# ST5227 Applied Data Mining: Part II

**Yeo Ming Jie, Jonathan**

**2023-02-28**

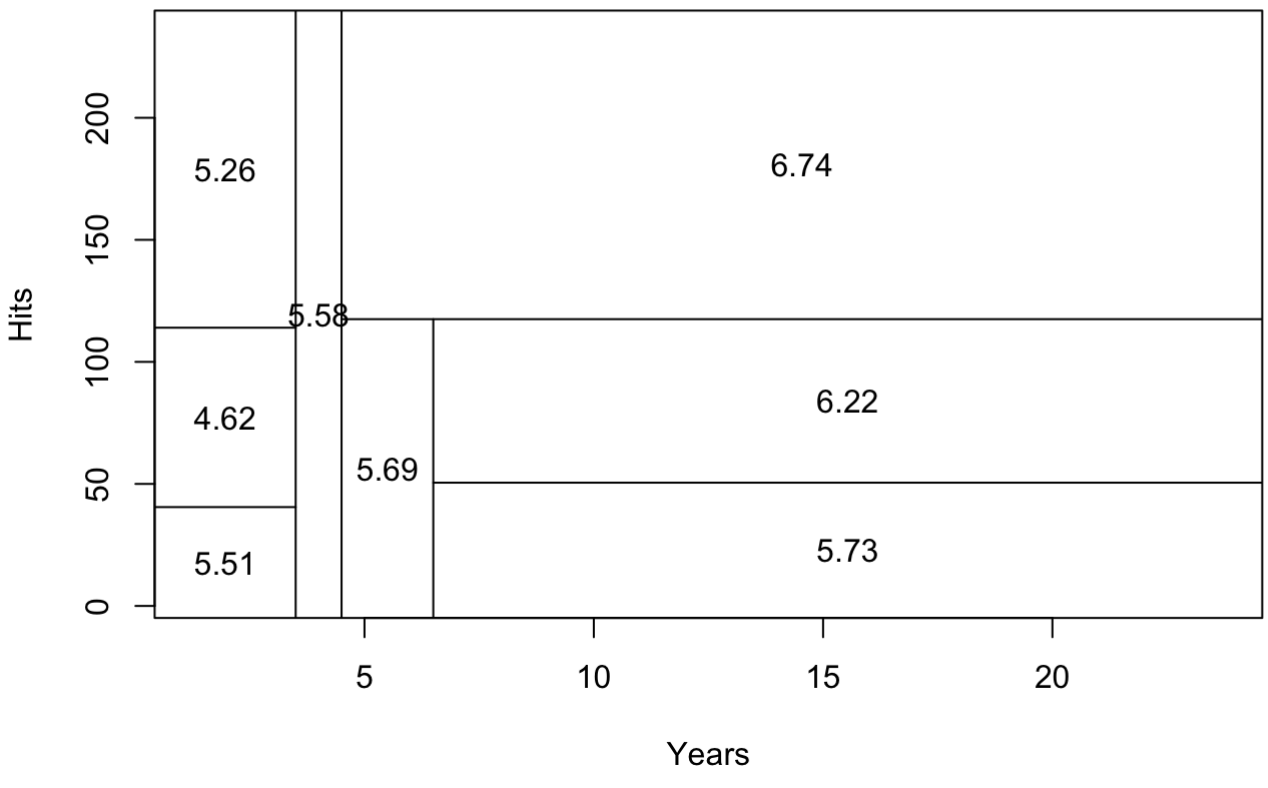
# Week 7: Tree-Based Methods

- Introducing CART - Classification and Regression Trees
- Want to apply CART to predict baseball player's salary (in thousands of dollars) based on $x_1$ = Hits and $x_2$ = Years (of experience).

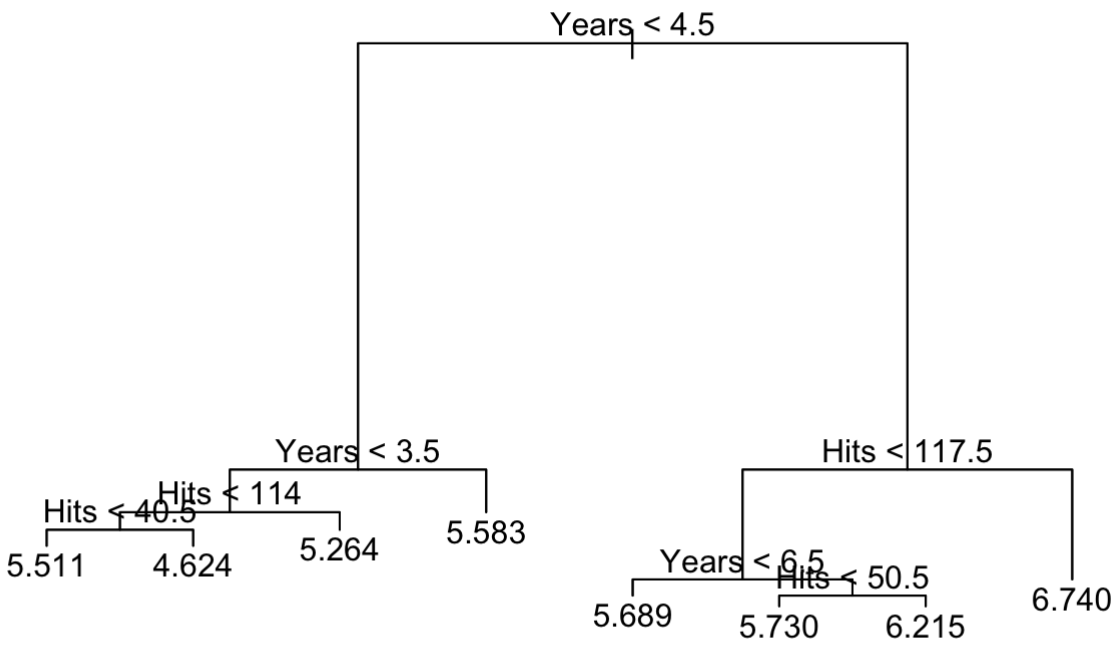
```
library(ISLR2); library(tree)
hit = Hitters[c(2,7,19)]; str(hit)
```

```
## 'data.frame':    322 obs. of  3 variables:
##  $ Hits  : int  66 81 130 141 87 169 37 73 81 92 ...
##  $ Years : int  1 14 3 11 2 11 2 3 2 13 ...
##  $ Salary: num  NA 475 480 500 91.5 750 70 100 75 1100 ...
```

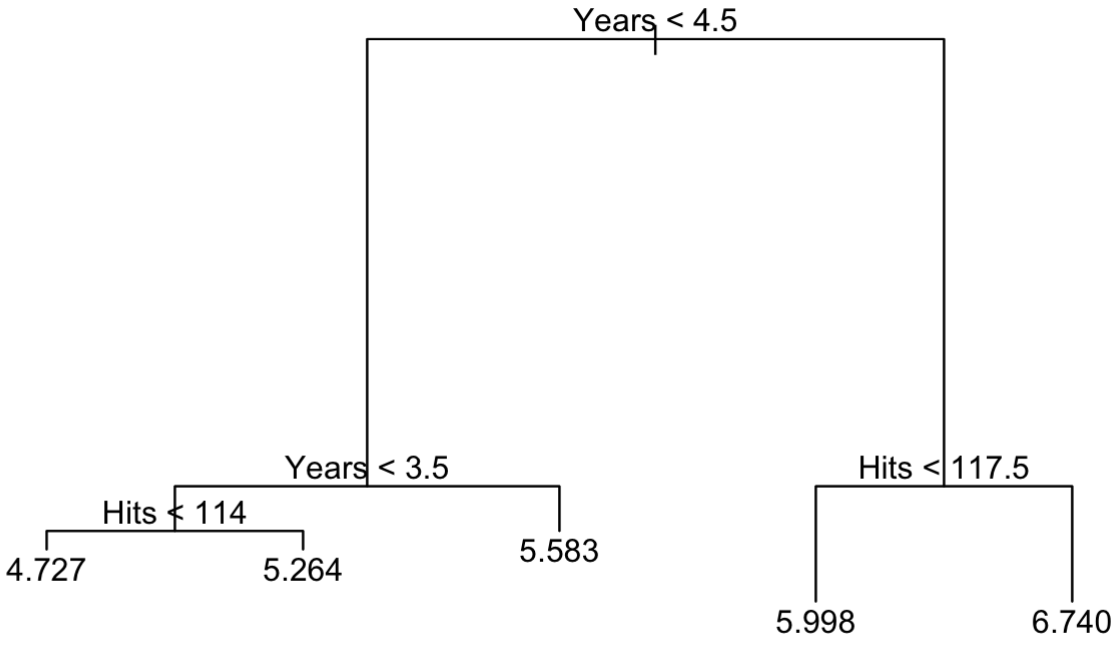
```
hit2 = na.omit(hit) # removing rows wth NA values
hit2[,3] = log(hit2[,3]) # log transform salary

# FITTING CART
hit.tree = tree(Salary ~ Hits + Years, data = hit2)
partition.tree(hit.tree) # plot tree partitions
```




```
plot(hit.tree); text(hit.tree, pretty = 0) # visualize splits
```




```
# Cost-Complexity Pruning
hit.prune = prune.tree(hit.tree, best = 5)
plot(hit.prune); text(hit.prune,pretty=0)
```




```
# Apply 10-fold CV to find best tree using first 7 predictor variables
# Loading Data
hit3=Hitters[c(1:7,19)]; hit3=na.omit(hit3)
hit3[,8]=log(hit3[,8])

# Fit CART (Grow Tree -> Prune down to Root)
hit3.tree=tree(Salary~AtBat+Hits+HmRun+Runs+RBI+Walks+Years, data=hit3)
hit3.prune=prune.tree(hit3.tree)
head(hit3.prune,3) # k denotes pruning parameter alpha of each tree in sequence
```

```
## $size
## [1] 10  9  8  7  5  4  3  2  1
##
## $dev
## [1]  60.35293  63.70414  67.18149  70.84106  78.32631  82.11985  91.32995
## [8] 115.05848 207.15373
##
## $k
## [1]      -Inf  3.351205  3.477350  3.659576  3.742623  3.793540  9.210099
## [8] 23.728527 92.095258
```

```
# Choose 9 alpha values (pruning parameter)
alpha=c(3.3,3.4,3.5,3.6,3.7,3.75,3.8,9,10)
cv.error=rep(0,times=9)

# Random permutation of n = 263 cases
set.seed(5227)
per=sample(c(1:263),263,replace=FALSE)
rhit=hit3[per,]

# Divide into Training-Test data for CV (take each fold as test)
for (k in 1:10){
  vec=c((26*(k-1)+1):(26*k)) # using fold size of 26
  test=rhit[vec,] # Training-Test split
  train=rhit[-vec,]
  cv.tree=tree(Salary~.,data=train)
  cv.prune=prune.tree(cv.tree)

  for(j in 1:9){
    ptree=prune.tree(cv.tree,k=alpha[j])
    pred=predict(ptree,test)
    mse=sum((test$Salary-pred)^2)/26 # computing MSE
    cv.error[j]=cv.error[j]+mse
    }
}

# Cross-Validation Error for each value of alpha
round(data.frame(alpha = alpha, cv_error = cv.error/10),4) # (optimal at alpha = 3.80)
```

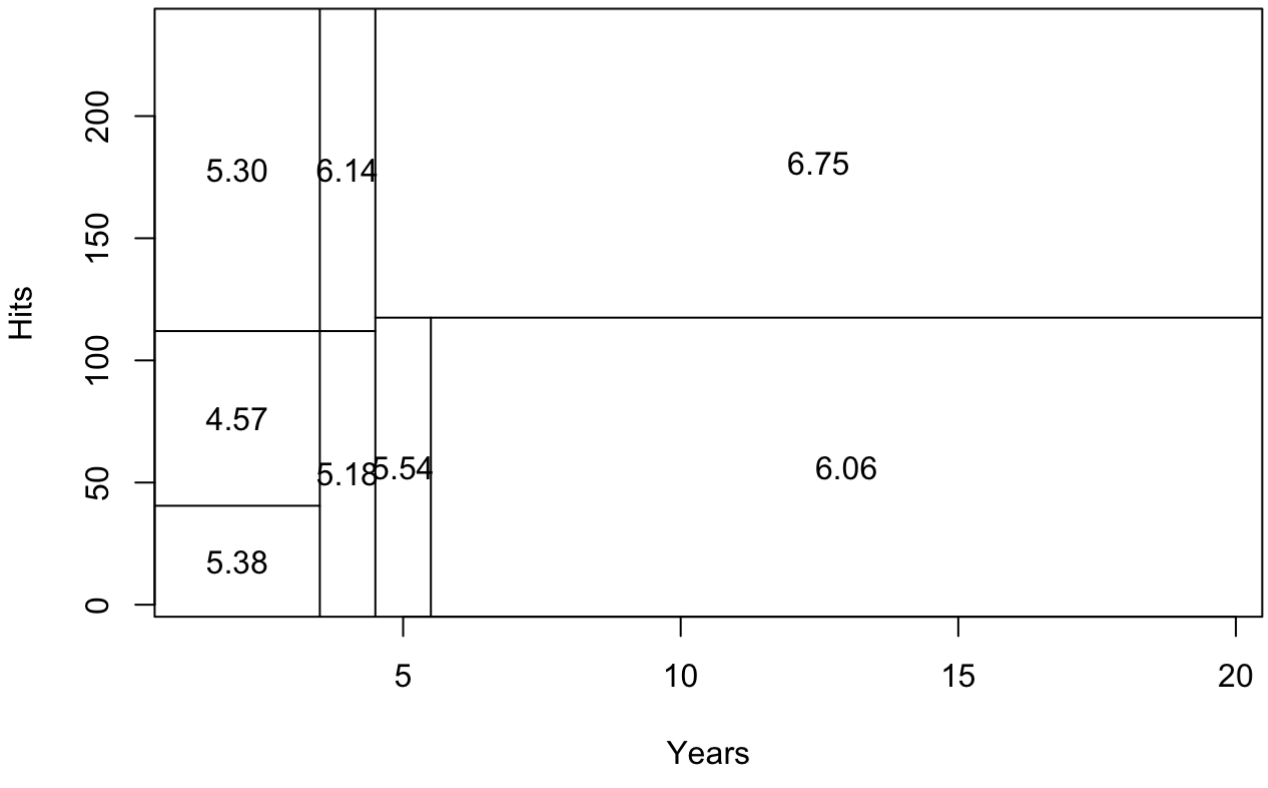
```
##   alpha cv_error
## 1  3.30   0.3680
## 2  3.40   0.3697
## 3  3.50   0.3742
## 4  3.60   0.3661
## 5  3.70   0.3627
## 6  3.75   0.3627
## 7  3.80   0.3593
## 8  9.00   0.3742
## 9 10.00   0.3644
```

```
prune.tree(hit3.tree,k=3.8) # Observe RSS decreasing with splits
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 263 207.20 5.927
##   2) Years < 4.5 90  42.35 5.107
##     4) Years < 3.5 62  23.01 4.892 *
##     5) Years > 3.5 28  10.13 5.583 *
##   3) Years > 4.5 173  72.71 6.354
##     6) Hits < 117.5 90  28.09 5.998 *
##     7) Hits > 117.5 83  20.88 6.740 *
```

## Bagging (or Bootstrap Aggregation)

```
b=1 # vary b to get different tree
set.seed(b)
sam=sample(c(1:263),263,replace=TRUE) # sampling with replacement for 263 cases
hit.sample=hit2[sam,] # Bootstrapped dataset (use to construct tree)
hit.tree=tree(Salary~Hits+Years,data=hit.sample)
partition.tree(hit.tree)
```




# Random Forest

```
##Example 13 Diving into training-test
set.seed(5227)
sam=sample(c(1:506),506,replace=FALSE)
b1=Boston[sam,]
train=b1[1:253,]
test=b1[254:506,]

##Example 14 Test MSE from CART
boston.CART=tree(medv~.,data=train)
yhat.CART=predict(boston.CART,test)
MSE.CART=mean((yhat.CART-test$medv)^2)
MSE.CART
```

```
## [1] 24.31669
```

```
##Example 15 Test MSE from bagging
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
boston.bag=randomForest(medv~.,data=train,mtry=12,importance=TRUE)
boston.bag
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = train, mtry = 12, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 13.02577
##                     % Var explained: 86.37
```

```
yhat.bag=predict(boston.bag,test)
MSE.bag=mean((yhat.bag-test$medv)^2)
MSE.bag
```

```
## [1] 16.95206
```

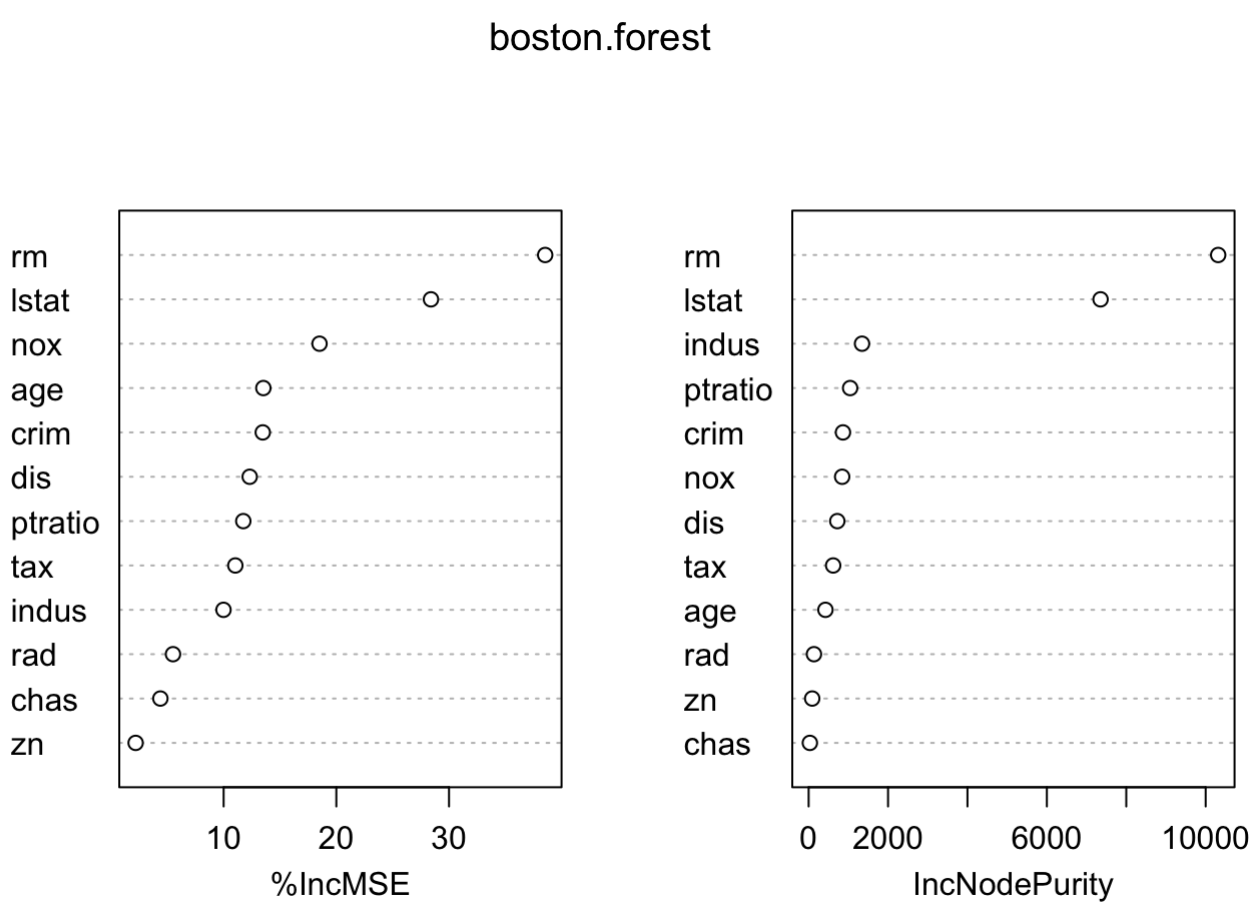
```
##Example 16 Test MSE from random Forest
set.seed(1)
boston.forest=randomForest(medv~.,data=train,mtry=6,importance=TRUE)
yhat.forest=predict(boston.forest,test)
MSE.forest=mean((yhat.forest-test$medv)^2)
MSE.forest
```

```
## [1] 15.71063
```

```
##Example 17 Importance variable output
importance(boston.forest)
```

```
##           %IncMSE IncNodePurity
## crim    13.472048     864.62932
## zn       2.192116      89.69032
## indus    9.987538    1344.71550
## chas     4.386529      30.91888
## nox     18.508327     844.19703
## rm      38.532802   10315.13112
## age     13.523261     422.91749
## dis     12.312679     721.64811
## rad      5.507385     133.81816
## tax     11.037820     617.58630
## ptratio 11.743412    1044.95855
## lstat   28.396637    7352.09040
```

```
varImpPlot(boston.forest)
```




# Boosting

```
##Example 18 Classification Tree
library(kmed)
library(tree)
heart$class=factor(heart$class)
heart.tree=tree(class~.,data=heart)
summary(heart.tree)
```

```
##
## Classification tree:
## tree(formula = class ~ ., data = heart)
## Variables actually used in tree construction:
##  [1] "thal"     "ca"       "age"      "thalach"  "cp"       "chol"
##  [7] "sex"      "trestbps" "oldpeak"  "restecg"
## Number of terminal nodes:  24
## Residual mean deviance:  1.347 = 367.7 / 273
## Misclassification error rate: 0.2862 = 85 / 297
```

```
##Example 19 Preliminaries
set.seed(297)
per=sample(c(1:297),297,replace=FALSE)
rheart=heart[per,]

##Example 20 Test error for CART
error.tree=0
for(k in 1:10){
vec=c((29*(k-1)+1):(29*k))
test=rheart[vec,]
train=rheart[-vec,]
cv.tree=tree(class~.,data=train)
predict.tree=predict(cv.tree,test)
predictclass.tree=apply(predict.tree,1,which.max)-1
error.tree=error.tree+sum(test$class!=predictclass.tree)
}
```

```
#Example 21 Output for Example 20
head(predict.tree)
```

```
##     0         1         2         3         4
## 146 1 0.0000000 0.0000000 0.0000000 0.0000000
## 147 0 0.0000000 0.1111111 0.5555556 0.3333333
## 109 0 0.2500000 0.7500000 0.0000000 0.0000000
## 301 0 0.6666667 0.0000000 0.0000000 0.3333333
## 82  1 0.0000000 0.0000000 0.0000000 0.0000000
## 176 0 0.0000000 0.1111111 0.5555556 0.3333333
```

```
head(predictclass.tree)
```

```
## 146 147 109 301  82 176
##   0   3   2   1   0   3
```

```
head(test$class)
```

```
## [1] 1 4 2 3 0 1
## Levels: 0 1 2 3 4
```

```
error.tree/290
```

```
## [1] 0.4724138
```

```
##Example 22 OOB error for bagging
library(randomForest)
set.seed(1)
heart.bag=randomForest(class~.,data=heart,mtry=13)
heart.bag
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, mtry = 13)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##         OOB estimate of  error rate: 45.45%
## Confusion matrix:
##     0  1  2 3 4 class.error
## 0 143 12  2 3 0   0.1062500
## 1  30  6 10 8 0   0.8888889
## 2  11  9  8 7 0   0.7714286
## 3   3 15 11 5 1   0.8571429
## 4   1  4  2 6 0   1.0000000
```

```
##Example 23 OOB error for random forest
set.seed(1)
heart.forest=randomForest(class~.,data=heart,mtry=6)
heart.forest
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = heart, mtry = 6)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 44.11%
## Confusion matrix:
##     0  1  2  3 4 class.error
## 0 144 11  2  3 0   0.1000000
## 1  33  4  7 10 0   0.9259259
## 2  10  6 11  8 0   0.6857143
## 3   7 12  8  7 1   0.8000000
## 4   2  3  1  7 0   1.0000000
```

```
##Example 24 Preliminaries
library(gbm)
rheart$sex=factor(rheart$sex)
rheart$fbs=factor(rheart$fbs)
rheart$exang=factor(rheart$exang)
set.seed(1)
error.gbm=0

##Example 25 Test error for boosting
for(k in 1:10){
vec=c((29*(k-1)+1):(29*k))
test=rheart[vec,]
train=rheart[-vec,]
cv.gbm=gbm(class~.,data=train,n.trees=5000,
distribution="multinomial",interaction.depth=4)
predict.gbm=predict(cv.gbm,test)
predictclass.gbm=apply(predict.gbm,1,which.max)-1
error.gbm=error.gbm+sum(test$class!=predictclass.gbm)
}
error.gbm/290
```

```
## [1] 0.4689655
```

# Week 8: kNN Regression and Classification

Loading Required Packages:

```
library(kknn)
```

Loading Boston Datset:

```
attach(Boston)
set.seed(5227)
sam=sample(c(1:506),506,replace=FALSE)
b1=Boston[sam,]
train=b1[1:253,]
test=b1[254:506,]
```

```
##Example 1 Applying 3-NN
boston.3nn=kknn(medv~.,train, test, scale=TRUE, distance=2, k=3, kernel="rectangular")
mse.3nn=mean((test$medv-boston.3nn$fitted.values)^2)
mse.3nn
```

```
## [1] 24.99328
```

```
##Example 2 Applying CV to find best k
mse=rep(0,times=12)
for(k in 1:12) for(j in 1:6){
    vec=c(((j-1)*42+1):(j*42))
    cv.test=train[vec,]
    cv.train=train[-vec,]
    boston.knn=kknn(medv~.,cv.train,cv.test,distance=2,k=k,kernel="rectangular")
    mse[k]=mse[k]+mean((cv.test$medv-boston.knn$fitted.values)^2)/6
    }

##Example 3 Looking at CV MSE
mse
```

```
##  [1] 24.82706 23.08860 22.46272 24.16004 24.39941 24.54455 23.77644 23.60605
##  [9] 25.21611 26.42865 26.65353 26.95675
```

```
##Example 4 Applying weighted kNN
boston.10t=kknn(medv~.,train,test,distance=2,k=9,kernel="triangular")
mse.10t=mean((test$medv-boston.10t$fitted.values)^2)
mse.10t
```

```
## [1] 25.7512
```

# Week 9: Unsupervised Learning

## Principal Component Analysis (PCA)

```
##Example 1 Extracting the first 6 data-points
attach(USArrests)
small=USArrests[1:6,]
small
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```
##Example 2 Notations
X=scale(small)
X
```

```
##                Murder    Assault   UrbanPop       Rape
## Alabama     1.8881483 -0.1903805 -0.5318145 -1.0883819
## Alaska      0.2551552  0.4658246 -1.0916193  1.1393749
## Arizona    -0.7144345  1.2192452  0.6997560 -0.1513855
## Arkansas   -0.3572173 -1.3083594 -0.9796584 -1.2509221
## California -0.2551552  0.7817751  1.3155412  0.7664886
## Colorado   -0.8164966 -0.9681050  0.5877950  0.5848260
## attr(,"scaled:center")
##    Murder   Assault  UrbanPop      Rape
##   9.50000 243.83333  67.50000  32.58333
## attr(,"scaled:scale")
##    Murder   Assault  UrbanPop      Rape
##  1.959592 41.145676 17.863370 10.458952
```

```
x1=X[1,]
x1
```

```
##     Murder    Assault   UrbanPop       Rape
##  1.8881483 -0.1903805 -0.5318145 -1.0883819
```

```
X2=X[,2]
X2
```

```
##    Alabama     Alaska    Arizona   Arkansas California   Colorado
## -0.1903805  0.4658246  1.2192452 -1.3083594  0.7817751 -0.9681050
```

```
x34=X[3,4]
x34
```

```
## [1] -0.1513855
```

```
##Example 3 Crime rate scores
v=c(1,1,0,1)/sqrt(3)
z=X%*%v
z
```

```
##                 [,1]
## Alabama     0.3518292
## Alaska      1.0740762
## Arizona     0.2040501
## Arkansas   -1.6838413
## California  0.7465765
## Colorado   -0.6926907
```

```
sum(z^2)
```

```
## [1] 5.191579
```

```
##Example 4 PCA on small dataset
arrests.pca=prcomp(X,scale=FALSE)
arrests.pca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.4347881 0.9970716 0.7971407 0.5583887
##
## Rotation (n x k) = (4 x 4):
##                 PC1         PC2        PC3        PC4
## Murder   -0.4328338  0.72647320 -0.0504191 -0.5313657
## Assault   0.4607234  0.67740933 -0.1151443  0.5617761
## UrbanPop  0.5577932 -0.07800317 -0.6590192 -0.4984737
## Rape      0.5378249  0.08525726  0.7415480 -0.3918956
```

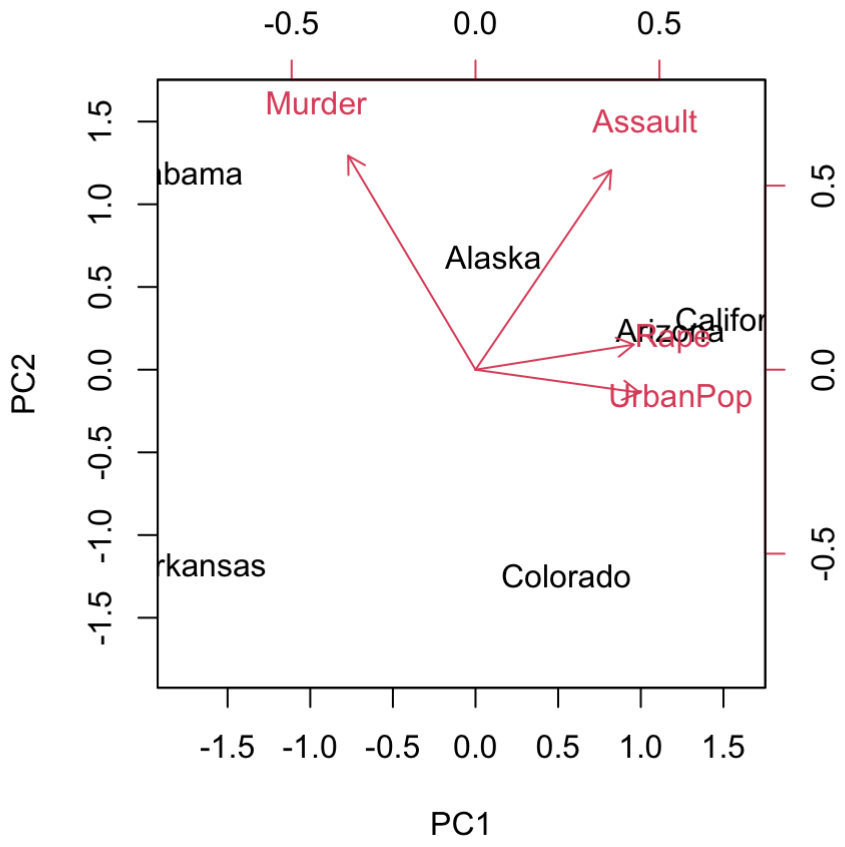
```
##Example 5 z-scores of PCA
arrests.pca$x
```

```
##                  PC1        PC2        PC3        PC4
## Alabama    -1.7869686  1.1914144 -0.5298889 -0.4186208
## Alaska      0.1080629  0.6832071  1.4977975  0.2237358
## Arizona     1.1798664  0.2394206 -0.6777802  0.7750861
## Arkansas   -1.6673999 -1.1760374 -0.1133444  0.4333727
## California  1.6166584  0.3069507 -0.3757311 -0.3813829
## Colorado    0.5497807 -1.2449554  0.1989471 -0.6321910
```

```
##Example 6 computing z-scores using matrix multiplication
X%*%arrests.pca$rotation[,1]
```

```
##                 [,1]
## Alabama    -1.7869686
## Alaska      0.1080629
## Arizona     1.1798664
## Arkansas   -1.6673999
## California  1.6166584
## Colorado    0.5497807
```

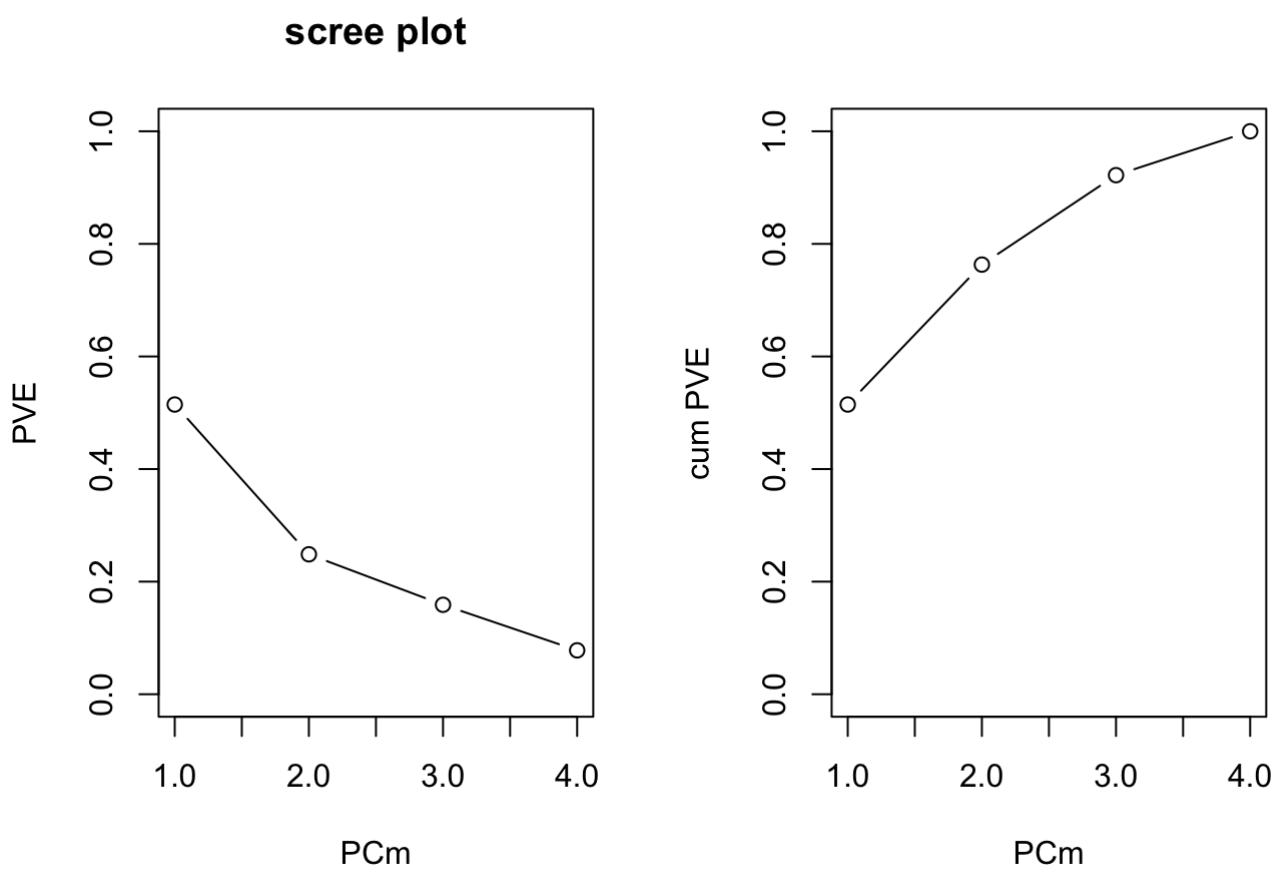
```
##Example 7 biplot
biplot(arrests.pca,scale=0)
```




```
##Example 8 PVE and cumulative PVE
imp=summary(arrests.pca)$importance
imp
```

```
##                             PC1       PC2       PC3       PC4
## Standard deviation     1.434788 0.9970716 0.7971407 0.5583887
## Proportion of Variance 0.514650 0.2485400 0.1588600 0.0779500
## Cumulative Proportion  0.514650 0.7631900 0.9220500 1.0000000
```

```
##Example 9 scree plot
par(mfrow=c(1,2))
plot(imp[2,],xlab="PCm",ylab="PVE",ylim=c(0,1),main="scree plot",type="b")
plot(imp[3,],xlab="PCm",ylab="cum PVE",ylim=c(0,1),type="b")
```




```
##Example 10 correlation matrix
A=t(X)%*%X
A
```

```
##              Murder     Assault  UrbanPop      Rape
## Murder    5.00000000 -0.05333087 -2.248256 -1.882385
## Assault  -0.05333087  5.00000000  2.187073  2.223084
## UrbanPop -2.24825578  2.18707349  5.000000  1.806702
## Rape     -1.88238504  2.22308391  1.806702  5.000000
```

```
##Example 11 eigendecomposition
A.eig=eigen(t(X)%*%X)
A.eig
```

```
## eigen() decomposition
## $values
## [1] 10.293085  4.970759  3.177166  1.558990
##
## $vectors
##            [,1]        [,2]       [,3]       [,4]
## [1,]  0.4328338  0.72647320 -0.0504191  0.5313657
## [2,] -0.4607234  0.67740933 -0.1151443 -0.5617761
## [3,] -0.5577932 -0.07800317 -0.6590192  0.4984737
## [4,] -0.5378249  0.08525726  0.7415480  0.3918956
```

```
##Example 12 checking the matrix relations
Q=A.eig$vectors
Lambda=diag(A.eig$values)
Q%*%t(Q)
```

```
##               [,1]         [,2]          [,3]          [,4]
## [1,]  1.000000e+00 2.053464e-16 -3.786028e-17 -2.349037e-17
## [2,]  2.053464e-16 1.000000e+00  1.648098e-17  1.136170e-16
## [3,] -3.786028e-17 1.648098e-17  1.000000e+00 -3.765190e-16
## [4,] -2.349037e-17 1.136170e-16 -3.765190e-16  1.000000e+00
```

```
Q%*%Lambda%*%t(Q)
```

```
##             [,1]        [,2]      [,3]      [,4]
## [1,]  5.00000000 -0.05333087 -2.248256 -1.882385
## [2,] -0.05333087  5.00000000  2.187073  2.223084
## [3,] -2.24825578  2.18707349  5.000000  1.806702
## [4,] -1.88238504  2.22308391  1.806702  5.000000
```

```
sqrt(A.eig$values/5)
```

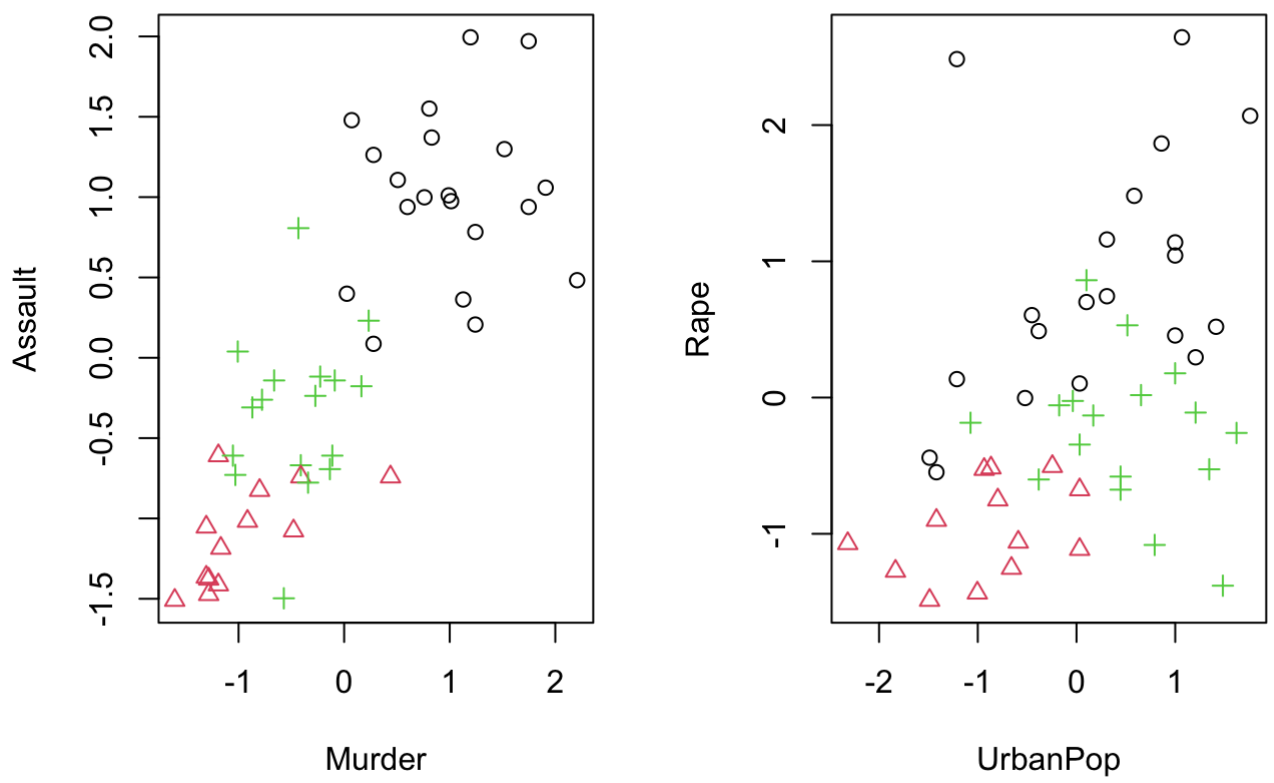
```
## [1] 1.4347881 0.9970716 0.7971407 0.5583887
```

## Clustering Methods: K-Means

```
##Example 13 Applying kmeans
us.scale=scale(USArrests)
set.seed(5227)
us.kmeans=kmeans(us.scale,centers=3,nstart=20)
us.kmeans
```

```
## K-means clustering with 3 clusters of sizes 20, 13, 17
##
## Cluster means:
##       Murder    Assault   UrbanPop       Rape
## 1  1.0049340  1.0138274  0.1975853  0.8469650
## 2 -0.9615407 -1.1066010 -0.9301069 -0.9667633
## 3 -0.4469795 -0.3465138  0.4788049 -0.2571398
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              3              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              1              3              3              1              1
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              3              2              1              3              2
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              3              2              1              2              1
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##              3              1              2              1              1
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##              2              2              1              2              3
##     New Mexico       New York North Carolina   North Dakota           Ohio
##              1              1              1              2              3
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##              3              3              3              3              1
##   South Dakota      Tennessee          Texas           Utah        Vermont
##              2              1              1              3              2
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##              3              3              2              2              3
##
## Within cluster sum of squares by cluster:
## [1] 46.74796 11.95246 19.62285
##  (between_SS / total_SS =  60.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

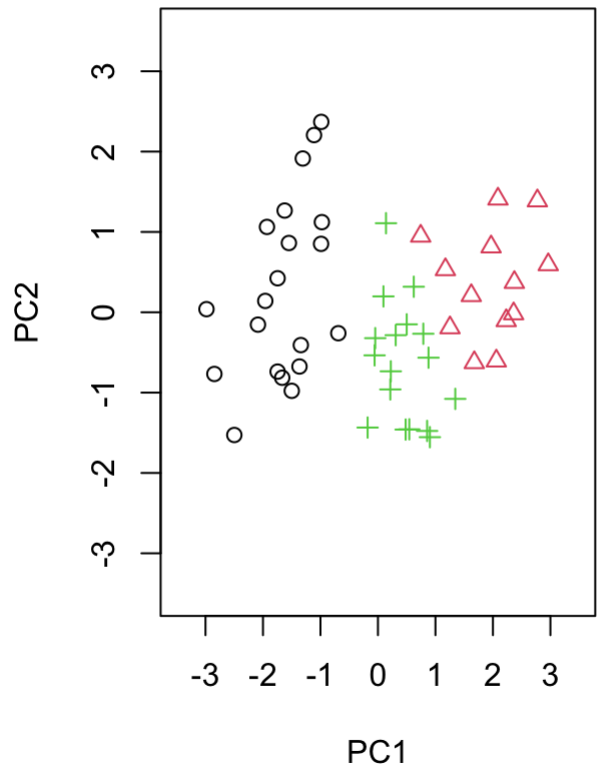
```
##Example 14 Plotting clusters with features as axes
par(mfrow=c(1,2))
plot(us.scale[,1],us.scale[,2],col=us.kmeans$cluster,
xlab="Murder",ylab="Assault",pch=us.kmeans$cluster)
plot(us.scale[,3],us.scale[,4],col=us.kmeans$cluster,
xlab="UrbanPop",ylab="Rape",pch=us.kmeans$cluster)
```




```
##Example 15 Plotting clusters with PC1-PC2 axes
us.pca=prcomp(us.scale,scale=FALSE)
us.pca
```

```
## Standard deviations (1, .., p=4):
## [1] 1.5748783 0.9948694 0.5971291 0.4164494
##
## Rotation (n x k) = (4 x 4):
##                 PC1        PC2        PC3         PC4
## Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
## Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
## UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
## Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

```
PC1=us.pca$x[,1]
PC2=us.pca$x[,2]
plot(PC1,PC2,col=us.kmeans$cluster,pch=us.kmeans$cluster,
xlim=c(-3.5,3.5),ylim=c(-3.5,3.5))
```

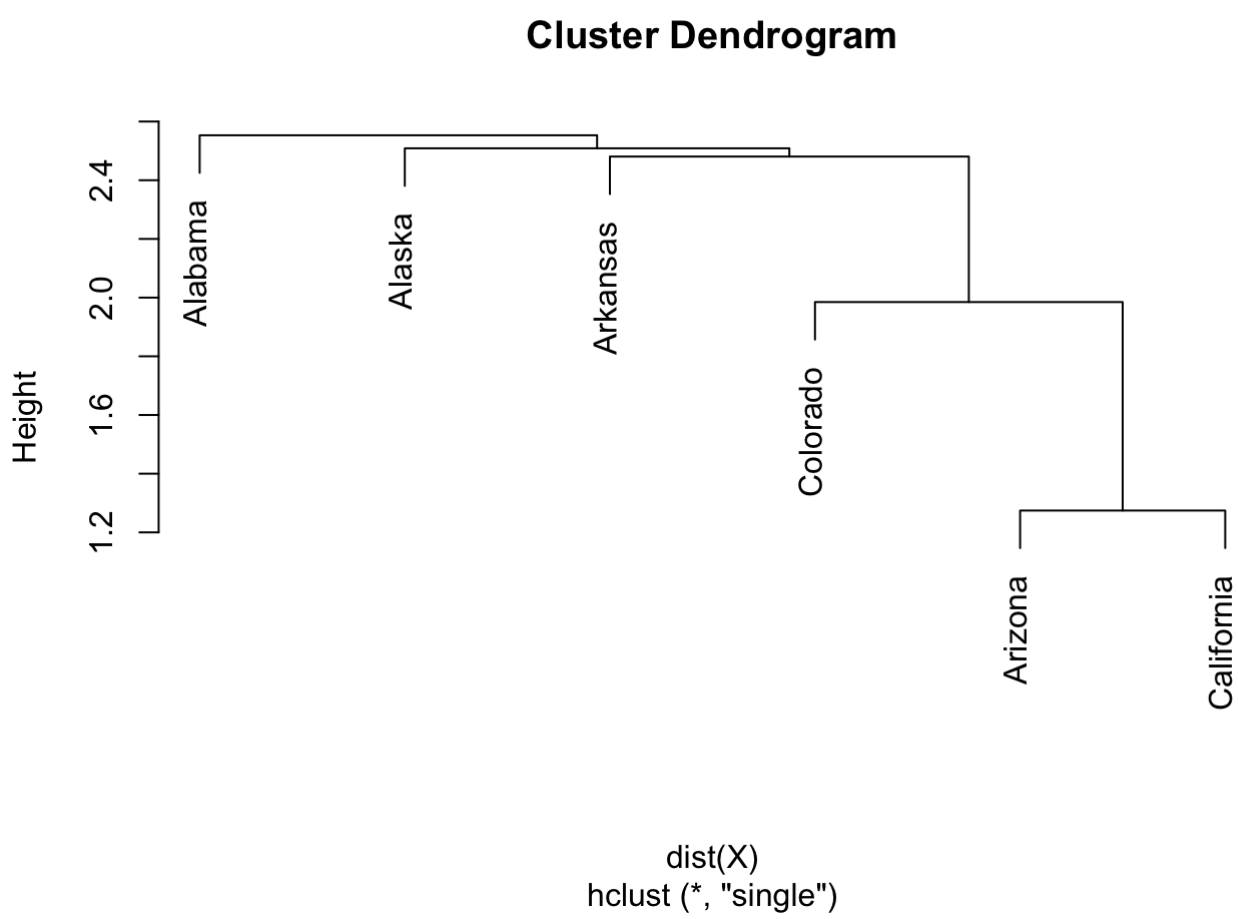



## Clustering Methods: Hierarchical clustering (HC)

```
##Example 16 HC of small dataset
small.hc=hclust(dist(X),method="single")
dist(X)
```

```
##             Alabama   Alaska  Arizona Arkansas California
## Alaska     2.893709
## Arizona    3.339942 2.526427
## Arkansas   2.553141 3.041181 3.247430
## California 3.520242 2.508726 1.274363 3.703642
## Colorado   3.460218 2.516456 2.312890 2.480643   1.984893
```

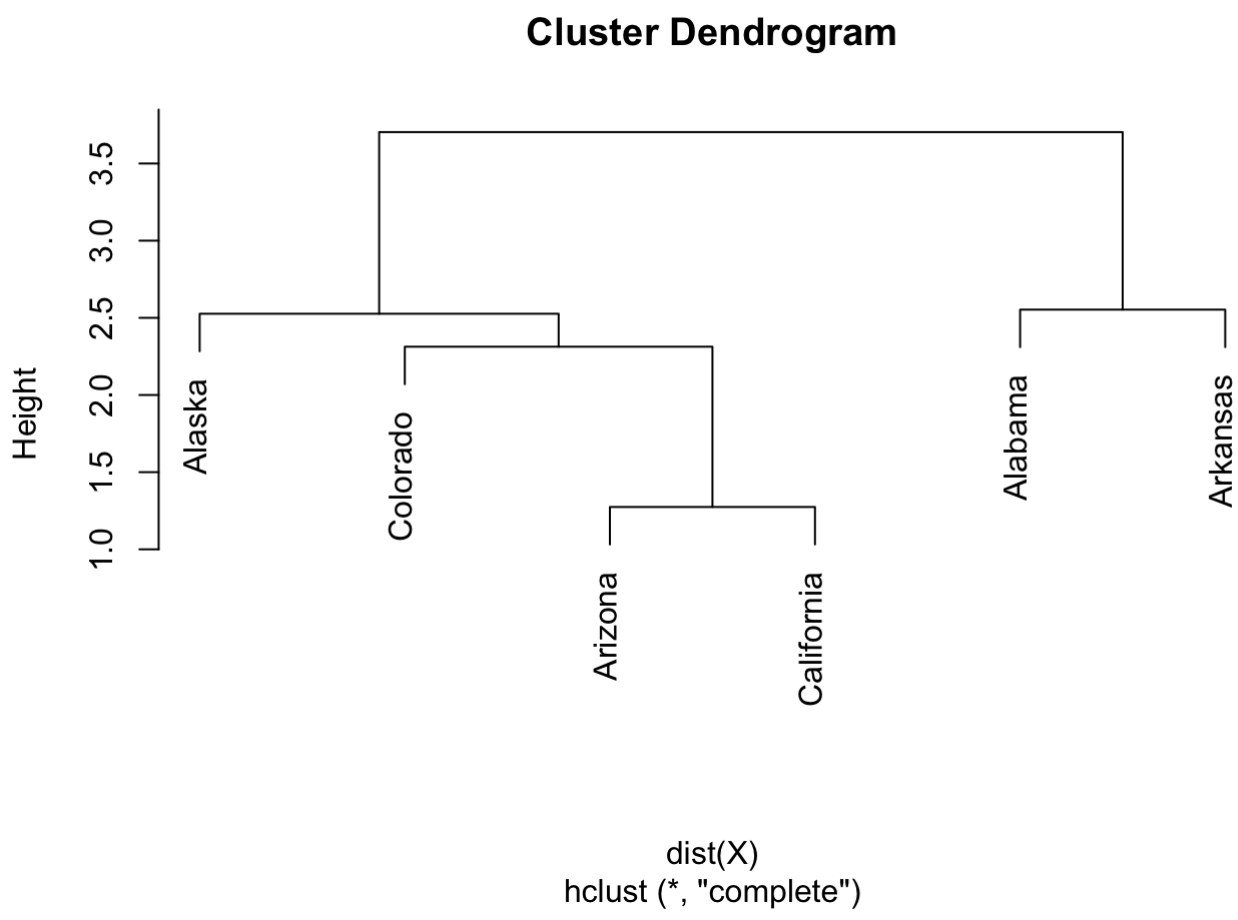
```
plot(small.hc)
```




```
##Example 17 cutree
cutree(small.hc,4)
```

```
##    Alabama     Alaska    Arizona   Arkansas California   Colorado
##          1          2          3          4          3          3
```

```
##Example 18 Complete linkage
small.complete=hclust(dist(X),method="complete")
plot(small.complete)
```




```
##Example 19 Computing correlation distance
X3=matrix(nrow=3,ncol=7)
X3[1,]=c(1,1,0,0,0,0,0)
X3[2,]=c(0,0,1,0,0,0,0)
X3[3,]=c(1,1,0,1,1,1,1)
X3.cor=cor(t(X3),method="pearson")
#Note that we use t(X) instead of X as we want to
#measure correlation between data-points and not features
X3.dcor=(1-X3.cor)/2
X3.dcor
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.0000000 0.6290994 0.3709006
## [2,] 0.6290994 0.0000000 1.0000000
## [3,] 0.3709006 1.0000000 0.0000000
```

# NCI60 Dataset

```
##Example 20 NCI60 data
attach(NCI60)
nci.data=NCI60$data
nci.labs=NCI60$labs
dim(nci.data)
```

```
## [1]   64 6830
```

```
length(nci.labs)
```

```
## [1] 64
```

```
head(nci.labs)
```

```
## [1] "CNS"    "CNS"    "CNS"    "RENAL"  "BREAST" "CNS"
```

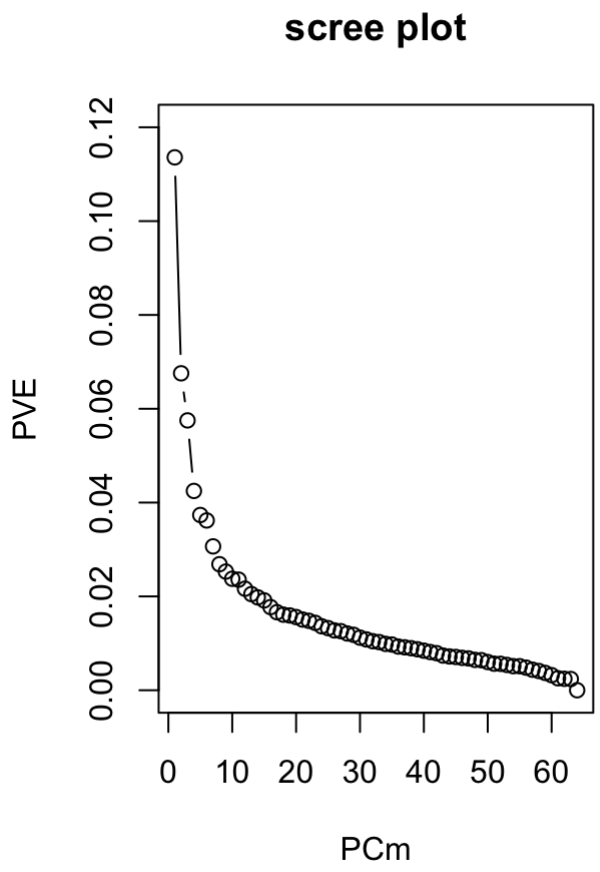
```
nci.abb=abbreviate(nci.labs)
table(nci.abb)
```

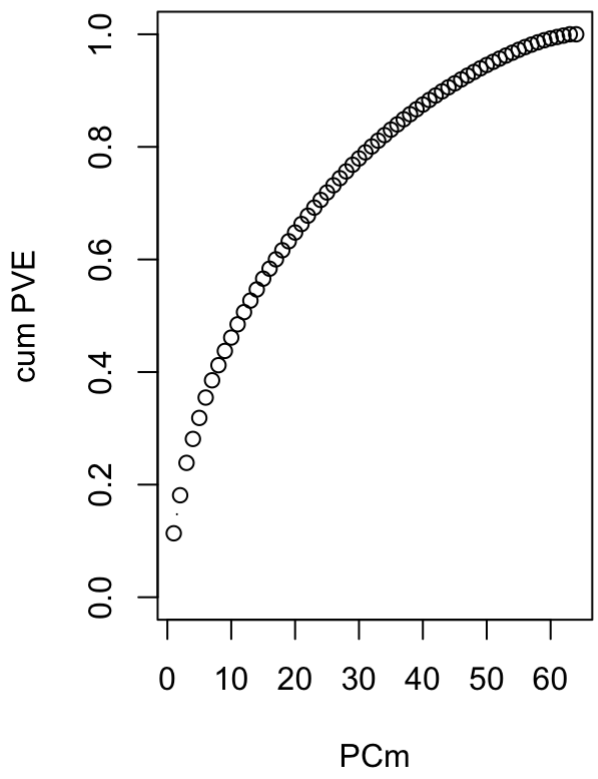
```
## nci.abb
##  BREA   CNS  COLO K562A K562B  LEUK MCF7A MCF7D  MELA  NSCL  OVAR  PROS  RENA
##     7     5     7     1     1     6     1     1     8     9     6     2     9
##  UNKN
##     1
```

```
##Example 21 HC without PCA
nci.sd=scale(nci.data)
nci.hc=hclust(dist(nci.sd),method="complete")
nci.k4=cutree(nci.hc,k=4)
table(nci.k4,nci.abb)
```

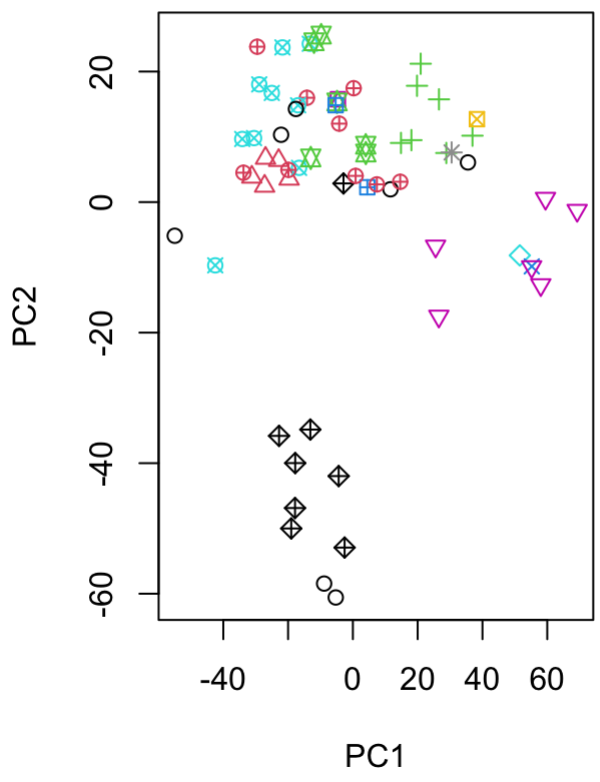
```
##       nci.abb
## nci.k4 BREA CNS COLO K562A K562B LEUK MCF7A MCF7D MELA NSCL OVAR PROS RENA UNKN
##      1    2   3    2     0     0    0     0     0    8    8    6    2    8    1
##      2    3   2    0     0     0    0     0     0    0    1    0    0    1    0
##      3    0   0    0     1     1    6     0     0    0    0    0    0    0    0
##      4    2   0    5     0     0    0     1     1    0    0    0    0    0    0
```

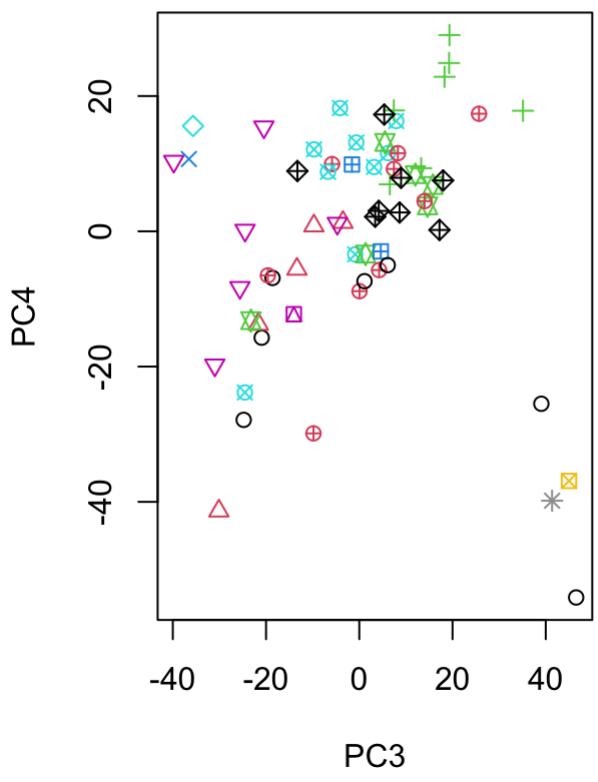
```
##Example 22 Doing PCA on NCI60 data
nci.pca=prcomp(nci.sd,scale=FALSE)
imp=summary(nci.pca)$importance
par(mfrow=c(1,2))
plot(imp[2,],xlab="PCm",ylab="PVE",ylim=c(0,0.12),main="scree plot",type="b")
plot(imp[3,],xlab="PCm",ylab="cum PVE",ylim=c(0,1),type="b")
```




```
##Example 23 Plotting cancer types on PC
PC1=nci.pca$x[,1]
PC2=nci.pca$x[,2]
PC3=nci.pca$x[,3]
PC4=nci.pca$x[,4]
cancer=as.numeric(factor(nci.abb))
par(mfrow=c(1,2))
plot(PC1,PC2,col=cancer,pch=cancer)
plot(PC3,PC4,col=cancer,pch=cancer)
```




```
##Example 24
nci.hc4=hclust(dist(nci.pca$x[,1:4]),method="complete")
nci.hc4k4=cutree(nci.hc4,k=4)
table(nci.hc4k4,nci.abb)
```

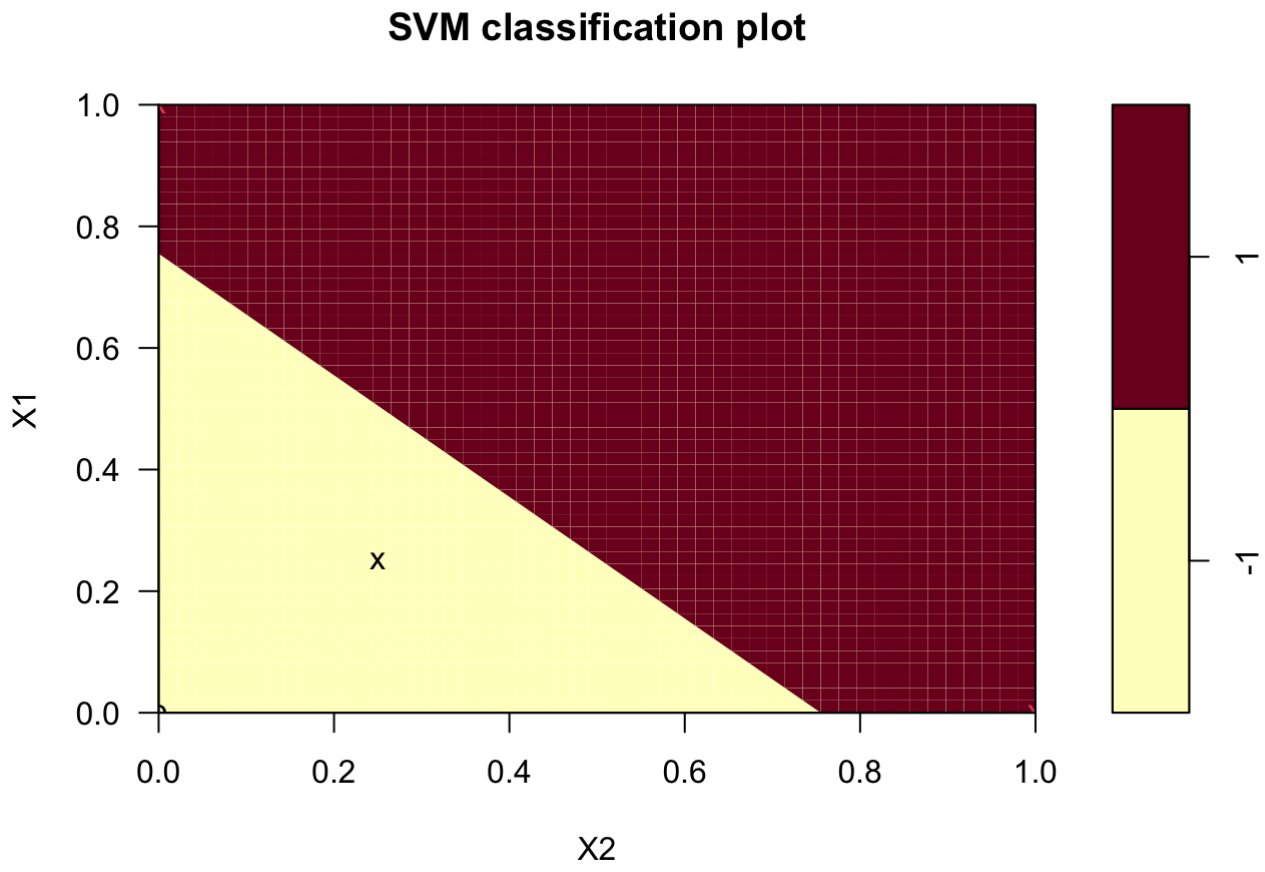
```
##          nci.abb
## nci.hc4k4 BREA CNS COLO K562A K562B LEUK MCF7A MCF7D MELA NSCL OVAR PROS RENA
##         1    5   5    0     0     0    0     0     0    7    2    1    0    2
##         2    0   0    7     0     0    2     0     0    1    7    5    2    7
##         3    0   0    0     1     1    4     0     0    0    0    0    0    0
##         4    2   0    0     0     0    0     1     1    0    0    0    0    0
##          nci.abb
## nci.hc4k4 UNKN
##         1    1
##         2    0
##         3    0
##         4    0
```

# Week 9: Support Vector Machines (SVM)

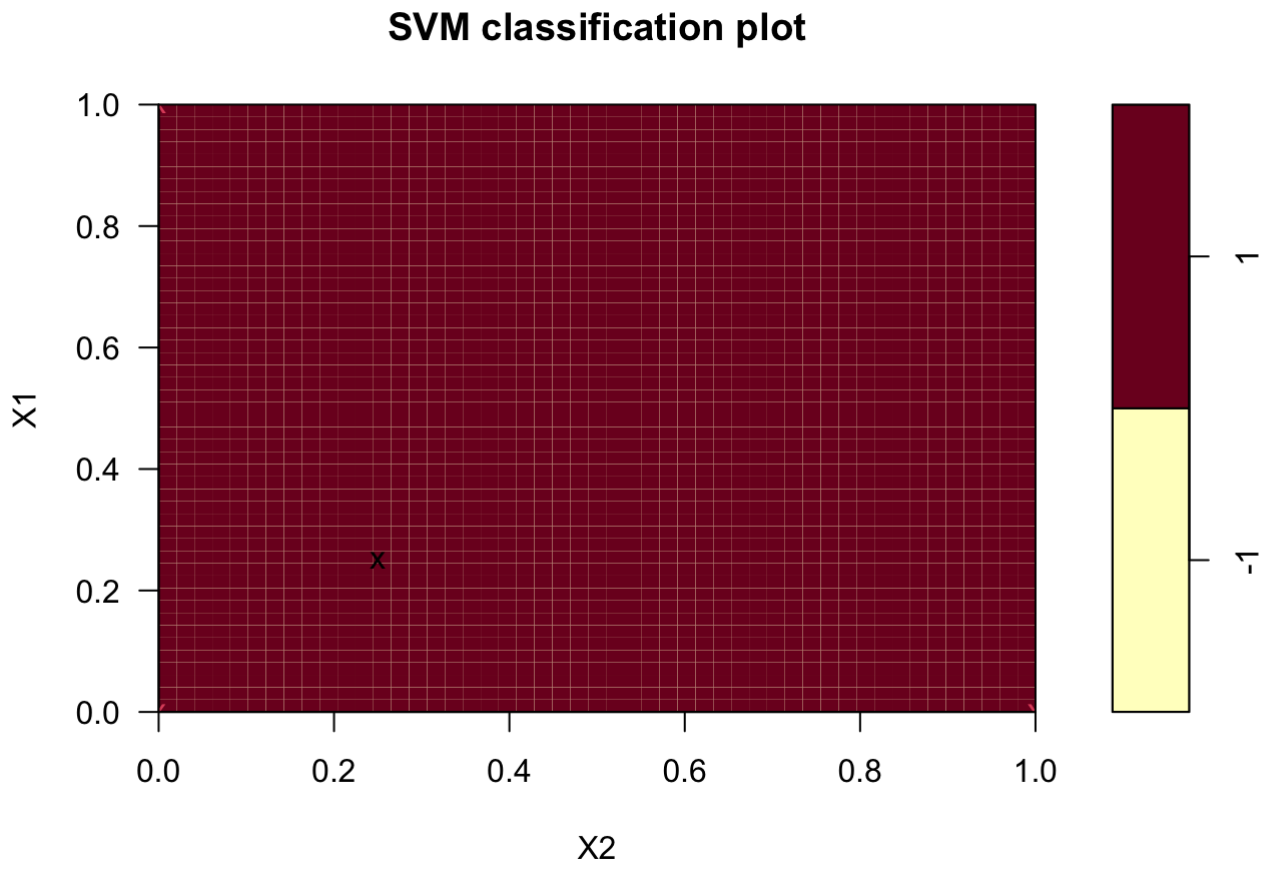
```
##Example 1 Creating datasets
library(e1071)
X=matrix(nrow=4,ncol=2)
X1=c(0,0.25,1,0)
X2=c(0,0.25,0,1)
y1=c(-1,-1,1,1)
y2=c(1,-1,1,1)
sep=data.frame(X1=X1,X2=X2,y=as.factor(y1))
notsep=data.frame(X1=X1,X2=X2,y=as.factor(y2))

##Example 2 Applying linear SVM on sep
svm.sep=svm(y~.,data=sep,scale=FALSE,cost=1000,kernel="linear")
summary(svm.sep)
```

```
##
## Call:
## svm(formula = y ~ ., data = sep, cost = 1000, kernel = "linear",
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1000
##
## Number of Support Vectors:  3
##
##  ( 1 2 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

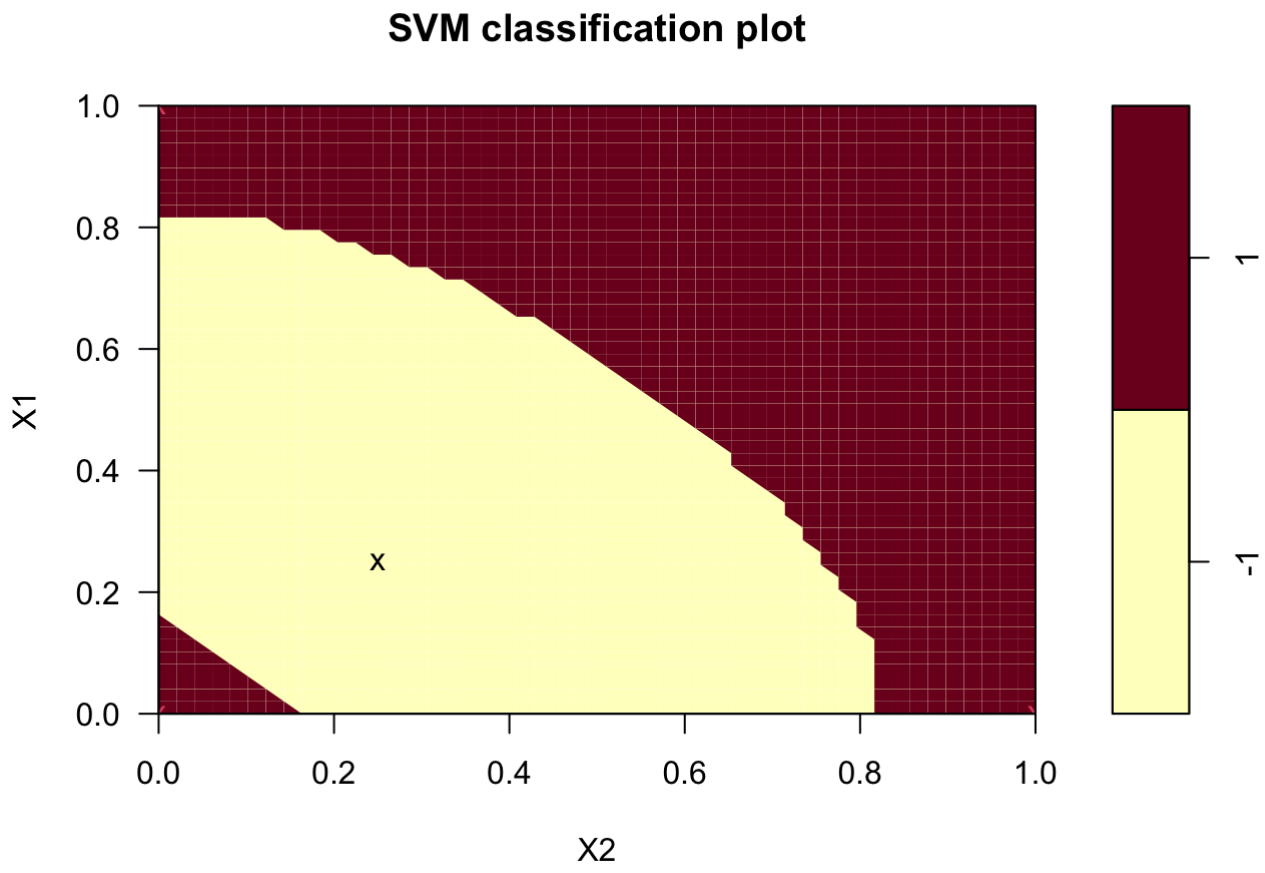
```
##Example 3
plot(svm.sep,data=sep)
```



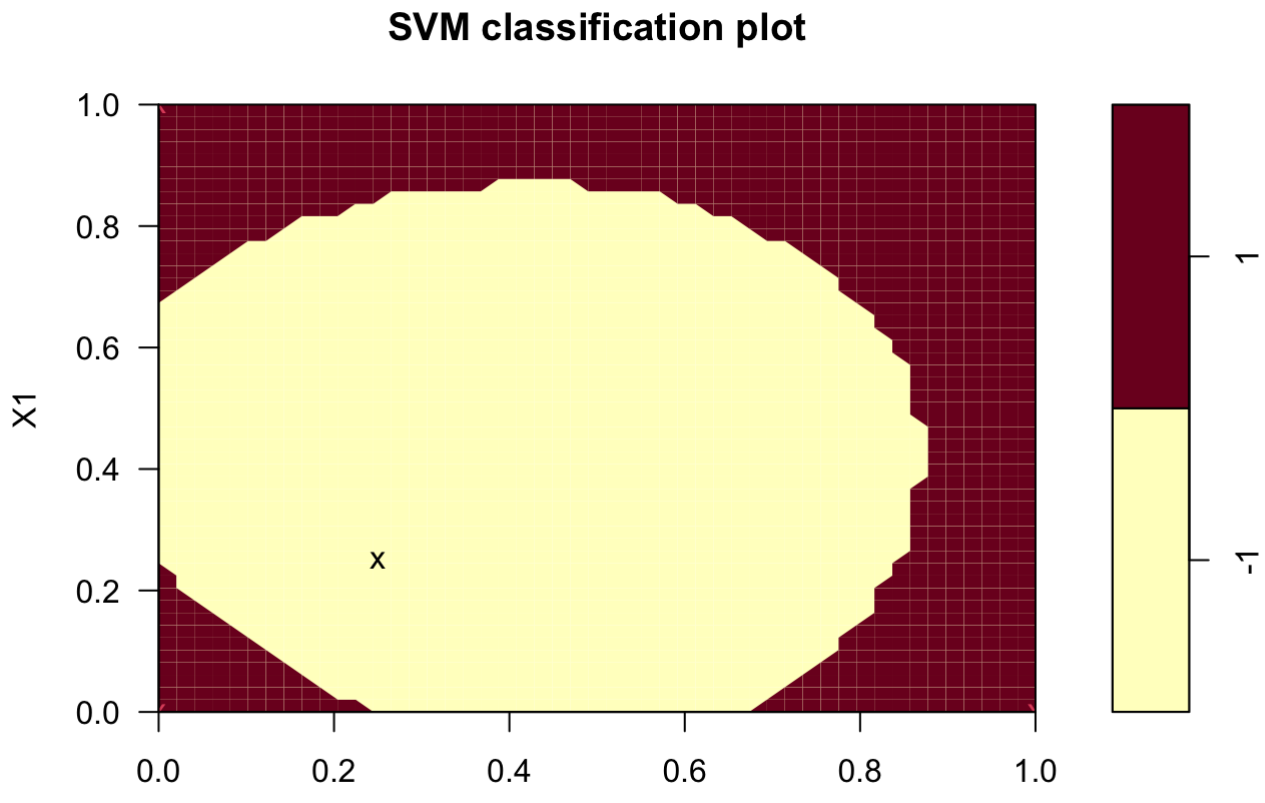

```
##Example 4 Applying linear SVM on notsep
svm.linear=svm(y~.,data=notsep,scale=FALSE,cost=1000,kernel="linear")
plot(svm.linear,data=notsep)
```




```
##Example 5 Quadratic kernel
svm.poly=svm(y~.,data=notsep,scale=TRUE,cost=1000,kernel="polynomial",degree=2)
plot(svm.poly,data=notsep)
```




```
##Example 6 Radial basis kernel
svm.radial=svm(y~.,data=notsep,scale=TRUE,cost=1000,kernel="radial")
plot(svm.radial,data=notsep)
```




```
##Subsampling from original dataset
##fraud=read.csv("D:/Data/creditcard_csv.csv")
##positive=fraud[fraud$Class=="'1'",]
##negative=fraud[fraud$Class=="'0'",]
##select=sample(c(1:284315),508,replace=FALSE)
##negative.sample=negative[select,]
##fraud.sample=rbind(positive,negative.sample)
##fraud.new=fraud.sample[,-1]
##write.table(fraud.new,"D:/Data/fraud.txt")
```

```
##Example 7 Visualizing the dataset
fraud=read.table("fraud.txt")
head(fraud[,c(1:3,29,30)])
```

```
##                V1        V2         V3 Amount Class
## 542  -2.312226542  1.951992 -1.6098507   0.00   '1'
## 624  -3.043540624 -3.157307  1.0884628 529.00   '1'
## 4921 -2.303349568  1.759247 -0.3597447 239.93   '1'
## 6109 -4.397974442  1.358367 -2.5928442  59.00   '1'
## 6330  1.234235046  3.019740 -4.3045969   1.00   '1'
## 6332  0.008430365  4.137837 -6.2406966   1.00   '1'
```

```
table(fraud$Class)
```

```
##
## '0' '1'
## 508 492
```

```
##Example 8 Processing the dataset
fraud$Amount=log(1+fraud$Amount)
fraud$Class=(fraud$Class=="'1'")-(fraud$Class=="'0'")
fraud$Class=as.factor(fraud$Class)
set.seed(5227)
v=sample(c(1:1000),500,replace=FALSE)
train=fraud[v,]
test=fraud[-v,]

##Example 9 Applying SVM on training set
fraud.svm=svm(Class~.,data=train,scale=FALSE)
summary(fraud.svm)
```

```
##
## Call:
## svm(formula = Class ~ ., data = train, scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  287
##
##  ( 90 197 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
##Prediction on test set
fraud.predict=predict(fraud.svm,test)
head(fraud.predict)
```

```
##  542  624 6330 6332 6335 6337
##    1    1    1    1    1    1
## Levels: -1 1
```

```
table(test$Class,fraud.predict)
```

```
##     fraud.predict
##       -1   1
##   -1 258  14
##   1   19 209
```