Computer Lab 2 Block 1

Rojan Karakaya and Suhani Ariga

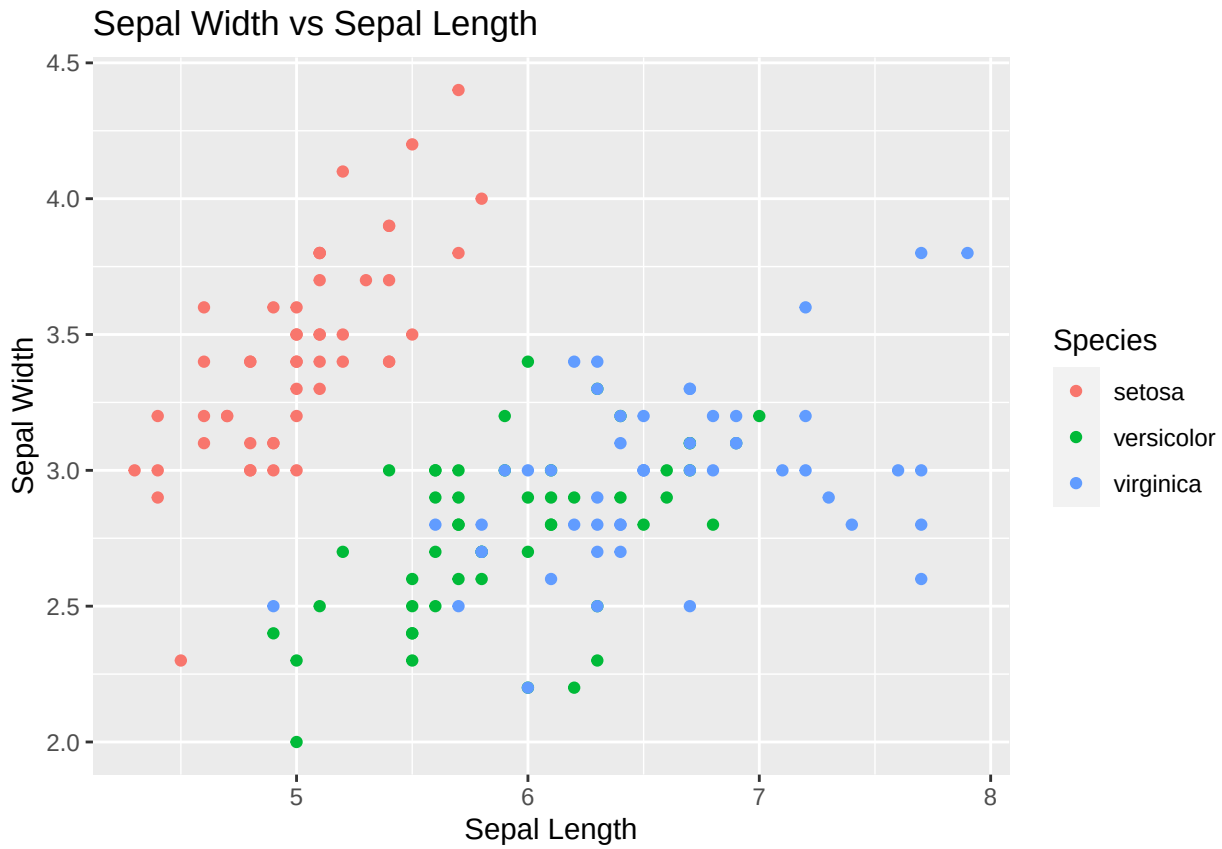
31/01/2021

Contents

Assignment 1: LDA and logistic regression

1.

```
ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
```



Linear Discriminant Analysis(LDA) focuses on maximizing seperatibility among known categories. In tht plot, setosa will be easily classified as they are separated but it will be difficult to separate versicolor and virginica using LDA since there is no border between the two.

2.

```
#a)
data = iris
X = data.frame(SL=data$Sepal.Length,SW=data$Sepal.Width)
Y = data$Species
X1 = X[Y=="setosa",]
X2 = X[Y=="versicolor",]
X3 = X[Y=="virginica",]

mean1 = c(mean(X1$SL) ,mean(X1$SW))#c(mean(X1$SW) ,mean(X1$SL))
```

```r
mean2 = c(mean(X2$SL) ,mean(X2$SW))#c(mean(X2$SW) ,mean(X2$SL))
mean3 = c(mean(X3$SL) ,mean(X3$SW))#c(mean(X3$SW) ,mean(X3$SL))
cat("Mean of setosa is :")
```

```
## Mean of setosa is :
```

```r
mean1
```

```
## [1] 5.006 3.428
```

```r
cat("Mean of versicolor is :")
```

```
## Mean of versicolor is :
```

```r
mean2
```

```
## [1] 5.936 2.770
```

```r
cat("Mean of virginica is :")
```

```
## Mean of virginica is :
```

```r
mean3
```

```
## [1] 6.588 2.974
```

```r
cat(paste("\nCovariance is :"))
```

```
##
## Covariance is :
```

```r
cov1 = cov(subset(X1))
cov1
```

```
##            SL         SW
## SL 0.12424898 0.09921633
## SW 0.09921633 0.14368980
```

```r
cov2 = cov(subset(X2))
cov2
```

```
##            SL         SW
## SL 0.26643265 0.08518367
## SW 0.08518367 0.09846939
```

```
cov3 = cov(subset(X3))
cov3
```

```
##             SL         SW
## SL 0.40434286 0.09376327
## SW 0.09376327 0.10400408
```

```
prior1 = nrow(X1) / nrow(X)
prior2 = nrow(X2) / nrow(X)
prior3 = nrow(X3) / nrow(X)

cat(paste("Prior probability of setosa is",prior1))
```

```
## Prior probability of setosa is 0.333333333333333
```

```
cat(paste("Prior probability of versicolor is",prior2))
```

```
## Prior probability of versicolor is 0.333333333333333
```

```
cat(paste("Prior probability of virginica is",prior3))
```

```
## Prior probability of virginica is 0.333333333333333
```

```
#b)
cat("\nOverall covariance is :")
```

```
##
## Overall covariance is :
```

```
overall_cov = (cov1*nrow(X1) + cov2*nrow(X2) + cov3*nrow(X3))/sum(nrow(X1),nrow(X2),nrow(X3))
overall_cov
```

```
##             SL         SW
## SL 0.26500816 0.09272109
## SW 0.09272109 0.11538776
```

c. Probabilistic model

Assumption

$$\sum_i = \sum, i = 1, ...K$$

$$x|y = C_i, \mu_i, \sum \sim N(\mu_i, \sum)$$

$$y|\pi \sim Multinomial(\pi_1, ...\pi_k)$$

Discriminate function : $\delta_k(x) = x^T\Sigma^{-1}\mu_k + (-1/2)\Sigma^{-1}\mu_k + log(\pi_k)$

$w_i = \Sigma^{-1}\mu_i \; w_{0i} = (-1/2)\Sigma^{-1}\mu_k + log(\pi_k)$

Decision Boundary :

$$w_{1i} * x + w_{0i} = w_{1j} * x + w_{0j}$$

$$(w_{1i} - w_{1j})x + (w_{0i} - w_{0j}) = 0$$

```r
#d)
disc_fun=function(x, mean, s, prior){
  w0 = ((-0.5) * t(mean) %*% solve(s) %*% mean) + log(prior)
  w1 = as.matrix(x) %*% solve(s) %*% mean
  delta = w1 + as.numeric(w0)
  return(delta)
}

res1 = disc_fun(X,mean1,overall_cov,prior1)
res2 = disc_fun(X,mean2,overall_cov,prior2)
res3 = disc_fun(X,mean3,overall_cov,prior3)

result = cbind(res1,res2,res3)

#e)
dec_bouw0 = function(s,mu1,mu2,prior1,prior2){
  w0 = ((-0.5) * t(mu1 + mu2) %*% solve(s) %*% (mu1 - mu2)) + log(prior1/prior2)
  return(w0)
}
dec_bouw1 = function(s,mu1,mu2,prior1,prior2){
  w1 = solve(s) %*% (mu1 - mu2)
  return(w1)
}
# cat("Decision boundary for sesota - versicolor")
# dec_bouw1(overall_cov,mean1,mean2,prior1,prior2)
# dec_bouw0(overall_cov,mean1,mean2,prior1,prior2)
#
# cat("Decision boundary for versicolor - virginica")
# dec_bouw1(overall_cov,mean2,mean3,prior1,prior2)
# dec_bouw0(overall_cov,mean2,mean3,prior1,prior2)
#
# cat("Decision boundary for virginica - sesota")
# dec_bouw1(overall_cov,mean3,mean1,prior1,prior2)
# dec_bouw0(overall_cov,mean3,mean1,prior1,prior2)
```

With $x = \binom{SL}{SW}$, we have the pairwise boundary equations:

for setosa-versicolor:

$$\begin{pmatrix} -7.66 \\ 11.86 \end{pmatrix}^T x - 7.461215 = 0$$

for versicolor-virginica:

$$\begin{pmatrix} 0.29 \\ -2.56 \end{pmatrix}^T x + 15.21 = 0$$

for virginica-setosa:

$$\begin{pmatrix} -12.15 \\ 10.22 \end{pmatrix}^T x - 20.36 = 0$$

3.

LDA using discriminant functions

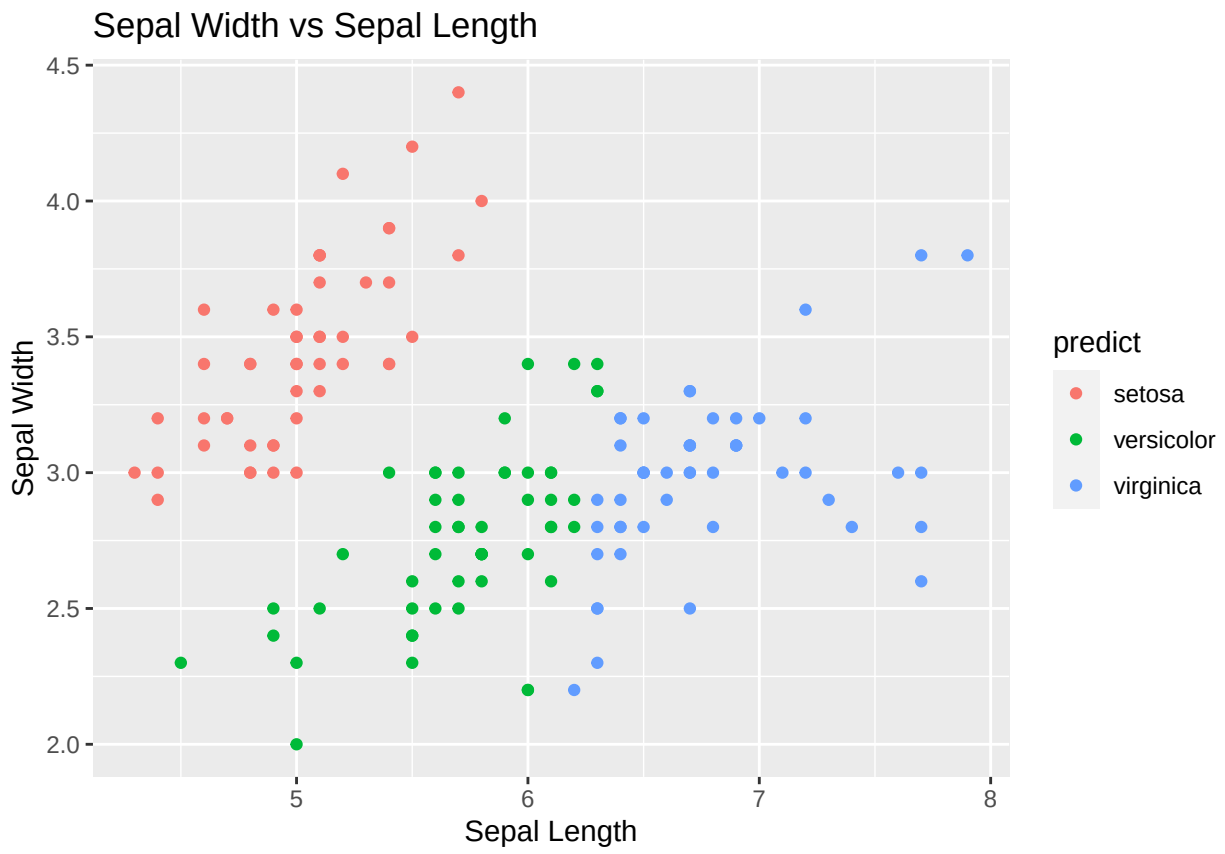
```
for (i in row(result)) {
  predict <- apply(result[,], 1, which.max)
}
df=iris
df = cbind(df,predict)
df$predict[df$predict == "1"] = "setosa"
df$predict[df$predict == "2"] = "versicolor"
df$predict[df$predict == "3"] = "virginica"

table(Actual=(actual<-iris$Species),
       Classified=(classified<-df$predict))
```

```
##             Classified
## Actual       setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

```
comp_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))
```

```
## Test error : 0.2
```

```
ggplot(data = df) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predict)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
```



LDA using lda()

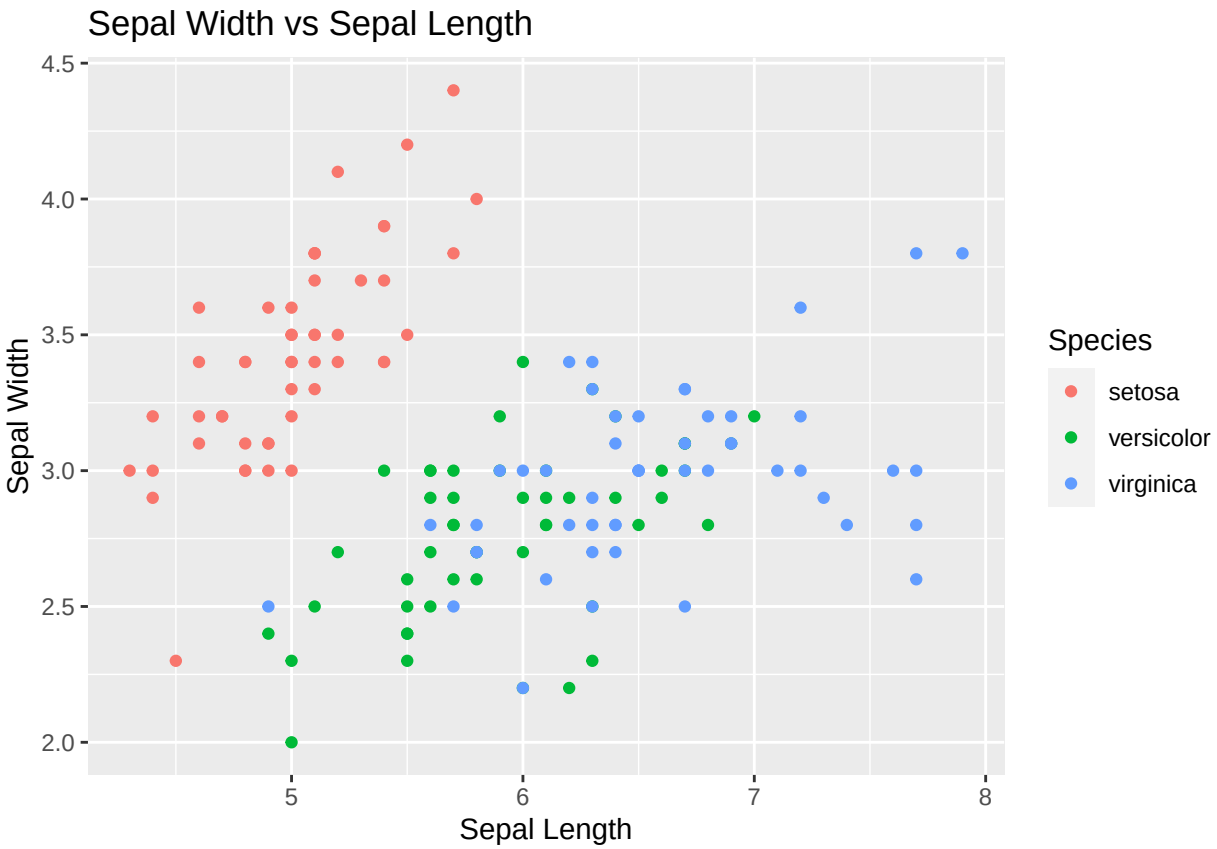
```
model = lda( iris$Species ~ iris$Sepal.Length + iris$Sepal.Width, iris)
predictedData = predict(model, iris[1:2])$class
table(Actual=(actual<-iris$Species),
      Classified=(classified<-predictedData))
```

```
##             Classified
## Actual       setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

```
LDA_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))
```

```
## Test error : 0.2
```

```
ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
```

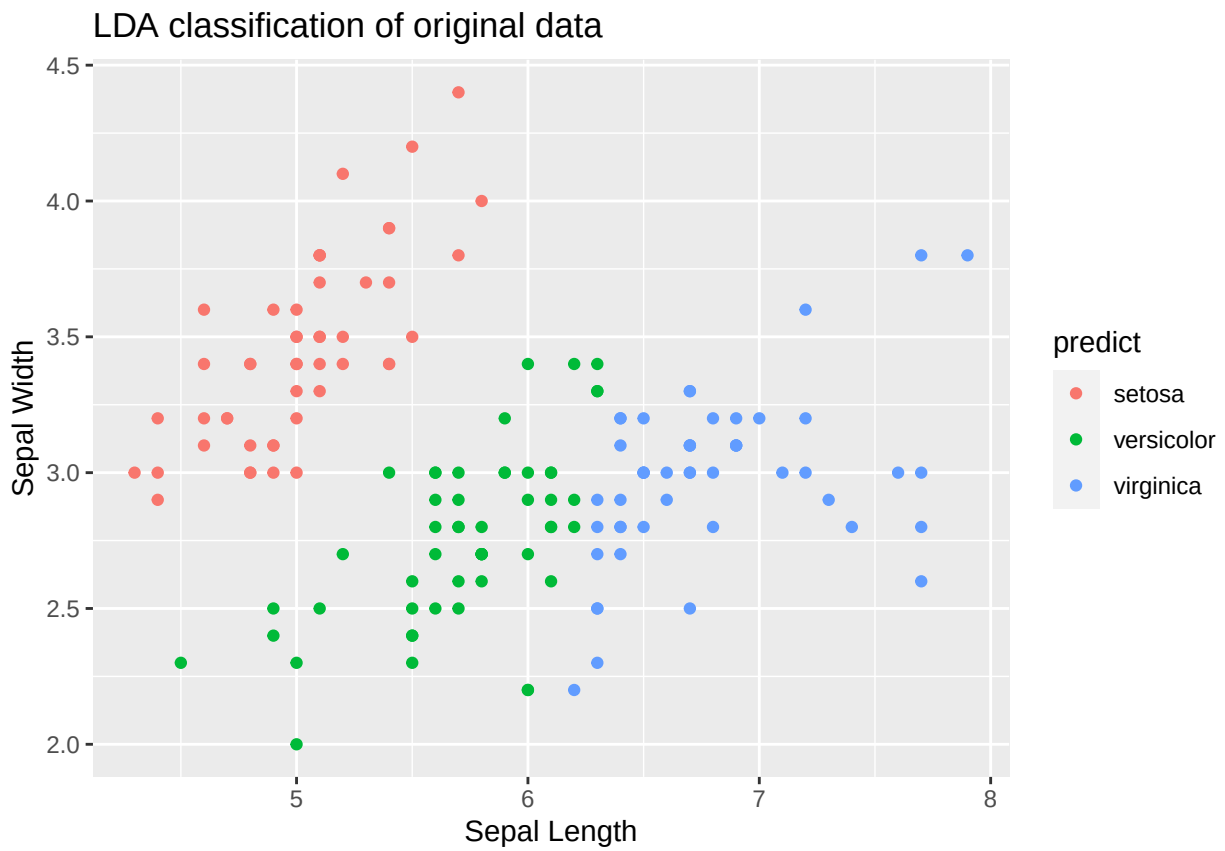


Test error is same for both validation. Results are also identical.

4.

Generate data using multinominal

```
ggplot(data = df) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predict)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("LDA classification of original data")
```



The shape of the distributions of the three species are the same in the generated data set, as a consequence of the equal covariance assumptions. The Virginica and Versicolor datapoints overlap in the same way that they do in the original data set. The classified datapoints do not overlap, however, since the LDA classification divides the plane into areas based on which single class is most likely to appear in it, thus losing the variation of the original data.

5.

```
iris$Species <- relevel(iris$Species, ref = "setosa")
multi = multinom(iris$Species ~ iris$Sepal.Width + iris$Sepal.Length, data=iris)
```

```
## # weights:  12 (6 variable)
## initial  value 164.791843
## iter  10 value 62.715967
## iter  20 value 59.808291
## iter  30 value 55.445984
## iter  40 value 55.375704
## iter  50 value 55.346472
## iter  60 value 55.301707
## iter  70 value 55.253532
## iter  80 value 55.243230
## iter  90 value 55.230241
## iter 100 value 55.212479
## final  value 55.212479
```

```
## stopped after 100 iterations
```

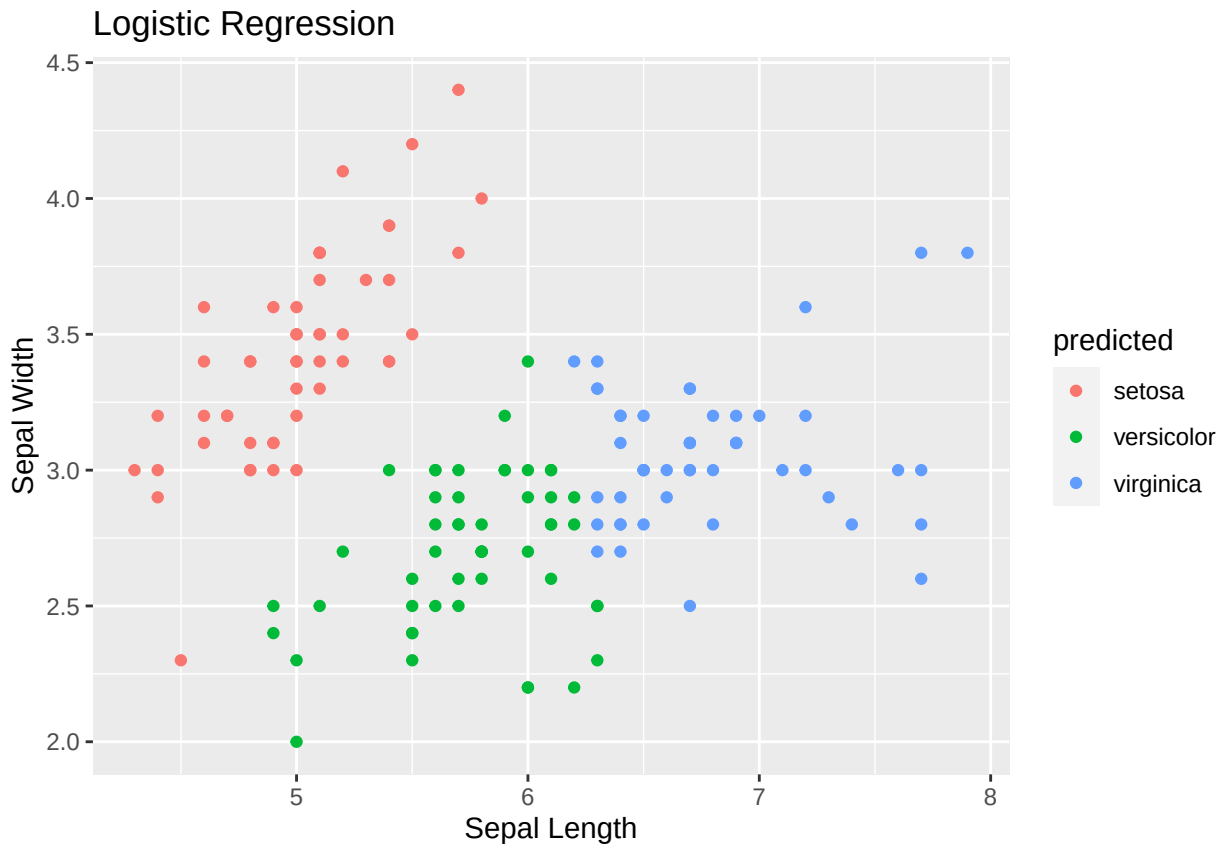
```
iris2=iris[,c(1,2)]
probabilities <- predict(multi, iris[,1:2])
iris2$predicted=probabilities

table(Actual=(actual<-iris$Species), Classified=(classified<-probabilities))
```

```
##             Classified
## Actual       setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         38        12
##   virginica       0         13        37
```

```
log_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))
```

```
## Test error : 0.166666666666667
```

```
ggplot(data = iris2) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predicted)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Logistic Regression")
```



As misclassification rate of logistic regression is less than LDA, hence we can say Logistic regression performance is slightly better than LDA on this data set. The logistic regression model does not make any demands on the relative shapes of the covariance matrices of the classes, and can thus be expected to work better in situations where they differ, such as here.

Assignment 3: Principal components for crime level analysis

1.

```
communities=read.csv("communities.csv")

ViolentCrimesPerPop=communities[["ViolentCrimesPerPop"]]
rawVariables=communities[,names(communities)!="ViolentCrimesPerPop"]#exclude state as well?

variablesScaled=scale(rawVariables)
covMatrix=cov(variablesScaled)

pca1Eigen=eigen(covMatrix)
pca1EigenVectors=pca1Eigen$vectors
pca1EigenValues=pca1Eigen$values

variablesPCA1=variablesScaled%*%pca1EigenVectors

#q3.1

print("The proportions of the variance explained by the first two principal components are")
```

```
## [1] "The proportions of the variance explained by the first two principal components are"
```

```
print(pca1EigenValues[1:2])
```

```
## [1] 25.01699 16.93597
```

```
print("The minimum number of principal components needed to explain 95% of the variance is")
```

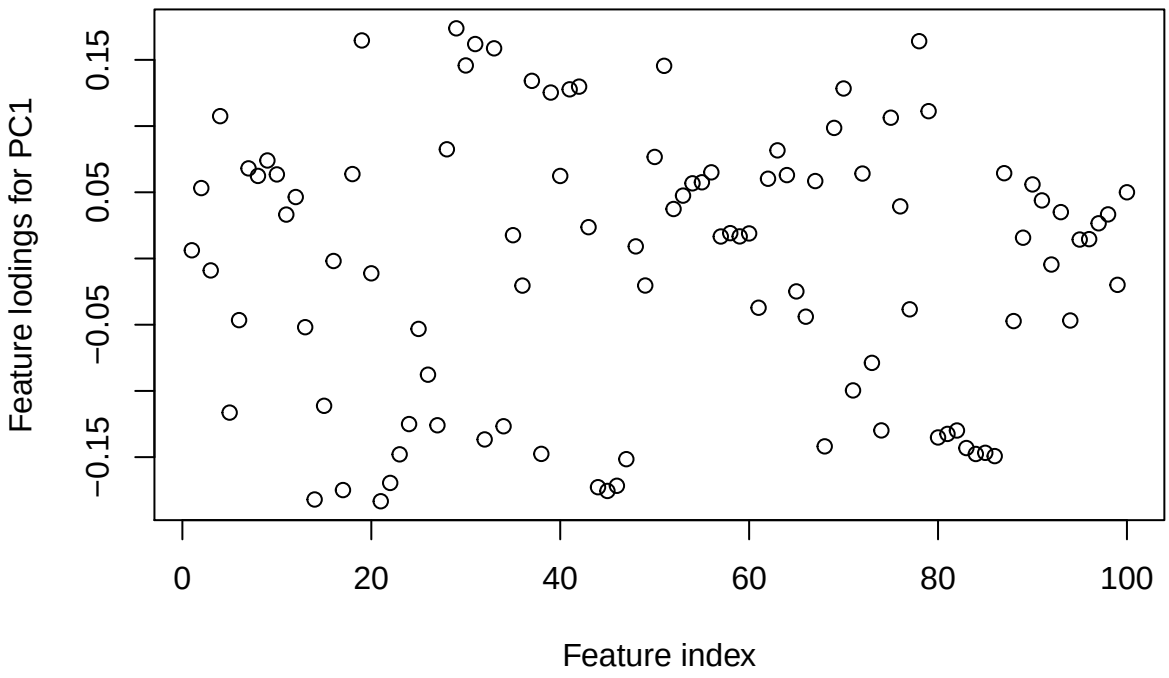
```
## [1] "The minimum number of principal components needed to explain 95% of the variance is"
```

```
print(which(cumsum(pca1EigenValues)>95)[1])
```

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

2.

```
pca2=princomp(variablesScaled)
#plot(pca2$scores[,1])

#Do many features have a notable contribution to this component?
plot(pca2[["loadings"]][,1],ylab="Feature lodings for PC1",xlab = "Feature index")
```




```
print("The number of features whose loadings surpass 0.1 in absolute terms is")
```

```
## [1] "The number of features whose loadings surpass 0.1 in absolute terms is"
```

```
sum(abs(pca2[["loadings"]][,1])>0.1)
```

```
## [1] 40
```

```
#Report which 5 features contribute mostly (by the absolute value) to the first principal component.
#Comment whether these features have anything in common and
#whether they may have a logical relationship to the crime level
a=sort(abs(pca2[["loadings"]][,1]),decreasing=TRUE)
print(a[1:5])
```

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

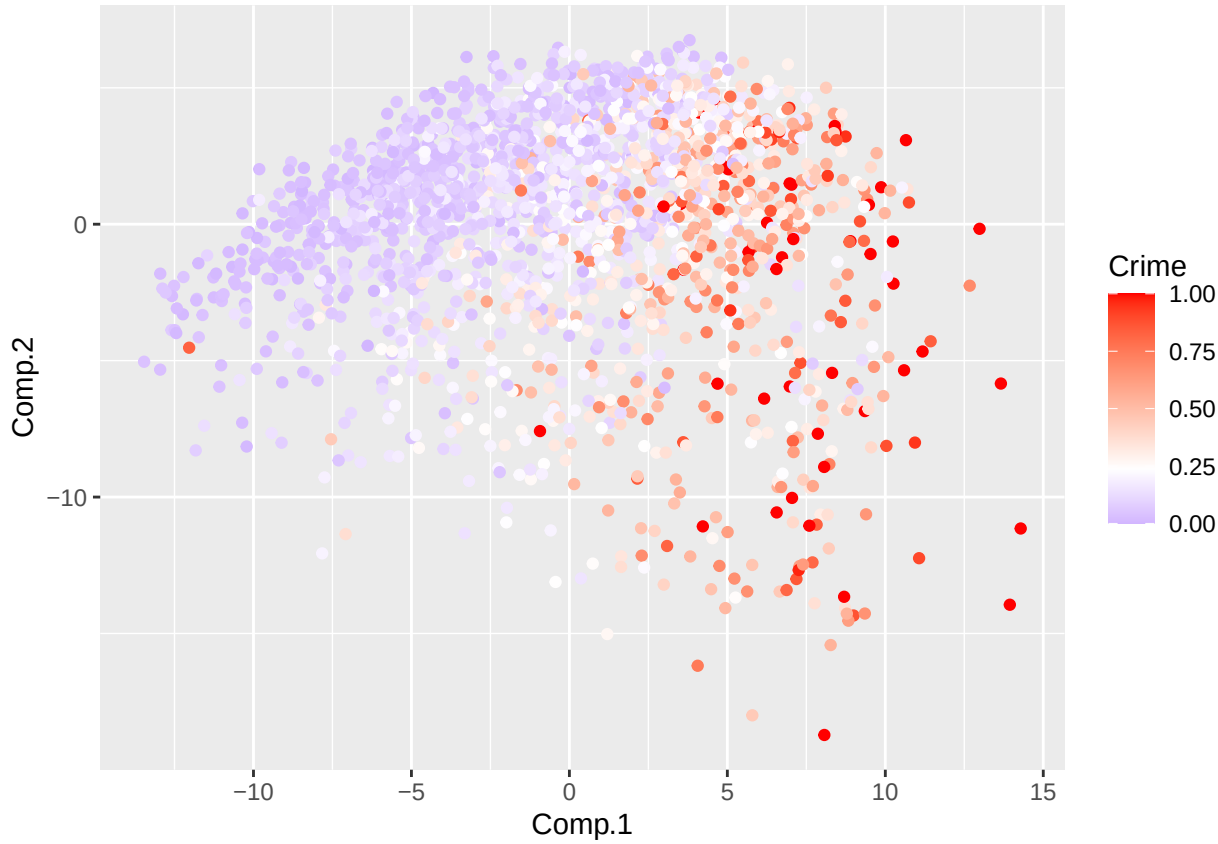
The 5 features which contribute the most (by absolute value) to the first principal component are:

medFamInc: median family income medIncome: median household income PctKids2Par: percentage of kids in family housing with two parents pctWInvInc: Percentage of households with investment / rent income in 1989 PctPopUnderPov: percentage of people under the poverty level

The relationship between these variables and crime levels are fairly uncontroversial; four of them are connected to family income and wealth, and in the USA "percentage of kids in family housing with two parents"

tends to work as an indirect measure of both social capital (couples with a large and strong mutual social network tend not to divorce/separate as easily) and of parental incarceration (the USA has a very high incarceration rate). In other words, all five of these variables measure socio-economic immiseration, and can therefore be expected to be positively correlated with crime.

```
library(ggplot2)

pca2Scores=pca2$scores[,1:2]
pca2ScoresWithCrime=data.frame(pca2$scores[,1:2],Crime=ViolentCrimesPerPop)

colorPlot<-ggplot(pca2ScoresWithCrime, aes(x=Comp.1, y=Comp.2, color=Crime)) + geom_point()

mid<-mean(pca2ScoresWithCrime$Crime)
colorPlot+scale_color_gradient2(midpoint=mid, low="blue", mid="white", high="red", space ="Lab" )
```

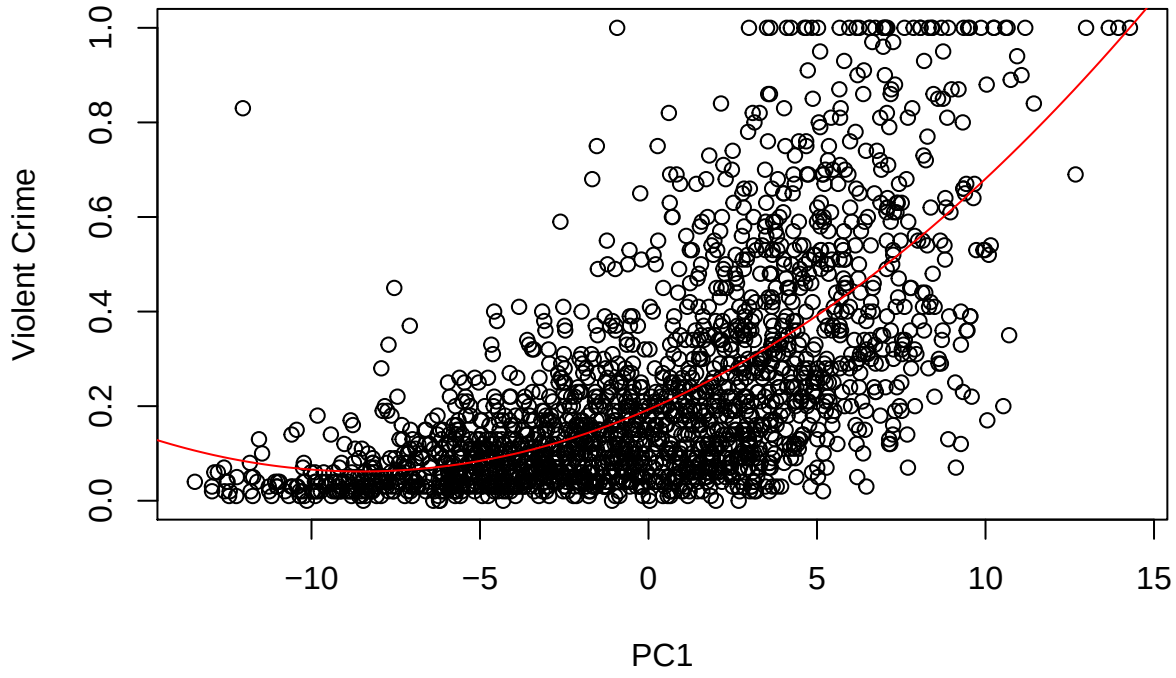


This plot shows that the crime variable seems to be strongly positively correlated with the first principal component, and somewhat negatively correlated to the second principal component.

3.

```
polynomialModel=lm(Crime~poly(Comp.1,2,raw=TRUE),data=pca2ScoresWithCrime)

x0=rep(1,301)
x1=seq(-15,15,0.1)
```

```
x2=x1^2

xMatrix=data.matrix(data.frame(x0,x1,x2))
y=xMatrix%*%polynomialModel$coefficients


plot(pca2ScoresWithCrime$Comp.1,pca2ScoresWithCrime$Crime,
     xlab = "PC1", ylab = "Violent Crime")
lines(x1,y,col="red")
```

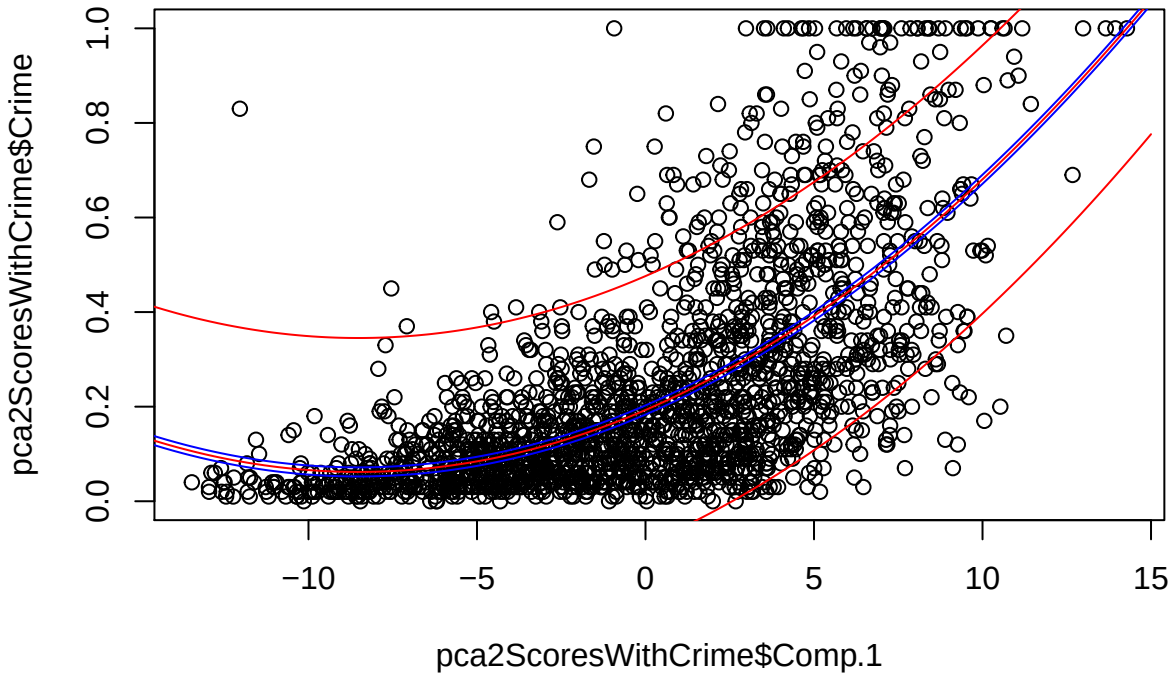


The broad strokes of the feature seems to be well-explained by this polynomial model, but there is still plenty of individual variation around the predicted line.

4.

```
K=10000
N=nrow(pca2ScoresWithCrime)
polynomialModelSummary=summary(polynomialModel)

coeffs=matrix(data=NA,nrow=K,ncol=5)

for(i in 1:K){
  res=rnorm(N,0,sd=polynomialModelSummary$sigma)
  xVals=runif(N,min=-15,max=15)
```

```
  mid=predict(polynomialModel, interval = "prediction",newdata=data.frame(Comp.1=xVals))
  newData=data.frame(Comp.1=xVals,Crime=mid[,1]+res)


  newModel=lm(Crime~poly(Comp.1,2,raw=TRUE),data=newData)

  residualsSorted=sort(newModel$res)

  #for 5th and 95th percentile ranges

  cutoff=round(N/20)
  lowCut=newModel$coefficients[1]+residualsSorted[cutoff]
  highCut=newModel$coefficients[1]+residualsSorted[N-cutoff]


  coeffs[i,]=c(newModel$coefficients[1:3],lowCut,highCut)
}

estimates=colSums(coeffs)/nrow(coeffs)
interceptEstimatesSorted=sort(coeffs[,1])
  cutoff=round(K/20)
  lowConf=interceptEstimatesSorted[cutoff]
  highConf=interceptEstimatesSorted[K-cutoff]



x0=rep(1,301)
x1=seq(-15,15,0.1)
x2=x1^2

xMatrix=data.matrix(data.frame(x0,x1,x2))
yMid=xMatrix%*%estimates[1:3]
yLowPred=xMatrix%*%c(estimates[4],estimates[2:3])
yHighPred=xMatrix%*%c(estimates[5],estimates[2:3])
yLowConf=xMatrix%*%c(lowConf,estimates[2:3])
yHighConf=xMatrix%*%c(highConf,estimates[2:3])

plot(pca2ScoresWithCrime$Comp.1,pca2ScoresWithCrime$Crime)
lines(x1,yMid,col="red")
lines(x1,yLowPred,col="red")
lines(x1,yHighPred,col="red")
lines(x1,yLowConf,col="blue")
lines(x1,yHighConf,col="blue")
```



While the confidence bands are quite narrow, the prediction bands are not, Most of the uncertainty in the prediction interval here stems from the randomness in the model itself, and not so much from uncertainty about the model's parameters.

Apendix

```
knitr::opts_chunk$set(echo = TRUE)
data(iris)
library(ggplot2)
library(MASS)
library(mvtnorm)
library(nnet)
ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
#a)
data = iris
X = data.frame(SL=data$Sepal.Length,SW=data$Sepal.Width)
Y = data$Species
X1 = X[Y=="setosa",]
X2 = X[Y=="versicolor",]
X3 = X[Y=="virginica",]

mean1 = c(mean(X1$SL) ,mean(X1$SW))#c(mean(X1$SW) ,mean(X1$SL))
```

```r
mean2 = c(mean(X2$SL) ,mean(X2$SW))#c(mean(X2$SW) ,mean(X2$SL))
mean3 = c(mean(X3$SL) ,mean(X3$SW))#c(mean(X3$SW) ,mean(X3$SL))
cat("Mean of setosa is :")
mean1
cat("Mean of versicolor is :")
mean2
cat("Mean of virginica is :")
mean3

cat(paste("\nCovariance is :"))
cov1 = cov(subset(X1))
cov1
cov2 = cov(subset(X2))
cov2
cov3 = cov(subset(X3))
cov3

prior1 = nrow(X1) / nrow(X)
prior2 = nrow(X2) / nrow(X)
prior3 = nrow(X3) / nrow(X)

cat(paste("Prior probability of setosa is",prior1))
cat(paste("Prior probability of versicolor is",prior2))
cat(paste("Prior probability of virginica is",prior3))

#b)
cat("\nOverall covariance is :")
overall_cov = (cov1*nrow(X1) + cov2*nrow(X2) + cov3*nrow(X3))/sum(nrow(X1),nrow(X2),nrow(X3))
overall_cov
#d)
disc_fun=function(x, mean, s, prior){
  w0 = ((-0.5) * t(mean) %*% solve(s) %*% mean) + log(prior)
  w1 = as.matrix(x) %*% solve(s) %*% mean
  delta = w1 + as.numeric(w0)
  return(delta)
}

res1 = disc_fun(X,mean1,overall_cov,prior1)
res2 = disc_fun(X,mean2,overall_cov,prior2)
res3 = disc_fun(X,mean3,overall_cov,prior3)

result = cbind(res1,res2,res3)

#e)
dec_bouw0 = function(s,mu1,mu2,prior1,prior2){
  w0 = ((-0.5) * t(mu1 + mu2) %*% solve(s) %*% (mu1 - mu2)) + log(prior1/prior2)
  return(w0)
}
dec_bouw1 = function(s,mu1,mu2,prior1,prior2){
  w1 = solve(s) %*% (mu1 - mu2)
  return(w1)
}
# cat("Decision boundary for sesota - versicolor")
```

```
# dec_bouw1(overall_cov,mean1,mean2,prior1,prior2)
# dec_bouw0(overall_cov,mean1,mean2,prior1,prior2)
#
# cat("Decision boundary for versicolor - virginica")
# dec_bouw1(overall_cov,mean2,mean3,prior1,prior2)
# dec_bouw0(overall_cov,mean2,mean3,prior1,prior2)
#
# cat("Decision boundary for virginica - sesota")
# dec_bouw1(overall_cov,mean3,mean1,prior1,prior2)
# dec_bouw0(overall_cov,mean3,mean1,prior1,prior2)
for (i in row(result)) {
  predict <- apply(result[,], 1, which.max)
}
df=iris
df = cbind(df,predict)
df$predict[df$predict == "1"] = "setosa"
df$predict[df$predict == "2"] = "versicolor"
df$predict[df$predict == "3"] = "virginica"

table(Actual=(actual<-iris$Species),
       Classified=(classified<-df$predict))
comp_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))

ggplot(data = df) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predict)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
model = lda( iris$Species ~ iris$Sepal.Length + iris$Sepal.Width, iris)
predictedData = predict(model, iris[1:2])$class
table(Actual=(actual<-iris$Species),
       Classified=(classified<-predictedData))
LDA_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))

ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
set.seed(12345)
gen1 <- rmvnorm(50, mean = mean1, sigma = cov1)
gen2 <- rmvnorm(50, mean = mean2, sigma = cov2)
gen3 <- rmvnorm(50, mean = mean3, sigma = cov3)
#Y_samples <- c(rep("setosa", 50), rep("versicolor", 50), rep("virginica", 50))
dataset <- as.data.frame(rbind(gen1, gen2, gen3))
species <- c(rep("setosa", 50), rep("versicolor", 50), rep("virginica", 50))
dataset <- as.data.frame(cbind(dataset,species))
colnames(dataset) <- c("Sepal.Length", "Sepal.Width", "Species")

ggplot(data = dataset) +
  geom_point(aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Generated new data")

ggplot(data = df) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predict)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Sepal Width vs Sepal Length")
ggplot(data = df) +
```

```
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predict)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("LDA classification of original data")
iris$Species <- relevel(iris$Species, ref = "setosa")
multi = multinom(iris$Species ~ iris$Sepal.Width + iris$Sepal.Length, data=iris)

iris2=iris[,c(1,2)]
probabilities <- predict(multi, iris[,1:2])
iris2$predicted=probabilities

table(Actual=(actual<-iris$Species), Classified=(classified<-probabilities))

log_error = cat(paste("Test error :", mean(is_miss <- actual != classified)))


ggplot(data = iris2) +
  geom_point(aes(x = Sepal.Length, y = Sepal.Width, color = predicted)) +
  xlab("Sepal Length") + ylab("Sepal Width") + ggtitle("Logistic Regression")
communities=read.csv("communities.csv")

ViolentCrimesPerPop=communities[["ViolentCrimesPerPop"]]
rawVariables=communities[,names(communities)!="ViolentCrimesPerPop"]#exclude state as well?

variablesScaled=scale(rawVariables)
covMatrix=cov(variablesScaled)

pca1Eigen=eigen(covMatrix)
pca1EigenVectors=pca1Eigen$vectors
pca1EigenValues=pca1Eigen$values

variablesPCA1=variablesScaled%*%pca1EigenVectors

#q3.1

print("The proportions of the variance explained by the first two principal components are")
print(pca1EigenValues[1:2])




print("The minimum number of principal components needed to explain 95% of the variance is")
print(which(cumsum(pca1EigenValues)>95)[1])

pca2=princomp(variablesScaled)
#plot(pca2$scores[,1])

#Do many features have a notable contribution to this component?
plot(pca2[["loadings"]][,1],ylab="Feature lodings for PC1",xlab = "Feature index")

print("The number of features whose loadings surpass 0.1 in absolute terms is")

sum(abs(pca2[["loadings"]][,1])>0.1)
#Report which 5 features contribute mostly (by the absolute value) to the first principal component.
#Comment whether these features have anything in common and
```

```r
#whether they may have a logical relationship to the crime level
a=sort(abs(pca2[["loadings"]][,1]),decreasing=TRUE)
print(a[1:5])
library(ggplot2)

pca2Scores=pca2$scores[,1:2]
pca2ScoresWithCrime=data.frame(pca2$scores[,1:2],Crime=ViolentCrimesPerPop)

colorPlot<-ggplot(pca2ScoresWithCrime, aes(x=Comp.1, y=Comp.2, color=Crime)) + geom_point()

mid<-mean(pca2ScoresWithCrime$Crime)
colorPlot+scale_color_gradient2(midpoint=mid, low="blue", mid="white", high="red", space ="Lab" )

polynomialModel=lm(Crime~poly(Comp.1,2,raw=TRUE),data=pca2ScoresWithCrime)

x0=rep(1,301)
x1=seq(-15,15,0.1)
x2=x1^2

xMatrix=data.matrix(data.frame(x0,x1,x2))
y=xMatrix%*%polynomialModel$coefficients


plot(pca2ScoresWithCrime$Comp.1,pca2ScoresWithCrime$Crime,
     xlab = "PC1", ylab = "Violent Crime")
lines(x1,y,col="red")


K=10000
N=nrow(pca2ScoresWithCrime)
polynomialModelSummary=summary(polynomialModel)

coeffs=matrix(data=NA,nrow=K,ncol=5)

for(i in 1:K){
  res=rnorm(N,0,sd=polynomialModelSummary$sigma)
  xVals=runif(N,min=-15,max=15)
  mid=predict(polynomialModel, interval = "prediction",newdata=data.frame(Comp.1=xVals))
  newData=data.frame(Comp.1=xVals,Crime=mid[,1]+res)


  newModel=lm(Crime~poly(Comp.1,2,raw=TRUE),data=newData)

  residualsSorted=sort(newModel$res)

  #for 5th and 95th percentile ranges

  cutoff=round(N/20)
  lowCut=newModel$coefficients[1]+residualsSorted[cutoff]
  highCut=newModel$coefficients[1]+residualsSorted[N-cutoff]


  coeffs[i,]=c(newModel$coefficients[1:3],lowCut,highCut)
```

```
}

estimates=colSums(coeffs)/nrow(coeffs)
interceptEstimatesSorted=sort(coeffs[,1])
  cutoff=round(K/20)
  lowConf=interceptEstimatesSorted[cutoff]
  highConf=interceptEstimatesSorted[K-cutoff]



x0=rep(1,301)
x1=seq(-15,15,0.1)
x2=x1^2

xMatrix=data.matrix(data.frame(x0,x1,x2))
yMid=xMatrix%*%estimates[1:3]
yLowPred=xMatrix%*%c(estimates[4],estimates[2:3])
yHighPred=xMatrix%*%c(estimates[5],estimates[2:3])
yLowConf=xMatrix%*%c(lowConf,estimates[2:3])
yHighConf=xMatrix%*%c(highConf,estimates[2:3])

plot(pca2ScoresWithCrime$Comp.1,pca2ScoresWithCrime$Crime)
lines(x1,yMid,col="red")
lines(x1,yLowPred,col="red")
lines(x1,yHighPred,col="red")
lines(x1,yLowConf,col="blue")
lines(x1,yHighConf,col="blue")
```