Final Project Paper

Sheikh-Sedat Touray

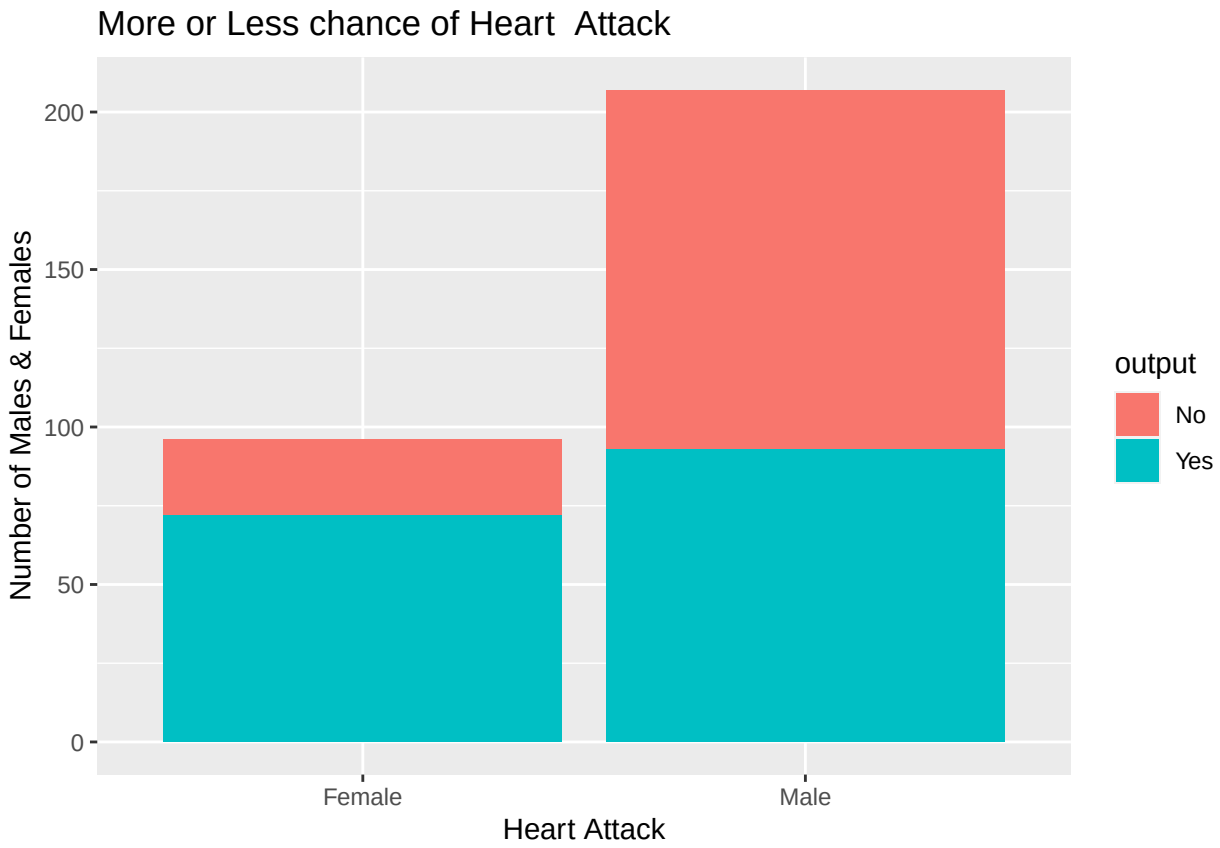
2023-10-20

```
heart <- read.csv('heart.csv')
head(heart)
```

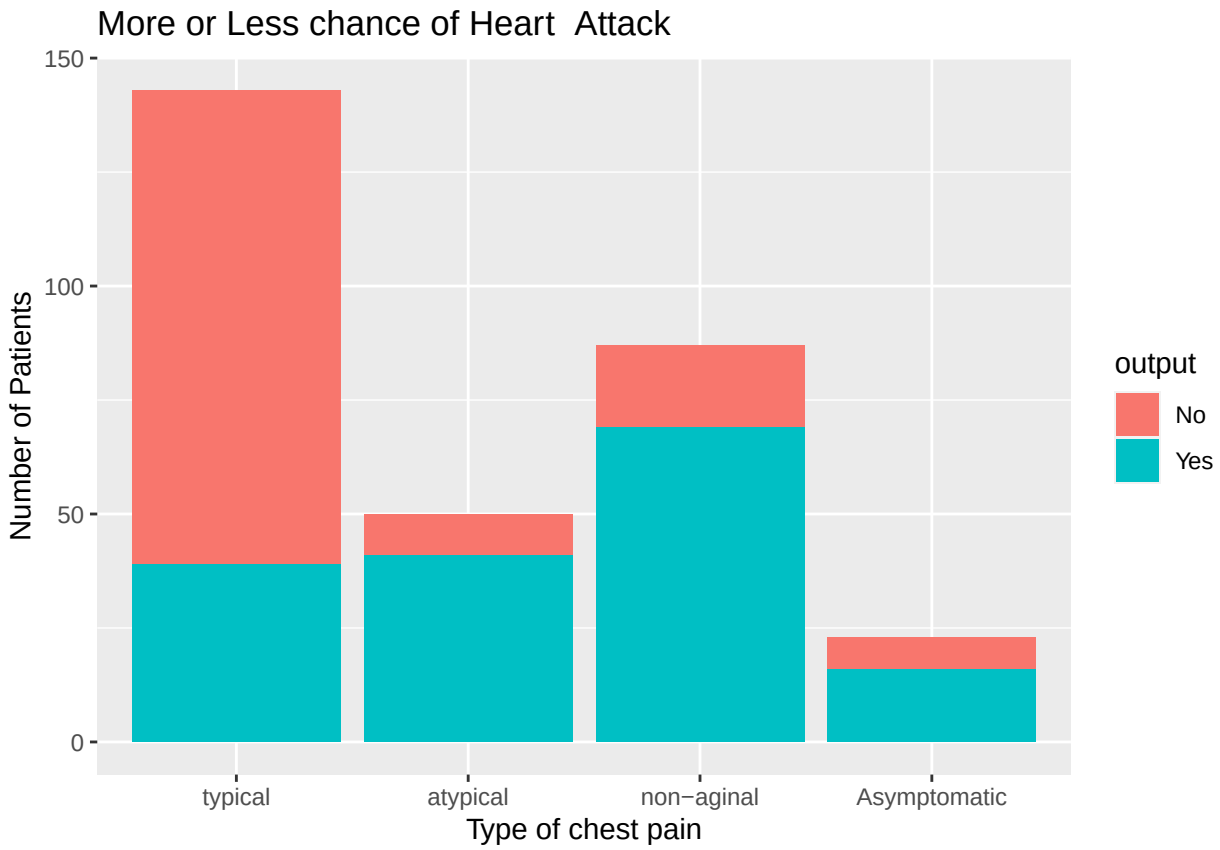
```
##   age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output
## 1  63   1  3    145  233   1       0      150    0     2.3   0   0     1      1
## 2  37   1  2    130  250   0       1      187    0     3.5   0   0     2      1
## 3  41   0  1    130  204   0       0      172    0     1.4   2   0     2      1
## 4  56   1  1    120  236   0       1      178    0     0.8   2   0     2      1
## 5  57   0  0    120  354   0       1      163    1     0.6   2   0     2      1
## 6  57   1  0    140  192   0       1      148    0     0.4   1   0     1      1
```

```
library(plyr)
heart$output <- as.factor(heart$output)
heart$output <- revalue(heart$output, c('1' = 'Yes', '0' = 'No'))
heart$sex <- as.factor(heart$sex)
heart$sex <- revalue(heart$sex, c('1' = 'Male', '0' = 'Female'))
heart$cp <- as.factor(heart$cp)
heart$cp <- revalue(heart$cp, c('0' = 'typical', '1' = 'atypical', '2' = 'non-aginal', '3'='Asymptomati
heart$restecg <- as.factor(heart$restecg)
heart$restecg <- revalue(heart$restecg, c('0' = 'Normal', '1' = 'Abnormal', '2' = 'LVH'))
heart$exng <- as.factor(heart$exng)
heart$exng <- revalue(heart$exng, c('1' = 'Yes', '0' = 'No'))
heart$slp <- as.factor(heart$slp)
heart$slp <- revalue(heart$slp, c('0' = 'Upsloping', '1' = 'Flat', '2' = 'Downsloping'))
heart$fbs <- as.factor(heart$fbs)
heart$fbs <- revalue(heart$fbs, c('1' = 'True', '0' = 'False'))
```

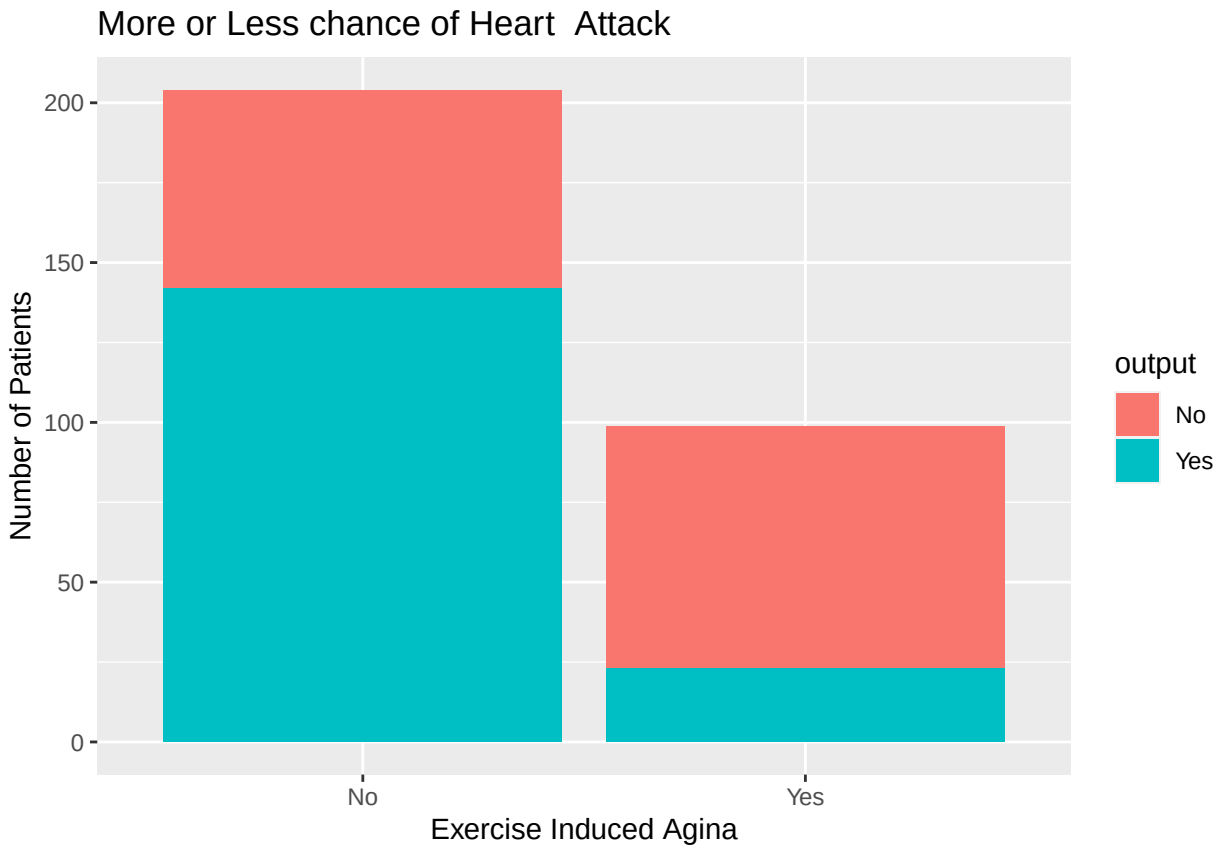
```
library(ggplot2)
classofpar1 <- ggplot(heart, aes(x=sex, fill=output))+
  geom_bar()+
  xlab("Heart Attack")+
  ylab("Number of Males & Females")+
  ggtitle("More or Less chance of Heart  Attack")
classofpar1
```



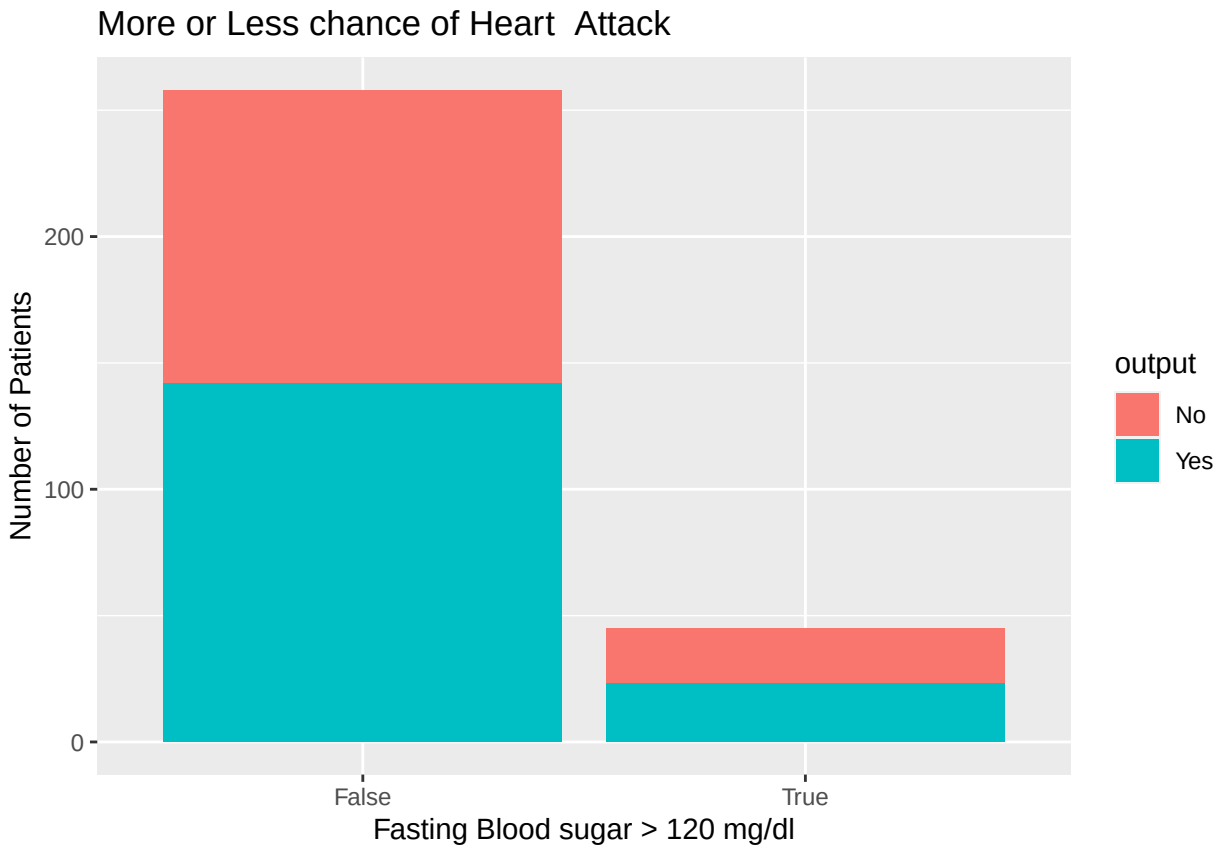

```
library(ggplot2)
classofpar2 <- ggplot(heart, aes(x=cp, fill=output))+
  geom_bar()+
  xlab("Type of chest pain")+
  ylab("Number of Patients")+
  ggtitle("More or Less chance of Heart  Attack")
classofpar2
```




```
classofpar3 <- ggplot(heart, aes(x=exng, fill=output))+
  geom_bar()+
  xlab("Exercise Induced Agina")+
  ylab("Number of Patients")+
  ggtitle("More or Less chance of Heart  Attack")
classofpar3
```




```
classofpar4 <- ggplot(heart, aes(x=fbs, fill=output))+
  geom_bar()+
  xlab("Fasting Blood sugar > 120 mg/dl")+
  ylab("Number of Patients")+
  ggtitle("More or Less chance of Heart  Attack")
classofpar4
```




```
summary(heart)
```

```
##       age            sex                    cp          trtbps
##  Min.   :29.00   Female: 96   typical     :143   Min.   : 94.0
##  1st Qu.:47.50   Male  :207   atypical    : 50   1st Qu.:120.0
##  Median :55.00                non-aginal  : 87   Median :130.0
##  Mean   :54.37                Asymptomatic: 23   Mean   :131.6
##  3rd Qu.:61.00                                   3rd Qu.:140.0
##  Max.   :77.00                                   Max.   :200.0
##       chol          fbs          restecg       thalachh      exng
##  Min.   :126.0   False:258   Normal  :147   Min.   : 71.0   No :204
##  1st Qu.:211.0   True : 45   Abnormal:152   1st Qu.:133.5   Yes: 99
##  Median :240.0               LVH     :  4   Median :153.0
##  Mean   :246.3                              Mean   :149.6
##  3rd Qu.:274.5                              3rd Qu.:166.0
##  Max.   :564.0                              Max.   :202.0
##     oldpeak              slp           caa             thall       output
##  Min.   :0.00   Upsloping  : 21   Min.   :0.0000   Min.   :0.000   No :138
##  1st Qu.:0.00   Flat       :140   1st Qu.:0.0000   1st Qu.:2.000   Yes:165
##  Median :0.80   Downsloping:142   Median :0.0000   Median :2.000
##  Mean   :1.04                     Mean   :0.7294   Mean   :2.314
##  3rd Qu.:1.60                     3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :6.20                     Max.   :4.0000   Max.   :3.000
```

```
str(heart)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : int  63 37 41 56 57 57 56 44 52 57 ...
```

```
##  $ sex     : Factor w/ 2 levels "Female","Male": 2 2 1 2 1 2 1 2 2 2 ...
##  $ cp      : Factor w/ 4 levels "typical","atypical",..: 4 3 2 2 1 1 2 2 3 3 ...
##  $ trtbps  : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol    : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs     : Factor w/ 2 levels "False","True": 2 1 1 1 1 1 1 1 2 1 ...
##  $ restecg : Factor w/ 3 levels "Normal","Abnormal",..: 1 2 1 2 2 2 1 2 2 2 ...
##  $ thalachh: int  150 187 172 178 163 148 153 173 162 174 ...
##  $ exng    : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
##  $ oldpeak : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ slp     : Factor w/ 3 levels "Upsloping","Flat",..: 1 1 3 3 3 2 2 3 3 3 ...
##  $ caa     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ thall   : int  1 2 2 2 2 1 2 3 3 2 ...
##  $ output  : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
```

Assumptions 1. Normality 2. Positive Determinant for variance co-variance matrix 3. Equal variance between two groups

"'d) Providing univariate means and variances

```
df <- data.frame(heart$age,heart$trtbps,heart$chol,heart$thalachh,heart$oldpeak)
by(df, heart$output, colMeans)
```

```
## heart$output: No
##      heart.age   heart.trtbps     heart.chol heart.thalachh  heart.oldpeak
##      56.601449     134.398551     251.086957     139.101449       1.585507
## ------------------------------------------------------------
## heart$output: Yes
##      heart.age   heart.trtbps     heart.chol heart.thalachh  heart.oldpeak
##     52.4969697    129.3030303    242.2303030    158.4666667      0.5830303
```

```
by(df, heart$output, var)
```

```
## heart$output: No
##                 heart.age heart.trtbps   heart.chol heart.thalachh
## heart.age       63.394742    35.831535   44.3414789     -23.886279
## heart.trtbps    35.831535   350.810801  125.3592510       2.368031
## heart.chol      44.341479   125.359251 2445.7588067      50.297683
## heart.thalachh -23.886279     2.368031   50.2976833     510.704961
## heart.oldpeak    1.066445     3.128446   -0.6016503      -6.267132
##                heart.oldpeak
## heart.age          1.0664445
## heart.trtbps       3.1284460
## heart.chol        -0.6016503
## heart.thalachh    -6.2671321
## heart.oldpeak      1.6908833
## ------------------------------------------------------------
## heart$output: Yes
##                 heart.age heart.trtbps  heart.chol heart.thalachh heart.oldpeak
## heart.age       91.214930    42.421656  131.525092     -96.288211     1.3017775
## heart.trtbps    42.421656   261.456393   80.783444       8.693089     2.2911493
## heart.chol     131.525092    80.783444 2867.910052      14.843089     2.4130783
## heart.thalachh -96.288211     8.693089   14.843089     367.652846    -2.7249593
## heart.oldpeak    1.301778     2.291149    2.413078      -2.724959     0.6094664
```

```
by(df, heart$output, cor)
```

```
## heart$output: No
```

```
##                 heart.age heart.trtbps  heart.chol heart.thalachh heart.oldpeak
## heart.age       1.0000000   0.24027153  0.11260994    -0.13275071    0.10300417
## heart.trtbps    0.2402715   1.00000000  0.13533593     0.00559456    0.12845037
## heart.chol      0.1126099   0.13533593  1.00000000     0.04500452   -0.00935579
## heart.thalachh -0.1327507   0.00559456  0.04500452     1.00000000   -0.21326864
## heart.oldpeak   0.1030042   0.12845037 -0.00935579    -0.21326864    1.00000000
## ------------------------------------------------------------
## heart$output: Yes
##                 heart.age heart.trtbps heart.chol heart.thalachh heart.oldpeak
## heart.age       1.0000000   0.27469770 0.25715377    -0.52580074    0.17459385
## heart.trtbps    0.2746977   1.00000000 0.09329105     0.02803855    0.18150094
## heart.chol      0.2571538   0.09329105 1.00000000     0.01445515    0.05771833
## heart.thalachh -0.5258007   0.02803855 0.01445515     1.00000000   -0.18203973
## heart.oldpeak   0.1745938   0.18150094 0.05771833    -0.18203973    1.00000000
```

```
library(mvnormtest)
multv <- t(df)

mshapiro.test(multv)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Z
## W = 0.93989, p-value = 9.335e-10
```

Highly significant P value we might have to reject the null hypothesis, does not pass normality test.

```
covheart <- cov(df, method = 'spearman')
det(covheart)
```

```
## [1] 1.476375e+19
```

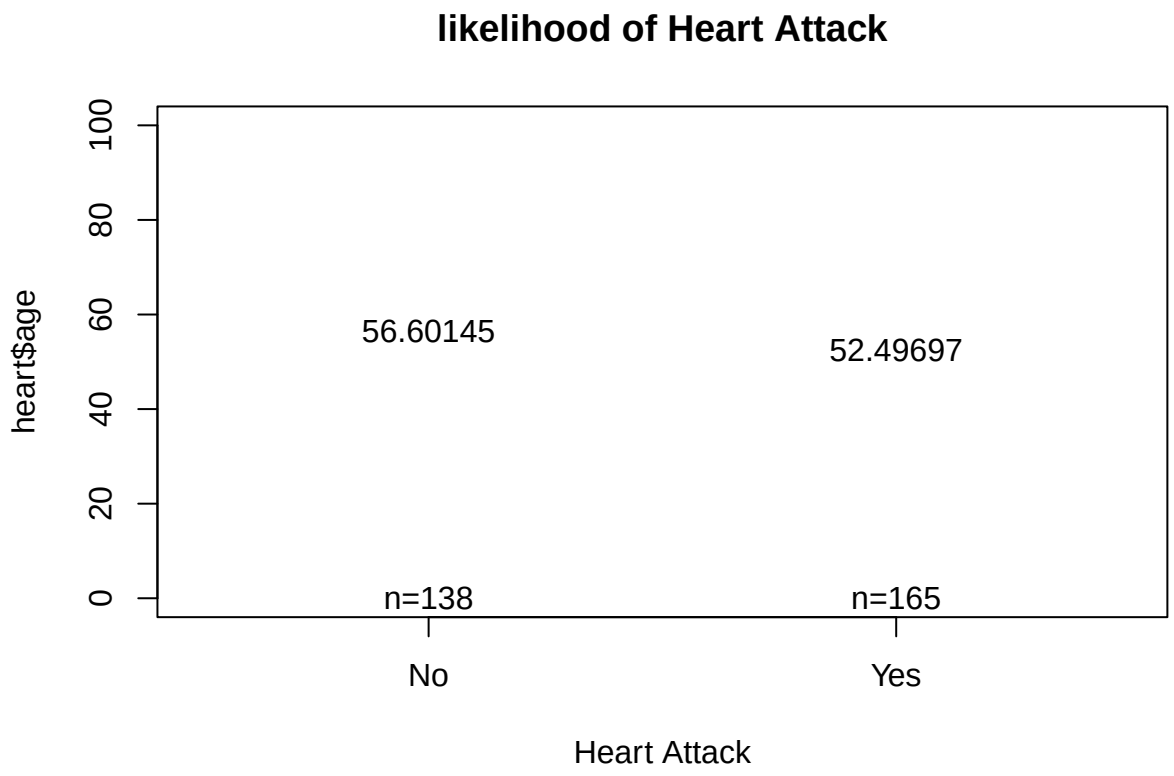
It passes the positive determinant test of variance-covariance matrix.

```
#Graph the means of the 5 variables for presence or absence of heart attack
library(gplots)
```

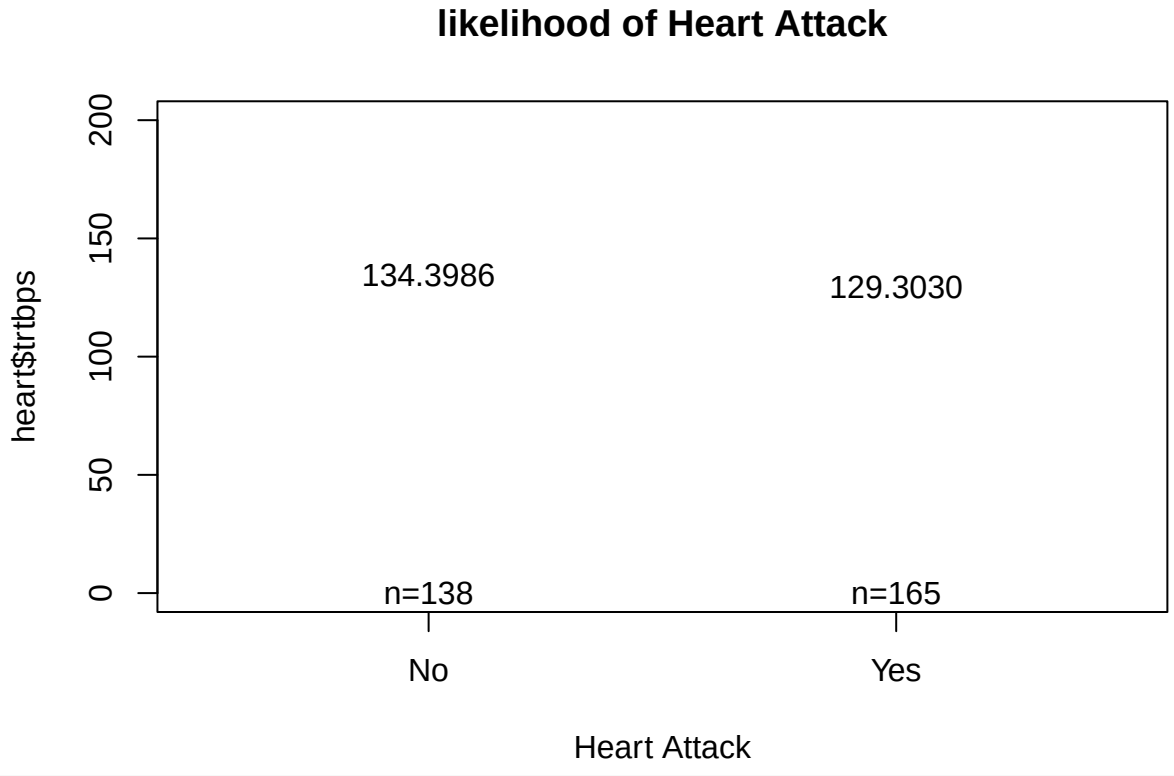
```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

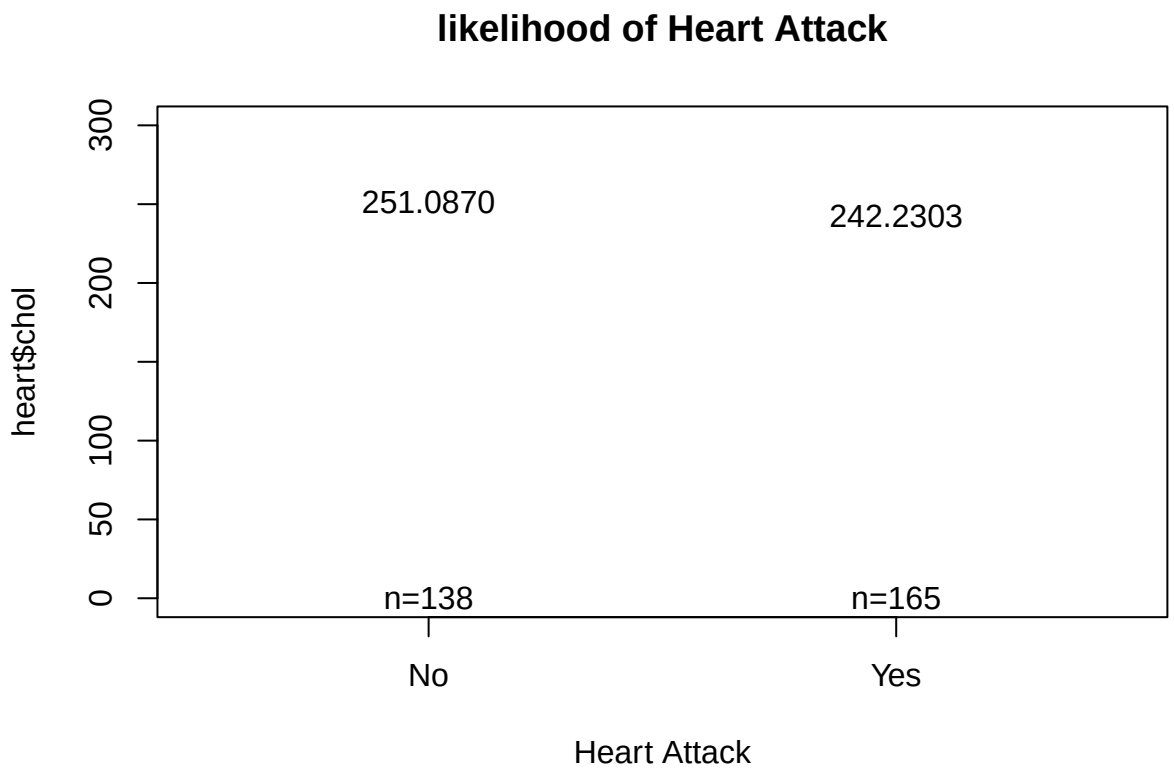
```
plotmeans(heart$age~heart$output, data=heart,ylim=c(0,100),xlab="Heart Attack",legends = c("No","Yes"),n
```



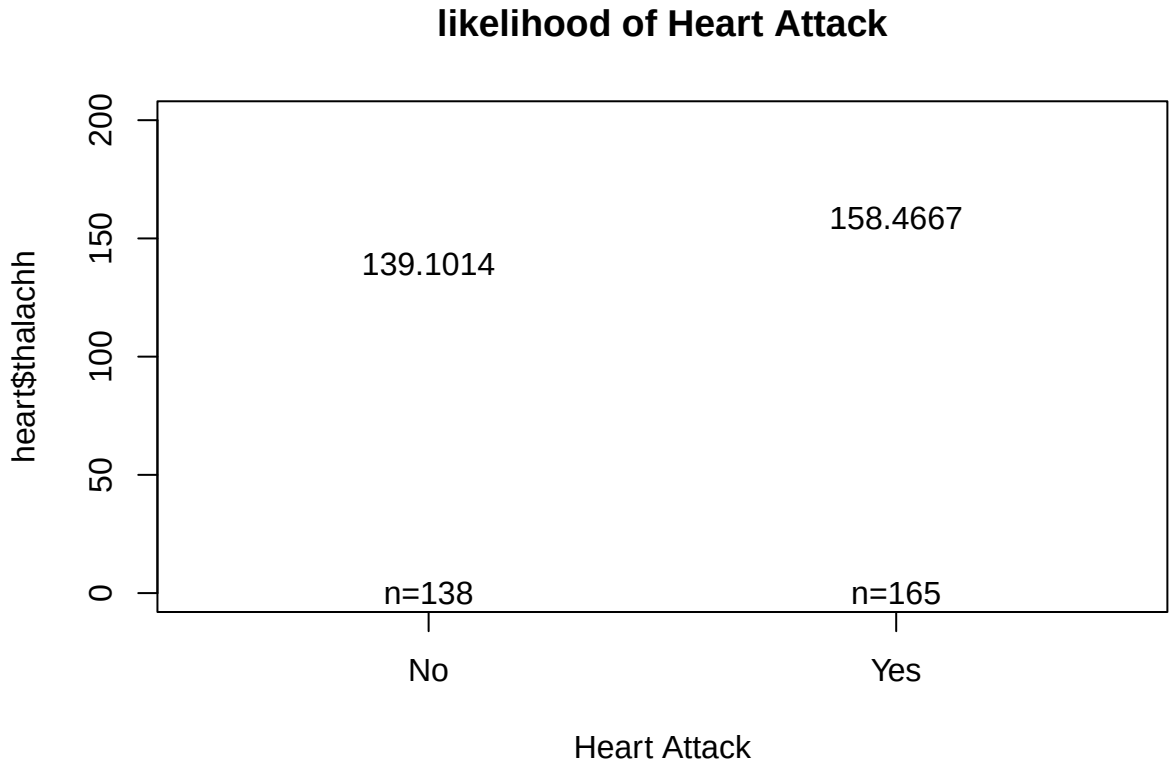

```
plotmeans(heart$trtbps~heart$output, data=heart,ylim=c(0,200),xlab="Heart Attack",legends = c("No","Yes
```



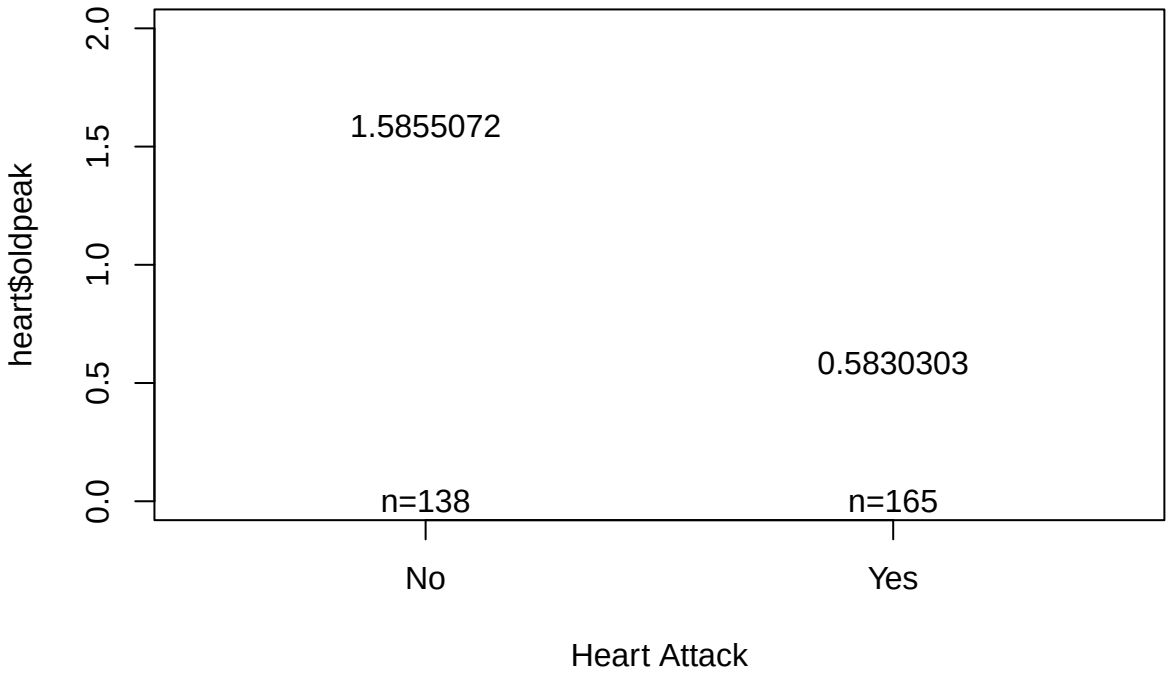

```
plotmeans(heart$chol~heart$output, data=heart,ylim=c(0,300),xlab="Heart Attack",legends = c("No","Yes")
```




```
plotmeans(heart$thalachh~heart$output, data=heart,ylim=c(0,200),xlab="Heart Attack",legends = c("No","Ye
```




```
plotmeans(heart$oldpeak~heart$output, data=heart,ylim=c(0,2),xlab="Heart Attack",legends = c("No","Yes"
```




```
library(ICSNP)
```

```
## Loading required package: mvtnorm
```

```
## Loading required package: ICS
```

```
library(mvtnorm)
```

```
result <- HotellingsT2(df[1:165,], df[166:303,])

# Print the test result
print(result)
```

```
##
##  Hotelling's two sample T2-test
##
## data:  df[1:165, ] and df[166:303, ]
## T.2 = 22.924, df1 = 5, df2 = 297, p-value < 2.2e-16
## alternative hypothesis: true location difference is not equal to c(0,0,0,0,0)
```

```
library(caTools)
set.seed(1028)
total_rows <- nrow(heart)

# Calculate the number of rows for the training and testing sets
train_rows <- round(0.8 * total_rows)  # 80% for training
test_rows <- total_rows - train_rows  # 20% for testing

# Generate random indices for the training set
train_indices <- sample(1:total_rows, train_rows)

# Create the training and testing datasets
```

```r
train_data <- heart[train_indices, ]
test_data <- heart[-train_indices, ]
```

```r
set.seed(1130)
dim(train_data)
```

```
## [1] 242  14
```

```r
dim(test_data)
```

```
## [1] 61 14
```

(c) Perform Classification Tree on the training data in order to predict output of the likelihood of the patient suffering from a heart attack. Use cross- validation to prune the tree. Plot the resulting tree. Evaluate performance on the test data. What test error do you obtain?

```r
library(tree)
attach(heart)
#change this to factors so that it can make a classification tree without having level issues
output= factor(output)
tree_output <- tree(output~.-output,train_data)
```

```r
summary(tree_output)#gives summary statistics of tree
```

```
##
## Classification tree:
## tree(formula = output ~ . - output, data = train_data)
## Variables actually used in tree construction:
## [1] "cp"       "caa"      "oldpeak"  "thall"    "thalachh" "chol"     "age"
## [8] "sex"
## Number of terminal nodes:  19
## Residual mean deviance:  0.4743 = 105.8 / 223
## Misclassification error rate: 0.1116 = 27 / 242
```

```r
#perform prediction on test data
tree_pred = predict(tree_output, test_data, type = "class")

table(tree_pred,test_data$output) #confusion matrix
```

```
##
## tree_pred No Yes
##       No  22   5
##       Yes  6  28
```

```r
mean(tree_pred != test_data$output) #test error
```

```
## [1] 0.1803279
```

```r
mean(tree_pred == test_data$output) #test accuracy
```

```
## [1] 0.8196721
```

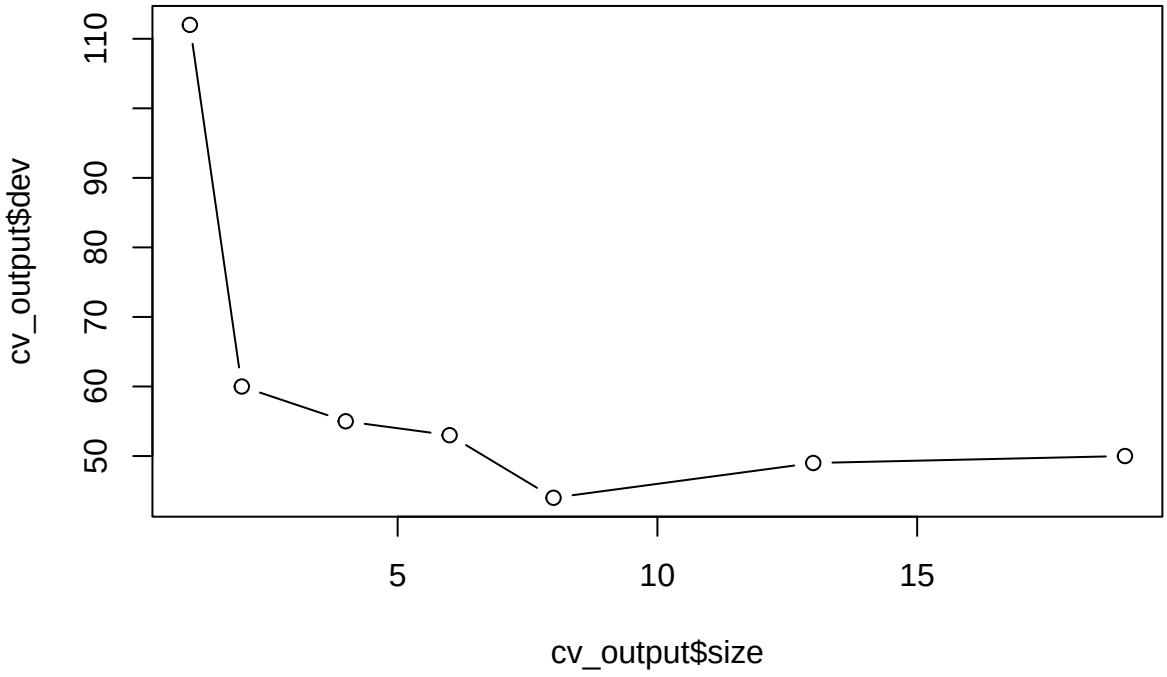
```r
set.seed(1111)
#perform cross validation
cv_output = cv.tree(tree_output, FUN = prune.misclass)

cv_output
```

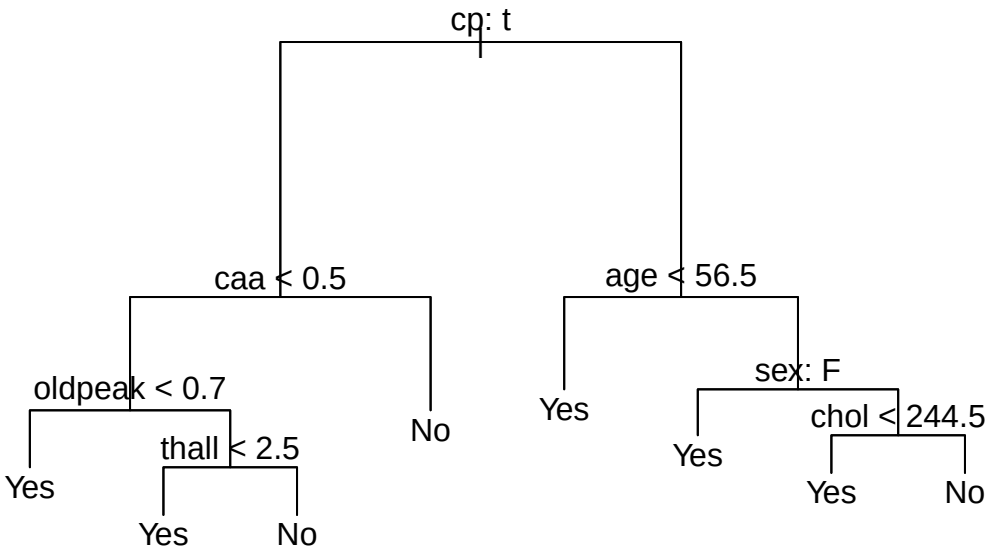
```
## $size
```

```
## [1] 19 13  8  6  4  2  1
##
## $dev
## [1]  50  49  44  53  55  60 112
##
## $k
## [1] -Inf    0    1    3    4    7   50
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
#plot the CV
plot(cv_output$size, cv_output$dev, type = "b")
text(tree_output, pretty=TRUE, cex=0.8)
```




```r
#prune tree and plot resulting tree
prune.output <- prune.tree(tree_output, best = 8)
plot(prune.output)
text(prune.output,pretty=TRUE)
```



find below the test terror for my pruned classification tree

```
#predict test data on pruned tree
prune.pred = predict(prune.output, test_data,type="class")
table(prune.pred,test_data$output)
```

```
##
## prune.pred No Yes
##        No  20   2
##        Yes  8  31
```

```
mean(prune.pred != test_data$output) #test error
```

```
## [1] 0.1639344
```

```
mean(prune.pred == test_data$output) #test accuracy for pruned tree
```

```
## [1] 0.8360656
```

```
#1. Accuracy - How often the model is correct overall
Acc =(31+20)/61
Acc
```

```
## [1] 0.8360656
```

```
#2. Recall - How often the model predicts "Yes", when the actual value is "Yes"
Recall=31/(31+8)
Recall
```

```
## [1] 0.7948718
```

```
#3. Precision - How often is the model correct when the predicted value is "Yes".
prec=31/(31+2)
prec
```

```
## [1] 0.9393939
```

```
#4. F1 Score - The weighted harmonic mean of recall and precision
f1 =2*(0.7949*0.9394)/(0.7949+0.9394)
f1
```

```
## [1] 0.8611302
```

e) Fit a Support Vector Classifier to the data with various values of cost, in order to predict whether a patient is likely to suffer from a heart attack or not. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```
library(e1071)

set.seed(821)
# SV classifier model
svmlinear =  svm(output ~ .-output, data = train_data, kernel = "linear", ranges = list(cost = c(0.001,
summary(svmlinear)
```

```
##
## Call:
## svm(formula = output ~ . - output, data = train_data, kernel = "linear",
##     ranges = list(cost = c(0.001, 0.01, 1, 5, 10, 100)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  107
##
##  ( 52 55 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
#prediction of SV classifier, table and error
pred.svmlinear <- predict(svmlinear,newdata = test_data)
table(test_data$output,pred.svmlinear)
```

```
##      pred.svmlinear
##       No Yes
##   No  21   7
##   Yes  1  32
```

```
mean(pred.svmlinear!=test_data$output)#test error
```

```
## [1] 0.1311475
```

```
mean(pred.svmlinear==test_data$output)#test accuracy
```

```
## [1] 0.8688525
```

As can be seen above the cross-validation gives a better error than just the classifier model

```
set.seed(433)
#cv on SV classifier
svmlineart =  tune(svm,output ~ .-output, data = train_data, kernel = "linear", ranges = list(cost = c(
summary(svmlineart)
```

```
##
## Parameter tuning of 'svm':
```

```
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   100
##
## - best performance: 0.1946667
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.4545000 0.06114166
## 2 1e-02 0.2151667 0.05860229
## 3 1e+00 0.1986667 0.07286805
## 4 5e+00 0.2030000 0.08022268
## 5 1e+01 0.1988333 0.07857547
## 6 1e+02 0.1946667 0.07664170
```

The lowest error was achieved by using a cost of 100, which was 0.194. The highest error was 0.4545 which was when cost was set to 0.001.

f) Now repeat (e), this time using Support Vector Machines (SVMs) with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
set.seed(439)
#SVM polynomial model
svmpol =  svm(output ~ .-output, data = train_data, kernel = "polynomial", ranges = list(cost = c(0.1,
summary(svmpol)
```

```
##
## Call:
## svm(formula = output ~ . - output, data = train_data, kernel = "polynomial",
##     ranges = list(cost = c(0.1, 1, 5, 10), degree = c(2, 3, 4)))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  192
##
##  ( 95 97 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
#predicting new data on the SVM model polynonial model
pred.svmpol <- predict(svmpol,newdata = test_data)
#confusion matrix
table(test_data$output,pred.svmpol)
```

```
##      pred.svmpol
##       No Yes
##   No  16  12
##   Yes  0  33
```

```
#computing test error obtained
mean(pred.svmpol!=test_data$output)
```

```
## [1] 0.1967213
```

```
#computing test Accuracy obtained
mean(pred.svmpol==test_data$output)
```

```
## [1] 0.8032787
```

```
set.seed(1145)
svmpolt =  tune(svm,output ~ .-output, data = train_data, kernel = "polynomial", ranges = list(cost = c
summary(svmpolt)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    10      3
##
## - best performance: 0.2021667
##
## - Detailed performance results:
##    cost degree     error dispersion
## 1   0.1      2 0.4298333 0.08117748
## 2   1.0      2 0.2025000 0.07964524
## 3   5.0      2 0.2105000 0.07795238
## 4  10.0      2 0.2228333 0.07521184
## 5   0.1      3 0.4381667 0.08838555
## 6   1.0      3 0.2855000 0.05503114
## 7   5.0      3 0.2148333 0.06421150
## 8  10.0      3 0.2021667 0.07073272
## 9   0.1      4 0.4548333 0.09699322
## 10  1.0      4 0.3885000 0.07872224
## 11  5.0      4 0.2938333 0.07067160
## 12 10.0      4 0.3100000 0.09032178
```

The lowest error was achieved by using a cost of 1 and degree of 2, which was 0.2025. The highest error was 0.4548 which was when cost was set to 0.1 and a degree 4.

```
set.seed(1146)
svmrad =  svm(output ~ .-output, data = train_data, kernel = "radial", ranges = list(cost = c(0.1, 1, 5
summary(svmrad)
```

```
##
## Call:
## svm(formula = output ~ . - output, data = train_data, kernel = "radial",
##     ranges = list(cost = c(0.1, 1, 5, 10), gamma = c(0.01, 0.1, 1,
##         5, 10, 100)))
```

```
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  139
##
##  ( 70 69 )
##
##
## Number of Classes:  2
##
## Levels:
##  No Yes
```

```
#prediction of SVM radial classifier, table and error
pred.svmrad <- predict(svmrad,newdata = test_data)
table(test_data$output,pred.svmrad)
```

```
##      pred.svmrad
##       No Yes
##   No  24   4
##   Yes  1  32
```

```
mean(pred.svmrad!=test_data$output)
```

```
## [1] 0.08196721
```

```
mean(pred.svmrad==test_data$output)
```

```
## [1] 0.9180328
```

```
#1. Accuracy - How often the model is correct overall
acc2 =(32+24)/61
acc2
```

```
## [1] 0.9180328
```

```
#2. Recall - How often the model predicts "Yes", when the actual value is "Yes".
Recall2=32/(31+1)
Recall2
```

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

```
#3. Precision - How often is the model correct when the predicted value is "Yes".
prec2=32/(32+4)
prec2
```

```
## [1] 0.8888889
```

```
#4. F1 Score - The weighted harmonic mean of recall and precision
f12 =2*(1.0*0.8889)/(1.0+0.8889)
f12
```

```
## [1] 0.9411827
```