ML Assignment 3

Usama Nadeem

2022-12-04

Q 3.1

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.2

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.2.2

## Loading required package: lattice
```

```
library (ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.2.2
```

```
library (ggforce)
```

```
## Warning: package 'ggforce' was built under R version 4.2.2
```

```
set.seed(12345)
data <- read.csv("communities.csv")
#scale all data except ViolentCrimesPerPop
ViolentCrimesPerPop_idx= which (colnames (data)== "ViolentCrimesPerPop")
scaler=preProcess(data[,-ViolentCrimesPerPop_idx])
ScaledData=predict(scaler,data[,-ViolentCrimesPerPop_idx])
CovScaleData <- cov(ScaledData)
# print (dim(CovScaleData))
EigenScaleData <- eigen(CovScaleData)
CommuVariance = 0
idx = 0

for(val in EigenScaleData$values)
{
  if(CommuVariance > 0.95)
  {
    break
  }
  else
  {
    CommuVariance = CommuVariance + (val / sum(EigenScaleData$values))
    idx = idx + 1
  }
}
print ("Total Components Needed to make the 95% varience in data:")
```

```
## [1] "Total Components Needed to make the 95% varience in data:"
```

```
print (idx)
```

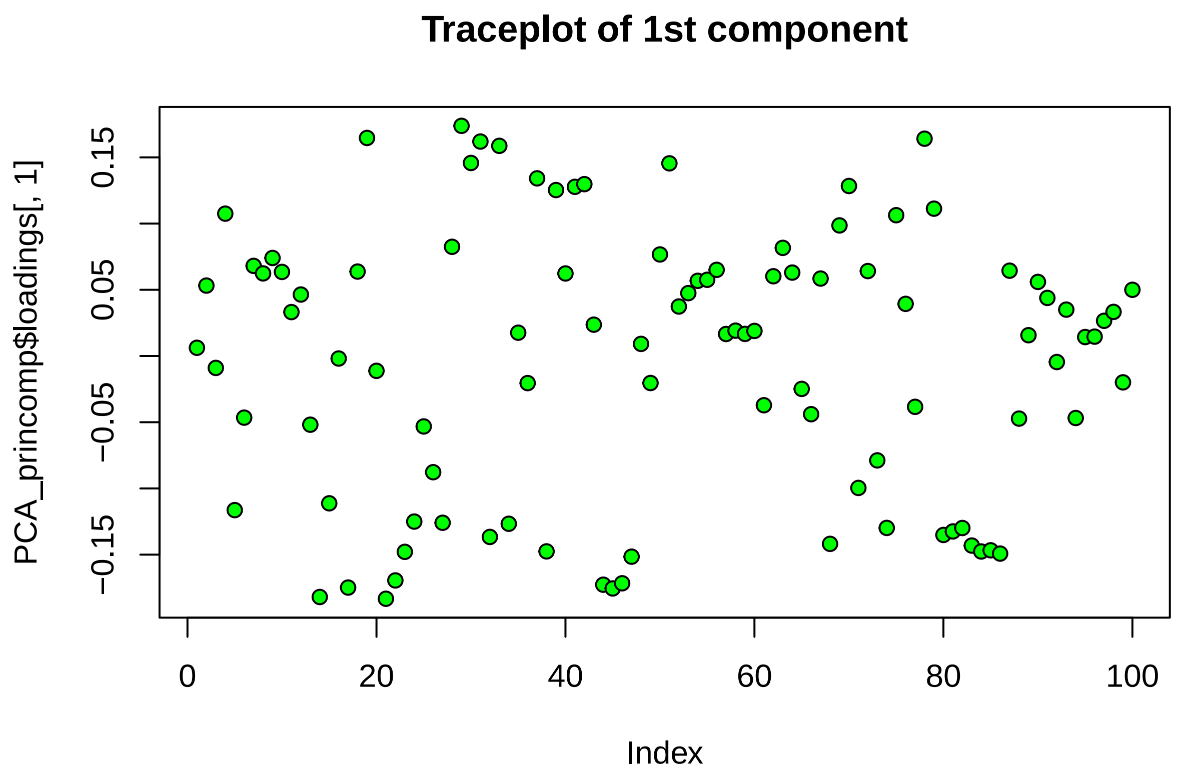
```
## [1] 35
```

```
SumOfAllPrincipalComponent = sum (EigenScaleData$values)
# First Two Principal Components are:
PC1 = EigenScaleData$values[1]
PC2 = EigenScaleData$values[2]
PC1Proportion = PC1/SumOfAllPrincipalComponent
PC2Proportion = PC2/SumOfAllPrincipalComponent
```

The First 35 components explain 0.953% of the variance. proportional variation explained by first 2 components is: 25% and 16.9% respectively.

Q 3.2

```
PCA_princomp <- princomp(ScaledData)

#make traceplot of 1st component
plot(PCA_princomp$loadings[,1],
     pch=21,
     bg="green",
     cex=1,
     main="Traceplot of 1st component"
)
```



features that contribute most to 1st principal component

```
FeaturesList <- abs(PCA_princomp$loadings[,1])
FeaturesListSorted = sort(FeaturesList)
print ("5 features that contribute most to 1st principal component: " )
```

```
## [1] "5 features that contribute most to 1st principal component: "
```

```
tail (FeaturesListSorted,5)
```

```
## PctPopUnderPov     pctWInvInc    PctKids2Par      medIncome      medFamInc
##      0.1737978      0.1748683      0.1755423      0.1819830      0.1833080
```

PctPopUnderPov: percentage of people under the poverty level

pctWInvInc: percentage of households with investment / rent income in 1989

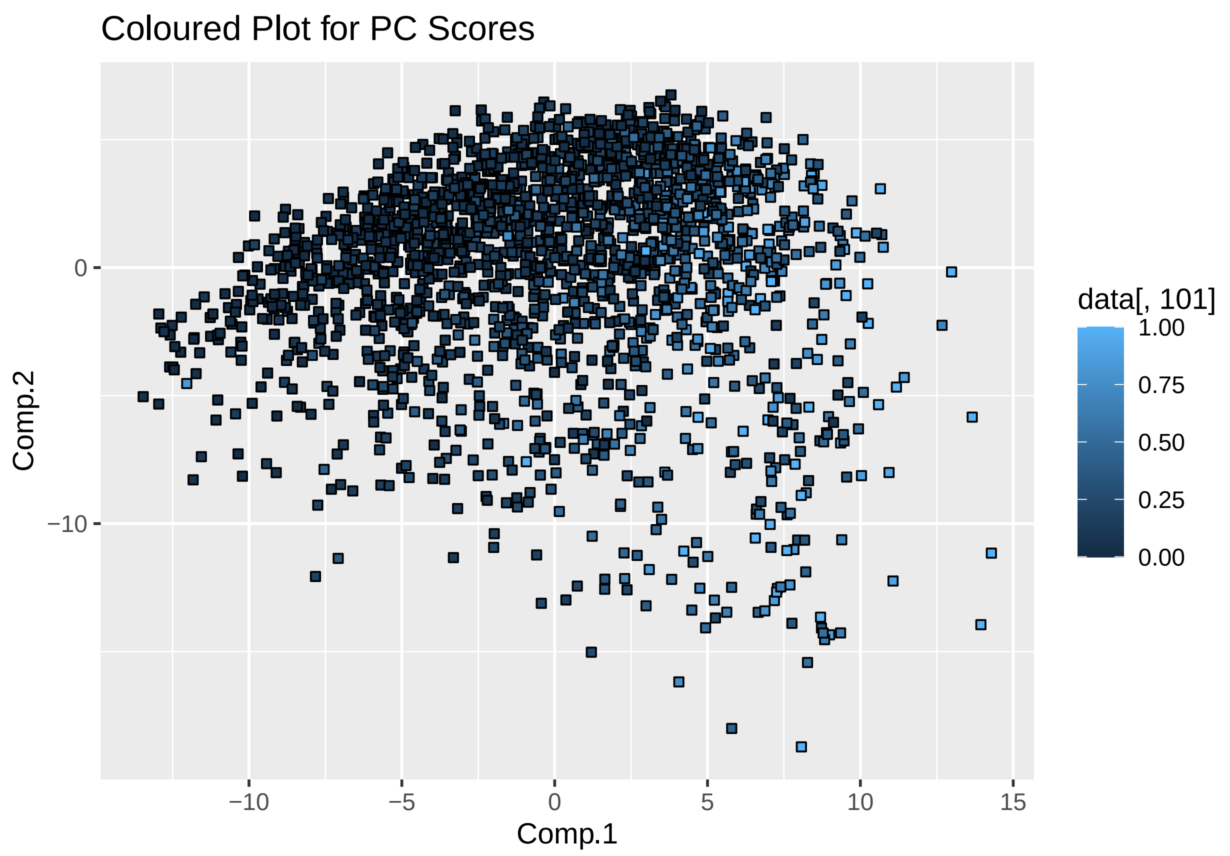
PctKids2Par: percentage of kids in family housing with two parents

medIncome: median household income

medFamInc: median family income (differs from household income for non-family households)

All of these factors are directly or indirectly related to the finances of the people. There is for sure the logical explanation of these factors on the crime level. Since people are disturbed financially, thereby they would be less resistant towards the criminal acts, hence impact on crime level.

```r
PC1PC2 = PCA_princomp$scores[, 1:2]
PCcoordinates = cbind(ScaledData, PC1PC2)
PCcoordinates_df <- as.data.frame(PCcoordinates)
ggplot(PCcoordinates_df, aes(x = Comp.1, y = Comp.2))+
  ggtitle("Coloured Plot for PC Scores")+
  geom_point(shape = 22, aes(fill = data[, 101]))
```



we can see that the left region of the graph has majority of the dark points, i.e. value of ViolentCrimesProp is near to zero, or very small. But as the value of comp 1 increase, the colour becomes lighter on the roght side of the graphs. If value of Comp 2 is increased too, along with Comp 1 the lighter region comes even at the middle-scale values of comp 1. This shows that both of these components play a vital role in depicting the values of ViolentCrimesProp. If value of both components is high, the violent crimes prop will be higher (top right side of the graph).

Q 3.3

```r
AllDataScaled <- as.data.frame(scale(data)) # Scale all data
Observations = dim(AllDataScaled)[1]
```

```
id=sample(1:Observations, Observations/2) #divide into 50-50
train = AllDataScaled [id,]
test = AllDataScaled [-id,]

LinearRegressionModel <- lm(ViolentCrimesPerPop ~ ., data = train)

CalculateMSE <- function(model, data)
{
  PredictedResult = predict(model, data)
  ActualResult = data[, "ViolentCrimesPerPop"]
  mean((ActualResult - PredictedResult)^2)
}

MSE_train <- CalculateMSE(LinearRegressionModel, train)
MSE_test <- CalculateMSE(LinearRegressionModel, test)

print ("Train MSE: ")
```

```
## [1] "Train MSE: "
```

```
print (MSE_train)
```

```
## [1] 0.2591772
```

```
print ("Test MSE: ")
```

```
## [1] "Test MSE: "
```

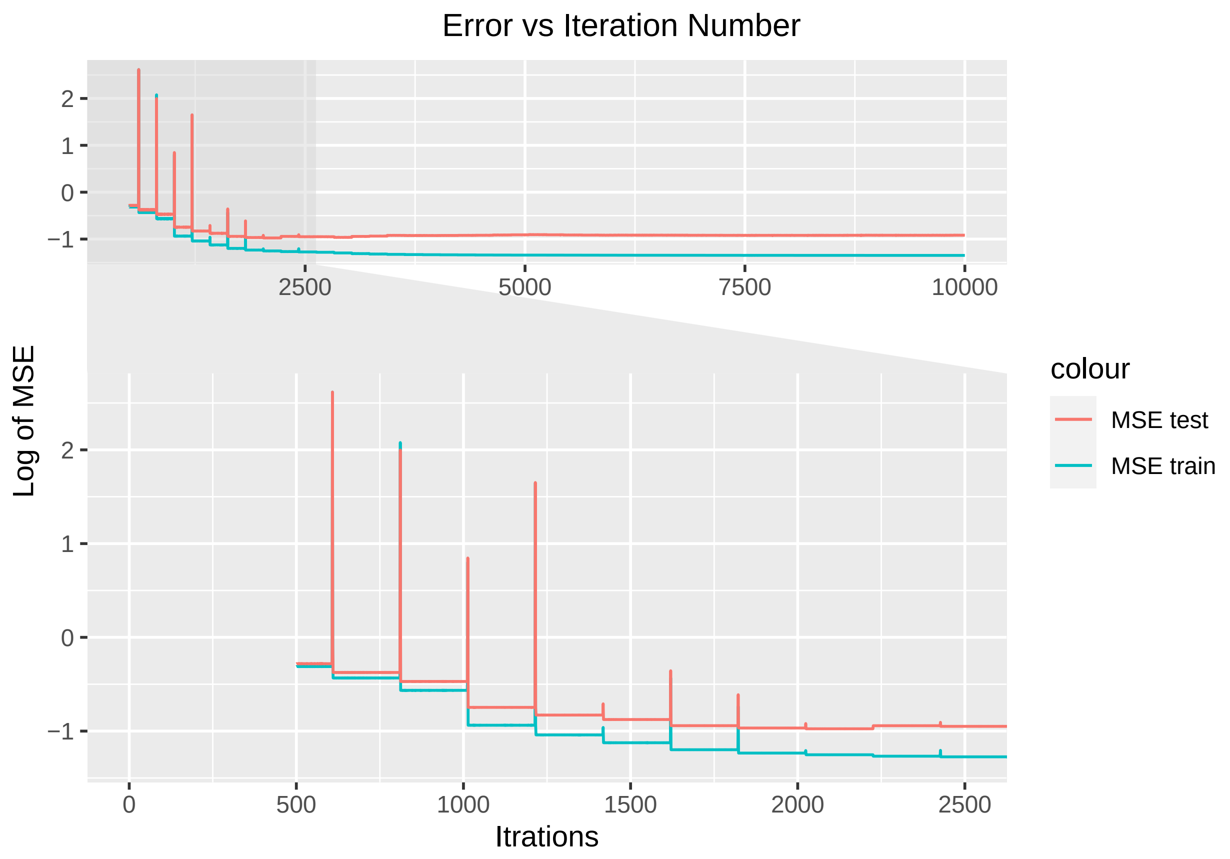
```
print (MSE_test)
```

```
## [1] 0.4000579
```

Q 3.4

```
Y_train = train$ViolentCrimesPerPop
train = train[,-ViolentCrimesPerPop_idx]
Y_test = test$ViolentCrimesPerPop
test = test[,-ViolentCrimesPerPop_idx]

X.train <- as.matrix(train)
X.test <- as.matrix(test)

n <- nrow(X.train)
theta <- rep(0, ncol(X.train))

TrainingError = list()
TestError = list()
k = 0
```

```r
CostFunc <- function(par){
  MSE_train <- sum((Y_train-(X.train %*% par))^2)/n # Being optimized
  k <<- k+1
  TrainingError[[k]] <<- MSE_train
  TestError[[k]] <<- sum((Y_test-(X.test %*% par))^2)/n # MSE for test
  return(MSE_train)
}
ans <-optim(par = theta, fn = CostFunc, method = "BFGS")


TrainTestItr <- do.call(rbind.data.frame, TrainingError)
TestError <- do.call(rbind.data.frame, TestError)
TrainTestItr <- cbind(TrainTestItr, TestError)
TrainTestItr$iteration = seq(1, nrow(TrainTestItr))
colnames(TrainTestItr) = c("MSE_train", "MSE_test", "Iteration")
temp <- TrainTestItr
TrainTestItr[, 1:2] <- log(TrainTestItr[,1:2])
Graph1 <- ggplot(data = TrainTestItr[500:10000,], aes(x=Iteration, fill=MSE_test))+
  geom_line(aes(y = MSE_train, color = "MSE train"))+
  geom_line(aes(y = MSE_test, color = "MSE test"))+
  ylab('Log of MSE')+
  xlab('Itrations') +
  facet_zoom(xlim = c(0, 2500))
annotate_figure(Graph1, top = text_grob("Error vs Iteration Number"))
```



Around the iteration 2000 to 2500, MSE for Test stops decreasing afterwards. This is the point where we should stop. Zoomed in graph, using ggforce library of R is also shown for better visualization. since lowest and stable MSE Test is nearly at Iteration Number # 2000, we can say this could bethe early stopping point. at this point looking at the values from graph, of we stop at itr 2000, we get:

$TrainMSE = exp(-1.2) = 0.30$

$TestMSE = exp(-0.9) = 0.40$