CIS8695 Final Project

Predicting Breast Cancer malign or Benign using different predicting models

Submitted By –
Shekha Saxena

Table of Contents

Project Description

Breast cancer (BC) is one of the most common cancers among women worldwide. The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumours can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling. We developed 4 robust classifiers to predict if a tumor is malign or benign based on the dataset recorded by University of Wisconsin Hospital at Madison which includes information regarding the texture, size etc. of the tumor. In this project, we reviewed the traditional and advanced deep learning approaches, and recent advances in this field. We show that with sufficient preprocessing and selecting the right feature set, and resampling to address imbalanced datasets, even simple logistic regression and neural network models give superior performance.

Dataset and features

The dataset has the following features represented in columns:

Id - ID number
Diagnosis - The diagnosis of breast tissues (M = malignant, B = benign)
radius_mean - mean of distances from center to points on the perimeter
texture_mean - standard deviation of gray-scale values
perimeter_mean - mean size of the core tumor
area_mean
smoothness_mean - mean of local variation in radius lengths
compactness_mean - mean of perimeter^2 / area - 1.0
concavity_mean - mean of severity of concave portions of the contour
concave points_mean - mean for number of concave portions of the contour
symmetry_mean
fractal_dimension_mean - mean for "coastline approximation" - 1
radius_se - standard error for the mean of distances from center to points on the perimeter
texture_se - standard error for standard deviation of gray-scale values
perimeter_se
area_se
smoothness_se - standard error for local variation in radius lengths
compactness_se - standard error for perimeter^2 / area - 1.0
concavity_se - standard error for severity of concave portions of the contour
concave points_se - standard error for number of concave portions of the contour

symmetry_se
fractal_dimension_se - standard error for "coastline approximation" - 1
radius_worst - "worst" or largest mean value for mean of distances from center to points on the perimeter
texture_worst - "worst" or largest mean value for standard deviation of gray-scale values
perimeter_worst
area_worst
smoothness_worst - worst" or largest mean value for local variation in radius lengths
compactness_worst - "worst" or largest mean value for perimeter^2 / area - 1.0
concavity_worst - "worst" or largest mean value for severity of concave portions of the contour
concave points_worst - "worst" or largest mean value for number of concave portions of the contour
symmetry_worst
fractal_dimension_worst - "worst" or largest mean value for "coastline approximation" – 1

Data Exploration and Preprocessing

For data preprocessing, firstly we need to check if there are any null values in the dataset, which we do in R as below:

```
#check missing values
colSums(is.na(mydata))
```

From the above result we get to know that, while fetching the dataset there is a column X which is also being fetched, and we do not require it. There is another column ID – which we do not require as it's the ID number of the patient and will not help in predicting if a cancer is malign or benign.

```
#removing unnecessary data columns
mydata <- mydata[ , -c(1, 33)]
```

On checking values of the dataset using head(mydata) we find that the values of the category diagnosis is subjective i.e. it's "M" – for Malign and "B" – for Benign. For this reason we need to change the values for Malign to be as 1 and Benign to be as 0.

We do this because we need our data to categorical in order to use categorical predictive models such as Logistic, etc. We make this change in the dataset with the following command

```
#changing to binary
mydata$diagnosis = ifelse( mydata$diagnosis=="M", 1, 0)
```

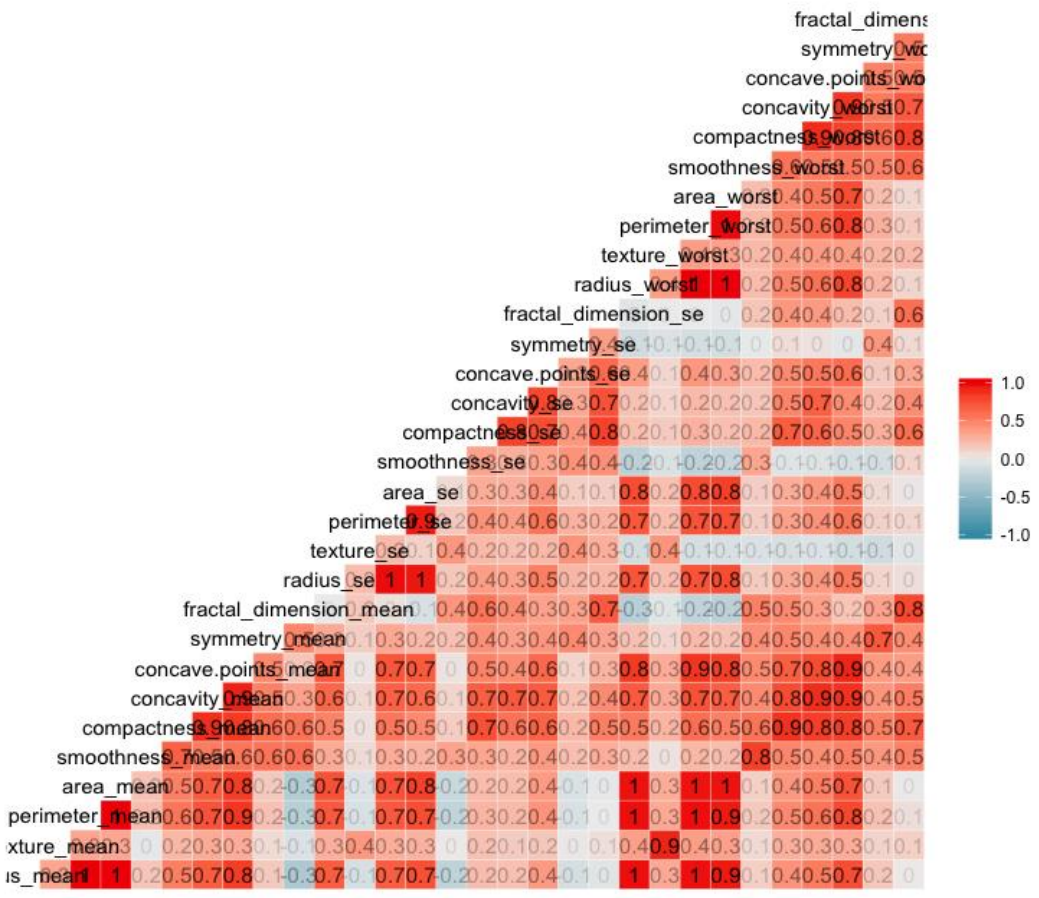
Now, we need to check multicollinearity between the features. We need to check and remove variable showing multicollinearity as they will not help us in making and predicting a robust model. We can check multicollinearity by plotting a graph and my using a function in R which are as follows:

```
#check multicollinearity
ggcorr(mydata[, -1],
      label = TRUE,
      label_alpha = TRUE)

cor(mydata[, -1])
```

As we can see there are many columns interdependent on each other, hence, showing multicollinearity.

Looking at the matrix, we can immediately verify the presence of multicollinearity between some of our variables. For instance, the **radius_mean** column has a correlation of 1 and 0.99 with **perimeter_mean** and **area_mean** columns, respectively. This is probably because the three columns essentially contain the same information, which is the physical size of the observation (the cell). Therefore, we should only pick one of the three columns when we go into further analysis.

Another place where multicollienartiy is apparent is between the "mean" columns and the "worst" column. For instance, the **radius_mean** column has a correlation of 0.97 with the **radius_worst** column. In fact, each of the 10 key attributes display very high (from 0.7 up to 0.97) correlations between its "mean" and "worst" columns. This is somewhat inevitable, because the "worst" columns are essentially just a subset of the "mean" columns; the "worst" columns are also the "mean" of some values (the three largest values among all observations). Therefore, I think we should discard the "worst" columns from our analysis and only focus on the "mean" columns.



Similarly, it seems like there is multicollinearity between the attribute’s **compactness**, **concavity**, and **concave points**. Just like what we did with the size attributes, we should pick only one of these three attributes that contain information on the shape of the cell. I think **compactness** is an attribute name that is straightforward, so I will remove the other two attributes.

We will now go head and drop all unnecessary columns.

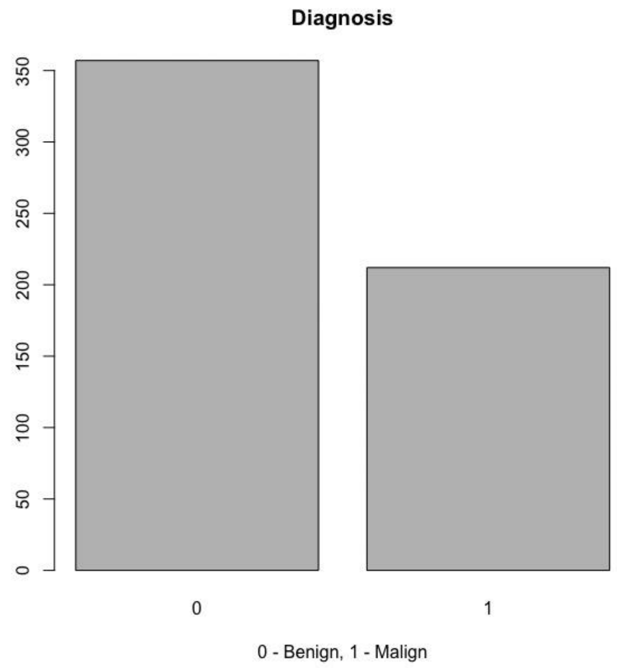
#removing all worst columns, perimeter and area

mydata <- mydata[, -c(4,5,14,15,22:31)]

#removing concavity and concave points columns

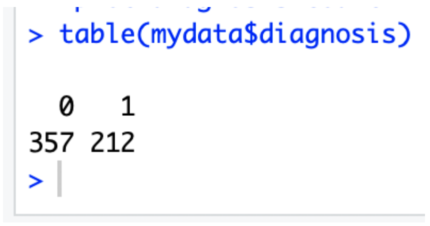
mydata <- mydata[, -c(6,7,14,15)]

After removing all these columns we still see that the correlation between compactness_se and fractal_dimension_se is above 80% and since we know that a correlation above 80% is to be removed, we still try to create a model using this, and if we find it of no use, we can remove it later.



For all the models we will be setting our training dataset with 60% of the original data and the validation dataset with the 40% of the original data. We do so in order make our model familiar with the patterns in the dataset with the 60% of the training data.

#plot diagnosis count
table(mydata$diagnosis)
barplot(table(mydata$diagnosis), main="Diagnosis",
        xlab="0 - Benign, 1 - Malign", density= 569)



```
> table(mydata$diagnosis)

  0   1 
357 212 
> 
```

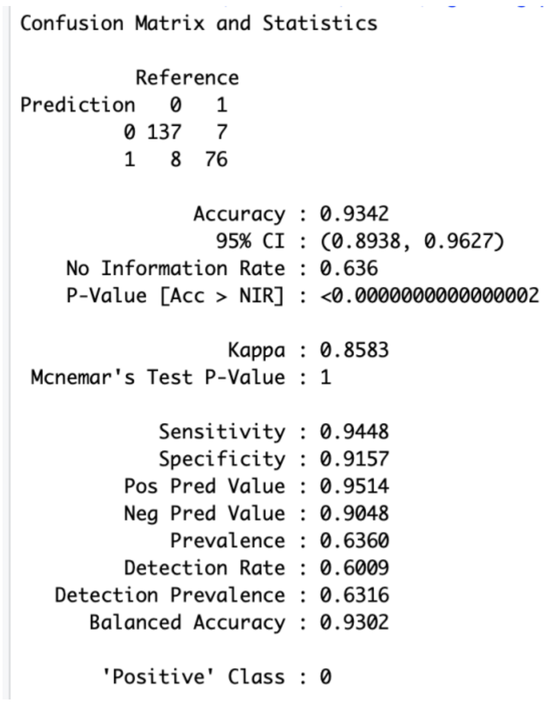
Models

Logistic Regression:

logit.reg <- glm(diagnosis ~ ., data = train, family = "binomial")

options(scipen=999) # remove scientific notation
summary(logit.reg)

logit.reg.pred <- predict(logit.reg, valid, type = "response")
library(caret)
confusionMatrix(as.factor(ifelse(logit.reg.pred > 0.5, 1, 0)), as.factor(valid$diagnosis))

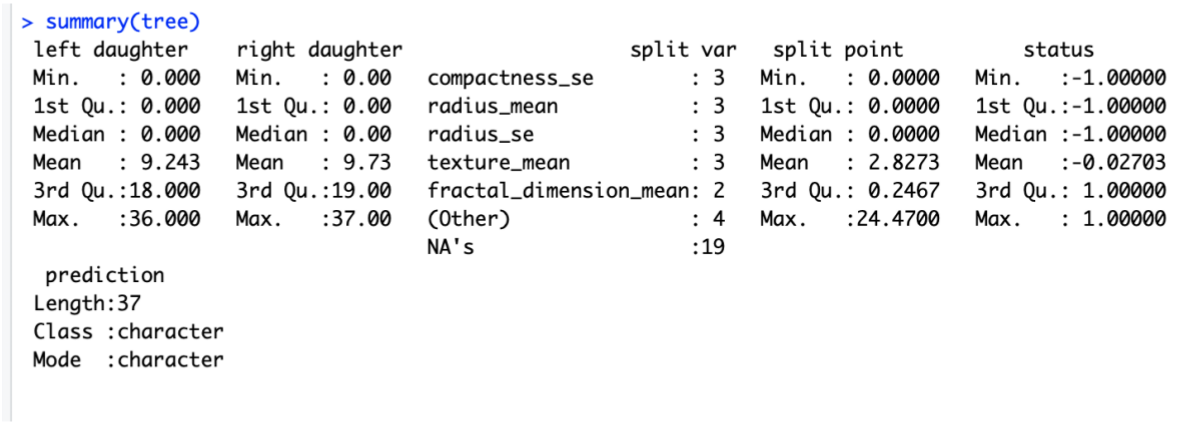


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 137   7
         1   8  76

               Accuracy : 0.9342
                 95% CI : (0.8938, 0.9627)
    No Information Rate : 0.636
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.8583
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9448
            Specificity : 0.9157
         Pos Pred Value : 0.9514
         Neg Pred Value : 0.9048
             Prevalence : 0.6360
         Detection Rate : 0.6009
   Detection Prevalence : 0.6316
      Balanced Accuracy : 0.9302

       'Positive' Class : 0
```

Random Forest:

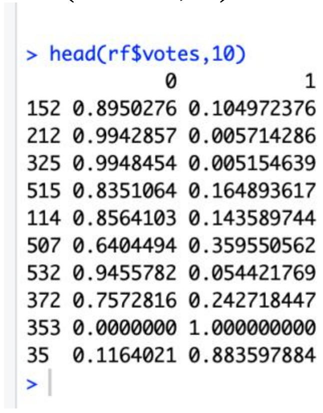
rf <- randomForest(as.factor(diagnosis) ~ ., data = train, ntree = 500,
 mtry = 4, nodesize = 5, importance = TRUE)
summary(rf)

summary(tree)



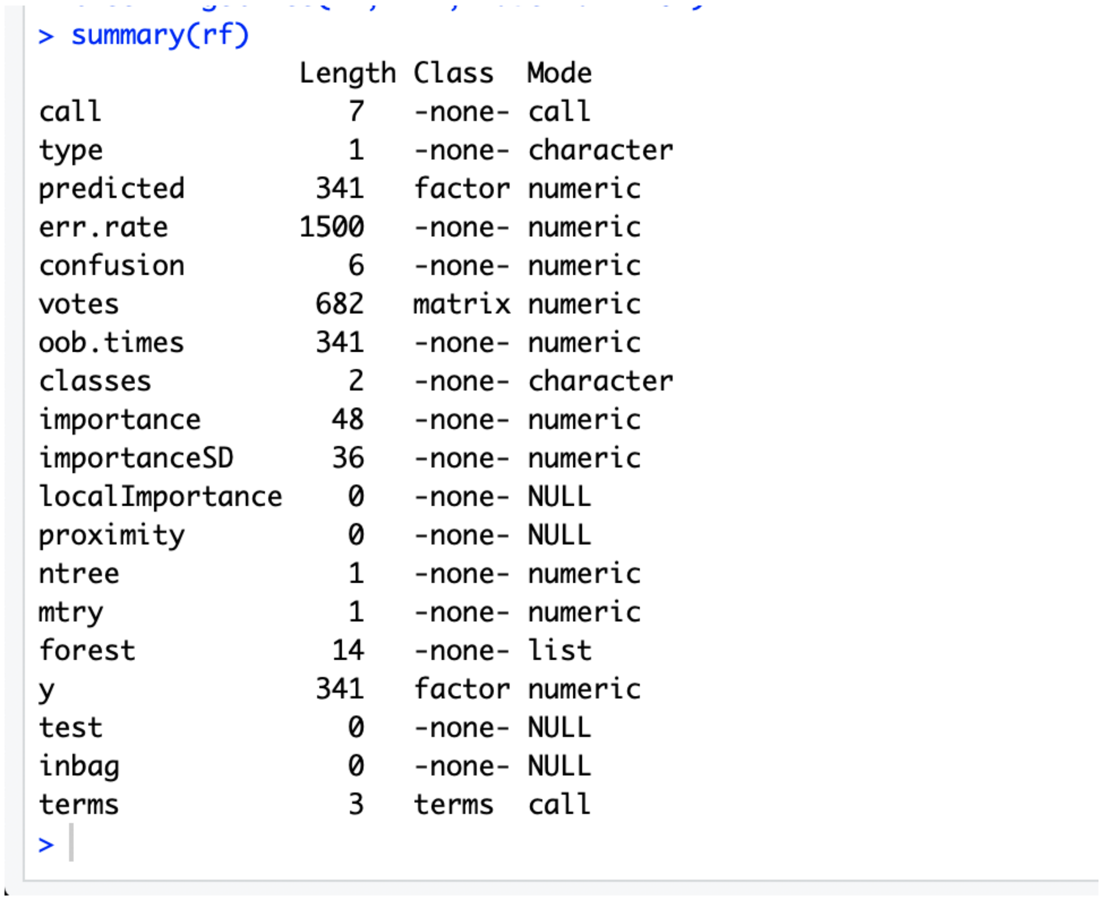
```
> summary(tree)
 left daughter    right daughter                      split var    split point          status
 Min.   : 0.000   Min.   : 0.00   compactness_se         : 3   Min.   : 0.0000   Min.   :-1.00000
 1st Qu.: 0.000   1st Qu.: 0.00   radius_mean            : 3   1st Qu.: 0.0000   1st Qu.:-1.00000
 Median : 0.000   Median : 0.00   radius_se              : 3   Median : 0.0000   Median :-1.00000
 Mean   : 9.243   Mean   : 9.73   texture_mean           : 3   Mean   : 2.8273   Mean   :-0.02703
 3rd Qu.:18.000   3rd Qu.:19.00   fractal_dimension_mean: 2   3rd Qu.: 0.2467   3rd Qu.: 1.00000
 Max.   :36.000   Max.   :37.00   (Other)                : 4   Max.   :24.4700   Max.   : 1.00000
                                  NA's                   :19
  prediction
 Length:37
 Class :character
 Mode  :character
```

head(rf$votes,10)

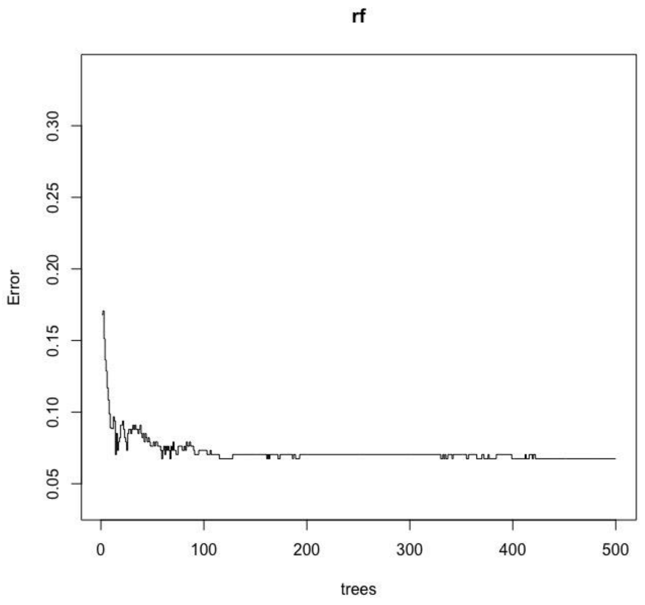


```
> head(rf$votes,10)
            0           1
152 0.8950276 0.104972376
212 0.9942857 0.005714286
325 0.9948454 0.005154639
515 0.8351064 0.164893617
114 0.8564103 0.143589744
507 0.6404494 0.359550562
532 0.9455782 0.054421769
372 0.7572816 0.242718447
353 0.0000000 1.000000000
35  0.1164021 0.883597884
>
```

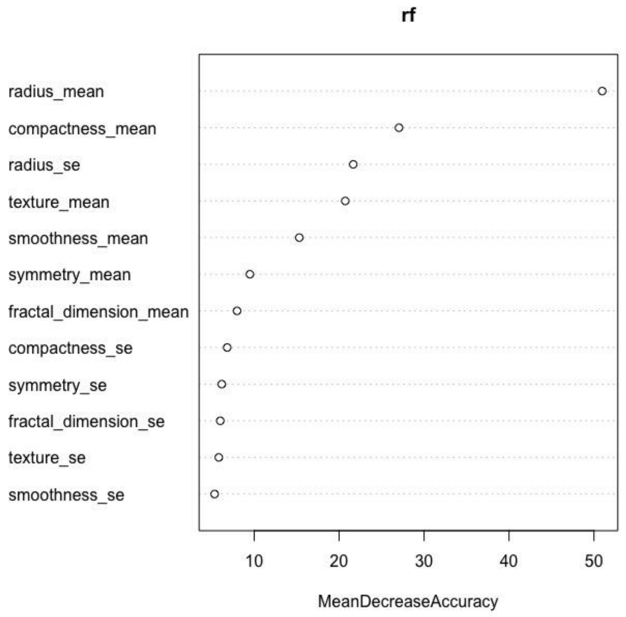
## Plot forest by prediction errors
plot(rf, type = "simple")



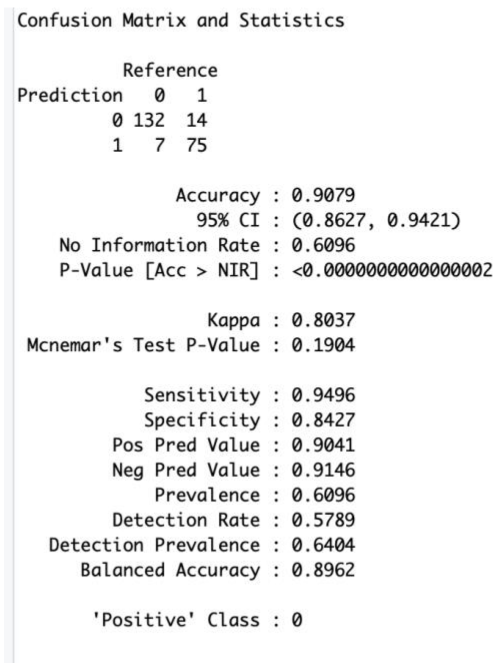
```
> summary(rf)
                Length Class  Mode
call               7   -none- call
type               1   -none- character
predicted        341   factor numeric
err.rate        1500   -none- numeric
confusion          6   -none- numeric
votes            682   matrix numeric
oob.times        341   -none- numeric
classes            2   -none- character
importance        48   -none- numeric
importanceSD      36   -none- numeric
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            14   -none- list
y                341   factor numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
>
```



```
## variable importance plot
varImpPlot(rf, type = 1)
```



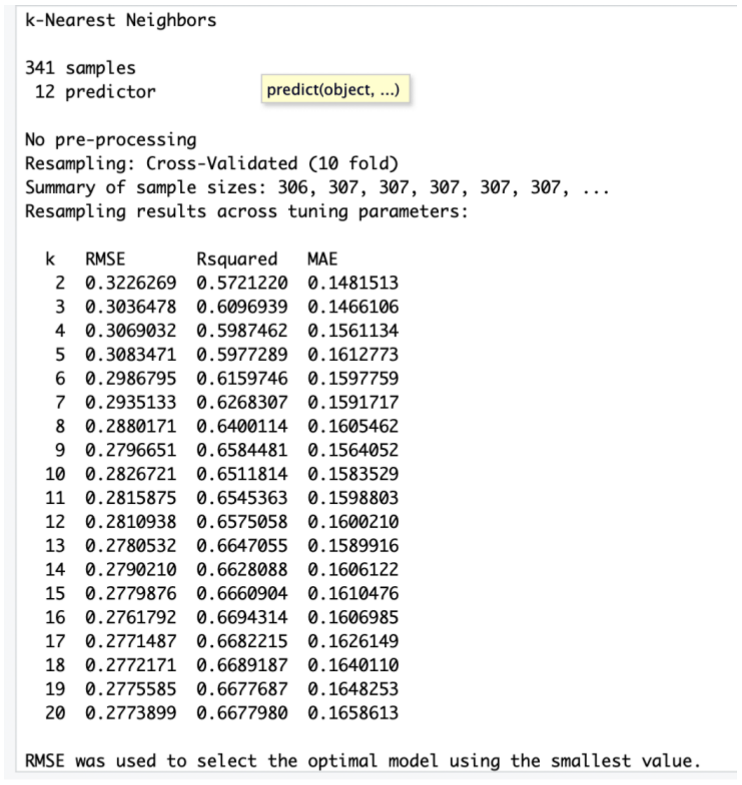
```
## confusion matrix
rf.pred <- predict(rf, valid)
library(caret)
confusionMatrix(rf.pred, as.factor(valid$diagnosis))
```



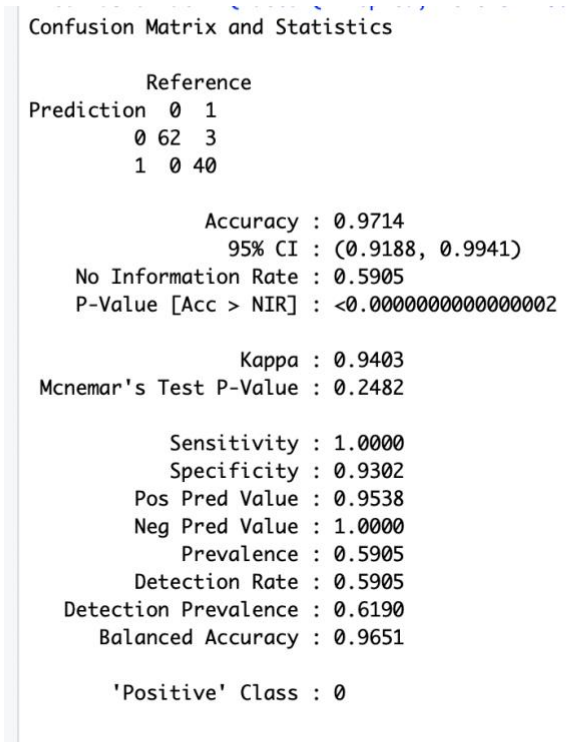
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 132  14
         1   7  75

               Accuracy : 0.9079
                 95% CI : (0.8627, 0.9421)
    No Information Rate : 0.6096
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.8037
 Mcnemar's Test P-Value : 0.1904

            Sensitivity : 0.9496
            Specificity : 0.8427
         Pos Pred Value : 0.9041
         Neg Pred Value : 0.9146
             Prevalence : 0.6096
         Detection Rate : 0.5789
   Detection Prevalence : 0.6404
      Balanced Accuracy : 0.8962

       'Positive' Class : 0
```

KNN:

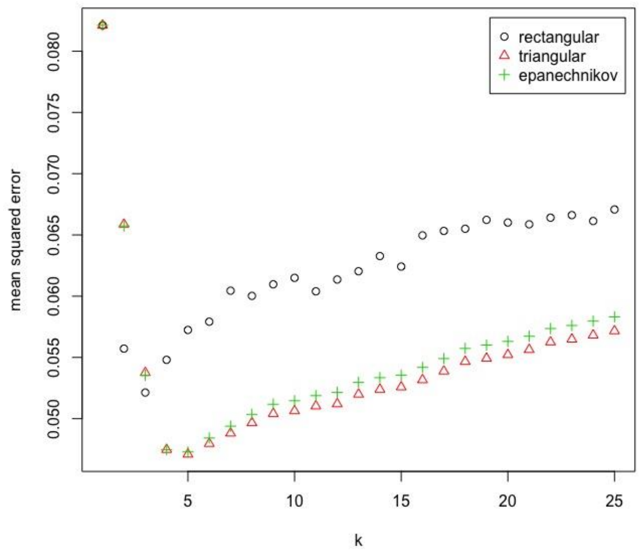
```
# Find optimal K

set.seed(502)
grid1 <- expand.grid(.k = seq(2, 20, by = 1))
control <- trainControl(method = "cv")
knn.train <- train(diagnosis ~ ., data = train,
                   method = "knn",
                   trControl = control,
                   tuneGrid = grid1)
knn.train


knn.pred <- predict(knn.train, newdata = valid)
confusionMatrix(factor(knn.pred, levels = 0:1),
factor(valid$diagnosis, levels = 0:1))
```



```
k-Nearest Neighbors

341 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 306, 307, 307, 307, 307, 307, ...
Resampling results across tuning parameters:

  k   RMSE       Rsquared   MAE
   2  0.3226269  0.5721220  0.1481513
   3  0.3036478  0.6096939  0.1466106
   4  0.3069032  0.5987462  0.1561134
   5  0.3083471  0.5977289  0.1612773
   6  0.2986795  0.6159746  0.1597759
   7  0.2935133  0.6268307  0.1591717
   8  0.2880171  0.6400114  0.1605462
   9  0.2796651  0.6584481  0.1564052
  10  0.2826721  0.6511814  0.1583529
  11  0.2815875  0.6545363  0.1598803
  12  0.2810938  0.6575058  0.1600210
  13  0.2780532  0.6647055  0.1589916
  14  0.2790210  0.6628088  0.1606122
  15  0.2779876  0.6660904  0.1610476
  16  0.2761792  0.6694314  0.1606985
  17  0.2771487  0.6682215  0.1626149
  18  0.2772171  0.6689187  0.1640110
  19  0.2775585  0.6677687  0.1648253
  20  0.2773899  0.6677980  0.1658613

RMSE was used to select the optimal model using the smallest value.
```

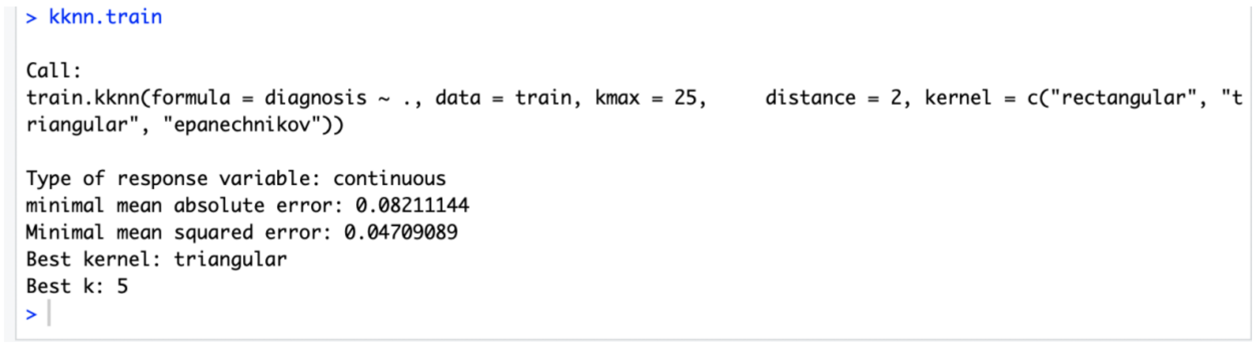


```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 62  3
         1  0 40

               Accuracy : 0.9714
                 95% CI : (0.9188, 0.9941)
    No Information Rate : 0.5905
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.9403
 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 1.0000
            Specificity : 0.9302
         Pos Pred Value : 0.9538
         Neg Pred Value : 1.0000
             Prevalence : 0.5905
         Detection Rate : 0.5905
   Detection Prevalence : 0.6190
      Balanced Accuracy : 0.9651

       'Positive' Class : 0
```



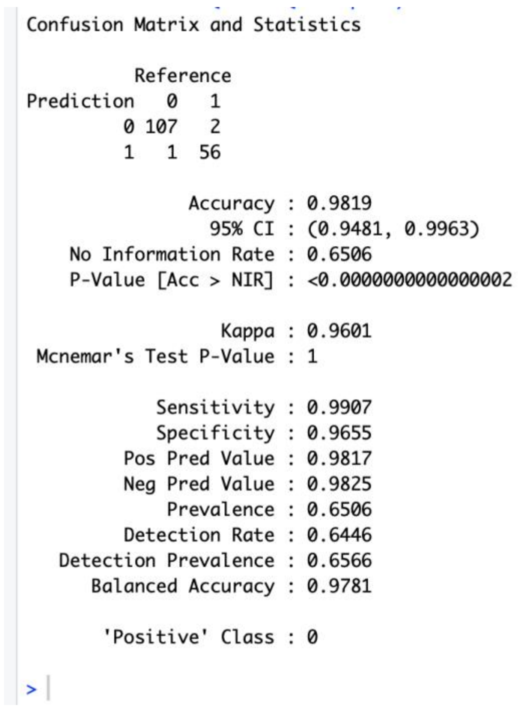
# Different distance weighting
#install.packages("kknn")
library(kknn)
set.seed(123)
kknn.train <- train.kknn(diagnosis ~ ., data = train, kmax = 25,
distance = 2,
kernel = c("rectangular", "triangular", "epanechnikov"))
plot(kknn.train)

kknn.train



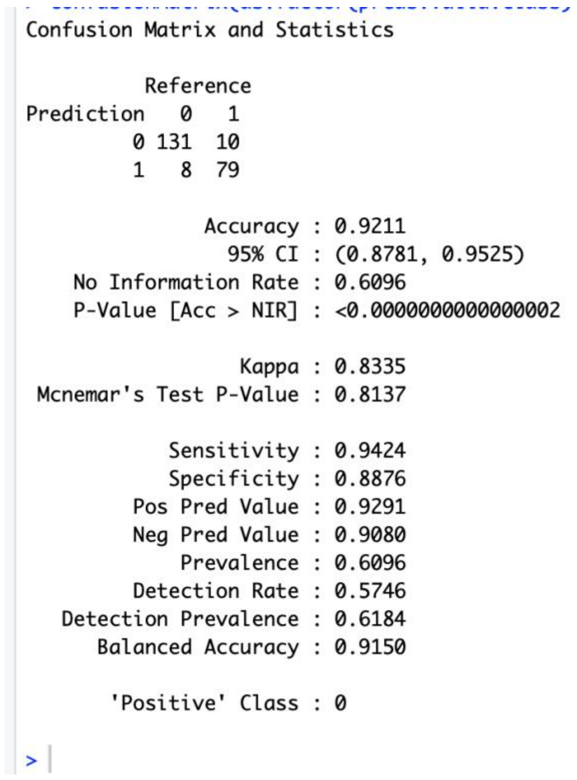
```
> kknn.train

Call:
train.kknn(formula = diagnosis ~ ., data = train, kmax = 25,     distance = 2, kernel = c("rectangular", "t
riangular", "epanechnikov"))

Type of response variable: continuous
minimal mean absolute error: 0.08211144
Minimal mean squared error: 0.04709089
Best kernel: triangular
Best k: 5
> 
```

kknn.pred <- predict(kknn.train, newdata = valid)
confusionMatrix(factor(kknn.pred, levels = 0:1), factor(valid$diagnosis, levels = 0:1))

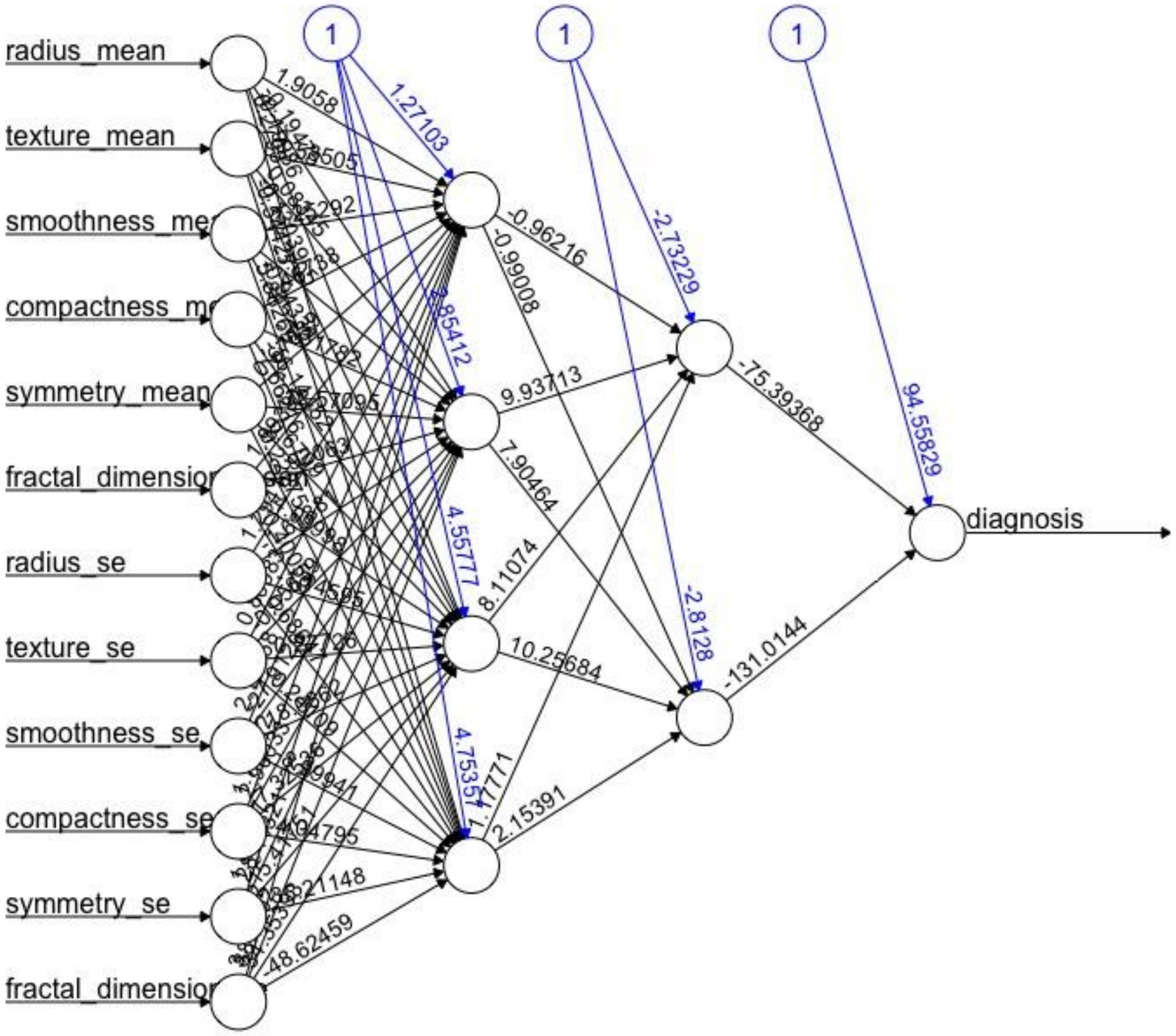

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 107   2
         1   1  56

               Accuracy : 0.9819
                 95% CI : (0.9481, 0.9963)
    No Information Rate : 0.6506
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.9601
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9907
            Specificity : 0.9655
         Pos Pred Value : 0.9817
         Neg Pred Value : 0.9825
             Prevalence : 0.6506
         Detection Rate : 0.6446
   Detection Prevalence : 0.6566
      Balanced Accuracy : 0.9781

       'Positive' Class : 0

>
```

Neural Network:

```
nn <- neuralnet(diagnosis ~ ., data = train, hidden = c(4,2), linear.output = FALSE)
plot(nn)

preds.valid <- compute(nn, valid[,-c(1)])
preds.valid.class <- ifelse(preds.valid$net.result>0.5,1,0)
confusionMatrix(as.factor(preds.valid.class),as.factor(valid$diagnosis))
```


```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  10
         1   8  79

               Accuracy : 0.9211
                 95% CI : (0.8781, 0.9525)
    No Information Rate : 0.6096
    P-Value [Acc > NIR] : <0.0000000000000002

                  Kappa : 0.8335
 Mcnemar's Test P-Value : 0.8137

            Sensitivity : 0.9424
            Specificity : 0.8876
         Pos Pred Value : 0.9291
         Neg Pred Value : 0.9080
             Prevalence : 0.6096
         Detection Rate : 0.5746
   Detection Prevalence : 0.6184
      Balanced Accuracy : 0.9150

       'Positive' Class : 0

>
```

Neural Net	92.11



Results and discussion:

Model	Accuracy %
Logistic Regression	93.42
Random Forest	90.79
KNN	98.19
Neural Network	92.11

It is evident from the table above that KNN model gave the best accuracy in our case. It has the accuracy of 98.19% with significantly lower false positive (1) and false negative (2) as compared to other models.

Note- Visualization and results are attached in the models above.

The output generated by the predictive models will be of great use for the doctors and researchers for finding the right prognosis for Breast Cancer Research.
The final output will be of valuable asset to the researchers as they will be able to predict in advance how soon a benign tumor can convert into a malignant one.
The researchers will be able to have a clear look at the parameters most responsible or the variable contributing most to the tumors for both malign and benign and will be able to make better predictions.

Summary

1) Practical implementation of the four models used.
2) Even though it is said that the deep learning techniques such as neural network provides the best results, it did not reflect the best result in our case. So, we can conclude that we cannot predict beforehand which model will best work for any dataset until and unless we have run all the different kind of models and evaluate their predicted accuracy.

References:

1. **https://www.kaggle.com/uciml/breast-cancer-wisconsin-data**
2. **https://www.kaggle.com/leemun1/predicting-breast-cancer-logistic-regression**
3. **https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/logistic-regression-analysis-r/tutorial/**
4. Class materials