Laboratory Practice I

Data Analytics – Mini Project

Shiva Saran
BE-B 20
2020-2021

Download Breast Cancer Wisconsin (Diagnostic) dataset. Using atleast 2 classification algorithm,

- Load the data from CSV file and split it into training and test datasets.
- Summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
- *Classify samples from a test dataset and a summarized training dataset.*
- *Compare the Confusion Matrix of the respective classification algorithms and find out which is the best algorithm.*

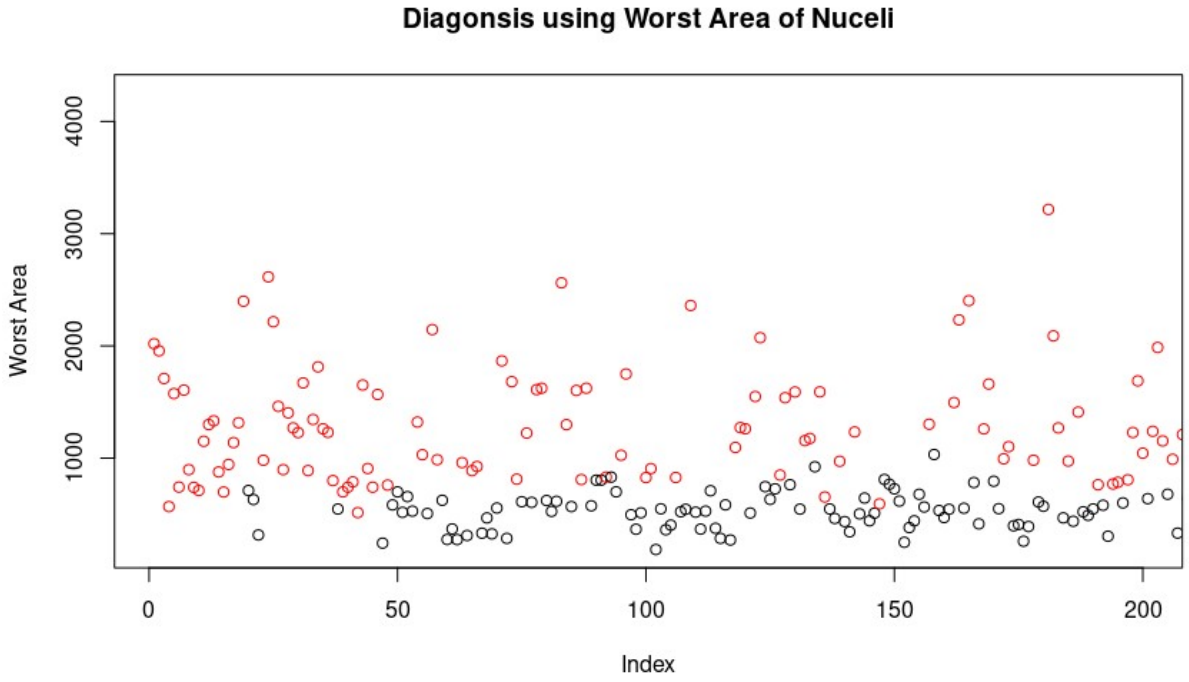
```
Dataset Link - https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

#Brief Overview of the Data set
Predicting Results
- predicting field 2, diagnosis: B = benign, M = malignant
- sets are linearly separable using all 30 input features
- best predictive accuracy obtained using one separating plane in the 3-D space of Worst Area, Worst
Smoothness and Mean Texture.  Estimated accuracy 97.5% using repeated 10-fold crossvalidations.
Classifier has correctly diagnosed 176 consecutive new patients as of November 1995.


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
        e) smoothness (local variation in radius lengths)
        f) compactness (perimeter^2 / area - 1.0)
        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)

Several of the papers listed above contain detailed descriptions of
how these features are computed.

The mean, standard error, and "worst" or largest (mean of the three
largest values) of these features were computed for each image,
resulting in 30 features.  For instance, field 3 is Mean Radius, field
13 is Radius SE, field 23 is Worst Radius.

All feature values are re-coded with four significant digits.
Class distribution: 357 benign, 212 malignant
```

```
> #Installing the libaries
>
> install.packages('e1071')
> install.packages('caTools')
>
>
> #Checking that the libraries are successfully installed
> library(caTools)
> library(e1071)
>
> #Importing The Dataset
> mydata <- read.csv("/home/shiva/Documents/BE/LP1/Cancer_Dataset/data.csv")
>
> #Checking the Dataset
> View(mydata)
>
> #Dimensions of the DataSet
> dim(mydata)
[1] 569  33
> names(mydata)
 [1] "id"                      "diagnosis"               "radius_mean"             "texture_mean"
"perimeter_mean"
 [6] "area_mean"               "smoothness_mean"         "compactness_mean"        "concavity_mean"
"concave.points_mean"
[11] "symmetry_mean"           "fractal_dimension_mean"  "radius_se"               "texture_se"
"perimeter_se"
[16] "area_se"                 "smoothness_se"           "compactness_se"          "concavity_se"
"concave.points_se"
[21] "symmetry_se"             "fractal_dimension_se"    "radius_worst"            "texture_worst"
"perimeter_worst"
[26] "area_worst"              "smoothness_worst"        "compactness_worst"       "concavity_worst"
"concave.points_worst"
[31] "symmetry_worst"          "fractal_dimension_worst" "X"
>
>
> #internal structure
> names(mydata)
 [1] "id"                      "diagnosis"               "radius_mean"             "texture_mean"
"perimeter_mean"
 [6] "area_mean"               "smoothness_mean"         "compactness_mean"        "concavity_mean"
"concave.points_mean"
[11] "symmetry_mean"           "fractal_dimension_mean"  "radius_se"               "texture_se"
"perimeter_se"
[16] "area_se"                 "smoothness_se"           "compactness_se"          "concavity_se"
"concave.points_se"
[21] "symmetry_se"             "fractal_dimension_se"    "radius_worst"            "texture_worst"
"perimeter_worst"
[26] "area_worst"              "smoothness_worst"        "compactness_worst"       "concavity_worst"
"concave.points_worst"
[31] "symmetry_worst"          "fractal_dimension_worst" "X"
```

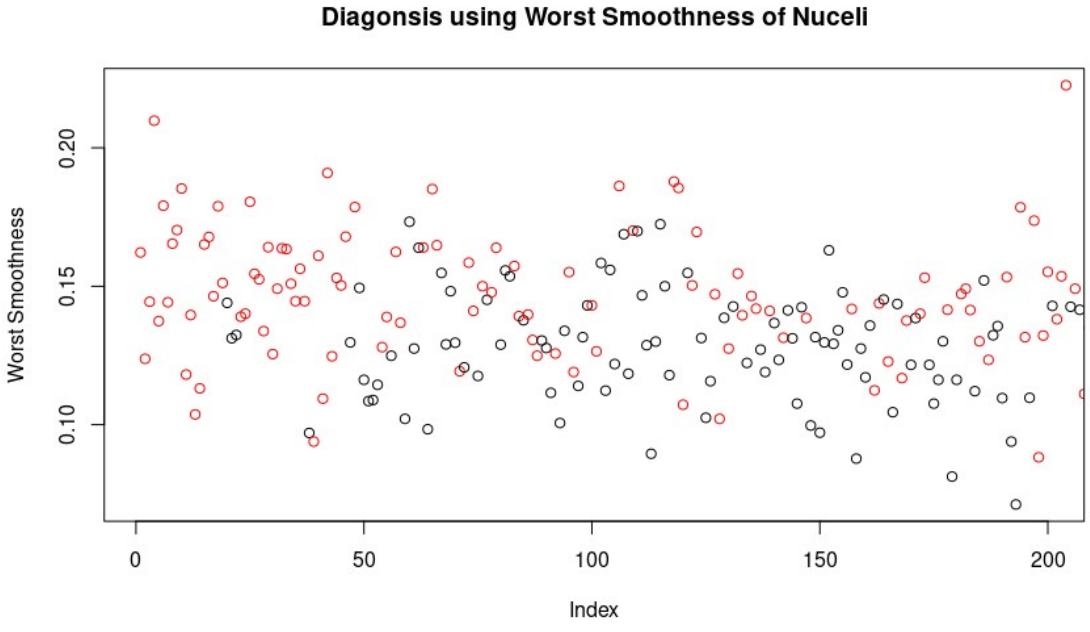
```
> #Statistics of Major attributes useful for predicting the diagonsis.
> #Min Values
> min(mydata$area_worst)
[1] 185.2
> min(mydata$smoothness_worst)
[1] 0.07117
> min(mydata$texture_mean)
[1] 9.71

>
>
> #Max Values
> max(mydata$area_worst)
[1] 4254
> max(mydata$smoothness_worst)
[1] 0.2226
> max(mydata$texture_mean)
[1] 39.28
>
>
> #Range
> range(mydata$area_worst)
[1]  185.2 4254.0
> range(mydata$smoothness_worst)
[1] 0.07117 0.22260
> range(mydata$texture_mean)
[1]  9.71 39.28
>
>
> #Standard Deviation
> sd(mydata$area_worst)
[1] 569.357
> sd(mydata$smoothness_worst)
[1] 0.02283243
> sd(mydata$texture_mean)
[1] 4.301036
>
>
> #Variance
> var(mydata$area_worst)
[1] 324167.4
> var(mydata$smoothness_worst)
[1] 0.0005213198
> var(mydata$texture_mean)
[1] 18.49891
>
>
> #Percentile
> quantile(mydata$area_worst)
     0%    25%    50%    75%   100%
 185.2  515.3  686.5 1084.0 4254.0
> quantile(mydata$smoothness_worst)
      0%     25%     50%     75%    100%
0.07117 0.11660 0.13130 0.14600 0.22260
> quantile(mydata$texture_mean)
```

```
    0%    25%    50%    75%   100%
 9.71  16.17  18.84  21.80  39.28
```

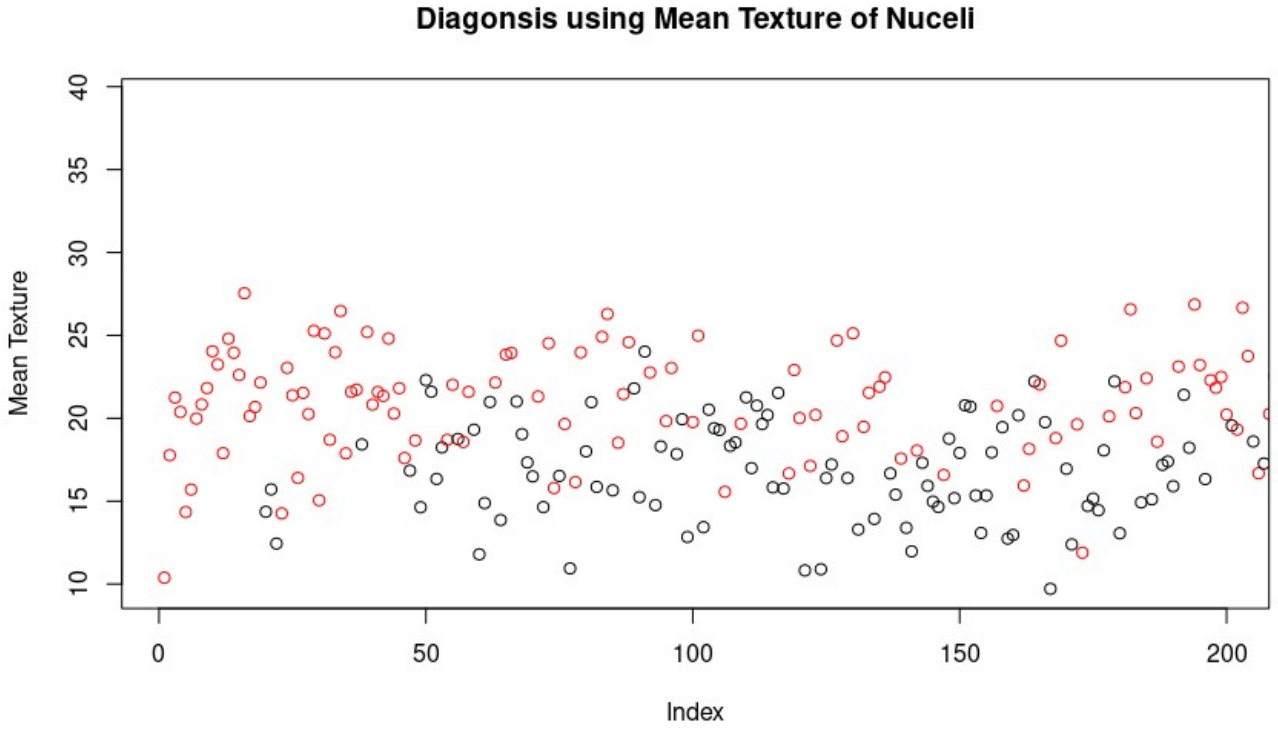
```
> #Data Visualisation
> plot(mydata$area_worst, main = "Diagonsis using Worst Area of Nuceli", ylab = "Worst Area",
col=mydata$diagnosis, xlim = c(1,200))
```



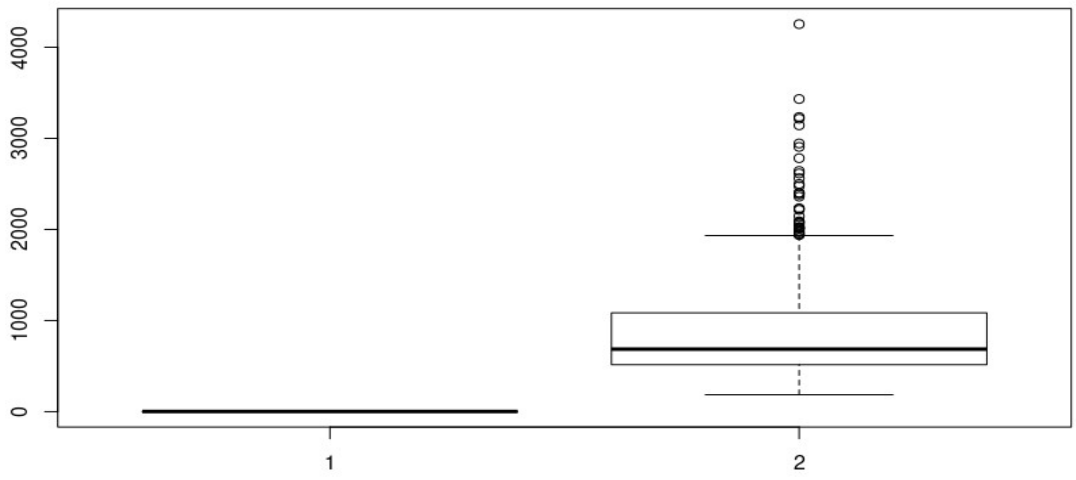

```
> #Using Worst Smoothness
> plot(mydata$smoothness_worst, main = "Diagonsis using Worst Smoothness of Nuceli", ylab = "Worst
Smoothness", col=mydata$diagnosis, xlim = c(1,200))
```



```
> #Using Mean Texture
> plot(mydata$texture_mean, main = "Diagonsis using Mean Texture of Nuceli", ylab = "Mean Texture",
col=mydata$diagnosis, xlim = c(1,200))
```




```
> #Boxplot for Diagnosis using Worst Area
> boxplot(c(mydata$diagnosis), mydata$area_worst)
```



```
> #Using Classification Algorithm to Predict Diagonsis
> #Splitting the Data into Training and Testing Dataset
> temp_field<-sample.split(mydata$diagnosis,SplitRatio=0.6)
> train<-subset(mydata, temp_field==TRUE)
> test<-subset(mydata, temp_field == FALSE)
>
>
> #Using Naive Bayes Algorithm
> my_model<-naiveBayes(as.factor(train$diagnosis)~.,train)
> pred1<-predict(my_model,test[,-2])

> #Creating Confusion Matrix
> ConFusNavB <- table(pred1, test$diagnosis, dnn=c("predicted", "Actual"))
> ConFusNavB
           Actual
predicted   B   M
          B 142  10
          M   1  75


#Combining the Test Data and Predicted Data
output<-cbind(test, pred1)
View(output)



> #Using SVM Algorithm
> split = sample.split(mydata$diagnosis, SplitRatio = 0.60)
> training_set = subset(mydata, split == TRUE)
> test_set = subset(mydata, split == FALSE)
> classifier = svm(formula = diagnosis ~ .,data = training_set,type = 'C-classification',kernel =
'linear')
> y_pred = predict(classifier, newdata = test_set[-2])
> ConFusSVM = table(test_set[, 2], y_pred, dnn=c("predicted", "Actual"))
> ConFusSVM
           Actual
predicted   B   M
          B 140   3
          M   2  83

> #Comparing Confusion Matrices
> #Naives Bayes
> #Correct predicitions Using Naive Bayes
> ConFusNavB[1]+ConFusNavB[4]
[1] 217
>
> #Incorrect predicitions Using Naive Bayes
> ConFusNavB[2]+ConFusNavB[3]
[1] 11
>
> #Correct predicitions % Using Naive Bayes
> CPNB = ((ConFusNavB[1]+ConFusNavB[4])/(ConFusNavB[1]+ConFusNavB[2]+ConFusNavB[3]+ConFusNavB[4]))*100
> CPNB
[1] 95.17544
```

```
>
> #Incorrect predicitions % Using Naive Bayes
> IPNB = ((ConFusNavB[2]+ConFusNavB[3])/(ConFusNavB[1]+ConFusNavB[2]+ConFusNavB[3]+ConFusNavB[4]))*100
> IPNB
[1] 4.824561
>
> #SVM
> #Correct predicitions Using SVM
> ConFusSVM[1]+ConFusSVM[4]
[1] 223
>
> #Incorrect predicitions Using SVM
> ConFusSVM[2]+ConFusSVM[3]
[1] 5
>
> #Correct predicitions % Using SVM
> CPSVM = ((ConFusSVM[1]+ConFusSVM[4])/(ConFusSVM[1]+ConFusSVM[2]+ConFusSVM[3]+ConFusSVM[4]))*100
> CPSVM
[1] 97.80702
>
> #Incorrect predicitions % Using SVM
> IPSVM = ((ConFusSVM[2]+ConFusSVM[3])/(ConFusSVM[1]+ConFusSVM[2]+ConFusSVM[3]+ConFusSVM[4]))*100
> IPSVM
[1] 2.192982
```