Breast Cancer Detection Using Classification Models

Sumanth Donthula(A20519856) and Ram Vaka (A20481446)

2022-12-01

Project Description:

The project is a classifier problem. The Data set contains the different dependent features to predict the Breast Cancer whether it is Benign or Malignant. The Data set is taken from the study done by University of California at Irvine from there ML Data Set Repository.

Reading the Data File

```
Data=read.table("data.csv", header = TRUE, sep = ",")
```

Checking if any null records are present in Data Set

```
sum(is.na(Data))
```

```
## [1] 0
```

The dataset is almost equally distributed for both Malignant and Benign cases

```
table(Data$diagnosis)
```

```
##
##   B   M
## 357 212
```

Displaying the Data Set

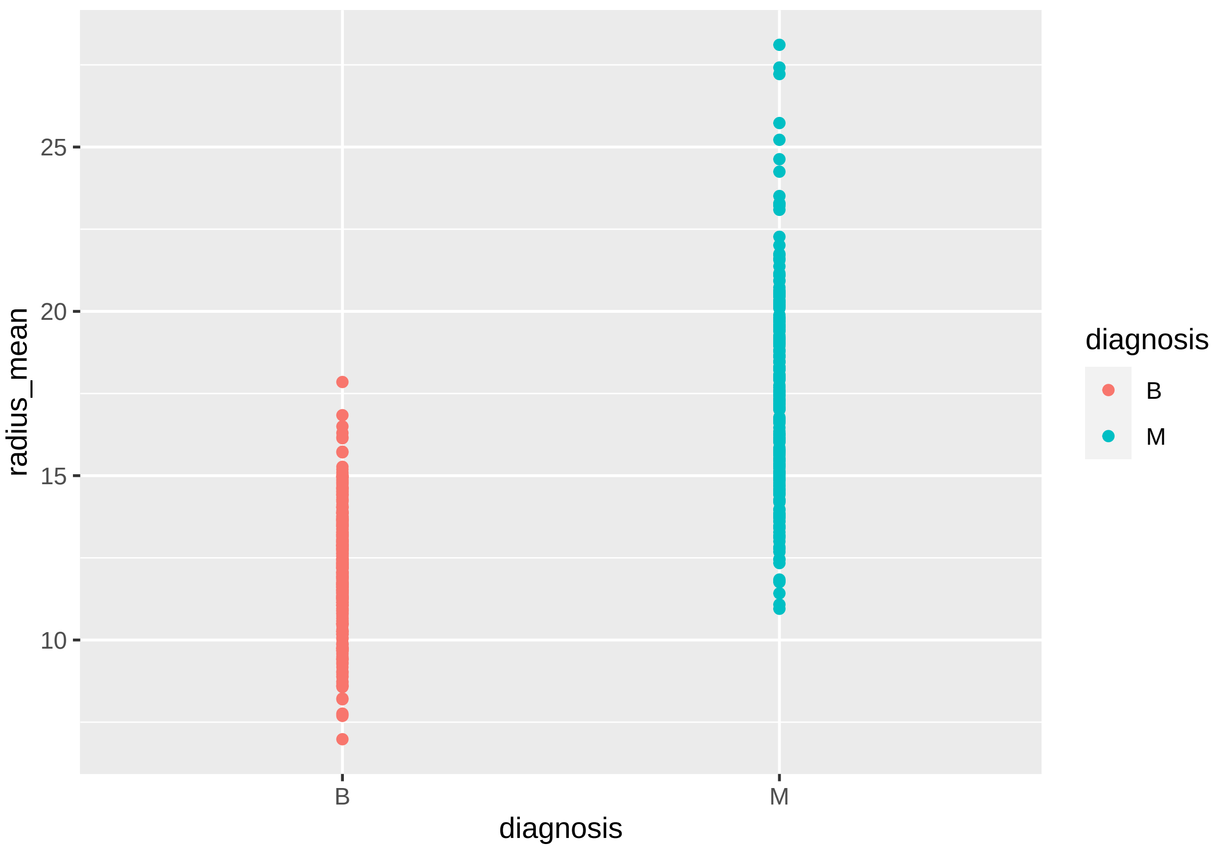
```
head(Data)
```

```
##         id diagnosis radius_mean texture_mean perimeter_mean area_mean
## 1   842302         M       17.99        10.38         122.80    1001.0
## 2   842517         M       20.57        17.77         132.90    1326.0
## 3 84300903         M       19.69        21.25         130.00    1203.0
## 4 84348301         M       11.42        20.38          77.58     386.1
## 5 84358402         M       20.29        14.34         135.10    1297.0
## 6   843786         M       12.45        15.70          82.57     477.1
##   smoothness_mean compactness_mean concavity_mean concave.points_mean
## 1         0.11840          0.27760         0.3001             0.14710
## 2         0.08474          0.07864         0.0869             0.07017
## 3         0.10960          0.15990         0.1974             0.12790
## 4         0.14250          0.28390         0.2414             0.10520
```

```
## 5         0.10030          0.13280         0.1980            0.10430
## 6         0.12780          0.17000         0.1578            0.08089
##   symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
## 1        0.2419                0.07871    1.0950     0.9053        8.589
## 2        0.1812                0.05667    0.5435     0.7339        3.398
## 3        0.2069                0.05999    0.7456     0.7869        4.585
## 4        0.2597                0.09744    0.4956     1.1560        3.445
## 5        0.1809                0.05883    0.7572     0.7813        5.438
## 6        0.2087                0.07613    0.3345     0.8902        2.217
##   area_se smoothness_se compactness_se concavity_se concave.points_se
## 1  153.40      0.006399        0.04904      0.05373           0.01587
## 2   74.08      0.005225        0.01308      0.01860           0.01340
## 3   94.03      0.006150        0.04006      0.03832           0.02058
## 4   27.23      0.009110        0.07458      0.05661           0.01867
## 5   94.44      0.011490        0.02461      0.05688           0.01885
## 6   27.19      0.007510        0.03345      0.03672           0.01137
##   symmetry_se fractal_dimension_se radius_worst texture_worst perimeter_worst
## 1     0.03003             0.006193        25.38         17.33          184.60
## 2     0.01389             0.003532        24.99         23.41          158.80
## 3     0.02250             0.004571        23.57         25.53          152.50
## 4     0.05963             0.009208        14.91         26.50           98.87
## 5     0.01756             0.005115        22.54         16.67          152.20
## 6     0.02165             0.005082        15.47         23.75          103.40
##   area_worst smoothness_worst compactness_worst concavity_worst
## 1     2019.0           0.1622            0.6656          0.7119
## 2     1956.0           0.1238            0.1866          0.2416
## 3     1709.0           0.1444            0.4245          0.4504
## 4      567.7           0.2098            0.8663          0.6869
## 5     1575.0           0.1374            0.2050          0.4000
## 6      741.6           0.1791            0.5249          0.5355
##   concave.points_worst symmetry_worst fractal_dimension_worst
## 1               0.2654         0.4601                 0.11890
## 2               0.1860         0.2750                 0.08902
## 3               0.2430         0.3613                 0.08758
## 4               0.2575         0.6638                 0.17300
## 5               0.1625         0.2364                 0.07678
## 6               0.1741         0.3985                 0.12440
```

Displaying the classifier data with one of the feature. The graph displays radius_mean, a feature from our data set to visualize the classifier problem.

```r
library(ggplot2)

# Scatter plot by group
ggplot(Data, aes(x = diagnosis, y = radius_mean, color = diagnosis)) +
geom_point()
```



We can see tha there is a colummn called id in our dataset which we dont require for trainig our model. Droppinf the column id. From the data we can see that the records of dependent variable contains mainly mean, standard error and worst features.

Exploratory Data Analysis :

```
summary(Data)
```

```
##        id             diagnosis          radius_mean      texture_mean
##  Min.   :     8670   Length:569         Min.   : 6.981   Min.   : 9.71
##  1st Qu.:   869218   Class :character   1st Qu.:11.700   1st Qu.:16.17
##  Median :   906024   Mode  :character   Median :13.370   Median :18.84
##  Mean   : 30371831                      Mean   :14.127   Mean   :19.29
##  3rd Qu.:  8813129                      3rd Qu.:15.780   3rd Qu.:21.80
##  Max.   :911320502                      Max.   :28.110   Max.   :39.28
##  perimeter_mean     area_mean      smoothness_mean   compactness_mean
##  Min.   : 43.79   Min.   : 143.5   Min.   :0.05263   Min.   :0.01938
##  1st Qu.: 75.17   1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492
##  Median : 86.24   Median : 551.1   Median :0.09587   Median :0.09263
##  Mean   : 91.97   Mean   : 654.9   Mean   :0.09636   Mean   :0.10434
##  3rd Qu.:104.10   3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040
##  Max.   :188.50   Max.   :2501.0   Max.   :0.16340   Max.   :0.34540
##  concavity_mean    concave.points_mean symmetry_mean    fractal_dimension_mean
##  Min.   :0.00000   Min.   :0.00000     Min.   :0.1060   Min.   :0.04996
##  1st Qu.:0.02956   1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770
##  Median :0.06154   Median :0.03350     Median :0.1792   Median :0.06154
```

```
##  Mean   :0.08880   Mean   :0.04892     Mean   :0.1812   Mean   :0.06280
##  3rd Qu.:0.13070   3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612
##  Max.   :0.42680   Max.   :0.20120     Max.   :0.3040   Max.   :0.09744
##    radius_se        texture_se      perimeter_se       area_se
##  Min.   :0.1115   Min.   :0.3602   Min.   : 0.757   Min.   :  6.802
##  1st Qu.:0.2324   1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850
##  Median :0.3242   Median :1.1080   Median : 2.287   Median : 24.530
##  Mean   :0.4052   Mean   :1.2169   Mean   : 2.866   Mean   : 40.337
##  3rd Qu.:0.4789   3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190
##  Max.   :2.8730   Max.   :4.8850   Max.   :21.980   Max.   :542.200
##  smoothness_se      compactness_se      concavity_se     concave.points_se
##  Min.   :0.001713   Min.   :0.002252   Min.   :0.00000   Min.   :0.000000
##  1st Qu.:0.005169   1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638
##  Median :0.006380   Median :0.020450   Median :0.02589   Median :0.010930
##  Mean   :0.007041   Mean   :0.025478   Mean   :0.03189   Mean   :0.011796
##  3rd Qu.:0.008146   3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710
##  Max.   :0.031130   Max.   :0.135400   Max.   :0.39600   Max.   :0.052790
##   symmetry_se       fractal_dimension_se  radius_worst   texture_worst
##  Min.   :0.007882   Min.   :0.0008948    Min.   : 7.93   Min.   :12.02
##  1st Qu.:0.015160   1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08
##  Median :0.018730   Median :0.0031870    Median :14.97   Median :25.41
##  Mean   :0.020542   Mean   :0.0037949    Mean   :16.27   Mean   :25.68
##  3rd Qu.:0.023480   3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72
##  Max.   :0.078950   Max.   :0.0298400    Max.   :36.04   Max.   :49.54
##  perimeter_worst    area_worst     smoothness_worst  compactness_worst
##  Min.   : 50.41   Min.   : 185.2   Min.   :0.07117   Min.   :0.02729
##  1st Qu.: 84.11   1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720
##  Median : 97.66   Median : 686.5   Median :0.13130   Median :0.21190
##  Mean   :107.26   Mean   : 880.6   Mean   :0.13237   Mean   :0.25427
##  3rd Qu.:125.40   3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910
##  Max.   :251.20   Max.   :4254.0   Max.   :0.22260   Max.   :1.05800
##  concavity_worst  concave.points_worst symmetry_worst   fractal_dimension_worst
##  Min.   :0.0000   Min.   :0.00000      Min.   :0.1565   Min.   :0.05504
##  1st Qu.:0.1145   1st Qu.:0.06493      1st Qu.:0.2504   1st Qu.:0.07146
##  Median :0.2267   Median :0.09993      Median :0.2822   Median :0.08004
##  Mean   :0.2722   Mean   :0.11461      Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.3829   3rd Qu.:0.16140      3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :1.2520   Max.   :0.29100      Max.   :0.6638   Max.   :0.20750
```

```
table(Data$diagnosis)
```

```
##
##   B   M
## 357 212
```

```
Data=subset(Data,select=(-1))
```

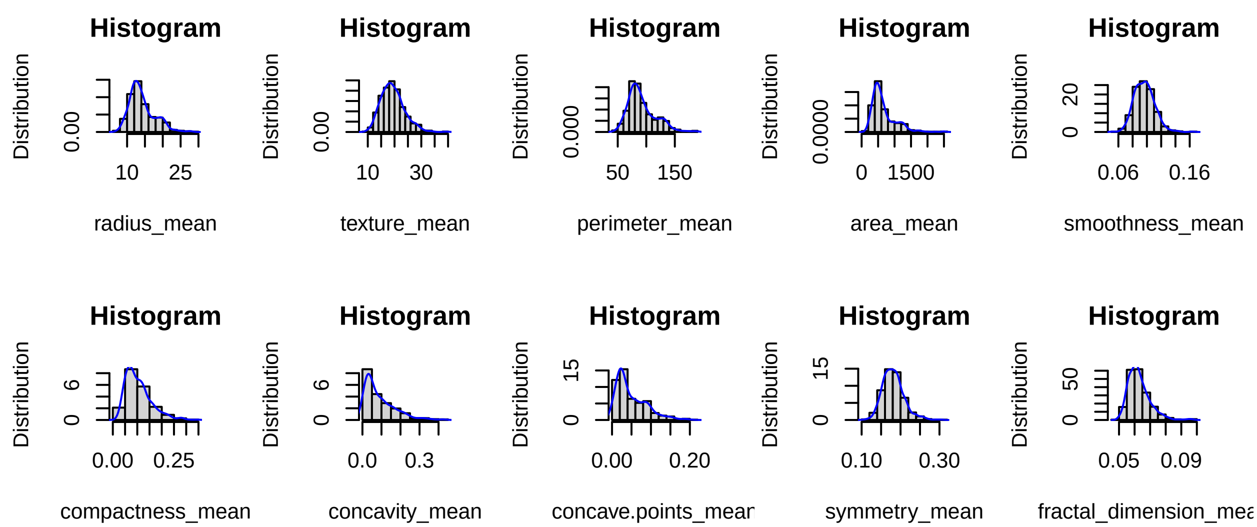
Grouping the dependent variables into the groups for easily analyzing them.

```
meanIdx = grepl('mean', colnames(Data))

seIdx = grepl('se',colnames(Data))
```

```
worstIdx= grepl('worst',colnames(Data))
```

Plotting the Histograms and observing the distribution for mean group data. We can see that some of the features like symmetric mean, smoothness mean, texture mean are uniform. Othere features a little skewed distributions.

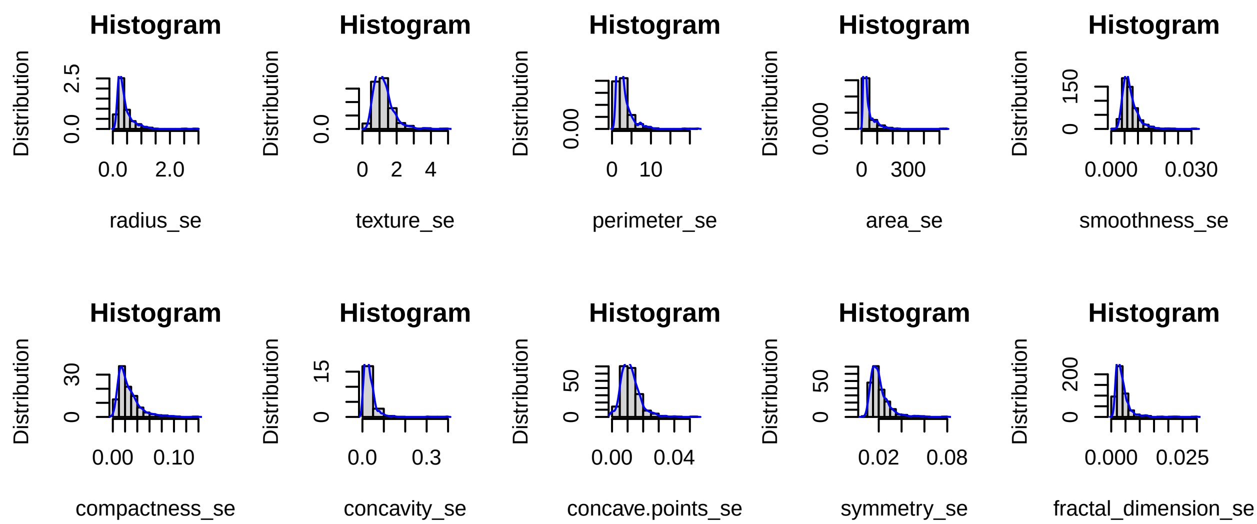
```
meanData=Data[meanIdx]

par(mfrow=c(3,5))

for(i in 1:ncol(meanData)) {       # for-loop over columns
  set.seed(seed = 49078)
  x <-  meanData[ , i]
  hist(main="Histogram", ylab="Distribution",xlab=colnames(meanData)[i],x = x, freq = FALSE)
  lines(x = density(x = x), col = "blue")
}
```



Plotting the Histograms and observing the distribution for standard error group data. Almost all the distributions of this group is appearing skewed.

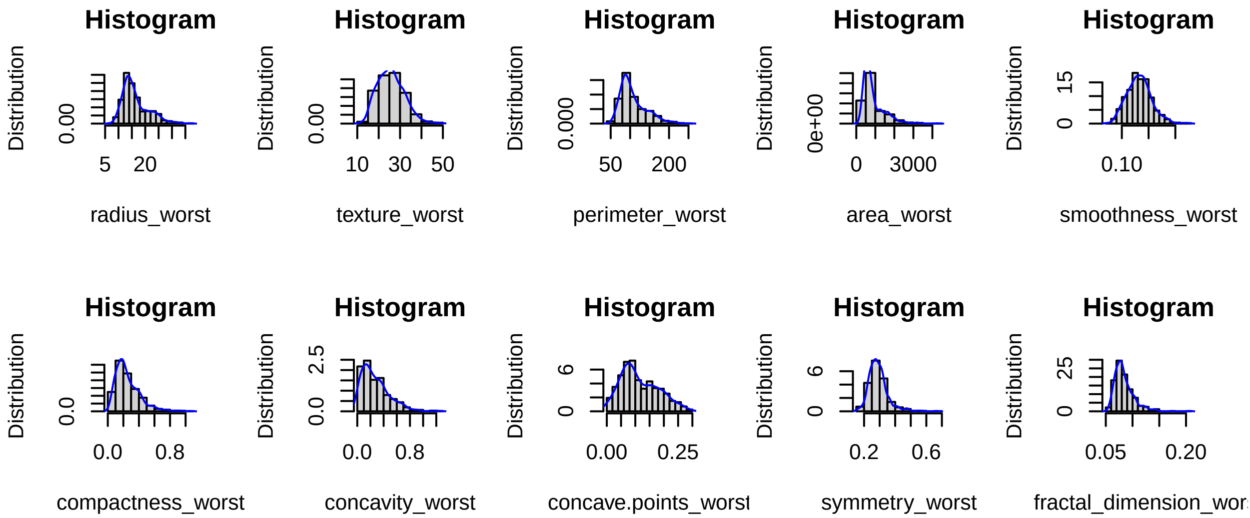
```
seData=Data[seIdx]

par(mfrow=c(3,5))
for(i in 1:ncol(seData)) {       # for-loop over columns
  set.seed(seed = 49078)
```

```
  x <-  seData[ , i]
  hist(main="Histogram", ylab="Distribution",xlab=colnames(seData)[i],x = x, freq = FALSE)
  lines(x = density(x = x), col = "blue")
}
```



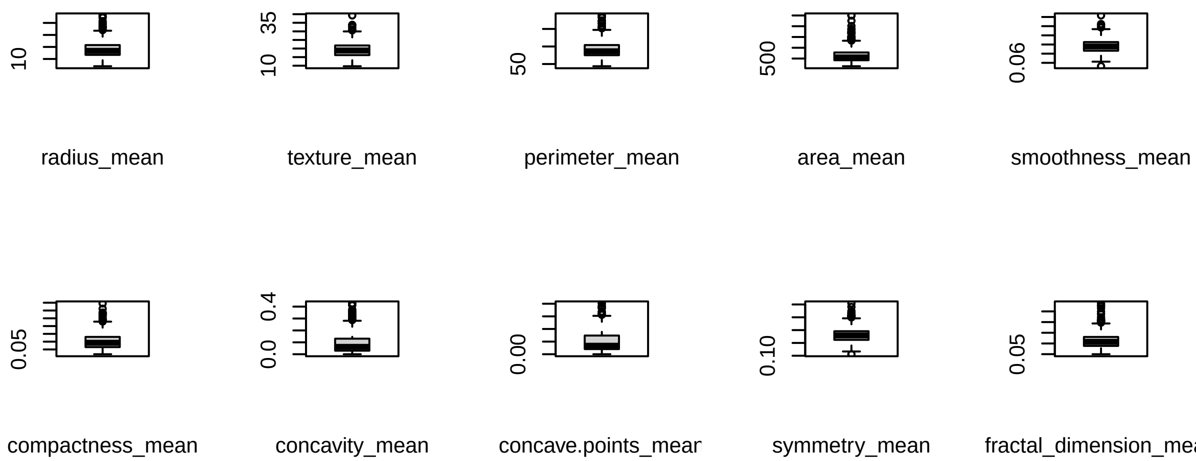
Plotting the Histograms and observing the distribution for worst group data. Almost all the distributions of this group is appearing skewed.

```
worstData=Data[worstIdx]

par(mfrow=c(3,5))
for(i in 1:ncol(worstData)) {       # for-loop over columns
  set.seed(seed = 49078)
  x <-  worstData[ , i]
  hist(main="Histogram", ylab="Distribution",xlab=colnames(worstData)[i],x = x, freq = FALSE)
  lines(x = density(x = x), col = "blue")
}
```



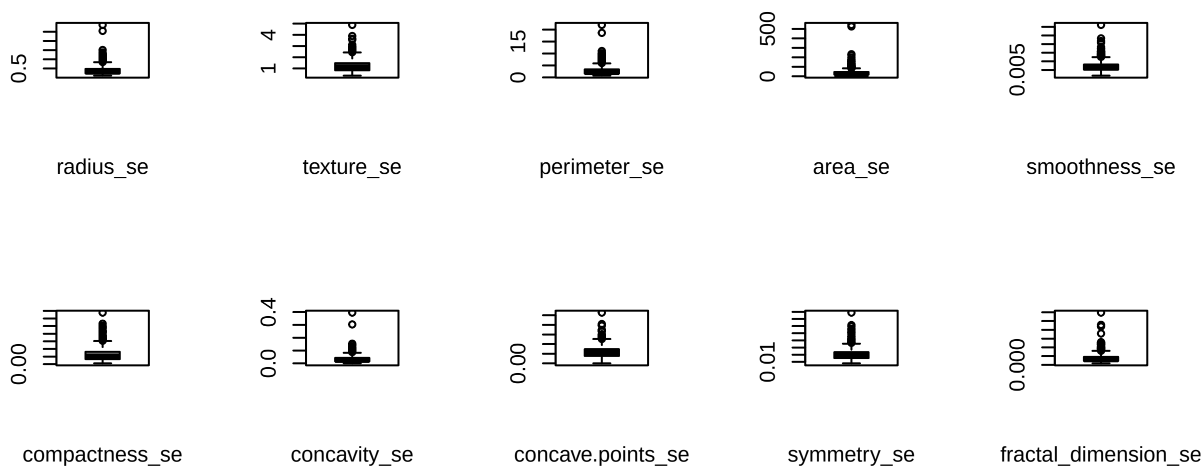
Plotting boxplots to see the outliers in mean data. Most of the dependent variables has outliers.

```
par(mfrow=c(3,5))

for(i in 1:ncol(meanData)) {        # for-loop over columns
  set.seed(seed = 49078)
  x <-  meanData[ , i]
  boxplot(xlab=colnames(meanData)[i],x = x, freq = FALSE)
}
```



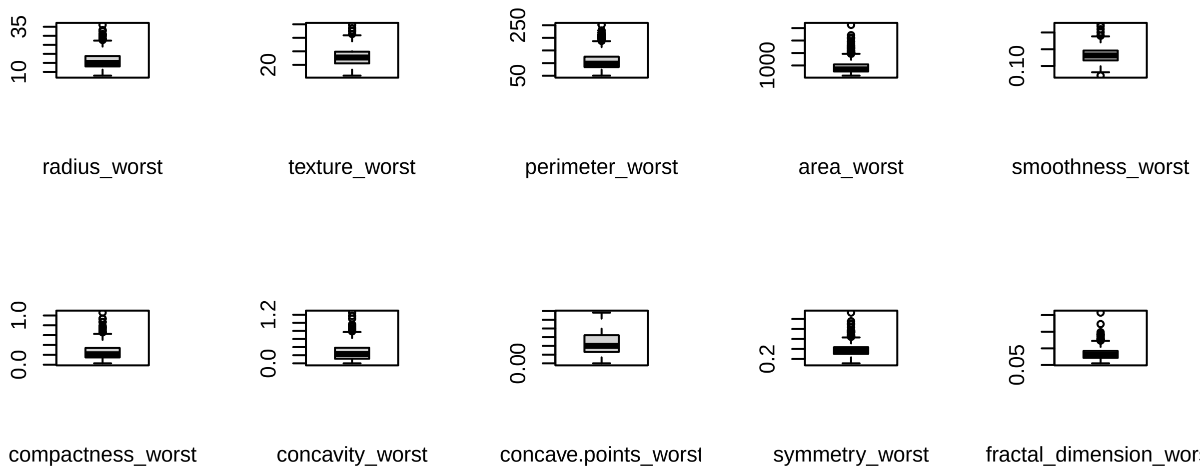
Plotting boxplots to see the outliers in se data. Most of the dependent variables has outliers.

```
par(mfrow=c(3,5))

for(i in 1:ncol(seData)) {       # for-loop over columns
  set.seed(seed = 49078)
  x <-  seData[ , i]
  boxplot(xlab=colnames(seData)[i],x = x, freq = FALSE)
}
```



Plotting boxplots to see the outliers in worst data. Most of the dependent variables has outliers.

```
par(mfrow=c(3,5))
for(i in 1:ncol(worstData)) {        # for-loop over columns
  set.seed(seed = 49078)
  x <-  worstData[ , i]
  boxplot(xlab=colnames(worstData)[i],x = x, freq = FALSE)
}
```



Checking the correlation of data set with the Dependent variable diagnosis we can see only few columns are correlated with diagnosis. We will use this columns to build our model in classification.

```
Data$diagnosis <- ifelse(Data$diagnosis=='M', 1, 0)
library(reshape2)

df=abs(cor(Data[-1:-2],Data[2]))>0.7
df=melt(df)
df[df$value==TRUE,-2]
```

```
##                      Var1 value
## 2          perimeter_mean  TRUE
## 3               area_mean  TRUE
## 7     concave.points_mean  TRUE
## 13                area_se  TRUE
## 20           radius_worst  TRUE
## 22        perimeter_worst  TRUE
## 23             area_worst  TRUE
## 27 concave.points_worst  TRUE
```

```
df$var2==TRUE
```

```
## logical(0)
```

Dividing and splitting the data into train and test data sets

```r
#+ perimeter_mean+ area_worst+ radius_mean
library(caTools)
# Splitting dataset
split <- sample.split(Data, SplitRatio = 0.8)
split
```

```
##  [1] FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## [25]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

```r
train_reg <- subset(Data, split == "TRUE")
test_reg <- subset(Data, split == "FALSE")
```

Implementing SVM Classifier

```r
library(e1071)

classifier = svm(diagnosis~concave.points_worst+ perimeter_worst+ concave.points_mean+ radius_worst+ ar

Y_predicion = predict(classifier, newdata = test_reg)


ConMat=table(test_reg$diagnosis, Y_predicion)
print("confusionMatrix")
```

```
## [1] "confusionMatrix"
```

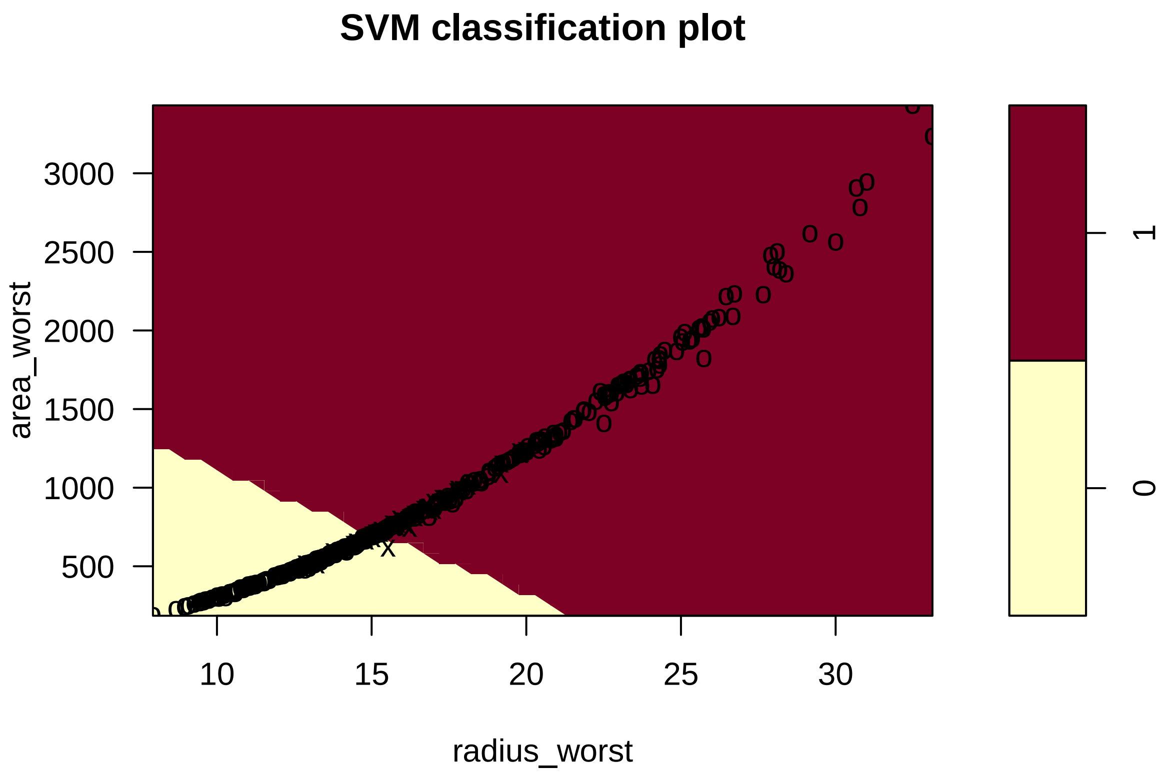
```r
ConMat
```

```
##    Y_predicion
##      0  1
##   0 81  2
##   1  2 44
```

```r
missing_classerr <- mean(Y_predicion != test_reg$diagnosis)
print(paste('Accuracy =', 1-missing_classerr))
```

```
## [1] "Accuracy = 0.968992248062015"
```

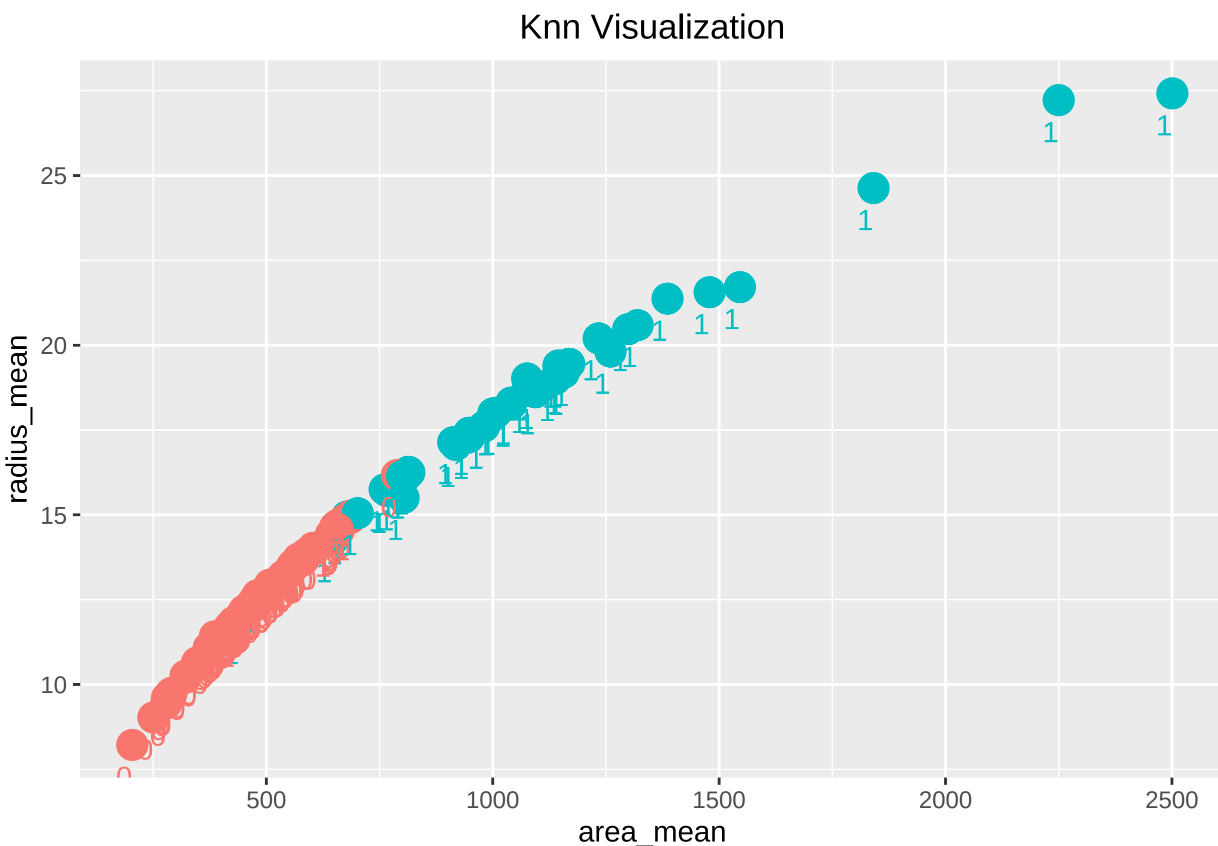
```r
plot(classifier, train_reg,area_worst~radius_worst)
```



Implementing KNN Classifier

```
library(class)

knnModel=knn(train=train_reg, test=test_reg, cl=train_reg$diagnosis, k=21)



# Notice that I am only getting 2 dimensions



 plot_predictions=data.frame(test_reg$diagnosis
,test_reg$concave.points_worst
,test_reg$perimeter_worst
,test_reg$concave.points_mean
,test_reg$radius_worst
,test_reg$area_mean
,test_reg$perimeter_mean
,test_reg$area_worst
,test_reg$radius_mean,predicted=knnModel)

colnames(plot_predictions) <- c("diagnosis",
                                "concave.points_worst",
                                "perimeter_worst",
                                "concave.points_mean",
                                "radius_worst",
```

```
                                  "area_mean",
                                  "perimeter_mean",
                                  "area_worst",
                                  "radius_mean",
                                  'predicted')
# Visualize the KNN algorithm results.
library(ggplot2)

ggplot(plot_predictions, aes(area_mean, radius_mean, color = predicted, fill = predicted)) +
  geom_point(size = 5) +
  geom_text(aes(label=diagnosis),hjust=1, vjust=2) +
  ggtitle("Knn Visualization") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(legend.position = "none")
```




```
confMatrix=table(test_reg$diagnosis, knnModel)
print("confusionMatrix")
```

```
## [1] "confusionMatrix"
```

```
confMatrix
```

```
##    knnModel
##       0   1
```

```
##   0 82  1
##   1  5 41
```

```
missing_classerr <- mean(test_reg$diagnosis != knnModel)
print(paste('Accuracy =', 1-missing_classerr))
```

```
## [1] "Accuracy = 0.953488372093023"
```

Implementing logistic regression

```
# Training model
logistic_model <- glm(diagnosis~concave.points_worst+ perimeter_worst+ concave.points_mean
                      + radius_worst+ area_mean+ perimeter_mean+ area_worst+ radius_mean
                      ,family=binomial("logit"),data = train_reg)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
predictData=predict(logistic_model,data.frame(concave.points_worst=test_reg$concave.points_worst,perime

library(InformationValue)

origTest=test_reg$diagnosis

#find optimal cutoff probability to use to maximize accuracy
optimal <- optimalCutoff(test_reg, predictData)[1]
optimal
```

```
## [1] 0.63
```

```
predictDif <- ifelse(predictData>optimal, 0, 1)

library(ggplot2)




print("confusionMatrix")
```

```
## [1] "confusionMatrix"
```

```
table(origTest, predictDif)
```

```
##         predictDif
## origTest  0  1
##        0  0 83
##        1 44  2
```

```r
missing_classerr <- mean(predictDif != origTest)
print(paste('Accuracy =', 1 - missing_classerr))
```

```
## [1] "Accuracy = 0.0155038759689923"
```

```r
ggplot(Data, aes(x=radius_mean, y=diagnosis)) + geom_point() +

      stat_smooth(method="glm", color="red", se=FALSE,

                method.args = list(family=binomial))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```