lab02-block02

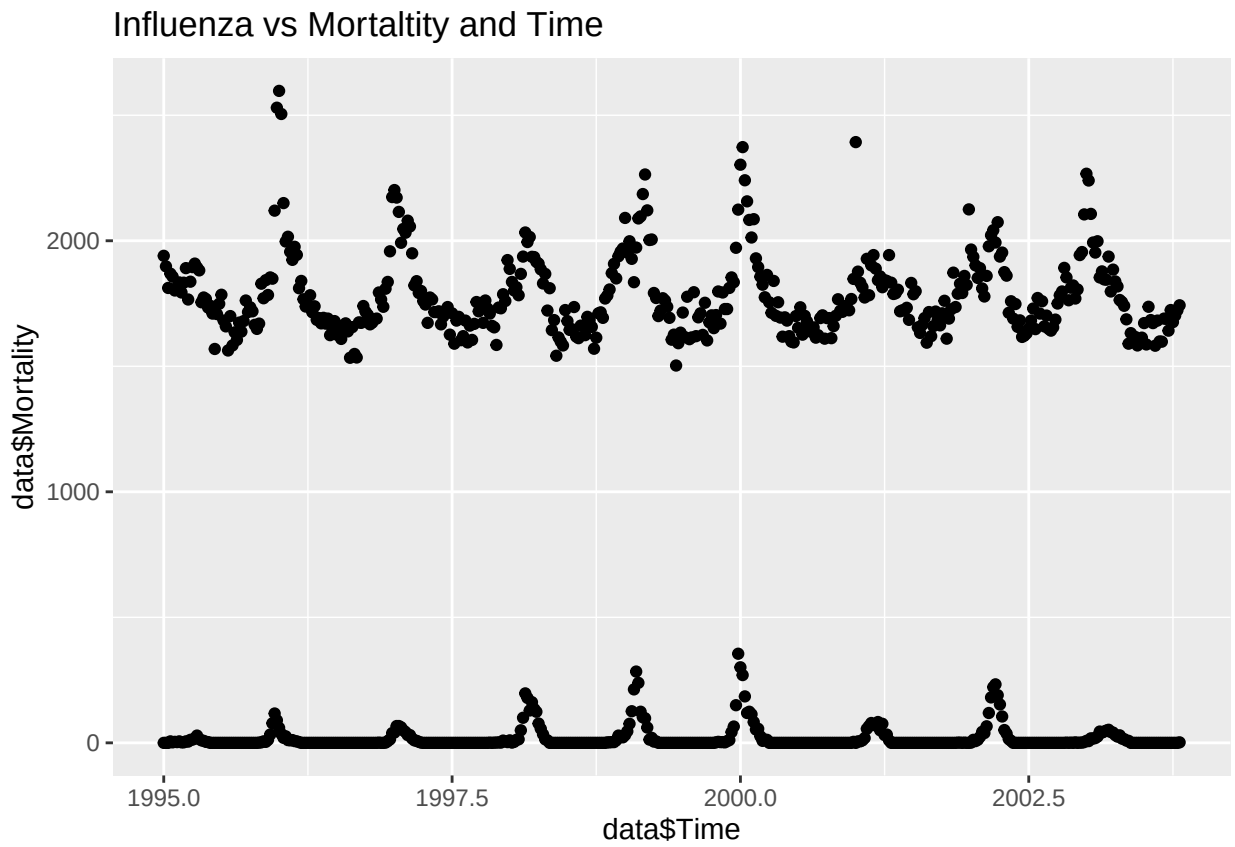
Mohammed Bakheet

15/12/2019

Assignment 1. Using GAM and GLM to examine the mortality rates

#1.1) plotting influenza vs mortality and time

```
ggplot(data = data)+geom_point(aes(data$Time, y = data$Mortality))+geom_point(aes(x = data$Time, y = da
```



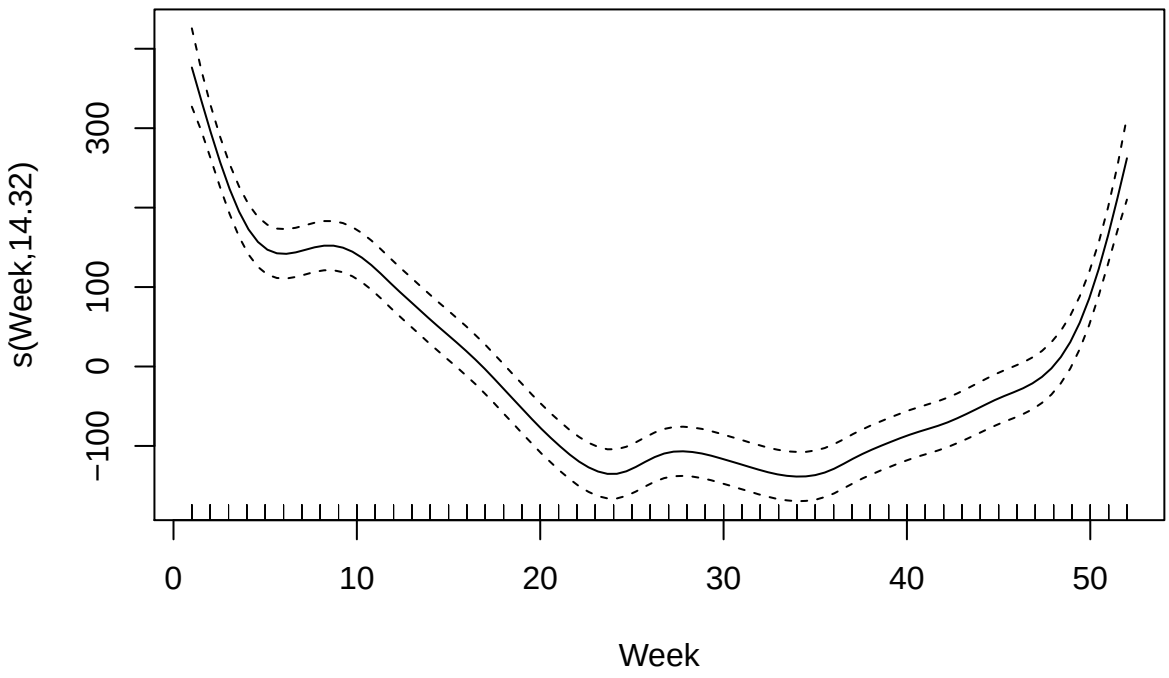
#The plot shows the realation between mortality and influenza with respect to time, it's clear that mortality rate increases as the influenza rate increases over time, So they do have a positive relationship.

#1.2)

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(data$Week)))
```

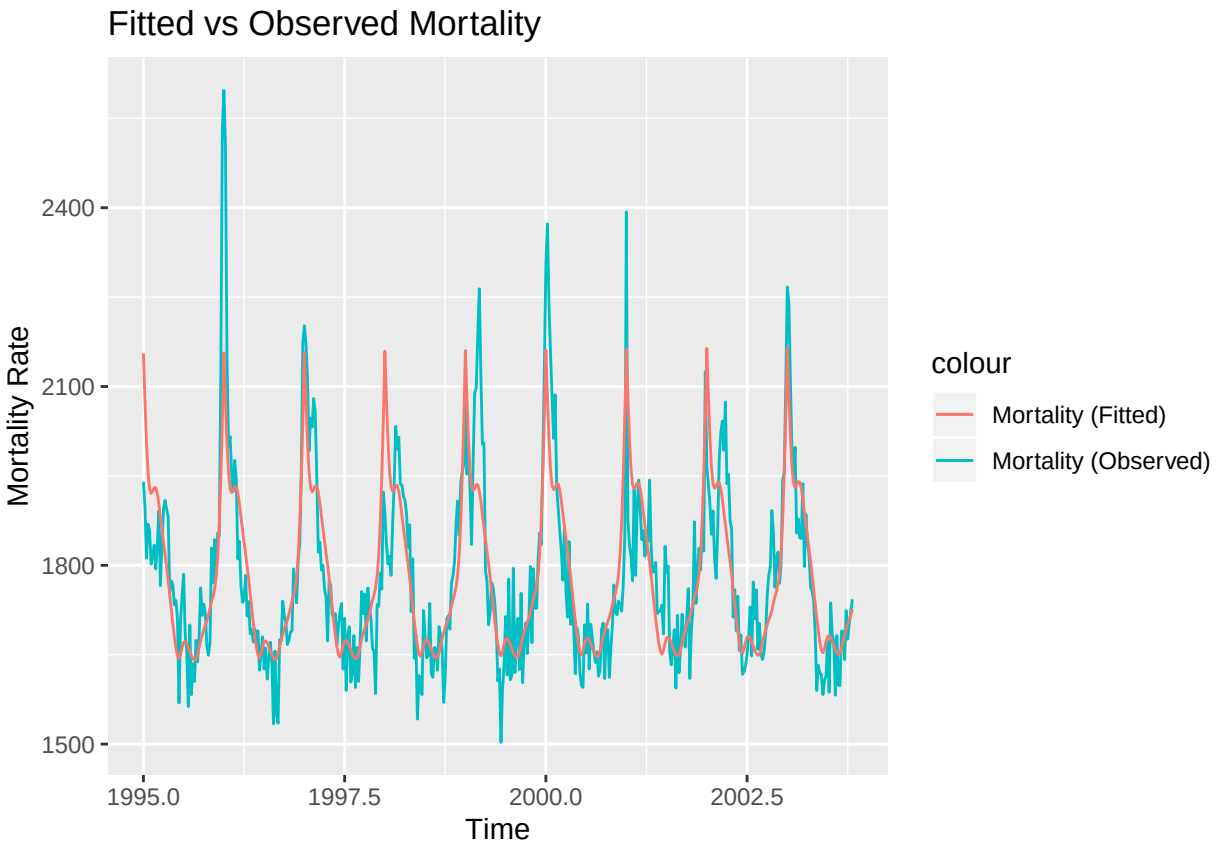
```
##
## Estimated degrees of freedom:
## 14.3  total = 16.32
##
## GCV score: 8708.581     rank: 52/53

##
## Family: gaussian
## Link function: identity
##
## Formula:
## Mortality ~ Year + s(Week, k = length(unique(data$Week)))
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -680.598   3367.760  -0.202    0.840
## Year           1.233      1.685   0.732    0.465
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Week) 14.32  17.87 53.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Rank: 52/53
## R-sq.(adj) =  0.677   Deviance explained = 68.8%
## GCV = 8708.6  Scale est. = 8398.9    n = 459
```

```
plot(res)
```



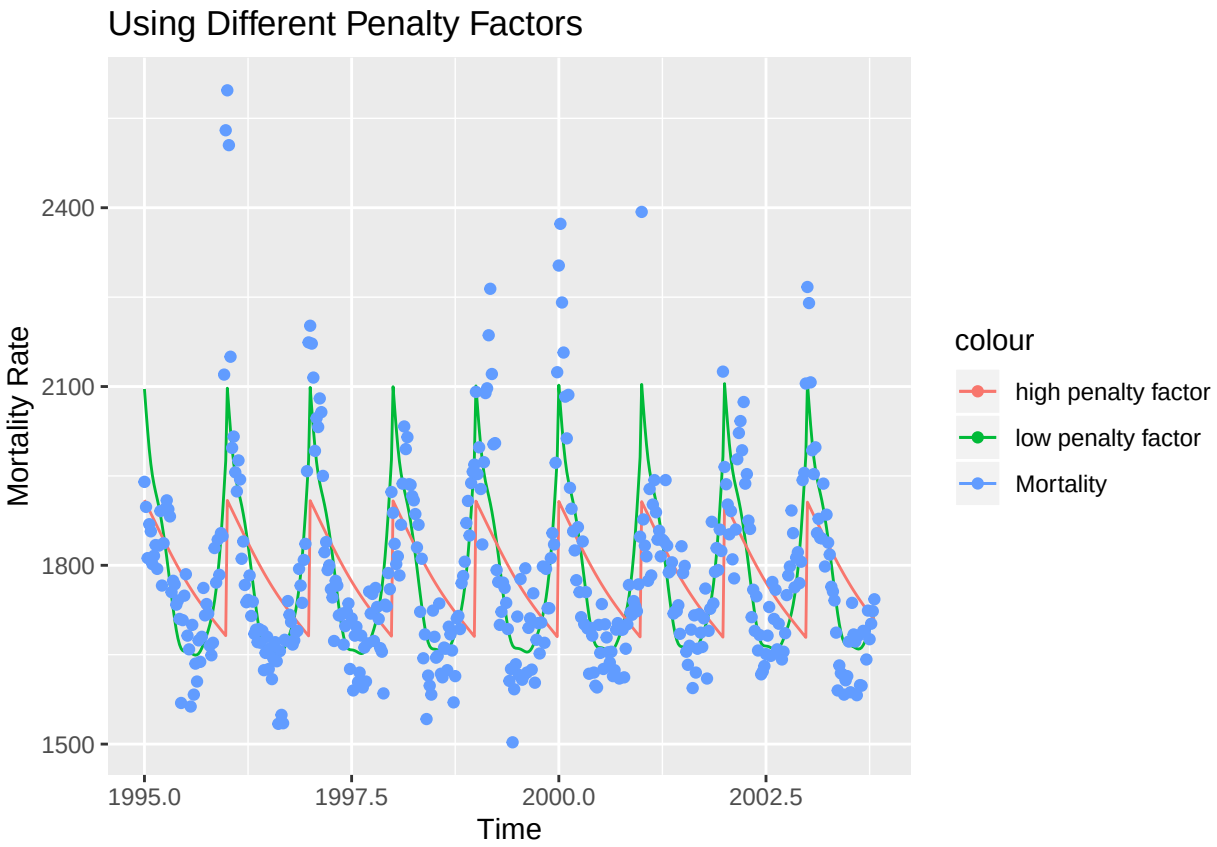
#1.3)

```
ggplot(data)+geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ggtitl
  geom_line(aes(x= Time, y = modelPrediction, col = "Mortality (Fitted)"))+ylab("Mortality Ra
```



#The model doesn't cover all data, and the trend appears to be the same for all years, this could be clearly seen in the graph. We could, therefore, conclude that the model is not the best fit.

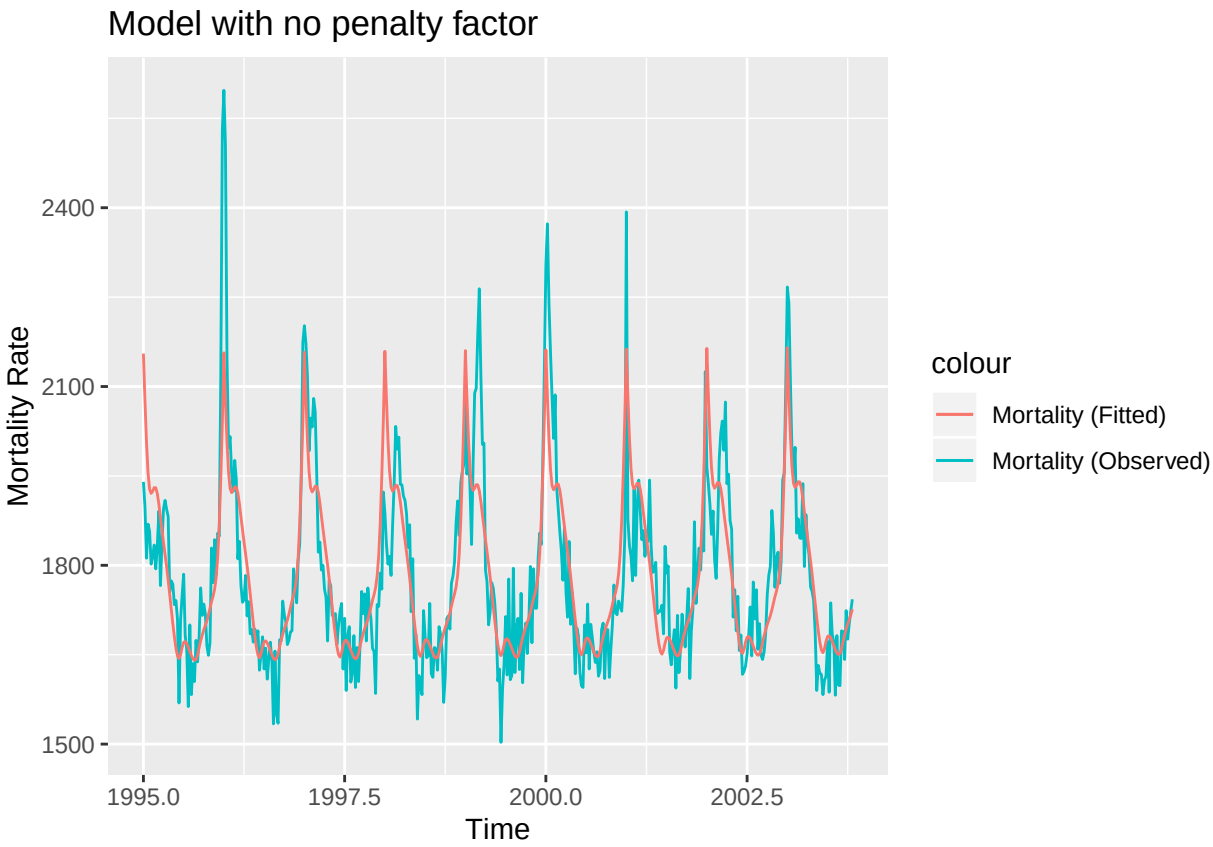
#1.4)



#With hight penalty factor then the model is underfitting the data, whereas when using a low penalty factor the model gets more overfitting.

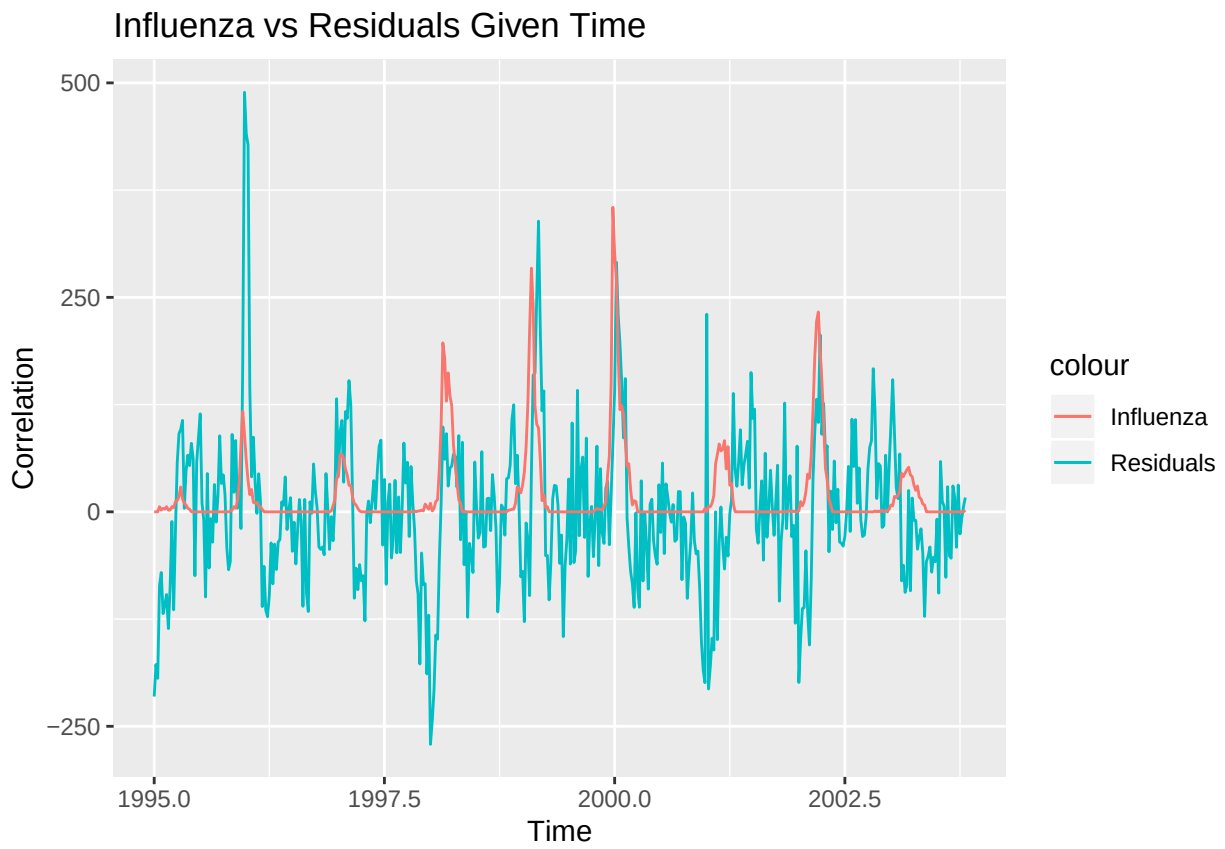
#Predicting the model without penalty factor

```
predictWithoutPenalty <- predict(modelWithNoDeviance, newdata = data)
ggplot(data)+geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ggtitl
  geom_line(aes(x= Time, y = modelPrediction, col = "Mortality (Fitted)"))+ylab("Mortality Ra
```



#1.5)

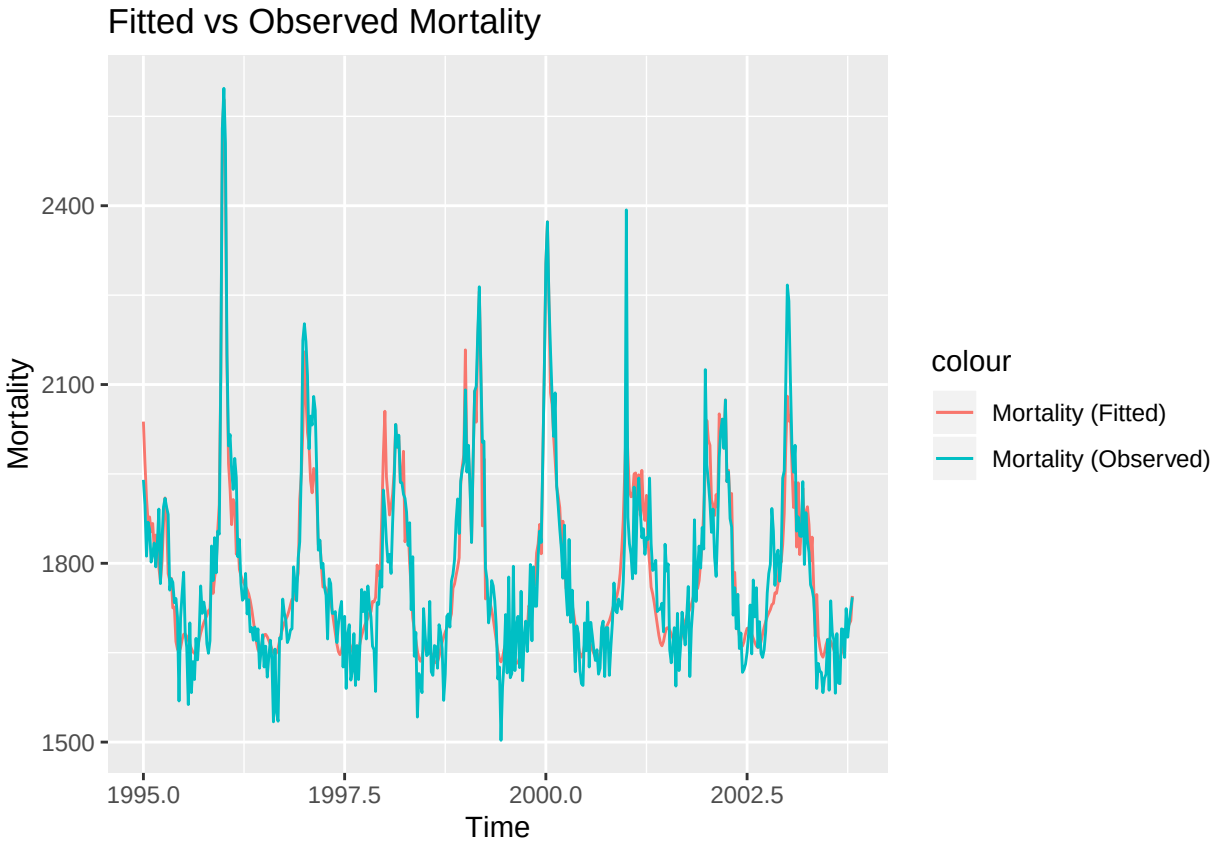
```
ggplot(data) + geom_line(aes(x = Time, y = residuals(res) , col = "Residuals"))+ggtitle("Influ
geom_line(aes(x = Time, y = Influenza, col = "Influenza"))+ylab("Correlation")+xlab("Time")
```



#As we can see from the plot that the residuals have a correlation with the influenza rate, they increase as the influenza increases.

#1.6)

```
ggplot(data) + geom_line(aes(x = Time, y = gamModelFitPrediction , col = "Mortality (Fitted)"
  geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ylab("Mortality"
```



#From the plot it appears that the model is better than all the previous model.

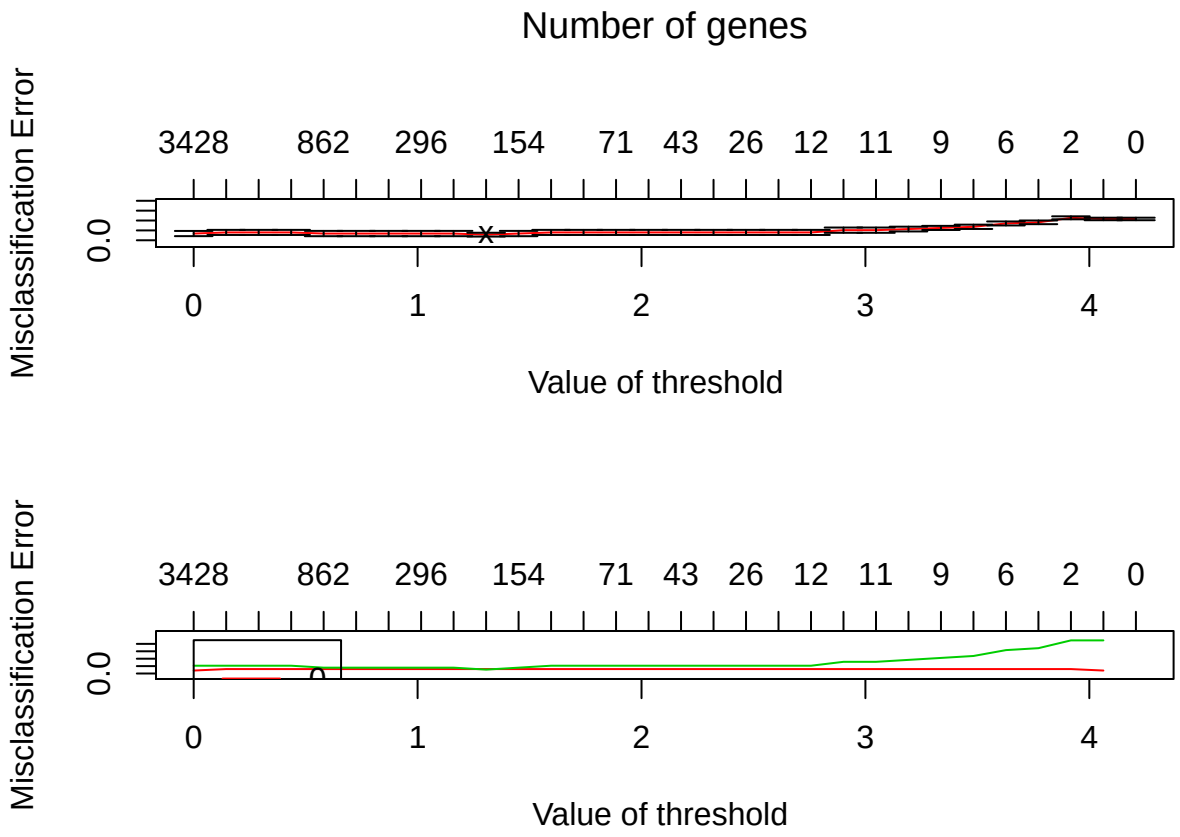
Assignment 2. High-dimensional methods

#2.1)

```
## 12345678910111213141516171819202122232425262728293O
```

```
## 12Fold 1 :12345678910111213141516171819202122232425262728293O
## Fold 2 :12345678910111213141516171819202122232425262728293O
## Fold 3 :12345678910111213141516171819202122232425262728293O
## Fold 4 :12345678910111213141516171819202122232425262728293O
## Fold 5 :12345678910111213141516171819202122232425262728293O
## Fold 6 :12345678910111213141516171819202122232425262728293O
## Fold 7 :12345678910111213141516171819202122232425262728293O
## Fold 8 :12345678910111213141516171819202122232425262728293O
## Fold 9 :12345678910111213141516171819202122232425262728293O
## Fold 10 :12345678910111213141516171819202122232425262728293O
```

```
##  [1] 0.0000000 0.1451037 0.2902074 0.4353112 0.5804149 0.7255186 0.8706223
##  [8] 1.0157260 1.1608297 1.3059335 1.4510372 1.5961409 1.7412446 1.8863483
## [15] 2.0314520 2.1765558 2.3216595 2.4667632 2.6118669 2.7569706 2.9020744
## [22] 3.0471781 3.1922818 3.3373855 3.4824892 3.6275929 3.7726967 3.9178004
## [29] 4.0629041 4.2080078
```



#Fitting the model with cross validated threshold

```
        model <- pamr.train(mydata, threshold = cv.fit$threshold[which.min(cv.fit$error)])
```

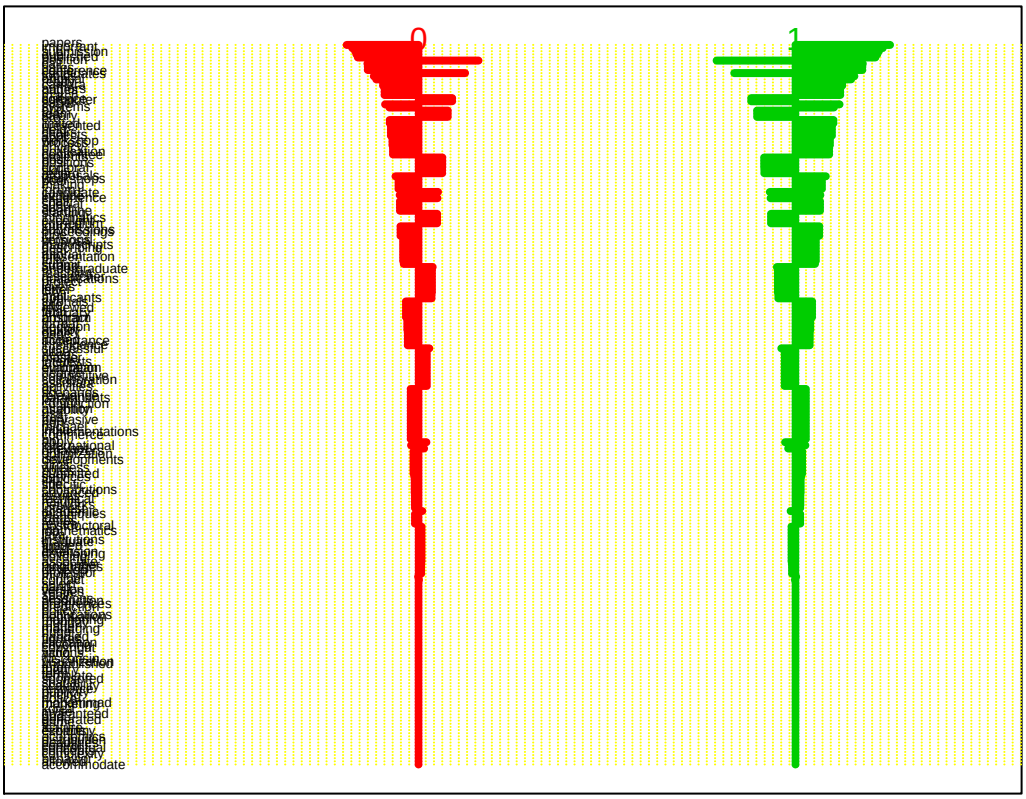
```
## 1
```

```
        summary(model)
```

```
##                  Length Class  Mode
## y                    44 factor numeric
## proby                 0 -none- NULL
## yhat                 44 factor numeric
## prob                 88 -none- numeric
## centroids          9404 -none- numeric
## centroid.overall   4702 -none- numeric
## sd                 4702 -none- numeric
## threshold             1 -none- numeric
## nonzero               1 -none- numeric
## threshold.scale       2 -none- numeric
## se.scale              2 table  numeric
## scale.sd              1 -none- logical
## call                  3 -none- call
## hetero                0 -none- NULL
## norm.cent             0 -none- NULL
## prior                 2 table  numeric
```

```
## offset              1   -none- numeric
## sign.contrast       1   -none- character
## errors              1   -none- numeric
## gene.subset      4702   -none- numeric
## sample.subset      44   -none- numeric
## ngroup.survival     1   -none- numeric
## problem.type        1   -none- character
```

#Plotting the centroid



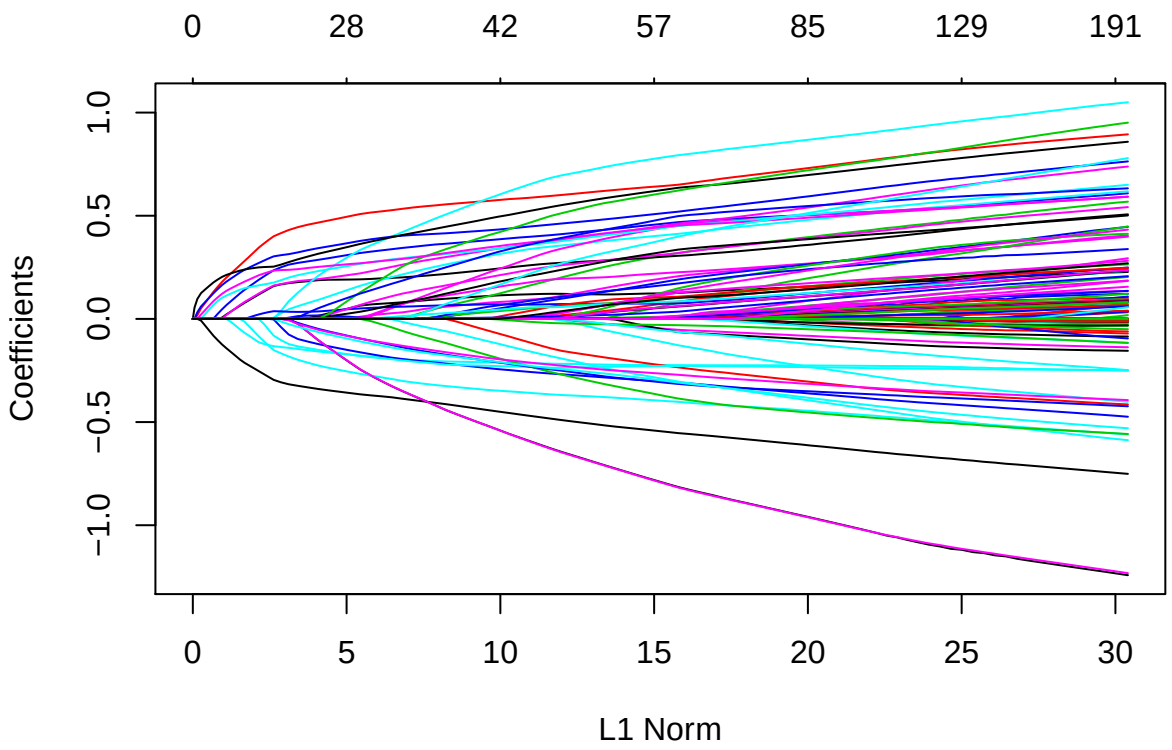
#Listing the most significant 10 genes:

#231 features are selected by the model using CV method.

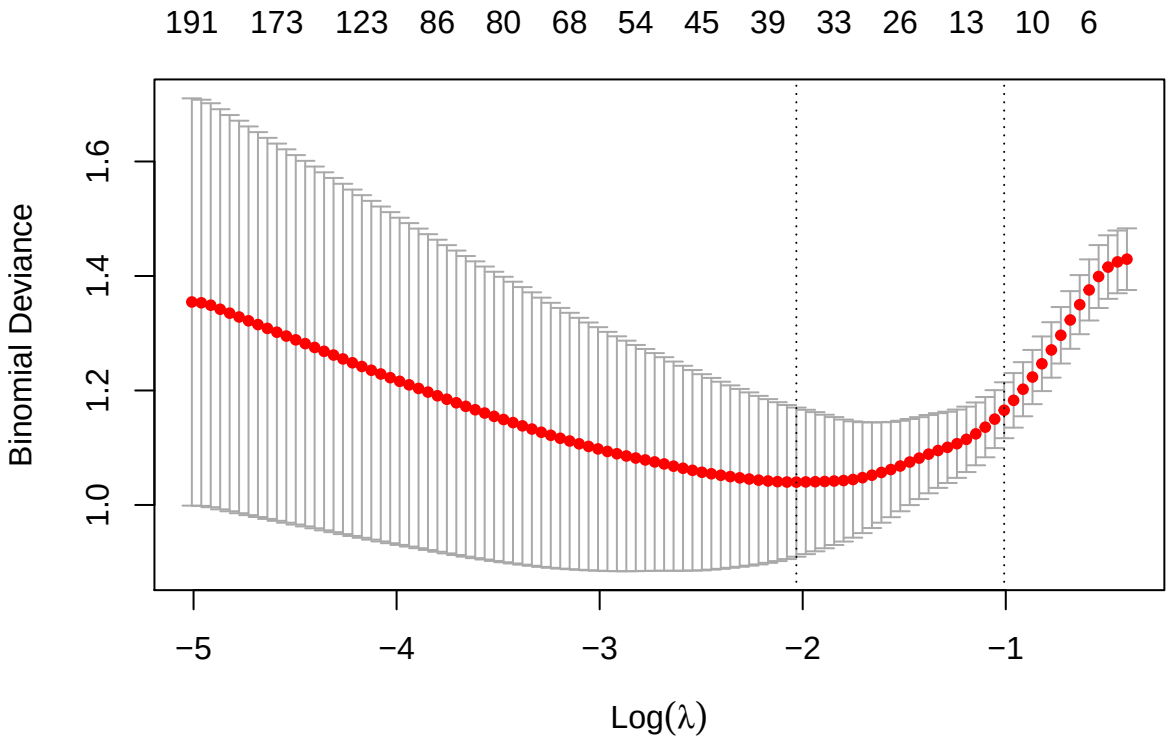
#The top ten features are: #papers
#important
#submission
#due
#published
#position
#call
#conference
#dates
#candidates
#All ten features selected are reasonable to contribute to deciding whether or not the email is a conference email.

#Test Error:

```
## [1] 0.1
```


0
28
42
57
85
129
191
1.0
0.5
0.0
−0.5
−1.0
Coefficients
0
5
10
15
20
25
30
L1 Norm

#2.2) #a)



#Elastic model error is:

```
## [1] 0.1
```

#The contributing features to the model are:

```
##  [1] "abstracts"     "aspects"       "attention"     "bio"
##  [5] "call"          "candidates"    "computer"      "conceptual"
##  [9] "conference"    "dates"         "due"           "evaluation"
## [13] "exhibits"      "forum"         "important"     "interests"
## [17] "languages"     "making"        "manuscripts"   "original"
## [21] "papers"        "peer"          "position"      "privacy"
## [25] "projects"      "proposals"     "published"     "queries"
## [29] "record"        "relevant"      "scalability"   "scenarios"
## [33] "spatial"       "submission"    "systems"       "team"
## [37] "versions"      "visualization"
```

```
## [1] 38
```

#b)Error rate and features for the kernel model

```
##  Setting default kernel parameters
```

#Testing error is:

```
## [1] 0.05
```

#The number of features for the kernel model is:

```
library(knitr)
              featuresMatrixKernel <- coef(kernfit)
              length(featuresMatrixKernel[[1]])
```

```
## [1] 43
```

```
comparativeDataFrame <- data.frame(Model = c("Centroid","Elastic", "SVM"), Features =c(231,38,43), Erro

kable(comparativeDataFrame, caption="Comparison Table")
```

Table 1: Comparison Table

Model	Features	Error
Centroid	231	0.10
Elastic	38	0.10
SVM	43	0.05

#SVM selects more features than Elastic and less than Centroid, however, it has the lowest missclassification rate of 0.05. SVM could be considered as best.

#2.3) The number of features to be rejected:

```
## [1] 39
```

#After calculating the p-values for all the features, ordering all unadjusted p-values, and finding the highest rank j for which the p-value (pj) is less than or equal (j/m)*alpha. We reject all the hypotheses for which pj <= p(L), that are 39 hypotheses. The 39 features are:

```
##  [1] "papers"        "submission"    "position"      "published"
##  [5] "important"     "call"          "conference"    "candidates"
##  [9] "dates"         "paper"         "topics"        "limited"
## [13] "candidate"     "camera"        "ready"         "authors"
## [17] "phd"           "projects"      "org"           "chairs"
## [21] "due"           "original"      "notification"  "salary"
## [25] "record"        "skills"        "held"          "team"
## [29] "pages"         "workshop"      "committee"     "proceedings"
## [33] "apply"         "strong"        "international" "degree"
## [37] "excellent"     "post"          "presented"
```

Code Appendix

```
RNGversion('3.5.1')
knitr::opts_chunk$set(echo = TRUE)
```

```r
packages <- c("ggplot2", "plotly","readxl","tree", "MASS", "e1071", "boot", "fastICA","mgcv","akima","pl
options(tinytex.verbose = TRUE)
library(mgcv)
library(akima)
library(plotly)
library(readxl)
data = read_excel("D:/Desktop/Machine Learning/Machine Learning/lab02 block 2/influenza.xlsx")
set.seed(12345)
ggplot(data = data)+geom_point(aes(data$Time, y = data$Mortality))+geom_point(aes(x = data$Time, y = da

res=gam(Mortality~Year+s(Week,
                         k=length(unique(data$Week))),
                         data=data, method = "GCV.Cp")

          s=interp(data$Year,data$Week, fitted(res), duplicate = TRUE)
          print(res)
          summary(res)

          plot(res)


          modelPrediction <- predict(res,newdata = data)
         ggplot(data)+geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ggtitl
           geom_line(aes(x= Time, y = modelPrediction, col = "Mortality (Fitted)"))+ylab("Mortality Ra
          modelWithNoDeviance=gam(Mortality~Year+s(Week,
                         k=length(unique(data$Week)), sp=0.002),
                         data = data)
  modelWithNoDeviance2=gam(Mortality~Year+s(Week,
                         k=length(unique(data$Week)), sp=10),
                         data = data)
  firstModel <- predict(modelWithNoDeviance)
  secondModel <- predict(modelWithNoDeviance2)
         ggplot(data)+geom_line(aes(x = Time, y = firstModel, col = "low penalty factor"))+ggtitle("Usi
           geom_line(aes(x= Time, y = secondModel, col = "high penalty factor"))+ylab("Mortality Rate"
          predictWithoutPenalty <- predict(modelWithNoDeviance, newdata = data)
          ggplot(data)+geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ggtitl
           geom_line(aes(x= Time, y = modelPrediction, col = "Mortality (Fitted)"))+ylab("Mortality Ra
          ggplot(data) + geom_line(aes(x = Time, y = residuals(res) , col = "Residuals"))+ggtitle("Infl
          geom_line(aes(x = Time, y = Influenza, col = "Influenza"))+ylab("Correlation")+xlab("Time")
          gamModelFit = gam(formula =  Mortality~s(Year, k=length(unique(data$Year)))
                           +s(Week, k=length(unique(data$Week)))
                           +s(Influenza, k=length(unique(data$Influenza)))
                           , data =  data)
          gamModelFitPrediction <- predict(gamModelFit, newdata = data)
          ggplot(data) + geom_line(aes(x = Time, y = gamModelFitPrediction , col = "Mortality (Fitted)"
           geom_line(aes(x = Time, y = data$Mortality, col = "Mortality (Observed)"))+ylab("Mortality"
#Importing the data from the csv file
data0=read.csv2("D:/Desktop/Machine Learning/Machine Learning/lab02 block 2/data.csv")
data=data0
data=as.data.frame(data)
data$Conference=as.factor(data$Conference)
rownames(data)=1:nrow(data)
```

```r
#Training and testing data
n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.7))
training=data[id,]
testing=data[-id,]
library(pamr)
        x=t(training[,-4703])
        y=training[[4703]]
        mydata=list(x=x,y=as.factor(y),geneid=as.character(1:nrow(x)), genenames=rownames(x))

    #Fitting the model
        model=pamr.train(mydata)

    #Cross-validating the penalty
        cv.fit <- pamr.cv(model, mydata)
        cv.fit$threshold
        pamr.plotcv(cv.fit)
        model <- pamr.train(mydata, threshold = cv.fit$threshold[which.min(cv.fit$error)])
        summary(model)
          pamr.plotcen(model, mydata, threshold=cv.fit$threshold[which.min(cv.fit$error)])

          cvMin <- cv.fit$threshold[which.min(cv.fit$error)]
          a<-pamr.listgenes(model,mydata,threshold=cvMin)
          cat(paste(colnames(data)[as.numeric(a[1:10,1])],collapse='\n'))
testingTranspose <- t(testing[,-4703])
predictModelTest <- pamr.predict(model,newx = testingTranspose, threshold =cv.fit$threshold[which.min(cv
summed <- table(testing[,4703], predictModelTest)

errorValue <- 1-sum(diag(summed))/sum(summed)
errorValue
library(glmnet)
          x2 <- as.matrix(training[,-4703])
          x3 <- as.matrix(testing[,-4703])
          yt <- as.matrix(testing[,4703])
          y <- as.matrix(training[,4703])
          elasticModel <- glmnet(x2, y, alpha = 0.5, family = "binomial")
          plot(elasticModel)
          #Calculating the penalty by corss-validation
          set.seed(12345)
          cv <- cv.glmnet(x2, y = y, alpha = 0.5,family = "binomial")
          plot(cv)
          #Fitting the model with the minimum cross-validated penalty
          elasticModel <- glmnet(x2, y, alpha = 0.5, lambda = cv$lambda.min, family = "binomial")



          elasticPredictTesting <- predict(elasticModel, x3, type = "class")
          elasticCM <- table(elasticPredictTesting, testing$Conference)

          elasticError <- 1-sum(diag(elasticCM))/sum(elasticCM)
elasticError
```

```
          #The contributing features are:
          featuresMatrix <- as.matrix(coef(elasticModel))
          k <- arrayInd(which((featuresMatrix>0) | (featuresMatrix<0)),dim(featuresMatrix))
          contributingFeature1 <- rownames(featuresMatrix)[k[-1,1]]
          totalFeaturesNo <- length(contributingFeature1)
          contributingFeature1
          totalFeaturesNo
library(kernlab)
          kernfit <- ksvm(Conference~., training, kernel = "vanilladot", scale = FALSE)
          kernfitPredict <- predict(kernfit,newdata=training)
          kernfitPredictTest <- predict(kernfit,newdata=testing, type="response")

          # Making the Confusion Matrix
          cm = table(kernfitPredict, training$Conference)
          cmTest = table(kernfitPredictTest, testing$Conference)
          errorKernal <- 1-sum(diag(cmTest)/sum(cmTest))
         errorKernal
library(knitr)
              featuresMatrixKernel <- coef(kernfit)
              length(featuresMatrixKernel[[1]])

comparativeDataFrame <- data.frame(Model = c("Centroid","Elastic", "SVM"), Features =c(231,38,43), Erro

kable(comparativeDataFrame, caption="Comparison Table")
#computing the p-value
          pValues <- c()
          limit <- ncol(data)-1
          for(feature in 1:limit){
            x <- data[,feature]
            y <- data[,4703]
            z <- t.test(as.matrix(x)~y, data = data, alternative = "two.sided")
            pValues <- c(pValues, z$p.value)
          }
#Ordered p values
          pValueDf <- data.frame("pvalue" = pValues, "Index" = 1:length(pValues))
          orederdPvalues <- pValueDf[order(pValueDf$pvalue),]

      significantBhFeatures <- matrix(ncol = 2,nrow = nrow(orederdPvalues))
      dfnames <- c("feature","index")
      colnames(significantBhFeatures)<- dfnames

#Benjamini and Hochberg FDR
          for(j in 1:nrow(orederdPvalues)){
            x <- (j/nrow(orederdPvalues))*0.05
            if (orederdPvalues[j,"pvalue"]<=x){
              significantBhFeatures[j,1] <- orederdPvalues[j,"pvalue"]
              significantBhFeatures[j,2] <- orederdPvalues[j,"Index"]
            }
          }
      maximumJ <- which.max(significantBhFeatures[,1])
      maximumJ

 colnames(data)[significantBhFeatures[1:39,"index"]]
```