Machine Learning Block 1 Lab 2

Siddharth Saminathan, Salvador Marti Roman, Nicolas Taba

06/12/2020

Statement of contribution

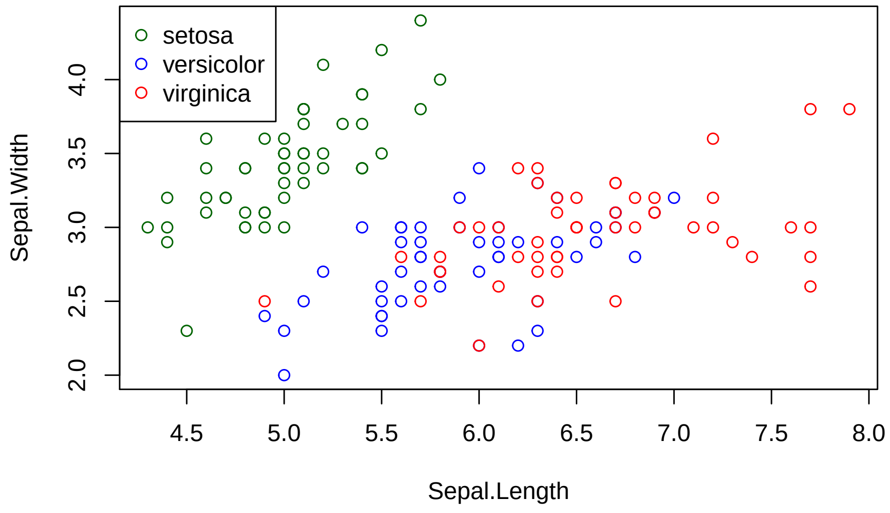
Salvador Marti Roman solved assignment 1, Siddharth Saminathan solved assignment 2 and Nicolas Taba solved assignment 3. All members of the group discussed solutions and problems during all the assignment period.

```
## Warning: package 'mvtnorm' was built under R version 4.0.3
```

Assignment 1. LDA and logistic regression

R data file "iris" (present in the default R installation) shows the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.The species are Iris setosa, versicolor, and virginica.

1. Make a scatterplot of Sepal Width versus Sepal Length where observations are colored by Species. Do you think that this data is easy to classify by linear discriminant analysis? Motivate your answer.



The versicolor and virginica classes will be difficult to classify as they are not linearly separable given the data taken into consideration.

2. Use basic R functions only to implement Linear Discriminant Analysis between the three species based on variables Sepal Length and Sepal Width:

a. Compute mean, covariance matrices (use cov()) and prior probabilities per class and report them

```
## [1] "Sample means"
```

```
##              setosa versicolor virginica
## Sepal.Length  5.006      5.936     6.588
## Sepal.Width   3.428      2.770     2.974
```

```
## [1] "Covariance"
```

```
## $setosa
##              Sepal.Length Sepal.Width
## Sepal.Length   0.12424898  0.09921633
## Sepal.Width    0.09921633  0.14368980
##
## $versicolor
##              Sepal.Length Sepal.Width
## Sepal.Length   0.26643265  0.08518367
## Sepal.Width    0.08518367  0.09846939
##
## $virginica
##              Sepal.Length Sepal.Width
## Sepal.Length   0.40434286  0.09376327
## Sepal.Width    0.09376327  0.10400408
```

```
## [1] "Priors"
```

```
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
```

b. Compute overall (pooled) covariance matrix and report it

The pooled covariance is a weighted average of the sample covariances for each group. $Prior * Covariance$ should work.

```
pooled_cov = function(covs, priors){
  pooled_covs = covs[[1]] * priors[[1]]
  for(class in 2:length(covs)){
    pooled_covs = pooled_covs + covs[[class]]*priors[[class]]
  }
  return(pooled_covs)
}
sample_pooled_cov = pooled_cov(sample_covs, sample_priors)
sample_pooled_cov
```

```
##              Sepal.Length Sepal.Width
## Sepal.Length   0.26500816  0.09272109
## Sepal.Width    0.09272109  0.11538776
```

c. Report the probabilistic model for the LDA

$$x \mid y = C_i, \mu_i, \Sigma \sim N(\mu_i, \Sigma)$$

$$y \mid \pi \sim Multinomial(\pi_1, ..., \pi_K)$$

d. Compute discriminant functions for each class

Assuming

$$\Sigma_i = \Sigma$$

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^- 1 \mu_k + \log \pi_k$$

```
##        setosa versicolor virginica
## [1,] 194.1401   196.8508  224.1077
## [2,] 181.6706   188.7776  215.6675
## [3,] 183.3493   186.5537  212.8730
## [4,] 180.1463   183.7705  209.8627
## [5,] 194.9795   195.7388  222.7105
## [6,] 205.7704   206.0359  233.9452
```

e. Compute equations of decision boundaries between classes and report them

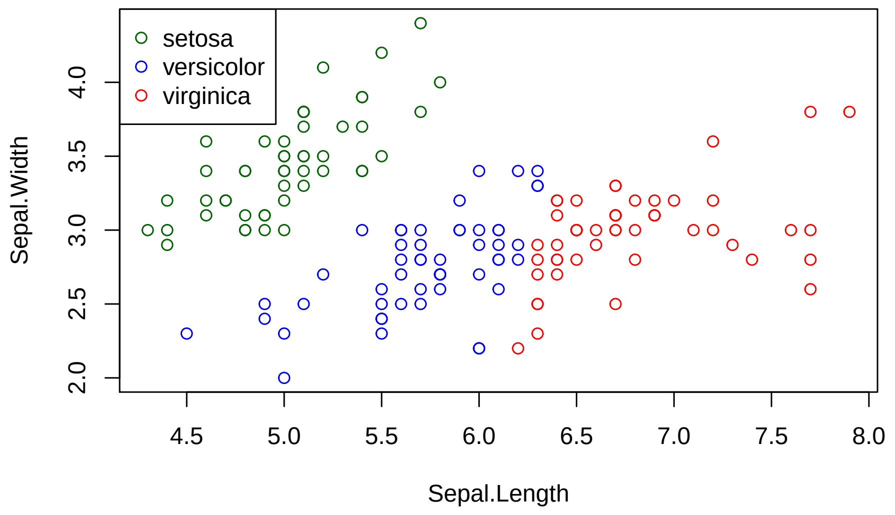
```
##                  w0i       w1        w2
## setosa     -65.32281 11.81827 20.211827
## virginica  -85.68398 22.03772  8.065317
## versicolor -70.47563 19.47567  8.356129
```

Do estimated covariance matrices seem to fulfill LDA assumptions?

They do not, each class covariance is different in reality.

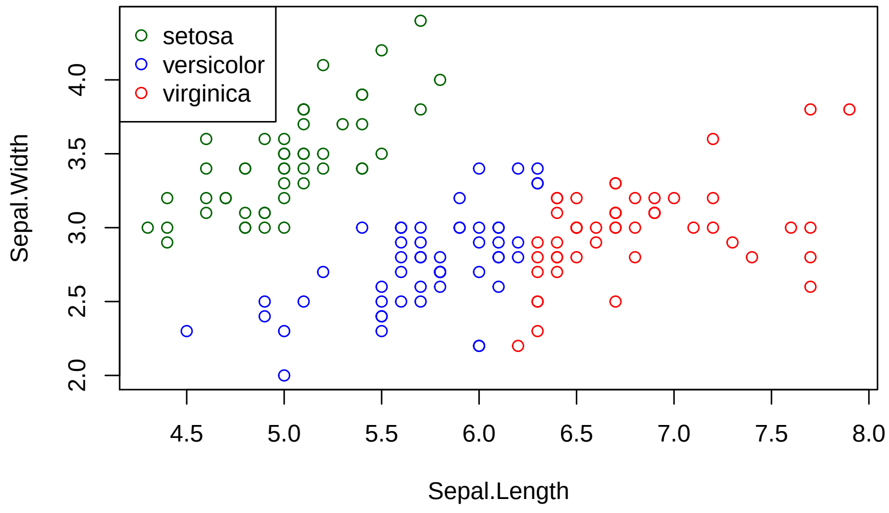
3. Use discriminant functions from step 2 to predict the species from the original data and make a scatterplot of Sepal Length versus Sepal Width in which color shows the predicted Species. Estimate the misclassification rate of the prediction. Comment on the quality of classification. Afterwards, perform the LDA analysis with lda() function and investigate whether you obtain the same test error by using this package. Should it be same?

Implemented LDA



```
## [1] "Misclassification rate 0.2"
```

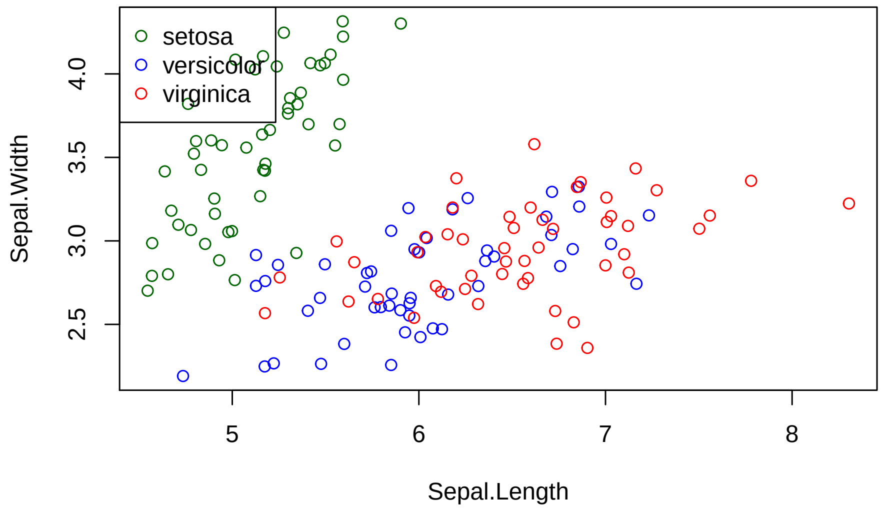
MASS package LDA



```
## [1] "Misclassification rate 0.2"
```

The results obtained are identical. Most of the missclasification errors come from being unable to separate the versicolor and virginica species. Since we are using a similar method it stands to reason the results would be identical for the same data given how it is optimised.

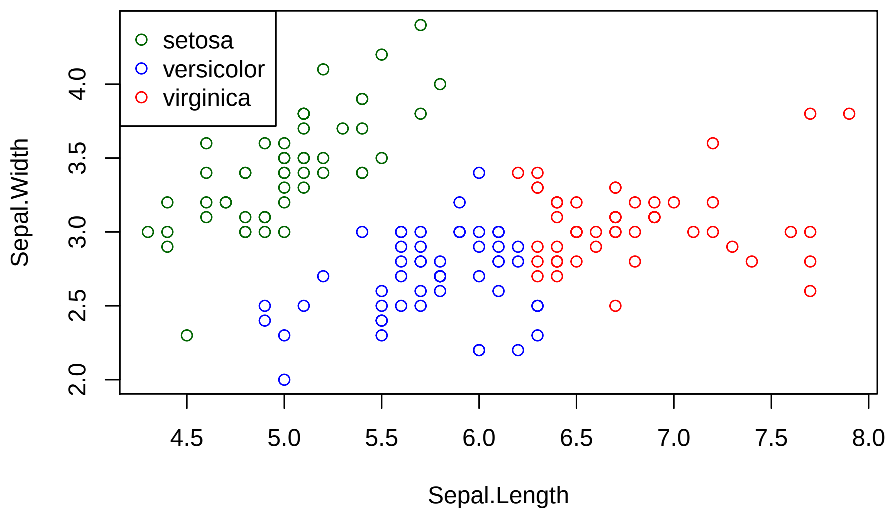
4. Use Models reported in 2c to generate new data of this kind with the same total number of cases as in the original data (hint: use sample() and rmvnorm() from package mvtnorm). Make a scatterplot of the same kind as in step 1 but for the new data and compare it with the plots for the original and the predicted data. Conclusions?



The generated data has an overall similar trend but not identical. While the sepal widths and lengths seems to have discrete levels in the true iris dataset the generated one does not as normal distributions are continuous.

5. Make a similar kind of classification by logistic regression (use function multinom() from nnet package), plot the classified data and compute the misclassification error. Compare these results with the LDA results.

```
## # 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
```



```
## [1] "Misclassification rate 0.166666666666667"
```

The missclassification rate in this function is less than with LDA. The improvements can be mostly seen in the decision boundry between versicolor and virginica where this method includes a few more points than LDA.

Assignment 2: Decision trees and Naïve Bayes from bank marketing

1

Importing data :

```
## Warning: package 'tree' was built under R version 4.0.3
## 'data.frame':    18084 obs. of  16 variables:
##  $ age      : int  53 46 41 43 47 35 60 42 53 30 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 10 8 3 10 11 10 4 8 10 5 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 1 2 3 2 3 1 2 2 1 2 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 2 2 2 2 1 2 1 3 4 3 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  305 327 3216 1937 289 127 1043 12223 629 8623 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 1 2 1 2 1 1 1 2 2 1 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 2 2 1 1 1 1 1 1 1 1 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 3 1 3 3 1 1 1 1 1 1 ...
##  $ day      : int  17 20 9 5 5 20 18 19 8 8 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 7 1 7 9 4 10 2 10 9 1 ...
##  $ campaign : int  1 2 3 1 1 2 4 1 1 3 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
```

2

Fitting Decision trees and computing errors

```
##      error1    error2     error3
## 1 0.1048441 0.1048441 0.09361867
```

```
##   error_val_1 error_val_2 error_val_3
## 1   0.1092679   0.1092679   0.1118484
```

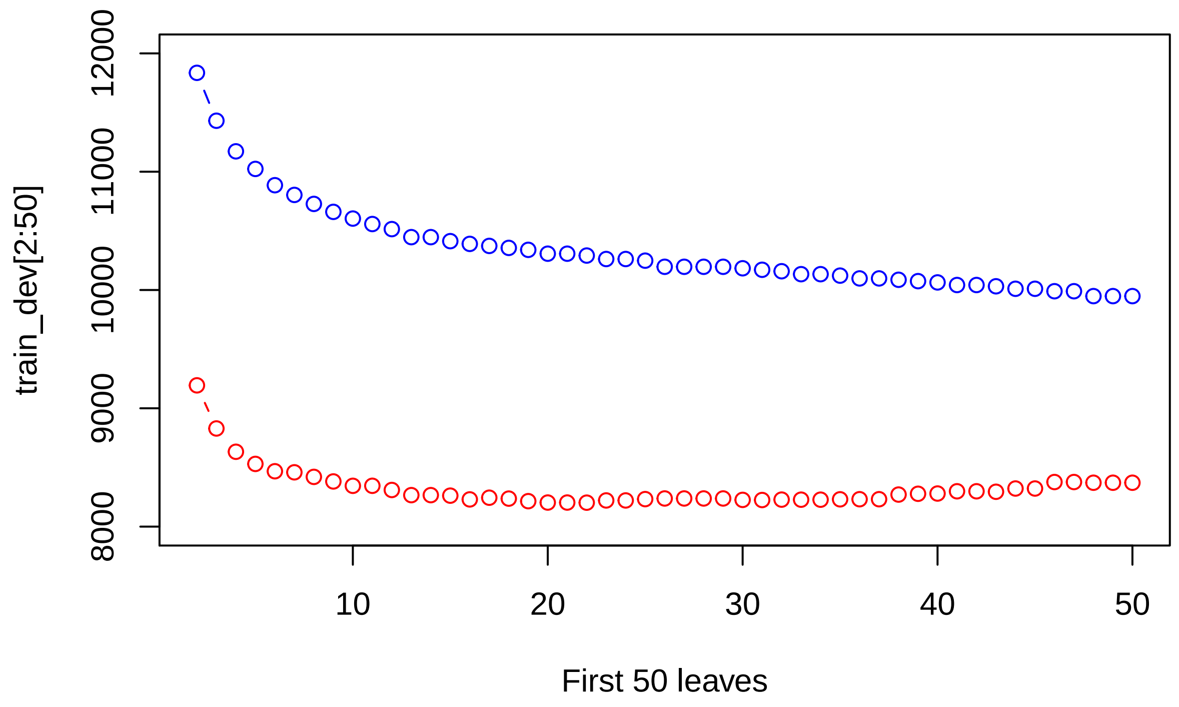
My suggestion would be to use the third model with small deviance as it grows more trees and takes into account more features for better classification.Eventhough it has a slightly higher validation error, This model uses more features than the first two.

Changing the node size didnt affect the trees much. However, changing the deviance increased the number of features to be used thus increasing the overall tree depth. Ensemble methods generally give a better performance for deeper trees. The above results shows us that our model's performence grows as the depth of trees increases which is expected.

3

Finding optimal Tree Depth

Studying trees upto first 50 Leaves



Optimal Tree Depth

```
## [1] 22
```

Important Features used

```
##  [1] "poutcome"  "month"     "contact"   "marital"   "day"       "pdays"
##  [7] "age"       "job"       "balance"   "housing"   "education"
```

Predicting test data with optimal model and finding errors

```
##
## pred_optimal    no   yes
##          no  11872  1371
##          yes   107   214
```

```
## [1] 0.1089649
```

The test error is lower that validation error,Hence the model has good predictive power.

4

Fitting a decision tree using a loss matrix

```
## [1] 0.140519
```

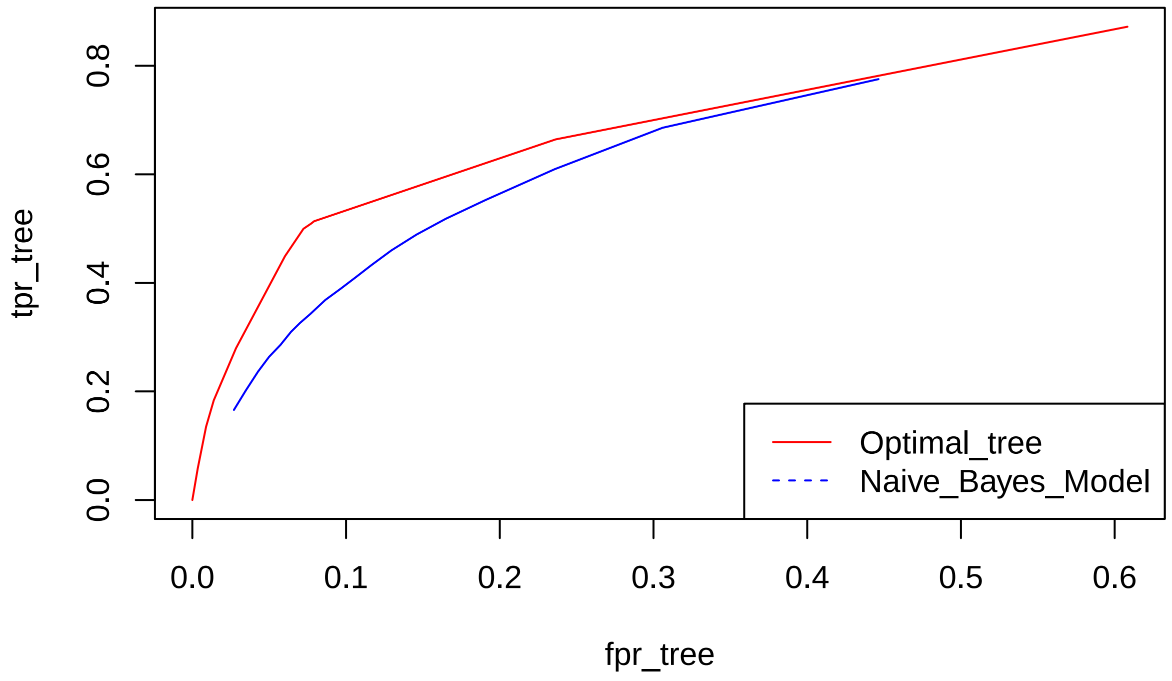
```
##
## pred_loss    no   yes
##       no  10880   807
##       yes  1099   778
```

Compared to 3 the error has increased. This is because of the loss matrix. The misclassification for "yes" is five times more now. Hence there is a increase in the misclassification error.

5

Comparing Optimal_model and Naive Bayes using ROC curves

```
## Warning: package 'e1071' was built under R version 4.0.3
```



From the above plot we can conclude that the optimal model from the decision tree has bigger area under the curve. Hence the optimal tree Model performs better than the naive bayes model.

Assignment 3: Principal components for crime level analysis

1. Proportion of data explained by the principal components

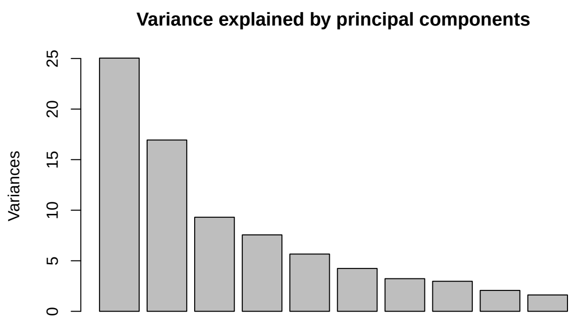
Scale all variables except of ViolentCrimesPerPop and implement PCA by using function eigen(). Report how many features are needed to obtain at least 95% of variance in the data. What is the proportion of variation explained by each of the first two principal components?

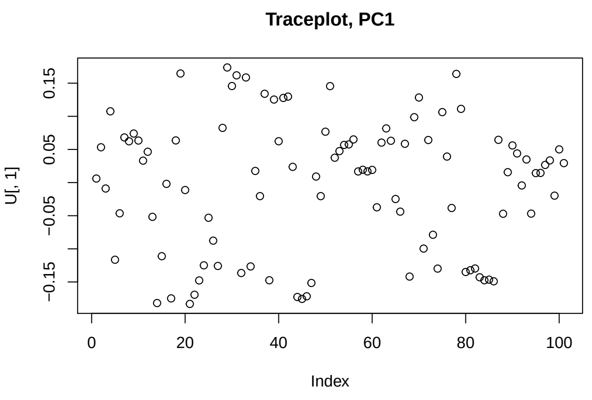
```
## 35 features are needed to explain 95% of the variance in the data
```

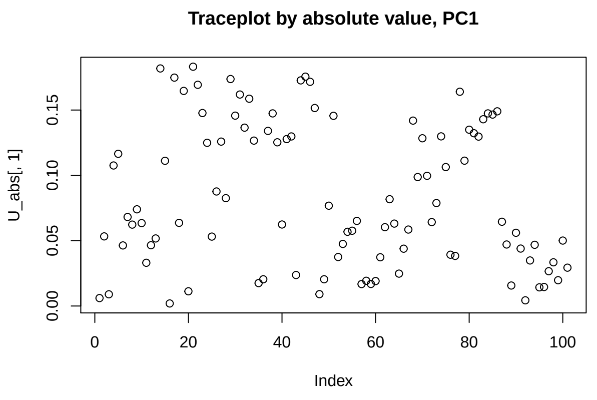
```
## The first two principal components account for  41.95749 % of the variance in the data.
```

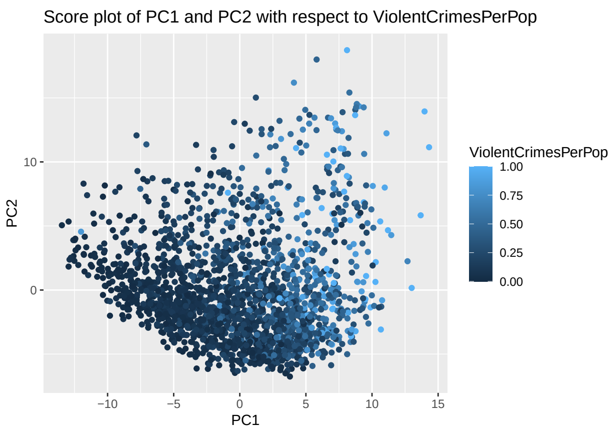
2. Use princomp function to analyze the first 2 principal components

```
##   [1] "25.025" "16.931" "9.298"  "7.554"  "5.660"  "4.236"  "3.226"  "2.969"
##   [9] "2.067"  "1.617"  "1.572"  "1.470"  "1.414"  "1.030"  "0.931"  "0.890"
##  [17] "0.748"  "0.705"  "0.649"  "0.638"  "0.624"  "0.568"  "0.542"  "0.519"
## [25] "0.504" "0.480" "0.466" "0.451" "0.431" "0.386" "0.365" "0.351"
## [33] "0.337" "0.311" "0.287" "0.259" "0.256" "0.245" "0.240" "0.222"
## [41] "0.211" "0.205" "0.200" "0.190" "0.183" "0.165" "0.160" "0.142"
## [49] "0.138" "0.126" "0.112" "0.107" "0.104" "0.100" "0.090" "0.081"
## [57] "0.077" "0.073" "0.069" "0.066" "0.064" "0.062" "0.058" "0.051"
## [65] "0.049" "0.046" "0.044" "0.042" "0.040" "0.038" "0.035" "0.034"
## [73] "0.032" "0.031" "0.028" "0.026" "0.024" "0.022" "0.021" "0.020"
## [81] "0.018" "0.018" "0.016" "0.015" "0.014" "0.013" "0.011" "0.009"
## [89] "0.008" "0.006" "0.006" "0.005" "0.004" "0.004" "0.003" "0.002"
## [97] "0.002" "0.001" "0.001" "0.001" "0.001"
```









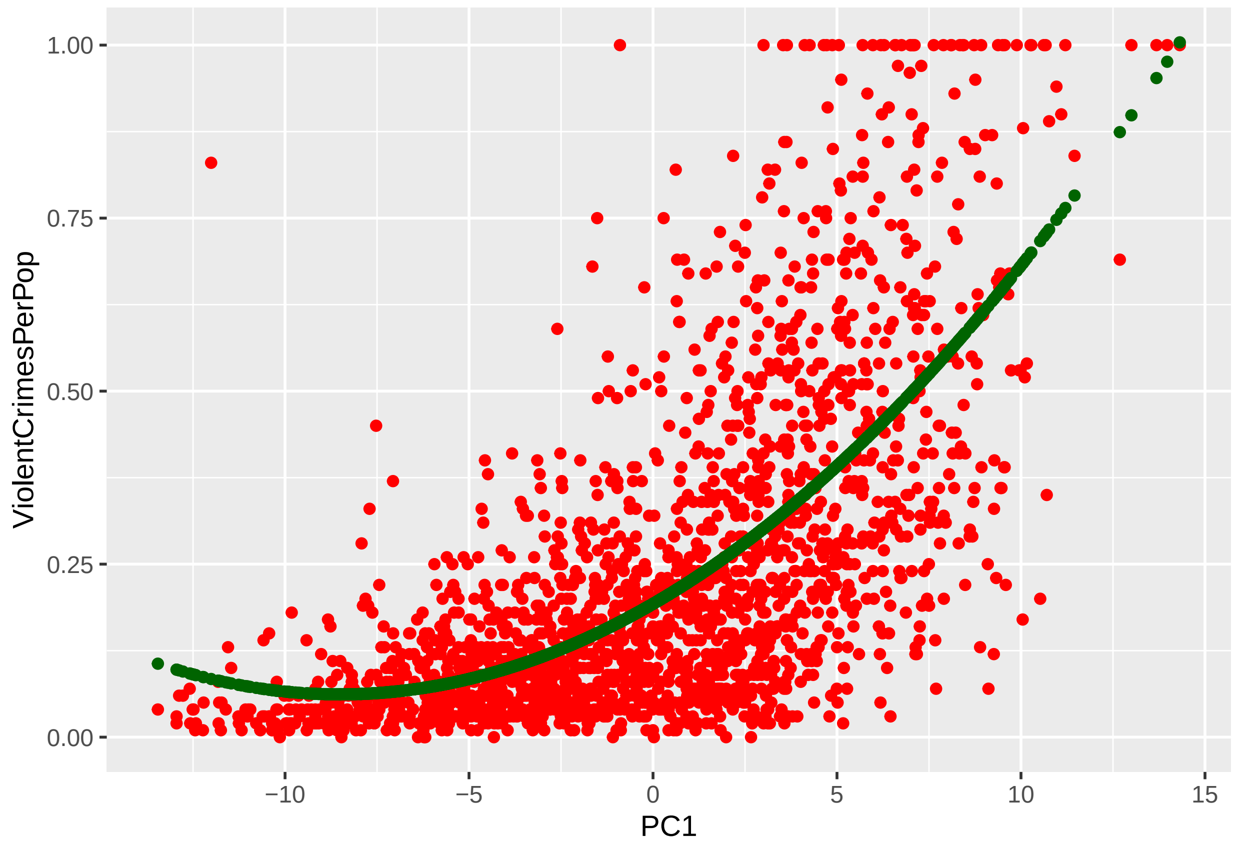
```
##
## The features that contribute the most to the first principle component are:
##  PctPopUnderPov pctWInvInc PctKids2Par medIncome medFamInc
```

With **PctPopUnderPov** being the percentage of people under the poverty level, **pctWInvInc** is the percentage of households with investment / rent income in 1989, **PctKids2Par**, is the percentage of kids in family housing with two parents, **medIncome** is the median household income, **medFamInc** is the median family income (differs from household income for non-family households). Most of these factors have to do with wealth. Economic background may influence the apppearing of crimes but a sociological analysis is required to understand this relationship.

By visual analysis of the absolute value of contribution of features for the first principal components, we can see that there are a dozen features that contribute by more than 0.15 to this principal components. However, no feature seem to clearly contribute more than others. In our score plot for the first two principal components, we can see that the values for **ViolentCrimesPerPop** are relatively small for larger values of PC2, whereas larger values of PC1 seem to account for larger values of **ViolentCrimesPerPop**. We can deduce from this plot that PC1 has a greater influence on **ViolentCrimesPerPop** than PC2. The first principal component will explain the change in data better than PC2.

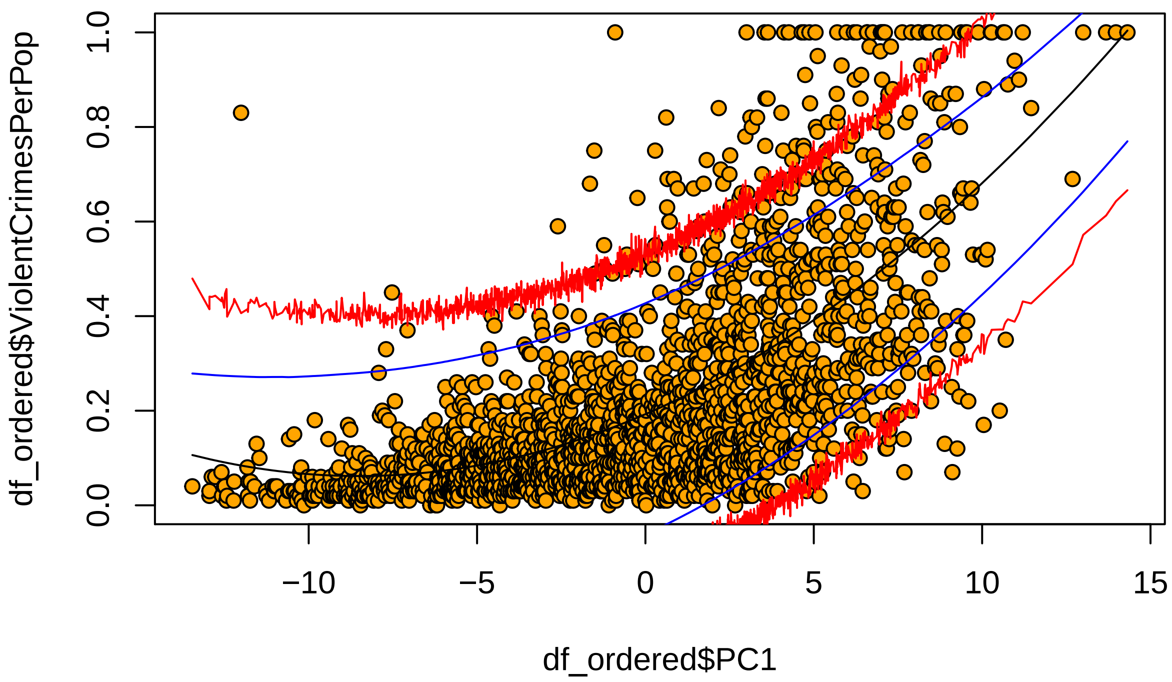
3. Fit a second order polynomial between ViolentCrimesPerPop and PC1

```
##
## Call:
## lm(formula = data[, 101] ~ poly(Pr_comp1, degree = 2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.55357 -0.09553 -0.02410  0.06868  0.83347
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.237979   0.003861   61.64   <2e-16 ***
## poly(Pr_comp1, degree = 2)1 6.557292   0.172391   38.04   <2e-16 ***
## poly(Pr_comp1, degree = 2)2 2.452782   0.172391   14.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1724 on 1991 degrees of freedom
## Multiple R-squared:  0.4531, Adjusted R-squared:  0.4525
## F-statistic: 824.6 on 2 and 1991 DF,  p-value: < 2.2e-16
```



We can see here that the first principal component (in green) fit with a second order polynomial follows the general trend of the data (in red). However, we observe a lot of noise that is unexplained by our linear model.

4 Perform parametric bootstrapping and plot confidence and prediction bands.



The confidence bands are marked in blue and the prediction bands are marked in red. Our prediction bands cover most of our data and we expect new data points to fall inside them with only a few outliers on top of the upper limit values. The confidence bands are wider for smaller values of **ViolentCrimesPerPop** than for larger ones, but we saw from the score plot that for smaller values of **ViolentCrimesPerPop**, we had more variability in the second principal component. This graph also tells us that for higher values of **ViolentCrimesPerPop**, we are more confident in our prediction as the interval becomes smaller.

Appendix: All code for this report

```
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)
library(ggplot2)
library(MASS)
library(mvtnorm)
library(nnet)


plot(Sepal.Width ~ Sepal.Length, data = iris, col = c("darkgreen", "blue", "red")[as.integer(iris$Speci
legend(x="topleft",
       legend=c("setosa","versicolor","virginica"),
       col=c("darkgreen", "blue", "red"),
       pch=c(1,1,1))


data_frame = iris[c("Sepal.Length", "Sepal.Width", "Species")]

species = unique(data_frame$Species)


get_class_means = function(label, data_frame){
  number_data = data_frame[data_frame$Species == label,][c("Sepal.Length", "Sepal.Width")]
  return(apply(number_data,2,mean))
}

get_class_covs = function(label, data_frame){
  number_data = data_frame[data_frame$Species == label,][c("Sepal.Length", "Sepal.Width")]
  return(cov(number_data))
}

get_class_priors = function(label, data_frame){
  return(
    dim(data_frame[data_frame$Species == label,])[1]/dim(data_frame)[1]
    )
}

sample_means = sapply(as.vector(species), get_class_means, data_frame=data_frame)

sample_covs = lapply(as.vector(species), get_class_covs, data_frame=data_frame)
names(sample_covs) = species

sample_priors = sapply(as.vector(species), get_class_priors, data_frame=data_frame)
print("Sample means")
print(sample_means)
print("Covariance")
print(sample_covs)
print("Priors")
print(sample_priors)


pooled_cov = function(covs, priors){
  pooled_covs = covs[[1]] * priors[[1]]
  for(class in 2:length(covs)){
    pooled_covs = pooled_covs + covs[[class]]*priors[[class]]
  }
  return(pooled_covs)
}
sample_pooled_cov = pooled_cov(sample_covs, sample_priors)
sample_pooled_cov


discriminant_function = function(class, sample_mean, sample_prior, sample_pooled_cov, data){
  x=as.matrix(data[c("Sepal.Length", "Sepal.Width")])
  sigma_minus = as.matrix(solve(sample_pooled_cov))
  mu_k = sample_mean[,class]
  pi_k = sample_prior[class]


  delta_k_1 = (x %*% sigma_minus) %*% mu_k

  delta_k_2 = as.numeric(((1/2)*t(mu_k)) %*% sigma_minus %*% mu_k)
  delta_k_3 = log(pi_k)
  return(delta_k_1 + delta_k_2 + delta_k_3)
}

discriminant_functions = matrix(,dim(data_frame)[1],length(species))
colnames(discriminant_functions) = species

for(class in 1:length(species)){
  discriminant_functions[,class] = discriminant_function(
  species[class],
  sample_mean = sample_means,
  sample_prior = sample_priors,
  sample_pooled_cov = sample_pooled_cov,
  data = data_frame)
}

head(discriminant_functions)


boundry = function(class, sample_mean, sample_prior, sample_pooled_cov){
  mu_1 = sample_mean[,class]
  pi_1 = sample_prior[class]
  sigma = solve(sample_pooled_cov)

  w_0i = -(1/2)*t(mu_1) %*% sigma %*% mu_1 + log(pi_1)
  w_i = sigma %*% mu_1

  return(c(w_0i, w_i))
}
setosa = boundry("setosa", sample_means, sample_priors, sample_pooled_cov)
virginica = boundry("virginica", sample_means, sample_priors, sample_pooled_cov)
versicolor = boundry("versicolor", sample_means, sample_priors, sample_pooled_cov)
```

```r
coeff = data.frame(
  w0i = c(setosa[1],virginica[1],versicolor[1]),
  w1 = c(setosa[2], virginica[2],versicolor[2]),
  w2 = c(setosa[3], virginica[3],versicolor[3])
  )
rownames(coeff) = c("setosa", "virginica", "versicolor")
coeff


calculate_y = function(x1, x2, w0, w1, w2){
  return(w0 + w1*x1 + w2*x2)
}

miss_rate = function(y_true, y_pred){
  return(mean(y_true != y_pred))
}

calculate_class = function(point, data, coeff){
  x1 = data[point,1]
  x2 = data[point,2]
  species = c("setosa", "versicolor", "virginica")
  class_scores = c()
  for(i in 1:length(species)){
    score = calculate_y(
      x1=x1,
      x2=x2,
      w0 = coeff[species[i],1],
      w1 = coeff[species[i],2],
      w2 = coeff[species[i],3]
      )
    class_scores = c(class_scores, score)
  }
  return(which.max(class_scores))
}

predict_y = function(data, coeff){
  points = 1:dim(data)[1]
  results = sapply(points, calculate_class, data=data, coeff=coeff)
  #results[results==1] = "setosa"
  #results[results==2] = "virginica"
  #results[results==3] = "versicolor"
  return(results)
}

predicted = predict_y(data=data_frame[c("Sepal.Length", "Sepal.Width")], coeff = coeff)

plot(Sepal.Width ~ Sepal.Length, data = iris, col = c("darkgreen", "blue", "red")[as.integer(predicted)
legend(x="topleft",
       legend=c("setosa","versicolor","virginica"),
       col=c("darkgreen", "blue", "red"),
       pch=c(1,1,1))

print(paste("Misclassification rate", as.character(miss_rate(as.integer(data_frame$Species), predicted)
model = lda(Species~., data=data_frame)
labels = predict(model, data_frame)$class

plot(Sepal.Width ~ Sepal.Length, data = iris, col = c("darkgreen", "blue", "red")[as.integer(labels)])
legend(x="topleft",
       legend=c("setosa","versicolor","virginica"),
       col=c("darkgreen", "blue", "red"),
       pch=c(1,1,1))


print(paste("Misclassification rate", as.character(miss_rate(as.integer(data_frame$Species), as.integer


sampling = rmvnorm(50, mean = sample_means[,1], sigma = sample_covs[[1]])
new_iris = data.frame(
  "Sepal.Length" = sampling[,1],
  "Sepal.Width" = sampling[,2],
  "Species" = replicate(50, species[1])
  )

for(i in 2:length(species)){
  sampling = rmvnorm(50, mean = sample_means[,i], sigma = sample_covs[[i]])
  new_species = data.frame(
    "Sepal.Length" = sampling[,1],
    "Sepal.Width" = sampling[,2],
    "Species" = replicate(50, species[i])
    )
  new_iris = rbind(new_iris, new_species)
}

plot(Sepal.Width ~ Sepal.Length, data = new_iris, col = c("darkgreen", "blue", "red")[as.integer(new_ir
legend(x="topleft",
       legend=c("setosa","versicolor","virginica"),
       col=c("darkgreen", "blue", "red"),
       pch=c(1,1,1))


model = multinom(Species~., data=data_frame)
multinom_predictions = predict(model, data_frame)

plot(Sepal.Width ~ Sepal.Length, data = iris, col = c("darkgreen", "blue", "red")[as.integer(multinom_p
legend(x="topleft",
       legend=c("setosa","versicolor","virginica"),
       col=c("darkgreen", "blue", "red"),
       pch=c(1,1,1))


print(paste("Misclassification rate", as.character(miss_rate(data_frame$Species, multinom_predictions))

#Importing and Dividing DATA
d<-(data.frame(read.csv("bank-full.csv")))
```

```r
library(tree)
library(rpart)


data<-read.table("bank-full.csv",stringsAsFactors = TRUE,sep=";",header=TRUE)
data=data[,-12]

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]
str(train)
#Training the Model using train dataset

model1<-tree(y~.,train)
model2<-tree(y~.,train,minsize=7000)
model3<-tree(y~.,train,mindev=0.0005)


#Training the Model using validation dataset
model_val_1<-predict(model1,valid,type="class")
model_val_2<-predict(model2,valid,type="class")
model_val_3<-predict(model3,valid,type="class")

confm_val_1<-table(model_val_1,valid$y)
confm_val_2<-table(model_val_2,valid$y)
confm_val_3<-table(model_val_3,valid$y)

error_val_1<-1-(sum(diag(confm_val_1))/sum(confm_val_1))
error_val_2<-1-(sum(diag(confm_val_2))/sum(confm_val_2))
error_val_3<-1-(sum(diag(confm_val_3))/sum(confm_val_3))

errors_validation<-data.frame(cbind(error_val_1,error_val_2,error_val_3))


#errors_train
misclass_tr_a<-misclass.tree(model1,detail=FALSE)
misclass_tr_b<-misclass.tree(model2,detail=FALSE)
misclass_tr_c<-misclass.tree(model3,detail=FALSE)
error1<-misclass_tr_a/nrow(train)
error2<-misclass_tr_b/nrow(train)
error3<-misclass_tr_c/nrow(train)
errors_tr<-as.data.frame(cbind(error1,error2,error3))
errors_tr
errors_validation

train_dev<-c()
```

```r
valid_dev<-c()
for(i in 2:50)
{
  prune_train<-prune.tree(model3,best=i)
  prune_pred<-predict(prune_train,newdata=na.tree.replace(valid),type="tree")
  train_dev[i]<-deviance(prune_train)
  valid_dev[i]<-deviance(prune_pred)
}

plot(2:50,train_dev[2:50],type ="b",col="blue",ylim = c(8000,12000),xlab="First 50 leaves" )
points(2:50,valid_dev[2:50],type="b",col="red")

best_node<-which.min(valid_dev)
best_node
as.character(summary(prune_train)$used)
final_opt_tree<-prune.tree(model3,best=best_node)
pred_optimal<-predict(final_opt_tree,newdata = test,type="class")
test_confm<-table(pred_optimal,test$y)
#confusion Matrix
test_confm
test_error<-1-sum(diag(test_confm))/sum(test_confm)
#misclassification error
test_error
loss_tree<-rpart(y~.,data=train,method="class",parm=list(loss=matrix(c(0,1,5,0),byrow=TRUE,nrow=2)),con
pred_loss<-predict(loss_tree,test,type="class")
loss_confm<-table(pred_loss,test$y)
error_loss<-1-sum(diag(loss_confm))/sum(loss_confm)
error_loss
loss_confm
library(MASS)
library(e1071)

fit<-naiveBayes(y~.,data=train)
pred_naive<-predict(fit,newdata=test,type="raw")

pred_optimal_2<-predict(final_opt_tree,newdata = test)


classify_tree<-list()
classify_naive<-list()
naive_conmat<-list()
tree_conmat<-list()
tpr_naive<-c()
fpr_naive<-c()
tpr_tree<-c()
fpr_tree<-c()

pi<-seq(0.05,0.95,0.05)

for(i in 1:length(pi))
{
  classify_tree<-factor(ifelse(pred_optimal_2[,2]>pi[i],"yes","no"),levels = c("no","yes"))
  classify_naive<-factor(ifelse(as.data.frame(pred_naive[,2])>pi[i],"yes","no"),levels = c("no","yes"))
  naive_conmat<-table("true"=test$y,"predicted"=classify_naive)
  tree_conmat<-table("true"=test$y,"predicted"=classify_tree)
  #naive_conmat[i]<-list(table(test$y,classify_naive[[i]]))
  #tree_conmat[i]<-list(table(test$y,classify_naive[[i]]))

  tpr_naive[i]<-naive_conmat[2,2]/sum(naive_conmat[2,2],naive_conmat[2,1])
  fpr_naive[i]<-naive_conmat[1,2]/sum(naive_conmat[1,1],naive_conmat[1,2])
  tpr_tree[i]<-tree_conmat[2,2]/sum(tree_conmat[2,2],tree_conmat[2,1])
  fpr_tree[i]<-tree_conmat[1,2]/sum(tree_conmat[1,2],tree_conmat[1,1])
  i=i+1
}


plot(fpr_tree,tpr_tree,type="l",col="red")
lines(fpr_naive,tpr_naive,type="l",col="blue")
legend('bottomright',legend=c('Optimal_tree','Naive_Bayes_Model'),lty=1:2,col=c("red","blue"))

# rm(list = ls())
setwd(dir = "~/732A99_ML_Lab2_block1")
data <- read.csv("communities.csv")
# Scale the data
data[, -c(101)] = scale(data[, -c(101)])

# Calculate the mean of every feature and compute the centered data matrix:
mean_vector <- c()
for(i in 1:101){
  mean_vector[i] <- mean(data[,i])
}

centered_data <- matrix(0, nrow = nrow(data), ncol= ncol(data))
for(i in 1:ncol(data)){
  for(j in 1:nrow(data)){
    centered_data[j,i] <- data[j,i]-mean_vector[i]
  }
}
#compute the covariance matrix
covar_matrix <- t(centered_data) %*% centered_data * (1/nrow(data))
#calculate eigenvalues
eigenvalues <- eigen(covar_matrix)$values

# Calculate how many features are needed to explain 95% of the variance of the data
cumul <- 0
i <- 0
while(cumul<95){
  i <- i+1
  cumul <- cumul + eigenvalues[i]
}
cat(i,"features are needed to explain 95% of the variance in the data")

proportion_firstPCs <- sum(eigenvalues[1:2])
```

```
cat("The first two principal components account for ", proportion_firstPCs,"% of the variance in the da
# Repeat PCA analysis by using princomp() function and make the score plot of the first principle compo
res <- prcomp(data)
lambda <- res$sdev^2
# lambda
sprintf("%2.3f", lambda/sum(lambda)*100)
screeplot(res, main="Variance explained by principal components")

# Plot the first principal component
U <- res$rotation
plot(U[,1], main ="Traceplot, PC1")

# 5 largest contributing factors
U_abs <- abs(res$rotation)
top_features <- tail(order(U_abs[,1]), 5)
plot(U_abs[,1], main = "Traceplot by absolute value, PC1")


# Plot of the PC scores in the coordinates (PC1, PC2) in which the color of the points is given by Viol
library(ggplot2)
df <- data.frame(res$x[,1], res$x[,2], data[,101])
colnames(df) <- c("PC1", "PC2", "ViolentCrimesPerPop")

ggplot(df, aes(x=PC1, y=PC2, color=ViolentCrimesPerPop))+
  geom_point()+
  labs(title="Score plot of PC1 and PC2 with respect to ViolentCrimesPerPop")


cat("\nThe features that contribute the most to the first principle component are: \n", colnames(data[t


# Assume a second order polynomial regression model in which ViolentCrimesPerPop is target and PC1 is t
Pr_comp1<- res$x[,1]
regress <- lm(data[,101]~poly(Pr_comp1, degree = 2))
summary(regress)

# Sscatterplot of the target versus the feature and present also the predicted values in this plot.
df <- data.frame(cbind(Pr_comp1,data[,101]))
colnames(df) <- c("PC1", "ViolentCrimesPerPop")

plot2 <- ggplot(data = df, aes(x= PC1, y=ViolentCrimesPerPop))+
  geom_point(col="red")+
  geom_point(aes(x= Pr_comp1 , y= regress$fitted.values), col = "dark green")
plot2

library(boot)
# Parametric bootstrap to estimate the confidence and prediction bands from the model from step 3 and a

df_ordered <- df[order(df$PC1),] # this is data2 in the slides. Ordered data with respect to the featur

#generate the data from a our data and model
rng <- function(df, regress){
  data1=data.frame(ViolentCrimesPerPop = df$ViolentCrimesPerPop, PC1 = df$PC1)
```

```r
  n <- length(data[,101])
  data1$ViolentCrimesPerPop <- rnorm(n, predict(regress, newdata=data1), sd(regress$residuals))
  return(data1)
}

# Predict from the newly generated data
f1 <- function(data1){
  res <- lm(data1$ViolentCrimesPerPop~poly(data1$PC1, degree = 2), data=data1)
  crime_predict <- predict(res, newdata =df_ordered)
  return(crime_predict)
}

# bootstrap
bootstrap <- boot(df_ordered, statistic = f1, R=1000, mle = regress, ran.gen = rng, sim="parametric")

e=envelope(bootstrap) #compute confidence bands: confidence in regression

fit <- lm(df_ordered$ViolentCrimesPerPop~poly(df_ordered$PC1, degree = 2), data = df_ordered)
crime_pred <- predict(fit)

plot(df_ordered$PC1, df_ordered$ViolentCrimesPerPop, pch = 21, bg="orange")
points(df_ordered$PC1, crime_pred, type="l")

# Enveloppe is calculated from the predicted model by adding and substracing away from the fit.
points(df_ordered$PC1, crime_pred + e$point[2,], type="l", col="blue")
points(df_ordered$PC1, crime_pred - e$point[1,], type="l", col="blue")


# Prediction bands: confidence in future data points
mle <- lm(df_ordered$ViolentCrimesPerPop~poly(df_ordered$PC1, degree = 2), data = df_ordered)

f2 <- function(data1){
  res <- lm(data1$ViolentCrimesPerPop~poly(data1$PC1, degree = 2), data = data1)
  crime <- predict(res, newdata = df_ordered)
  n <- length(df_ordered$ViolentCrimesPerPop)
  predicted_crimes <- rnorm(n, crime, sd(mle$residuals))
  return(predicted_crimes)
}

results <- boot(df_ordered, statistic = f2, R = 1000, mle = mle, ran.gen = rng, sim = "parametric")
e2 <- envelope(results, level = 0.95)


points(df_ordered$PC1, e2$point[2,], type="l", col="red")
points(df_ordered$PC1, e2$point[1,], type="l", col="red")
```