred Value : 0.9500
             Prevalence : 0.6520
         Detection Rate : 0.6344
   Detection Prevalence : 0.6476
      Balanced Accuracy : 0.9675

       'Positive' Class : 2
```

Figure 5.4 Confusion Matrix Summary [5]

In the confusion matrix and statistics, each column holds the reference (actual data) and within each row is the prediction. The diagonal result represent the correct predicted instances, vice versa.

Overall accuracy is calculated at somewhere over 96% with a very minimal p-value of 2 x 10^-16. We can conclude that the classifier is doing a reasonable job classifying the cancer class.

Next, we can look at the sensitivity and specificity to help us measure the performance of the classifier in correctly predicting the actual class of an item, it measures true positive and true negative performance respectively. From the data above, we can see that our classifier correctly predicted the cancer classes 97.30% of the time. Further, when we shouldn't have predicted classes we didn't we performed a 96.20% correctly. We can further visualize this in the ROC curve Figure 3.

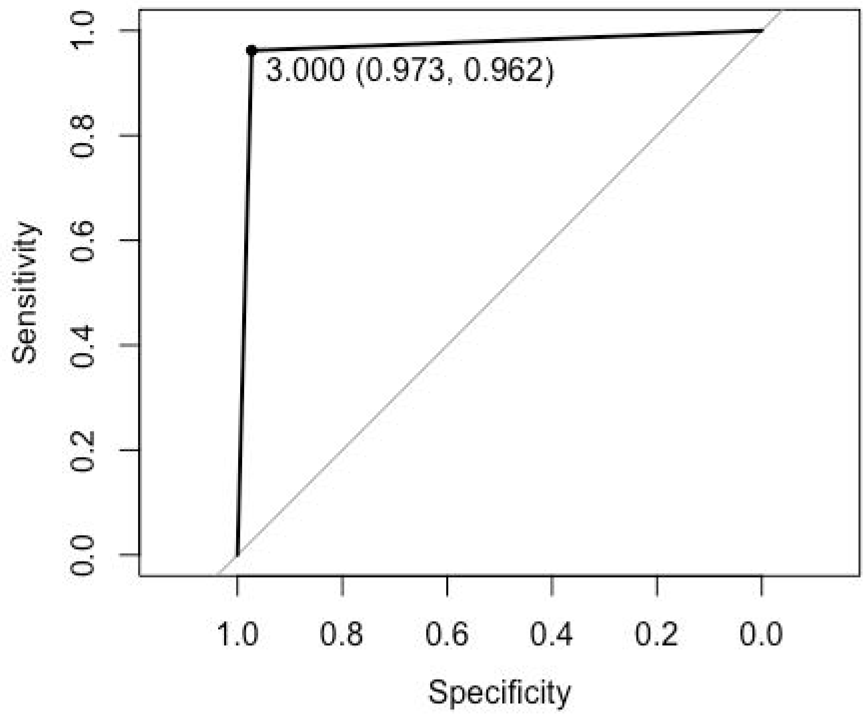


Figure 5.5. ROC curves of Random Forest Classification Wisconsin Breast Cancer Dataset result.

By comparing ROC curves[10], we want the area under ROC curve to skew towards sensitivity (True positive) to prove it is doing an excellent predicting job. In this case, our proposed method after ROC analyzing shows that it did a excellent prediction.

Conclusion

Dataset	Classification Method	Classification Accuracy
Wisconsin Breast Cancer Dataset	K-Nearest Neighbor with varying ntrees	95%-97%
	Random Forest Classification with varying ntrees and GINI index	96%~98%

The classification of KNN method shown an a classification accuracy ranged from 95% to 97%, whereas the classification of Random Forest method shown a classification accuracy from 96% to 98%. We can conclude that random forest have a slight advantage over K-NN when it comes to prediction accuracy.

From the conclusion table above, we can overall happy to assume that both classification method perform very well in classifying each cases. However, K-NN method do have a really good prediction performance in this particular dataset because of the nature of the attributes which was arranged in scaled order (1-10). Also, we can prove that K-NN perform better in this data since it doesn't really have any noise[11]. Given if the datasets contains more random variables and numerical values, random forest will be a better method to choose.

We believe this 2 classification methods will be very helpful to the physicians and doctors for their second opinion for them before making final decision in classifying patients' health status. By using such efficient machine learning tools, they can make almost very accurate decisions.

References

[1] "Nearest Neighbors Classification." Nearest Neighbors Classification — Scikit-learn 0.18.1 Documentation. N.p., n.d. Web. 07 Dec. 2016.
url:http://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py

[2] 4-1. Chapter 4 Measures of Distance between Samples: Euclidean (n.d.): n. pag. Web.
url: http://84.89.132.1/~michael/stanford/maeb4.pdf

[3] Morris, Stephanie J. "The Pythagorean Theorem." The Pythagorean Theorem. N.p., n.d. Web. 07 Dec. 2016.
url:http://jwilson.coe.uga.edu/emt669/student.folders/morris.stephanie/emt.669/essay.1/pythagorean.html

[4]"R: K-Nearest Neighbour Classification." R: K-Nearest Neighbour Classification. N.p., n.d. Web. 07 Dec. 2016.
url: http://stat.ethz.ch/R-manual/R-devel/library/class/html/knn.html

[5] Tobias Sing, Oliver Sander, Niko Beerenwinkel, Thomas Lengauer (May 16,2013) Package'ROCR'

url:http://rocr.bioinf.mpi-sb.mpg.de/ROCR.pdf

[6] Deepanshu Bhalla (August,2016) Random Forest Explained in Simple Terms
url:http://www.listendata.com/2014/11/random-forest-with-r.html

[7] Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener.(October,2015) Package "Random Forest".
url:https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

[8] Paul Ingles (July 2012) Evaluating classifier results with R Caret

url:http://oobaloo.co.uk/evaluating-classifier-results-with-r-part-2

[9]Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller (May 2015) Package "pROC"
url:https://cran.r-project.org/web/packages/pROC/index.html

[10] Raffael Vogler (June 23 2015) Illustrated Guide to ROC and AUC
url:https://www.r-bloggers.com/illustrated-guide-to-roc-and-auc/

[11] R.Y. Wang, V.C. Storey, C.P. Firth (December 2011) Introduction to noise in data mining
url: http://sci2s.ugr.es/noisydata