Your report looks really good and nice explanations!

LAB 2 BLOCK 1 - Group A15

*Zuxiang Li (**zuxli371**), Marcos F. Mourao (**marfr825**), Agustín Valencia (**aguva779**)*

08 December 2019

Assignment 2: Analysis of credit scoring

1. Import the data to R and divide into training/validation/test as 50/25/25: use data partitioning code specified in Lecture 1e.

The data was divided as requested and all code used is in the appendix.

2. Fit a decision tree to the training data by using the following measures of impurity: *deviance and Gini index*, and report the misclassification rates for the training and test data. Choose the measure providing the better results for the following steps.

Misclassification rates for **deviance** impurity measure:

```
## [1] "Misclassification rate for TRAINING data:  21.2 %."
```

```
## [1] "Misclassification rate for TESTING data:  26.8 %."
```

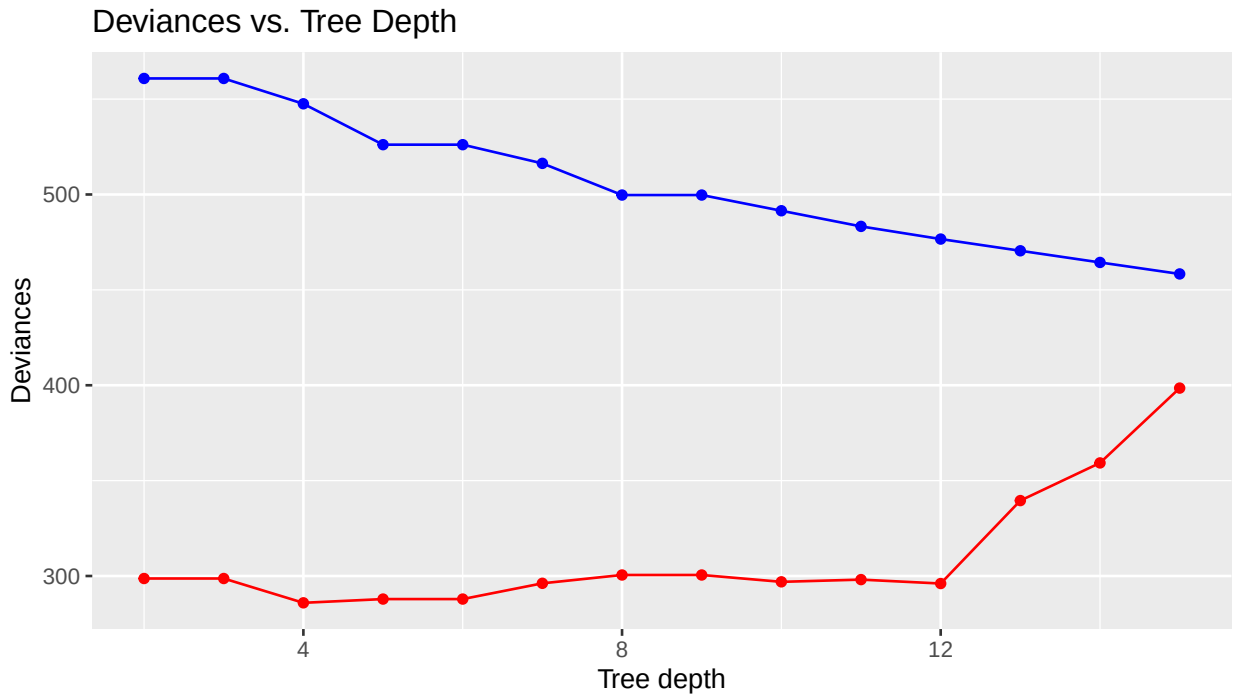
Misclassification rates for **Gini's index** impurity measure:

```
## [1] "Misclassification rate for TRAINING data:  23.8 %."
```

```
## [1] "Misclassification rate for TESTING data:  37.2 %."
```

Altough showing similar results, we choose the deviance measure as it has a lower misclassification rate for both training (21.2% against 23.8%) and testing data (26.8% against 37.2%).

3. Use training and validation sets to choose the optimal tree depth. Present the graphs of the dependence of deviances for the training and the validation data on the number of leaves. Report the optimal tree, report it's depth and the variables used by the tree. Interpret the information provided by the tree structure. Estimate the misclassification rate for the test data.

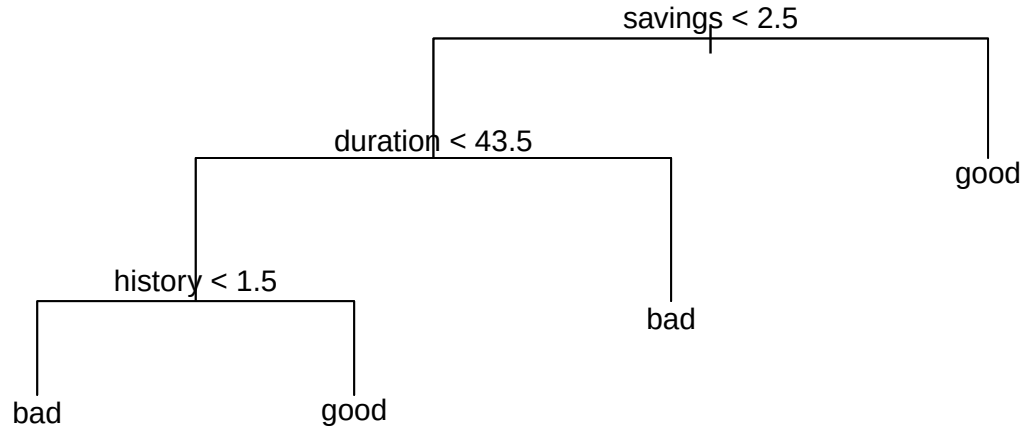


The deviance plot suggests that an optimal tree depth (minimal deviance in testing) of 4 should be used. Deeper trees perform better on training, but worse in testing due to overfitting. The optimal tree is shown below. The misclassification rate of the optimal tree on the testing data is 25.6%, a better result compared to item 2 (26.8%).

The optimal tree uses the variables: `savings`, `duration` and `history`. The root node, `savings`, indicates that costumers that save more are more likely to pay their loans. `duration` implies that loans that take too long to be paid are more likely to never be actually totally paid. Finally, `history` says that older costumers within the company are more likely to pay their loans.

It is important to notice that in the data, out of 1000 entries, 700 of them are classified as "good", and only 300 are classified as "bad". Such disparity will impact on the impurity of leafs, specially in a shallow tree.

Good explanations!




```
##       predicted
## true   bad good
##   bad   18   58
##   good   6  168
```

```
## [1] "Misclassification rate of the OPTIMAL TREE for TESTING data:  25.6 %."
```

4. Use training data to perform classification using Naïve Bayes and report the confusion matrices and misclassification rates for the training and for the test data. Compare the results with those from step 3.

```
##       predicted
## true   bad good
##   bad   95   52
##   good  98  255
```

```
## [1] "Naive Bayes misclassification rate for TRAINING data:  30 %."
```

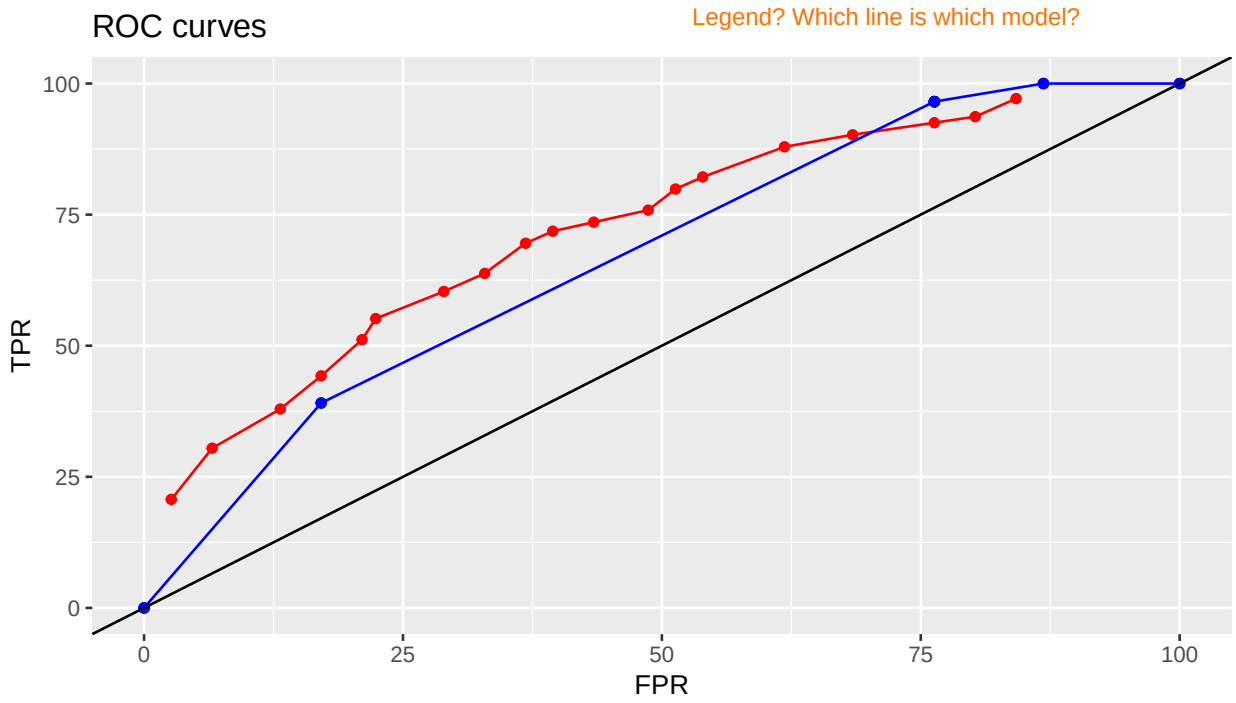
```
##       predicted
## true   bad good
##   bad   46   30
##   good  49  125
```

```
## [1] "Naive Bayes misclassification rate for TESTING data:  31.6 %."
```

The Naive Bayes classifier perfomerd worse than the optimal tree in both test and training, with higher misclassification rates.

5. Use the optimal tree and the Naïve Bayes model to classify the test data by using the following principle: $\hat{Y} = 1$ if $p(Y = good|X) > \pi$, otherwise $\hat{Y} = 0$ where $\pi = 0.05, 0.1, 0.15, ..., 0.9, 0.95$. Compute the TPR and FPR values for the two models and plot the corresponding ROC curves. Conclusion?

The ROC curves for the optimal tree and Naive Bayes model are shown below:



The ROC curves indicate that the Naive Bayes classifier performs better (it is closer to the perfect classification point (100,0)). Note - higher area under the curve.

6. Repeat Naïve Bayes classification as it was in step 4 but use the following loss matrix:

$$L = \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

and report the confusion matrix for the training and test data. Compare the results with the results from step 4 and discuss how the rates has changed and why.

```
##       predicted
## true     0   1
##   bad   27 120
##   good  17 336
```

These results are wrong for the custom loss matrix.

```
## [1] "Naive Bayes misclassification rate for TRAINING data:  27.4 %."
```

```
##       predicted
## true     0   1
##   bad   14  62
##   good  10 164
```

```
## [1] "Naive Bayes misclassification rate for TESTING data:  28.8 %."
```

The given loss matrix penalizes false positives ten times more than false negatives. The confusion matrix produced by this new penalization presents, of course, less false positives. The misclassification rates also were reduced for both training and testing data.

In this particular setting, a false positive is indeed "worse" than a false negative. Classifying a costumer as "good" when he/she actually does not pay his/her loan ("bad") incurs in greater financial loss to the company than classifying a "good" costumer as "bad" and not lend him/her money. This model, therefore, is preffered over the one proposed in item 4.

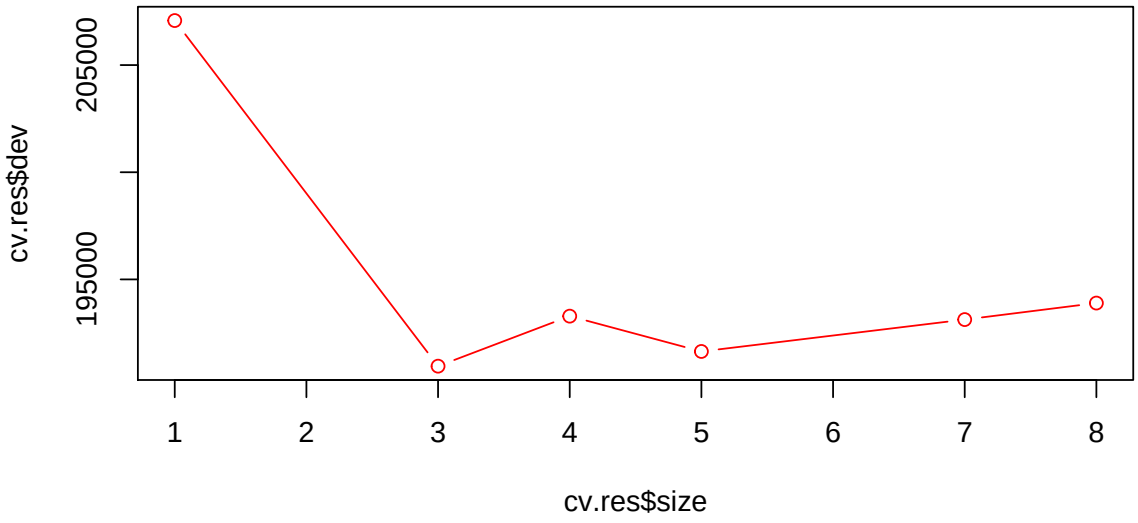
Assignment 3. Uncertainty estimation

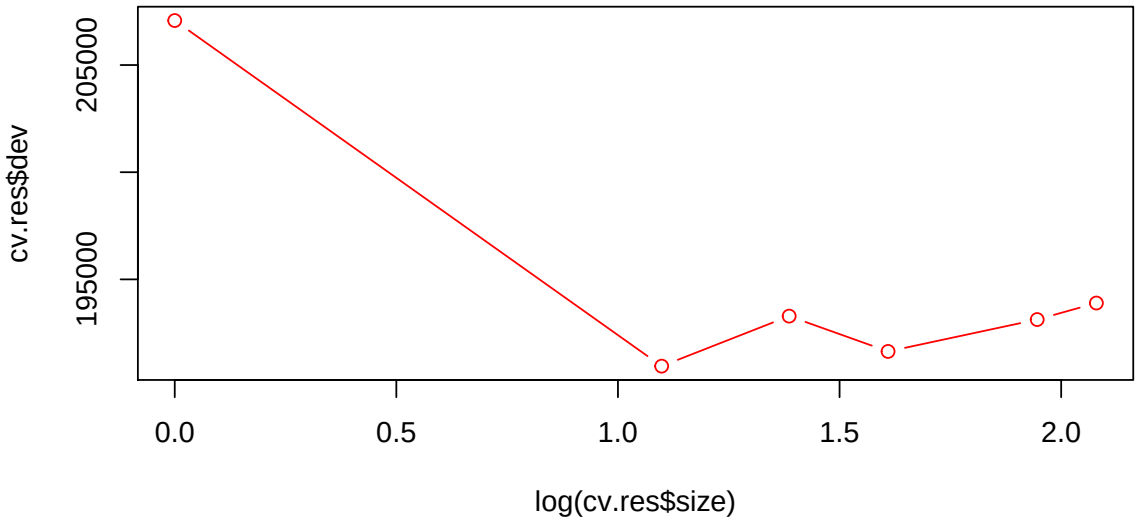
The data file State.csv contains per capita state and local public expenditures and associated state demographic and economic characteristics, 1960, and there are variables

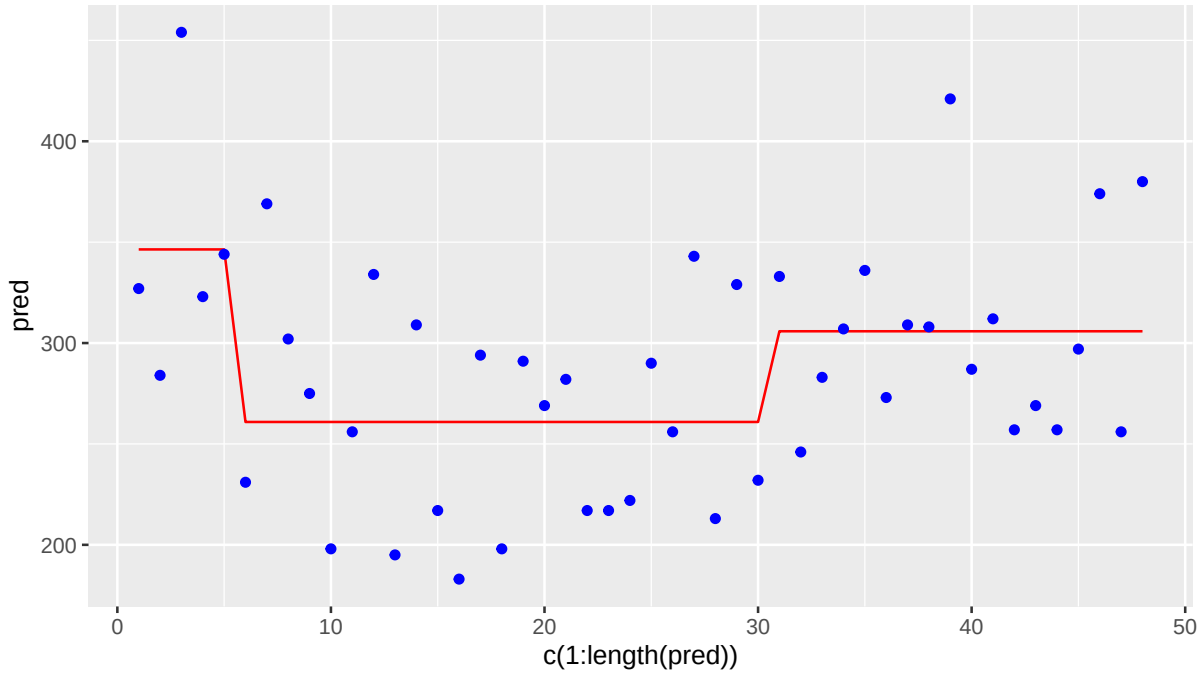
- MET: Percentage of population living in standard metropolitan areas
- EX: Per capita state and local public expenditures ($)

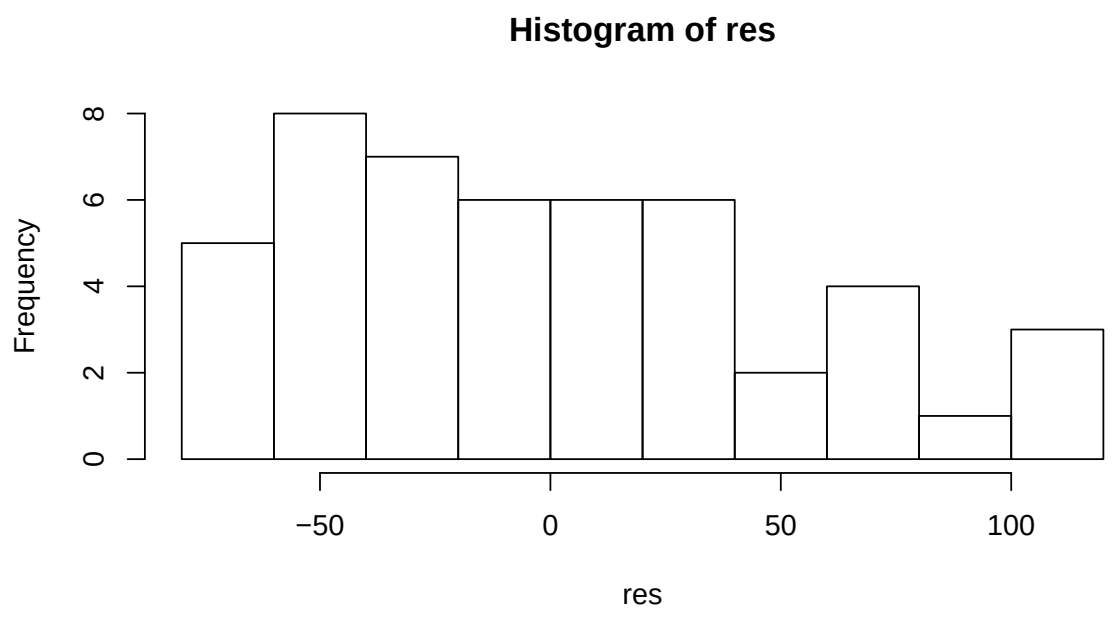
1. Reorder your data with respect to the increase of MET and plot EX versus MET. Discuss what kind of model can be appropriate here. Use the reordered data in steps 2-5.

2. Use package tree and fit a regression tree model with target EX and feature MET in which the number of the leaves is selected by cross-validation, use the entire data set and set minimum number of observations in a leaf equal to 8 (setting minsize in tree.control). Report the selected tree. Plot the original and the fitted data and histogram of residuals. Comment on the distribution of the residuals and the quality of the fit.




cv.res$dev
205000
195000
0.0
0.5
1.0
1.5
2.0
log(cv.res$size)


pred
400
300
200
0
10
20
30
40
50
c(1:length(pred))




```
## [1] 1.184223e-14
```

```
## [1] 50.82183
```

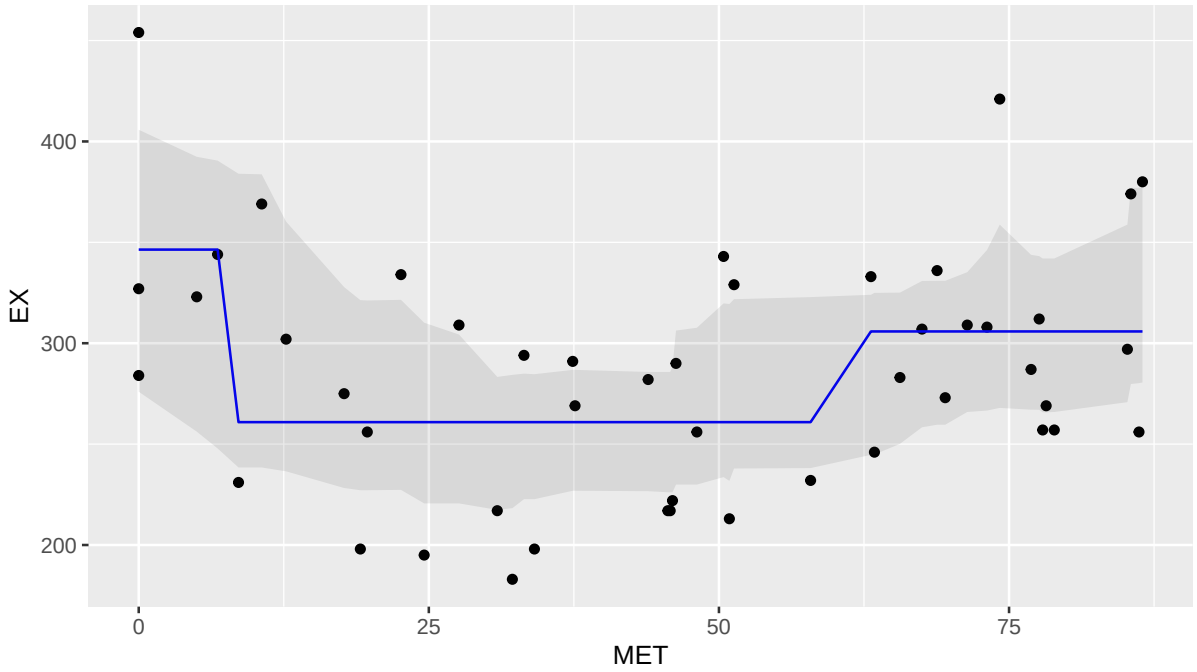
I don't think your residuals are normally distributed based on above plot.

The selected tree should be 3 in this case, the residuals distributed as normal distribution with mean 0 and standard deviance 50.8. The histogram of residuals suggest it is slightly skewed to the right, which means it's a good fit initially.

This is not correct. Negative error is also still error.

3. Compute and plot the 95% confidence bands for the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a non-parametric bootstrap. Comment whether the band is smooth or bumpy and try to explain why. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable.

```
## Warning in prune.tree(tree_mod, best = 3): best is bigger than tree size
```



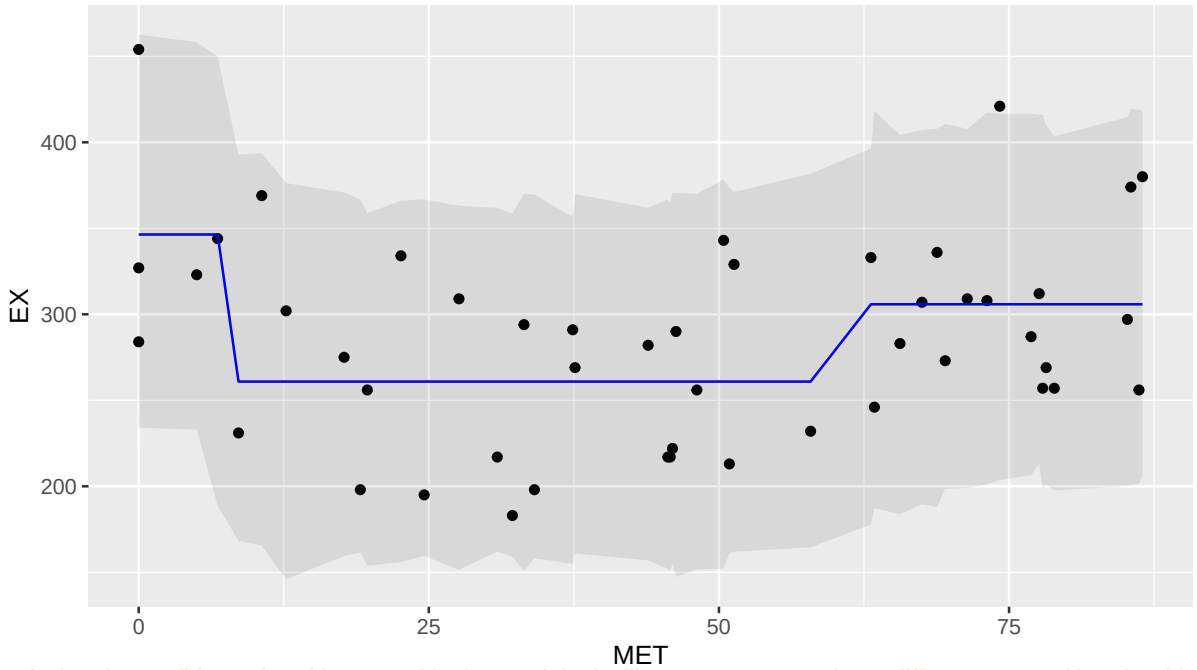
From the plot, we can see that the confidence band is quite bumpy and almost half of the dots are outside of the confidence band, which indicate that it's not a reliable model. The band is bumpy because nonparametric bootstrap does not work well for discrete estimators.

The confidence band is for the mean of your predictions - you need not expect most of your data points to lie within this band.

4. Compute and plot the 95% confidence and prediction bands the regression tree model from step 2 (fit a regression tree with the same settings and the same number of leaves as in step 2 to the resampled data) by using a parametric bootstrap, assume are labels in the tree leaves and is the residual variance. Consider the width of the confidence band and comment whether results of the regression model in step 2 seem to be reliable. Does it look like only 5% of data are outside the prediction band? Should it be?

```
## Warning in envelope(boot_res1, level = 0.95): unable to achieve requested
## overall error rate
```

If your data has less noise, then the confidence band is going to be narrower and still a part of the data points are going to lie outside of this band.



You are missing the confidence band here. And in the text it looks like you are comparing 2 different types of bands which is wrong.

From the plot we can see that there are two points are outside the confidence band, it's more reliable than previous models.In addition, the sample with small size using nonparametric bootstrap can underestimate the amount of variation, but using parametric bootstrap which take values from known distribution covering a wider range.

Good that you thought about this!

5. Consider the histogram of residuals from step 2 and suggest what kind of bootstrap is actually more appropriate here.

Parametric bootstrap is more suitble here since the histrogram of residuals looks like normal distribution.

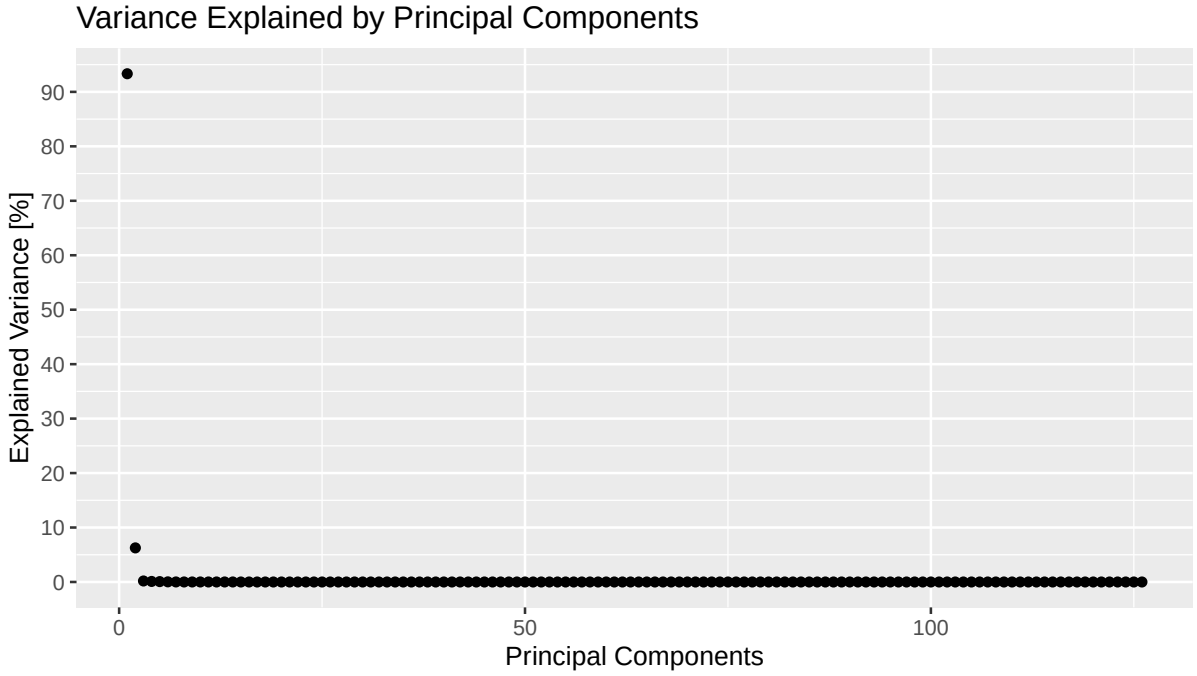
Really? :D You need to look up how normal distribution looks.

Assignment 4 - Principal components

The data file NIRspectra.csv contains near-infrared spectra and viscosity levels for a collection of diesel fuels. Your task is to investigate how the measured spectra can be used to predict the viscosity.

1. Conduct a standard PCA by using the feature space and provide a plot explaining how much variation is explained by each feature. Does the plot show how many PC should be extracted? Select the minimal number of components explaining at least 99% of the total variance. Provide also a plot of the scores in the coordinates (PC1, PC2). Are there unusual diesel fuels according to this plot?

According to the following plot very few Principal Components explain most of the variance, which implies that the data set dimension could be reduced from over a hundred to just a few variables.

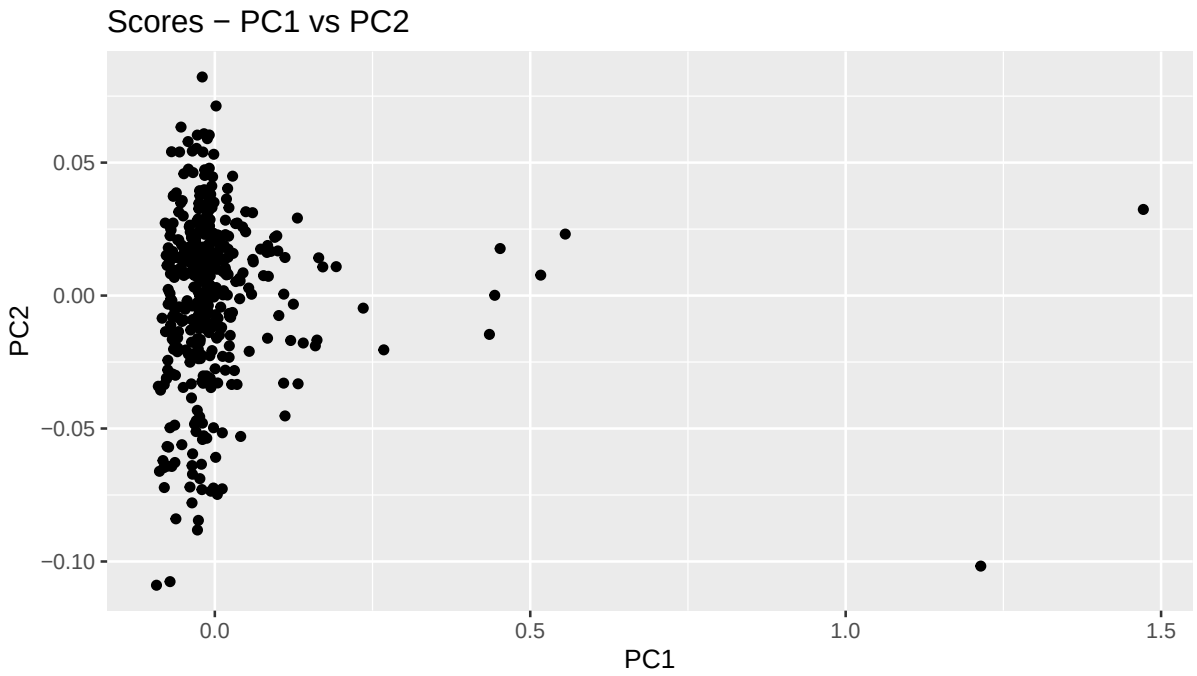


The printout below (close to zero values have been omitted) confirms that, after sorting their contributions, just the two first PCs represent 99.6% of the total variance. So the dimensionality could be reduced from 127 dimensions to just two.

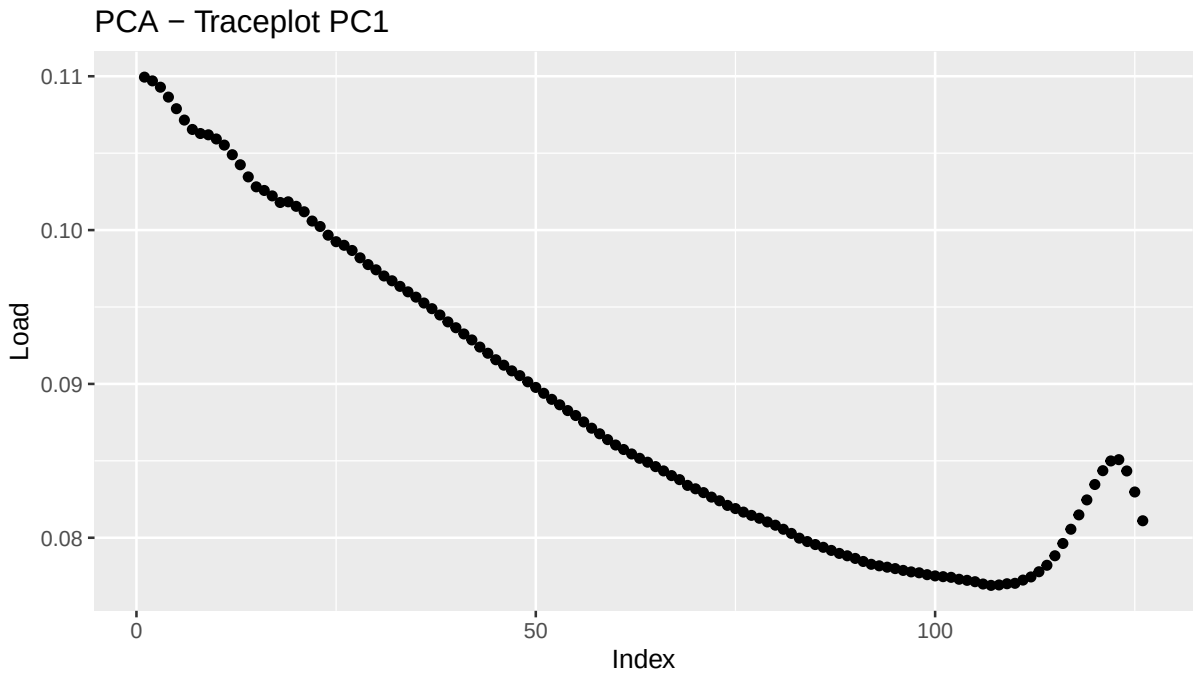
```
## [1] "Principal Components Contributions"
```

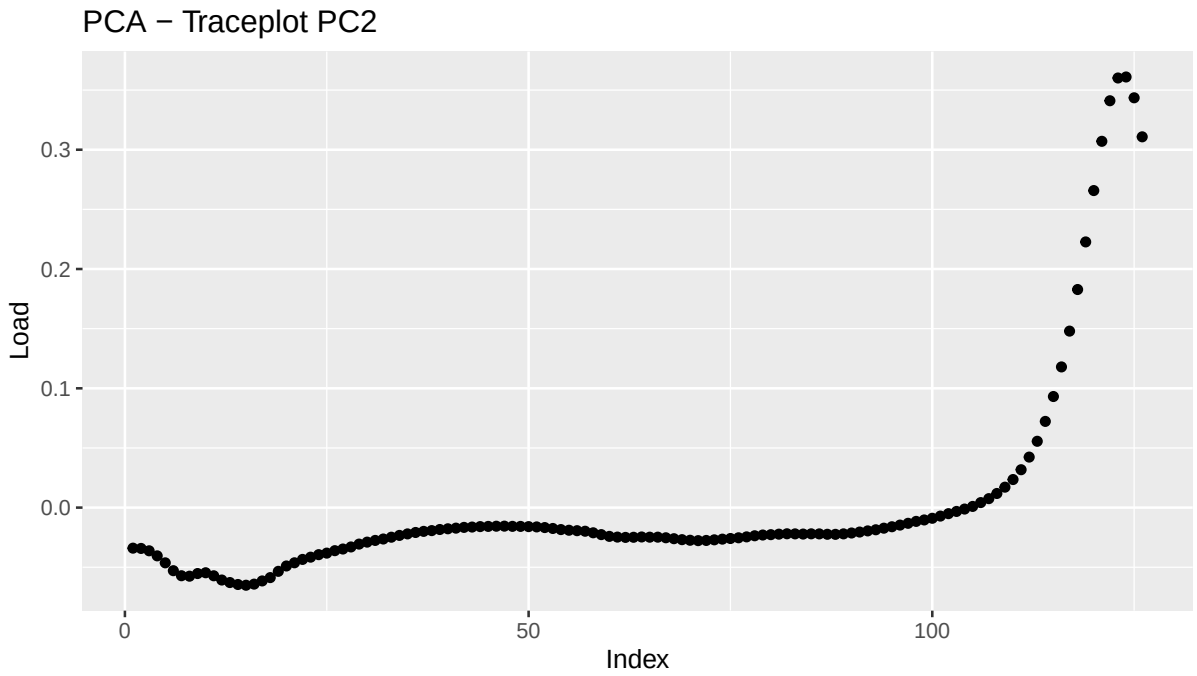
```
##  [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"  "0.003"
##  [9] "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"  "0.000"  "0.000"
```

The following plot show how the data is explained when the dimensionality is reduced to two principal components out of 127. Since these two new features are a linear combination of the 127 original characteristics, it looses interpretability, though for computational purposes it implies a good improvement. It can be seen some anormalities along the PC1



2. Make trace plots of the loadings of the components selected in step 1. Is there any principle component that is explained by mainly a few original features?





It can be seen that the two components are comprised by a linear combination of all the original variables. Although for the first component, the contributions come mainly for the first variables, and in the second the opposite, no weight represents more than a half of all the weights, thus it cannot be said that there is a principal component being represented mainly by one in the original space.

Maybe look closer at the weights for the 2nd component.

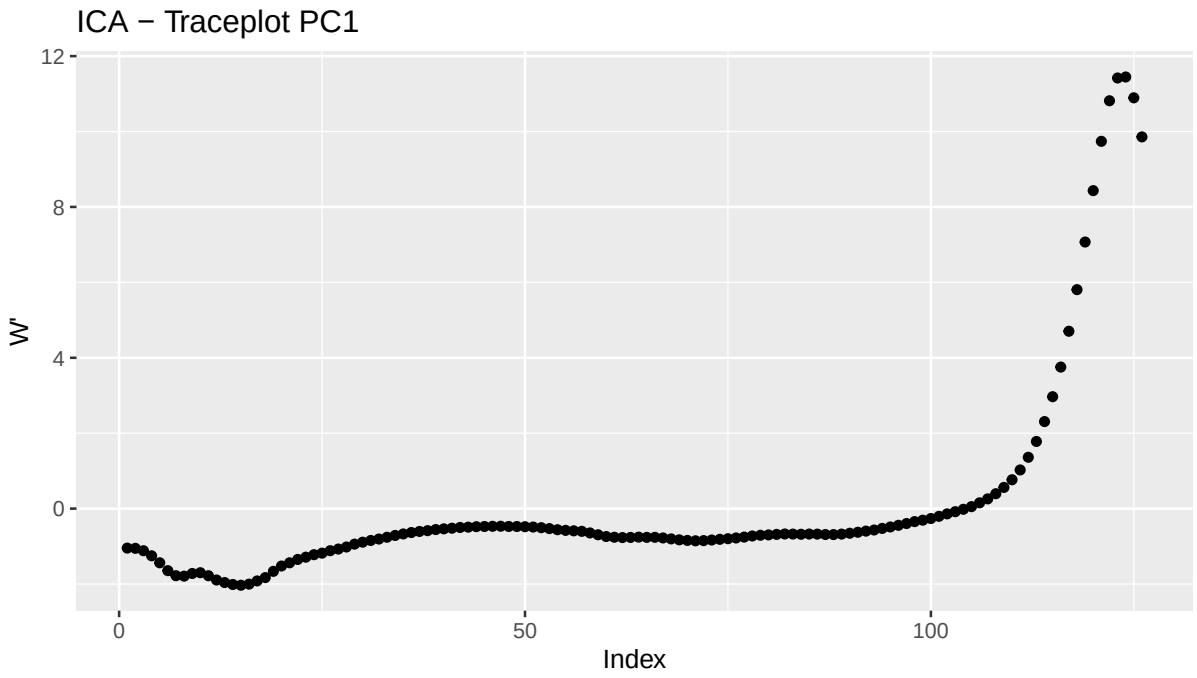
3. Perform Independent Component Analysis with the number of components selected in step 1 (set seed 12345). Check the documentation for the fastICA method in R and do the following:

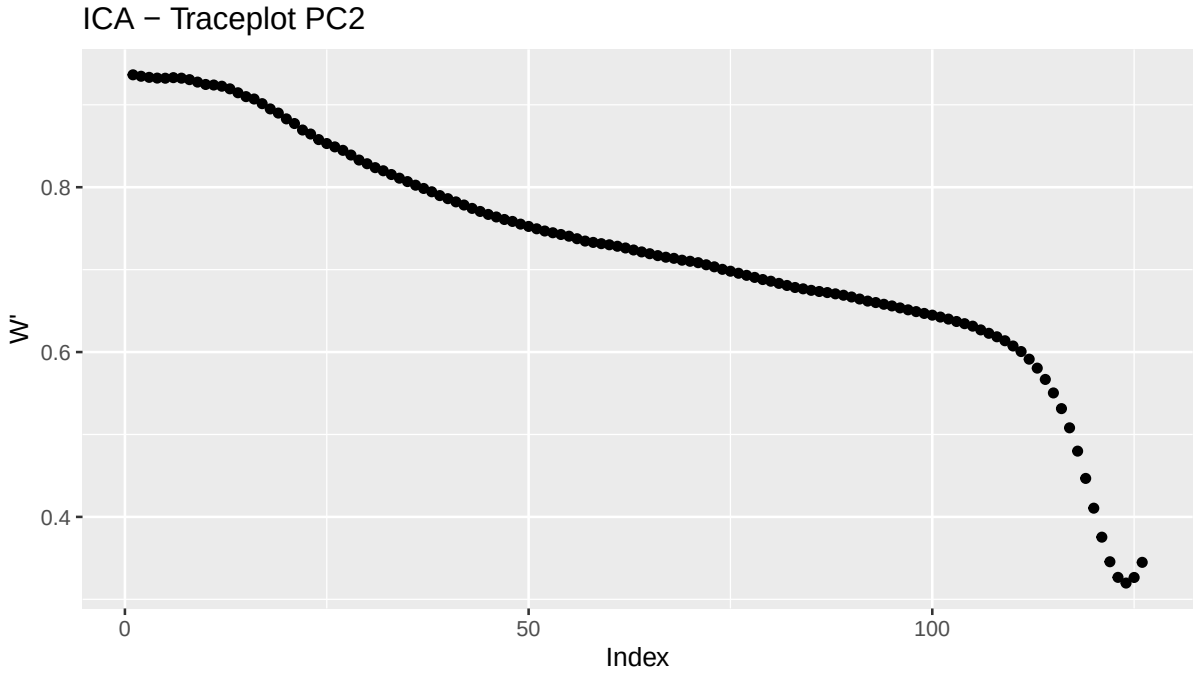
a. Compute $W' = K \cdot W$ and present the columns of W' in form of the trace plots. Compare with the trace plots in step 2 and make conclusions. What kind of measure is represented by the matrix W'?

W matrix contains the unmixing coefficients. It can be seen that the traceplots from it are somewhat similar but in a mirrored fashion. This will allow to *undo* the compression and get an approximation of the original variables before PCA.

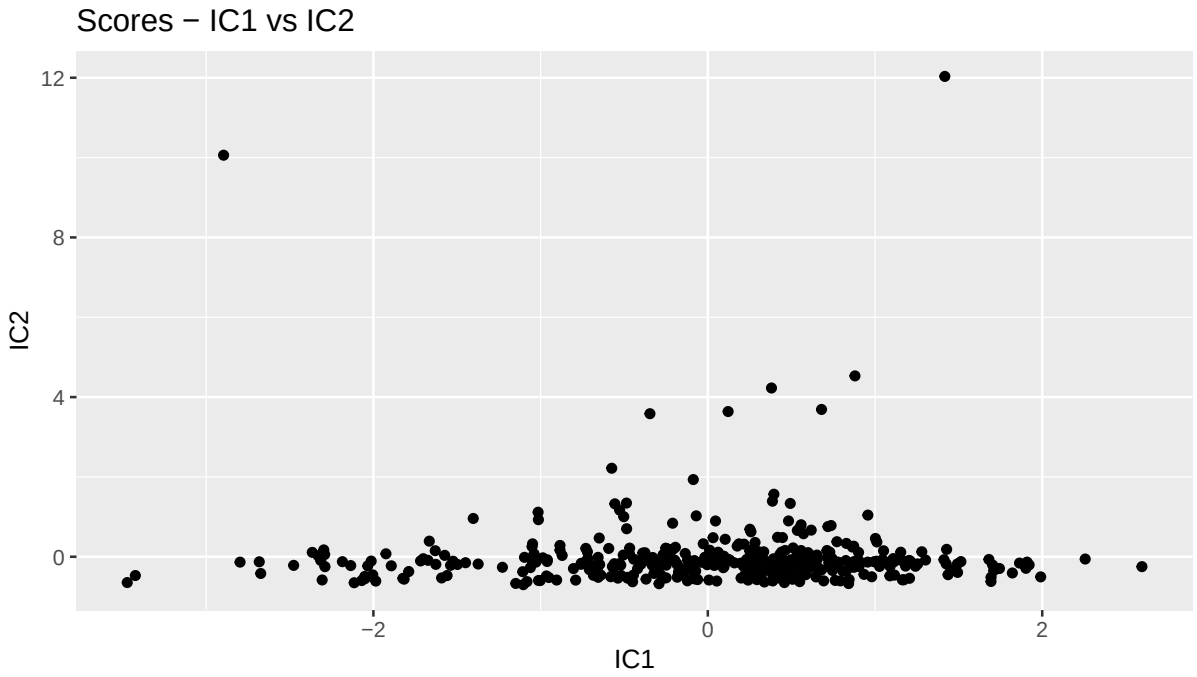
You are asked about W' and not W.

Unmixing is fine but the rest doesn't sound correct.
It just projects the values to the new components.


ICA − Traceplot PC1
12
8
4
0
W'
0
50
100
Index


ICA − Traceplot PC2
0.8
0.6
0.4
W'
0
50
100
Index

b. Make a plot of the scores of the first two latent features and compare it with the score plot from step 1.



It can be seen that the scores looks rotated when compared to the PCA scores.

Code Appendix

Assignment 2

2.1

```
#1. importing data
data_path <- "./Data/creditscoring.xls"
library(readxl)
data <- read_xls(data_path)
#changing to factor in order to trees to work
data$good_bad <- as.factor(data$good_bad)

#1. dividing into test, train and validation
n=dim(data)[1]
id=sample(1:n, floor(n*0.5))
train=data[id,]

id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]

id3=setdiff(id1,id2)
test=data[id3,]
```

2.2

```
#2. Fit decision trees
form <- good_bad ~ .

#deviance TRAINING
fit_dev <- tree(form, train, split = "deviance")
# plot(fit_dev)
# text(fit_dev, pretty = 0)
# fit_dev
# summary(fit_dev)
Ypred <- predict(fit_dev, newdata = train, type = "class")
confusion <- table(true = train$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TRAINING data: ", miss , "%.")

#deviance TESTING
Ypred <- predict(fit_dev, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TESTING data: ", miss , "%.")

#gini TRAINING
fit_gini <- tree(form, train, split = "gini")
```

```r
# plot(fit_gini)
# text(fit_gini, pretty = 0)
# fit_gini
# summary(fit_gini)
Ypred <- predict(fit_gini, newdata = train, type = "class")
confusion <- table(true = train$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TRAINING data: ", miss , "%.")

#gini TESTING
Ypred <- predict(fit_gini, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred )
#confusion
miss <- round((100 * (confusion[1,2] +
                          confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate for TESTING data: ", miss , "%.")
```

2.3

```r
fit <- tree(form, data = train, split = "deviance")
trainScore <- rep(0,15)
validScore <- rep(0,15)

#get deviances for different number of leafs
for (i in 2:15) {
    prunedTree <- prune.tree(fit, best = i)
    pred <- predict(prunedTree, newdata = valid, type = "tree")
    trainScore[i] <- deviance(prunedTree)
    validScore[i] <- deviance(pred)
}

#deviance plot
ggplot()+
    geom_line(aes(x = 2:15, y = trainScore[2:15]), col = "blue") +
    geom_point(aes(x = 2:15, y = trainScore[2:15]), col = "blue") +
    geom_line(aes(x = 2:15, y = validScore[2:15]), col = "red") +
    geom_point(aes(x = 2:15, y = validScore[2:15]), col = "red") +
    ggtitle(label = "Deviances vs. Tree Depth" ) +
    ylab("Deviances") +
    xlab("Tree depth")


# plot(x = 2:12, y = trainScore[2:12], type = "b",
#main = "Deviances vs. Leaf count", xlab = "leaf count", ylab = "Deviance", col = "red")
# plot(x = 2:12, y = validScore[2:12], type = "b",
#main = "Deviances vs. Leaf count", xlab = "leaf count", ylab = "Deviance", col = "blue")

#optimal tree
optimalTree <- prune.tree(fit, best = 4)
#summary(optimalTree)
plot(optimalTree)
```

```
text(optimalTree, pretty = 0)
Ypred_opt <- predict(optimalTree, newdata = test, type = "class")
confusion <- table(true = test$good_bad, predicted = Ypred_opt)
confusion
miss <- round((100 * (confusion[1,2] + confusion[2,1]) / sum(confusion)), digits = 2)
paste("Misclassification rate of the OPTIMAL TREE for TESTING data: ", miss , "%.")
```

2.4

```
naive_fit <- naiveBayes(form, data = train)
#training data
Ypred_naive_train <- predict(naive_fit, newdata = train)
confusion_naive_train <- table(true = train$good_bad, predicted = Ypred_naive_train)
confusion_naive_train
miss_naive_train <- round((100 * (confusion_naive_train[1,2] +
                                    confusion_naive_train[2,1]) /
                              sum(confusion_naive_train)), digits = 2)
paste("Naive Bayes misclassification rate for TRAINING data: ", miss_naive_train, "%.")

#test data
Ypred_naive_test <- predict(naive_fit, newdata = test)
confusion_naive_test <- table(true = test$good_bad, predicted = Ypred_naive_test)
confusion_naive_test
miss_naive_test <- round((100 * (confusion_naive_test[1,2] +
                                    confusion_naive_test[2,1]) /
                            sum(confusion_naive_test)), digits = 2)
paste("Naive Bayes misclassification rate for TESTING data: ", miss_naive_test , "%.")
```

2.5

```
prob_naive <- predict(naive_fit, newdata = test, type = "raw")
prob_opt <- predict(optimalTree, newdata = test)
pi <- seq(0.05, 0.95, 0.05)
#initializing
TPR_naive <- vector(length = length(pi))
TPR_opt <- vector(length = length(pi))
FPR_naive <- vector(length = length(pi))
FPR_opt <- vector(length = length(pi))
Yhat_naive <- data.frame(matrix(nrow=length(test$good_bad), ncol= length(pi)))
Yhat_opt <- data.frame(matrix(nrow=length(test$good_bad), ncol= length(pi)))

for (i in 1:length(pi)){
    for(j in 1:length(test$good_bad)){
        Yhat_naive[j,i] <- as.integer(prob_naive[j,2] > pi[i])
        Yhat_opt[j,i] <- as.integer(prob_opt[j,2] > pi[i])
    }
}


for(k in 1:length(pi)){
    #naive bayes
    conf1 <- table(test$good_bad, factor(Yhat_naive[,k], levels = c(0,1)))
```

```
    FPR_naive[k] <- 100*conf1[1,2] / (conf1[1,2] + conf1[1,1])
    TPR_naive[k] <- 100*conf1[2,2] / (conf1[2,1] + conf1[2,2])

    #optimal tree
    conf2<- table(test$good_bad, factor(Yhat_opt[,k], levels = c(0,1)))
    FPR_opt[k] <- 100*conf2[1,2] / (conf2[1,2] + conf2[1,1])
    TPR_opt[k] <- 100*conf2[2,2] / (conf2[2,1] + conf2[2,2])
}

#ROC curves
#naive bayes
ggplot() +
    geom_line(aes(x = FPR_naive, y = TPR_naive), col = "red") +
    geom_point(aes(x = FPR_naive, y = TPR_naive), col = "red") +
    geom_line(aes(x = FPR_opt, y = TPR_opt), col = "blue") +
    geom_point(aes(x = FPR_opt, y = TPR_opt), col = "blue") +
    geom_abline(slope = 1, intercept = 0, col = "black") +
    ggtitle(label = "ROC curves" ) +
    ylab("TPR") +
    xlab("FPR")



# plot(x = FPR_naive, y = TPR_naive, type = "b",
#main = "ROC - Naive Bayes", col = "red")
# #optimal tree
# plot(x = FPR_opt, y = TPR_opt, type = "b",
#main = "ROC - Optimal tree", col = "blue")
```

2.6

```
#formula
form <- good_bad ~ .
#model
naive_fit <- naiveBayes(form, data = train)
#loss matrix
L <- matrix(c(0,10,1,0), nrow = 2)


#TRAINING
prob_naive_train <- predict(naive_fit, newdata = train, type = "raw")
Yhat_train <- vector(length = dim(prob_naive_train)[1])
for (i in 1:dim(prob_naive_train)[1]){
    if(prob_naive_train[i,1]/prob_naive_train[i,2] > L[2,1]/L[1,2]){
        Yhat_train[i] <- 0
    } else {
        Yhat_train[i] <- 1
    }
}

confusion_naive_train <- table(true = train$good_bad, predicted = Yhat_train)
confusion_naive_train
miss_naive_train <- round((100 * (confusion_naive_train[1,2] +
```

```
                                    confusion_naive_train[2,1]) /
                              sum(confusion_naive_train)), digits = 2)
paste("Naive Bayes misclassification rate for TRAINING data: ", miss_naive_train, "%.")


#TESTING
prob_naive_test <- predict(naive_fit, newdata = test, type = "raw")
Yhat_test <- vector(length = dim(prob_naive_test)[1])
for (i in 1:dim(prob_naive_test)[1]){
    if(prob_naive_test[i,1]/prob_naive_test[i,2] > L[2,1]/L[1,2]){
        Yhat_test[i] <- 0
    } else {
        Yhat_test[i] <- 1
    }
}

confusion_naive_test <- table(true = test$good_bad, predicted = Yhat_test)
confusion_naive_test
miss_naive_test <- round((100 * (confusion_naive_test[1,2] +
                                    confusion_naive_test[2,1]) /
                              sum(confusion_naive_test)), digits = 2)
paste("Naive Bayes misclassification rate for TESTING data: ", miss_naive_test, "%.")
```

Assignment 3

```
RNGversion("3.5.1")
data<-read.csv("data/State.csv",header = TRUE,sep=";",dec = ",")
set.seed(12345)
data<-data[order(data$MET),]
```

3.1.

```
tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                       nobs = nrow(data)))
cv.res=cv.tree(tree_mod)
plot(cv.res$size, cv.res$dev, type="b", col="red")
plot(log(cv.res$size), cv.res$dev, type="b", col="red")

prunedTree=prune.tree(tree_mod,best=3)
pred=predict(prunedTree,data=data)

ggplot()+
  geom_line(aes(x=c(1:length(pred)),y=pred),col="red")+
  geom_point(aes(x=c(1:length(data$EX)),y=data$EX),col="blue")
res=data$EX-pred
hist(res)

mean(res)
sd(res)
```

3.2.

```
f=function(data,ind){
  data1=data[ind,]
  tree_mod<-tree(EX~MET,data=data1,control = tree.control(minsize = 8,
                                                          nobs = nrow(data)))
  prunedTree=prune.tree(tree_mod,best=3)
  pred=predict(prunedTree,data)
  return(pred)
}

boot_res=boot(data,f,R=1000)
CE<-envelope(boot_res,level = 0.95)


tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                       nobs = nrow(data)))
prunedTree=prune.tree(tree_mod,best=3)
pred=predict(prunedTree,data)

CI_band=data.frame(upper=CE$point[1,],
                   lower=CE$point[2,],
                   EX=data$EX,MET=data$MET,pred=pred)
ggplot(data=CI_band)+
  geom_point(aes(x=MET,y=EX))+
  geom_line(aes(x=MET,y=pred),col="blue")+
  geom_ribbon(aes(x=MET,ymin=lower,ymax=upper),alpha=0.1)
```

3.3.

```
regression_tree=tree(EX~MET,data=data,control = tree.control(minsize = 8,
                                                             nobs = nrow(data)))
mle=prune.tree(regression_tree,best=3)

rng<-function(data,mle){
  data1=data.frame(EX=data$EX,MET=data$MET)
  data1$EX=rnorm(length(data1$EX),predict(mle,data1),sd=sd(residuals(mle)))
  return(data1)
}
f1<-function(data1){
  new_tree<-tree(EX~MET,data=data1,control = tree.control(minsize = 8,
                                                          nobs = nrow(data)))
  new_prunedTree=prune.tree(new_tree,best=3)
  new_pred<-predict(new_prunedTree,data)
  new_pred<-rnorm(length(new_pred),mean=new_pred,sd=sd(residuals(mle)))
  return(new_pred)
}

boot_res1=boot(data,statistic = f1,R=1000,mle = prunedTree,ran.gen = rng,sim="parametric")

CE<-envelope(boot_res1,level = 0.95)

tree_mod<-tree(EX~MET,data=data,control = tree.control(minsize = 8,nobs = nrow(data)))
prunedTree=prune.tree(tree_mod,best=3)
```

```
pred=predict(prunedTree,data)

CI_band=data.frame(upper=CE$point[1,],
                   lower=CE$point[2,],
                   EX=data$EX,MET=data$MET,pred=pred)
ggplot(data=CI_band)+
  geom_point(aes(x=MET,y=EX))+
  geom_line(aes(x=MET,y=pred),col="blue")+
  geom_ribbon(aes(x=MET,ymin=lower,ymax=upper),alpha=0.1)
```

3.4.

Assignment 4

```
data <- read.csv2("data/NIRSpectra.csv")
# removing viscosity since it is the variable to be predicted
data <- data[,-ncol(data)]
# 4.1 - PCA
res <- prcomp(data)
lambda <- res$sdev^2
prop <- lambda/sum(lambda) * 100
prop <- prop[order(prop, decreasing = TRUE)]
ggplot() + geom_point(aes(x=1:length(prop), y=prop)) +
    ggtitle("Variance Explained by Principal Components") +
    xlab("Principal Components") + ylab("Explained Variance [%]") +
    scale_y_continuous(breaks = seq(0, 110, by=10))
print("Principal Components Contributions")
sprintf("%2.3f", prop[1:16])
ggplot() + geom_point( aes(x=res$x[,1], y=res$x[,2]) ) +
    ggtitle("Scores - PC1 vs PC2") +
    xlab("PC1") + ylab("PC2")
```

4.1.

```
# 4.2 - PCA Trace plots
U <- res$rotation
x <- 1:nrow(res$rotation)
ggplot() + geom_point(aes(x=x, y=U[,1])) + ggtitle("PCA - Traceplot PC1") +
    xlab("Index") + ylab("Load")
ggplot() + geom_point(aes(x=x, y=U[,2])) + ggtitle("PCA - Traceplot PC2") +
    xlab("Index") + ylab("Load")
```

4.2.

```
# 4.3.a - Fast ICA
ica <- fastICA(X = data, n.comp = 2, alg.typ = "parallel")
W_ <- ica$K %*% ica$W
ggplot() + geom_point(aes(x=x, y=W_[,1])) + ggtitle("ICA - Traceplot PC1") +
    xlab("Index") + ylab("W'")
ggplot() + geom_point(aes(x=x, y=W_[,2])) + ggtitle("ICA - Traceplot PC2") +
    xlab("Index") + ylab("W'")
```

```r
# 4.3.b - Scores
ggplot() + geom_point( aes(x=ica$S[,1], y=ica$S[,2]) ) +
    ggtitle("Scores - IC1 vs IC2") +
    xlab("IC1") + ylab("IC2")
```

4.3.