

Term Project - Predictive Modeling

Loding important packages

```
library(readr, warn.conflicts=F)
library(RColorBrewer, warn.conflicts=F) #Rcolorbrewer palette
library(corrplot, warn.conflicts=F)

## corrplot 0.84 loaded

library(ggcorrplot, warn.conflicts=F)

## Loading required package: ggplot2

library(plotly, warn.conflicts=F)
library(ggplot2, warn.conflicts=F)
library(reshape, warn.conflicts=F)
library(viridis, warn.conflicts=F)

## Loading required package: viridisLite

library(tidyverse, warn.conflicts=F)

## -- Attaching packages ----- tidyverse 1.3.0 --

## v tibble 3.1.0     v dplyr  1.0.5
## v tidyr  1.1.2     v stringr 1.4.0
## v purrr   0.3.4    vforcats 0.5.0

## -- Conflicts ----- tidyverse_conflicts() --
## x tidyrr::expand() masks reshape::expand()
## x dplyr::filter() masks plotly::filter(), stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::rename() masks reshape::rename(), plotly::rename()

library(hrbrthemes, warn.conflicts=F)

## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.

##      Please use hrbrthemes::importRobotoCondensed() to install Roboto Condensed and

##      if Arial Narrow is not on your system, please see https://bit.ly/arialnarrow
```

```

library(psych, warn.conflicts=F)
library(class, warn.conflicts=F)
library(caret, warn.conflicts = F)

## Loading required package: lattice

library(DescTools)

##
## Attaching package: 'DescTools'

## The following objects are masked from 'package:caret':
## 
##     MAE, RMSE

## The following objects are masked from 'package:psych':
## 
##     AUC, ICC, SD

library(sjPlot)

```

```

## #refugeeswelcome

set.seed(123456789)

```

Importing Dataset

```

data <- read.csv("~/Breast-Cancer-Prediction/data/data.csv", header=T)
cancerData <- read.csv("~/Breast-Cancer-Prediction/data/data.csv", header=T)

```

Looking at dataset

```

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 X
## 1 0.2654       0.4601     0.11890    NA
## 2 0.1860       0.2750     0.08902    NA
## 3 0.2430       0.3613     0.08758    NA
## 4 0.2575       0.6638     0.17300    NA
## 5 0.1625       0.2364     0.07678    NA
## 6 0.1741       0.3985     0.12440    NA

```

Columns in dataset

```
colnames(data)
```

```

## [1] "id"                      "diagnosis"
## [3] "radius_mean"              "texture_mean"
## [5] "perimeter_mean"           "area_mean"
## [7] "smoothness_mean"           "compactness_mean"
## [9] "concavity_mean"            "concave.points_mean"
## [11] "symmetry_mean"             "fractal_dimension_mean"
## [13] "radius_se"                 "texture_se"
## [15] "perimeter_se"              "area_se"
## [17] "smoothness_se"              "compactness_se"
## [19] "concavity_se"               "concave.points_se"
## [21] "symmetry_se"                "fractal_dimension_se"
## [23] "radius_worst"               "texture_worst"
## [25] "perimeter_worst"             "area_worst"
## [27] "smoothness_worst"            "compactness_worst"

```

```
## [29] "concavity_worst"           "concave.points_worst"  
## [31] "symmetry_worst"             "fractal_dimension_worst"  
## [33] "X"
```

Checking for null values

```
lapply(data,function(x) { length(which(is.na(x)))})
```

```
## $id  
## [1] 0  
##  
## $diagnosis  
## [1] 0  
##  
## $radius_mean  
## [1] 0  
##  
## $texture_mean  
## [1] 0  
##  
## $perimeter_mean  
## [1] 0  
##  
## $area_mean  
## [1] 0  
##  
## $smoothness_mean  
## [1] 0  
##  
## $compactness_mean  
## [1] 0  
##  
## $concavity_mean  
## [1] 0  
##  
## $concave.points_mean  
## [1] 0  
##  
## $symmetry_mean  
## [1] 0  
##  
## $fractal_dimension_mean  
## [1] 0  
##  
## $radius_se  
## [1] 0  
##  
## $texture_se  
## [1] 0  
##  
## $perimeter_se  
## [1] 0  
##
```

```

## $area_se
## [1] 0
##
## $smoothness_se
## [1] 0
##
## $compactness_se
## [1] 0
##
## $concavity_se
## [1] 0
##
## $concave.points_se
## [1] 0
##
## $symmetry_se
## [1] 0
##
## $fractal_dimension_se
## [1] 0
##
## $radius_worst
## [1] 0
##
## $texture_worst
## [1] 0
##
## $perimeter_worst
## [1] 0
##
## $area_worst
## [1] 0
##
## $smoothness_worst
## [1] 0
##
## $compactness_worst
## [1] 0
##
## $concavity_worst
## [1] 0
##
## $concave.points_worst
## [1] 0
##
## $symmetry_worst
## [1] 0
##
## $fractal_dimension_worst
## [1] 0
##
## $X
## [1] 569

```

We can notice, that there seems to be three category in dataset. They're: mean, se and worst
DATA WRANGLING Deleting X column as it seems to be a mistake while importing the dataset

```
drops <- c("X")
data <- data[ , !(names(data) %in% drops)]

lapply(data,function(x) { length(which(is.na(x)))})

## $id
## [1] 0
##
## $diagnosis
## [1] 0
##
## $radius_mean
## [1] 0
##
## $texture_mean
## [1] 0
##
## $perimeter_mean
## [1] 0
##
## $area_mean
## [1] 0
##
## $smoothness_mean
## [1] 0
##
## $compactness_mean
## [1] 0
##
## $concavity_mean
## [1] 0
##
## $concave.points_mean
## [1] 0
##
## $symmetry_mean
## [1] 0
##
## $fractal_dimension_mean
## [1] 0
##
## $radius_se
## [1] 0
##
## $texture_se
## [1] 0
##
## $perimeter_se
## [1] 0
##
##
```

```

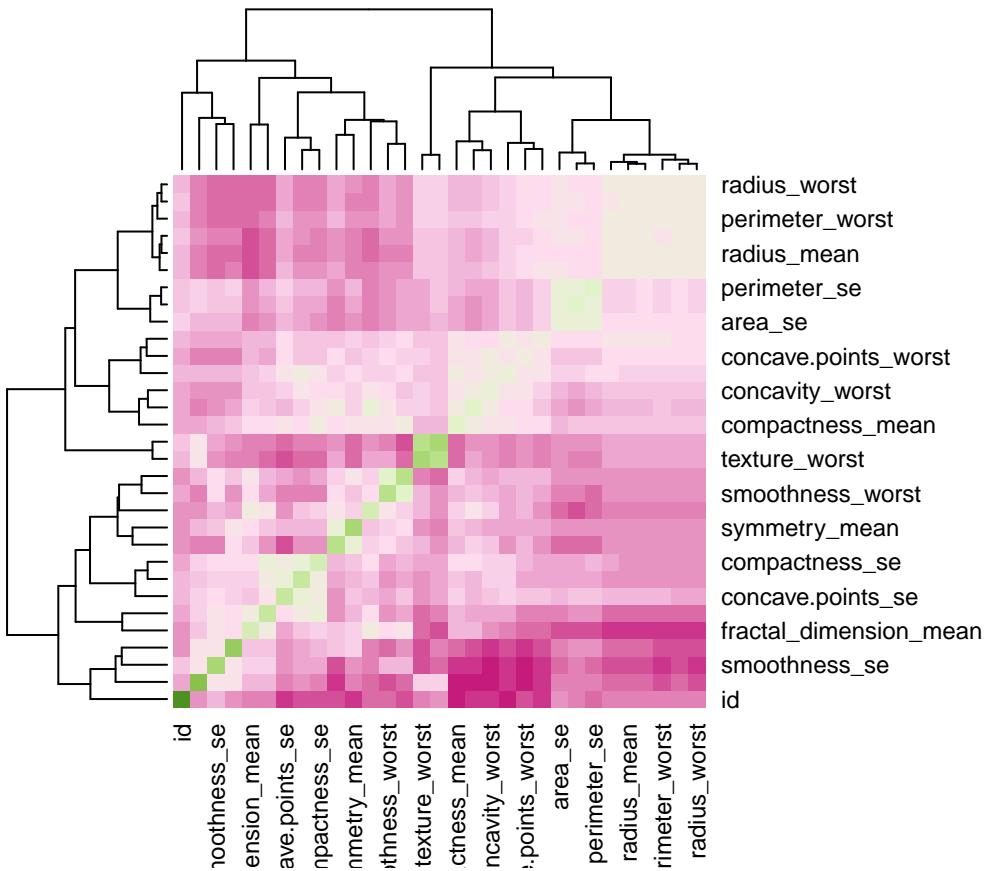
## $area_se
## [1] 0
##
## $smoothness_se
## [1] 0
##
## $compactness_se
## [1] 0
##
## $concavity_se
## [1] 0
##
## $concave.points_se
## [1] 0
##
## $symmetry_se
## [1] 0
##
## $fractal_dimension_se
## [1] 0
##
## $radius_worst
## [1] 0
##
## $texture_worst
## [1] 0
##
## $perimeter_worst
## [1] 0
##
## $area_worst
## [1] 0
##
## $smoothness_worst
## [1] 0
##
## $compactness_worst
## [1] 0
##
## $concavity_worst
## [1] 0
##
## $concave.points_worst
## [1] 0
##
## $symmetry_worst
## [1] 0
##
## $fractal_dimension_worst
## [1] 0

```

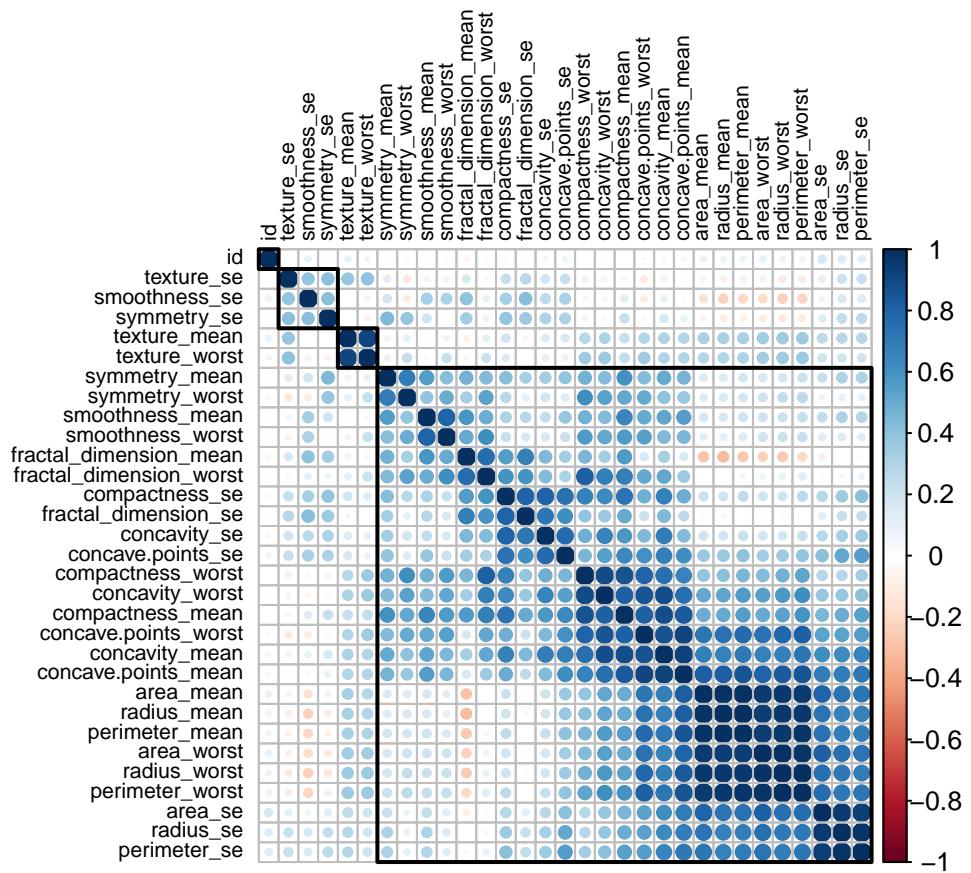
As we can notice now we do not have any missing data

Let's looking into correlation matrix to see correlation between all the variables

```
matrixData <- cor(data[sapply(data,is.numeric)], method="pearson")
# Rcolorbrewer palette
coul <- colorRampPalette(brewer.pal(8, "PiYG"))(25)
heatmap(matrixData, scale="column", col = coul)
```



```
corrplot(matrixData, tl.col = "black", order = "hclust", hclust.method = "average", addrect = 4, tl.cex
```

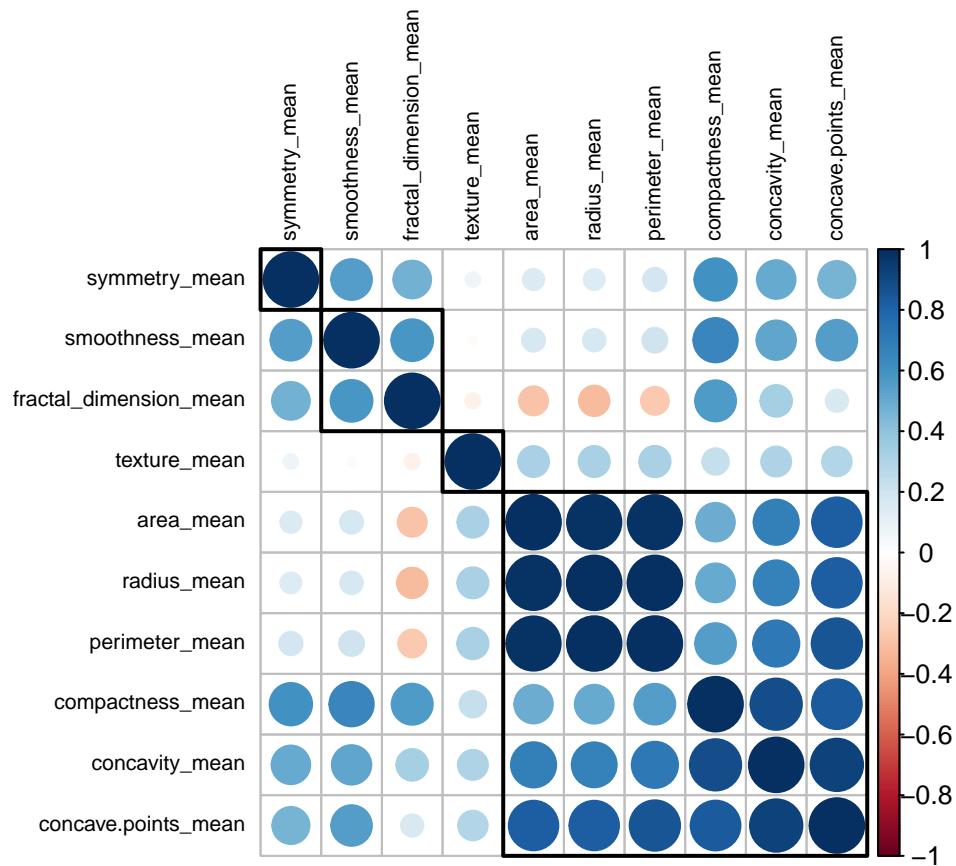


```

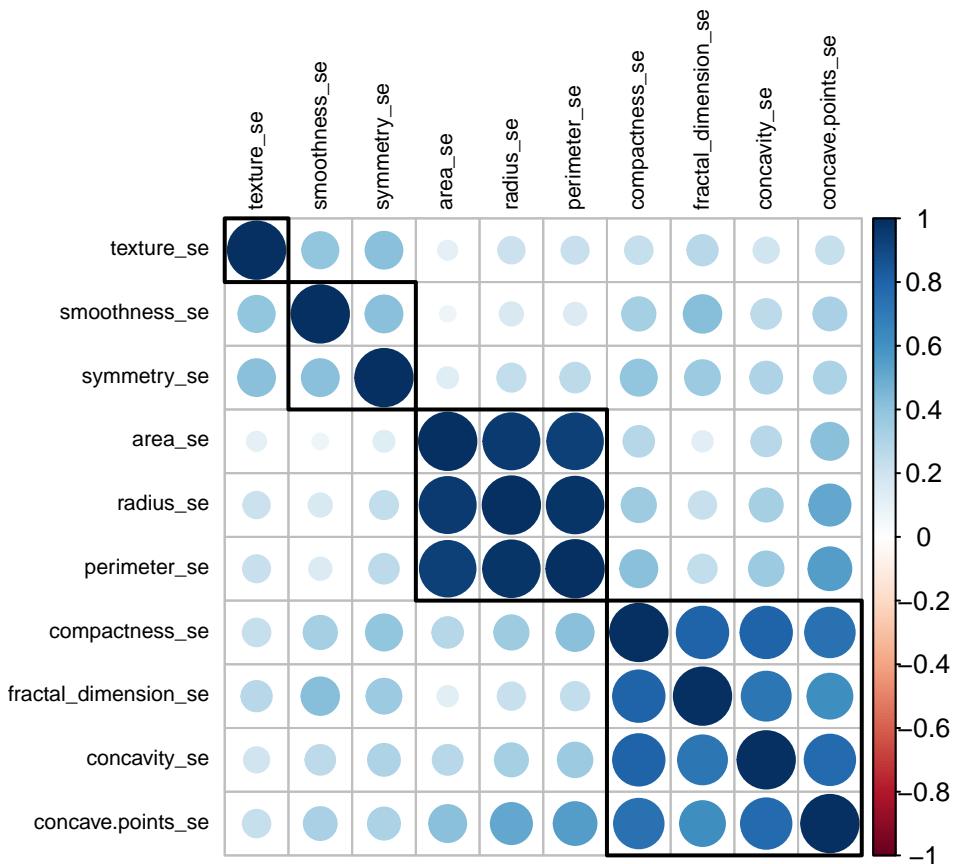
#data <- sapply(data, is.numeric)
data.mean <- cor(data[,c(3:12)], method="pearson")
data.se <- cor(data[,c(13:22)], method="pearson")
data.worst <- cor(data[,c(23:32)], method="pearson")

corrplot(data.mean, tl.col = "black", order = "hclust", hclust.method = "average", addrect = 4, tl.cex =

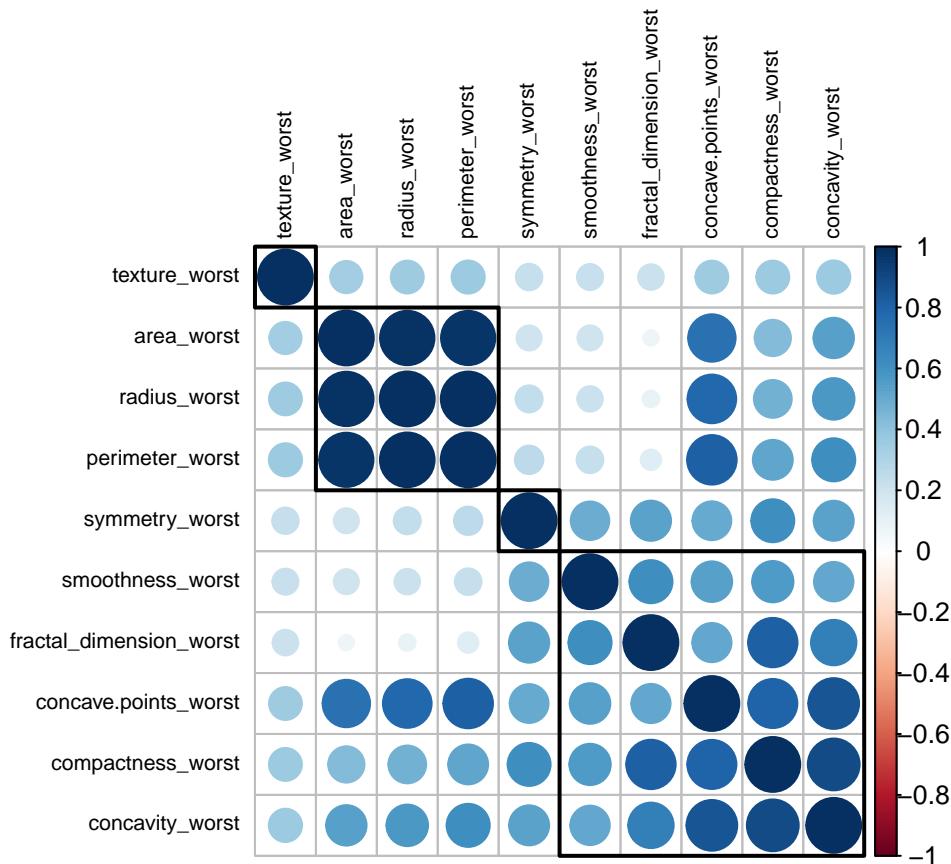
```



```
corrplot(data.se, tl.col = "black", order = "hclust", hclust.method = "average", addrect = 4, tl.cex = 1)
```



```
corrplot(data.worst, tl.col = "black", order = "hclust", hclust.method = "average", addrect = 4, tl.cex
```



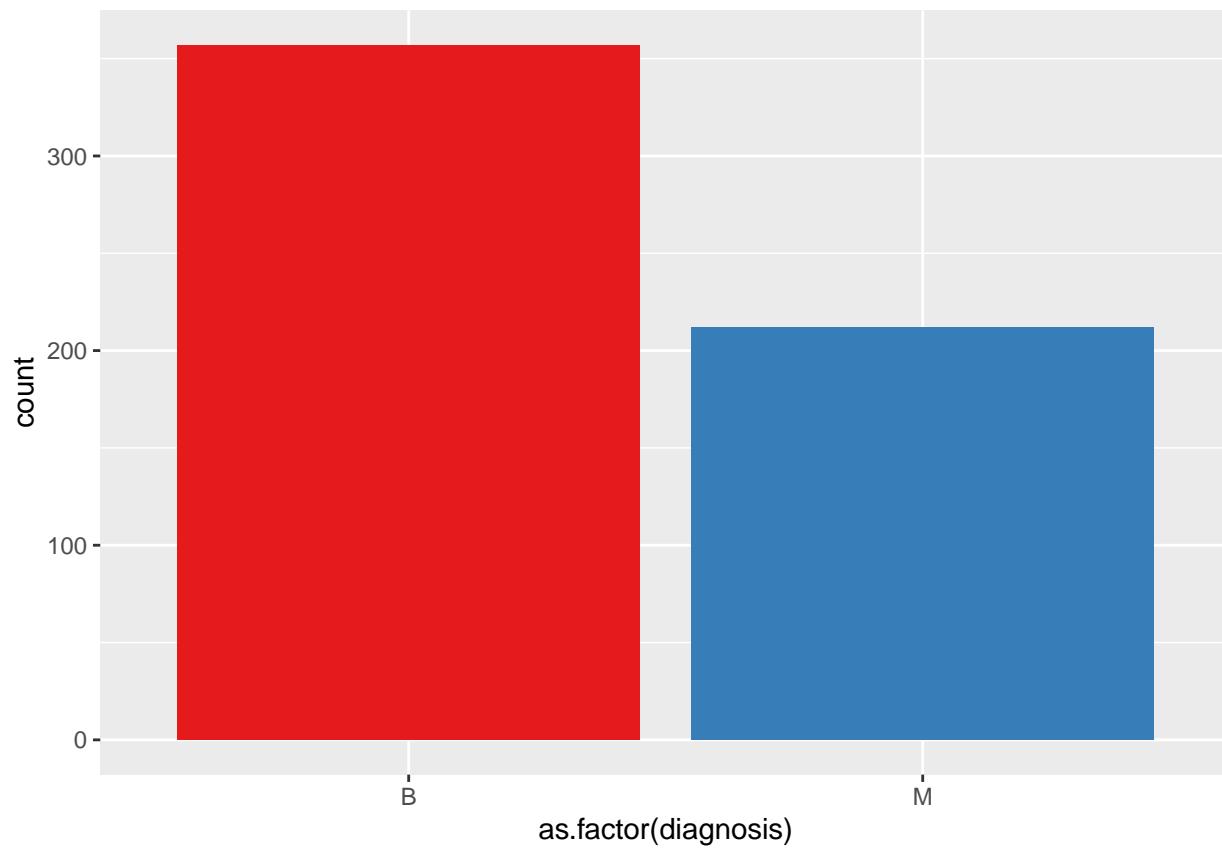
```
table(data$diagnosis)
```

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

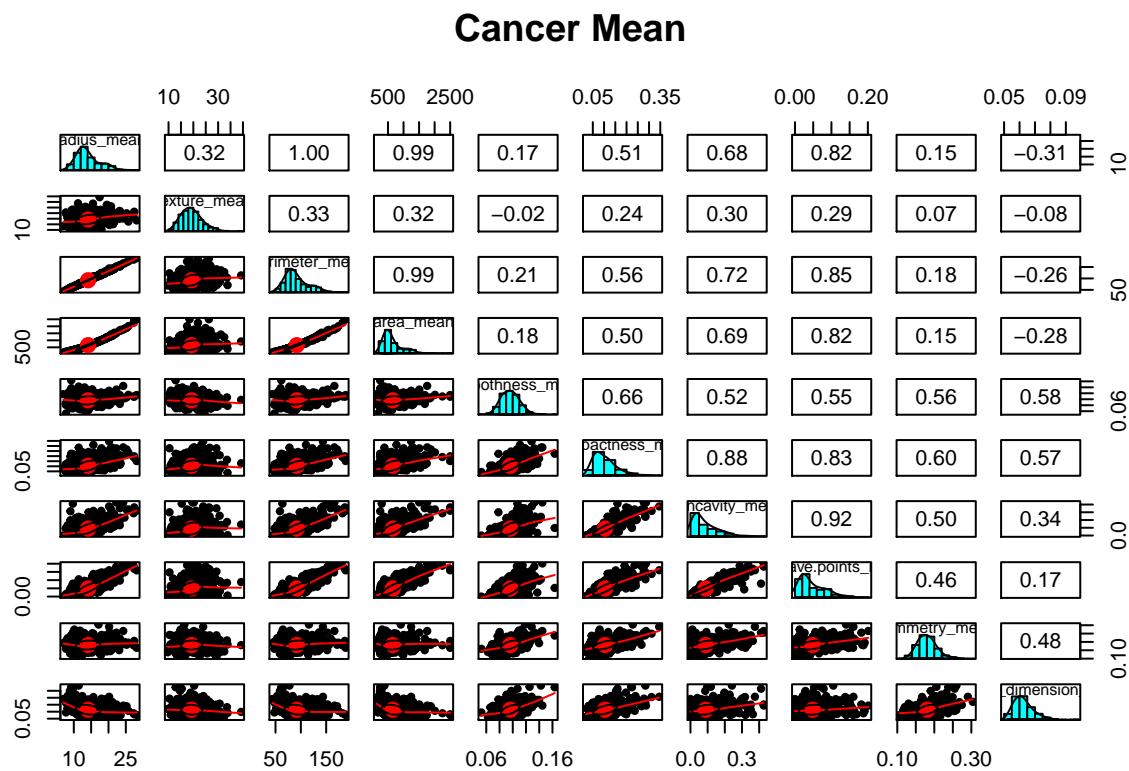
```
prop.table(table(data$diagnosis))*100
```

```
##  
##      B      M  
## 62.74165 37.25835
```

```
ggplot(data, aes(x=as.factor(diagnosis), fill=as.factor(diagnosis) )) +  
  geom_bar( ) +  
  scale_fill_brewer(palette = "Set1") +  
  theme(legend.position="none")
```

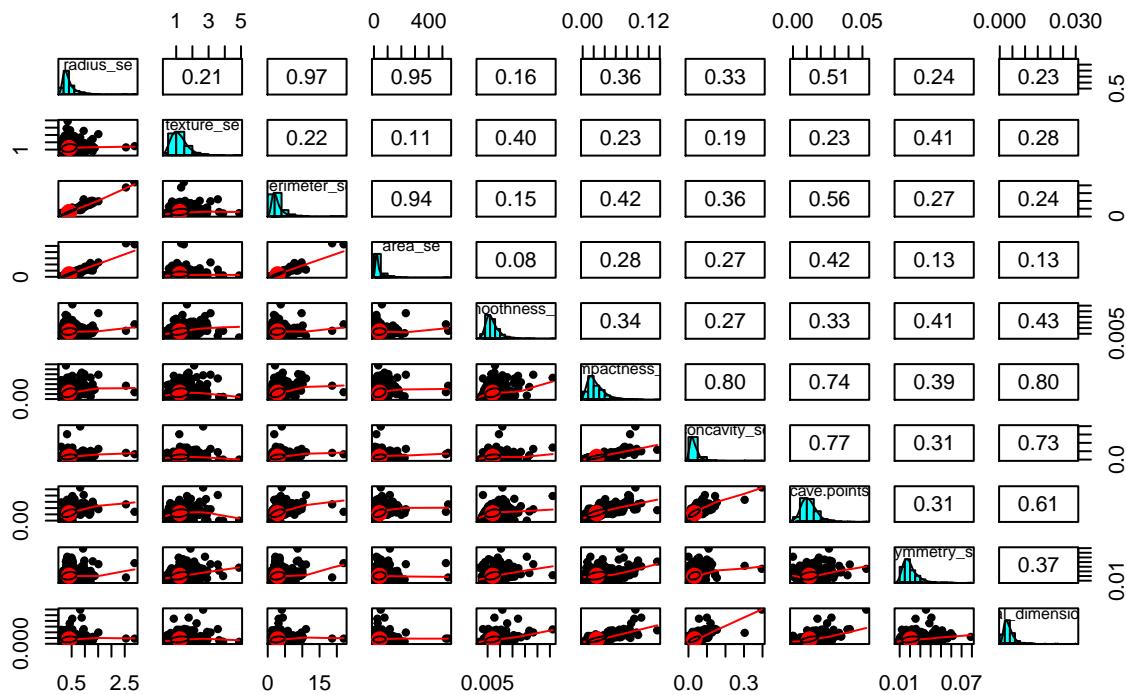


```
pairs.panels(data[,c(3:12)], main="Cancer Mean")
```



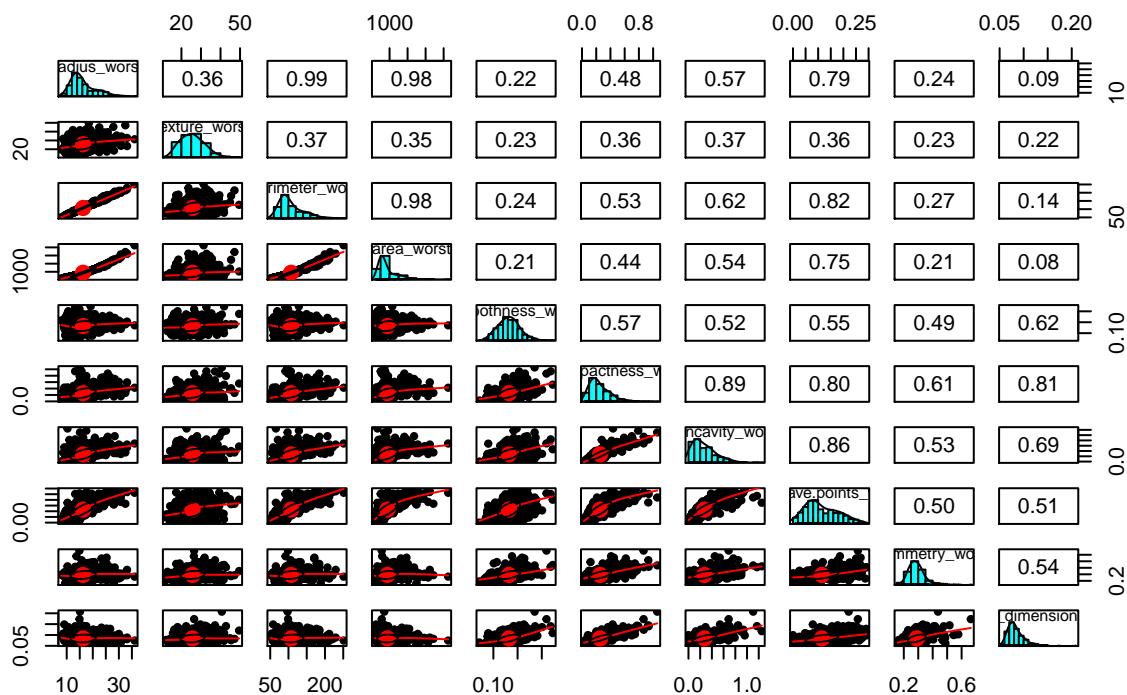
```
pairs.panels(data[,c(13:22)], main="Cancer SE")
```

Cancer SE

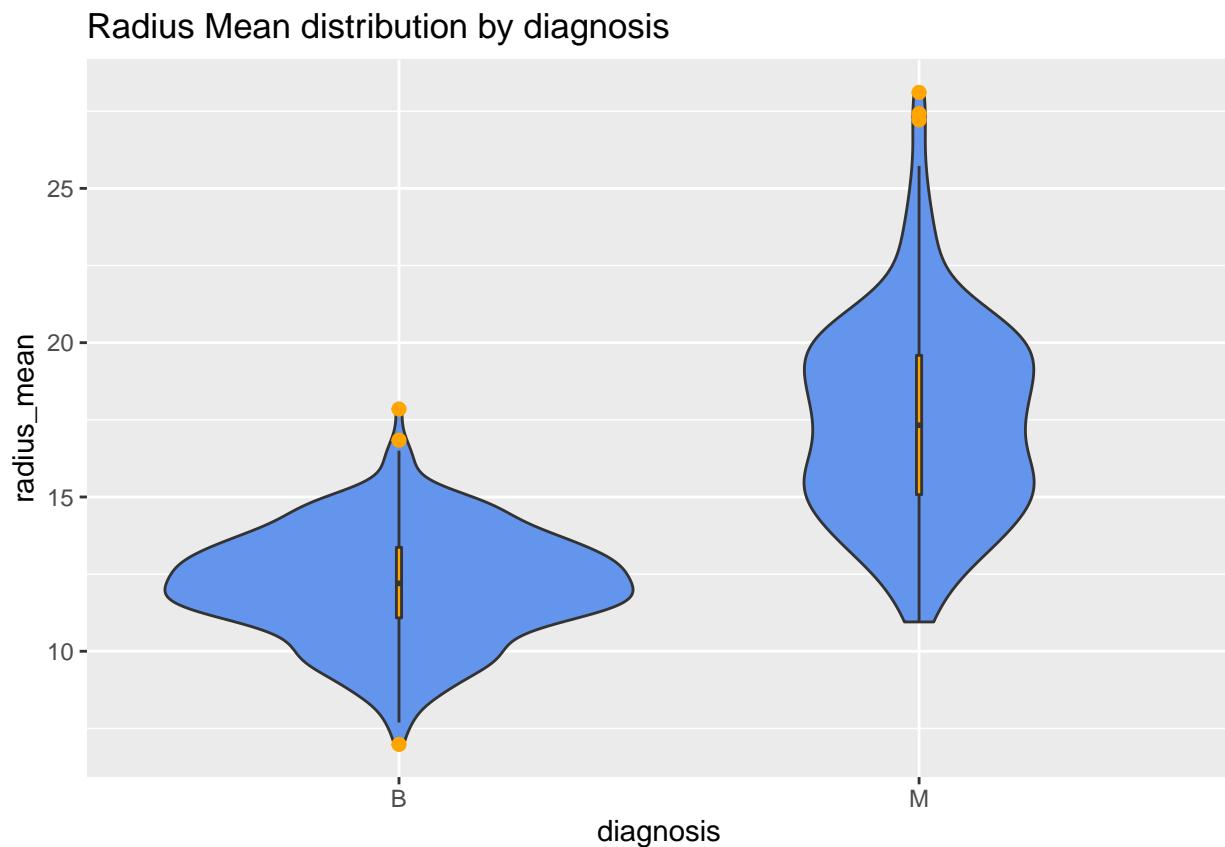


```
pairs.panels(data[,c(23:32)], main="Cancer Worst")
```

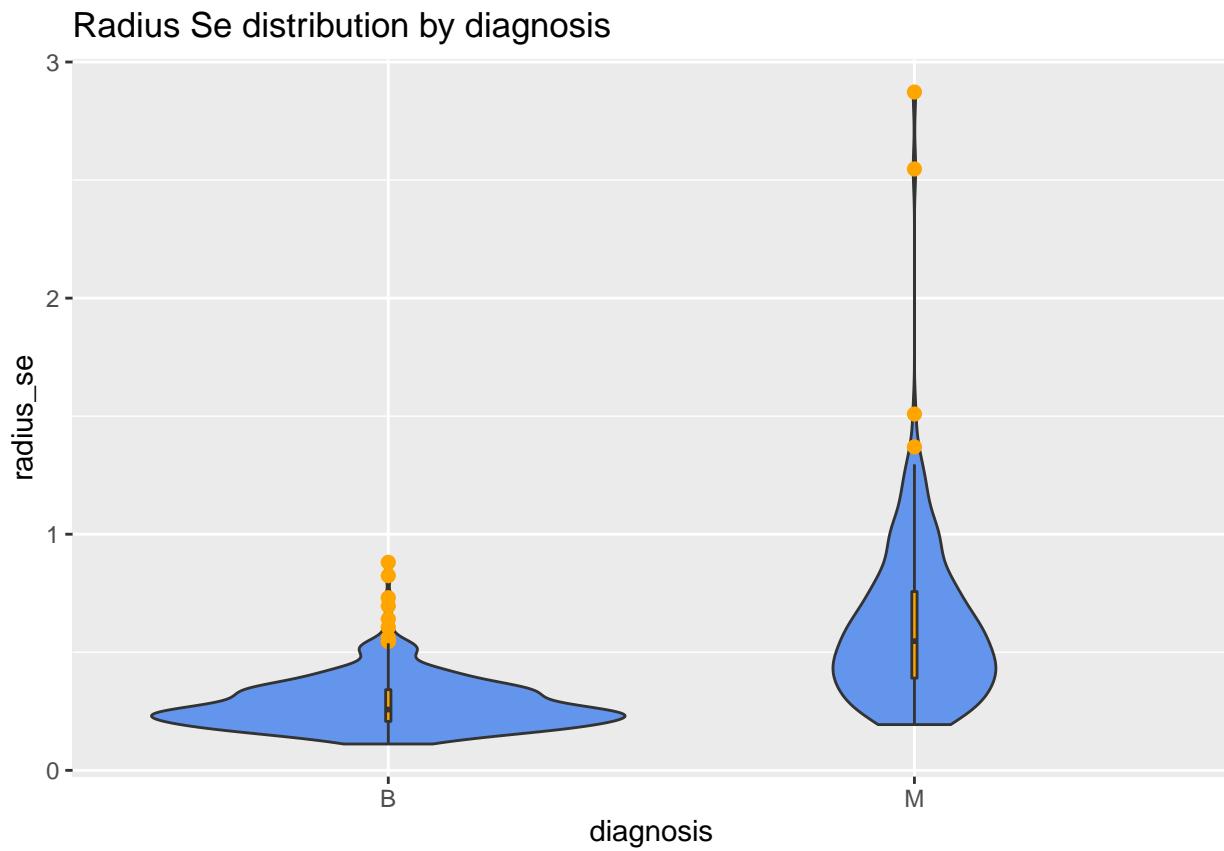
Cancer Worst



```
ggplot(data, aes(x = diagnosis,  
y = radius_mean)) + geom_violin(fill = "cornflowerblue") + geom_boxplot(width = .01,  
fill = "orange", outlier.color = "orange", outlier.size = 2) +  
labs(title = "Radius Mean distribution by diagnosis")
```

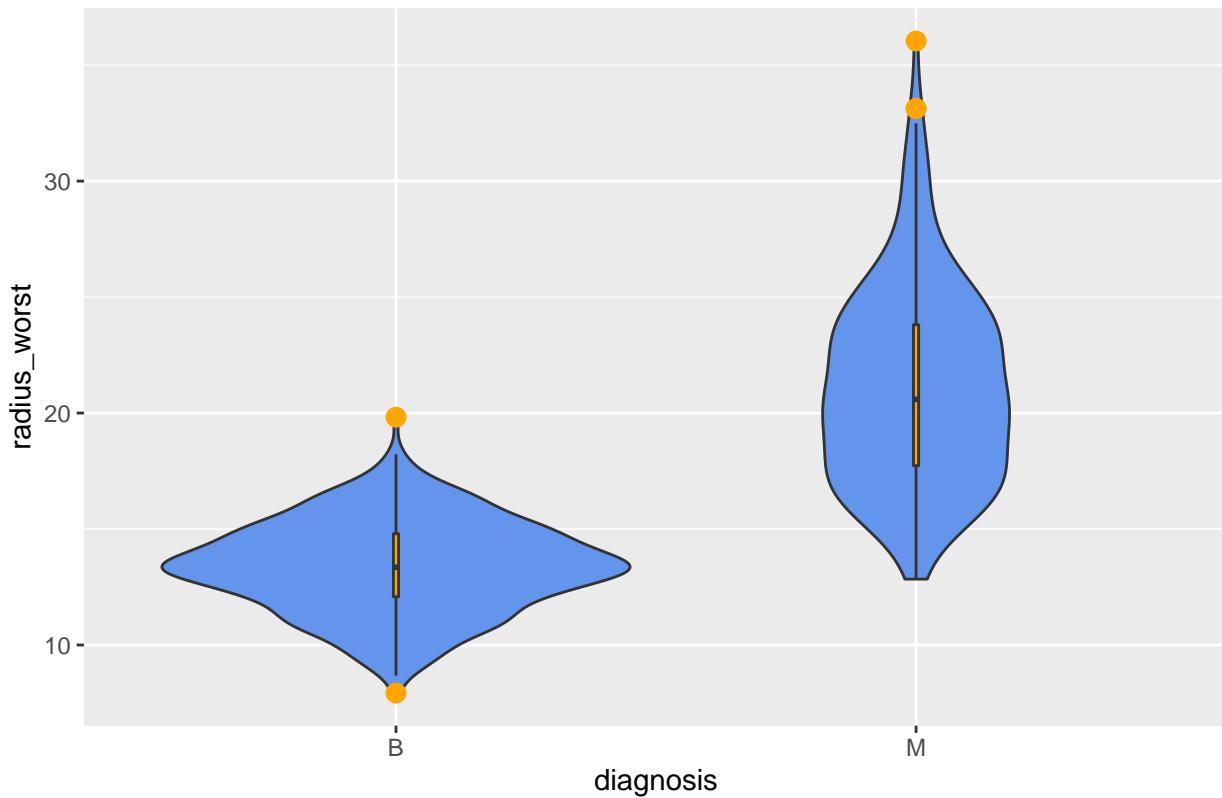


```
ggplot(data, aes(x = diagnosis,  
y = radius_se)) + geom_violin(fill = "cornflowerblue") + geom_boxplot(width = .01,  
fill = "orange", outlier.color = "orange", outlier.size = 2) +  
labs(title = "Radius Se distribution by diagnosis")
```



```
ggplot(data, aes(x = diagnosis,
y = radius_worst)) + geom_violin(fill = "cornflowerblue") + geom_boxplot(width = .01,
fill = "orange", outlier.color = "orange", outlier.size = 3) +
labs(title = "Radius Wprst distribution by diagnosis")
```

Radius Wprst distribution by diagnosis



Let's split the data now to see how tumors differ for M and B

```
dataNew <- split(data, data$diagnosis)
dataB <- dataNew$B
dataM <- dataNew$M
```

Now we have two different datasets for B and M

####(I will do this work later, I can't find perfect visualization technique for this)

Model Building

```
cancerData <- select(cancerData, -id, -X)
cancerData$diagnosis<- as.numeric(as.factor(cancerData$diagnosis))

library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:psych':
##   alpha

## The following object is masked from 'package:purrr':
##   cross
```

```

## The following object is masked from 'package:ggplot2':
##
##      alpha

ind <- sample(1:dim(cancerData)[1], 170)
train <- cancerData[-ind, ]
test <- cancerData[ind, ]

library(e1071)

svmfit = svm(diagnosis ~ ., data = train, kernel = "linear", cost = 5, scale = FALSE)
tune.out <- tune(svm, diagnosis ~ ., data = train, kernel = "linear",
                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
# extract the best model
(bestmod <- tune.out$best.model)

##
## Call:
## best.tune(method = svm, train.x = diagnosis ~ ., data = train, ranges = list(cost = c(0.001,
##      0.01, 0.1, 1, 5, 10, 100)), kernel = "linear")
##
##
## Parameters:
##   SVM-Type:  eps-regression
##   SVM-Kernel: linear
##   cost: 0.1
##   gamma: 0.03333333
##   epsilon: 0.1
##
##
## Number of Support Vectors:  321

prediction <- predict(bestmod, test)
prediction_train <- predict(bestmod, train)

library(Metrics)

##
## Attaching package: 'Metrics'

## The following objects are masked from 'package:caret':
##
##      precision, recall

cor(test$diagnosis, prediction)*100

## [1] 85.40166

cmBest = table(test[,1], prediction)
#cm
cm2Best = table(train[,1], prediction_train )
#cm2

```

```

classifier = svm(formula = diagnosis ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = 'linear')
y_pred = predict(classifier, test)
y_train_pred = predict(classifier, train)

summary (classifier )

## 
## Call:
## svm(formula = diagnosis ~ ., data = train, type = "C-classification",
##       kernel = "linear")
##
## 
## Parameters:
##   SVM-Type:  C-classification
##   SVM-Kernel: linear
##   cost:    1
##
## Number of Support Vectors:  32
##
##  ( 15 17 )
##
## 
## Number of Classes:  2
##
## Levels:
##  1 2

cmBEST = table(test[,1], y_pred)
cm2BEST = table(train[,1], y_train_pred )

#cor(test$diagnosis,y_pred)*100

```

#7 incorrect in train and 2 incorrect in test

```

library(caret)
confusionMatrix(cmBEST)

## Confusion Matrix and Statistics
##
##      y_pred
##      1 2
##  1 99 2
##  2  1 68
##
##          Accuracy : 0.9824
##          95% CI : (0.9493, 0.9963)
##  No Information Rate : 0.5882
##  P-Value [Acc > NIR] : <2e-16
##
```

```

##                               Kappa : 0.9635
##
##  Mcnemar's Test P-Value : 1
##
##                               Sensitivity : 0.9900
##                               Specificity : 0.9714
##                               Pos Pred Value : 0.9802
##                               Neg Pred Value : 0.9855
##                               Prevalence : 0.5882
##                               Detection Rate : 0.5824
##  Detection Prevalence : 0.5941
##                               Balanced Accuracy : 0.9807
##
##                               'Positive' Class : 1
##
```

```
confusionMatrix(cm2BEST)
```

```

## Confusion Matrix and Statistics
##
##   y_train_pred
##   1     2
## 1 255    1
## 2    3 140
##
##                               Accuracy : 0.99
##                               95% CI : (0.9745, 0.9973)
##  No Information Rate : 0.6466
##  P-Value [Acc > NIR] : <2e-16
##
##                               Kappa : 0.9781
##
##  Mcnemar's Test P-Value : 0.6171
##
##                               Sensitivity : 0.9884
##                               Specificity : 0.9929
##                               Pos Pred Value : 0.9961
##                               Neg Pred Value : 0.9790
##                               Prevalence : 0.6466
##                               Detection Rate : 0.6391
##  Detection Prevalence : 0.6416
##                               Balanced Accuracy : 0.9906
##
##                               'Positive' Class : 1
##
```

Model accuracy of 98% obtained on test set and also 98% on train set.

```
###Code from stackoverflow ###https://stackoverflow.com/questions/23891140/r-how-to-visualize-confusion-matrix-using-the-caret-package/42940553
```

```
cm <- confusionMatrix(cmBEST)
draw_confusion_matrix <- function(cm) {
```

```

total <- sum(cm$table)
res <- as.numeric(cm$table)

# Generate color gradients. Palettes come from RColorBrewer.
greenPalette <- c("#F7FCF5", "#E5F5E0", "#C7E9C0", "#A1D99B", "#74C476", "#41AB5D", "#238B45", "#006D2C", "#004422")
redPalette <- c("#FFF5F0", "#FEE0D2", "#FCBBA1", "#FC9272", "#FB6A4A", "#EF3B2C", "#CB181D", "#A50F15", "#67000D")
getColor <- function (greenOrRed = "green", amount = 0) {
  if (amount == 0)
    return("#FFFFFF")
  palette <- greenPalette
  if (greenOrRed == "red")
    palette <- redPalette
  colorRampPalette(palette)(100)[10 + ceiling(90 * amount / total)]
}

# set the basic layout
layout(matrix(c(1,1,2)))
par(mar=c(2,2,2,2))
plot(c(100, 345), c(300, 450), type = "n", xlab="", ylab="", xaxt='n', yaxt='n')
title('CONFUSION MATRIX', cex.main=2)

# create the matrix
classes = colnames(cm$table)
rect(150, 430, 240, 370, col=getColor("green", res[1]))
text(195, 435, classes[1], cex=1.2)
rect(250, 430, 340, 370, col=getColor("red", res[3]))
text(295, 435, classes[2], cex=1.2)
text(125, 370, 'Predicted', cex=1.3, srt=90, font=2)
text(245, 450, 'Actual', cex=1.3, font=2)
rect(150, 305, 240, 365, col=getColor("red", res[2]))
rect(250, 305, 340, 365, col=getColor("green", res[4]))
text(140, 400, classes[1], cex=1.2, srt=90)
text(140, 335, classes[2], cex=1.2, srt=90)

# add in the cm results
text(195, 400, res[1], cex=1.6, font=2, col='white')
text(195, 335, res[2], cex=1.6, font=2, col='white')
text(295, 400, res[3], cex=1.6, font=2, col='white')
text(295, 335, res[4], cex=1.6, font=2, col='white')

# add in the specifics
plot(c(100, 0), c(100, 0), type = "n", xlab="", ylab="", main = "DETAILS", xaxt='n', yaxt='n')
text(10, 85, names(cm$byClass[1]), cex=1.2, font=2)
text(10, 70, round(as.numeric(cm$byClass[1]), 3), cex=1.2)
text(30, 85, names(cm$byClass[2]), cex=1.2, font=2)
text(30, 70, round(as.numeric(cm$byClass[2]), 3), cex=1.2)
text(50, 85, names(cm$byClass[5]), cex=1.2, font=2)
text(50, 70, round(as.numeric(cm$byClass[5]), 3), cex=1.2)
text(70, 85, names(cm$byClass[6]), cex=1.2, font=2)
text(70, 70, round(as.numeric(cm$byClass[6]), 3), cex=1.2)
text(90, 85, names(cm$byClass[7]), cex=1.2, font=2)
text(90, 70, round(as.numeric(cm$byClass[7]), 3), cex=1.2)

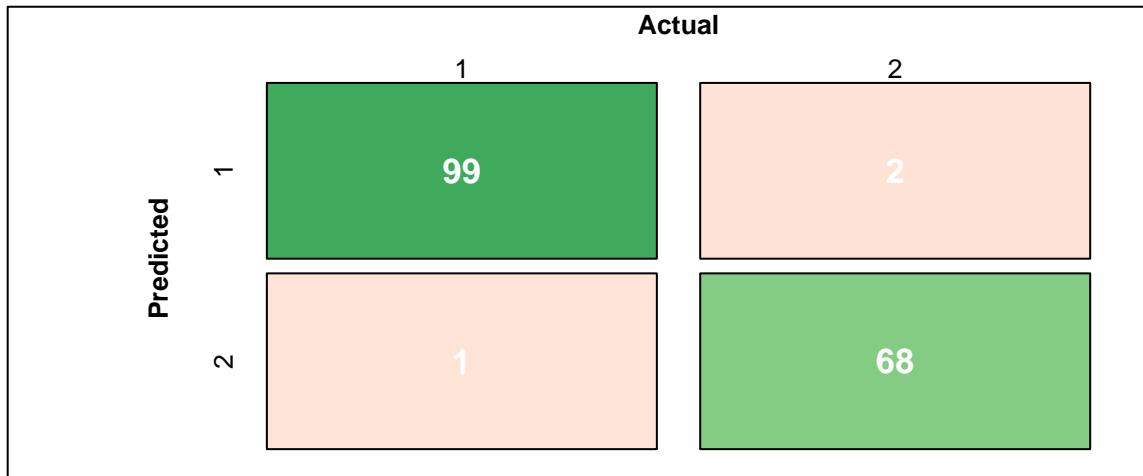
```

```

# add in the accuracy information
text(30, 35, names(cm$overall[1]), cex=1.5, font=2)
text(30, 20, round(as.numeric(cm$overall[1]), 3), cex=1.4)
text(70, 35, names(cm$overall[2]), cex=1.5, font=2)
text(70, 20, round(as.numeric(cm$overall[2]), 3), cex=1.4)
}
draw_confusion_matrix(cm)

```

CONFUSION MATRIX

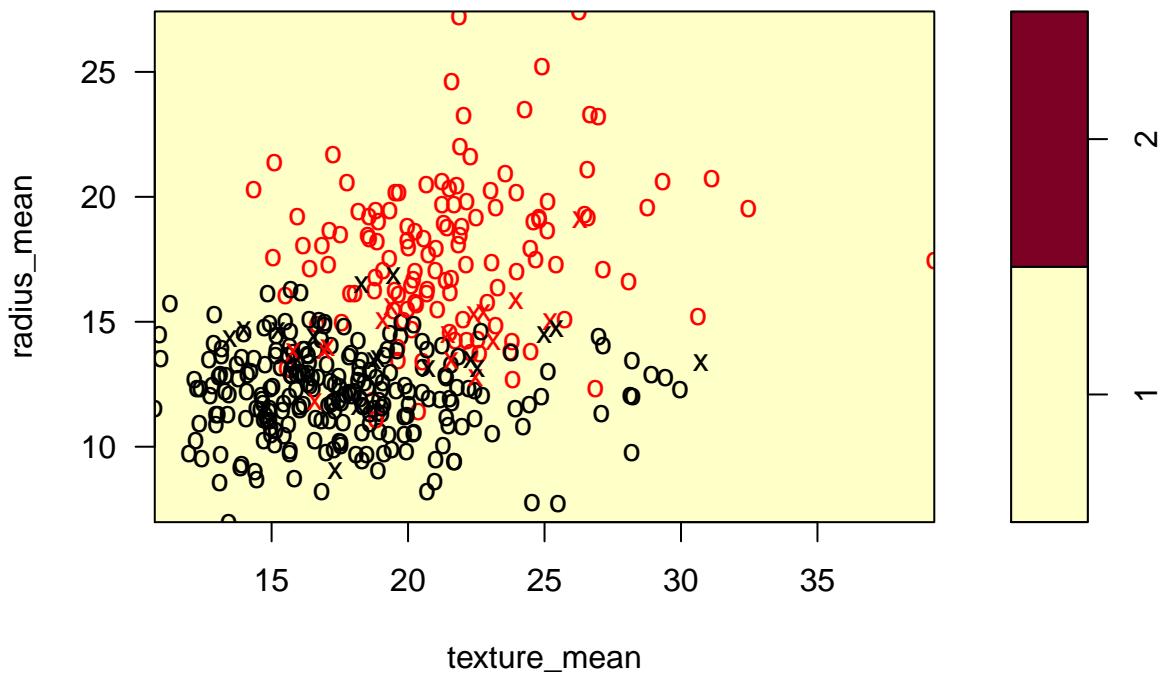


DETAILS

Sensitivity 0.99	Specificity 0.971	Precision 0.98	Recall 0.99	F1 0.985
	Accuracy 0.982		Kappa 0.963	

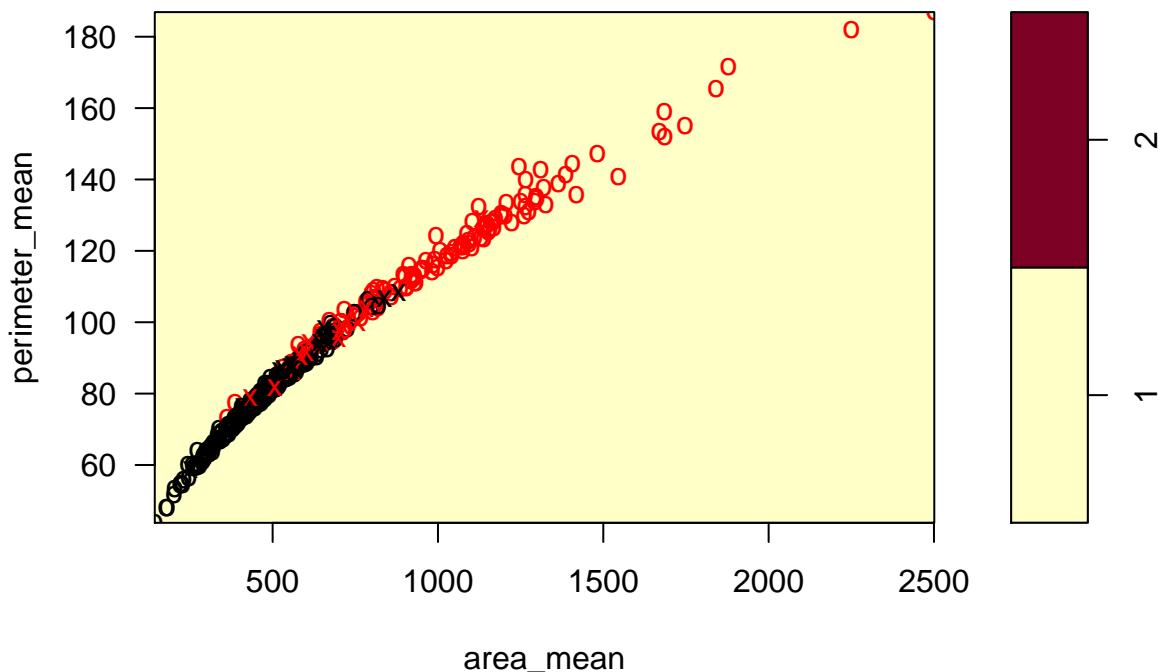
```
plot(classifier , train, radius_mean~texture_mean )
```

SVM classification plot



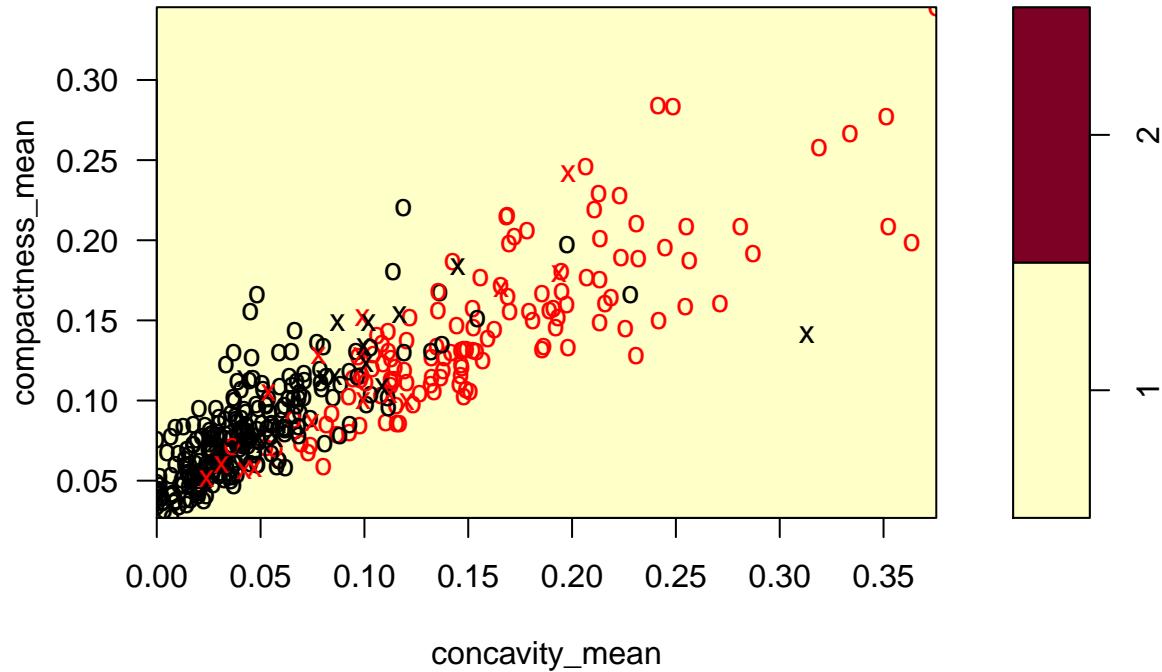
```
plot(classifier , train, perimeter_mean~area_mean )
```

SVM classification plot



```
plot(classifier , train, compactness_mean~concavity_mean )
```

SVM classification plot



```
plot(classifier , train, radius_se~radius_mean )
```

SVM classification plot

