STA631Project

Abu, Sayed, Saheli

4/25/2023

Utility Functions

```
precision = function(tbl) {
  denom = (tbl[2,1]+tbl[2,2])
  if(denom == 0){
    return (NA)
  }
  return ((tbl[2,2]/denom) * 100)
}

recall = function(tbl) {
  denom = (tbl[1,2]+tbl[2,2])
  if(denom == 0){
    return (NA)
  }
  return ((tbl[2,2]/denom) * 100)
}

accuracy = function(tbl) {
  return (((tbl[1,1]+tbl[2,2])/(tbl[1,1]+tbl[1,2]+tbl[2,1]+tbl[2,2])) * 100)
}

minmax=function(v){
  mx=max(v);
  mn=min(v);
  return((v-mn)/(mx-mn))
}
```

Exploratory Analysis

Load Data

```
# Load necessary libraries
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load Data
bcw_data = read.csv("breast-cancer-wisconsin.data", sep = ",", header = FALSE)

# Rename columns
colnames(bcw_data) = c("id", "clumpThickness", "unifCellSize", "unifCellShape", "MarginalAdhesio
n", "SingEpCellSize", "BareNuclei", "BlandChromatin", "NormalNucleoli", "Mitosis", "Diagnosis")

# Check summary for missing/noisy data
for (i in 2:ncol(bcw_data)) {
  bcw_data[,i] = as.factor(bcw_data[,i])
}

summary(bcw_data[,2:11], maxsum = 20)
```

```
##  clumpThickness unifCellSize unifCellShape MarginalAdhesion SingEpCellSize
##  1 :145         1 :384       1 :353        1 :407           1 : 47
##  2 : 50         2 : 45       2 : 59        2 : 58           2 :386
##  3 :108         3 : 52       3 : 56        3 : 58           3 : 72
##  4 : 80         4 : 40       4 : 44        4 : 33           4 : 48
##  5 :130         5 : 30       5 : 34        5 : 23           5 : 39
##  6 : 34         6 : 27       6 : 30        6 : 22           6 : 41
##  7 : 23         7 : 19       7 : 30        7 : 13           7 : 12
##  8 : 46         8 : 29       8 : 28        8 : 25           8 : 21
##  9 : 14         9 :  6       9 :  7        9 :  5           9 :  2
##  10: 69         10: 67       10: 58        10: 55           10: 31
##
##  BareNuclei BlandChromatin NormalNucleoli Mitosis  Diagnosis
##  ? : 16     1 :152         1 :443         1 :579   2:458
##  1 :402     2 :166         2 : 36         2 : 35   4:241
##  10:132     3 :165         3 : 44         3 : 33
##  2 : 30     4 : 40         4 : 18         4 : 12
##  3 : 28     5 : 34         5 : 19         5 :  6
##  4 : 19     6 : 10         6 : 22         6 :  3
##  5 : 30     7 : 73         7 : 16         7 :  9
##  6 :  4     8 : 28         8 : 24         8 :  8
##  7 :  8     9 : 11         9 : 16         10: 14
##  8 : 21     10: 20         10: 61
##  9 :  9
```

```
# Clean missing data
bcw_data %>%
  select(BareNuclei, Diagnosis) %>%
  filter(BareNuclei == "?") %>%
  group_by(Diagnosis) %>%
  summarise(n())
```

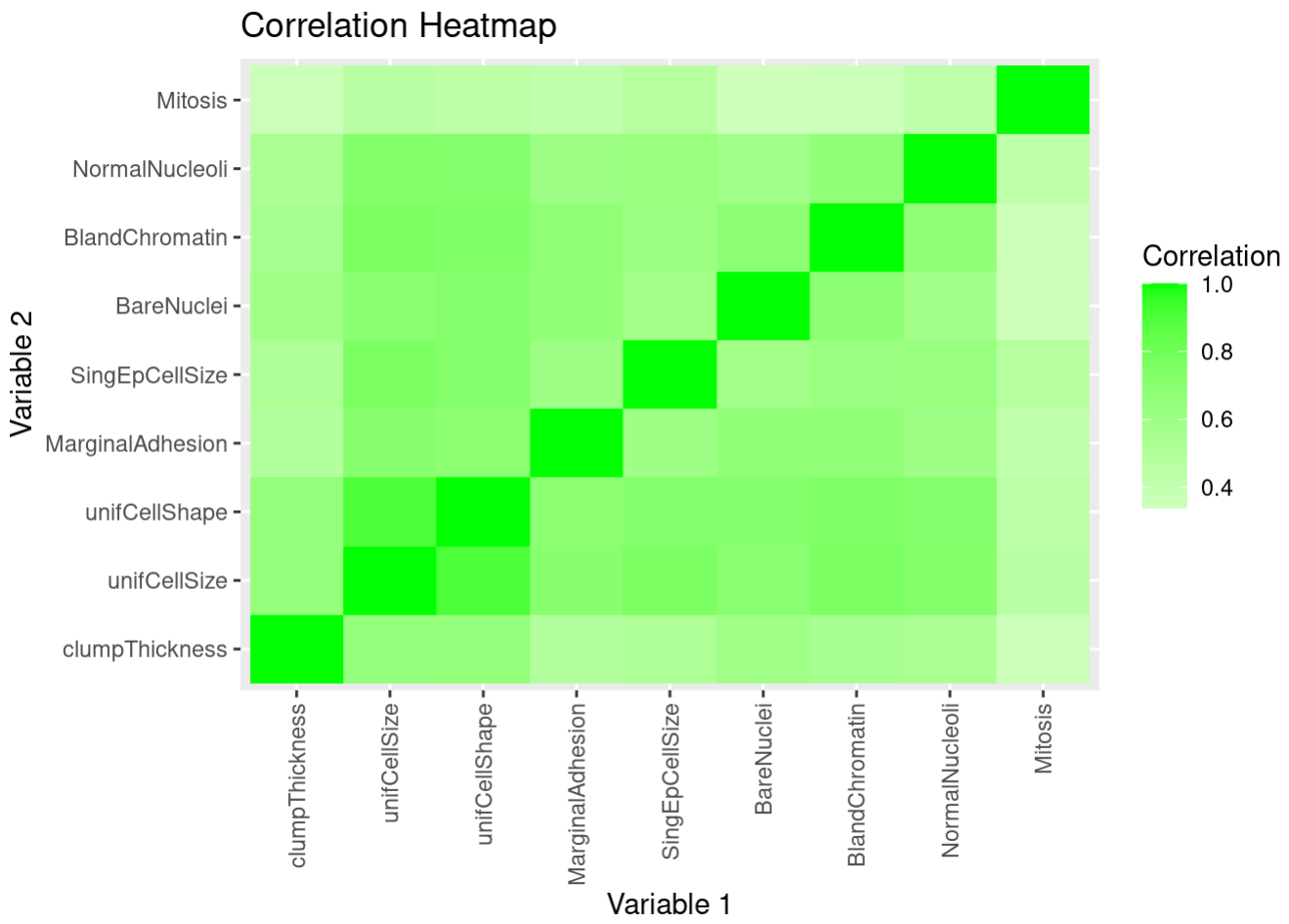
```
## # A tibble: 2 × 2
##   Diagnosis `n()`
##   <fct>     <int>
## 1 2            14
## 2 4             2
```

```
bcw_data %>%
  select(BareNuclei, Diagnosis) %>%
  group_by(BareNuclei, Diagnosis) %>%
  summarise(n())
```

```
## `summarise()` has grouped output by 'BareNuclei'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 20 × 3
## # Groups:   BareNuclei [11]
##    BareNuclei Diagnosis `n()`
##    <fct>      <fct>     <int>
##  1 ?          2            14
##  2 ?          4             2
##  3 1          2           387
##  4 1          4            15
##  5 10         2             3
##  6 10         4           129
##  7 2          2            21
##  8 2          4             9
##  9 3          2            14
## 10 3          4            14
## 11 4          2             6
## 12 4          4            13
## 13 5          2            10
## 14 5          4            20
## 15 6          4             4
## 16 7          2             1
## 17 7          4             7
## 18 8          2             2
## 19 8          4            19
## 20 9          4             9
```

```
bcw_data[bcw_data$BareNuclei=="?" & bcw_data$Diagnosis==2, 7] = 1
bcw_data[bcw_data$BareNuclei=="?" & bcw_data$Diagnosis==4, 7] = 10

# Convert columns to numeric and class to factor
for(i in 1:(ncol(bcw_data) - 1)) {
  bcw_data[, i] = as.numeric(as.character(bcw_data[, i]))
}

bcw_data[, 11] = as.factor(bcw_data[, 11])

# Write cleaned data to CSV file
write.csv(bcw_data, 'cleaneddata.csv', row.names = FALSE);

# Compute correlation matrix
cor_matrix <- cor(bcw_data[,2:10])

# Plot heatmap of correlation matrix
library(ggplot2)
library(reshape2)

cor_melted <- melt(cor_matrix)
names(cor_melted) <- c("Variable 1", "Variable 2", "Correlation")
ggplot(cor_melted, aes(x = `Variable 1`, y = `Variable 2`, fill = `Correlation`)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "green", midpoint = 0) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  labs(title = "Correlation Heatmap")
```



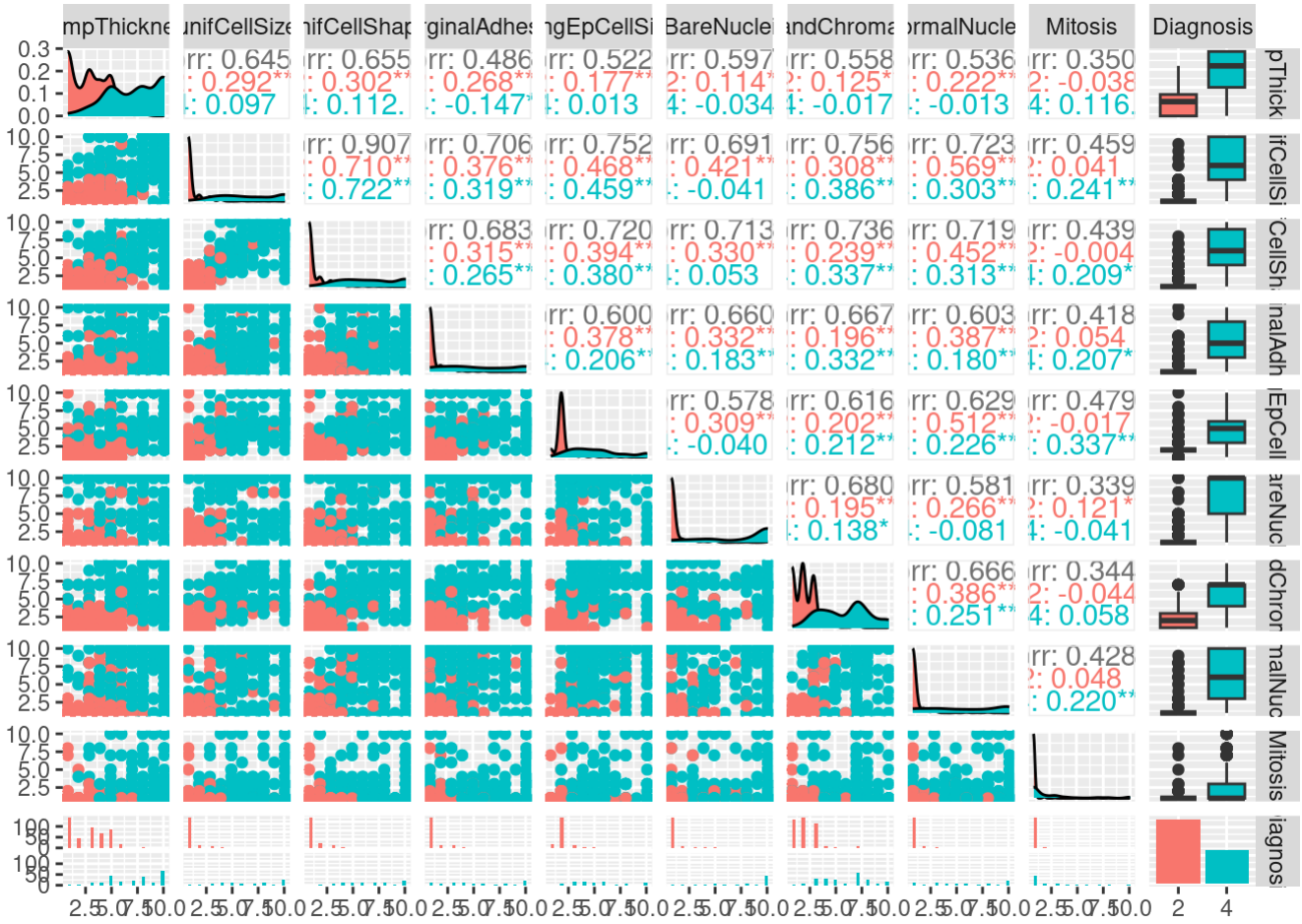

```
# Scatterplot matrix
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
ggpairs(bcw_data[, 2:11], mapping = aes(color = Diagnosis))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Regression analysis
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked _by_ '.GlobalEnv':
##
##     precision, recall
```

```
# Load cleaned data
bcw_data <- read.csv("cleaneddata.csv")
bcw_data$Diagnosis <- factor(bcw_data$Diagnosis, levels = c(2, 4), labels = c("benign", "maligna
nt"))


# Check data structure
str(bcw_data)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ id              : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 103
3078 1033078 ...
##  $ clumpThickness  : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ unifCellSize    : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ unifCellShape   : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ MarginalAdhesion: int  1 5 1 1 3 8 1 1 1 1 ...
##  $ SingEpCellSize  : int  2 7 2 3 2 7 2 2 2 2 ...
##  $ BareNuclei      : int  1 10 2 4 1 10 10 1 1 1 ...
##  $ BlandChromatin  : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ NormalNucleoli  : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ Mitosis         : int  1 1 1 1 1 1 1 1 5 1 ...
##  $ Diagnosis       : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

```
# Split data into training and testing sets
trainIndex <- createDataPartition(bcw_data$Diagnosis, p = 0.8, list = FALSE, times = 1)
train <- bcw_data[trainIndex,]
test <- bcw_data[-trainIndex,]

# Fit logistic regression model
logistic_model <- train(Diagnosis ~ ., data = train, method = "glm", family = "binomial")
```

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

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

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

```
# Check model performance on training data
logistic_preds_train <- predict(logistic_model, newdata = train)
confusionMatrix(logistic_preds_train, train$Diagnosis)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       358        10
##   malignant      9       183
##
##                Accuracy : 0.9661
##                  95% CI : (0.9475, 0.9795)
##     No Information Rate : 0.6554
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9248
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9755
##             Specificity : 0.9482
##          Pos Pred Value : 0.9728
##          Neg Pred Value : 0.9531
##              Prevalence : 0.6554
##          Detection Rate : 0.6393
##    Detection Prevalence : 0.6571
##       Balanced Accuracy : 0.9618
##
##        'Positive' Class : benign
##
```

```
# Predict on testing set
logistic_preds_test <- predict(logistic_model, newdata = test)

# Evaluate model performance
cm.log <- confusionMatrix(logistic_preds_test, test$Diagnosis)
```

Sampling for Data separation

```
set.seed(1)

partitions = sample(nrow(bcw_data)) %% 10

table(partitions)
```

```
## partitions
##  0  1  2  3  4  5  6  7  8  9
## 69 70 70 70 70 70 70 70 70 70
```

10-Fold Cross Validation

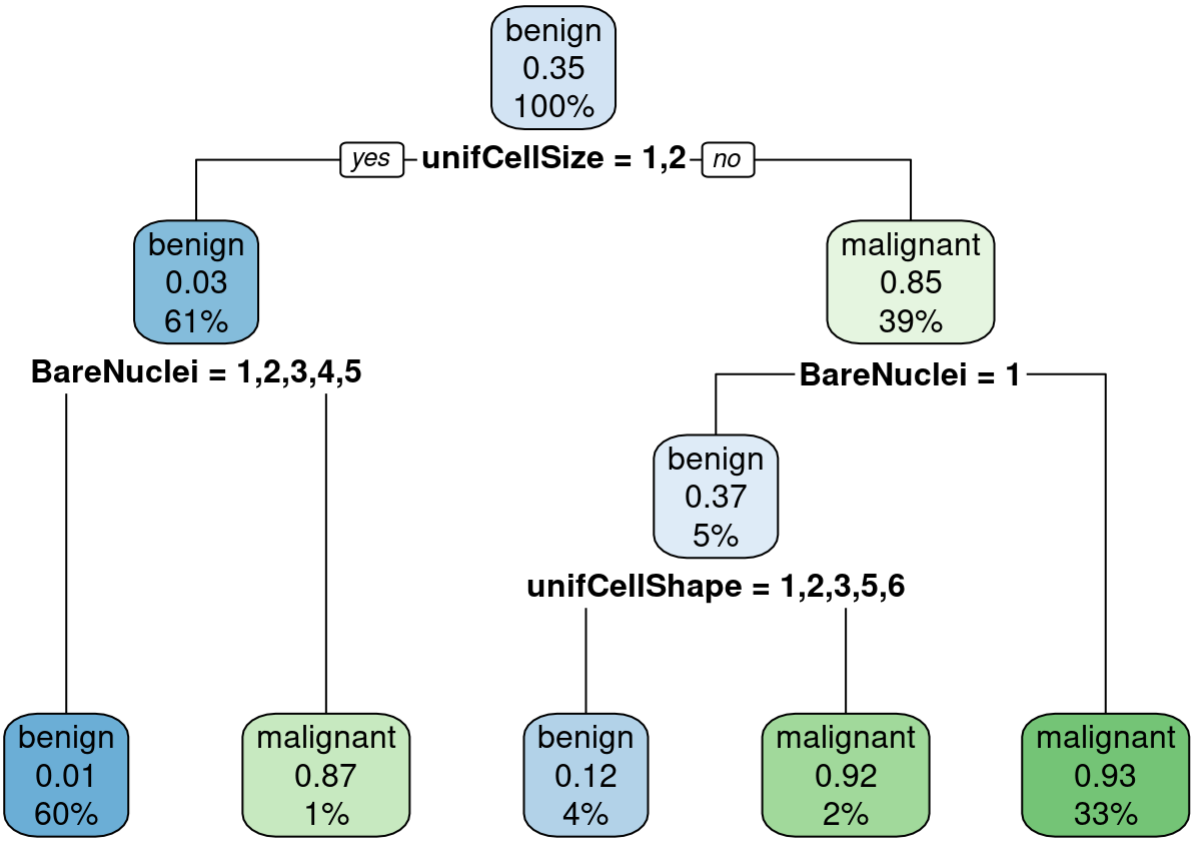
```
### Include required libraries

library(rpart.plot)
```

```
## Loading required package: rpart
```

```
library(e1071)
library(nnet)
library(fpc)
library(NeuralNetTools)

tbl = matrix(c(0,0,0,0), nrow = 2, byrow = TRUE)
colnames(tbl) = c("actual(0)","actual(1)")
rownames(tbl) = c("prediction(0)", "prediction(1)")


### 10-Fold CV with Decision tree

tree_data = bcw_data[-1]


### change data for decision tree

for (i in 1:ncol(tree_data)) {
  tree_data[ , i] = as.factor(tree_data[ , i])
}

summary(tree_data, maxsum = 20)
```

```
##  clumpThickness unifCellSize unifCellShape MarginalAdhesion SingEpCellSize
##  1 :145         1 :384       1 :353        1 :407           1 : 47
##  2 : 50         2 : 45       2 : 59        2 : 58           2 :386
##  3 :108         3 : 52       3 : 56        3 : 58           3 : 72
##  4 : 80         4 : 40       4 : 44        4 : 33           4 : 48
##  5 :130         5 : 30       5 : 34        5 : 23           5 : 39
##  6 : 34         6 : 27       6 : 30        6 : 22           6 : 41
##  7 : 23         7 : 19       7 : 30        7 : 13           7 : 12
##  8 : 46         8 : 29       8 : 28        8 : 25           8 : 21
##  9 : 14         9 :  6       9 :  7        9 :  5           9 :  2
##  10: 69         10: 67       10: 58        10: 55           10: 31
##  BareNuclei BlandChromatin NormalNucleoli Mitosis      Diagnosis
##  1 :416     1 :152         1 :443         1 :579   benign   :458
##  2 : 30     2 :166         2 : 36         2 : 35   malignant:241
##  3 : 28     3 :165         3 : 44         3 : 33
##  4 : 19     4 : 40         4 : 18         4 : 12
##  5 : 30     5 : 34         5 : 19         5 :  6
##  6 :  4     6 : 10         6 : 22         6 :  3
##  7 :  8     7 : 73         7 : 16         7 :  9
##  8 : 21     8 : 28         8 : 24         8 :  8
##  9 :  9     9 : 11         9 : 16         10: 14
##  10:134     10: 20         10: 61
```

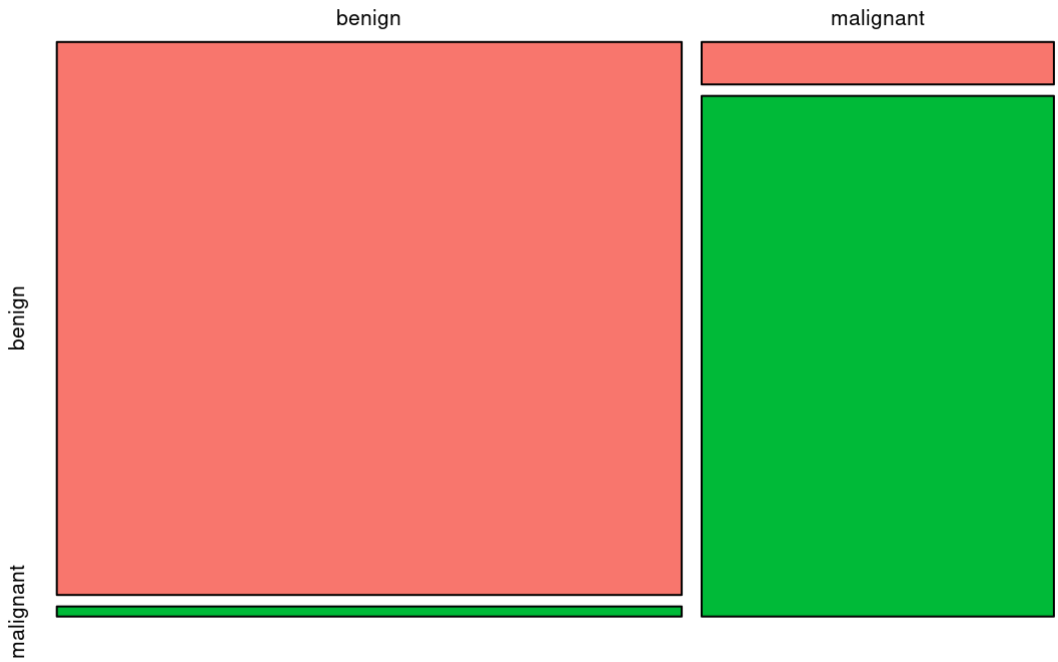
```
cm.decision_tree = tbl

for (i in 0:9){

  ### Split data

  ind = partitions == i
  test_data = tree_data[ind, ]
  train_data = tree_data[-ind, ]


  ### Model for Decision Tree

  set.seed(1)

  model.decision_tree = rpart(Diagnosis~., data = train_data)

  pred = predict(model.decision_tree, select(test_data, -Diagnosis), type = "class")

  cm.decision_tree = cm.decision_tree + table(pred, test_data$Diagnosis)

}

rpart.plot(model.decision_tree)
```




```
### Confusion matrix

colnames(cm.decision_tree) = c('benign', 'malignant')
rownames(cm.decision_tree) = c('benign', 'malignant')

cm.decision_tree
```

```
##           benign malignant
## benign       439         8
## malignant     19       233
```

```
plot(cm.decision_tree, main = "Decision Tree Confusion Matrix", col = c("#F8766D", "#00BA38"),
     sub = paste("Accuracy =", round(accuracy(cm.decision_tree), 3)))
```




```
### Accuracy of Decision Tree

print(c("Precision:", precision(cm.decision_tree)))
```

```
## [1] "Precision:"       "92.4603174603175"
```

```
print(c("Recall:", recall(cm.decision_tree)))
```

```
## [1] "Recall:"          "96.6804979253112"
```

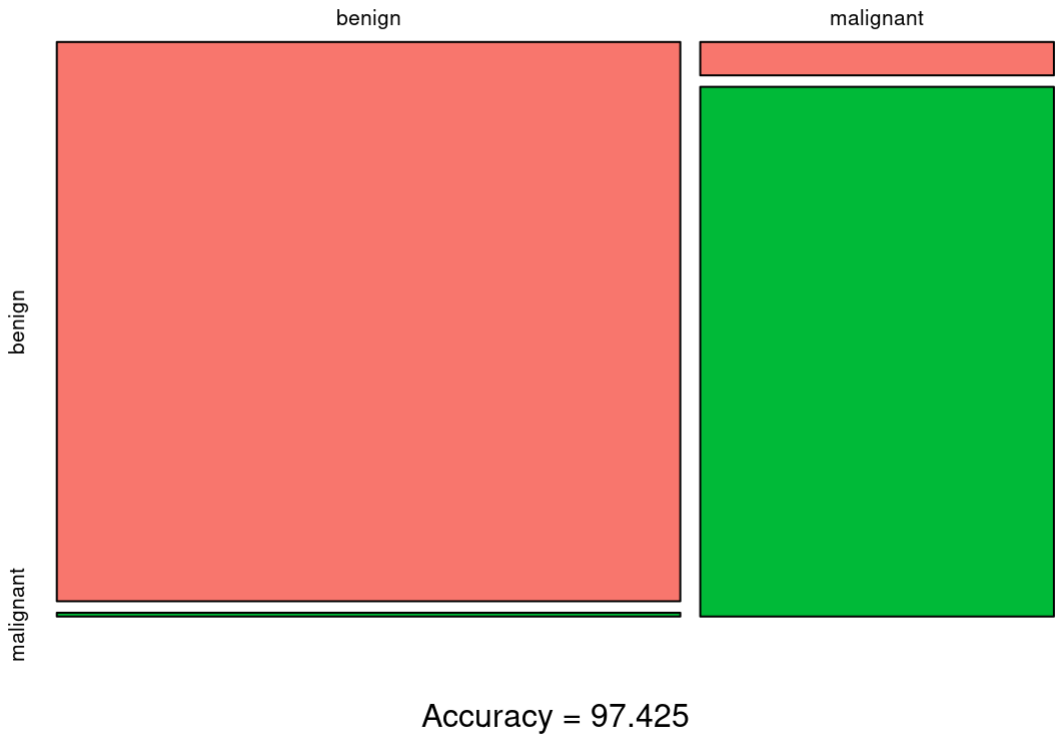
```
print(c("Accuracy:", accuracy(cm.decision_tree)))
```

```
## [1] "Accuracy:"       "96.137339055794"
```

```
### 10-Fold CV with Naive Bayes

nv_data = tree_data

cm.naive_bayes = tbl

for (i in 0:9) {

  ### Split data

  ind = partitions == i
  test_data = nv_data[ind, ]
  train_data = nv_data[-ind, ]


  ### Model for Naive Bayes

  set.seed(1)

  model.naive_bayes = naiveBayes(Diagnosis~., data = train_data)

  pred = predict(model.naive_bayes, select(test_data, -Diagnosis))

  cm.naive_bayes = cm.naive_bayes + table(pred, test_data$Diagnosis)

}


### Confusion matrix

colnames(cm.naive_bayes) = c('benign', 'malignant')
rownames(cm.naive_bayes) = c('benign', 'malignant')

cm.naive_bayes
```

```
##           benign malignant
## benign       443         3
## malignant     15       238
```

```
plot(cm.naive_bayes, main = "Naive Bayes Confusion Matrix", col = c("#F8766D", "#00BA38"),
     sub = paste("Accuracy =", round(accuracy(cm.naive_bayes), 3)))
```




```
### Accuracy of Naive Byes

print(c("Precision:", precision(cm.naive_bayes)))
```

```
## [1] "Precision:"       "94.0711462450593"
```

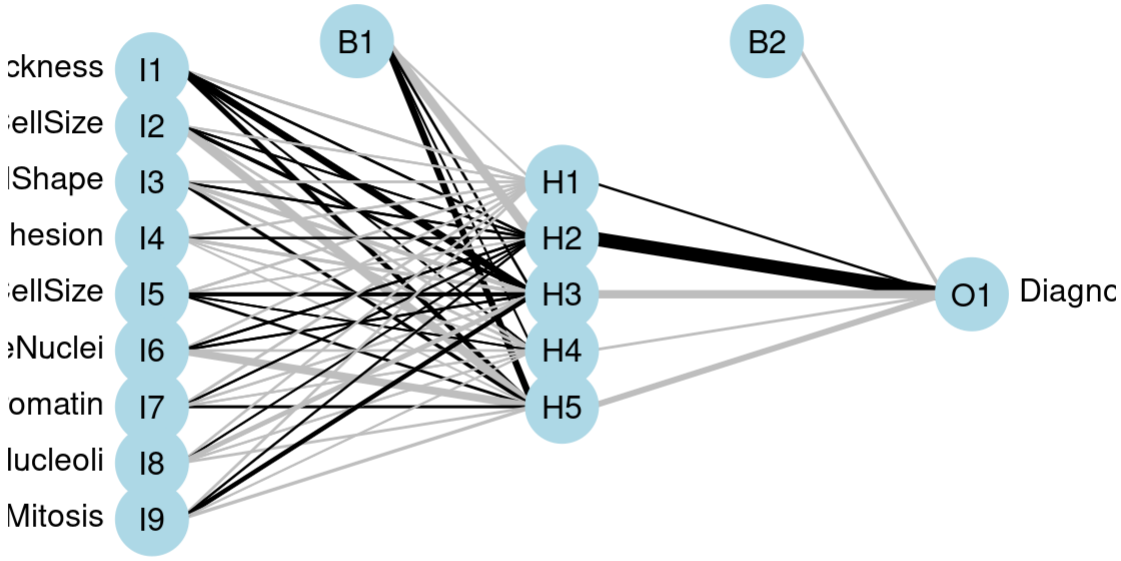
```
print(c("Recall:", recall(cm.naive_bayes)))
```

```
## [1] "Recall:"          "98.7551867219917"
```

```
print(c("Accuracy:", accuracy(cm.naive_bayes)))
```

```
## [1] "Accuracy:"        "97.4248927038627"
```

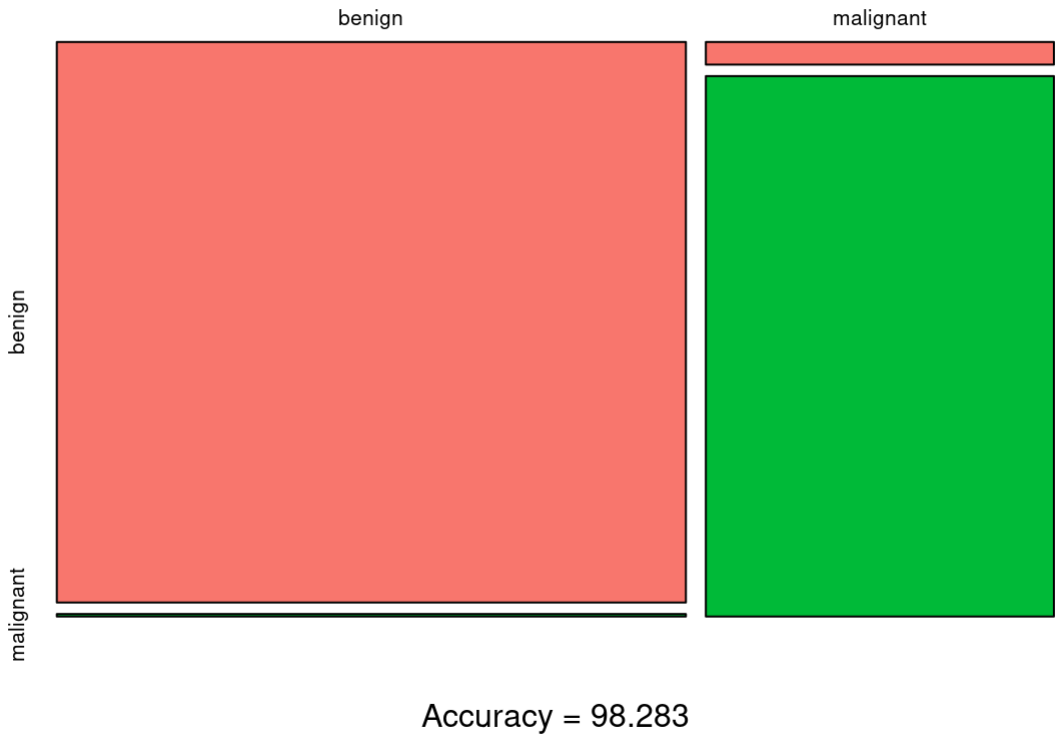
```
### 10-Fold CV with ANN

ann_data = bcw_data[-1]

ann_data[, 10] = as.numeric(ann_data[, 10])
ann_data[ann_data$Diagnosis == 1 , 10] = 0
ann_data[ann_data$Diagnosis == 2 , 10] = 1

cm.ann = tbl

for (i in 0:9) {

  ### Split data

  ind = partitions == i
  test_data = ann_data[ind, ]
  train_data = ann_data[-ind, ]

  ### Model for ANN

  set.seed(1)

  model.ann = nnet(Diagnosis~., train_data, size = 5, type = "class", trace = FALSE, wgts = 0.1)

  pred = round(predict(model.ann, select(test_data, -Diagnosis)))

  cm.ann = cm.ann + table(pred, test_data$Diagnosis)

}

plotnet(model.ann)
```




```
### Confusion matrix

colnames(cm.ann) = c('benign', 'malignant')
rownames(cm.ann) = c('benign', 'malignant')

cm.ann
```

```
##           benign malignant
## benign       448         2
## malignant     10       239
```

```
plot(cm.ann, main = "ANN Confusion Matrix", col = c("#F8766D", "#00BA38"),
     sub = paste("Accuracy =", round(accuracy(cm.ann), 3)))
```




```
### Accuracy of ANN

print(c("Precision:", precision(cm.ann)))
```

```
## [1] "Precision:"       "95.9839357429719"
```

```
print(c("Recall:", recall(cm.ann)))
```

```
## [1] "Recall:"          "99.1701244813278"
```

```
print(c("Accuracy:", accuracy(cm.ann)))
```

```
## [1] "Accuracy:"        "98.2832618025751"
```

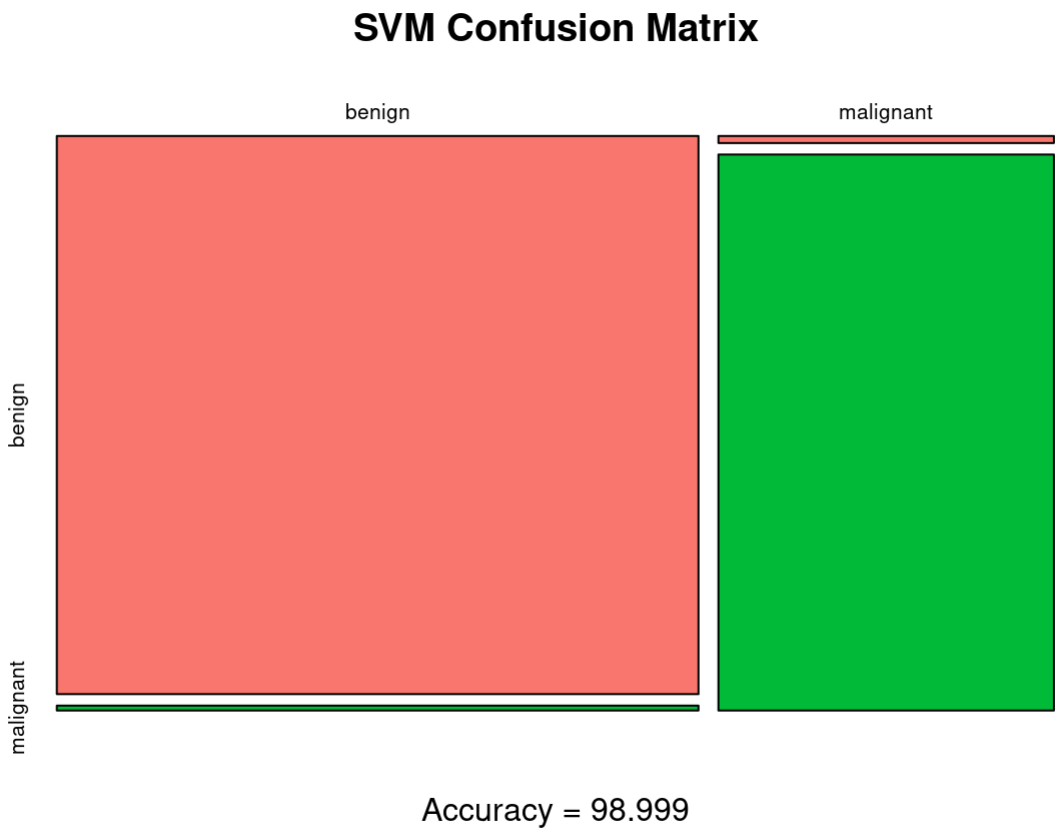
```
### 10-Fold CV with Support Vector Machine

svm_data = tree_data

cm.svm = tbl

for (i in 0:9) {

  ### Split data

  ind = partitions == i
  test_data = svm_data[ind, ]
  train_data = svm_data[-ind, ]

  ### Model for SVM

  set.seed(1)

  model.svm = svm(Diagnosis~., data = train_data, kernel = "linear", scale = FALSE)

  pred = predict(model.svm, select(test_data, -Diagnosis))

  cm.svm = cm.svm + table(pred, test_data$Diagnosis)

}

### SVM model

model.svm
```

```
##
## Call:
## svm(formula = Diagnosis ~ ., data = train_data, kernel = "linear",
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  78
```

```
### Confusion matrix

colnames(cm.svm) = c('benign', 'malignant')
rownames(cm.svm) = c('benign', 'malignant')

cm.svm
```

```
##           benign malignant
## benign       455         4
## malignant      3       237
```

```
plot(cm.svm, main = "SVM Confusion Matrix", col = c("#F8766D", "#00BA38"),
     sub = paste("Accuracy =", round(accuracy(cm.svm), 3)))
```




```
### Accuracy of Support Vector Machine

print(c("Precision:", precision(cm.svm)))
```

```
## [1] "Precision:" "98.75"
```

```
print(c("Recall:", recall(cm.svm)))
```

```
## [1] "Recall:"          "98.3402489626556"
```

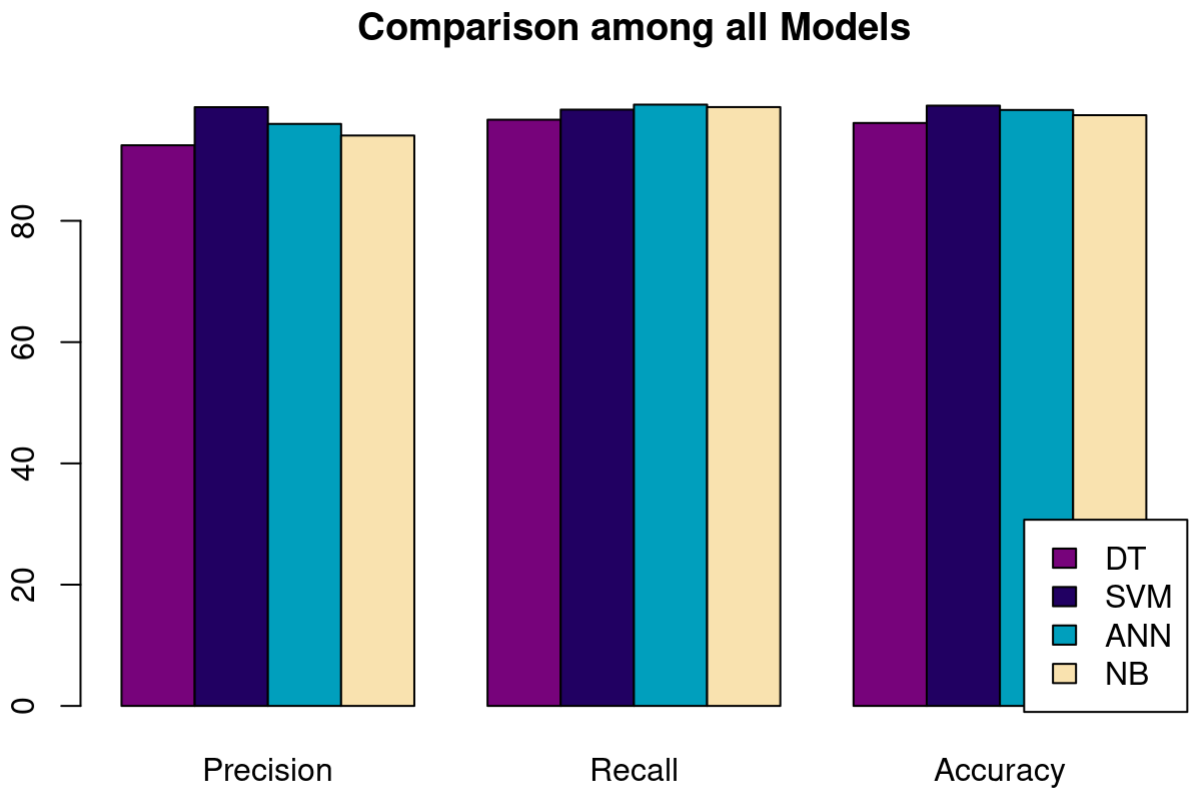
```
print(c("Accuracy:", accuracy(cm.svm)))
```

```
## [1] "Accuracy:"        "98.9985693848355"
```

```
### Comparisions

rbind(c("Model", "Accuracy(%)"),c("Decision Tree", accuracy(cm.decision_tree)), c("Naive Bayes",
accuracy(cm.naive_bayes)), c("ANN", accuracy(cm.ann)), c("SVM", accuracy(cm.svm)))
```

```
##      [,1]            [,2]
## [1,] "Model"         "Accuracy(%)"
## [2,] "Decision Tree" "96.137339055794"
## [3,] "Naive Bayes"   "97.4248927038627"
## [4,] "ANN"           "98.2832618025751"
## [5,] "SVM"           "98.9985693848355"
```

```
sbs_barplot = matrix(nrow = 4, ncol = 3, dimnames = list(c('DT', 'SVM', 'ANN', 'NB'), c('Precisi
on', 'Recall', 'Accuracy')))

sbs_barplot['DT','Precision'] = precision(cm.decision_tree)
sbs_barplot['DT','Recall'] = recall(cm.decision_tree)
sbs_barplot['DT','Accuracy'] = accuracy(cm.decision_tree)

sbs_barplot['SVM','Precision'] = precision(cm.svm)
sbs_barplot['SVM','Recall'] = recall(cm.svm)
sbs_barplot['SVM','Accuracy'] = accuracy(cm.svm)

sbs_barplot['ANN','Precision'] = precision(cm.ann)
sbs_barplot['ANN','Recall'] = recall(cm.ann)
sbs_barplot['ANN','Accuracy'] = accuracy(cm.ann)

sbs_barplot['NB','Precision'] = precision(cm.naive_bayes)
sbs_barplot['NB','Recall'] = recall(cm.naive_bayes)
sbs_barplot['NB','Accuracy'] = accuracy(cm.naive_bayes)

#png("plot.png", width=900, height=600)

barplot(sbs_barplot, main = "Comparison among all Models",beside = TRUE,
        col=c('#77037B', '#210062', '#009FBD', '#F9E2AF'),
        legend.text = rownames(sbs_barplot),
        args.legend = list(x = "bottomright"))
```


Comparison among all Models
0
20
40
60
80
Precision
Recall
Accuracy
DT
SVM
ANN
NB