CSE – 4020 – Machine Learning

Name: Priyank Kumar

Reg. No: 16BCE2031

Slot: L59 + L60

Lab Assignment – 5

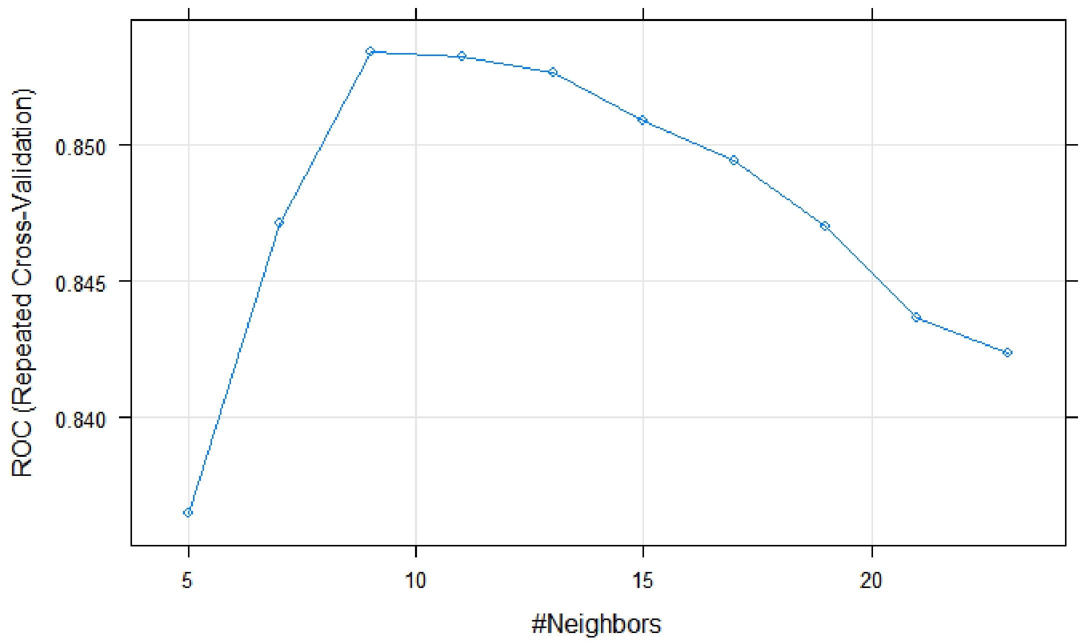
1. Implement kNN(k-nearest neighbors) in R for classification(Consider binary class of predictors of any data sets of your choice)

Soln.

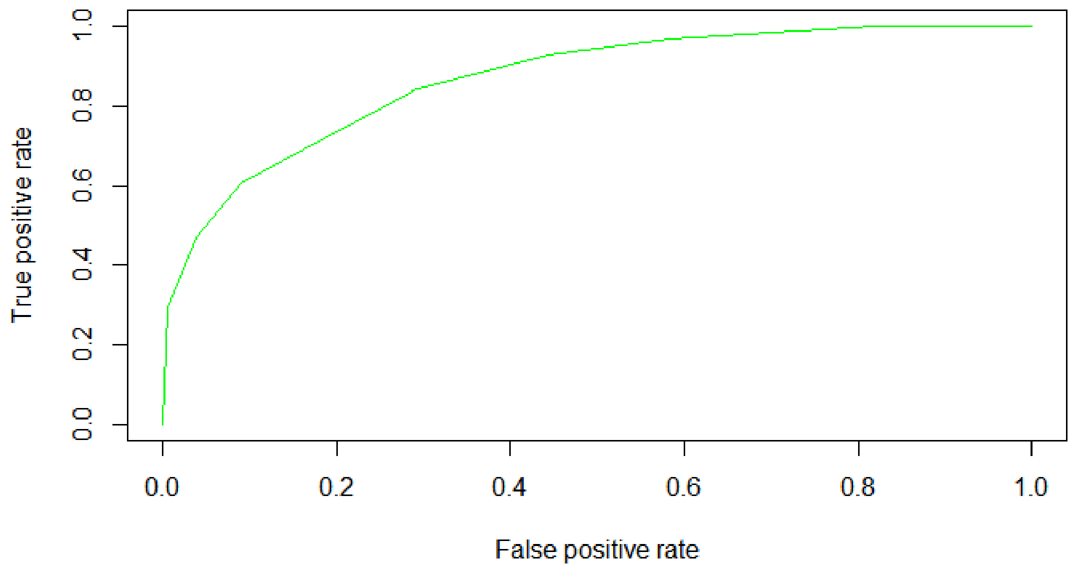
```
> # Read data
> # load library
> library(caret)
> library(e1071)
> # Transforming the dependent variable to a factor
> data1$Win.Loss = as.factor(data1$Win.Loss)
> #Partitioning the data into training and validation data
> set.seed(101)
> index = createDataPartition(data1$Win.Loss, p = 0.7, list = F )
> train = data1[index,]
> validation = data1[-index,]
> # Explore data
> dim(train)
[1] 1068   14
> dim(validation)
[1] 456  14
> names(train)
 [1] "Win.Loss"        "Optimism"        "Pessimism"       "PastUsed"        "FutureUsed"
"PresentUsed"     "OwnPartyCount"  "OppPartyCount"
 [9] "NumericContent" "Extra"           "Emoti"           "Agree"           "Consc"
"Openn"
> head(train)
   Win.Loss   Optimism  Pessimism  PastUsed FutureUsed PresentUsed OwnPartyCount
OppPartyCount NumericContent Extra Emoti Agree Consc Openn
1         1 0.10450450 0.05045045 0.4381443  0.4948454  0.06701031             2
2     0.001877543 4.041 4.049 3.469 2.450 2.548
3         1 0.11257190 0.04930156 0.4159664  0.5168067  0.06722689             1
1     0.002131163 3.463 4.039 3.284 2.159 2.465
5         1 0.10582640 0.05172414 0.3342618  0.5821727  0.08356546             3
4     0.002229220 4.658 4.023 3.283 2.415 2.836
7         1 0.09838275 0.06401617 0.3240741  0.6018519  0.07407407             6
4     0.002251985 3.727 4.108 3.357 2.128 2.231
9         1 0.10610734 0.04688464 0.3633540  0.5372671  0.09937888             2
5     0.002446440 4.119 4.396 3.661 2.572 2.599
10        1 0.10066128 0.05951506 0.3554817  0.5382060  0.10631229             1
2     0.002107436 3.800 4.501 3.624 2.117 2.154
```

```
> head(validation)
   Win.Loss   Optimism  Pessimism  PastUsed FutureUsed PresentUsed OwnPartyCount
OppPartyCount NumericContent Extra Emoti Agree Consc Openn
2          1 0.11457521 0.05923617 0.2912621  0.6213592  0.08737864             1
4     0.001418909 3.446 3.633 3.528 2.402 2.831
4          1 0.10723350 0.04631980 0.4634921  0.4666667  0.06984127             1
3     0.001871715 4.195 4.661 4.007 2.801 3.067
6          1 0.07586207 0.03448276 0.2800000  0.5200000  0.20000000             0
0     0.003290827 2.843 3.563 3.075 1.769 1.479
8          1 0.10377924 0.05638872 0.3692722  0.5498652  0.08086253             2
4     0.002215028 4.027 4.631 3.920 2.417 2.291
17         1 0.11289199 0.05505227 0.3891051  0.5214008  0.08949416             2
7     0.001165647 4.086 4.173 3.368 2.348 2.412
21         1 0.11466373 0.03858875 0.2736842  0.6210526  0.10526316             1
7     0.003105161 3.770 3.858 2.874 1.949 2.006
> # Setting levels for both training and validation data
> levels(train$Win.Loss) <- make.names(levels(factor(train$Win.Loss)))
> levels(validation$Win.Loss) <- make.names(levels(factor(validation$Win.Loss)))
> # Setting levels for both training and validation data
> levels(train$Win.Loss) <- make.names(levels(factor(train$Win.Loss)))
> levels(validation$Win.Loss) <- make.names(levels(factor(validation$Win.Loss)))
> # Setting up train controls
> repeats = 3
> numbers = 10
> tunel = 10
> set.seed(1234)
> x = trainControl(method = "repeatedcv",
+                  number = numbers,
+                  repeats = repeats,
+                  classProbs = TRUE,
+                  summaryFunction = twoClassSummary)
> model1 <- train(Win.Loss~. , data = train, method = "knn",
+                 preProcess = c("center","scale"),
+                 trControl = x,
+                 metric = "ROC",
+                 tuneLength = tunel)
> # Summary of model
> model1
k-Nearest Neighbors

1068 samples
  13 predictor
   2 classes: 'X0', 'X1'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 962, 961, 961, 961, 961, 961, ...
Resampling results across tuning parameters:

  k   ROC        Sens       Spec
   5  0.8364872  0.6900890  0.8412665
   7  0.8471507  0.6684475  0.8494250
   9  0.8534144  0.6587689  0.8525019
  11  0.8532324  0.6540457  0.8602020
  13  0.8526851  0.6531940  0.8683994
  15  0.8509041  0.6491096  0.8607071
  17  0.8494333  0.6411537  0.8560995
  19  0.8470142  0.6267325  0.8612432
  21  0.8436754  0.6146922  0.8637529
  23  0.8423458  0.6042973  0.8714375

ROC was used to select the optimal model using the largest value.
```

```
The final value used for the model was k = 9.
> plot(model1)
```




```
> # Validation
> valid_pred <- predict(model1,validation, type = "prob")
> #Storing Model Performance Scores
> library(ROCR)
> pred_val <-prediction(valid_pred[,2],validation$Win.Loss)
> # Calculating Area under Curve (AUC)
> perf_val <- performance(pred_val,"auc")
> perf_val
An object of class "performance"
Slot "x.name":
[1] "None"

Slot "y.name":
[1] "Area under the ROC curve"

Slot "alpha.name":
[1] "none"

Slot "x.values":
list()

Slot "y.values":
[[1]]
[1] 0.8670378


Slot "alpha.values":
list()

> # Plot AUC
> perf_val <- performance(pred_val, "tpr", "fpr")
> plot(perf_val, col = "green", lwd = 1.5)
```



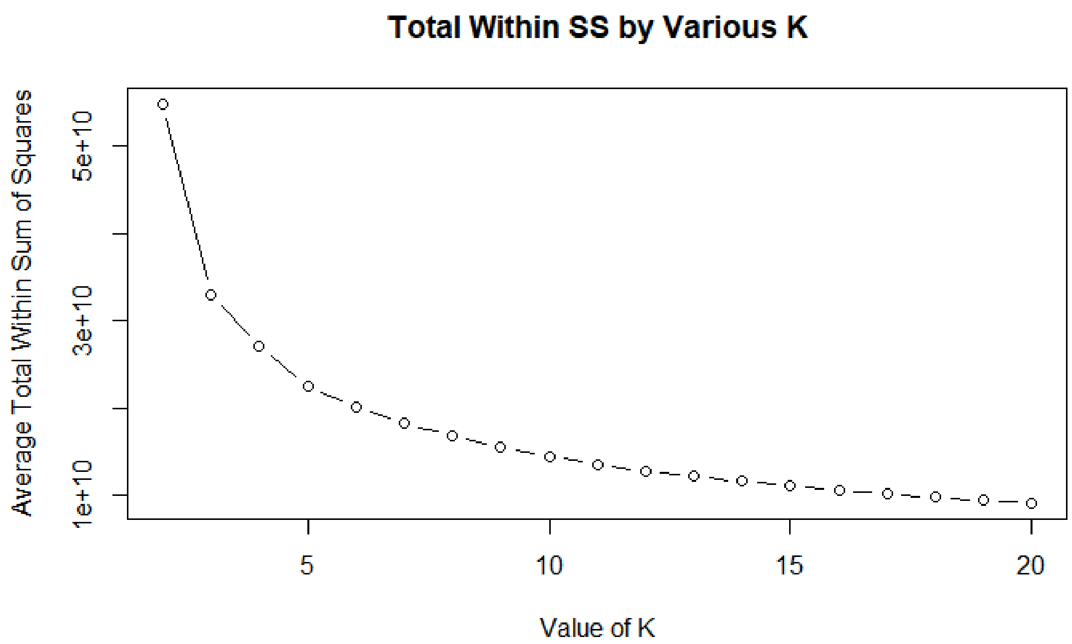
3. Implement K-means clustering? (Consider any clustering data set from internet except iris data)

Soln:

Dataset: Wholesale Customers - UCI

```
> summary(data)
    Channel          Region          Fresh             Milk          Grocery          Frozen         
Detergents_Paper    Delicassen      
 Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55   Min.   :    3   Min.   :   
25.0   Min.   :    3.0   Min.   :    3.0  
 1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533   1st Qu.: 2153   1st Qu.:  
742.2   1st Qu.:  256.8   1st Qu.:  408.2  
 Median :1.000   Median :3.000   Median :  8504   Median : 3627   Median : 4756   Median : 
1526.0   Median :  816.5   Median :  965.5  
 Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796   Mean   : 7951   Mean   : 
3071.9   Mean   : 2881.5   Mean   : 1524.9  
 3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190   3rd Qu.:10656   3rd Qu.: 
3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2  
 Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498   Max.   :92780   Max.   :
60869.0   Max.   :40827.0   Max.   :47943.0  
> top.n.custs <- function (data,cols,n=5) { #Requires some data frame and the top N to remove
+   idx.to.remove <-integer(0) #Initialize a vector to hold customers being removed
+   for (c in cols){ # For every column in the data we passed to this function
+     col.order <-order(data[,c],decreasing=T) #Sort column "c" in descending order (bigger 
on top)
+     #Order returns the sorted index (e.g. row 15, 3, 7, 1, ...) rather than the actual 
values sorted.
+     idx <-head(col.order, n) #Take the first n of the sorted column C to
+     idx.to.remove <-union(idx.to.remove,idx) #Combine and de-duplicate the row ids that 
need to be removed
+   }
```

```
+   return(idx.to.remove) #Return the indexes of customers to be removed
+ }
> top.custs <-top.n.custs(data,cols=3:8,n=5)
> length(top.custs) #How Many Customers to be Removed?
[1] 19
> data[top.custs,] #Examine the customers
    Channel Region  Fresh  Milk Grocery Frozen Detergents_Paper Delicassen
182       1      3 112151 29627   18148  16745             4948       8550
126       1      3  76237  3473    7102  16538              778        918
285       1      3  68951  4411   12609   8692              751       2406
40        1      3  56159   555     902  10002              212       2916
259       1      1  56083  4563    2124   6422              730       3321
87        2      3  22925 73498   32114    987            20070        903
48        2      3  44466 54259   55571   7782            24171       6465
86        2      3  16117 46197   92780   1026            40827       2944
184       1      3  36847 43950   20170  36534              239      47943
62        2      3  35942 38369   59598   3254            26701       2017
334       2      2   8565  4980   67298    131            38102       1215
66        2      3     85 20959   45828     36            24231       1423
326       1      2  32717 16784   13626  60869             1272       5609
94        1      3  11314  3090    2062  35009               71       2698
197       1      1  30624  7209    4897  18711              763       2876
104       1      3  56082  3504    8906  18028             1480       2498
24        2      3  26373 36423   22019   5154             4337      16523
72        1      3  18291  1266   21042   5373             4173      14472
88        1      3  43265  5025    8117   6312             1579      14351
> data.rm.top <-data[-c(top.custs),] #Remove the Customers
> set.seed(76964057) #Set the seed for reproducibility
> k <-kmeans(data.rm.top[,-c(1,2)], centers=5) #Create 5 clusters, Remove columns 1 and 2
> k$centers #Display cluster centers
      Fresh      Milk   Grocery   Frozen Detergents_Paper Delicassen
1  4189.747  7645.639 11015.277 1335.145        4750.4819  1387.1205
2 16470.870  3026.491  4264.741 3217.306         996.5556  1319.7593
3 33120.163  4896.977  5579.860 3823.372         945.4651  1620.1860
4  5830.214 15295.048 23449.167 1936.452       10361.6429  1912.7381
5  5043.434  2329.683  2786.138 2689.814         652.8276   849.8414
> table(k$cluster) #Give a count of data points in each cluster

  1   2   3   4   5 
 83 108  43  42 145 
> rng<-2:20 #K from 2 to 20
> tries<-100 #Run the K Means algorithm 100 times
> avg.totw.ss<-integer(length(rng)) #Set up an empty vector to hold all of points
> for(v in rng){ # For each value of the range variable
+   v.totw.ss<-integer(tries) #Set up an empty vector to hold the 100 tries
+   for(i in 1:tries){
+     k.temp<-kmeans(data.rm.top,centers=v) #Run kmeans
+     v.totw.ss[i]<-k.temp$tot.withinss#Store the total withinss
+   }
+   avg.totw.ss[v-1]<-mean(v.totw.ss) #Average the 100 total withinss
+ }
> plot(rng,avg.totw.ss,type="b", main="Total Within SS by Various K",
+      ylab="Average Total Within Sum of Squares",
+      xlab="Value of K")
```



4. Spam classification using any Ensemble classifier? Find AUC, ROC, Confusion Matrix, and accuracy?
Data set link : https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data
The last attribute is the predictor

Soln:

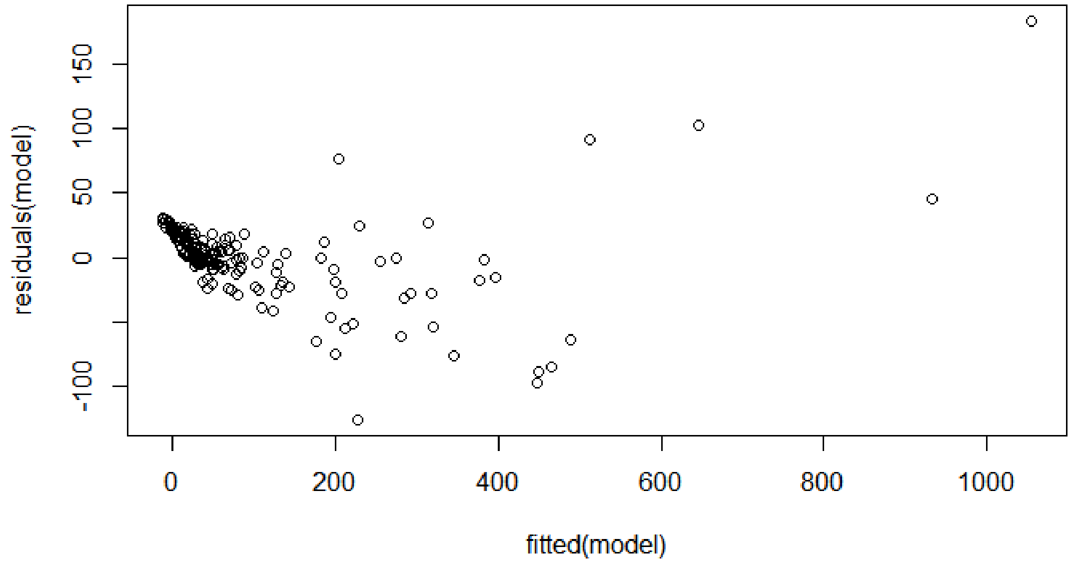
```
> names <- read.csv("C:/Users/chait/Desktop/VIT/Machine_Learning/Lab/Assesment -
5/names.csv",header=FALSE,sep=";")
> names(dataset) <- sapply((1:nrow(names)),function(i) toString(names[i,1]))
> dataset$y <- as.factor(dataset$y)
> sample <- dataset[sample(nrow(dataset), 300),]
> smp_size = floor(nrow(sample)*0.7)
> ind = sample(seq_len(nrow(sample)),size = smp_size)
> df_train = sample[ind,]
> df_test = sample[-ind,]
> resample.spam.train <- function()
+ {
+   indices <- sample(1:nrow(df_train),nrow(df_train),replace=TRUE)
+   df <- df_train[indices,]
+   return(df)
+ }
> bag.trees <- function(B) # B is the number of bootstrap samples
+ {
+   bootstrap.samples <- list()
+   pred.mat <- matrix(NA, nrow = nrow(df_test), ncol = B)
+
+   for(i in 1:B)
+   {
+     spam.sample <- resample.spam.train() # gets a bootstrap sample
+     tree <- rpart(y ~ .,
+                   data = spam.sample) # fits a tree
+
+     pruned.tree <- prune(tree,
+                          cp= tree$cptable[which.min(tree$cptable[,"xerror"]),
+                                           "CP"]) #prunes tree
```

```
+
+     # predictions
+     pred.mat[,i] <- predict(pruned.tree,
+                            newdata = df_test[,-ncol(df_test)],
+                            type = "class")
+
+     pred.mat[,i] <- pred.mat[,i] - 1 # convert to (0/1)
+   }
+   return(pred.mat)
+ }
> library(rpart)
> set.seed(11)
> bag <- bag.trees(50)
> # the 0/1 output is the one that the majority of the trees select
> bagged.preds <- (rowMeans(bag) >= 0.5)+0
> # final classification accuracy
> round(mean(bagged.preds == df_test$spam),2)
0.8
```

5. Implement polynomial regression and find all the necessary errors (Take any regression data from UCI machine learning repository) (if possible, in MS EXCEL; R2 and RMSE are expected to be calculated as I have demonstrated in both F1 and F2 slot classes)

Soln:

```
> indexes = sample(1:nrow(df), size=0.2*nrow(df))
> train = df[-indexes,]
> test = df[indexes,]
> model = lm(formula = ERP ~ MYCT+MMIN+MMAX+CACH+CHMIN+CHMAX+PRP,
data=train)
> confint(model)
                   2.5 %        97.5 %
(Intercept) -42.403438945 -21.223996296
MYCT          0.015685479   0.056745731
MMIN          0.003530103   0.008609015
MMAX          0.002112134   0.004028666
CACH          0.023046146   0.353958726
CHMIN        -0.705511161   1.310470281
CHMAX        -0.067176460   0.513565018
PRP           0.481381893   0.650260325
> plot(fitted(model),residuals(model))
```




```
> predicted.intervals = predict(model,df,interval='confidence',level=0.99)
> actual = df[,"ERP"]
> predicted = predicted.intervals[,"fit"]
> rss = sum((actual-predicted)^2)
> tss = sum((actual-mean(actual))^2)
> rsq = 1 - rss/tss
> rmse = (rss/nrow(df))^(.5)
> cat("RMSE value is",rmse)
RMSE value is 31.44807
> cat("R-squared value is",rsq)
R-squared value is 0.9585075
```

6. Implement PCA with high dimension data set.

7. Hierarchical clustering with any data set of your choice.

Soln of 6 & 7:

```
> library(dplyr)
> library(tibble)
> library(ggplot2)
> library(readr)
> proteoms <- read.csv("C:/Users/chait/Desktop/VIT/Machine_Learning/Lab/Assesment -
5/data.csv")
> colnames(proteoms)
 [1] "RefSeq_accession_number" "gene_symbol"             "gene_name"
"AO.A12D.01TCGA"          "C8.A131.01TCGA"
 [6] "AO.A12B.01TCGA"          "BH.A18Q.02TCGA"          "C8.A130.02TCGA"
"C8.A138.03TCGA"          "E2.A154.03TCGA"
[11] "C8.A12L.04TCGA"          "A2.A0EX.04TCGA"          "AO.A12D.05TCGA"
"AN.A04A.05TCGA"          "BH.A0AV.05TCGA"
[16] "C8.A12T.06TCGA"          "A8.A06Z.07TCGA"          "A2.A0CM.07TCGA"
"BH.A18U.08TCGA"          "A2.A0EQ.08TCGA"
```

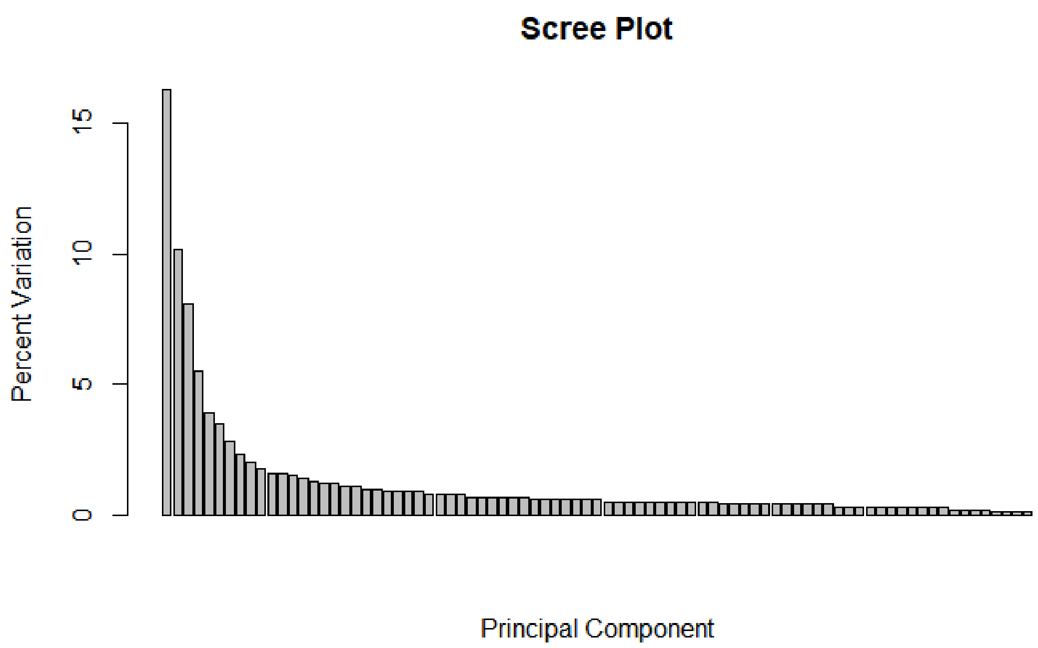
```
[21] "AR.A0U4.09TCGA"          "AO.A0J9.10TCGA"          "AR.A1AP.11TCGA"
"AN.A0FK.11TCGA"          "AO.A0J6.11TCGA"
[26] "A7.A13F.12TCGA"          "BH.A0E1.12TCGA"          "A7.A0CE.13TCGA"
"A2.A0YC.13TCGA"          "AO.A0JC.14TCGA"
[31] "A8.A08Z.14TCGA"          "AR.A0TX.14TCGA"          "A8.A076.15TCGA"
"AO.A126.15TCGA"          "BH.A0C1.16TCGA"
[36] "A2.A0EY.16TCGA"          "AR.A1AW.17TCGA"          "AR.A1AV.17TCGA"
"C8.A135.17TCGA"          "A2.A0EV.18TCGA"
[41] "AN.A0AM.18TCGA"          "D8.A142.18TCGA"          "AN.A0FL.19TCGA"
"BH.A0DG.19TCGA"          "AR.A0TV.20TCGA"
[46] "C8.A12Z.20TCGA"          "AO.A0JJ.20TCGA"          "AO.A0JE.21TCGA"
"AN.A0AJ.21TCGA"          "A7.A0CJ.22TCGA"
[51] "AO.A12F.22TCGA"          "A8.A079.23TCGA"          "A2.A0T3.24TCGA"
"A2.A0YD.24TCGA"          "AR.A0TR.25TCGA"
[56] "AO.A03O.25TCGA"          "AO.A12E.26TCGA"          "A8.A06N.26TCGA"
"A2.A0YG.27TCGA"          "BH.A18N.27TCGA"
[61] "AN.A0AL.28TCGA"          "A2.A0T6.29TCGA"          "E2.A158.29TCGA"
"E2.A15A.29TCGA"          "AO.A0JM.30TCGA"
[66] "C8.A12V.30TCGA"          "A2.A0D2.31TCGA"          "C8.A12U.31TCGA"
"AR.A1AS.31TCGA"          "A8.A09G.32TCGA"
[71] "C8.A131.32TCGA"          "C8.A134.32TCGA"          "A2.A0YF.33TCGA"
"BH.A0DD.33TCGA"          "BH.A0E9.33TCGA"
[76] "AR.A0TT.34TCGA"          "AO.A12B.34TCGA"          "A2.A0SW.35TCGA"
"AO.A0JL.35TCGA"          "BH.A0BV.35TCGA"
[81] "A2.A0YM.36TCGA"          "BH.A0C7.36TCGA"          "A2.A0SX.36TCGA"
"X263d3f.I.CPTAC"         "blcdb9.I.CPTAC"
[86] "c4155b.C.CPTAC"
> clean.proteoms <- na.omit(proteoms)
> head(as.matrix(clean.proteoms[,4:length(colnames(clean.proteoms))]))
   AO.A12D.01TCGA C8.A131.01TCGA AO.A12B.01TCGA BH.A18Q.02TCGA C8.A130.02TCGA C8.A138.03TCGA
E2.A154.03TCGA C8.A12L.04TCGA A2.A0EX.04TCGA
1       1.0961312     2.60994298     -0.6598280      0.1953407     -0.4940596      2.7650807
0.8626593    1.407570262       1.185108
3       1.1113704     2.65042179     -0.6542851      0.2154129     -0.5006193      2.7797092
0.8701860    1.410311827       1.188860
6       1.1075606     2.64637391     -0.6542851      0.2154129     -0.5038992      2.7797092
0.8701860    1.407570262       1.188860
7       1.1113704     2.65042179     -0.6487422      0.2154129     -0.5006193      2.7833664
0.8701860    1.410311827       1.188860
10      0.4827537    -1.04529350      1.2220027     -0.5172257     -0.4055031      0.7499970
2.3491966   -0.007077155       2.138081
11      0.2617854    -0.03737115      1.0196851     -0.7246394     -0.7039712     -0.1569735
1.5814659   -0.023526544       1.732880
   AO.A12D.05TCGA AN.A04A.05TCGA BH.A0AV.05TCGA C8.A12T.06TCGA A8.A06Z.07TCGA A2.A0CM.07TCGA
BH.A18U.08TCGA A2.A0EQ.08TCGA AR.A0U4.09TCGA
1       1.1006881     0.38458773     0.35053566     -0.2049179     -0.4964091      0.6834035
-0.2650304     -0.9126703    -0.03322133
3       1.1006881     0.37139283     0.36740533     -0.1666684     -0.4964091      0.6980976
-0.2516423     -0.9279787    -0.02721152
6       1.0970232     0.37799028     0.36740533     -0.1666684     -0.4964091      0.6980976
-0.2516423     -0.9279787    -0.03021642
7       1.0970232     0.37469156     0.36065746     -0.1666684     -0.4964091      0.6980976
-0.2516423     -0.9279787    -0.03021642
10      0.5436299     0.05141659     0.40114465      0.4708229      1.3493193     -0.9843733
-0.6465901     -2.2789429    -2.10059548
11      0.8404833    -1.62433531    -0.05433625      1.5163087     -1.9641727     -0.5472247
0.2336749     -2.2368449    -0.78745230
   AO.A0J9.10TCGA AR.A1AP.11TCGA AN.A0FK.11TCGA AO.A0J6.11TCGA A7.A13F.12TCGA BH.A0E1.12TCGA
A7.A0CE.13TCGA A2.A0YC.13TCGA AO.A0JC.14TCGA
1      0.02000705      0.4610875      0.9735642      0.8311317      1.2791847      0.7620444
-1.123173      0.8188241     -0.3072668
```

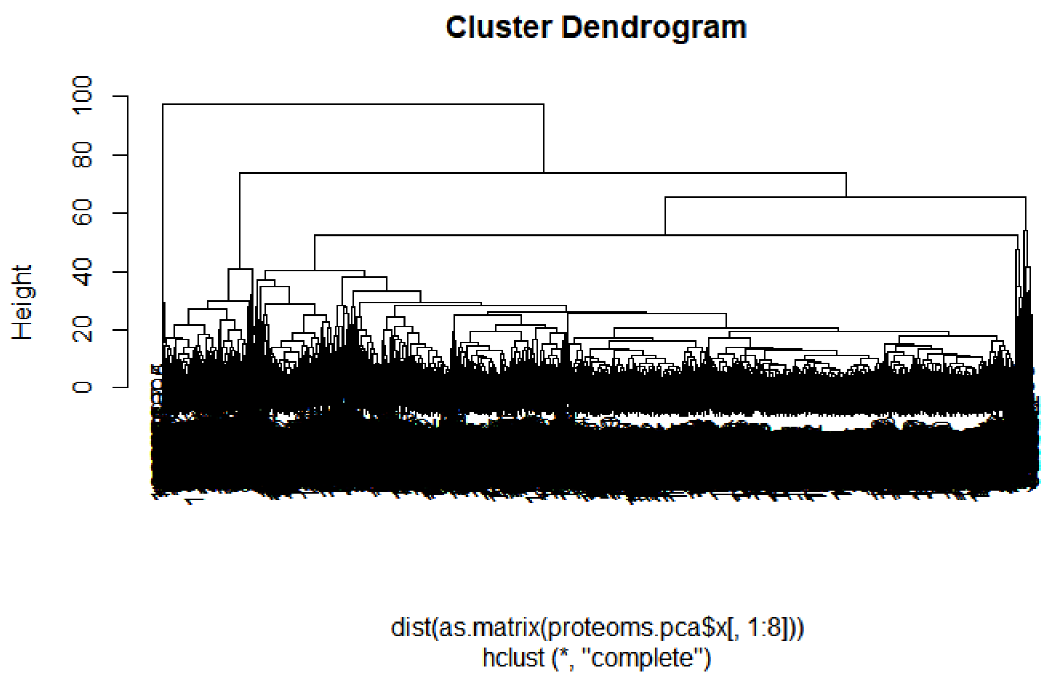
3 0.01195532 0.4610875 0.9774761 0.8565398 1.2751671 0.7663844
-1.116861 0.8148772 -0.3072668
6 0.01195532 0.4610875 0.9774761 0.8565398 1.2791847 0.7620444
-1.120017 0.8148772 -0.3072668
7 0.01195532 0.4610875 0.9774761 0.8508936 1.2791847 0.7620444
-1.123173 0.8148772 -0.3072668
10 0.32597284 -0.2836251 3.2620206 2.8383705 0.9617945 2.1855460
-2.575065 1.0516860 1.1702139
11 -0.53153654 -0.4656659 2.4248758 3.4058177 0.2145214 1.8079709
-2.742348 0.8661858 0.9551376
A8.A08Z.14TCGA AR.A0TX.14TCGA A8.A076.15TCGA AO.A126.15TCGA BH.A0C1.16TCGA A2.A0EY.16TCGA
AR.A1AW.17TCGA AR.A1AV.17TCGA C8.A135.17TCGA
1 0.5688946 -0.5834286 1.873982 0.1958767 -0.5183665 1.1748810
0.5783087 -0.7598231 1.120502
3 0.5688946 -0.5671090 1.870383 0.1958767 -0.5072138 1.1832088
0.5783087 -0.7491137 1.137618
6 0.5688946 -0.5779888 1.870383 0.1997197 -0.5072138 1.1832088
0.5783087 -0.7437590 1.127348
7 0.5688946 -0.5779888 1.870383 0.1997197 -0.5100020 1.1832088
0.5822129 -0.7544684 1.137618
10 0.8877477 1.6387646 1.377356 0.6531924 -2.1745373 0.8251135
-1.6978260 0.7261015 2.257001
11 1.2173487 1.7883614 0.981495 0.6378204 -2.5342107 0.3087899
-1.7602928 -1.4130944 2.154306
A2.A0EV.18TCGA AN.A0AM.18TCGA D8.A142.18TCGA AN.A0FL.19TCGA BH.A0DG.19TCGA AR.A0TV.20TCGA
C8.A12Z.20TCGA AO.A0JJ.20TCGA AO.A0JE.21TCGA
1 0.4529859 1.501967 0.5385958 2.455138 -0.2056375 -1.514278
-0.78719498 0.7571881 0.5597770
3 0.4725901 1.501967 0.5422105 2.480137 -0.2056375 -1.528285
-0.75594056 0.7741042 0.5597770
6 0.4725901 1.510348 0.5422105 2.471046 -0.2103218 -1.525484
-0.77156777 0.7774874 0.5597770
7 0.4725901 1.506158 0.5422105 2.480137 -0.2056375 -1.525484
-0.77156777 0.7774874 0.5597770
10 1.8112774 1.552255 0.2674902 -1.858279 0.2721667 -1.738392
0.02542003 1.9277820 1.7540156
11 1.9569086 1.149947 0.1590480 -1.117407 -1.0253996 -2.231442
0.96305274 1.1428751 0.7303825
AN.A0AJ.21TCGA A7.A0CJ.22TCGA AO.A12F.22TCGA A8.A079.23TCGA A2.A0T3.24TCGA A2.A0YD.24TCGA
AR.A0TR.25TCGA AO.A03O.25TCGA AO.A12E.26TCGA
1 -0.4281815 -1.0012398 -1.947792 1.048959 0.5837133 0.06377853
-1.1016752 1.053225 0.26485911
3 -0.4063780 -1.0046198 -1.955180 1.052257 0.5806231 0.08446902
-1.1087826 1.055948 0.27571131
6 -0.4063780 -1.0012398 -1.955180 1.052257 0.5868034 0.09333637
-1.1064135 1.055948 0.27299826
7 -0.4063780 -1.0012398 -1.955180 1.052257 0.5868034 0.08446902
-1.1087826 1.058671 0.27571131
10 0.9720842 0.5636905 -2.551133 3.436487 1.5014792 1.30520788
1.0755482 -1.250614 0.13191966
11 0.9018286 1.2227864 -1.915778 3.489250 1.0534116 1.38501405
-0.0000333 -1.375882 0.07765866
A8.A06N.26TCGA A2.A0YG.27TCGA BH.A18N.27TCGA AN.A0AL.28TCGA A2.A0T6.29TCGA E2.A158.29TCGA
E2.A15A.29TCGA AO.A0JM.30TCGA C8.A12V.30TCGA
1 0.2385471 -0.07820182 1.101261 0.3236627 0.7939756 -1.086529
2.1801233 1.395247 0.6739047
3 0.2441826 -0.07143937 1.097767 0.3269726 0.8147235 -1.095492
2.1801233 1.412341 0.6887176
6 0.2498182 -0.06805814 1.101261 0.3269726 0.8112655 -1.093252
2.1801233 1.412341 0.6887176
7 0.2441826 -0.07143937 1.101261 0.3269726 0.8112655 -1.093252
2.1801233 1.412341 0.6887176

```
10     -0.2292004    -0.28783799       2.103995     -1.9105423      1.5651080      -1.149272
0.8962164       1.802077      -2.1331321
11      0.5344115    -1.56594237       1.146681     -1.2551755      1.5097801      -1.270277
0.8112191       1.655071      -1.6628238
   A2.A0D2.31TCGA C8.A12U.31TCGA AR.A1AS.31TCGA A8.A09G.32TCGA C8.A131.32TCGA C8.A134.32TCGA
A2.A0YF.33TCGA BH.A0DD.33TCGA BH.A0E9.33TCGA
1       0.1074909      -0.4815502       1.222507     -1.5233435      2.7072502      0.1401818
0.3113192     -0.6923158       1.466665
3       0.1074909      -0.4815502       1.222507     -1.5099719      2.7376293      0.1331178
0.2961771     -0.6641611       1.474474
6       0.1041645      -0.4852105       1.218974     -1.5126462      2.7376293      0.1260538
0.2961771     -0.6618149       1.474474
7       0.1041645      -0.4815502       1.222507     -1.5153205      2.7376293      0.1154577
0.2961771     -0.6641611       1.474474
10     -1.5124729      -0.9976551       1.741961      0.1347327     -1.1205244     -2.3604810
2.3517137      0.9805402       2.813743
11     -1.4060276      -1.3527060       1.038755     -0.1139788     -0.6800265     -1.8589356
2.4955634      1.6914453       2.341290
   AR.A0TT.34TCGA AO.A12B.34TCGA A2.A0SW.35TCGA AO.A0JL.35TCGA BH.A0BV.35TCGA A2.A0YM.36TCGA
BH.A0C7.36TCGA A2.A0SX.36TCGA X263d3f.I.CPTAC
1      -0.51142119      -0.9639039     -0.4877725      -0.106680    -0.06583842      0.6558497
-0.5522120      -0.3985598       0.5985845
3      -0.52606668      -0.9439194     -0.4877725      -0.106680    -0.06583842      0.6558497
-0.5522120      -0.3926014       0.6039931
6      -0.52972805      -0.9382095     -0.4877725      -0.106680    -0.05589267      0.6581426
-0.5477494      -0.3926014       0.6066975
7      -0.52972805      -0.9439194     -0.4877725      -0.106680    -0.06252317      0.6558497
-0.5522120      -0.3926014       0.6039931
10     -0.52606668       1.3257521      0.7311482      -1.177327     0.70993057      1.3070365
0.4875738       0.6948104       2.7782634
11     -0.06107243       1.6455045      0.3580749      -3.466189    -0.50345170      1.3001577
0.7218603       0.5726628       3.8464683
   blcdb9.I.CPTAC c4155b.C.CPTAC
1       -0.1912845       0.5669753
3       -0.1860225       0.5767473
6       -0.1839177       0.5787017
7       -0.1860225       0.5767473
10       1.3673298       3.2151898
11       1.0884420       3.4809884
> proteoms.pca <- prcomp(clean.proteoms[,4:length(colnames(clean.proteoms))], center = T)
> names(proteoms.pca)
[1] "sdev"     "rotation" "center"   "scale"    "x"
> head(proteoms.pca$x)
         PC1       PC2        PC3        PC4      PC5        PC6       PC7        PC8
PC9        PC10        PC11        PC12        PC13
1  -3.599764 -1.566944  0.5272164 -3.2158079 3.177636 -0.7555303  1.236568 -1.2388591
-0.7678642 -1.2713068  0.1458712 -1.6907957  0.5198961
3  -3.663029 -1.561971  0.5348120 -3.1743285 3.207102 -0.7749749  1.242069 -1.2660165
-0.7820288 -1.2971958  0.1380706 -1.6795248  0.5232510
6  -3.659489 -1.562818  0.5234919 -3.1827055 3.196925 -0.7673091  1.240862 -1.2665938
-0.7776568 -1.2972613  0.1396268 -1.6770480  0.5201774
7  -3.653587 -1.567978  0.5246178 -3.1841037 3.201360 -0.7729774  1.241310 -1.2640772
-0.7821890 -1.2963795  0.1389090 -1.6797551  0.5243077
10 -6.987977  3.848287 -8.3213847 -1.8855901 3.140966 -0.9524179 -1.123623 -0.8112765
0.2357237  0.6794838  0.1724692  1.1846840 -0.7580815
11 -6.221785  5.179066 -6.4654576  0.2749185 3.310336 -0.7082599 -1.232022 -1.8427013
-0.2702671 -1.3154296 -1.2513509 -0.2886961 -1.3710253
         PC14        PC15         PC16       PC17        PC18        PC19        PC20      PC21
PC22        PC23         PC24         PC25
1   0.1278848 -0.3091430  0.05983299 0.2519446 -0.7335441 -0.6369246 -2.0793540 1.630920
0.3721621 -0.1727348 -1.17669988  1.1744078
```

```
3   0.1187915 -0.3098289  0.05973205 0.2378868 -0.7372402 -0.6419921 -2.0874183 1.649980
0.3701661 -0.1682483 -1.19661315  1.1759788
6   0.1144474 -0.3055530  0.05963252 0.2414510 -0.7345915 -0.6479520 -2.0839287 1.648393
0.3849554 -0.1689400 -1.18751383  1.1778023
7   0.1230106 -0.3094607  0.06432937 0.2444940 -0.7423673 -0.6436291 -2.0861637 1.653336
0.3773675 -0.1672175 -1.19194202  1.1769329
10 -0.7586473 -0.2943068  0.01816792 1.1820989 -0.1288812 -0.3481304  0.4277494 1.253868
1.4022907 -0.2483602 -0.04800123 -0.1380246
11  1.2258601 -1.0441680 -1.57195407 0.8718249 -1.4045816 -0.9149134  0.7500041 2.066840
0.4275349 -1.6789903  0.09785348 -0.4863625
         PC26       PC27       PC28       PC29      PC30       PC31      PC32      PC33
PC34        PC35        PC36      PC37
1   1.2730789  0.1341272 -0.7448445 -0.7943628 1.3235885 -0.4043017 -1.841656 0.9764388
-0.005675926 -0.31359739 0.011202981 -1.043821
3   1.2903815  0.1275161 -0.7459513 -0.8061584 1.3376292 -0.4174232 -1.854814 0.9651947
-0.011545822 -0.31674621 0.007301704 -1.055907
6   1.2946446  0.1220445 -0.7505712 -0.8057893 1.3398976 -0.4177089 -1.853340 0.9623030
-0.014780420 -0.32223840 0.004752804 -1.056947
7   1.2950546  0.1262595 -0.7518949 -0.8091630 1.3413802 -0.4184353 -1.855226 0.9640567
-0.012931139 -0.32380739 0.006513727 -1.061367
10 -0.2838253 -1.7303282 -2.2830296 -1.4509413 0.1545269  0.5648301 -2.599145 1.5567802
1.095657011 -0.03378417 1.110011541 -1.045809
11 -0.2806181 -1.1364446 -3.3429159 -2.0558134 0.1427084  0.9110828 -2.865932 1.7171899
1.734126207 -0.43004480 1.076600608 -1.313402
         PC38       PC39      PC40      PC41       PC42       PC43       PC44       PC45
PC46        PC47       PC48      PC49
1   0.4677184 -0.5213558 0.5530872 1.4508069  0.1519437 -0.5177118  0.1571119  0.3512621
-0.6542297 -0.21949651  0.8885389 0.6484622
3   0.4707762 -0.5327662 0.5713006 1.4725120  0.1616131 -0.5057932  0.1645801  0.3607477
-0.6630465 -0.21168309  0.8934769 0.6415484
6   0.4661709 -0.5350788 0.5665955 1.4757970  0.1662754 -0.5058287  0.1650478  0.3631333
-0.6674184 -0.21289532  0.8896392 0.6441700
7   0.4677679 -0.5323697 0.5670929 1.4795896  0.1629602 -0.5067911  0.1672954  0.3605918
-0.6706043 -0.21082389  0.8894013 0.6380101
10 -0.3650018  0.3574800 0.6458406 0.1879803  0.8767433 -1.7227723 -0.8167122 -0.7023169
0.6607983 -0.90787179 -0.5494380 0.6938012
11  0.5048332  0.3143009 0.5880441 0.2633403 -0.1787429 -1.1603788 -0.9667014 -0.8833431
0.5741339  0.07906612 -1.2556128 1.5393121
        PC50      PC51       PC52       PC53          PC54        PC55       PC56        PC57
PC58        PC59      PC60       PC61
1  0.7389762 0.5054221  1.3039426 -0.5745544 -0.4280911943  1.25497826  0.5175599 -0.95682545
0.08759537  0.5995759 1.1135901 -0.1424623
3  0.7420167 0.4995166  1.2962319 -0.5827225 -0.4308485220  1.25084658  0.5160885 -0.96543051
0.08165372  0.5916432 1.1302282 -0.1359832
6  0.7398428 0.4983926  1.2970568 -0.5863308 -0.4288411489  1.24522666  0.5174385 -0.96601479
0.08335439  0.5927713 1.1296760 -0.1349522
7  0.7430489 0.5042278  1.3003611 -0.5866777 -0.4302598463  1.24729669  0.5138128 -0.97060698
0.08265221  0.5936489 1.1300810 -0.1339361
10 1.1010316 0.4874672 -0.2434193 -0.8707798 -0.0006285706 -0.04885111 -0.9857605 -0.19902650
0.21558285 -0.3040391 0.3998504 -0.6369640
11 1.5013689 0.1482857 -0.4205078 -0.7720899  0.3185529833  0.23411109 -0.2767125  0.03969188
0.01679283  0.7545664 0.4022292 -0.2994323
         PC62       PC63      PC64       PC65        PC66       PC67       PC68       PC69
PC70          PC71        PC72       PC73
1  -0.4396710 -0.8274045 0.6913277 -0.2008835 -0.58697057 -0.1704272 -0.5212419 -0.1104781
-0.61561488  0.013746580  0.07298438 -0.8132648
3  -0.4306439 -0.8304625 0.6993866 -0.2046255 -0.59201806 -0.1717550 -0.5219260 -0.1103553
-0.61985783  0.010947670  0.06679688 -0.8287507
6  -0.4337972 -0.8346528 0.7011320 -0.2076947 -0.58788279 -0.1741851 -0.5224849 -0.1067539
-0.61508595  0.008312970  0.06248424 -0.8347449
7  -0.4299272 -0.8311223 0.6978375 -0.2044995 -0.59147944 -0.1698670 -0.5273007 -0.1072980
-0.61530007  0.007864979  0.06575833 -0.8336471
```

```
10  0.1201587 -0.4771048 0.6366893  1.6612986 -0.19757528 -0.7090803 -0.6617768 -0.5074338
0.09124969  0.500687867 -0.11114594  0.3997009
11  0.1137704  0.6415849 0.3727557  1.3547096  0.03217751 -0.7852475 -0.6970829 -0.6019859
0.47869164 -0.263715277 -0.38793632  0.1247979
          PC74        PC75        PC76       PC77       PC78       PC79       PC80
PC81       PC82         PC83
1  -0.22110227 -0.07298477 -0.09755746  0.5551588  0.1641169 -0.1179238  0.1555411
0.03029241 -0.1525013 -0.017847283
3  -0.22229465 -0.07052686 -0.10640685  0.5404295  0.1648348 -0.1177008  0.1487354
0.02516098 -0.1655298 -0.011432206
6  -0.22163271 -0.07040523 -0.10422538  0.5396141  0.1684001 -0.1177256  0.1535451
0.02395941 -0.1671409 -0.008769818
7  -0.22310874 -0.07021702 -0.10415823  0.5431360  0.1649801 -0.1163722  0.1468095
0.02538999 -0.1617406 -0.009363245
10  1.02721192  0.04138428  0.07813679  0.6814881 -0.4870878  0.4792118 -0.0723777
-0.64051379  0.1210410  0.156150165
11  0.09586813  0.23616488 -0.28745803 -0.5772096  0.3530266 -0.6093605  0.0755825
-0.94027481 -0.2943216 -0.267363756
> pca.var <- proteoms.pca$sdev^2
> pca.var.per <- round(pca.var/sum(pca.var)*100,1)
> barplot(pca.var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent
Variation")
> (x <- cbind(1:length(pca.var.per),cumsum(pca.var.per)))[x[,1] == 8,]
[1]  8.0 52.6
> hclust.out <- hclust(dist(as.matrix(proteoms.pca$x[,1:8])),  method = "complete")
> plot(hclust.out)
```





8. Mention one catchy and contemporary problem statement that can be solved by machine learning with 10 sentences (No coding/ program is required only problem statement and technical requirements are to be written)

Soln:

Machine Learning can be used to detect objects, persons from blurred out images or videos captured from phones, webcam, or cctv cameras to correctly identify a person. The model can be trained according to the given input dataset and according to which the person can be identified even if the images or videos are blurry.

Also, machine learning can be used in robotics, to allow them to learn actions, by action recognition and training according to it. It can be trained on a variety of actions to identify and perform.