WAYNE STATE UNIVERSITY
COLLEGE OF ENGINEERING

COMPUTER SCIENCE DEPARTMENT

Course Project

CSC7810: Data Mining: Algorithms and Applications

Breast Cancer Prediction using Data Mining

Supervised by

Dr. Suzan Arslantürk

Submitted by:

Debadeep Pharikal
Email: dpharikal@wayne.edu
Access id – gu5308
Cell# 248-882-5025

Bipasha Banerjee
Email: bipasha@wayne.edu
Access id – gv1591
Cell# 248-425-1255

Table of Contents

ABSTRACT

This course projects includes analysis on Breast Cancer Prediction using Data Mining based on Breast Cancer Wisconsin (Diagnostic) Dataset from UCI repository. The major goal of this course project is to experiment data mining techniques covered under CSC 7810 - Data Mining: Algorithms and Applications and work towards implementation of those knowledge to develop a near to perfect model in predicting Breast Cancer. The more accurate the model are, more chances of artificial systems to predict if the person is having Breast Cancer. Hence the main outline of this project lies on studying existing literature, draw comparison between Unsupervised, Supervised & Semi-Supervised Learning. This course work also covers novel techniques of predicting Breast Cancer using Semi-Supervised Learning. Additionally, experiments are also performed in Neural Networks to build a robust model implementing backpropagation algorithm using R interface to Keras framework & TensorFlow backend engine and comparisons are drawn using multiple hidden layers and activation functions.

1. Introduction:

1.1 Overview

Breast Cancer is a group of disease in which cells in breast tissue change and divide uncontrollably leading to lump or mass. It is the most common cancer diagnosed among women and is the one of the leading causes of death among women after lung cancer in the United States. It is the most common type of cancer which causes 411,000 annual deaths worldwide.

1.2 Literature Review

In multiple literatures, various machine learning models - both supervised and unsupervised models have been suggested to classify Breast Cancer. However, we find till date, most approaches suggested in the literatures differ mostly in the adopted data mining technique and how to deal with the missing attribute values and labels. An important shortcoming that most of these methods share is that they are either designed for big datasets or have not been tested enough to address the challenge of data scarcity, which is often the case for cancer datasets. We take this opportunity to build our model using different approach that will address this challenge
Below mentioned literatures are studied as part of this contribution.

Method	Reference
Self-training	(Yarowsky, 1995)
SETRED	(Li and Zhou, 2005)
SNNRCE	(Wang et al., 2010)
Tri-training	(Zhou and Li, 2005)
Co-Bagging	(Blum and Mitchell, 1998)
Democratic-Co	(Zhou and Goldman, 2004)

Here is the short summary of these literatures:
Self-Training is a basic semi-supervised learning method where a classifier is initially trained with a small number of labeled examples and this classifier is used to predict the labels of unlabeled samples.
Few literatures already implemented SSL which includes *Self-Training* technique, however as *per our best of knowledge following methods of semi-supervised learning for a breast cancer problem has not been explored before.*
Self-Training with editing (SETRED) is an SSL technique in which a learner keeps on labeling unlabeled examples and retraining itself on an enlarged labeled training set. Since the self-training process may erroneously label some of the unlabeled examples, sometimes the learned hypothesis does not perform well.
Self-training Nearest Neighbor Rule using Cut Edges (SNNRCE) is a variant of the self-training classification method with a different addition mechanism and a fixed learning scheme (1-NN). The mislabeled examples are identified using the local information provided by the neighborhood graph. A statistical test using cut edge weight is used to modify the labels of the misclassified examples.
Tri-training algorithm generates **three** classifiers from the original labeled example set. These classifiers are then refined using unlabeled examples in the tri-training process. In detail, at each round of tri-training, an unlabeled example is labeled for a classifier only if the other two classifiers agree on the labeling, under certain conditions.
Co-Training by Committee is a SSL technique that requires two *views* of the data. It assumes that each example is described using two different feature sets that provide different, complementary information about the instance. Ideally, the two views are conditionally independent, and each view is enough. Co-training first learns a separate classifier for each view using any labeled examples. The most confident predictions of each classifier on the unlabeled data are then used to iteratively construct additional labeled training data.
Democratic co-learning is the methodology in which multiple algorithms instead of multiple views enable learners to label data for each other. The technique leverages off the fact that different learning algorithms have different inductive biases and that better predictions can be made by the voted majority.

1.3 Objective & Contribution

The major motivation of this course project is to develop an effective product using Data Mining in medical domain that will transform the data and reduce dimension to reveal patterns in the dataset and create a robust analysis.

- Our **novel contribution** lies in utilizing multiple semi supervised learning (SSL) methods to build models and demonstrate its comparable accuracy with Supervised model since SSL technique is proven better and useful for a medical domain problem.
- Additionally, we performed end to end analysis and produced comparable results for Supervised & Unsupervised Learning too.
- Furthermore, as a part of Supervised Model analysis, we focused on Neural Network to compare with multiple hidden layers and activation function.

The optimal models are selected based on comparable factors like balanced accuracy, sensitivity, specificity and F1 score, among other factors.

2. Data Analysis

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The data used for this project was collected by the University of Wisconsin and its composed by the biopsy result of 569 patients in Wisconsin hospital. They describe characteristics of the cell nuclei present in the image. The dataset is created by Dr. Wolberg a physician at University of Wisconsin and can be found at UCI repository [Web Link].

Number of instances: 569
Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)

2.1 Attribute Information:

The dataset's features describe characteristics of the cell nuclei on the image.

wdbc.data		
1.	ID number	Identification #
2.	Diagnosis	M = malignant, B = benign
3-32	Ten real-valued features are computed for each cell nucleus:	
a.	radius	mean of distances from center to points on the perimeter
b.	texture	standard deviation of gray-scale values
c.	perimeter	
d.	area	Number of pixels inside contour + ½ for pixels on perimeter
e.	smoothness	local variation in radius lengths
f.	compactness	perimeter^2 / area - 1.0; this dimensionless number is at a minimum with a circular disk and increases with the irregularity of the boundary, but this measure also increases for elongated cell nuclei, which is not indicative of malignancy
g.	concavity	severity of concave portions of the contour
h.	concave points	number of concave portions of the contour
i.	symmetry	
j.	fractal dimension	"coastline approximation" – 1; a higher value corresponds a less regular contour and thus to a higher probability of malignancy

2.2 Data Exploratory Analysis:

We plotted the data on a histogram to understand its distribution. This step is essential fruitful analysis and understand the data pertaining to Breast Cancer disease prediction problem. This type of analysis is also useful when we do not know the right range of value, an attribute can hold especially for a dataset like this.

2.2.1 Data Summary

We observed from data summary that the data frame consists of 32 variables

```
> str(cancer_data)
'data.frame':        569 obs. of  32 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 8
4458202 844981 84501001 ...
```

```
$ diagnosis              : chr  "M" "M" "M" "M" ...
$ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
$ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
$ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
$ area_mean              : num  1001 1326 1203 386 1297 ...
$ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
$ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
$ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
$ concave_points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
$ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
$ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
$ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
$ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
$ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
$ area_se                : num  153.4 74.1 94 27.2 94.4 ...
$ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
$ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
$ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
$ concave_points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
$ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
$ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
$ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
$ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
$ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
$ area_worst             : num  2019 1956 1709 568 1575 ...
$ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
$ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
$ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
$ concave_points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
$ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
$ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
```

```
> head(cancer_data)
        id diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean
1   842302         M       17.99        10.38         122.80    1001.0         0.11840
2   842517         M       20.57        17.77         132.90    1326.0         0.08474
3 84300903         M       19.69        21.25         130.00    1203.0         0.10960
4 84348301         M       11.42        20.38          77.58     386.1         0.14250
5 84358402         M       20.29        14.34         135.10    1297.0         0.10030
6   843786         M       12.45        15.70          82.57     477.1         0.12780
  compactness_mean concavity_mean concave_points_mean symmetry_mean fractal_dimension_mean
1          0.27760         0.3001             0.14710        0.2419                 0.07871
2          0.07864         0.0869             0.07017        0.1812                 0.05667
3          0.15990         0.1974             0.12790        0.2069                 0.05999
4          0.28390         0.2414             0.10520        0.2597                 0.09744
5          0.13280         0.1980             0.10430        0.1809                 0.05883
6          0.17000         0.1578             0.08089        0.2087                 0.07613
  radius_se texture_se perimeter_se area_se smoothness_se compactness_se concavity_se
1    1.0950     0.9053        8.589  153.40      0.006399        0.04904      0.05373
2    0.5435     0.7339        3.398   74.08      0.005225        0.01308      0.01860
3    0.7456     0.7869        4.585   94.03      0.006150        0.04006      0.03832
4    0.4956     1.1560        3.445   27.23      0.009110        0.07458      0.05661
5    0.7572     0.7813        5.438   94.44      0.011490        0.02461      0.05688
6    0.3345     0.8902        2.217   27.19      0.007510        0.03345      0.03672
  concave_points_se symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
1           0.01587     0.03003             0.006193        25.38         17.33          184.60
2           0.01340     0.01389             0.003532        24.99         23.41          158.80
3           0.02058     0.02250             0.004571        23.57         25.53          152.50
4           0.01867     0.05963             0.009208        14.91         26.50           98.87
5           0.01885     0.01756             0.005115        22.54         16.67          152.20
6           0.01137     0.02165             0.005082        15.47         23.75          103.40
  area_worst smoothness_worst compactness_worst concavity_worst concave_points_worst
1     2019.0           0.1622            0.6656          0.7119               0.2654
2     1956.0           0.1238            0.1866          0.2416               0.1860
3     1709.0           0.1444            0.4245          0.4504               0.2430
4      567.7           0.2098            0.8663          0.6869               0.2575
5     1575.0           0.1374            0.2050          0.4000               0.1625
6      741.6           0.1791            0.5249          0.5355               0.1741
  symmetry_worst fractal_dimension_worst
1         0.4601                0.11890
2         0.2750                0.08902
3         0.3613                0.08758
4         0.6638                0.17300
5         0.2364                0.07678
6         0.3985                0.12440
```

We observed that 'id' column has no contribution in class prediction and hence we dropped it.

```
> dim(cancer_data)
[1] 569  31
```

We further check if the dataset has any missing value:

```
> map(cancer_data, function(.x) sum(is.na(.x)))
$diagnosis
[1] 0
$radius_mean
[1] 0
$texture_mean
[1] 0
$perimeter_mean
[1] 0
$area_mean
[1] 0
$compactness_se
[1] 0
$concavity_se
[1] 0
$concave_points_se
[1] 0
$symmetry_se
[1] 0
$fractal_dimension_se
[1] 0
```

```
$smoothness_mean
[1] 0
$compactness_mean
[1] 0
$concavity_mean
[1] 0
$concave_points_mean
[1] 0
$symmetry_mean
[1] 0
$fractal_dimension_mean
[1] 0
$radius_se
[1] 0
$texture_se
[1] 0
$perimeter_se
[1] 0
$area_se
[1] 0
$smoothness_se
[1] 0
```

```
$radius_worst
[1] 0
$texture_worst
[1] 0
$perimeter_worst
[1] 0
$area_worst
[1] 0
$smoothness_worst
[1] 0
$compactness_worst
[1] 0
$concavity_worst
[1] 0
$concave_points_worst
[1] 0
$symmetry_worst
[1] 0
$fractal_dimension_worst
[1] 0
```

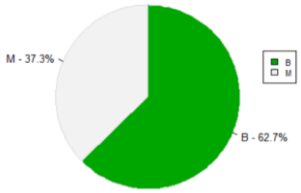
Hence the summarization of data is as follows:

- **Diagnosis is a categorical variable.**
- **All feature values are recoded with four significant digits.**
- **No missing attribute values**
- **Data is well organized**
- **Class distribution: 357 benign, 212 malignant**

2.2.2 Data Visualization

We perform Data Visualization to find which features are most helpful in predicting malignant or benign cancer.

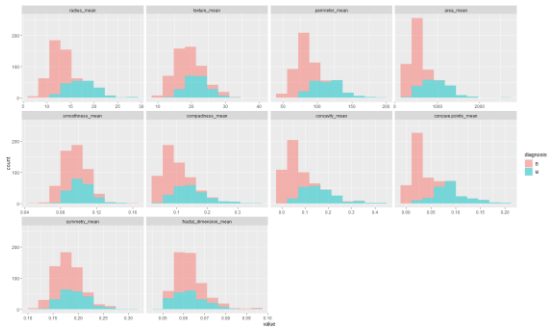
```
> prop.table(diagnosis.table)
        B         M
0.6274165 0.3725835
```

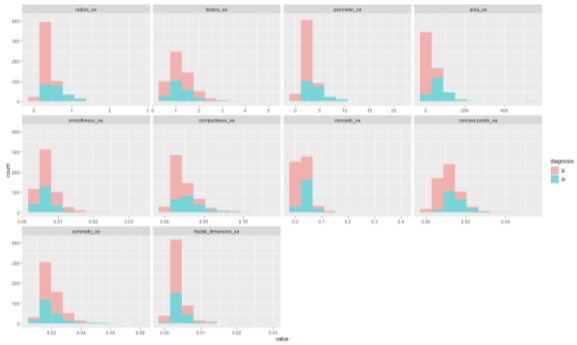


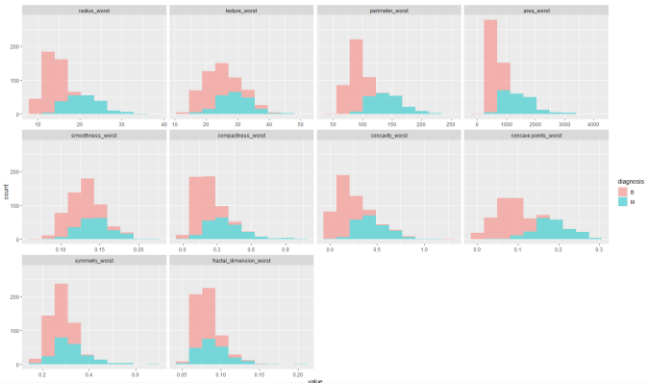
M= Malignant (indicates presence of cancer cells); B= Benign (indicates absence)

357 observations which account for 62.7% of all observations indicating the absence of cancer cells, 212 which account for 37.3% of all observations shows the presence of cancerous cell.
By analyzing the dataset, it is found that it is a **bit unbalanced in its proportions.**

Next plot is using histograms to show how individual factor affecting **malignancy**.





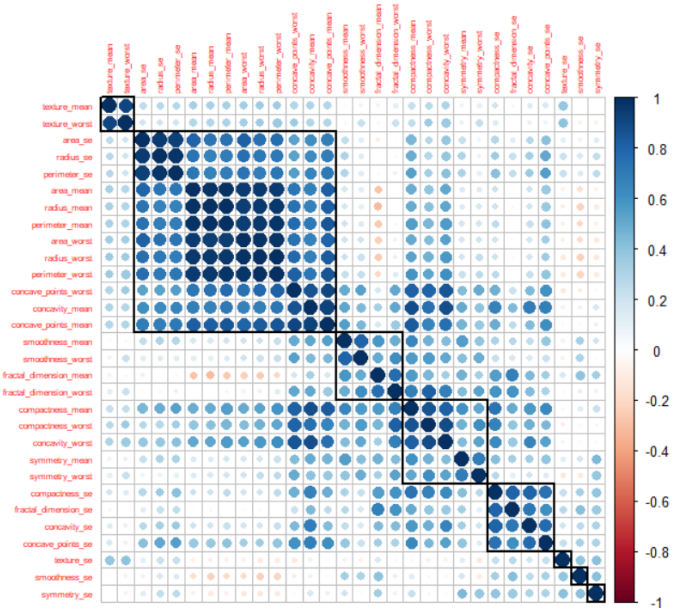


By looking into the histograms:

- We find most of the features are **normally distributed**.
- Comparison of radius distribution by malignancy shows that there is no perfect separation between any of the features.
- We do have good separations for *concave_points_worst, concavity_worst, perimeter_worst, area_mean, perimeter_mean*.
- We do have as well tight superposition for some of the values, like *symmetry_se, smoothness_se*.

2.2.3 Bi-variate/Multivariate Analysis

An important step in exploratory data analysis step is to identify if there is any correlation between any of these variables. By using **Pearson's correlation**, the below plot was created. The positive correlation between two variables is demonstrated through the darkness of blue color, i.e**. darker the blue colored box, stronger is the positive correlation between respective variables**. Similarly, negative correlation between two variables is demonstrated through the darkness of orange color, i.e. **darker the orange colored box, stronger is the negative correlation between respective variables**.



Often, we have features that are highly correlated and those provide redundant information.
It is observed that quite few variables which are co-related. By **eliminating highly correlated features we can avoid a predictive bias for the information contained in these features**. This also shows us, that when we want to make assumptions about the biological/ medical significance of specific features, we need to keep in mind that just

because they are suitable to predicting an outcome, **they are not necessarily causal - they could simply be correlated with causal factors.**

All features with a correlation higher than 0.9 are removed, keeping the features with the lower mean.

```
> print(highlyCorrelated)
 [1]  7  8 23 21  3 24  1 13 14  2
```

We identified 10 variables are highly co-related and we removed those variables. The transformed dataset is 10 variables shorter.

```
> # Check column count after removing correlated variables
> ncol(cancer_data_wcor)
[1] 20
```

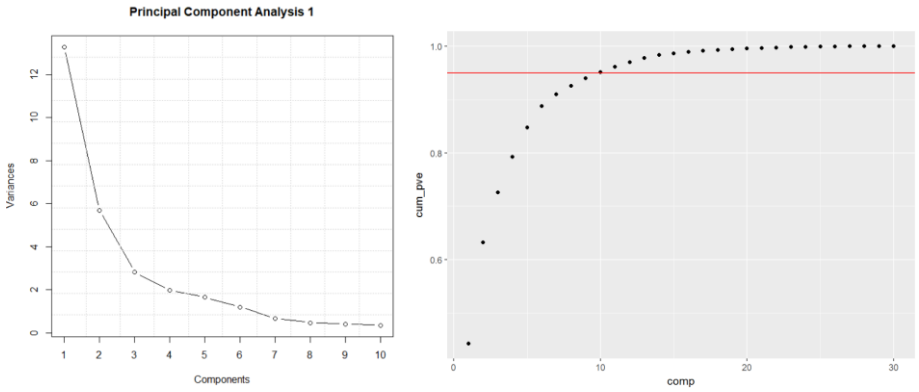
2.3 Dimensionality Reduction:

2.3.1 Principal Component Analysis

Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize. For two-dimensional data, PCA seeks to rotate these two axes so that the new axis *X'* lies along the direction of maximum variation in the data. PCA requires that the axes be perpendicular, so in two dimensions the choice of *X'* will determine *Y'*.

[Source: https://setosa.io/ev/principal-component-analysis/]

```
> summary(pca_res_data1)
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9     PC10
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172 0.69037 0.6457 0.59219
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251 0.01589 0.0139 0.01169
Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010 0.92598 0.9399 0.95157
                          PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19
Standard deviation     0.5421 0.51104 0.49128 0.39624 0.30681 0.28260 0.24372 0.22939 0.22244
Proportion of Variance 0.0098 0.00871 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175 0.00165
Cumulative Proportion  0.9614 0.97007 0.97812 0.98335 0.98649 0.98915 0.99113 0.99288 0.99453
                          PC20   PC21    PC22    PC23   PC24    PC25    PC26    PC27    PC28
Standard deviation     0.17652 0.1731 0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
Proportion of Variance 0.00104 0.0010 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
Cumulative Proportion  0.99557 0.9966 0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
                          PC29    PC30
Standard deviation     0.02736 0.01153
Proportion of Variance 0.00002 0.00000
Cumulative Proportion  1.00000 1.00000
```

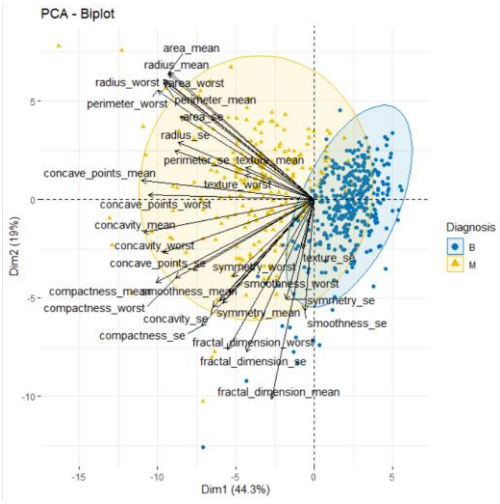


From the above result we found first two components explains the 0.6324 of the variances.

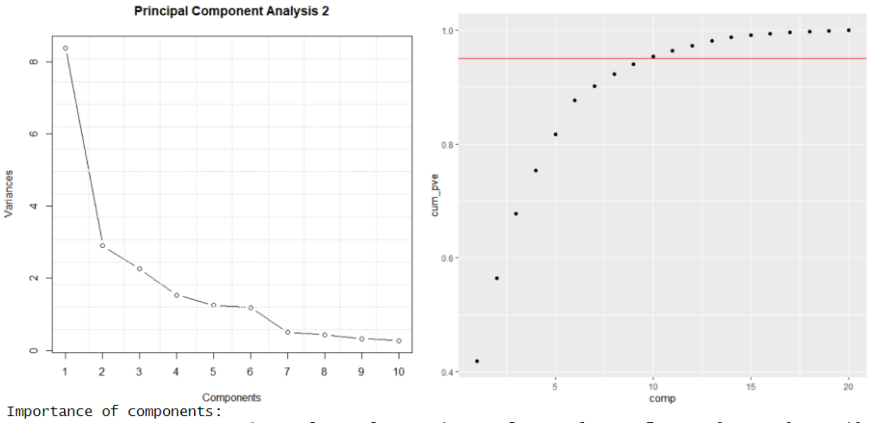
10 principal components explain more than 0.95 of the variances and 17 to explain more than 0.99.

PCA- Biplot:

A biplot simultaneously plots information on the observations and the variables in a multidimensional dataset.

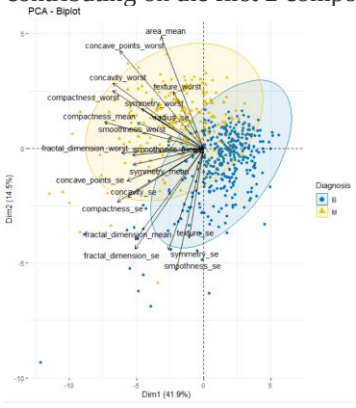


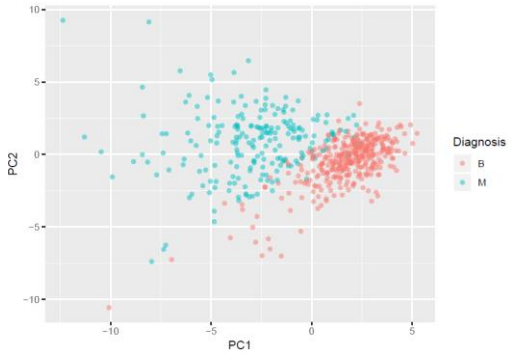
Now we perform Principal Component Analysis without the co-related variables.



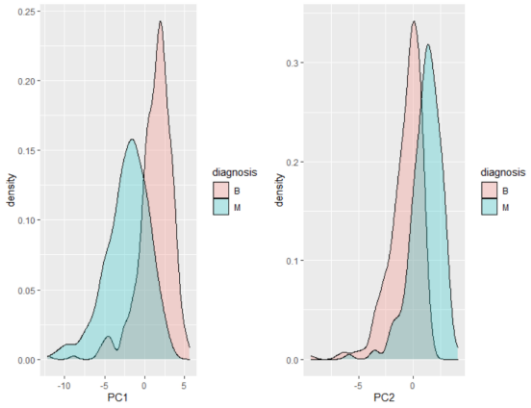

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
Standard deviation     2.8944 1.7019 1.5051 1.23948 1.12116 1.09232 0.71120 0.66260 0.57349 0.52238
Proportion of Variance 0.4189 0.1448 0.1133 0.07682 0.06285 0.05966 0.02529 0.02195 0.01644 0.01364
Cumulative Proportion  0.4189 0.5637 0.6770 0.75377 0.81662 0.87628 0.90157 0.92352 0.93997 0.95361
                          PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19
Standard deviation     0.46342 0.41049 0.40363 0.36237 0.27413 0.23742 0.21163 0.18323 0.15048
Proportion of Variance 0.01074 0.00842 0.00815 0.00657 0.00376 0.00282 0.00224 0.00168 0.00113
Cumulative Proportion  0.96435 0.97277 0.98092 0.98748 0.99124 0.99406 0.99630 0.99798 0.99911
                          PC20
Standard deviation     0.13340
Proportion of Variance 0.00089
Cumulative Proportion  1.00000
```

From the above table it is observed that 95% of the variance is explained with 10 PC's in the transformed dataset. To visualize which variables are the most contributing on the first 2 components, below is plotted:





The data of the first 2 components can be easily separated into two classes. This is caused by the fact that the variance explained by these components is not large. The data can be easily separated.



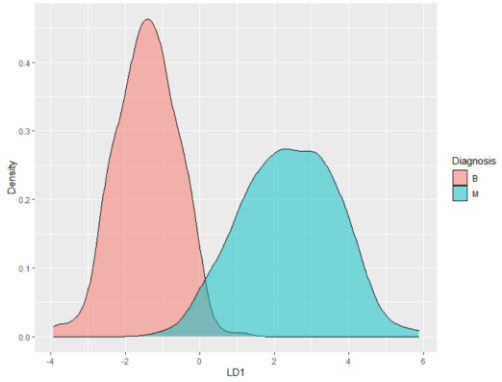
Even we tried visualizing the first 3 components.

As it can be seen from the above plots the first 3 principal components separate the two classes to some extent only, this is expected since the variance explained by these components is not large.
We used the caret preProcess to apply pca with a **0.99 threshold.**

2.3.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is a linear combination of features that characterizes or separates two or more classes of objects or events.

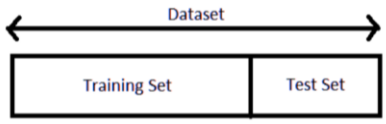
```
> #Linear Discriminant Analysis (LDA)
> lda_res_data <- MASS::lda(diagnosis~., data = cancer_data, center = TRUE, scale = TRUE)
> lda_res_data
Call:
lda(diagnosis ~ ., data = cancer_data, center = TRUE, scale = TRUE)

Prior probabilities of groups:
        B         M
0.6274165 0.3725835

Group means:
  radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean
B    12.14652     17.91476       78.07541  462.7902       0.09247765        0.08008462
M    17.46283     21.60491      115.36538  978.3764       0.10289849        0.14518778
  concavity_mean concave_points_mean symmetry_mean fractal_dimension_mean radius_se
B     0.04605762          0.02571741      0.174186              0.06286739 0.2840824
M     0.16077472          0.08799000      0.192909              0.06268009 0.6090825
  texture_se perimeter_se  area_se smoothness_se compactness_se concavity_se
B   1.220380     2.000321 21.13515   0.007195902      0.02143825   0.02599674
M   1.210915     4.323929 72.67241   0.006780094      0.03228117   0.04182401
  concave_points_se symmetry_se fractal_dimension_se radius_worst texture_worst
B       0.009857653  0.02058381          0.003636051     13.37980      23.51507
M       0.015060472  0.02047240          0.004062406     21.13481      29.31821
  perimeter_worst area_worst smoothness_worst compactness_worst concavity_worst
B        87.00594   558.8994        0.1249595         0.1826725       0.1662377
M       141.37033  1422.2863        0.1448452         0.3748241       0.4506056
  concave_points_worst symmetry_worst fractal_dimension_worst
B           0.07444434      0.2702459             0.07944207
M           0.18223731      0.3234679             0.09152995
Coefficients of linear discriminants:
                                 LD1
radius_mean                -1.075583600
texture_mean                0.022450225
perimeter_mean              0.117251982
area_mean                   0.001569797
smoothness_mean             0.418282533
compactness_mean          -20.852775912
concavity_mean              6.904756198
concave_points_mean        10.578586272
symmetry_mean               0.507284238
fractal_dimension_mean      0.164280222
radius_se                   2.148262164
texture_se                 -0.033380325
perimeter_se               -0.111228320
area_se                    -0.004559805
smoothness_se              78.305030179
compactness_se              0.320560148
concavity_se              -17.609967822
concave_points_se          52.195471457
symmetry_se                 8.383223501
fractal_dimension_se      -35.296511336
radius_worst                0.964016085
texture_worst               0.035360398
perimeter_worst            -0.012026798
area_worst                 -0.004994466
smoothness_worst            2.681188528
compactness_worst           0.331697102
concavity_worst             1.882716394
concave_points_worst        2.293242388
symmetry_worst              2.749992654
fractal_dimension_worst    21.255049570
```



These LDA features are used later as part of Neural Network Analysis.

2.4 Data Preparation:

Data preparation is an important step when building models. We split the dataset into Train set (80%) and Test set (20%).




```
> dim(train_data)
[1] 456  21
> dim(test_data)
[1] 113  21
```

3. Modeling

Data mining techniques have been traditionally used to extract the hidden predictive information in many diverse contexts. 3 different learning methods we experimented in our modeling technique.

- **Supervised learning** aims to learn a function that, given a sample of data and desired outputs, approximates a function that maps inputs to outputs.
- **Semi-supervised learning** aims to label unlabeled data points using knowledge learned from a small number of labeled data points.
- **Unsupervised learning** does not have (or need) any labeled outputs, so its goal is to infer the natural structure present within a set of data points."

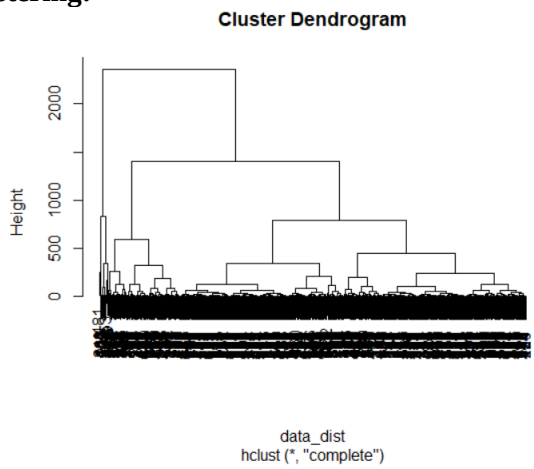
3.1 Model Analysis

Usually datasets contained thousands of examples. Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with the labelled data, to build better classifiers and higher accuracy, located between supervised learning with fully labelled and unsupervised learning without any labelled. Labelled instances, however, are often difficult and expensive to obtain. Meanwhile, unlabeled data may be relatively easy to collect, but there have been few ways to use them. So, we tried to compare all the models to see which suits best.

3.1.1 Un-Supervised methods of Classifying Breast Cancer:

Unsupervised learning is a type of machine learning algorithm used to draw inferences from datasets consisting of input data without labeled responses. Normally when performing unsupervised learning like this, a target variable (i.e. known answer or labels) isn't available. We do have it with this dataset however, this label information is only used to verify the un-supervised outcomes with actual ones.

3.1.1.1 Hierarchical Clustering:



We can see if we cut the dendrogram into 4. It will give us the main clusters and then we have a couple tiny ones on the left.

```
> # Compare hierarchical to actual diagnosis
```

```
     B   M
1    1   93
2  356  107
3    0    9
4    0    3
```

Here we picked four clusters and see that cluster 1 largely corresponds to malignant cells whereas cluster 2 largely corresponds to benign cells. In these two clusters there are (1+107) 108 instances that are not correctly classified.

```
> # No of incorrect Classification in hierarchical clustering
 [1] 108
```

3.1.1.2 K-means Clustering:

We further experimented with k-means clustering algorithm to analyze how it works with our dataset.

There are 2 clusters created corresponding to the actual number of diagnosis. We repeat the algorithm 20 times. Running multiple times, as this will help to find a well performing model.

```
> # Compare k-means to actual diagnosis

     B   M
  1 356  95
  2   1 117
```

In these two clusters there are (1+95) 96 instances that are not correctly classified.

```
> # No of incorrect Classification in k-means clustering
[1] 96
```

If we compare this result with our previous hierarchical clustering as the number of incorrect instances is greater than k-means by 12.

```
> # Compare k-means to hierarchical clustering

     1   2
1    0  94
2  451  12
3    0   9
4    0   3

> # Difference between incorrect instance classified in two clusters
 [1] 12
```

Conclusion

Though k-means clustering performs better than hierarchical one, but it failed to classify **16.87%** (96/569 *100 = 16.87%) of data. This is quite risky in terms of medical domain**. It identified 95 patients as cancer patient who are not actually cancer patient and this percentage will exponentially increase in case of real-world scenario!**

This is fatal for predicting Breast Cancer Problem and hence we discard Unsupervised Learning model and proceed with Supervised and Semi-Supervised Methods.

3.2.2 Supervised methods of Classifying Breast Cancer:

3.2.2.1 Naïve Bayes Model

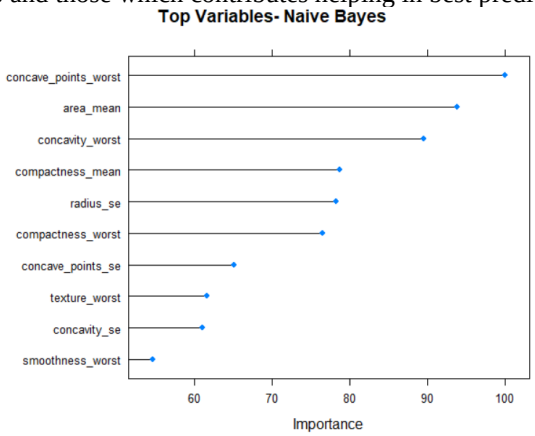
A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

```
> confusionmatrix_naiveb
Confusion Matrix and Statistics

Reference
Prediction  B  M
B 67  5
M  4 37

Accuracy : 0.9204
95% CI : (0.8542, 0.9629)
No Information Rate : 0.6283
P-Value [Acc > NIR] : 9.656e-13
```

```
Kappa : 0.8286

Mcnemar's Test P-Value : 1

Sensitivity : 0.8810
Specificity : 0.9437
Pos Pred Value : 0.9024
Neg Pred Value : 0.9306
Prevalence : 0.3717
Detection Rate : 0.3274
Detection Prevalence : 0.3628
Balanced Accuracy : 0.9123

'Positive' Class : M
```

The topmost significant variables and those which contributes helping in best prediction are as follows:

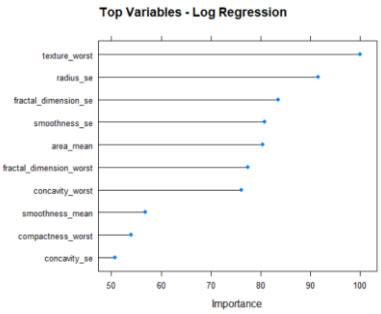


3.2.2.2 Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is binary.

```
> confusionmatrix_logreg
     Confusion Matrix and Statistics

               Reference
     Prediction  B  M
              B 67  0
              M  4 42

                   Accuracy : 0.9646
                     95% CI : (0.9118, 0.9903)
        No Information Rate : 0.6283
        P-Value [Acc > NIR] : <2e-16

                      Kappa : 0.9257

     Mcnemar's Test P-Value : 0.1336

                Sensitivity : 1.0000
                Specificity : 0.9437
             Pos Pred Value : 0.9130
             Neg Pred Value : 1.0000
                 Prevalence : 0.3717
             Detection Rate : 0.3717
       Detection Prevalence : 0.4071
          Balanced Accuracy : 0.9718

           'Positive' Class : M
```

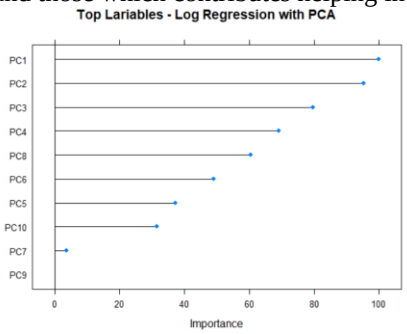
The topmost significant variables and those which contributes helping in best prediction are as follows:



Tried plotting using PCA variables and found that the accuracy is same with the model with non-PCA variables.

```
> confusionmatrix_logreg_pca
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 67  0
         M  4 42

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9257

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 1.0000
            Specificity : 0.9437
         Pos Pred Value : 0.9130
         Neg Pred Value : 1.0000
             Prevalence : 0.3717
         Detection Rate : 0.3717
   Detection Prevalence : 0.4071
      Balanced Accuracy : 0.9718

       'Positive' Class : M
```

The topmost significant variables and those which contributes helping in best prediction are as follows:



3.2.2.3 SVM (with radial Kernel)

Support vector machines (SVM) is a ***supervised learning algorithm*** based on the idea of finding a hyperplane that best separates the features into different domains. **Gaussian RBF (Radial Basis Function)** is popular Kernel method used in SVM models. RBF kernel is a function whose value depends on the distance from the origin or from some point.

Gaussian Kernel is of the following format:

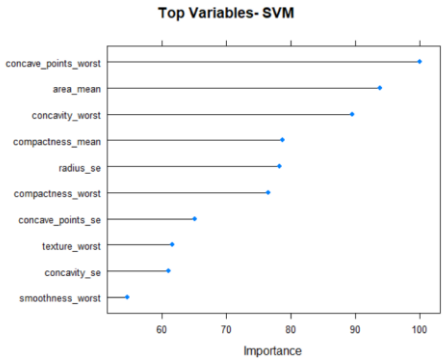
$$K(X_1, X_2) = exponent(-\gamma \|X_1 - X_2\|^2)$$

||X1 — X2 || = Euclidean distance between X1 & X2

```
> confusionmatrix_svm
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 69  2
```

```
         M  2 40

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9242

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9524
            Specificity : 0.9718
         Pos Pred Value : 0.9524
         Neg Pred Value : 0.9718
             Prevalence : 0.3717
         Detection Rate : 0.3540
   Detection Prevalence : 0.3717
      Balanced Accuracy : 0.9621

       'Positive' Class : M
```

The topmost significant variables and those which contributes helping in best prediction are as follows:

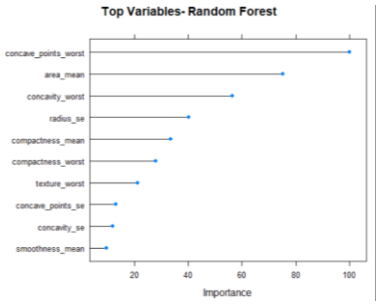


3.2.2.4 Random Forest Classifier

Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes the model's prediction.[Source: https://towardsdatascience.com/understanding-random-forest-58381e0602d2]

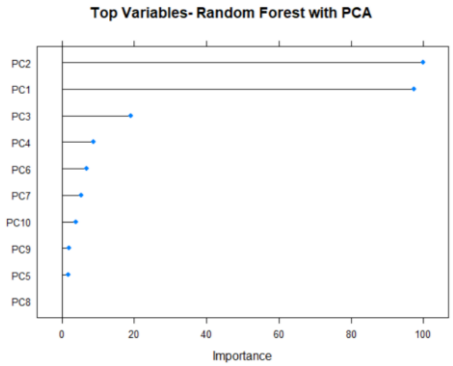
```
> confusionmatrix_randomforest
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 70  3
         M  1 39

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9235

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.9286
            Specificity : 0.9859
         Pos Pred Value : 0.9750
         Neg Pred Value : 0.9589
             Prevalence : 0.3717
         Detection Rate : 0.3451
   Detection Prevalence : 0.3540
      Balanced Accuracy : 0.9572

       'Positive' Class : M
```

The topmost significant variables and those which contributes helping in best prediction are as follows:



Tried checking Random Forest algorithm with PCA variables. Results are as follows:

```
> confusionmatrix_randomforest_pca
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 70  5
         M  1 37

               Accuracy : 0.9469
                 95% CI : (0.888, 0.9803)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : 1.866e-15

                  Kappa : 0.8841

 Mcnemar's Test P-Value : 0.2207

            Sensitivity : 0.8810
            Specificity : 0.9859
         Pos Pred Value : 0.9737
         Neg Pred Value : 0.9333
             Prevalence : 0.3717
         Detection Rate : 0.3274
   Detection Prevalence : 0.3363
      Balanced Accuracy : 0.9334

       'Positive' Class : M
```



The result does not show significant increase.

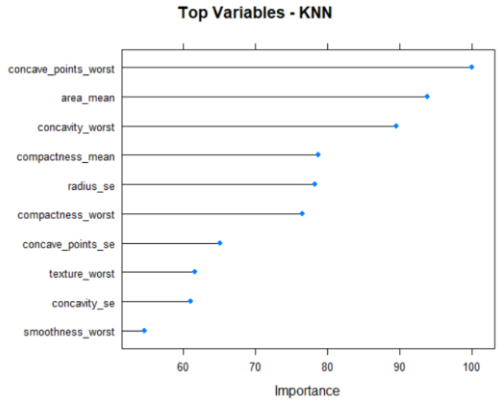
3.2.2.5 K-Nearest Neighbor

The k-nearest neighbor (KNN) algorithm classifies objects on closest training examples in the feature space.

```
> confusionmatrix_knn
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 71  6
         M  0 36

               Accuracy : 0.9469
                 95% CI : (0.888, 0.9803)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : 1.866e-15

                  Kappa : 0.8829
```

```
 Mcnemar's Test P-Value : 0.04123

            Sensitivity : 0.8571
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9221
             Prevalence : 0.3717
         Detection Rate : 0.3186
   Detection Prevalence : 0.3186
      Balanced Accuracy : 0.9286

       'Positive' Class : M
```

Topmost significant variables are as follows:



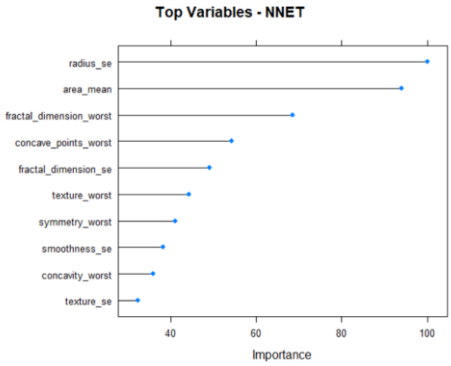
3.2.2.6 ANN: Single Layer Perceptron

"Artificial Neural Networks or ANN is an information processing paradigm that is inspired by the way the biological nervous system work, such as brain process information".

We use 'nnet' package to fit single-hidden-layer neural network, possibly with skip-layer connections.

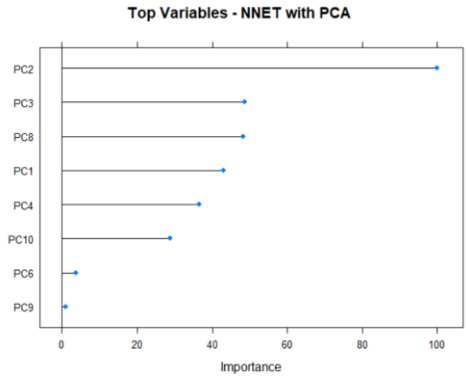
Neural Network (all variables)

```
> confusionmatrix_nnet
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 69  2
         M  2 40

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9242

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9524
            Specificity : 0.9718
         Pos Pred Value : 0.9524
         Neg Pred Value : 0.9718
             Prevalence : 0.3717
         Detection Rate : 0.3540
   Detection Prevalence : 0.3717
      Balanced Accuracy : 0.9621

       'Positive' Class : M
```



Neural Network (PCA)

```
> confusionmatrix_nnet_pca
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 68  1
         M  3 41

               Accuracy : 0.9646
                 95% CI : (0.9118, 0.9903)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9249

 Mcnemar's Test P-Value : 0.6171

            Sensitivity : 0.9762
            Specificity : 0.9577
         Pos Pred Value : 0.9318
         Neg Pred Value : 0.9855
             Prevalence : 0.3717
         Detection Rate : 0.3628
   Detection Prevalence : 0.3894
      Balanced Accuracy : 0.9670

       'Positive' Class : M
```



Neural Network (LDA)

```
> confusionmatrix_nnet_lda
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 70  1
         M  1 41

               Accuracy : 0.9823
                 95% CI : (0.9375, 0.9978)
    No Information Rate : 0.6283
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9621
```

```
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9762
            Specificity : 0.9859
         Pos Pred Value : 0.9762
         Neg Pred Value : 0.9859
             Prevalence : 0.3717
         Detection Rate : 0.3628
   Detection Prevalence : 0.3717
      Balanced Accuracy : 0.9811

       'Positive' Class : M
```

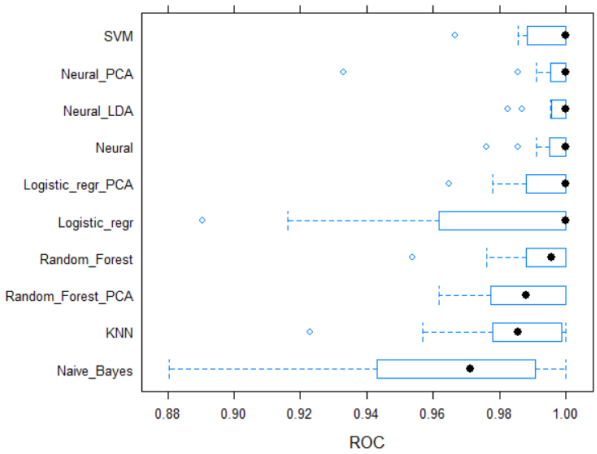
We experimented Neural Network with 3 different types of variables as we see it gives great accuracy in all the models explored. Specifically, for LDA variables it works impressively. We explored this option with LDA variable as we studied in literature, LDA actually works as a linear classifier.

Comparison

We compared all the models processed so far and listed the summary as follows:

```
> summary(models_results)

Call:
summary.resamples(object = models_results)

Models: Naive_Bayes, Logistic_regr, Logistic_regr_PCA, SVM, Random_Forest, Random_Forest_PCA, KNN
, Neural, Neural_PCA, Neural_LDA
Number of resamples: 15

ROC
                     Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
Naive_Bayes       0.8803828 0.9429825 0.9712919 0.9630569 0.9906699    1    0
Logistic_regr     0.8903509 0.9617225 1.0000000 0.9767411 1.0000000    1    0
Logistic_regr_PCA 0.9649123 0.9880383 1.0000000 0.9930090 1.0000000    1    0
SVM               0.9665072 0.9882775 1.0000000 0.9939979 1.0000000    1    0
Random_Forest     0.9539474 0.9880383 0.9956140 0.9913211 1.0000000    1    0
Random_Forest_PCA 0.9617225 0.9772727 0.9880383 0.9866560 1.0000000    1    0
KNN               0.9229167 0.9780702 0.9856459 0.9829127 0.9989035    1    0
Neural            0.9760766 0.9952153 1.0000000 0.9962254 1.0000000    1    0
Neural_PCA        0.9330144 0.9954147 1.0000000 0.9933812 1.0000000    1    0
Neural_LDA        0.9824561 0.9956140 1.0000000 0.9970654 1.0000000    1    0

Sens
                     Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
Naive_Bayes       0.8421053 0.8947368 0.9473684 0.9370175   0.975    1    0
Logistic_regr     0.8947368 0.9473684 1.0000000 0.9687719   1.000    1    0
Logistic_regr_PCA 0.8947368 0.9736842 1.0000000 0.9824561   1.000    1    0
SVM               0.8947368 0.9473684 1.0000000 0.9754386   1.000    1    0
Random_Forest     0.8947368 0.9736842 1.0000000 0.9824561   1.000    1    0
Random_Forest_PCA 0.8421053 0.9473684 1.0000000 0.9649123   1.000    1    0
KNN               0.9473684 0.9473684 1.0000000 0.9791228   1.000    1    0
Neural            0.8947368 0.9473684 1.0000000 0.9789474   1.000    1    0
Neural_PCA        0.8947368 1.0000000 1.0000000 0.9859649   1.000    1    0
Neural_LDA        0.9473684 1.0000000 1.0000000 0.9894737   1.000    1    0

Spec
                     Min.   1st Qu.    Median      Mean   3rd Qu. Max. NA's
Naive_Bayes       0.7272727 0.8257576 0.9090909 0.8873737 0.9583333    1    0
Logistic_regr     0.7272727 0.8750000 1.0000000 0.9297980 1.0000000    1    0
Logistic_regr_PCA 0.8181818 0.9090909 1.0000000 0.9469697 1.0000000    1    0
SVM               0.8333333 0.9090909 1.0000000 0.9590909 1.0000000    1    0
Random_Forest     0.7500000 0.8712121 0.9090909 0.9141414 1.0000000    1    0
Random_Forest_PCA 0.8181818 0.9090909 0.9090909 0.9116162 0.9166667    1    0
KNN               0.7272727 0.7840909 0.8333333 0.8575758 0.9128788    1    0
Neural            0.9090909 0.9128788 1.0000000 0.9646465 1.0000000    1    0
Neural_PCA        0.8333333 0.9090909 1.0000000 0.9530303 1.0000000    1    0
Neural_LDA        0.8181818 0.9128788 1.0000000 0.9585859 1.0000000    1    0
```

From the following plot, one can observe two models, Naive_bayes and Logistic_regression have great variability, depending of the processed sample.



The Neural Network LDA model achieve a great Area Under the ROC Curve with some variability.

```
> confusionmatrix_list_results %>% knitr::kable()
```

	Naive_Bayes	Logistic_regr	Logistic_regr_PCA	SVM	Random_Forest	Random_Forest_PCA	KNN	Neural	Neural_PCA	Neural_LDA
Sensitivity	0.8809524	1.0000000	1.0000000	0.9523810	0.9285714	0.8809524	0.8571429	0.9523810	0.9761905	0.9761905
Specificity	0.9436620	0.9436620	0.9436620	0.9718310	0.9859155	0.9859155	1.0000000	0.9718310	0.9577465	0.9859155
Pos Pred Value	0.9024390	0.9130435	0.9130435	0.9523810	0.9750000	0.9736842	1.0000000	0.9523810	0.9318182	0.9761905
Neg Pred Value	0.9305556	1.0000000	1.0000000	0.9718310	0.9589041	0.9333333	0.9220779	0.9718310	0.9855072	0.9859155
Precision	0.9024390	0.9130435	0.9130435	0.9523810	0.9750000	0.9736842	1.0000000	0.9523810	0.9318182	0.9761905
Recall	0.8809524	1.0000000	1.0000000	0.9523810	0.9285714	0.8809524	0.8571429	0.9523810	0.9761905	0.9761905
F1	0.8915663	0.9545455	0.9545455	0.9523810	0.9512195	0.9250000	0.9230769	0.9523810	0.9534884	0.9761905
Prevalence	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814	0.3716814
Detection Rate	0.3274336	0.3716814	0.3716814	0.3539823	0.3451327	0.3274336	0.3185841	0.3539823	0.3628319	0.3628319
Detection Prevalence	0.3628319	0.4070796	0.4070796	0.3716814	0.3539823	0.3362832	0.3185841	0.3716814	0.3893805	0.3716814
Balanced Accuracy	0.9123072	0.9718310	0.9718310	0.9621060	0.9572435	0.9334339	0.9285714	0.9621060	0.9669685	0.9810530

Here LDA worked as a classifier and posteriorly it reduced the dimensionality of the dataset and the neural network performed the classification task. Since ANN gave highest accuracy to the Breast Cancer dataset, our further work will be with neural network to experiment how multi-layer perceptron model will impact the accuracy. Also, **high accuracy for all the models motivated us to fix the problem with unbalanced dataset as mentioned earlier**. Hence, we will be exploring Multi-Layer Perceptrons in the following section with scaled data.

3.2.2.7 Multi-Layer Perceptron

The application was developed using R interface to Keras framework and TensorFlow backend engine in order to **train a sequential model which uses the backpropagation algorithm**. The basic principle of multi-layer perceptrons is the parallelism and the mathematical background. Parallelism is based on computation of the neurons at the same layer can be independent from each other. If we treat **the input layer, the hidden layer and the output layer as three nodes in a Markov Chain model, where the computation at each layer except for the input layer is dependent on the computation at its subsequent lower layer.** Hence to apply our learning in this course we are motivated to work with **Multi-layer Perceptron**.

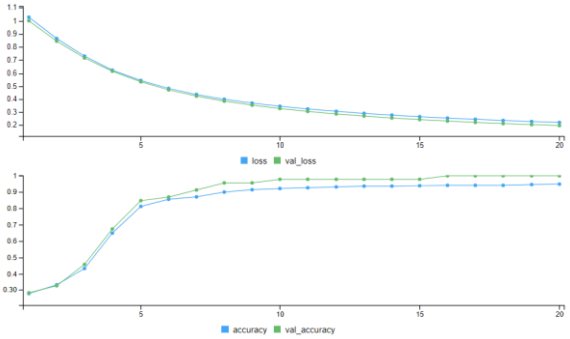
Handling the data problem: The original data analysis revealed to us that the features have different order of magnitude. In order to avoid a situation where features with higher values have greater contribution to the outcome of the model, each feature was normalized using R build-in scale() function. This function **transforms each value by subtracting the mean value of the feature and dividing it by the standard deviation.**

Experiments with Activation Function:

ReLU

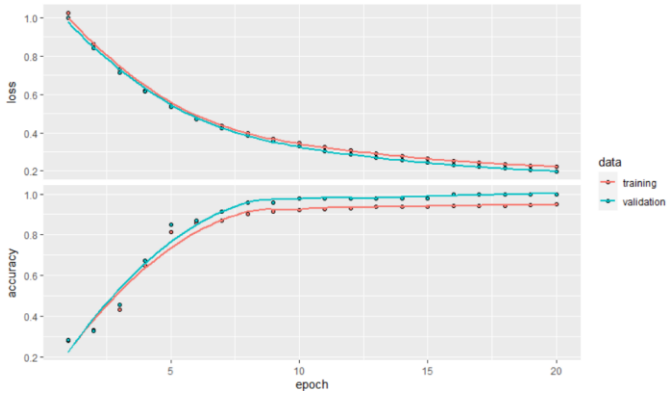
```
Train on 410 samples, validate on 46 samples
Epoch 1/20
410/410 [==============================] - 1s 3ms/sample - loss: 1.0276 - accuracy: 0.2780 - val_
loss: 1.0005 - val_accuracy: 0.2826
Epoch 2/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.8640 - accuracy: 0.3317 - val_
loss: 0.8440 - val_accuracy: 0.3261
Epoch 3/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.7297 - accuracy: 0.4317 - val_
loss: 0.7151 - val_accuracy: 0.4565
Epoch 4/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.6219 - accuracy: 0.6488 - val_
loss: 0.6137 - val_accuracy: 0.6739
Epoch 5/20
```

```
410/410 [==============================] - 1s 1ms/sample - loss: 0.5427 - accuracy: 0.8122 - val_
loss: 0.5325 - val_accuracy: 0.8478
Epoch 6/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.4818 - accuracy: 0.8561 - val_
loss: 0.4709 - val_accuracy: 0.8696
Epoch 7/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.4357 - accuracy: 0.8707 - val_
loss: 0.4233 - val_accuracy: 0.9130
Epoch 8/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.3985 - accuracy: 0.9000 - val_
loss: 0.3847 - val_accuracy: 0.9565
Epoch 9/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.3691 - accuracy: 0.9146 - val_
loss: 0.3541 - val_accuracy: 0.9565
Epoch 10/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.3450 - accuracy: 0.9220 - val_
loss: 0.3280 - val_accuracy: 0.9783
Epoch 11/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.3243 - accuracy: 0.9268 - val_
loss: 0.3058 - val_accuracy: 0.9783
Epoch 12/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.3064 - accuracy: 0.9317 - val_
loss: 0.2865 - val_accuracy: 0.9783
Epoch 13/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.2908 - accuracy: 0.9366 - val_
loss: 0.2703 - val_accuracy: 0.9783
Epoch 14/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.2775 - accuracy: 0.9366 - val_
loss: 0.2559 - val_accuracy: 0.9783
Epoch 15/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.2652 - accuracy: 0.9390 - val_
loss: 0.2432 - val_accuracy: 0.9783
Epoch 16/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.2544 - accuracy: 0.9415 - val_
loss: 0.2320 - val_accuracy: 1.0000
Epoch 17/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.2450 - accuracy: 0.9415 - val_
loss: 0.2217 - val_accuracy: 1.0000
Epoch 18/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.2363 - accuracy: 0.9415 - val_
loss: 0.2126 - val_accuracy: 1.0000
Epoch 19/20
410/410 [==============================] - 1s 1ms/sample - loss: 0.2281 - accuracy: 0.9463 - val_
loss: 0.2048 - val_accuracy: 1.0000
Epoch 20/20
410/410 [==============================] - 0s 1ms/sample - loss: 0.2210 - accuracy: 0.9488 - val_
loss: 0.1973 - val_accuracy: 1.0000
```




```
> test.result.32
$loss
[1] 0.23472

$accuracy
[1] 0.920354
```

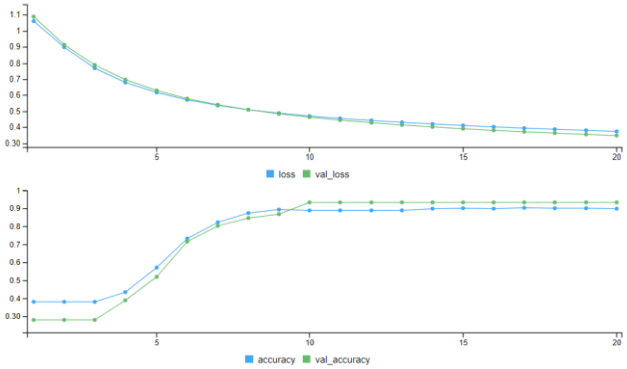
Observation: We trained the model for 20 epochs. The accuracy of the model increases significantly during the first 5 epochs, after which stabilizes. After epoch 5, the loss values continue to drop, and the accuracy keeps increasing, however at a lower rate.



Sigmoid

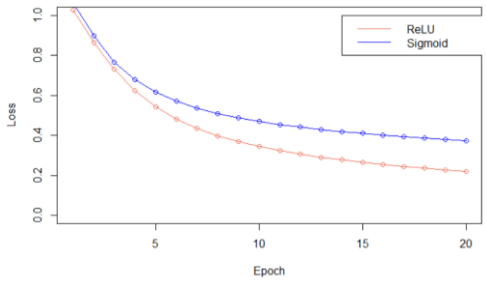
```
Train on 410 samples, validate on 46 samples
Epoch 1/20
410/410 [==============================] - 1s 3ms/sample - loss: 1.0613 - accuracy: 0.3829 - val_
loss: 1.0893 - val_accuracy: 0.2826
Epoch 2/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.8987 - accuracy: 0.3829 - val_
loss: 0.9140 - val_accuracy: 0.2826
Epoch 3/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.7676 - accuracy: 0.3829 - val_
loss: 0.7877 - val_accuracy: 0.2826
Epoch 4/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.6787 - accuracy: 0.4366 - val_
loss: 0.6961 - val_accuracy: 0.3913
Epoch 5/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.6159 - accuracy: 0.5732 - val_
loss: 0.6286 - val_accuracy: 0.5217
Epoch 6/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.5700 - accuracy: 0.7341 - val_
loss: 0.5775 - val_accuracy: 0.7174
Epoch 7/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.5349 - accuracy: 0.8244 - val_
loss: 0.5394 - val_accuracy: 0.8043
Epoch 8/20
410/410 [==============================] - 1s 3ms/sample - loss: 0.5081 - accuracy: 0.8756 - val_
loss: 0.5088 - val_accuracy: 0.8478
Epoch 9/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4871 - accuracy: 0.8951 - val_
loss: 0.4826 - val_accuracy: 0.8696
Epoch 10/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4686 - accuracy: 0.8902 - val_
loss: 0.4622 - val_accuracy: 0.9348
Epoch 11/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4540 - accuracy: 0.8902 - val_
loss: 0.4435 - val_accuracy: 0.9348
Epoch 12/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4410 - accuracy: 0.8902 - val_
loss: 0.4275 - val_accuracy: 0.9348
Epoch 13/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4297 - accuracy: 0.8902 - val_
loss: 0.4133 - val_accuracy: 0.9348
Epoch 14/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4193 - accuracy: 0.9000 - val_
loss: 0.4014 - val_accuracy: 0.9348
Epoch 15/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4103 - accuracy: 0.9024 - val_
loss: 0.3898 - val_accuracy: 0.9348
Epoch 16/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.4016 - accuracy: 0.9000 - val_
loss: 0.3798 - val_accuracy: 0.9348
Epoch 17/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.3938 - accuracy: 0.9049 - val_
loss: 0.3707 - val_accuracy: 0.9348
Epoch 18/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.3864 - accuracy: 0.9024 - val_
loss: 0.3627 - val_accuracy: 0.9348
Epoch 19/20
410/410 [==============================] - 1s 2ms/sample - loss: 0.3797 - accuracy: 0.9024 - val_
loss: 0.3545 - val_accuracy: 0.9348
Epoch 20/20
```

```
410/410 [==============================] - 1s 2ms/sample - loss: 0.3730 - accuracy: 0.9000 - val_
loss: 0.3473 - val_accuracy: 0.9348
```

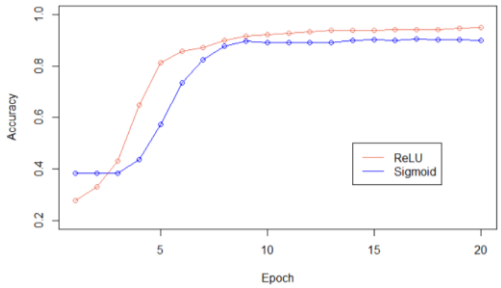



```
> evaluate(ann.sigmoid, test.x, test.y, verbose = 0)
$loss
[1] 0.39895

$accuracy
[1] 0.8584071
```

From below plot we can see how the loss decreases as the training progresses. It is depicted that the curves have similar shape however, the loss corresponding to ReLU function seems to decrease slightly faster but most importantly, it settles at a lower value than the Sigmoid curve.



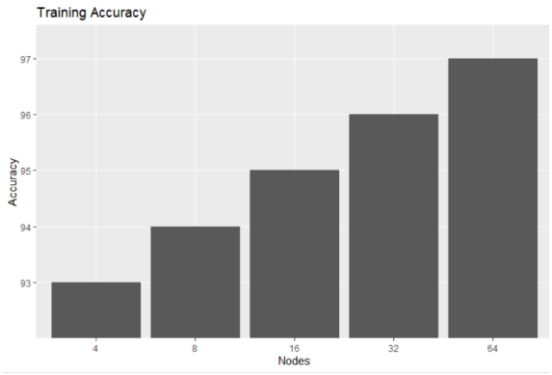
Following the loss for each function, it was examined how the training accuracy of the model changes as the training progresses. On the below plot we can see the values for both activation functions. The sigmoid curve follows the familiar "S" shape and results on slightly lower accuracy. The ReLU curve has the expected shape based on its function and at the end of the training phase has achieved slightly higher accuracy.



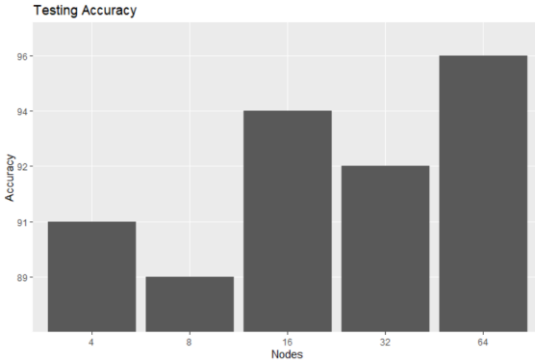
We also examined the architecture to see how many nodes on hidden layer gives best accuracy. For that purpose, models with 4, 8, 16 and 64 nodes on the hidden layer were created and compared to the one already examined with 32 nodes on the hidden layer.

Results

The below bar plot shows the training accuracy of the above models (epoch = 20). The model with the 64 nodes appears to have very high training accuracy (97%).



The bar plot below describes the findings regarding the testing accuracy for the above models. It appears that the model with the worst accuracy is the one with 8 nodes on its hidden layer whereas the models with 16 and 64 nodes have the highest accuracy.




```
> # Confusion Matrix results of all models
> table(prediction.4, test.y)
            test.y
prediction.4  0  1
           0 32  0
           1 10 71
> table(prediction.8, test.y)
            test.y
prediction.8  0  1
           0 32  2
           1 10 69
> table(prediction.16, test.y)
             test.y
prediction.16  0  1
            0 37  2
            1  5 69
> table(prediction.32, test.y)
             test.y
prediction.32  0  1
            0 35  2
            1  7 69
> table(prediction.64, test.y)
             test.y
prediction.64  0  1
            0 40  2
            1  2 69
```

Therefore, we see with **changing the activation function and number of hidden layers does impact the prediction of result**.

Comparison

We compared and found "**Neural Network**" is the best model for Supervised Learning. While analyzing MLP, we fixed the data imbalance issue which is the biggest hindrance for working with an imbalance dataset like this. Accuracy of the MLP model is approximately near to SLP however MLP stand out better owing to data scaling factor. However, the topic is debatable and we can try increasing accuracy of this Multi-layer Perceptron by varying learning rate, batch size, decay rate and loss ratio which is the future scope of this project. Hence to summarize for a Breast Cancer problem with FNA dataset can be proceeded with Neural Networks.

3.2.3 Semi-Supervised methods of Classifying Breast Cancer:

The crux of SSL is to learn from unlabeled as much as from labeled data to predict accurate data (Chapelle et al., 2006). In SSL, data can be separated into two sets: $L = \{x_1, \ldots, x_l\}$ with its known labels $Y_l = \{y_1, \ldots, y_l\}$, and $U = \{x_{l+1}, \ldots, x_{l+u}\}$ for which labels are unknown.

In our SSL methods we tried to implement two different setting such as transductive and inductive learning.

- ***Transductive learning*** concerns the problem of predicting the labels of the unlabeled samples provided during the training phase.
- ***Inductive learning*** considers the labeled and unlabeled data provided as the training samples, and its objective is to predict the label of unseen data.

SVM with the RBF kernel function (kernel matrix using the Gaussian radial basis function (RBF)) is used as **base classifier** and benchmark supervised classifier in all comparisons. The semi-supervised methods evaluated here are: *selfTraining*, *setred*, *coBC*, *triTraining* and *Democratic co-learning*. ***Out of which setred, coBC, triTraining and Democratic co-learning method is our noble contribution and is compared against traditional selfTraining method. Brief algorithm is described as follows:***

Training Methods:

Our training approach includes six labeled methods. Detail is mentioned as follows:

- **Addition mechanism** describes a scheme in which the enlarged labeled set (EL) is formed. In incremental scheme, the algorithm starts with $EL = L$ and adds, step by step. Another scheme is amending, which differs from incremental in that it can iteratively add or remove any instance that meets a certain criterion.
- **Classifiers** refers to whether it uses one or multiple classifiers during the enlarging phase of the labeled set. Multi-classifier methods combine the learned hypotheses with several classifiers to predict the class of unlabeled instances.
- **Learning** specifies whether the models are constituted by the single or multiple learning algorithms. Multi-learning approaches are closely linked with multi-classifier models; a multi-learning method is itself a multi-classifier method in which the different classifiers come from different learning methods. On the other hand, a single-learning approach can be linked to both single and multi-classifiers.
- **Teaching** is a mutual-teaching approach, the classifiers teach each other their most confident predicted examples. Each *Ci* classifier has its own *ELi* which uses for training at each stage. *ELi* is increased with the most confident labeled examples obtained from remaining classifiers. By contrast, the self-teaching property refers to those classifiers that maintain a single EL.
- **Stopping criteria** This is related to the mechanism used to stop the self-labeling process. It is an important factor since it influences the size of EL and therefore the learned hypothesis. Some of the approaches for this are: (i) repeat the self-labeling process until a portion of U has been exhausted, (ii) establish a limited number of iterations, and (iii) the learned hypothesis remains stable between two consecutive stages.

Method	Addition mechanism	Classifiers	Learning paradigm	Teaching	Stopping criteria
Self-training	incremental	single	single	self	i
SETRED	amending	single	single	self	i
SNNRCE	amending	single	single	self	i
Tri-training	incremental	multi	single	mutual	iii
Co-Bagging	incremental	multi	single	mutual	i
Democratic-Co	incremental	multi	multi	mutual	iii

Individual approach for each algorithm implemented are discussed are as follows:

SETRED

SETRED initiates the self-labeling process by training a model from the original labeled set. In each iteration, the **learner** function detects unlabeled examples for which it makes the most confident prediction and labels those examples according to the **pred** function. The identification of mislabeled examples is performed using a neighborhood graph created from the distance matrix.

SNNRCE

SNNRCE initiates the self-labeling process by training a 1-NN from the original labeled set. This method attempts to reduce the noise in examples by labeling those instances with no cut edges in the initial stages of self-labeling learning. These highly confident examples are added into the training set. The remaining

examples follow the standard self-training process until a minimum number of examples will be labeled for each class.

Tri-Training

Tri-Training initiates the self-labeling process by training three models from the original labeled set, using the **learner** function. In each iteration, the algorithm detects unlabeled examples on which two classifiers agree with the classification and includes these instances in the enlarged set of the third classifier under certain conditions. The generation of the final hypothesis is produced via the majority voting. The iteration process ends when no changes occur in any model during a complete iteration.

Co-Training by Committee

CoBC is a semi-supervised learning algorithm with a co-training style. The method trains an **ensemble of diverse classifiers**. To promote the initial diversity the classifiers are trained from the reduced set of labeled examples by **Bagging**. This algorithm trains N classifiers with the learning scheme defined in the learner argument using a reduced set of labeled examples. For each iteration, an unlabeled example is labeled for a classifier if the most confident classifications assigned by the other **N-1** classifiers agree on the labeling proposed. The unlabeled examples candidates are selected randomly from a pool of size u.

Democratic co-learning

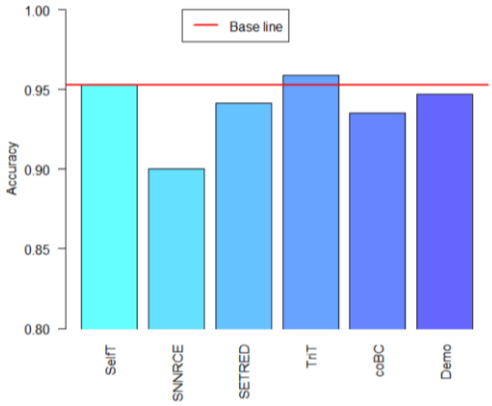
Demo is a semi-supervised learning algorithm with a co-training style. This algorithm trains N classifiers with different learning schemes defined in list gen.learners. During the iterative process, the multiple classifiers with different inductive biases label data for each other.

Training Approach

- To support semi-supervised approach, 70 percent of training data is unlabeled and replaced with NA.
- We used k-SVM as base classifier for all our models.
- In case of democratic method more than one base classifier - KNN, SVM and decision trees(C5.0) are used
- Finally, these models are compared using a supervised classifier that is built on k-SVM as this is the base classifier for all our SSL models.

We compare the accuracy as follows:

```
> acc
    SelfT    SNNRCE    SETRED      TriT     coBC      Demo
0.9529412 0.9000000 0.9411765 0.9588235 0.9352941 0.9470588
> acc.svm
[1] 0.9529412
```



Comparison

Here we find Tri-training obtains the most accurate results and the accuracy is more than traditional supervised learning method. Even Self-training learning method of SSL performed well and provided similar accuracy like supervised model but less than tri-training method. So, a semi-supervised learning performed better than supervised learning in our case.

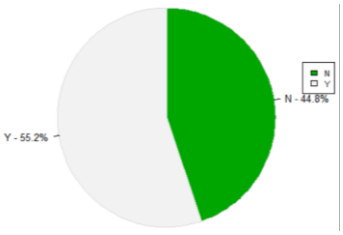
Additional Analysis (with different dataset)

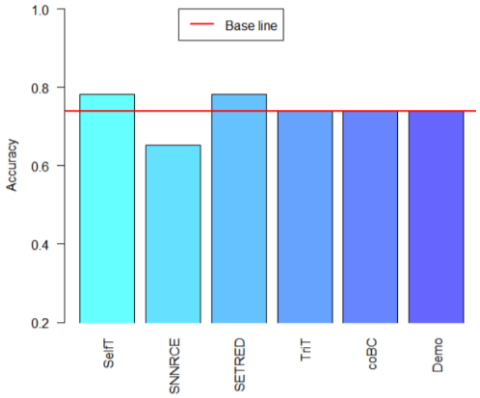
We performed additional analysis by implementing **only on SSL** models using a different dataset(Source:Link) and just to see if novel Semi supervised methods are performing better than supervised method. No other metrics is taken into consideration as the dataset has much less records after outlier removal.

Attribute Information: There are 10 predictors, all quantitative, and a binary dependent variable like the original dataset, indicating the presence or absence of breast cancer.
The predictors are anthropometric data and parameters which can be gathered in routine blood analysis.
Prediction models based on these predictors, if accurate, can potentially be used as a biomarker of breast cancer.

Quantitative Attributes:
Age (years)
BMI (kg/m2)
Glucose (mg/dL)
Insulin (µU/mL)
HOMA
Leptin (ng/mL)
Adiponectin (µg/mL)
Resistin (ng/mL)
MCP-1(pg/dL)

Labels:
1=Healthy controls
2=Patients





Result: Our additional experiment is successful as we can see Semi-Supervised Learning outperform supervised learning algorithm. We see 4 out of 6 methods SSL methods performed better than the Supervised method. We could not explore about the accuracy metric of this analysis owing to time-constraint.

4. Conclusion

Data mining literature offer some nice classification techniques. But when we implement well known effective classification techniques, the results are found unreliable. The efficacy of the technique comes under scrutiny. We tried finding an optimal method by solving data imbalance issue and we overcome it by data scaling. After fixing the problem, we also implemented robust way of implementing neural network using *R interface to Keras framework and TensorFlow*. The multi-layer perceptron model gave almost same accuracy as compared to single layer perceptron model tested before with imbalanced dataset.

Here, the proposal is about an integrated framework, which ensures the reliability of the class labels assigned to a dataset whose class labels are unknown. Heterogeneous datasets with unknown class labels but known number of classes, after being treated through all the novel SSL models would be able to find the class labels for a significant portion of the data and may be accepted with reliability. Even though our novel methods got little less accuracy than supervised models, however, to solve a typical problem like this where label data is often difficult and expensive to obtain, **we propose Semi-Supervised learning mechanisms**.

References

1. Analysis of Semi-Supervised Learning with the Yarowsky Algorithm by GholamReza Haffari and Anoop Sarkar School of Computing Science Simon Fraser University 2007
2. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the 11th Annual Conference on Computational Learning Theory, New York, NY, pp. 92–100 (1998)Google Scholar

3. Li, Ming & Zhou, Zhi-Hua. (2005). SETRED: Self-training with editing. 3518. 611-621. 10.1007/11430919_71. https://www.researchgate.net/publication/220895380_SETRED_Self-training_with_editing
4. Wang, Yu & Xu, Xiaoyan & Zhao, Haifeng & Hua, Zhongsheng. (2010). Semi-supervised learning based on nearest neighbor rule and cut edges. Knowledge-Based Systems. 23. 547-554. 10.1016/j.knosys.2010.03.012.
5. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers Zhi-Hua Zhou, Member, IEEE and Ming L
6. Yan Zhou and Sally Goldman.*Democratic co-learning*. In IEEE 16th International Conference on Tools with Artificial Intelligence (ICTAI), pages 594-602. IEEE, Nov 2004. doi: 10.1109/ICTAI.2004.48.
7. https://cran.r-project.org/web/packages/ssc/vignettes/ssc.pdf
8. https://towardsdatascience.com/deep-learning-in-winonsin-breast-cancer-diagnosis-6bab13838abd
9. https://www.freecodecamp.org/news/how-to-program-a-neural-network-to-predict-breast-cancer-in-only-5-minutes-23289d62a4c1/
10. https://towardsdatascience.com/understanding-the-different-types-of-machine-learning-models-9c47350bb68a

[Note: Code Snippets Document is moved separately due to high numbers of LOC]