Machine Learning: Assignment 1

Group members:

1. Akshat lal (2018B4A70051P)
2. Kalash Shah (2018A7PS0213P)
3. Aakash (2018B4A70887P)

Aim : To implement supervised and unsupervised machine learning algorithms on a dataset and compare and document the results.

Data Set Used: Letter Recognition Data Set

Algorithms Used:

1. Fisher's Linear Discriminant (Supervised)
2. Principal Component Analysis (Unsupervised)
3. Random Forests (Supervised)

For the purpose of comparison, SVM has been implemented on RAW data as a contrast to FLD and PCA

Libraries used :

```
library(e1071)
library(caret)
library(MASS)
library(ggord)
library(corrplot)
require(foreign)
require(nnet)
require(ggplot2)
library(useful)
require(reshape2)
library(randomForest)
```

The seed is set to generate fixed random numbers. Data is read from the datasheet. Summary is generated.

```
origdata <- read.table("letter-recognition.data", header=FALSE,sep=",")
summary(origdata)
```

```
##        V1              V2              V3              V4
##  U      :  813   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  D      :  805   1st Qu.: 3.000   1st Qu.: 5.000   1st Qu.: 4.000
##  P      :  803   Median : 4.000   Median : 7.000   Median : 5.000
##  T      :  796   Mean   : 4.024   Mean   : 7.035   Mean   : 5.122
```

```
##  M      :  792   3rd Qu.: 5.000   3rd Qu.: 9.000   3rd Qu.: 6.000
##  A      :  789   Max.   :15.000   Max.   :15.000   Max.   :15.000
##  (Other):15202
##        V5               V6               V7               V8
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.0
##  1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.: 6.000   1st Qu.: 6.0
##  Median : 6.000   Median : 3.000   Median : 7.000   Median : 7.0
##  Mean   : 5.372   Mean   : 3.506   Mean   : 6.898   Mean   : 7.5
##  3rd Qu.: 7.000   3rd Qu.: 5.000   3rd Qu.: 8.000   3rd Qu.: 9.0
##  Max.   :15.000   Max.   :15.000   Max.   :15.000   Max.   :15.0
##
##        V9              V10              V11              V12
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 3.000   1st Qu.: 4.000   1st Qu.: 7.000   1st Qu.: 5.000
##  Median : 4.000   Median : 5.000   Median : 8.000   Median : 6.000
##  Mean   : 4.629   Mean   : 5.179   Mean   : 8.282   Mean   : 6.454
##  3rd Qu.: 6.000   3rd Qu.: 7.000   3rd Qu.:10.000   3rd Qu.: 8.000
##  Max.   :15.000   Max.   :15.000   Max.   :15.000   Max.   :15.000
##
##       V13              V14              V15              V16
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 7.000   1st Qu.: 1.000   1st Qu.: 8.000   1st Qu.: 2.000
##  Median : 8.000   Median : 3.000   Median : 8.000   Median : 3.000
##  Mean   : 7.929   Mean   : 3.046   Mean   : 8.339   Mean   : 3.692
##  3rd Qu.: 9.000   3rd Qu.: 4.000   3rd Qu.: 9.000   3rd Qu.: 5.000
##  Max.   :15.000   Max.   :15.000   Max.   :15.000   Max.   :15.000
##
##       V17
##  Min.   : 0.000
##  1st Qu.: 7.000
##  Median : 8.000
##  Mean   : 7.801
##  3rd Qu.: 9.000
##  Max.   :15.000
##
```

No missing values are there, so no preprocessing is required.

Breaking Data Into Train and Test

```r
set.seed(7)
dt = sort(sample(nrow(origdata), nrow(origdata)*.7))
mydata.train <- origdata[dt,]
mydata.test <- origdata[-dt,]
```

Implementing SVM on Raw Data :

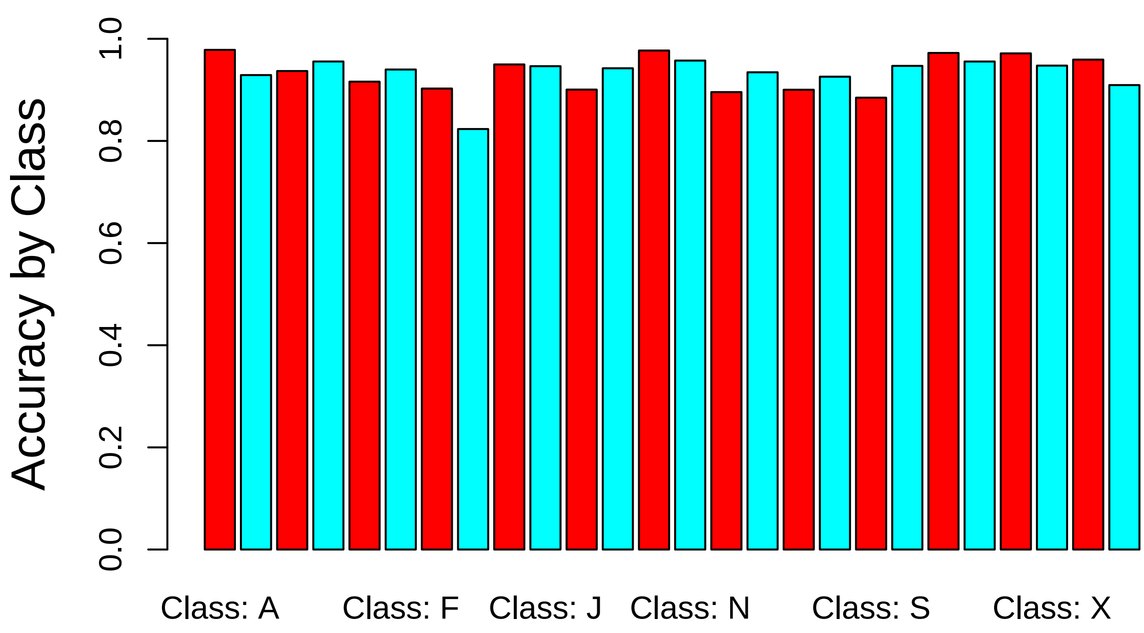
1) Training Data

```
raw.svm <- svm(V1~.,data = mydata.train,kernel ="linear", cost=1, scale=FALSE)
rawSVMTrainPrediction <- predict(raw.svm,mydata.train[2:17])
raw.tab1 <- table(Predicted = rawSVMTrainPrediction, Actual = mydata.train$V1)
raw.TrainAccuracy <- sum(diag(raw.tab1))/sum(raw.tab1)
raw.TrainAccuracy
```

```
## [1] 0.8722857
```

Confusion Matrix:

```
cmat <- confusionMatrix( raw.tab1 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



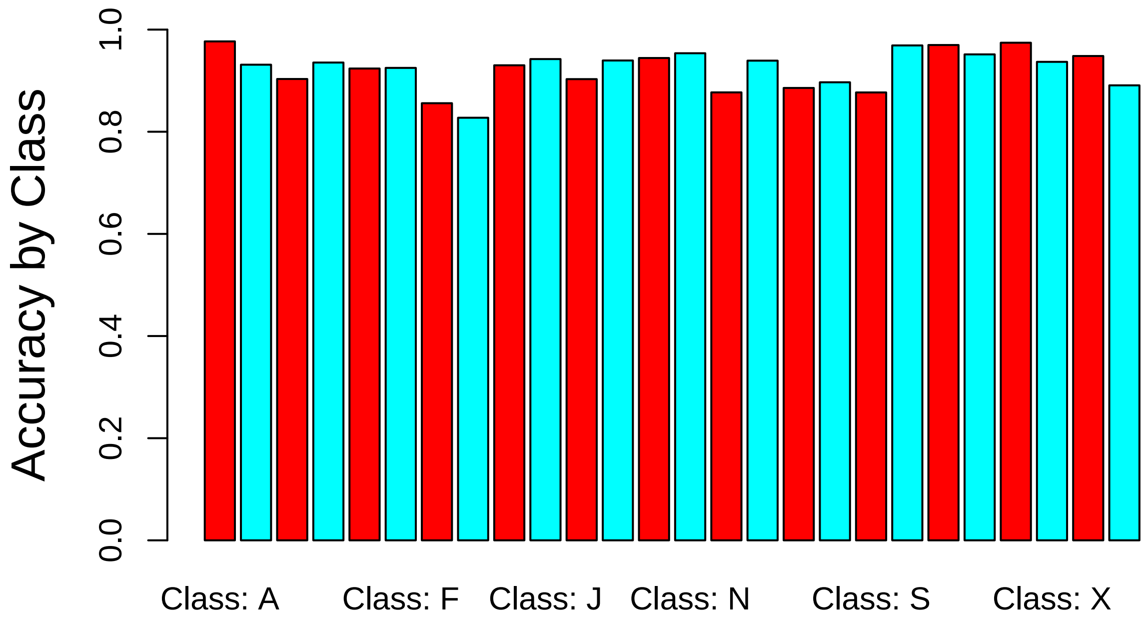
2) Test Data

```
rawSVMTestPrediction <- predict(raw.svm,mydata.test[2:17])
raw.tab2 <- table(Predicted = rawSVMTestPrediction, Actual = mydata.test$V1)
raw.TestAccuracy <- sum(diag(raw.tab2))/sum(raw.tab2)
raw.TestAccuracy
```

```
## [1] 0.8531667
```

Confusion Matrix:

```r
cmat <- confusionMatrix( raw.tab2 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



Linear Discriminant Analysis

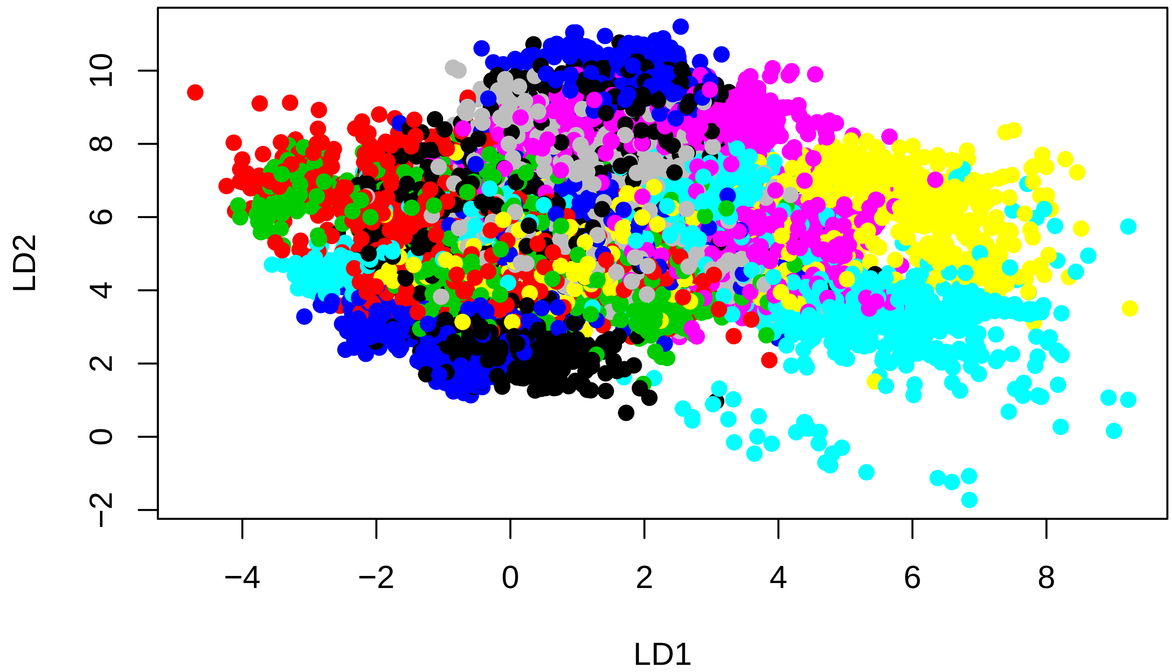
Training LDA and summary:

```
linear <- lda(V1~., mydata.train)
linear$counts
```

```
##   A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T
## 550 529 518 561 537 526 514 519 527 518 515 542 563 550 510 568 546 528 548 567
##   U   V   W   X   Y   Z
## 564 540 541 543 557 519
```

Projecting and Plotting the data along the 2 best projection vectors (LD1 & LD2):

```
projected_dataTrain = as.matrix( mydata.train[, 2:17] ) %*% linear$scaling
plot( projected_dataTrain, col = mydata.train[,1], pch = 19 )
```



Accuray using the LDA classifier :

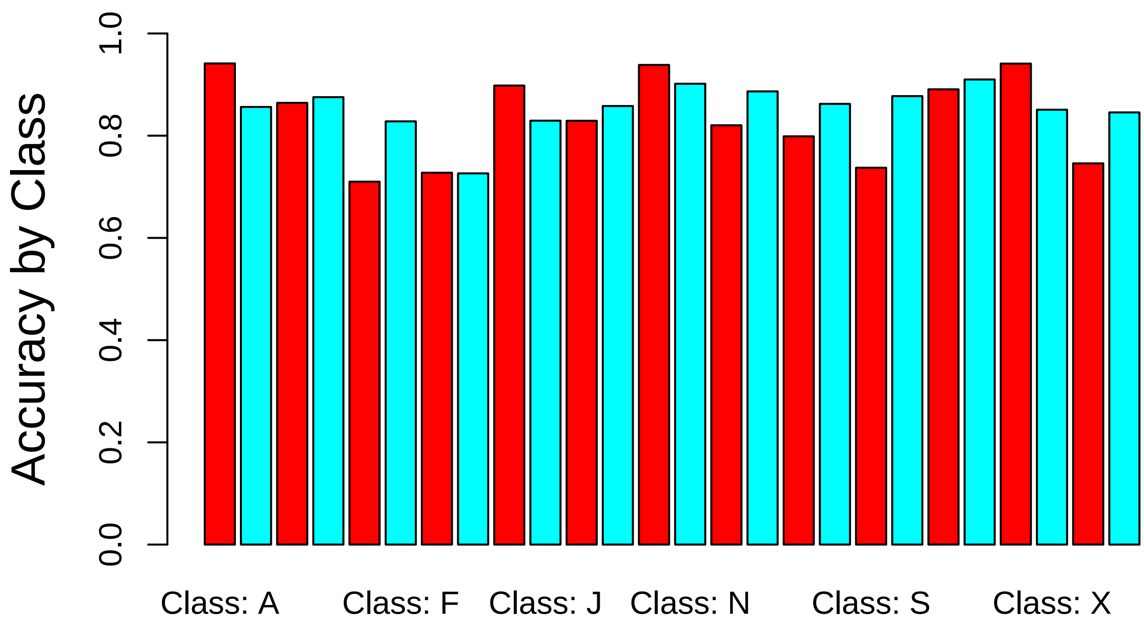
```
X_test = mydata.test[,2:17]
linear.results = predict( linear, X_test )

t = table(linear.results$class, mydata.test[,1] )
sum(diag(t))/sum(t)
```

```
## [1] 0.7006667
```

Confusion Matrix:

```
cmat <- confusionMatrix( t )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



New Dataset obtained upon projecting data along the projection vectors:

```
projected_dataTrain = data.frame(Predictions = mydata.train$V1, projected_dataTrain)
svm.lda <- svm(Predictions~., data=projected_dataTrain , kernel ="linear", cost=100, scale=FALSE)
LDATrainSVMprediction <- predict(svm.lda,projected_dataTrain[,2:17])
tab1 <- table(Predicted = LDATrainSVMprediction, Actual = mydata.train$V1)
```

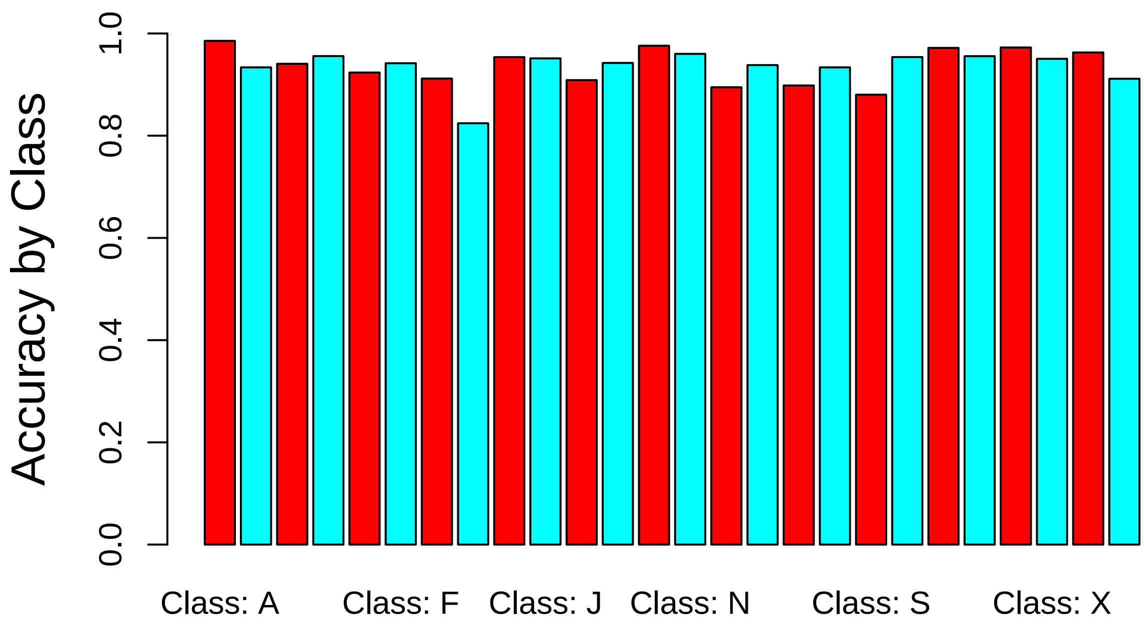
Training accuracy:

```
TrainAccuracy <- sum(diag(tab1))/sum(tab1)
TrainAccuracy
```

```
## [1] 0.8777143
```

Confusion Matrix:

```
cmat <- confusionMatrix( tab1 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



Projecting Test Data:

```
projected_dataTest = as.matrix( mydata.test[, 2:17] ) %*% linear$scaling
LDATestSVMprediction <- predict(svm.lda,projected_dataTest)
```

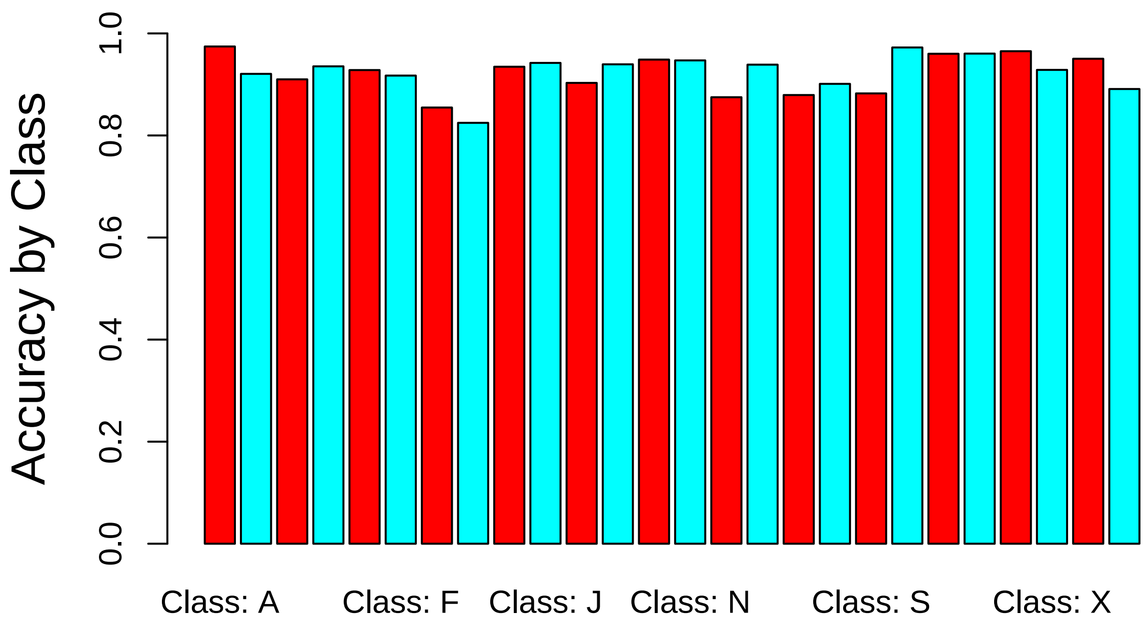
Test accuracy:

```
tab2 <- table(Predicted = LDATestSVMprediction, Actual = mydata.test$V1)
TestAccuracy <- sum(diag(tab2))/sum(tab2)
TestAccuracy
```

```
## [1] 0.8511667
```

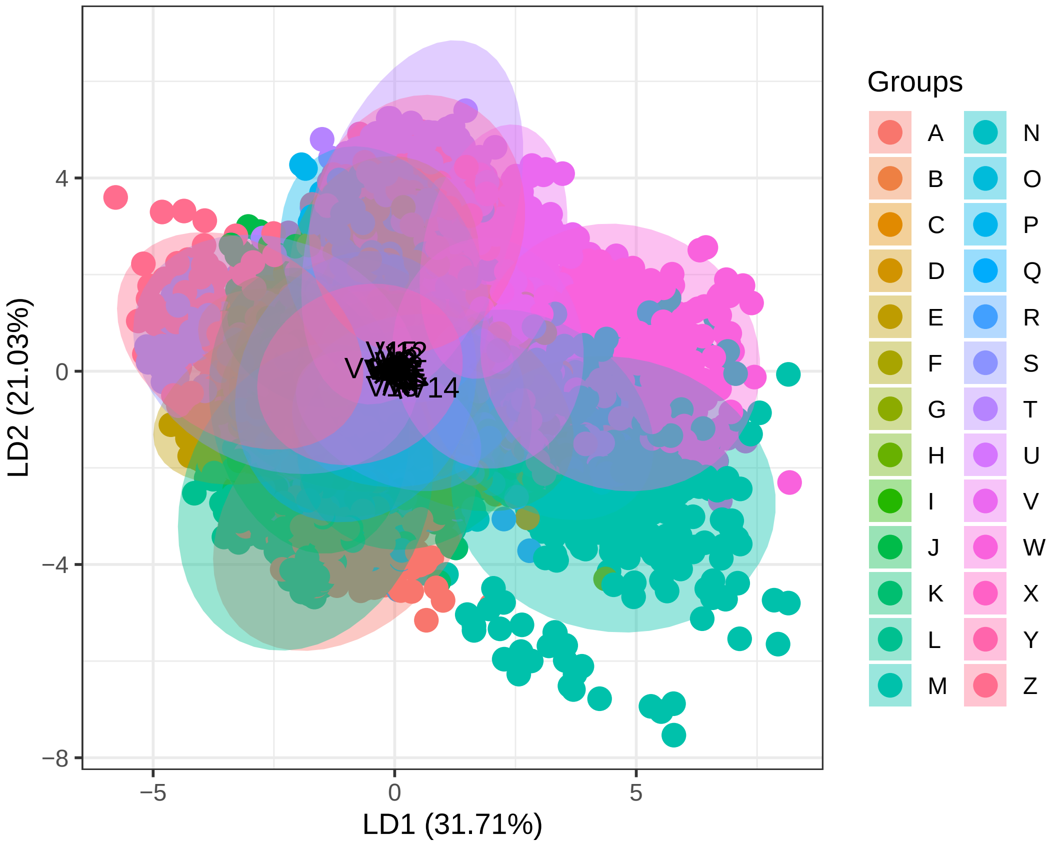
Confusion Matrix:

```
cmat <- confusionMatrix( tab2 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



Visualising separation for training data

```
ggord(linear, mydata.train$V1)
```



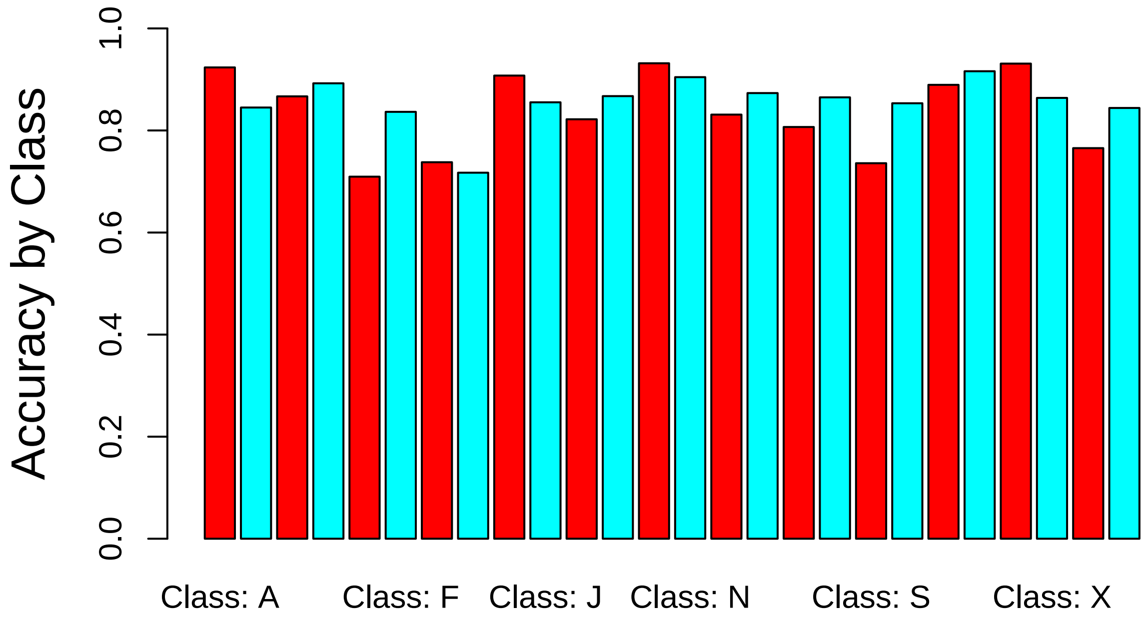
Using LDA as a classifier: Accuracy on Training data:

```
p1 <- predict(linear, mydata.train)$class
tab1 <- table(Predicted = p1, Actual = mydata.train$V1)
sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.7047857
```

Confusion Matrix:

```
cmat <- confusionMatrix( tab1 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, be
```



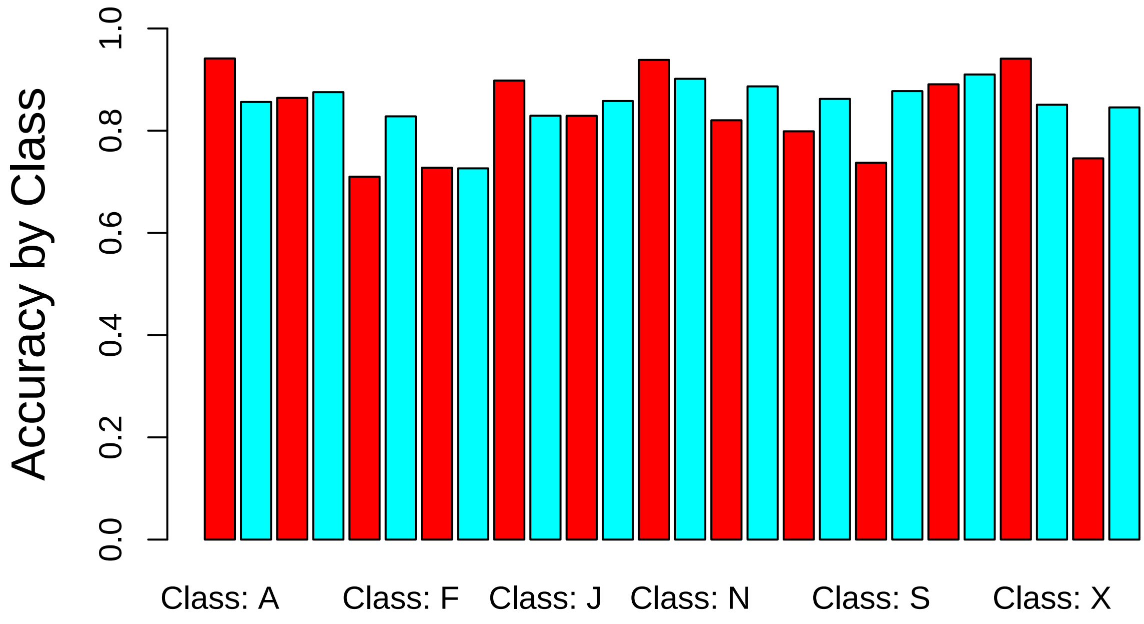
Accuracy on test data

```
p2 <- predict(linear, mydata.test)$class
tab2 <- table(Predicted = p2, Actual = mydata.test$V1)
sum(diag(tab2))/sum(tab2)
```

```
## [1] 0.7006667
```

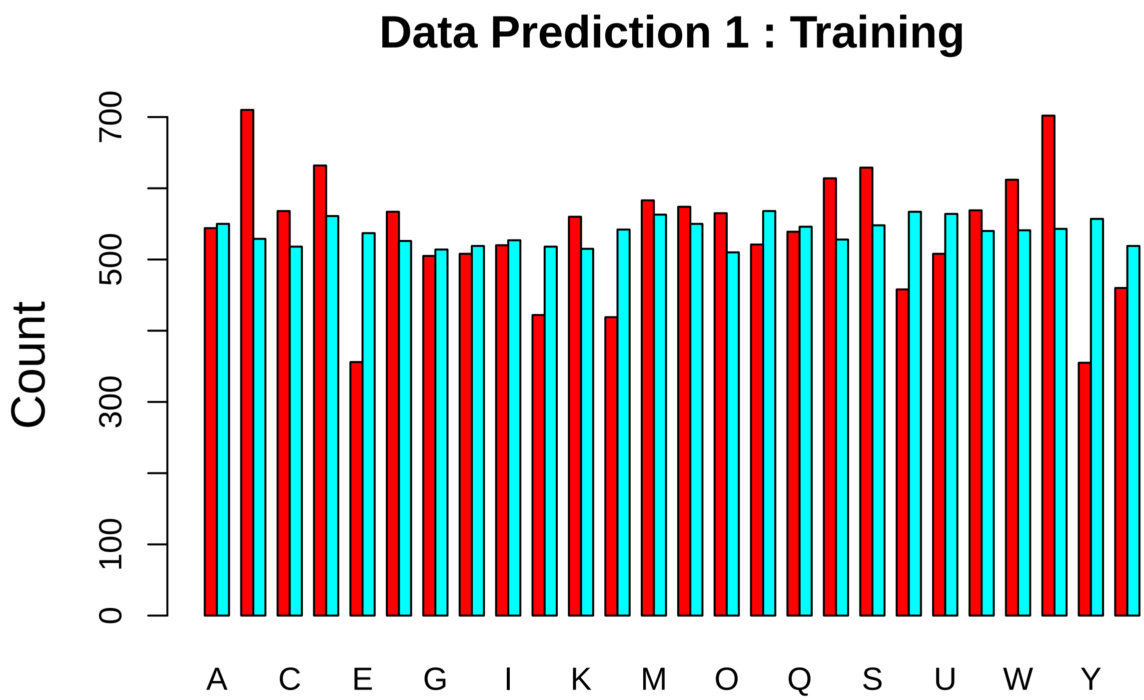
Confusion Matrix:

```
cmat <- confusionMatrix( tab2 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



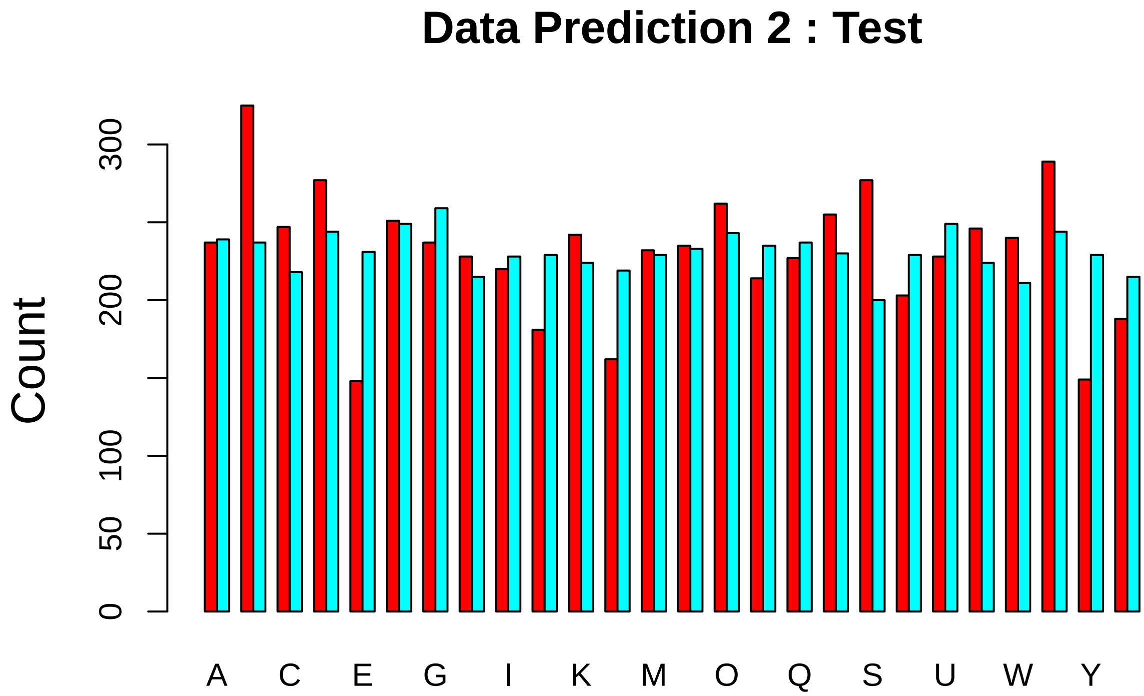
Bar Plot for Training Data:

```
report1<-rbind(summary(p1),summary(mydata.train$V1))
barplot(as.matrix(report1), main="Data Prediction 1 : Training", ylab = "Count", cex.lab = 1.5, cex.mai
```



BarPlot for test data

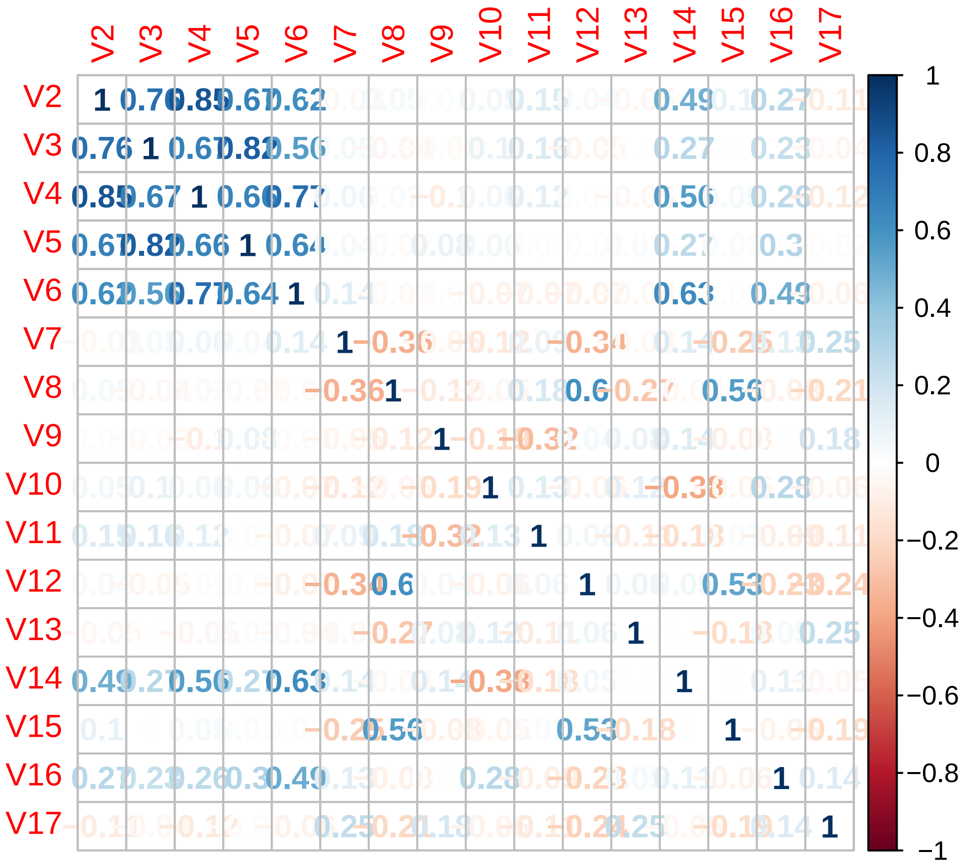
```
report2<-rbind(summary(p2),summary(mydata.test$V1))
barplot(as.matrix(report2), main="Data Prediction 2 : Test", ylab = "Count", cex.lab = 1.5, cex.main = 
```


Data Prediction 2 : Test
Count
0
50
100
200
300
A
C
E
G
I
K
M
O
Q
S
U
W
Y

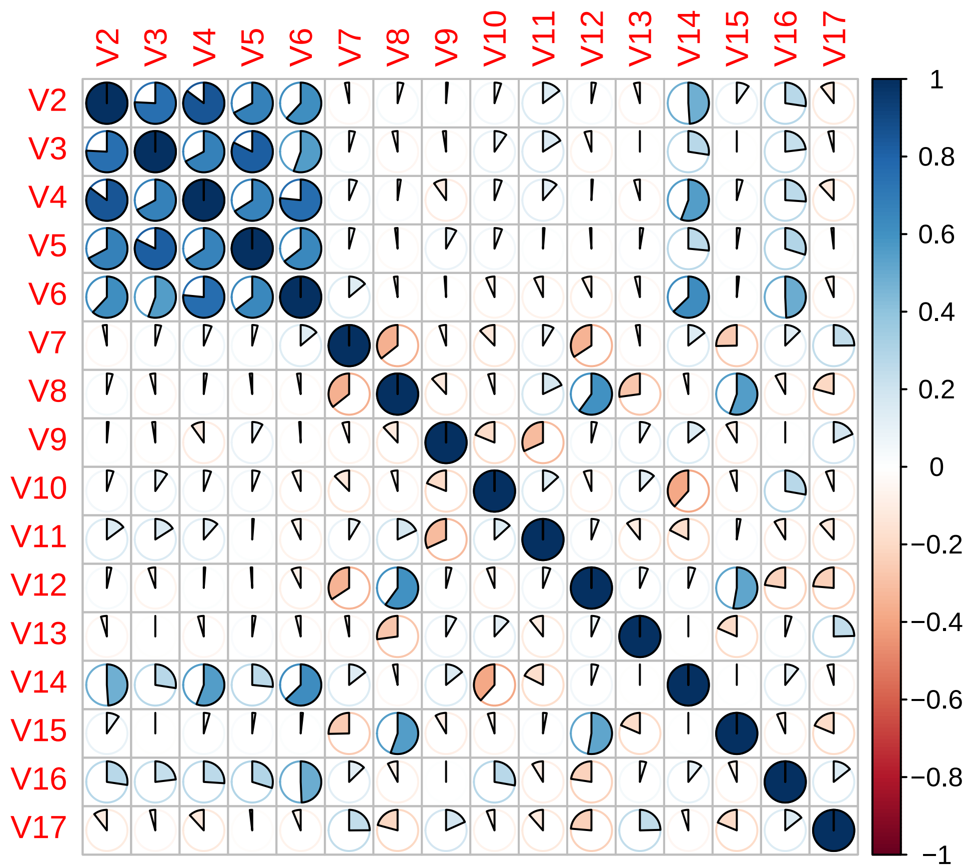
Principal Component Analysis

Visualizing dependency of variables with each other

```
crp1 = cor(origdata[sapply(origdata,is.numeric)],method="pearson")
corrplot(crp1,method="number")
```




```
corrplot(crp1,method="pie")
```

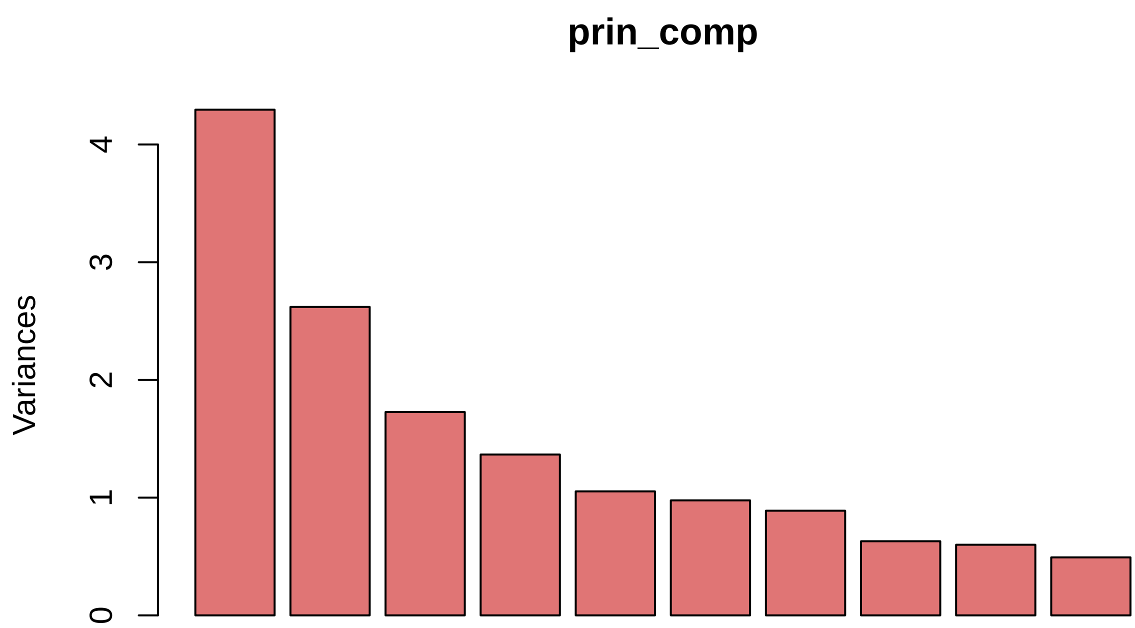


Generating Vectors of PCA and plotting the variances

```
prin_comp <- prcomp(mydata.train[,2:ncol(mydata.train)], scale. = T)
summary(prin_comp)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.0725 1.6187 1.3142 1.16879 1.02633 0.98845 0.94282
## Proportion of Variance 0.2685 0.1638 0.1080 0.08538 0.06583 0.06106 0.05556
## Cumulative Proportion  0.2685 0.4322 0.5402 0.62555 0.69138 0.75245 0.80800
##                            PC8    PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     0.79352 0.7746 0.70186 0.65146 0.51437 0.5044 0.46167
## Proportion of Variance 0.03936 0.0375 0.03079 0.02653 0.01654 0.0159 0.01332
## Cumulative Proportion  0.84736 0.8849 0.91564 0.94217 0.95870 0.9746 0.98792
##                          PC15    PC16
## Standard deviation     0.3442 0.27343
## Proportion of Variance 0.0074 0.00467
## Cumulative Proportion  0.9953 1.00000
```

```
plot(prin_comp, col=rgb(0.8,0.1,0.1,0.6))
```

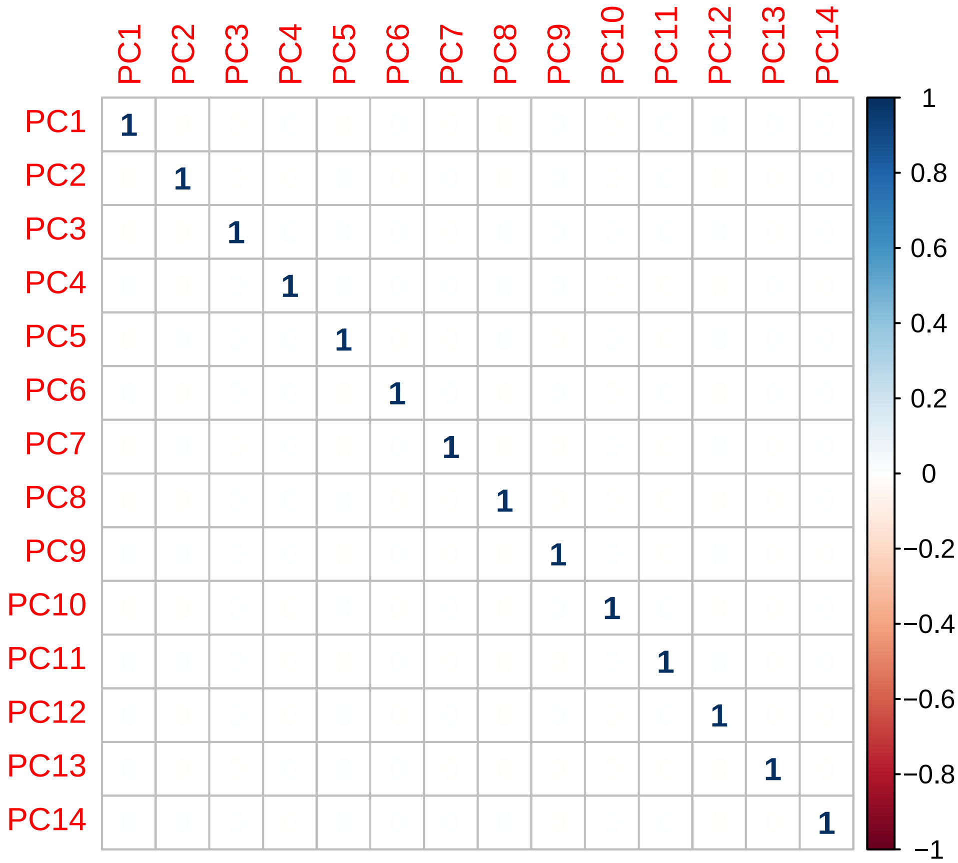


Taking Training Data and Joining with Labelled Letters

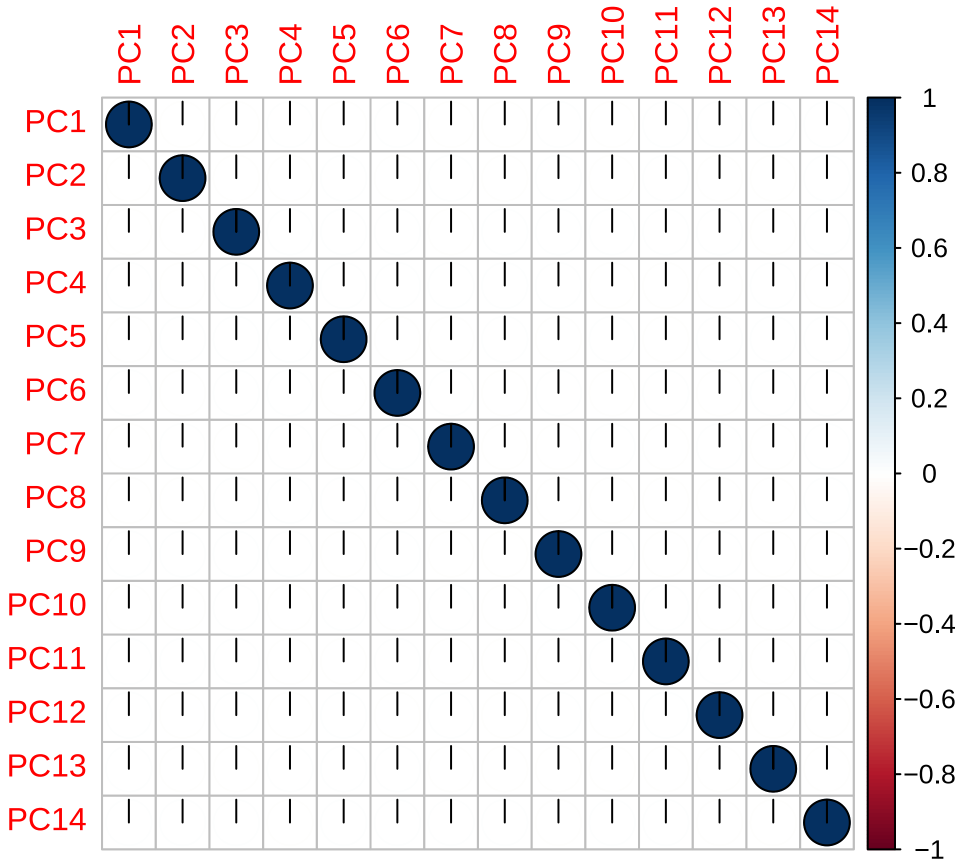
```
train.data <- data.frame(Predictions = mydata.train$V1, prin_comp$x)
train.data <- train.data[,1:15]
modelLF <- svm(Predictions~., data=train.data , kernel ="linear", cost=100, scale=FALSE)
```

Correlation between the principal components

```
crp2 = cor(train.data[sapply(train.data,is.numeric)],method="pearson")
corrplot(crp2,method="number")
```




```
corrplot(crp2,method="pie")
```



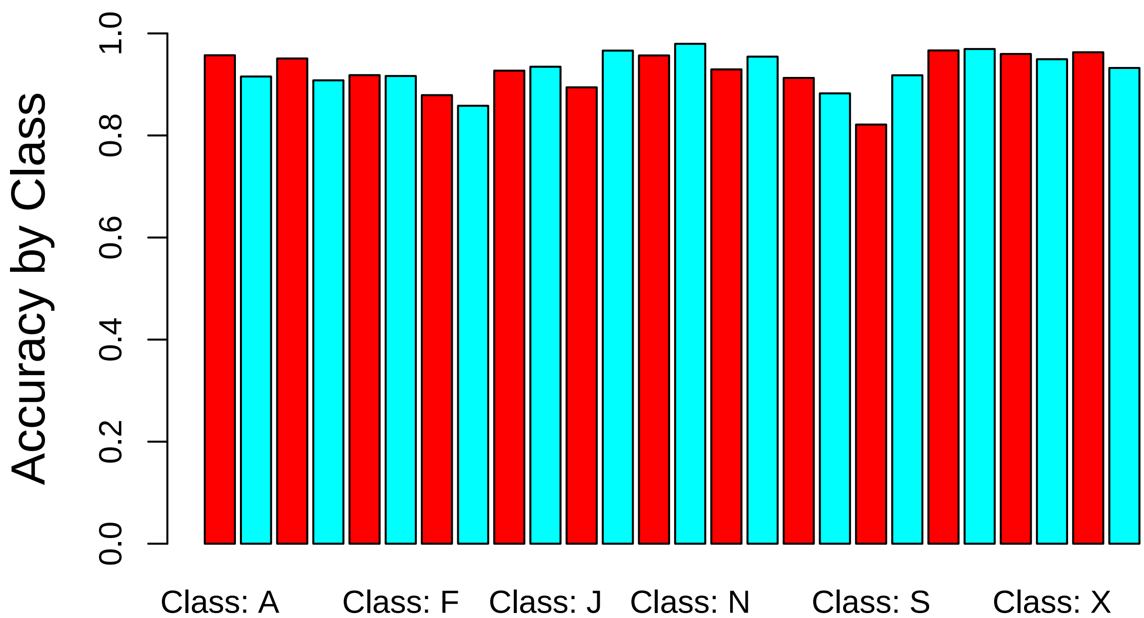
Testing Model for Training Data

```
trainDataPredictionLF = predict(modelLF,prin_comp$x[,1:14])
xtabLF <- table(mydata.train$V1, trainDataPredictionLF)
accuracy1LF <- sum(diag(xtabLF)) / sum(xtabLF)
accuracy1LF
```

```
## [1] 0.8598571
```

Confusion Matrix:

```
cmat <- confusionMatrix( xtabLF )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```

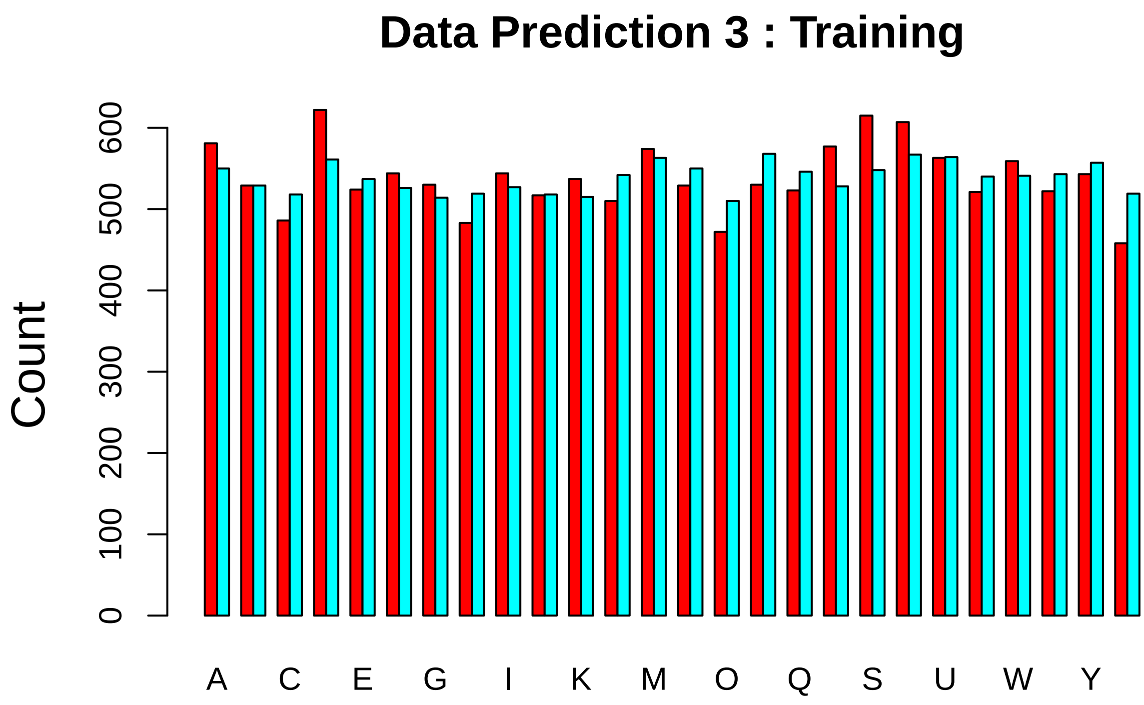


Summarising and Bar plot for Test Data

```
xLF <- compare.list(trainDataPredictionLF,mydata.train$V1)
summary(xLF)
```

```
##    Mode   FALSE    TRUE
## logical    1962   12038
```

```
report<-rbind(summary(trainDataPredictionLF),summary(mydata.train$V1))
barplot(as.matrix(report), main="Data Prediction 3 : Training", ylab = "Count", cex.lab = 1.5, cex.main
```



Changing Test Data using PCA vectors of Training Data

```
test.data <- predict(prin_comp, newdata = mydata.test[,2:ncol(mydata.test)])
test.data <- as.data.frame(test.data)
test.data <- test.data[,1:14]
```

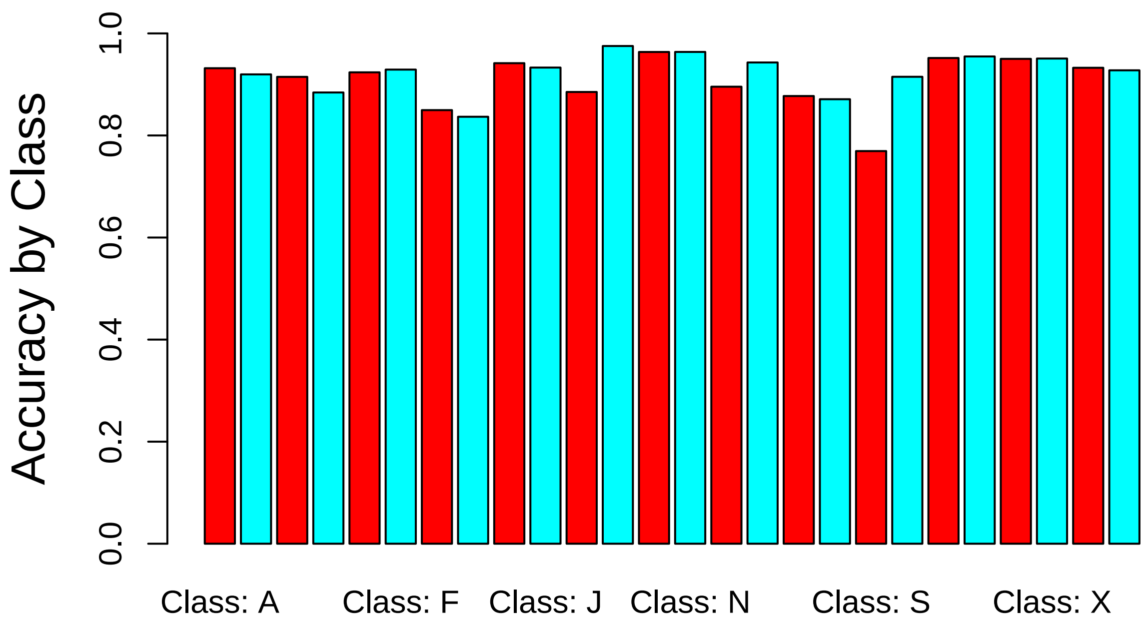
Test Data accuracy

```
testDataPredictionLF = predict(modelLF,test.data)
ytabLF <- table(mydata.test$V1, testDataPredictionLF)
accuracy2LF <- sum(diag(ytabLF)) / sum(ytabLF)
accuracy2LF
```

```
## [1] 0.8336667
```

Confusion Matrix:

```
cmat <- confusionMatrix( ytabLF )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, be
```

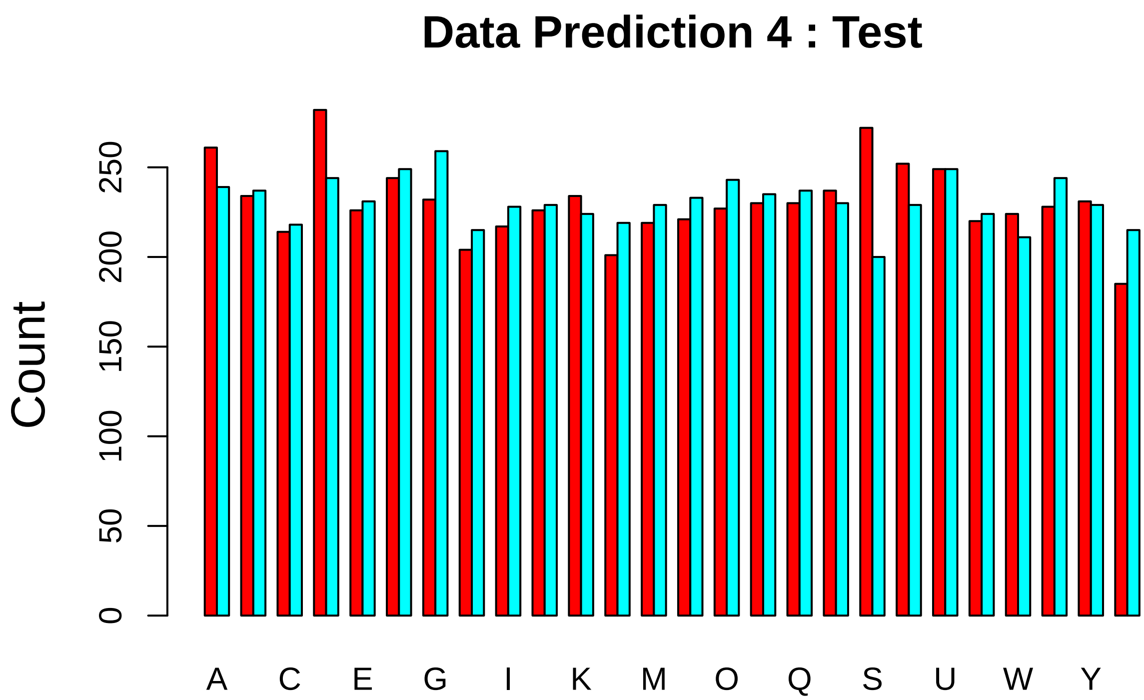


Summarising and Bar plot for Test Data

```
yLF <- compare.list(testDataPredictionLF,mydata.test$V1)
summary(yLF)
```

```
##    Mode   FALSE    TRUE
## logical     998    5002
```

```
report<-rbind(summary(testDataPredictionLF),summary(mydata.test$V1))
barplot(as.matrix(report), main="Data Prediction 4 : Test", ylab = "Count", cex.lab = 1.5, cex.main = 1
```


Data Prediction 4 : Test
Count
0
50
100
150
200
250
A
C
E
G
I
K
M
O
Q
S
U
W
Y

Random Forests

Variable selection from random forests uses both backwards variable elimination (for the selection of small sets of non-redundant variables) and selection based on the importance spectrum. It is an ensemble learning method for classification that takes only a section of all the variables, by the aforementioned ways, which makes it a supervised dimensionality reduction model.

Training the model for Random Forests:

```
t1 <- mydata.train
t2 <- mydata.test
model <- randomForest(V1 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14+V15+V16+V17,data=t1,ntree=10,impo
```

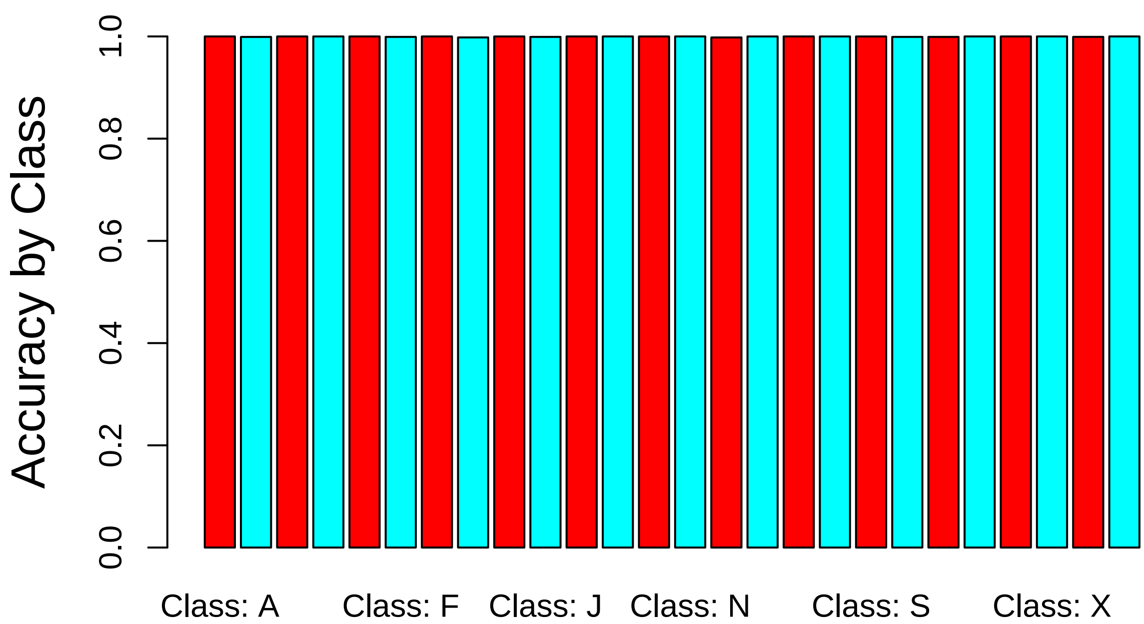
Training Accuracy:

```
pred <- predict(model,newdata=t1)
tab3 <- table(pred,t1$V1)
sum(diag(tab3))/sum(tab3)
```

```
## [1] 0.9992857
```

Confusion Matrix:

```
cmat <- confusionMatrix( tab3 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



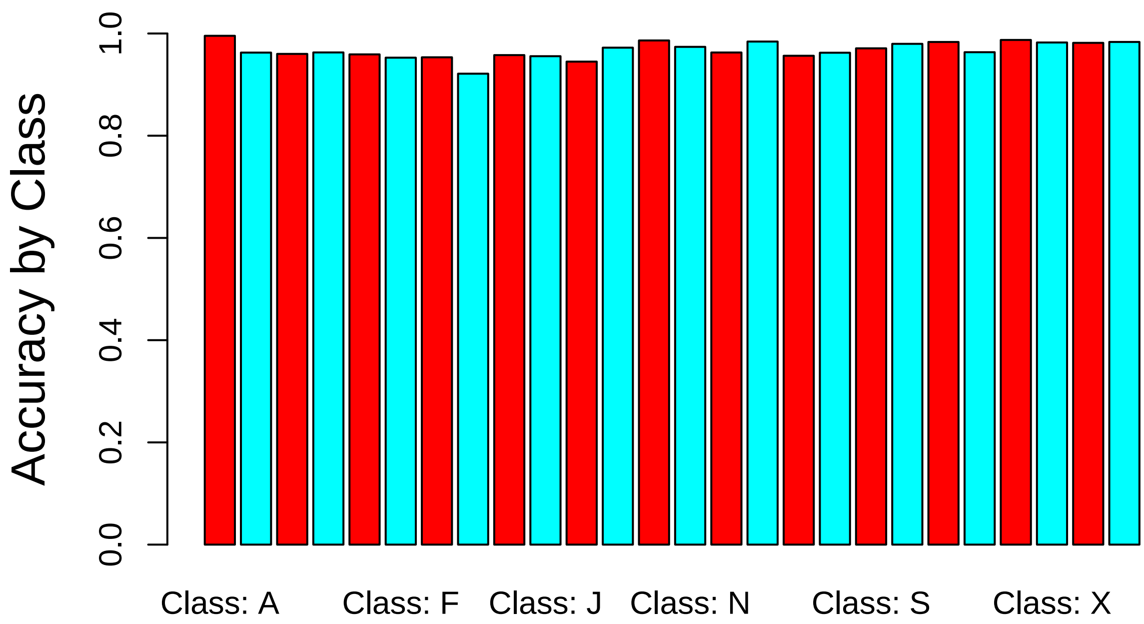
Test Data Accuracy:

```r
pred <- predict(model,newdata=t2)
tab4 <- table(pred,t2$V1)
sum(diag(tab4))/sum(tab4)
```

```
## [1] 0.9376667
```

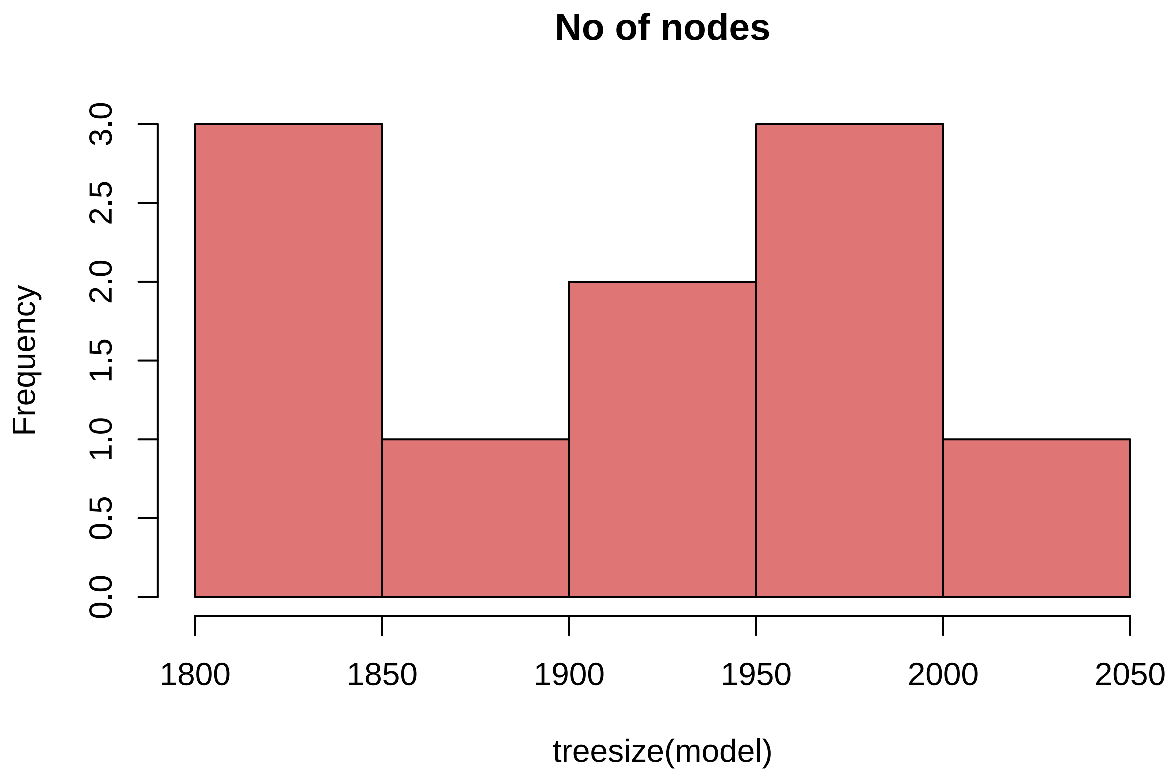
Confusion Matrix:

```r
cmat <- confusionMatrix( tab4 )
barplot(cmat$byClass[,11], ylim = c(0,1), ylab = "Accuracy by Class", cex.lab = 1.5, cex.main = 1.4, bes
```



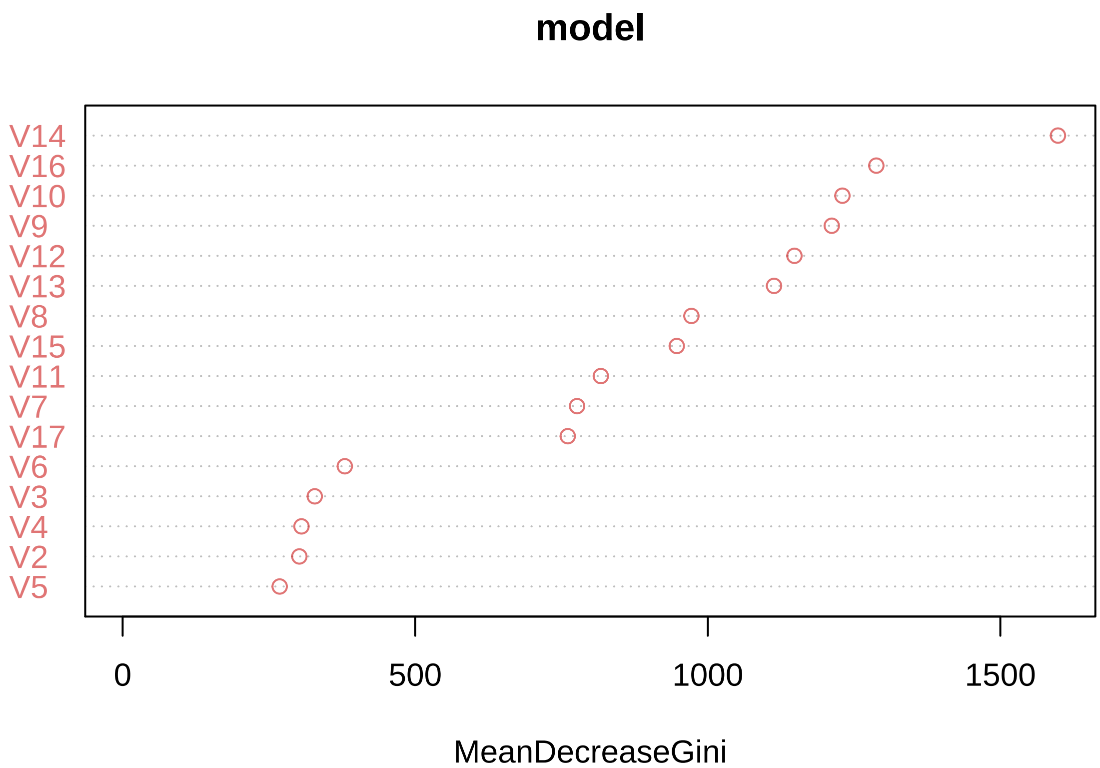
No of nodes in the tree vs frequency:

```r
hist(treesize(model), main = "No of nodes", col=rgb(0.8,0.1,0.1,0.6))
```



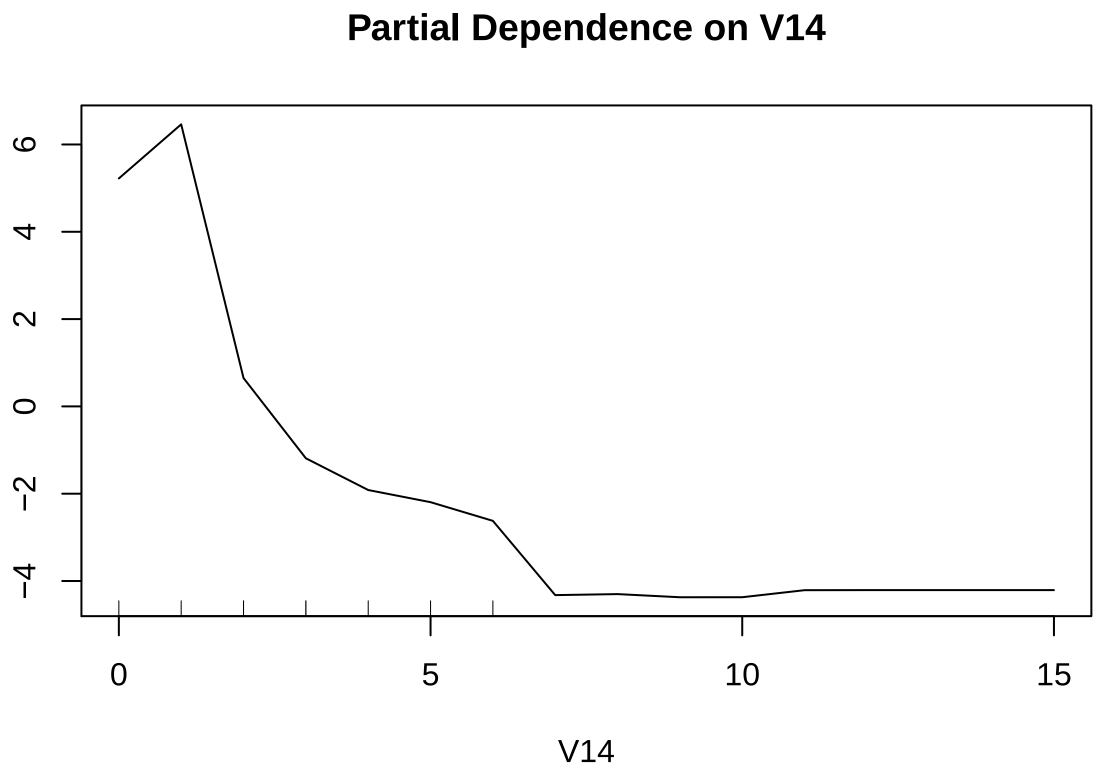
Importance of the various variables by virtue of their variance:

```
varImpPlot(model, col=rgb(0.8,0.1,0.1,0.6))
```

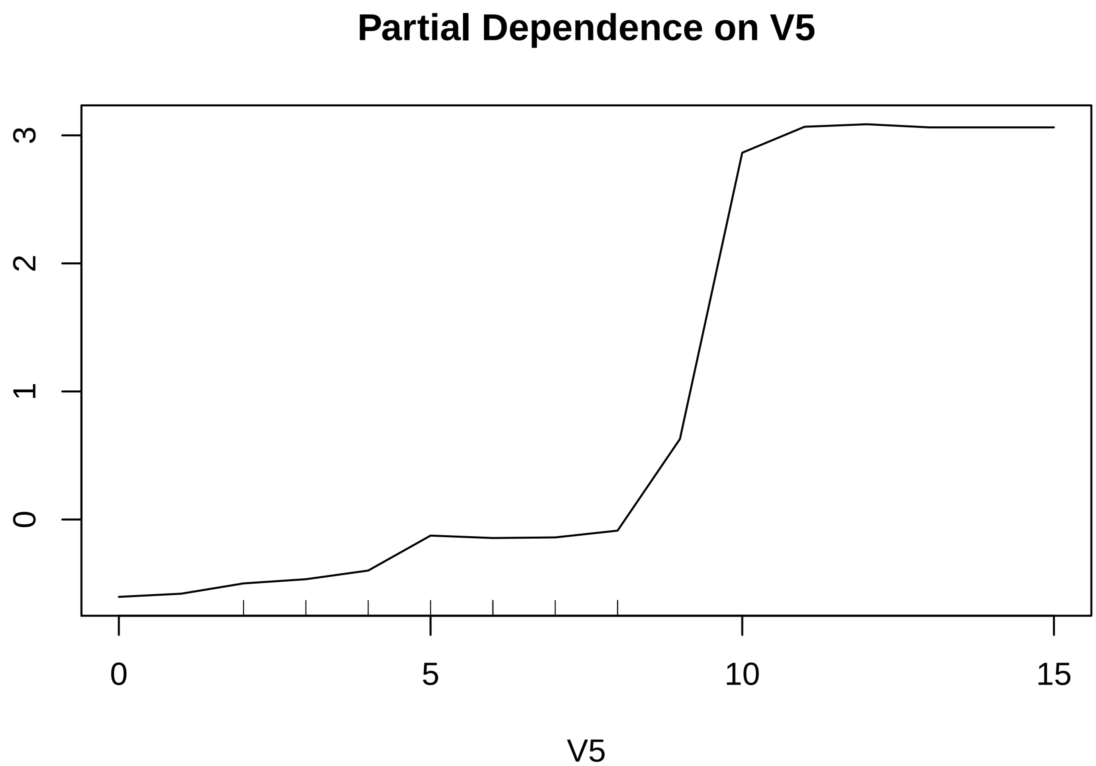


Partial Plots for plotting dependency of variable vs class label. 4 examples are illustrated:

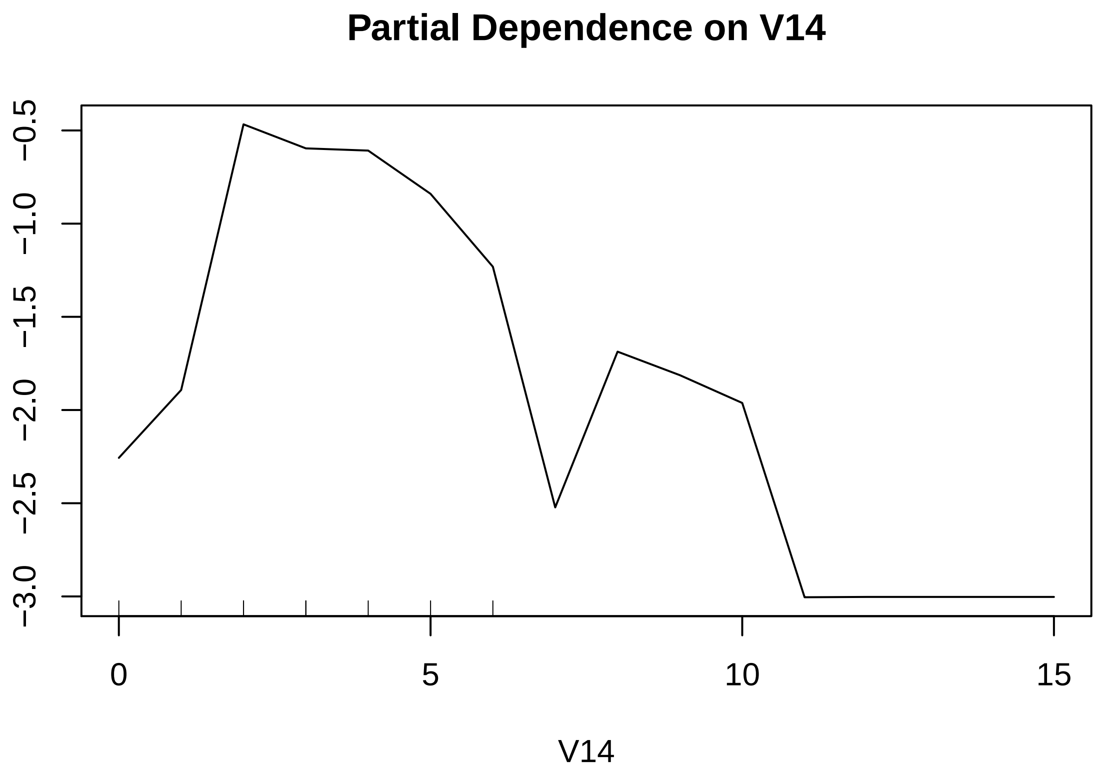
```
partialPlot(model, t1, V14, "Z")
```



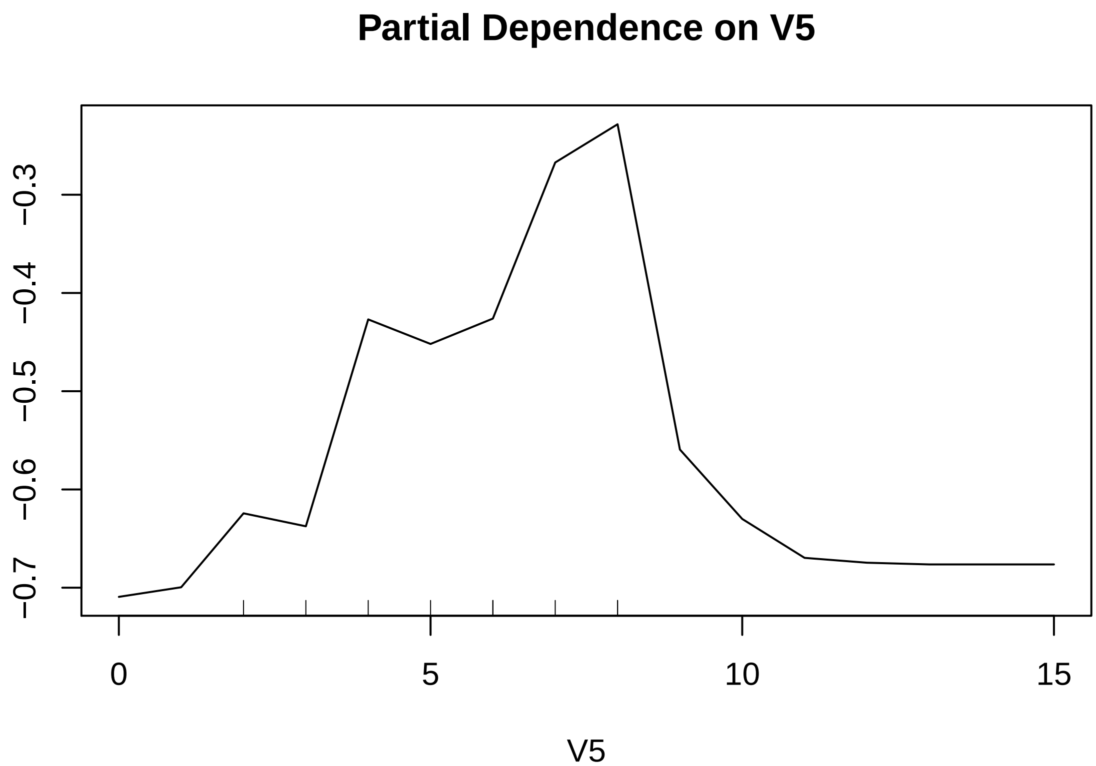

```r
partialPlot(model, t1, V5, "Z")
```




```r
partialPlot(model, t1, V14, "A")
```




```r
partialPlot(model, t1, V5, "A")
```


Partial Dependence on V5
−0.3
−0.4
−0.5
−0.6
−0.7
0
5
10
15
V5

Contributions to the assignment:

Aakash: Resolving princial components from data.

Akshat Lal: Code for LDA (classifier), Random Forest. Writing the full documentation/ report.

Kalash Shah: Code for LDA (Dimensionality reduction). Building upon PCA classifers to project data for dimensionality reduction.