Breast Cancer Analysys in R

Cynthia L. McGinnis

February 23, 2019

Breast Cancer Analysis

Introduction - This script uses the "Breast Cancer Wisconsin (Diagnostic) Data Set" to predict cancer diagnosis based on cell features.

The data was obtained from the Breast Cancer Wisconsin (Diagnostic) Data Set Data Folder: https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/

The description of the data set can be found on: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

"Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at [Web Link]

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34]."

This database is also available through the UW CS ftp server: ftp ftp.cs.wisc.edu cd math-prog/cpo-dataset/machine-learn/WDBC/ Attribute Domain

Features:

1. Sample code number id number
2. Clump Thickness 1 - 10
3. Uniformity of Cell Size 1 - 10
4. Uniformity of Cell Shape 1 - 10
5. Marginal Adhesion 1 - 10
6. Single Epithelial Cell Size 1 - 10
7. Bare Nuclei 1 - 10
8. Bland Chromatin 1 - 10
9. Normal Nucleoli 1 - 10
10. Mitoses 1 - 10 1
11. Class: (2 for benign, 4 for malignant)

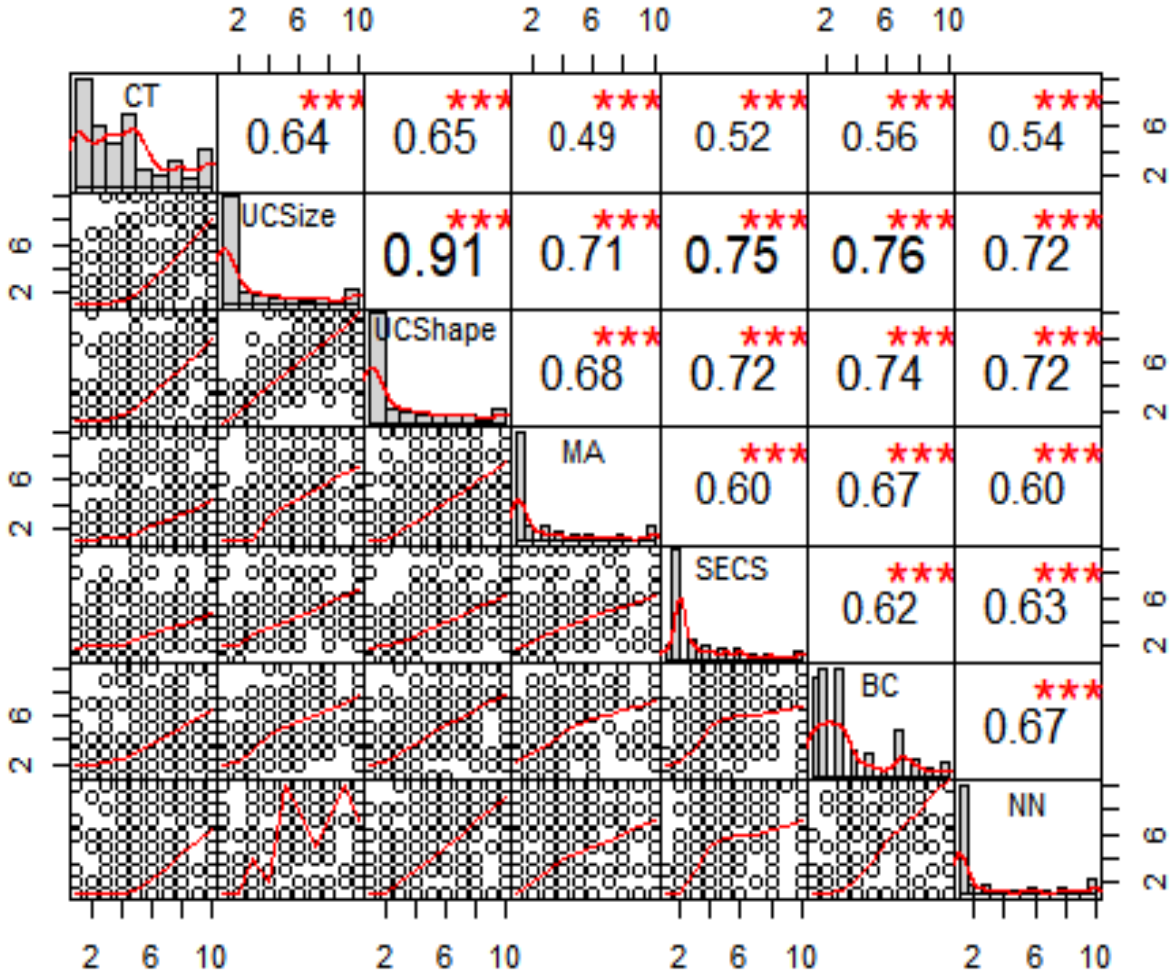
There are 16 instances in Groups 1 to 6 that contain a single missing (i.e., unavailable) attribute value, now denoted by "?".

To create a workable data set, the id number was dropped, the "?" removed from the data and the diagnosis was converted to 1 for malignant and 0 for benign.

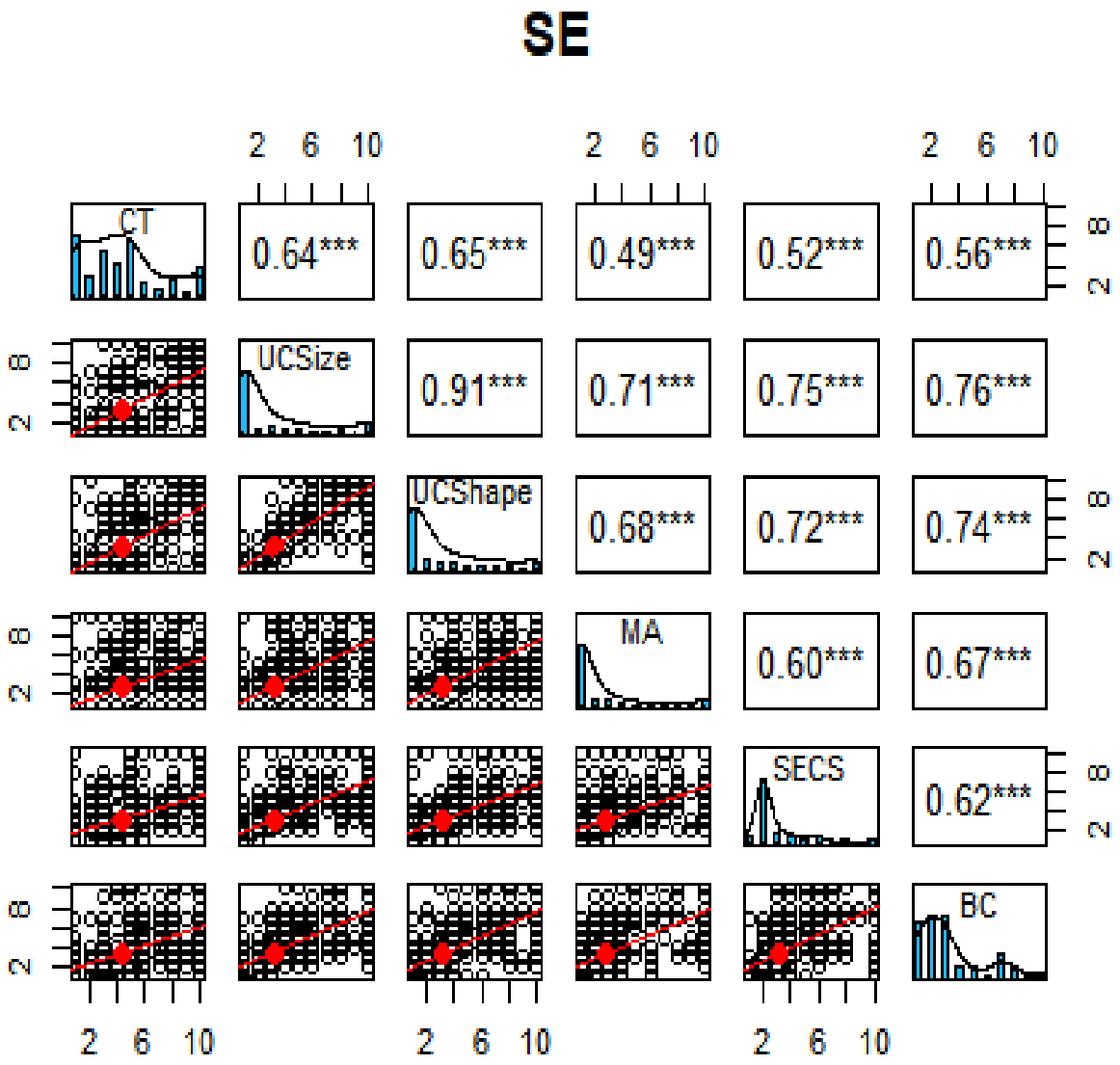
	CT <int>	UCSize <int>	UCShape <int>	MA <int>	SECS <int>	BC <int>	NN <int>	M <int>	diagnosis <int>
1	5	1	1	1	2	3	1	1	2
2	5	4	4	5	7	3	2	1	2
3	3	1	1	1	2	3	1	1	2
4	6	8	8	1	3	3	7	1	2
5	4	1	1	3	2	3	1	1	2
6	8	10	10	8	7	9	7	1	4

Data Visualizations

Correlation Chart for Means



Correlation Chart for SE



Split the dataset into the training sample and the testing sample

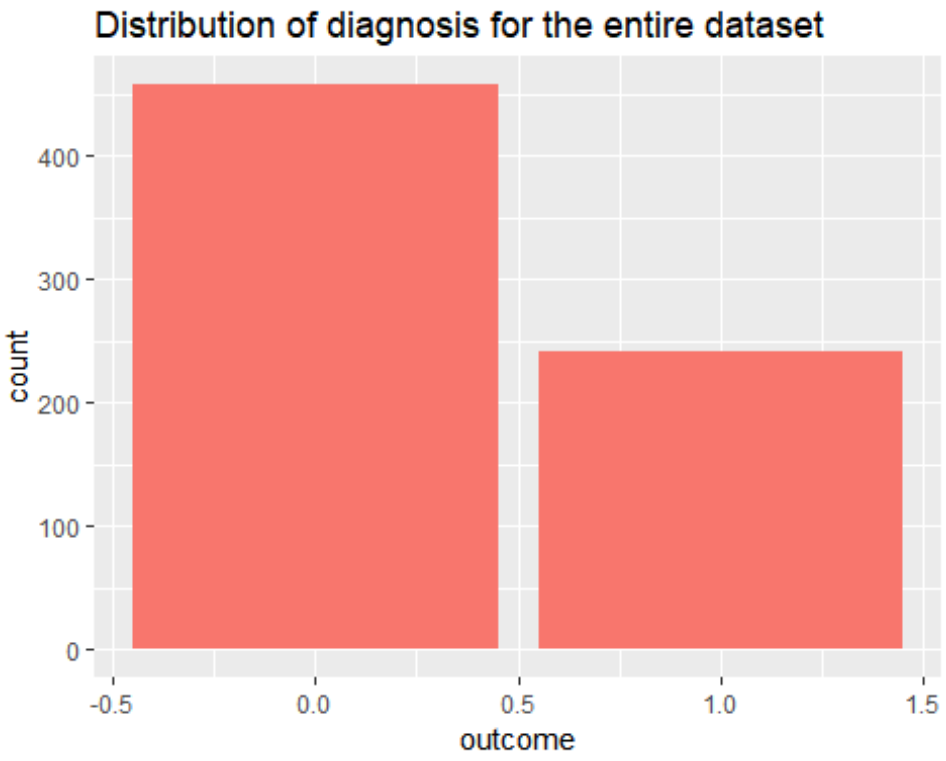
```
sample_size = floor(0.5 * nrow(data2))

# set the seed to make your partition reproductible
set.seed(1729)
train_set = sample(seq_len(nrow(data2)), size = sample_size)

training = data2[train_set, ]

testing = data2[-train_set, ]

head(training)
```

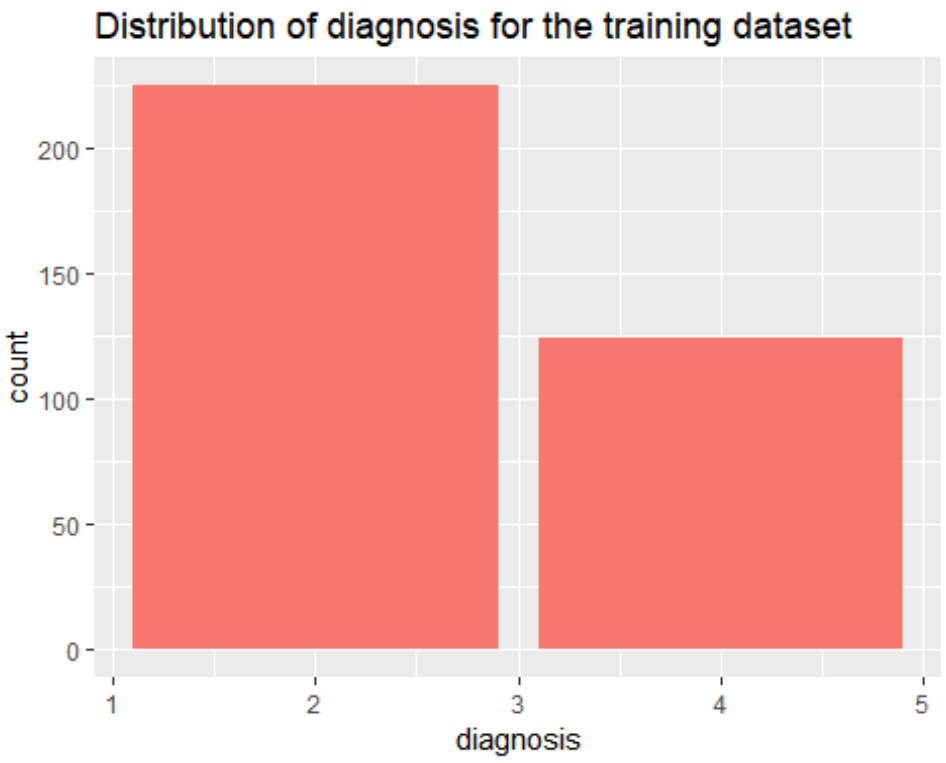
```
##     CT UCSize UCShape MA SECS BC NN M diagnosis outcome
## 410  3      1       2  1    2  2  1 1         2       0
## 306 10      8       4  4    4  3 10 4         4       1
## 400  1      2       3  1    2  1  1 1         2       0
## 246  5      1       1  2    2  3  1 1         2       0
## 599  3      1       1  1    2  2  1 1         2       0
## 286  8     10      10 10    8 10  7 3         4       1
```

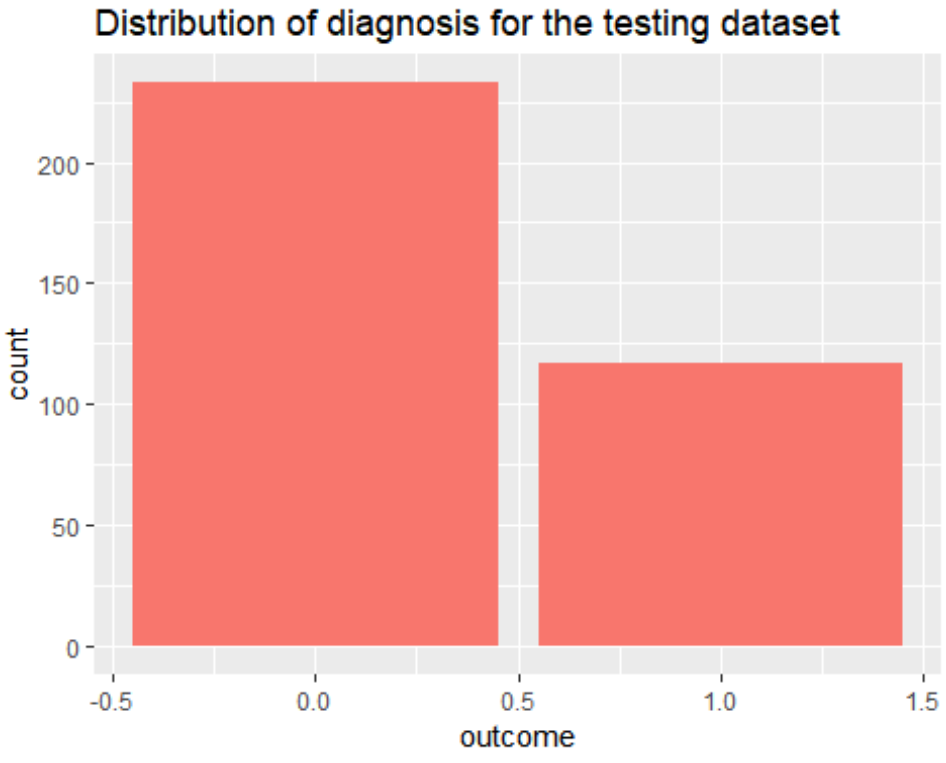
```
head(testing)
```

```
##    CT UCSize UCShape MA SECS BC NN M diagnosis outcome
## 2   5      4       4  5    7  3  2 1         2       0
## 4   6      8       8  1    3  3  7 1         2       0
## 6   8     10      10  8    7  9  7 1         4       1
## 7   1      1       1  1    2  3  1 1         2       0
## 8   2      1       2  1    2  3  1 1         2       0
## 13  5      3       3  3    2  4  4 1         4       1
```

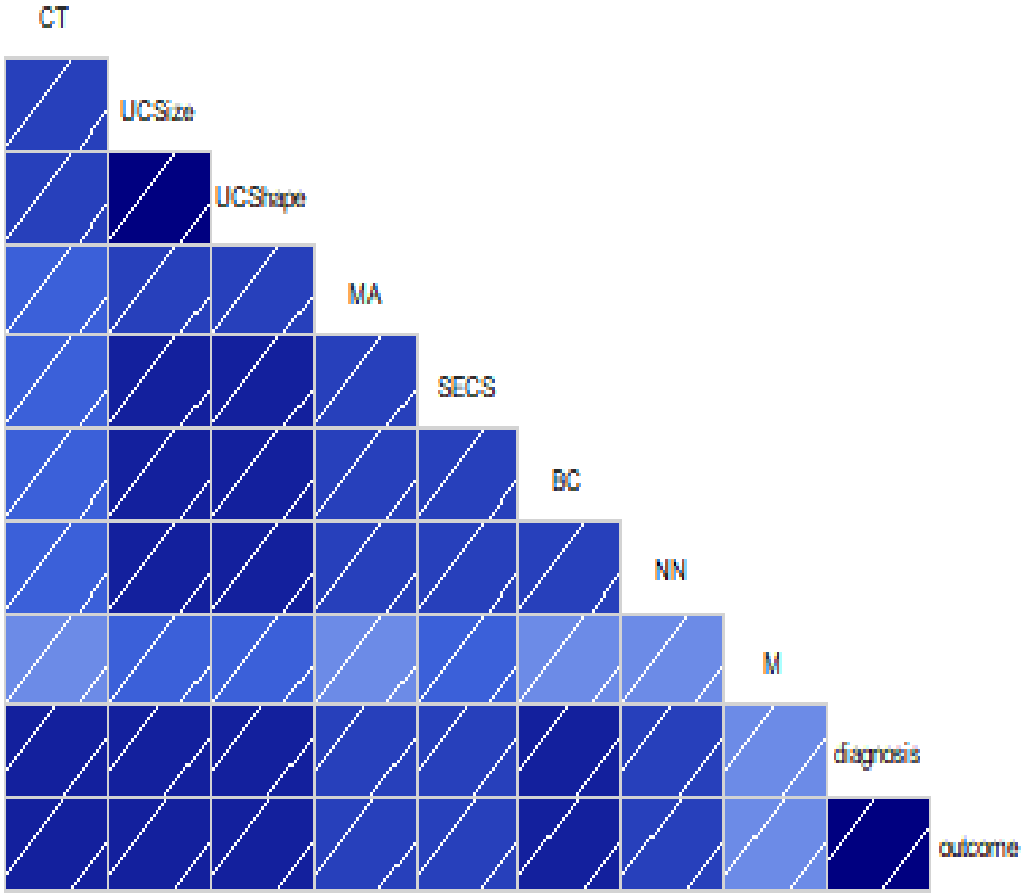
Does the Data Set, Training Data and Testing Data have the same characteristics?



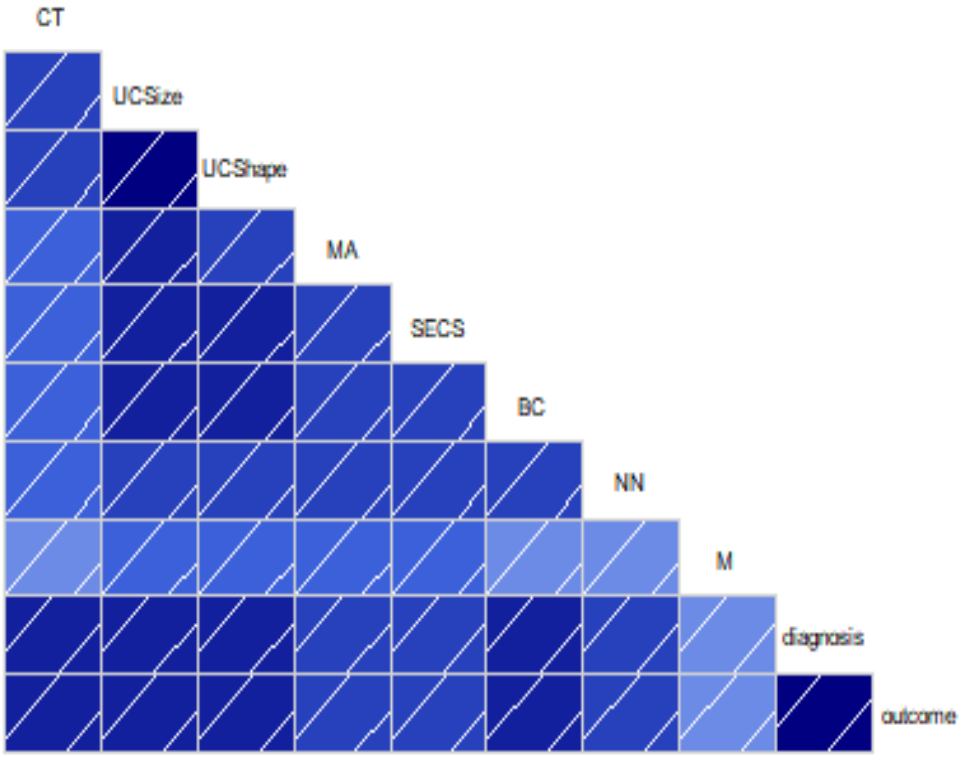

Distribution of diagnosis for the training dataset
count
200
150
100
50
0
1
2
3
4
5
diagnosis


Distribution of diagnosis for the testing dataset
count
200
150
100
50
0
-0.5
0.0
0.5
1.0
1.5
outcome

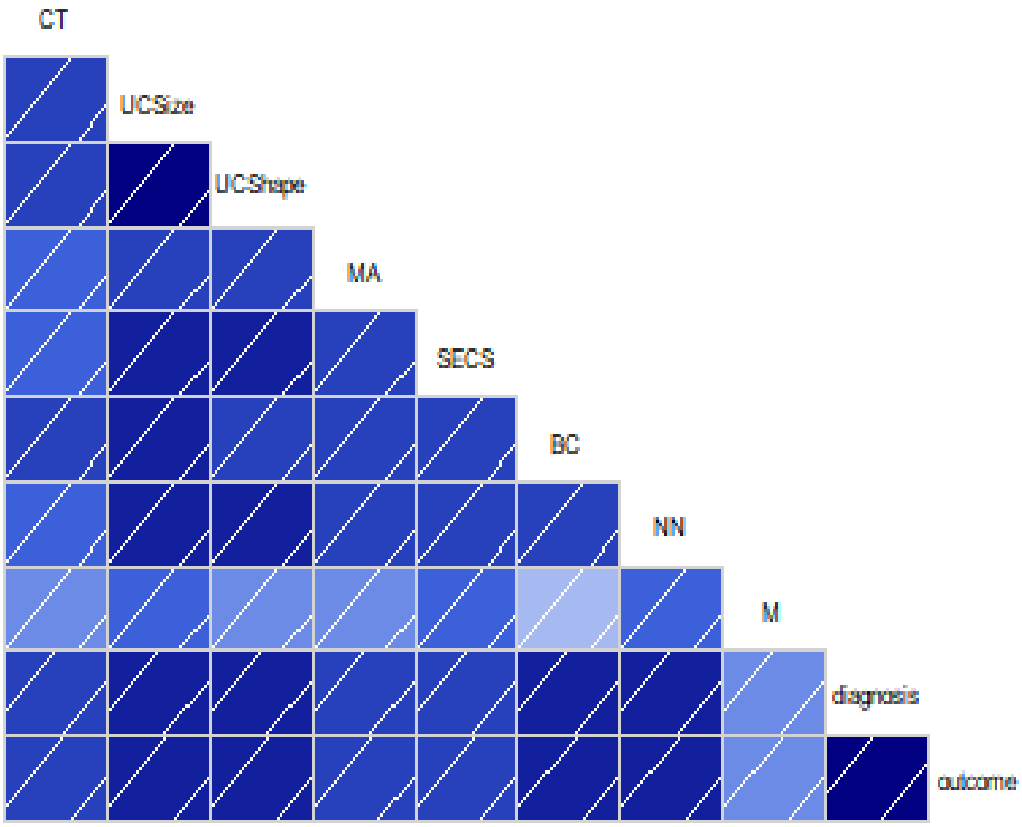
Corrgram of the data



Corrgram of the training data



Corrgram of the testing data



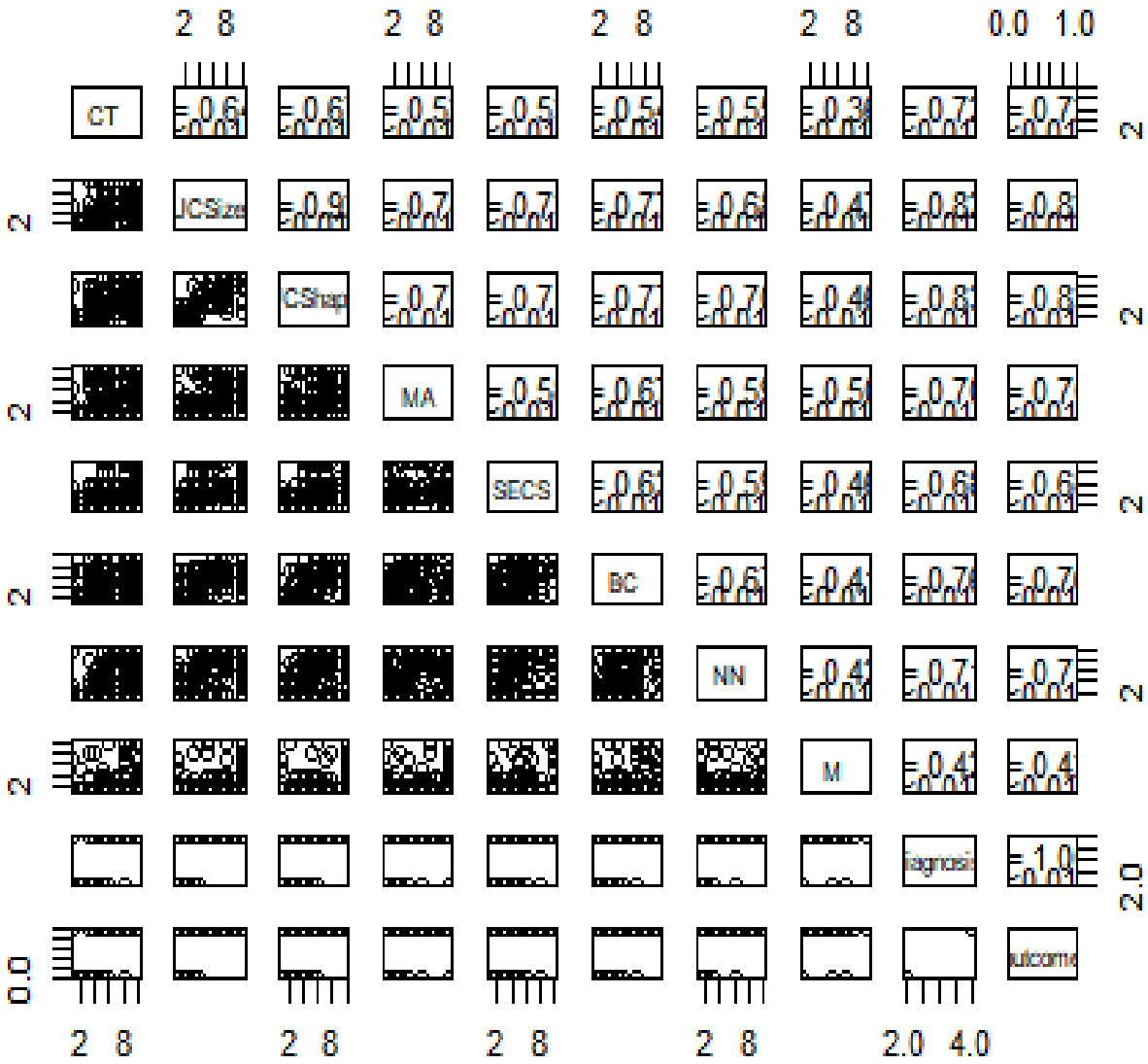
Data Analysis

Calculating the Correlation Coeficients and p-values

```
panel.cor <- function(x, y, digits = 2, cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  # correlation coefficient
  r <- cor(x, y)
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste("r= ", txt, sep = "")
  text(0.5, 0.6, txt)

  # p-value calculation
  p <- cor.test(x, y)$p.value
  txt2 <- format(c(p, 0.123456789), digits = digits)[1]
  txt2 <- paste("p= ", txt2, sep = "")
  if(p<0.01) txt2 <- paste("p= ", "<0.01", sep = "")
  text(0.2, 0.1, txt2)
}
```

Graph the linear correlation coef

```
pairs(training, upper.panel = panel.cor)
```



Fit the model using glm Generalize Linear Model

```
# Model Fitting
# Start off with this (alpha = 0.05)
model_algorithm = model = glm(outcome ~ CT +
                                UCSize +
                                UCShape +
                                MA +
                                SECS +
                                BC  +
                                NN  +
                                M ,
                              family=binomial(link='logit'), control = list(
maxit = 50),data=training)

print(summary(model_algorithm))

##
## Call:
## glm(formula = outcome ~ CT + UCSize + UCShape + MA + SECS + BC +
##     NN + M, family = binomial(link = "logit"), data = training,
##     control = list(maxit = 50))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.85231  -0.11326  -0.05213   0.00449   2.99989
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.0443     2.0547  -5.862 4.58e-09 ***
## CT            0.7113     0.2176   3.268  0.00108 **
## UCSize        0.3156     0.4604   0.686  0.49293
## UCShape       0.5103     0.4737   1.077  0.28134
## MA            0.3715     0.1783   2.083  0.03721 *
## SECS          0.1318     0.2286   0.577  0.56416
## BC            0.7649     0.2959   2.585  0.00974 **
## NN            0.2009     0.2191   0.917  0.35896
## M             0.9075     0.5499   1.650  0.09891 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 454.165  on 348  degrees of freedom
## Residual deviance:  50.827  on 340  degrees of freedom
## AIC: 68.827
##
## Number of Fisher Scoring iterations: 10

print(anova(model_algorithm, test="Chisq"))
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: outcome
##
## Terms added sequentially (first to last)
##
##
##         Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                      348     454.17
## CT       1  225.630       347     228.54 < 2.2e-16 ***
## UCSize   1  144.642       346      83.89 < 2.2e-16 ***
## UCShape  1   11.469       345      72.42 0.0007078 ***
## MA       1    6.520       344      65.90 0.0106666 *
## SECS     1    2.019       343      63.89 0.1553909
## BC       1   10.654       342      53.23 0.0010984 **
## NN       1    1.012       341      52.22 0.3144747
## M        1    1.393       340      50.83 0.2378188
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using Uniform Cell size and Uniform Cell Shape as predictors of diagnosis

```
# Settled Uniform Cell Size and Uniform Cell Shape
model_algorithm_final = model = glm(outcome ~ UCSize + UCShape ,
                                    family=binomial(link='logit'), control =
list(maxit = 50),data=training)

print(summary(model_algorithm_final))

##
## Call:
## glm(formula = outcome ~ UCSize + UCShape, family = binomial(link = "logit"
),
##     data = training, control = list(maxit = 50))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.72698  -0.18243  -0.18243   0.00852   2.86504
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.0123     0.6645  -9.047  < 2e-16 ***
## UCSize        0.9213     0.2853   3.229 0.001243 **
## UCShape       1.0035     0.2631   3.814 0.000137 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 454.165  on 348  degrees of freedom
## Residual deviance:  98.788  on 346  degrees of freedom
## AIC: 104.79
##
## Number of Fisher Scoring iterations: 7
```

```
model_algorithm_final = model = glm(outcome ~ UCSize + UCShape + MA ,
                                    family=binomial(link='logit'), control =
list(maxit = 50),data=training)

print(summary(model_algorithm_final))
```

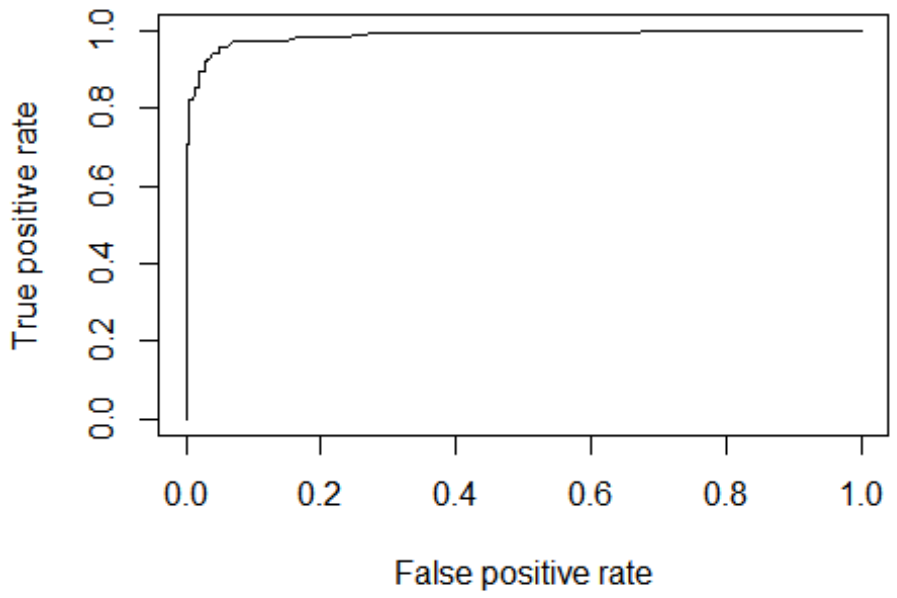
```
##
## Call:
## glm(formula = outcome ~ UCSize + UCShape + MA, family = binomial(link = "l
ogit"),
##     data = training, control = list(maxit = 50))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.72927  -0.16094  -0.16094   0.00576   2.95060
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -6.5049     0.7524  -8.645  < 2e-16 ***
## UCSize        0.7823     0.3031   2.581 0.009864 **
## UCShape       0.9761     0.2760   3.537 0.000405 ***
## MA            0.4064     0.1508   2.695 0.007042 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 454.17  on 348  degrees of freedom
## Residual deviance:  90.25  on 345  degrees of freedom
## AIC: 98.25
##
## Number of Fisher Scoring iterations: 8
```

Apply the algorithm to the training sample

```
prediction_training = predict(model_algorithm_final,training, type = "respons
e")
prediction_training = ifelse(prediction_training > 0.5, 1, 0)
error = mean(prediction_training != training$outcome)
print(paste('Model Accuracy',1-error))

## [1] "Model Accuracy 0.951289398280802"
```

Calculate the ROC curve and the AUC

```
# Get the ROC curve and the AUC
p = predict(model_algorithm_final, training, type="response")
pr = prediction(p, training$outcome)
prf = performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```



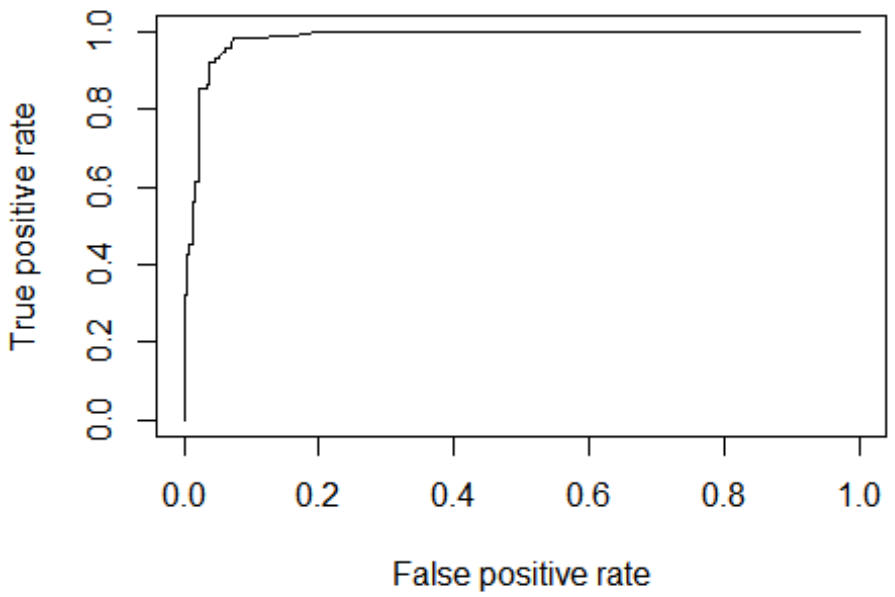
```r
auc = performance(pr, measure = "auc")
auc = auc@y.values[[1]]
print(paste("Model Accuracy", auc))
```

```
## [1] "Model Accuracy 0.986236559139785"
```

```r
# Apply the algorithm to the testing sample
prediction_testing = predict(model_algorithm_final,testing, type = "response"
)
prediction_testing = ifelse(prediction_testing > 0.5, 1, 0)
error = mean(prediction_testing != testing$outcome)
print(paste('Model Accuracy',1-error))
```

```
## [1] "Model Accuracy 0.942857142857143"
```

```r
# Get the ROC curve and the AUC
p = predict(model_algorithm_final, testing, type="response")
pr = prediction(p, testing$outcome)
prf = performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```



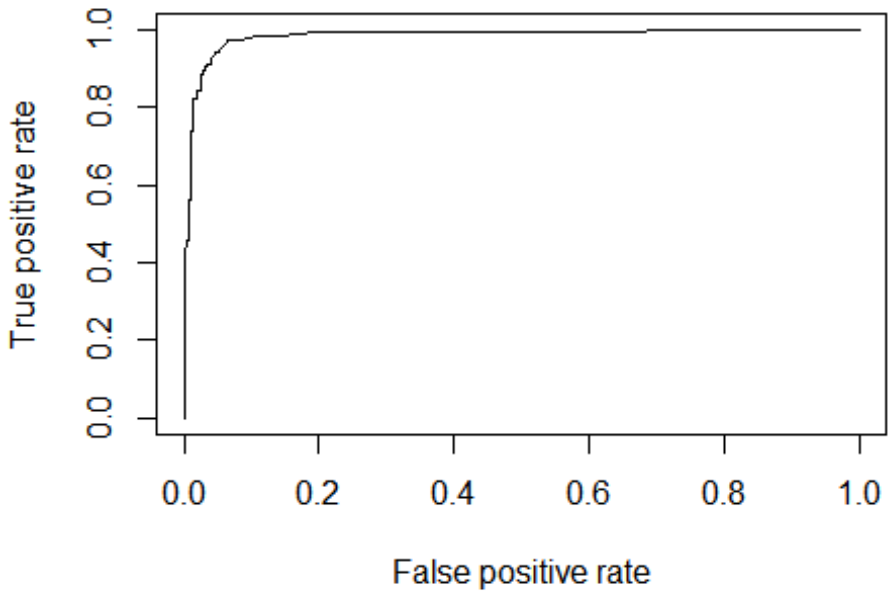

```r
auc = performance(pr, measure = "auc")
auc = auc@y.values[[1]]
print(paste("Model Accuracy", auc))
```

```
## [1] "Model Accuracy 0.982997689006273"
```

```r
# Apply the algorithm to the entire dataset
prediction_data = predict(model_algorithm_final,data, type = "response")
prediction_data = ifelse(prediction_data > 0.5, 1, 0)
error = mean(prediction_data != data$outcome)
print(paste('Model Accuracy',1-error))
```

```
## [1] "Model Accuracy 0.947067238912732"
```

```r
# Get the ROC curve and the AUC
p = predict(model_algorithm_final, data, type="response")
pr = prediction(p, data$outcome)
prf = performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```r
auc = performance(pr, measure = "auc")
auc = auc@y.values[[1]]
print(paste("Model Accuracy", auc))
```

```
## [1] "Model Accuracy 0.984245048832195"
```

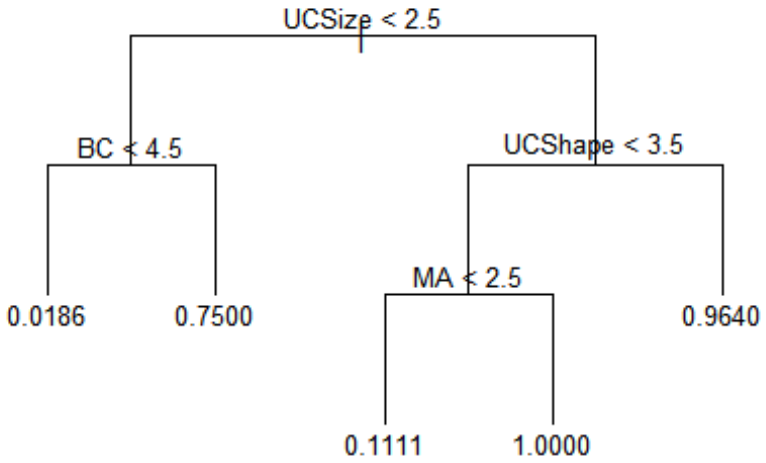
Decision Tree

```r
# Droping the outcome variable which was used for the logistic model

training$outcome = NULL
testing$outcome = NULL

training$diagnosis[training$diagnosis == 4] = 1
```

```r
training$diagnosis[training$diagnosis ==2] = 0

# Running our first tree
model_tree = tree(diagnosis ~ UCSize +
                    UCShape +
                    MA +
                    SECS +
                    BC  +
                    NN  +
                    M,
                    data = training)

summary(model_tree)
```

```
##
## Regression tree:
## tree(formula = diagnosis ~ UCSize + UCShape + MA + SECS + BC +
##     NN + M, data = training)
## Variables actually used in tree construction:
## [1] "UCSize"  "BC"      "UCShape" "MA"
## Number of terminal nodes:  5
## Residual mean deviance:  0.02956 = 10.17 / 344
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.96400 -0.01860 -0.01860  0.00000  0.03604  0.98140
```

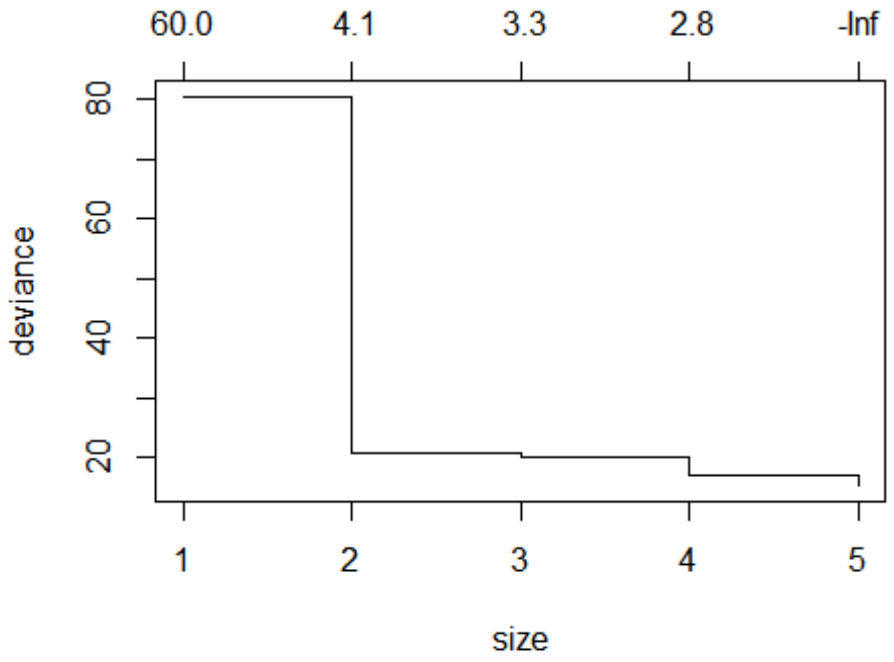
```r
# Now we want to plot our results
plot(model_tree, type = "uniform")

# Add some text to the plot
text(model_tree, pretty = 0, cex=0.8)
```



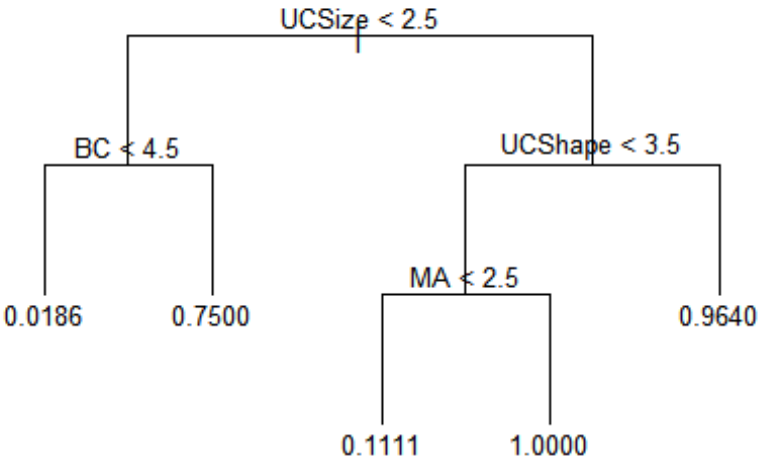

```
# Check the tree on the training data
# Distributional prediction

model_tree_pred_train = predict(model_tree, training) # gives the probability
for each class

model_tree_pred_test = predict(model_tree, testing) # gives the probability f
or each class

# Try to prune the tree to avoid over fitting
cv.tree(model_tree)
```

```
## $size
## [1] 5 4 3 2 1
##
## $dev
## [1] 19.77385 21.55135 22.03910 21.66386 80.32060
##
## $k
## [1]      -Inf  2.844444  3.267954  4.125988 59.533981
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
plot(cv.tree(model_tree)) # Seems like a tree of size 5 might be best
```



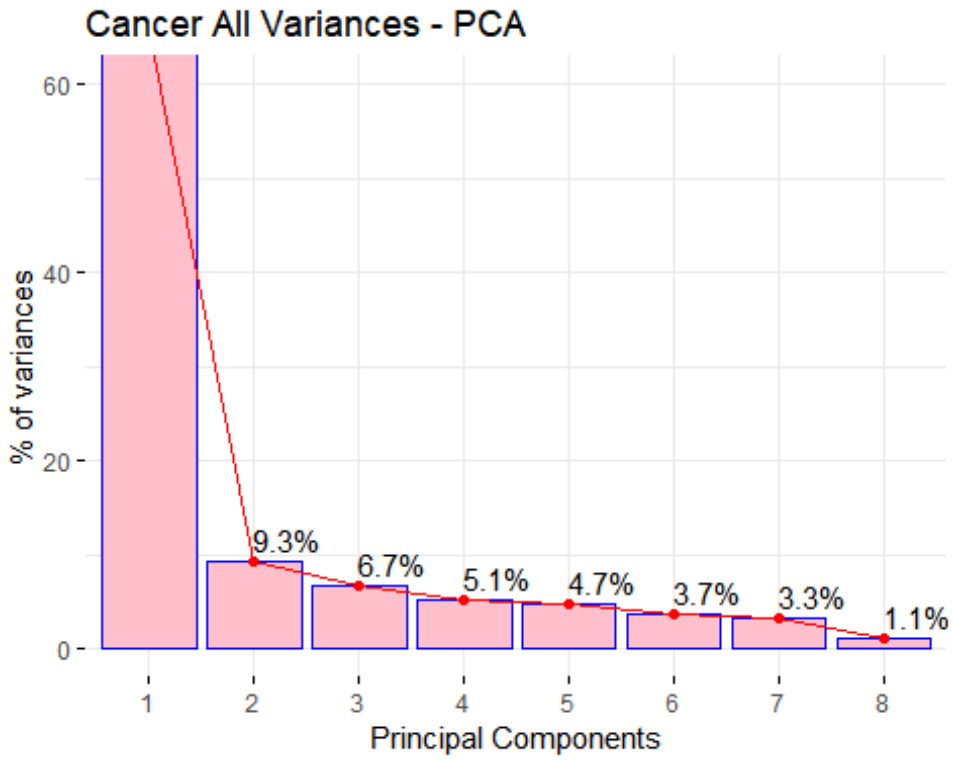

```r
# Pruned model
model_tree_prune = prune.tree(model_tree, best = 5)
summary(model_tree_prune)

##
## Regression tree:
## tree(formula = diagnosis ~ UCSize + UCShape + MA + SECS + BC +
##     NN + M, data = training)
## Variables actually used in tree construction:
## [1] "UCSize"  "BC"      "UCShape" "MA"
## Number of terminal nodes:  5
## Residual mean deviance:  0.02956 = 10.17 / 344
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.96400 -0.01860 -0.01860  0.00000  0.03604  0.98140

# Now we want to plot our results
plot(model_tree_prune, type = "uniform")

# Add some text to the plot
text(model_tree, pretty = 0, cex=0.8)
```

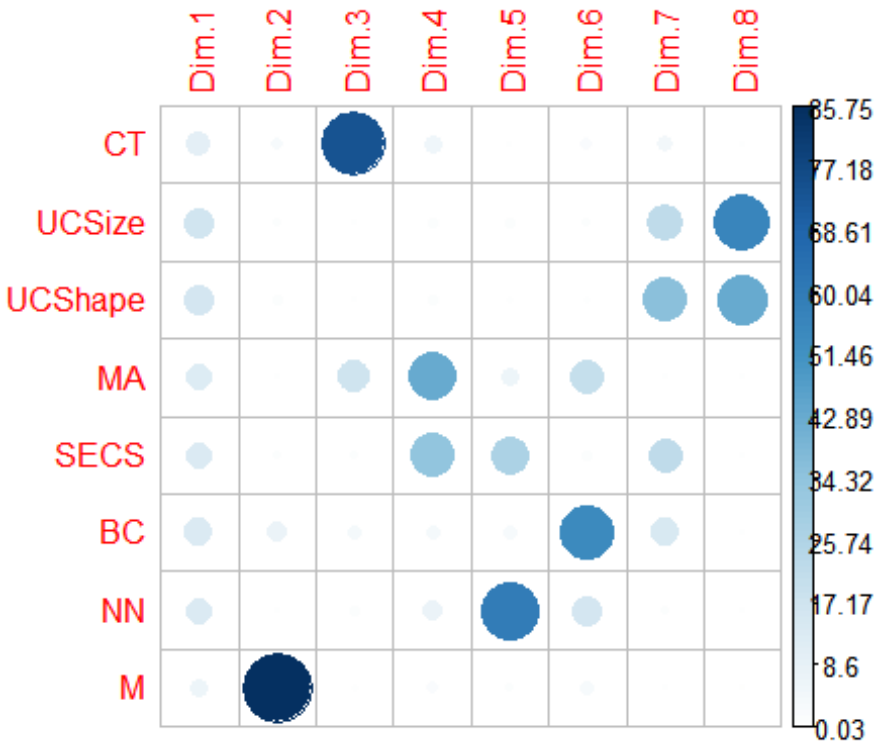

UCSize < 2.5
BC < 4.5
UCShape < 3.5
MA < 2.5
0.0186
0.7500
0.9640
0.1111
1.0000

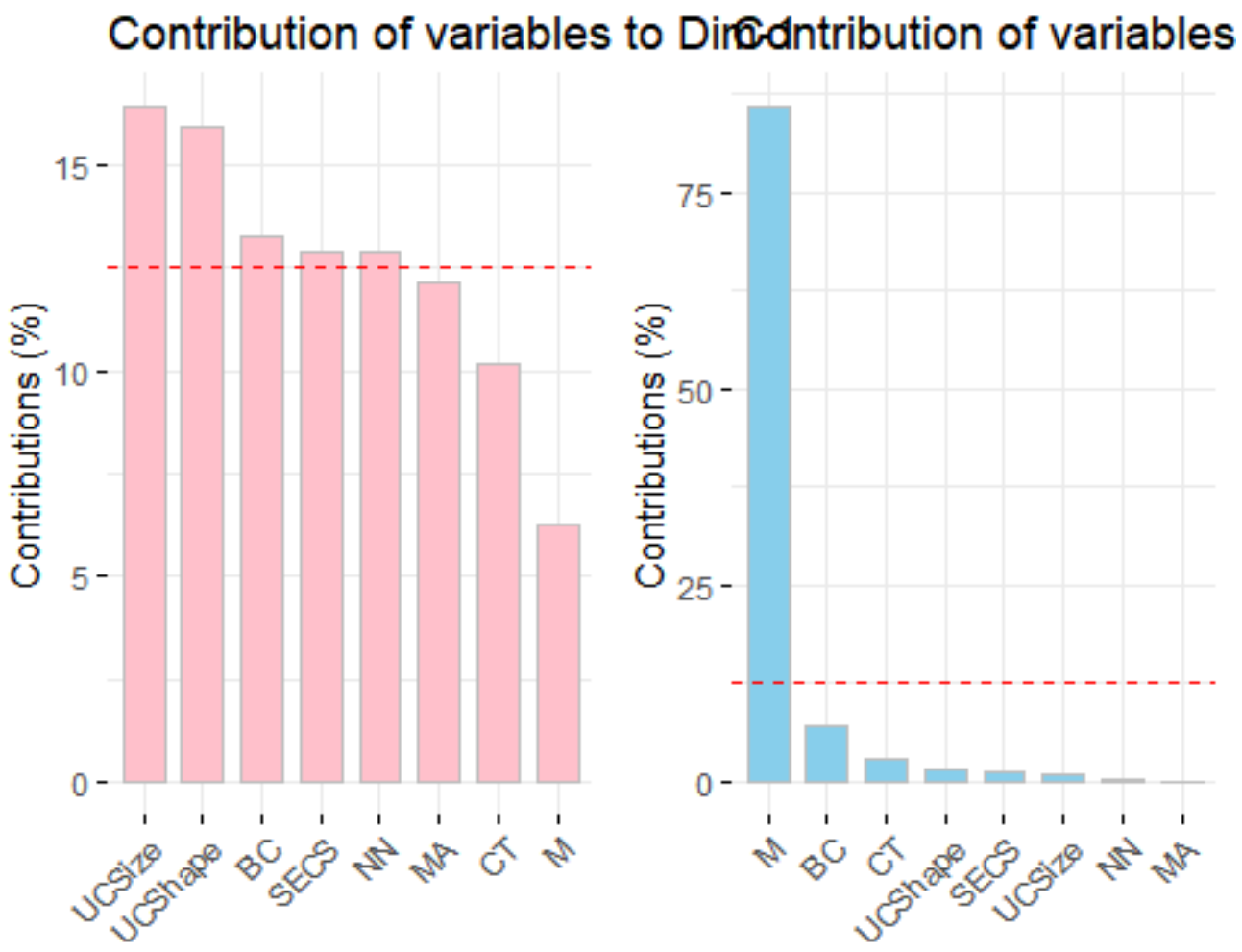
Principle Component Analysis

```
summary(all_pca)
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     2.2975 0.86438 0.73405 0.63905 0.61290 0.54618
## Proportion of Variance 0.6598 0.09339 0.06735 0.05105 0.04696 0.03729
## Cumulative Proportion  0.6598 0.75322 0.82057 0.87162 0.91858 0.95586
##                           PC7    PC8
## Standard deviation     0.51251 0.3007
## Proportion of Variance 0.03283 0.0113
## Cumulative Proportion  0.98870 1.0000
```




Dim.1
Dim.2
Dim.3
Dim.4
Dim.5
Dim.6
Dim.7
Dim.8
CT
UCSize
UCShape
MA
SECS
BC
NN
M
85.75
77.18
68.61
60.04
51.46
42.89
34.32
25.74
17.17
8.6
0.03


Contribution of variables to Dim-1
Contribution of variables
Contributions (%)
15
10
5
0
UCSize
UCShape
BC
SECS
NN
MA
CT
M
Contributions (%)
75
50
25
0
M
BC
CT
UCShape
SECS
UCSize
NN
MA

Conclusion

From the above analysis, two of the most likely determining factors for malignant or benign is clump thickness and uniform cell size. The model accuracy was 0.95. Further analysis was done using PCA's which also show clump thickness and uniform cell size as predictors.