TUMOR PREDICTION USING BREAST CANCER DATASET

MOHAMED FARHAN

DATASET DESCRIPTION

- **Name:** UCI_Breast_Cancer.csv
- **Number of samples:** 569
- **Number of attributes:** 33
- **Source:** https://www.kaggle.com/uciml/breast-cancer-wisconsin-data/
- **Target Variable: *diagnosis*** will be the target variable for my project. It will determine whether tumor is benign or malignant.

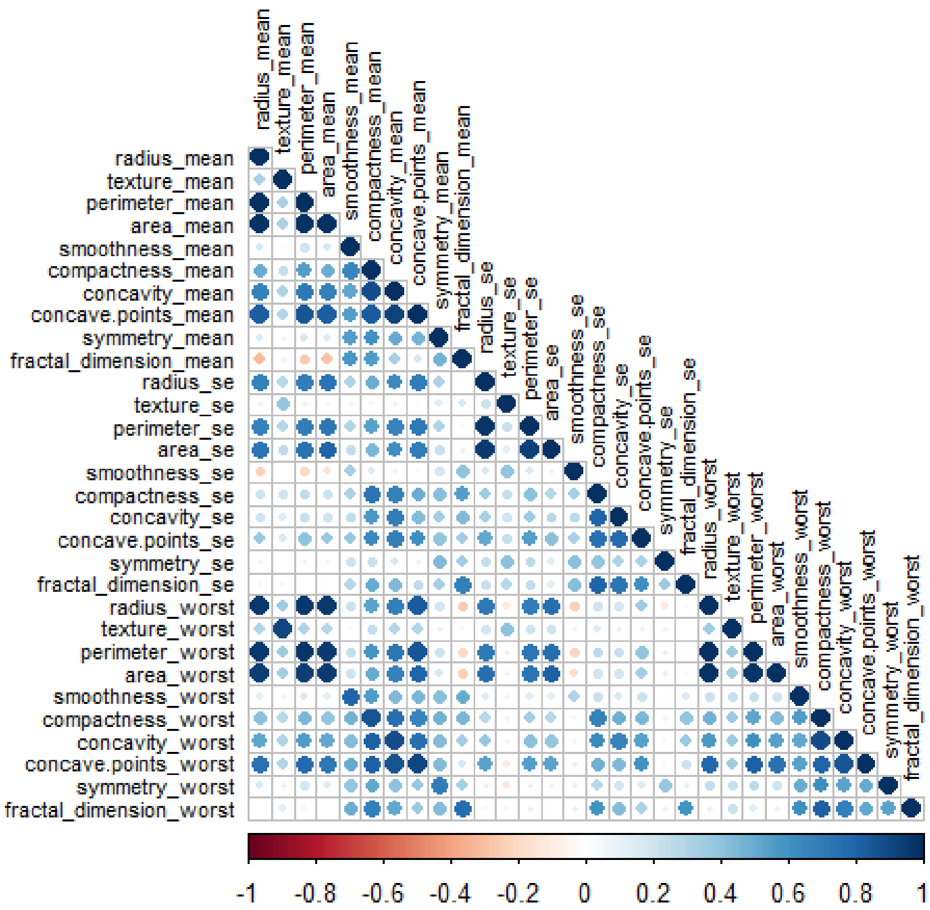
DATA PREPROCESSING

- We look for n/a, NaN and infinity values in the dataset which may produce skewed results. This is an integral step in preprocessing as it may influence the model's accuracy significantly.
- We find that predictor ***X*** contains missing values and therefore drop it.
- We also drop the ***id*** column as it is irrelevant.

```
> sapply(data,function(count) sum(is.na(count)))
                     id               diagnosis             radius_mean            texture_mean 
                      0                       0                       0                       0 
         perimeter_mean               area_mean         smoothness_mean        compactness_mean 
                      0                       0                       0                       0 
         concavity_mean     concave.points_mean           symmetry_mean  fractal_dimension_mean 
                      0                       0                       0                       0 
              radius_se              texture_se            perimeter_se                 area_se 
                      0                       0                       0                       0 
          smoothness_se          compactness_se            concavity_se       concave.points_se 
                      0                       0                       0                       0 
            symmetry_se    fractal_dimension_se            radius_worst           texture_worst 
                      0                       0                       0                       0 
        perimeter_worst              area_worst        smoothness_worst       compactness_worst 
                      0                       0                       0                       0 
        concavity_worst    concave.points_worst          symmetry_worst fractal_dimension_worst 
                      0                       0                       0                       0 
                      X 
                    569 
```

CORRELATION

- Correlation matrix can be used to analyze which variables in the dataset are strongly connected to each other.
- Correlation can be both positive or negative.
- If correlation between two variables is high, we can drop any one of the two variables to reduce collinearity among independent variables.
- We drop variables that have more than 75% correlation between them.



PRINCIPAL COMPONENT ANALYSIS (PCA)

- **PCA tells us how many principal components put together represent the most important or relevant information.**
- **From the above summary, we can say that the first 7 PCs covers enough relevant/important information to perform analysis or to fit a model.**

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

TRAINING AND TEST SETS

- The training set contains 60% of the entire dataset.
- The test set contains remaining 40% of the dataset.

SUPPORT VECTOR MACHINE (SVM) MODEL

- SVM model implemented using ***tune()*** from ***e1071 package.***
- We use the first 7 principal components from our PCA to fit the model.
- Gamma values used to tune model: **2^-1 to 2^1**
- Cost values used to tune model: **(0.01, 0.1, 0.5, 1, 5)**

```
> summary(svm_mod)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
   0.5    5

- best performance: 0.0325641

- Detailed performance results:
   gamma cost      error dispersion
1    0.5 0.01 0.07782051 0.05737424
2    1.0 0.01 0.07782051 0.05737424
3    2.0 0.01 0.07782051 0.05737424
4    0.5 0.10 0.04012821 0.03578814
5    1.0 0.10 0.04012821 0.03578814
6    2.0 0.10 0.04012821 0.03578814
7    0.5 0.50 0.03256410 0.02646435
8    1.0 0.50 0.03256410 0.02646435
9    2.0 0.50 0.03256410 0.02646435
10   0.5 1.00 0.03756410 0.03382446
11   1.0 1.00 0.03756410 0.03382446
12   2.0 1.00 0.03756410 0.03382446
13   0.5 5.00 0.03256410 0.02646435
14   1.0 5.00 0.03256410 0.02646435
15   2.0 5.00 0.03256410 0.02646435
```

RESULTS OBTAINED USING SVM MODEL

- After running the model on the test set, the following statistics are obtained:
- Accuracy: 97.06%
- Error rate: 2.94%

```
> best_mod

Call:
best.tune(method = svm, train.x = out ~ ., data = traindata, ranges = list(gamma = 2^(-1:1),
    cost = c(0.01, 0.1, 0.5, 1, 5)), kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  5
      gamma:  0.5

Number of Support Vectors:  35
```

```
> conf_mat1
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 106   4
         M   1  59

               Accuracy : 0.9706
                 95% CI : (0.9327, 0.9904)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9363
 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9365
            Specificity : 0.9907
         Pos Pred Value : 0.9833
         Neg Pred Value : 0.9636
             Prevalence : 0.3706
         Detection Rate : 0.3471
   Detection Prevalence : 0.3529
      Balanced Accuracy : 0.9636

       'Positive' Class : M
```

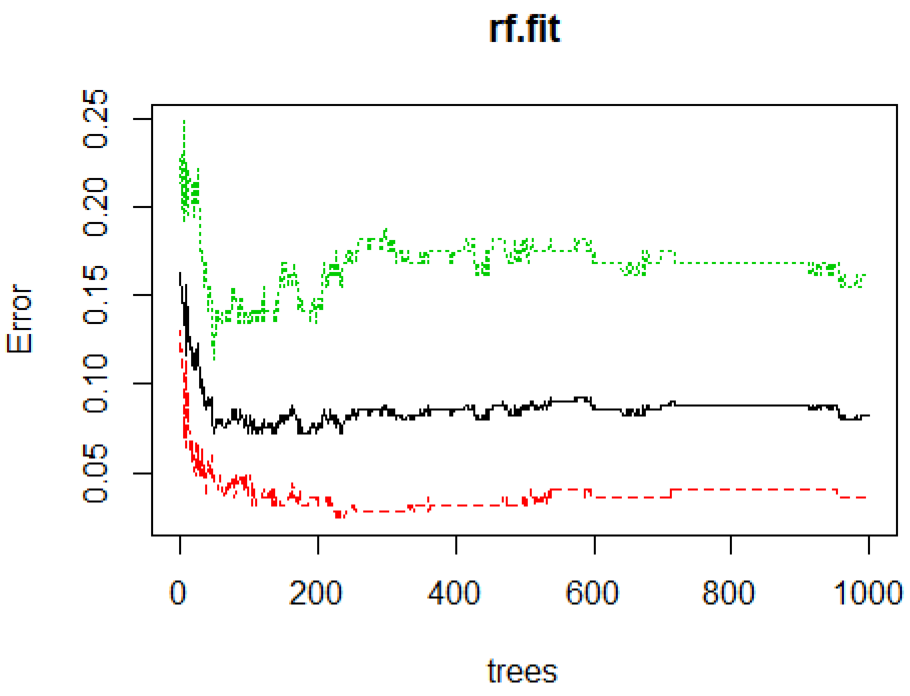
RANDOM FOREST MODEL

- Random Forest is a powerful model based off decision trees.
- We use the first 7 principal components from our to fit the model.
- We obtain Out-of-Bag (OOB) error as 8.27%

```
> rf.fit

Call:
 randomForest(formula = train$`data$dummy.outcome` ~ ., data = train,      importance = TRUE, ntree = 1000)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 2

        OOB estimate of  error rate: 8.27%
Confusion matrix:
    0   1 class.error
0 241   9   0.0360000
1  24 125   0.1610738
```



RESULTS OBTAINED USING RANDOM FOREST MODEL

- After running the model on the test set, the following statistics are obtained:
- Accuracy: 94.71%
- Error rate: 5.29%
- Larger the Mean Decrease Gini, more important the variable is for the model.

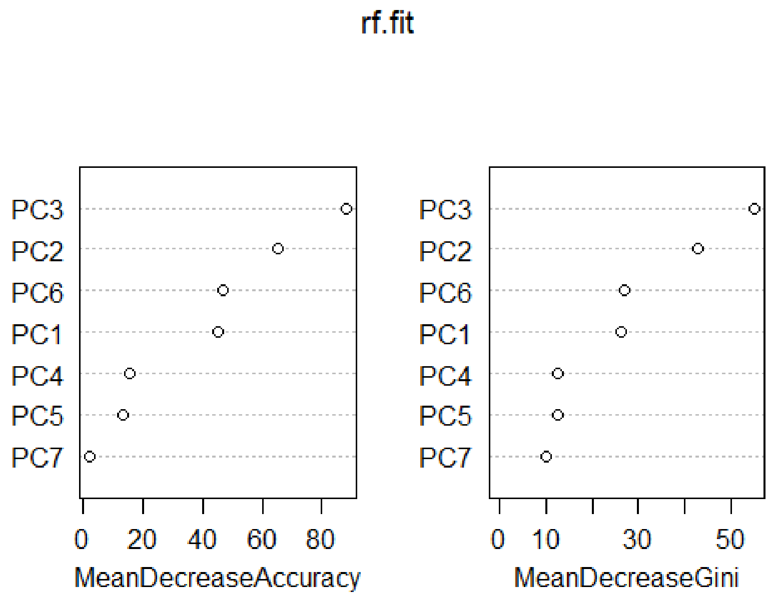
```
> confusionMatrix(rf_pred, test$`data$dummy.outcome` )
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 105   7
         1   2  56

               Accuracy : 0.9471
                 95% CI : (0.9019, 0.9755)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8846
 Mcnemar's Test P-Value : 0.1824

            Sensitivity : 0.9813
            Specificity : 0.8889
```



k-NEAREST NEIGHBORS (kNN) MODEL

- We use the first 7 principal components from our PCA to fit the model.
- The data is centered and scaled before fitting.

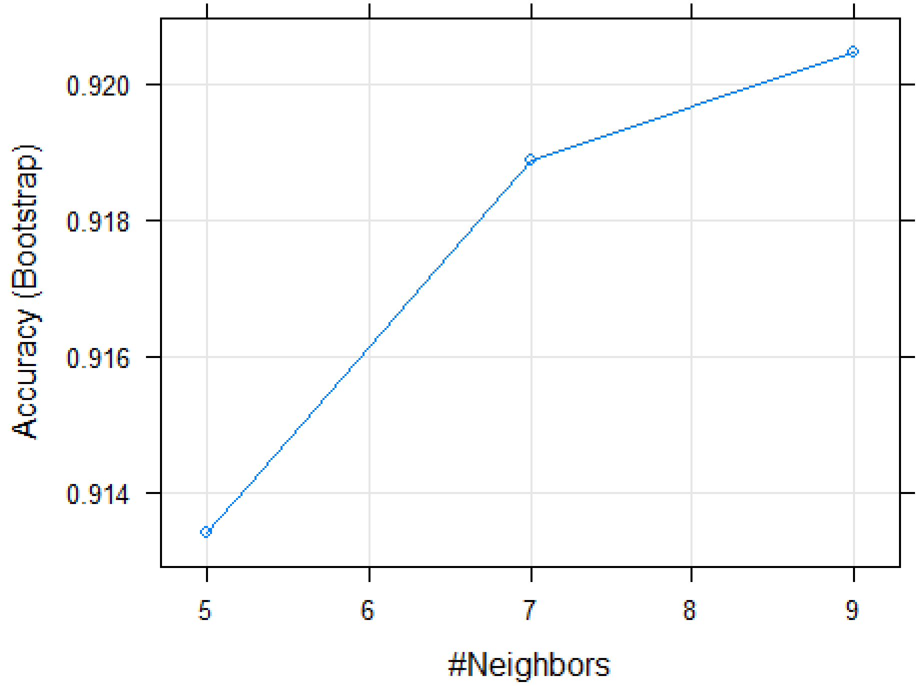
```
> knnFit_pca
k-Nearest Neighbors

399 samples
  7 predictor
  2 classes: 'B', 'M'

Pre-processing: centered (7), scaled (7)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 399, 399, 399, 399, 399, 399, ...
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.9134036  0.8107618
  7  0.9188868  0.8221248
  9  0.9204623  0.8247662

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 9.
```



RESULTS OBTAINED USING kNN MODEL

- After running the model on the test set, the following statistics are obtained:
- Accuracy: 92.35%
- Error rate: 7.65%
- Optimal k: 9

```
> confusionMatrix(knnPredict_pca, testdata$out )
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 107  13
         M   0  50

               Accuracy : 0.9235
                 95% CI : (0.8728, 0.9587)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8288
 Mcnemar's Test P-Value : 0.0008741

            Sensitivity : 1.0000
            Specificity : 0.7937
```

LOGISTIC REGRESSION MODEL

- Logistic regression model uses logistic function and also takes into consideration the p-value. Variables with large p-values can be omitted.
- We use the first 7 principal components from our PCA to fit the model.
- Here we combine both forward and backward method for variable selection.

```
> summary(model_log.fit3)

Call:
glm(formula = out ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6, family = "binomial",
    data = traindata)

Deviance Residuals:
    Min       1Q    Median       3Q      Max
-1.6826  -0.0267  -0.0019   0.0001   3.8120

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.5663     0.3950  -1.434  0.15169
PC1          -3.5788     0.7374  -4.853 1.22e-06 ***
PC2           1.8844     0.4398   4.285 1.83e-05 ***
PC3          -0.6284     0.2654  -2.368  0.01787 *
PC4          -0.7583     0.2710  -2.798  0.00514 **
PC5           1.7883     0.5730   3.121  0.00180 **
PC6          -0.6722     0.3316  -2.027  0.04267 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 527.285  on 398  degrees of freedom
Residual deviance:  47.772  on 392  degrees of freedom
AIC: 61.772
```

RESULTS OBTAINED USING LOGISTIC REGRESSION MODEL

- After running the model on the test set, the following statistics are obtained:
- Accuracy: 97.06%
- Error rate: 2.94%

```
> confusionMatrix(ifelse(p_3>0.5,'M','B'),testdata$out)
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 105   3
         M   2  60

               Accuracy : 0.9706
                 95% CI : (0.9327, 0.9904)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9367
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9813
            Specificity : 0.9524
```

CONCLUSION

- Three different models were used to measure performance
- SVM model
- Random forest model
- kNN model
- Logistic regression model

We observed that both SVM and Logistic regression models deliver similar performances **(Accuracy= 97.06%)** when compared with kNN and Random forest models.

THANK YOU