Machine Learning Lab2-block1 Group Report

GroupA10, combined by Weng Hang Wong

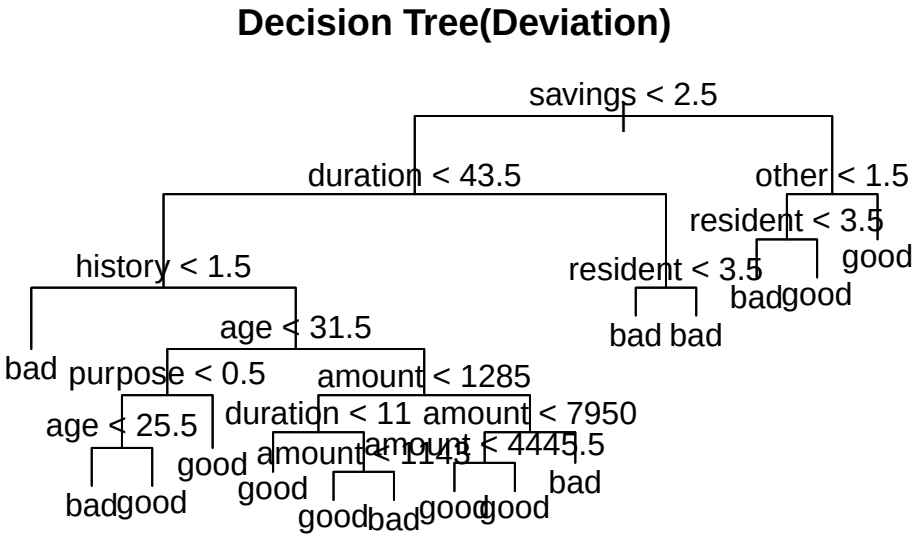
Fengjuan Chen, Jooyoung Lee, Weng Hang Wong

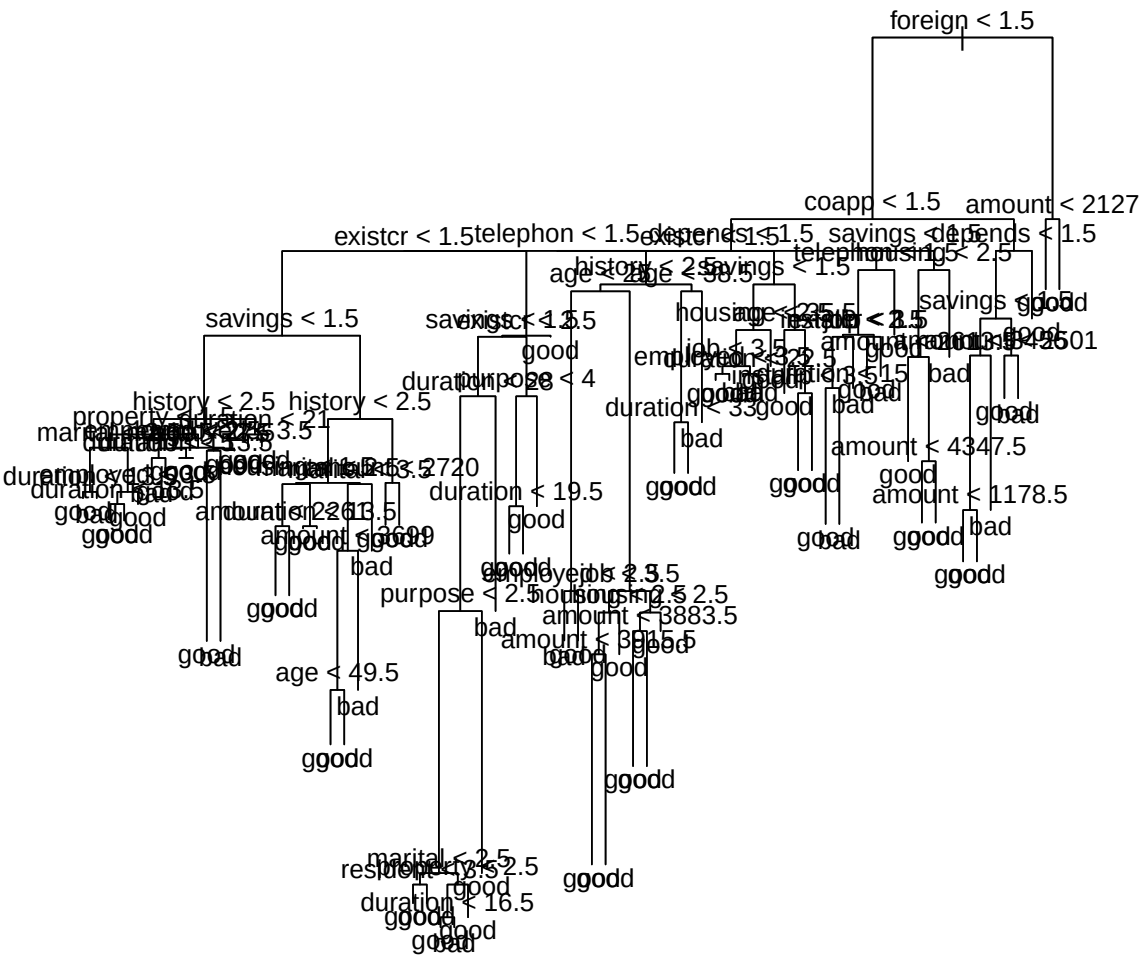
12/08/2019

Assignment2 *Jooyoung Lee*

Importing and Particianing Data

Fitting Decision Trees

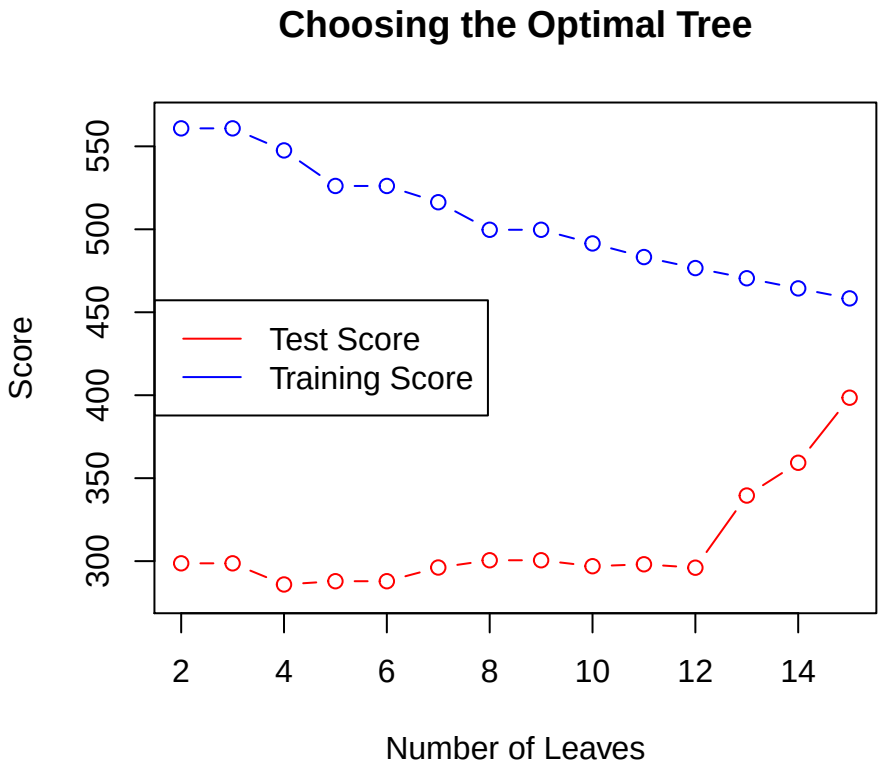





```
##          Training_Data Test_Data
## Deviance         0.212     0.268
## Gini             0.238     0.372
```

Decision trees using “deviance” and “gini” as impurity measure show the above result. Their misclassification rates for the training and test dataset are shown as well. When “deviance” is used as an impurity measure, both misclassification rates were lower. Thus, in the following analysis, “deviance” will be used as impurity measure.

Finding Optimal Tree




```
##   Training  Test
## 1    0.252 0.256
```

Above plot shows the minimum score is obtained when the number of leaves is 4. This means the misclassification with the model is least severe when the target is classified with four end-nodes. When the decision tree model with 4 leaves is fit, the misclassification rates on both training and test data are shown as above, that the rate is higher with training data, but lower with test data than previous analysis.

Naive Bayes Model

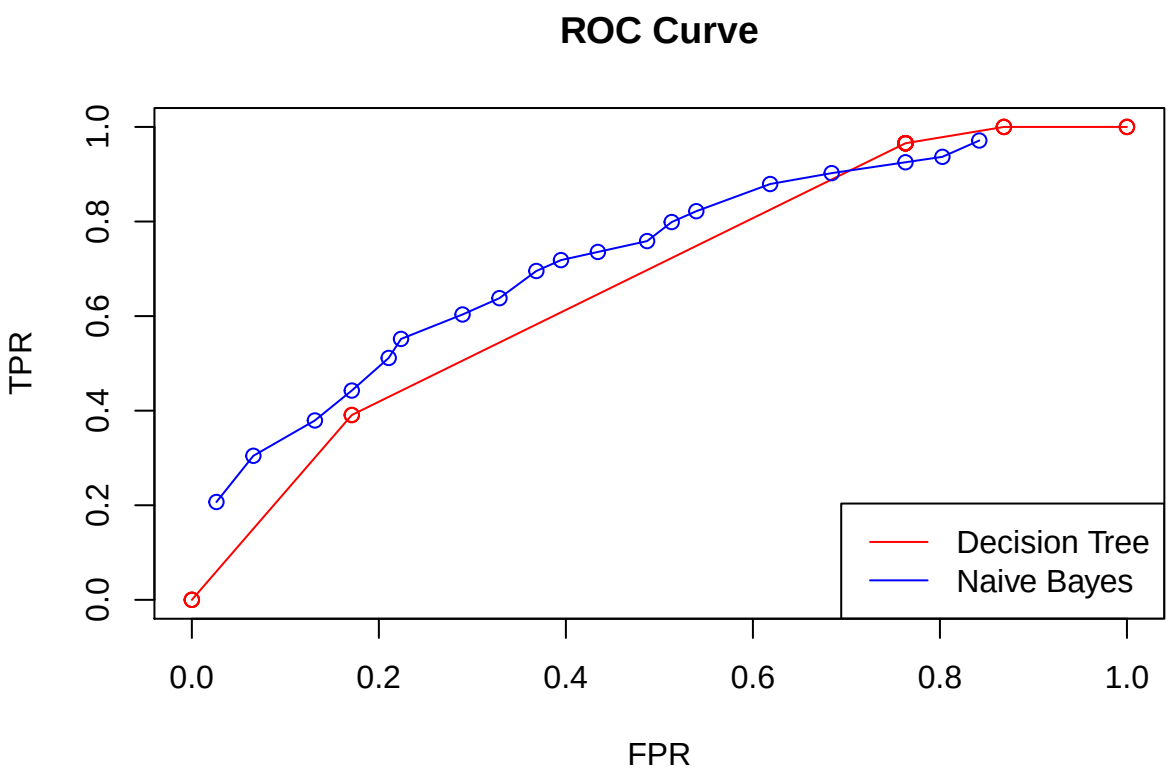
```
##       fit_tr
##        bad good
##   bad   95   52
##   good  98  255

##       fit_te
##        bad good
##   bad   46   30
##   good  49  125

##               Training  Test
## Decision Tree    0.252 0.256
## N_Bayes          0.300 0.316
```

Confusion matrix for training and test data and misclassification rates from different models are shown as above. Compare to the the optimal decision tree, naive bayes shows higher misclassification rates with both training and test data.

Comparing Models with the Given Principle



With the given conditions, at the same level of FPR, predictions based on Naive Bayes model showed higher TPR. If the prediction is made with the given conditions, Naive Bayes method gives better result; Naive Bayes is the better classifier in this case.

Naive Bayes Model with the Given Loss Matrix

```
##   Training  Test
## 1    0.546 0.508
```

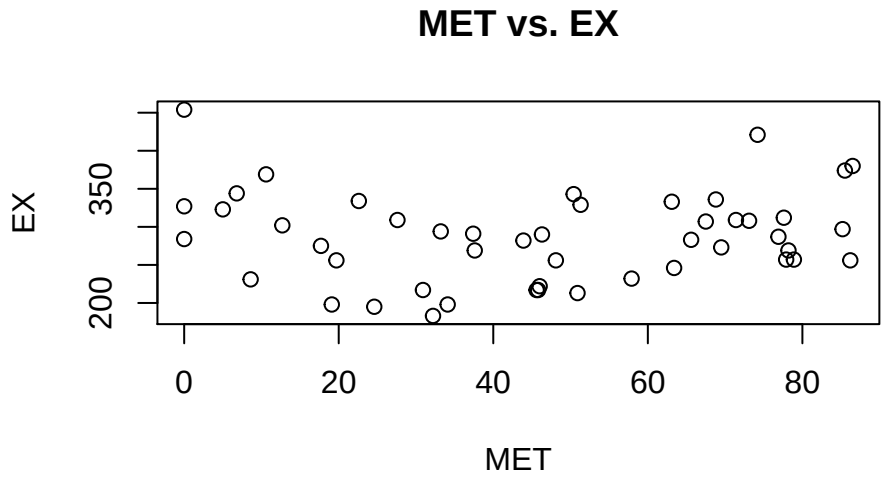
Misclassification rates on with both training and test dataset are very high that exceeds 50%, which are clearly higher than those without the given loss matrix. This is because penalty on penalty on FP was too high that it deteriorated the prediction from the model.

Assignment3 *Fengjuan Chen*

1

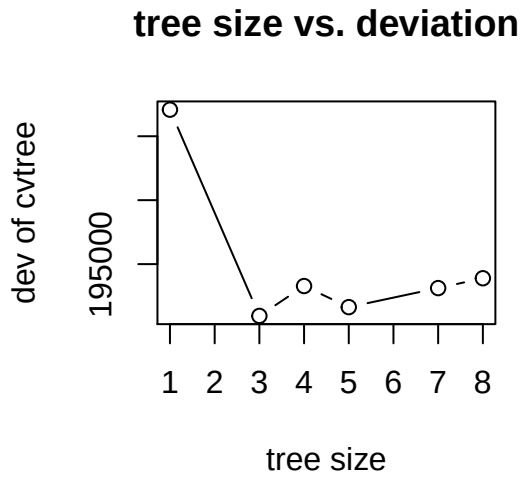
From the scatterplot of MET and EX, we can not find obvious distribution with the data. So, we can try linear regression model or some regression tree model with different parameters. Then decide the optimal model by comparing the training errors and test errors.

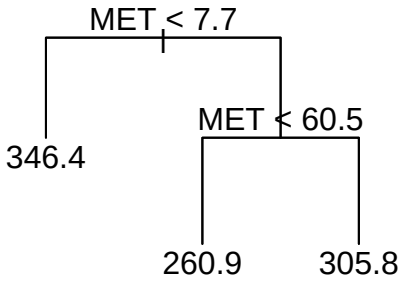
```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-
## uniform 'Rounding' sampler used
```

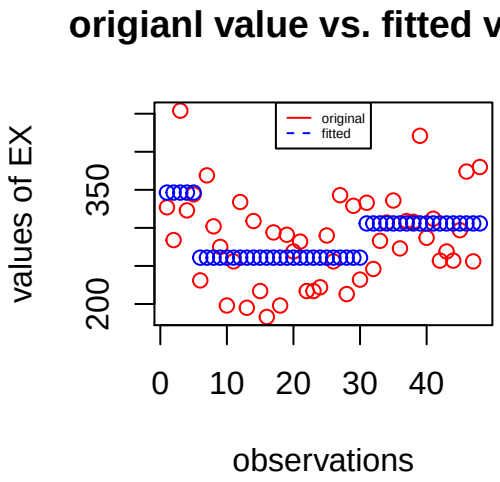


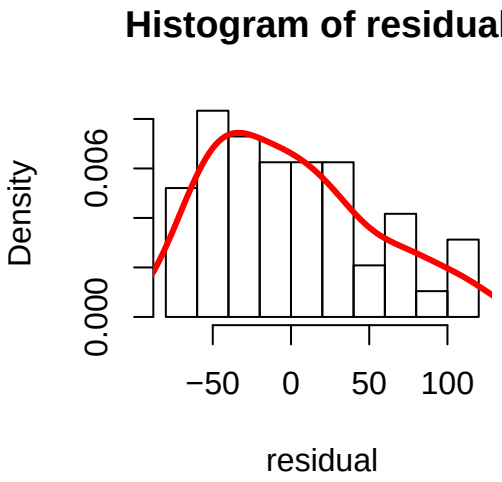
2

From the plot of tree-size against deviations, we choose 3 as the tree size since it has the lowest deciations in all of the tree sizes.





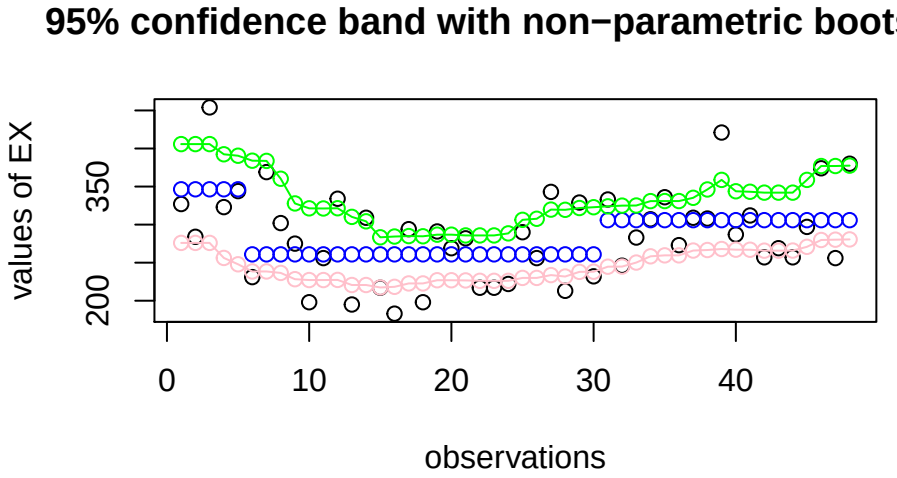




The distribution of the residuals is not normal distribution from the histogram. It shows some skewed tail on right.

From the picture of original data and the fitted data, we find the quality of the fit is not very good. Since in the regression tree, we use the mean of data of leaves in the same branch as the prediction value, it may exist significant difference between the original data and fitted data.

3

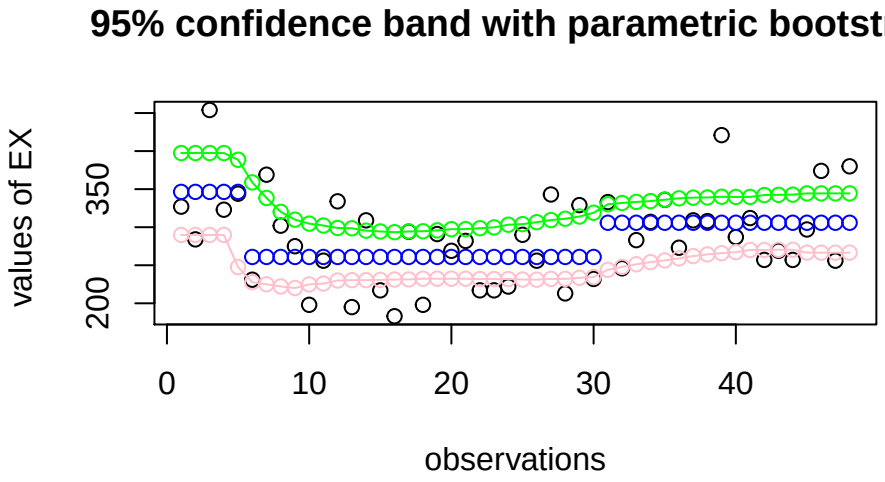


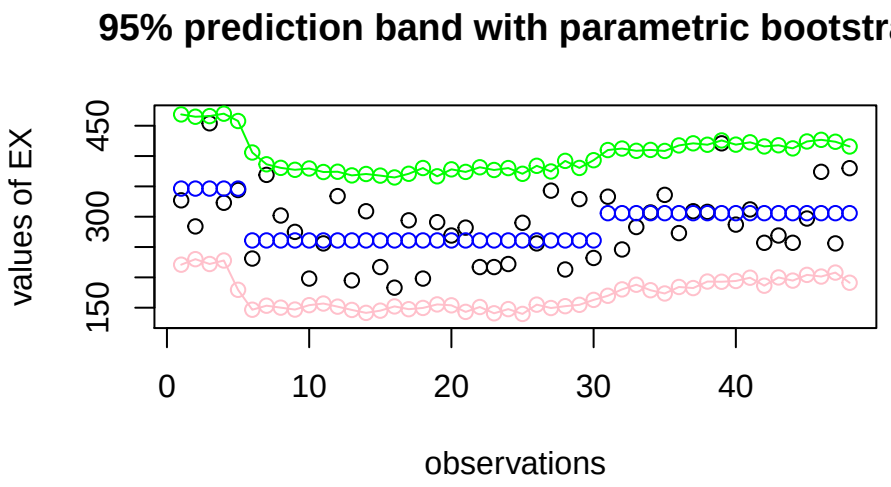
In the picture, the original values of EX are shown by black points, and predicted values are blue points. Then green points (and lines) are used for the upper confidence band and pink points (and lines) for the lower confidence band.

Obviously, the confidence bands are bumpy. That is because the distribution of residual is not normal distribution.

Even the confidence bands are not smooth, the predicted values by regression tree model still locate between the upper and lower bands. That means the results of the regression model in step 2 seem to be reliable.

4





The width of the confidence band in a parametric bootstrap is narrower than width in a non-parametric bootstra. Many predicted values are outside of this condfidence band. From this, the results of the regression model in step 2 seem to be unreliable.

The prediction band looks great. It looks like only 5% of data are outside the prediction band. It should happend because we get the prediction band from the normal distribution with mean of original value and standard deviation of the residual.

5

Because the histogram of residuals is not normally distributed, we suggest that non-parametric bootstrap is more appropriate here.

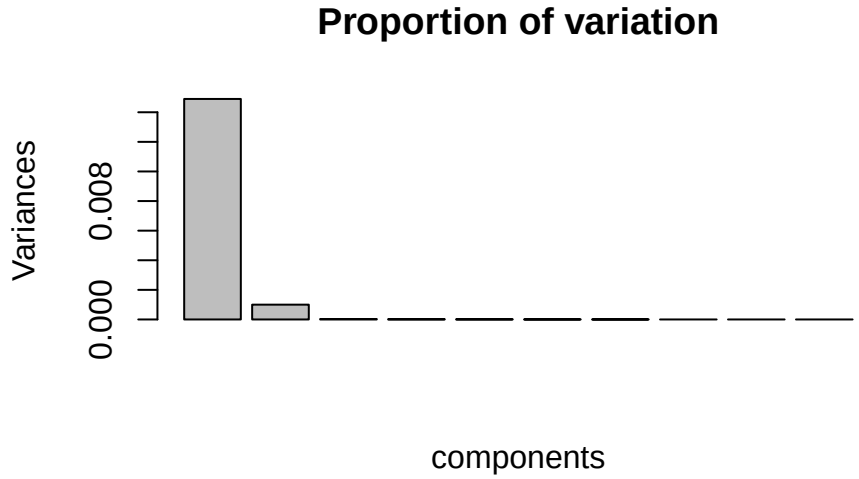
Assignment 4 *Weng Hang Wong*

1

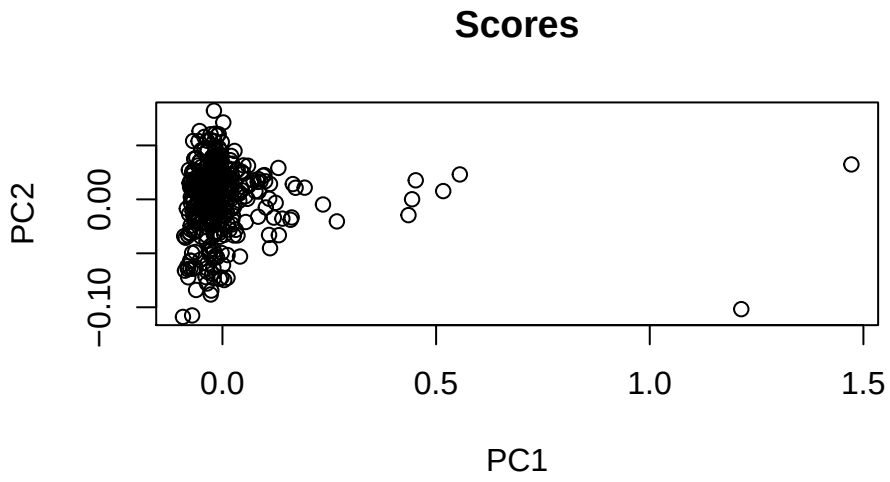
By conducting the PCA, we can see that the variation is explanied by each features. The first two components dominantly explained 93.332%+6.263% of the total variance, which is closed to 100% and they can represent the data.

The belowed graph shows the difference between component 1 and component 2.

```
##   [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"
##   [8] "0.003"  "0.003"  "0.002"  "0.001"  "0.001"  "0.001"  "0.001"
##  [15] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [22] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [29] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [36] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [43] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [50] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [57] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [64] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [71] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [78] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [85] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [92] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
##  [99] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [106] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [113] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
## [120] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
```



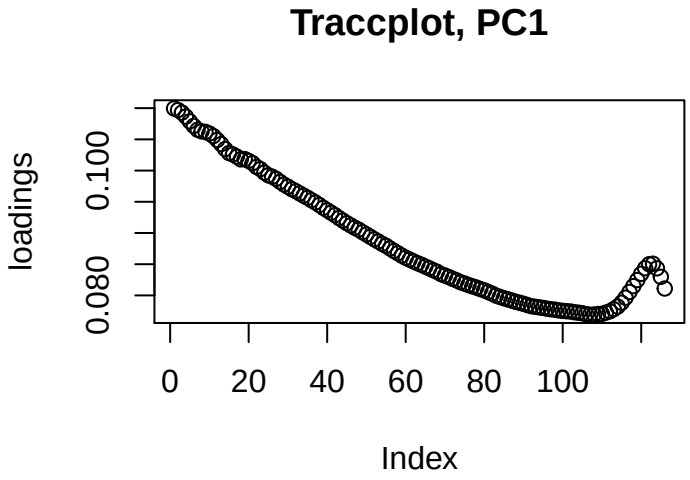
According to the belowed plot, we can see that there are 7 points that are not clustered together with the main cluster,and there are two that is far away from the main cluster, therefore, we consider them as the

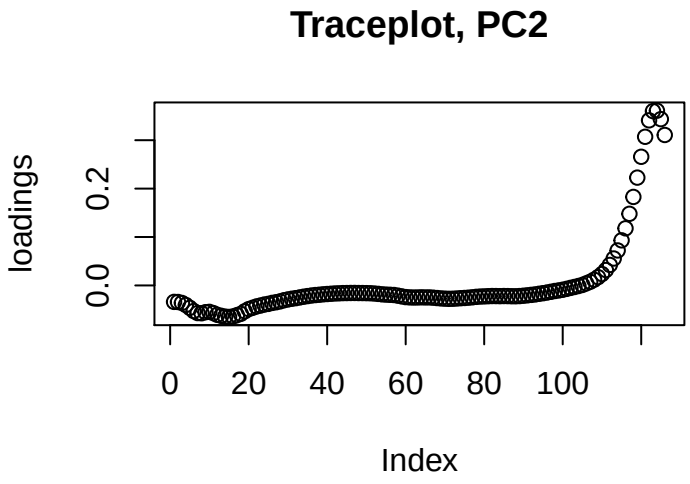


outliners.

2

Trace plots of the loadings of the components are selected as below. The values from PC1 mostly are not closed to 0, but the values from PC2 mostly are closed to 0, we can see that PC1 is explained by the majority features, meanwhile PC2 is explained by only a few features.

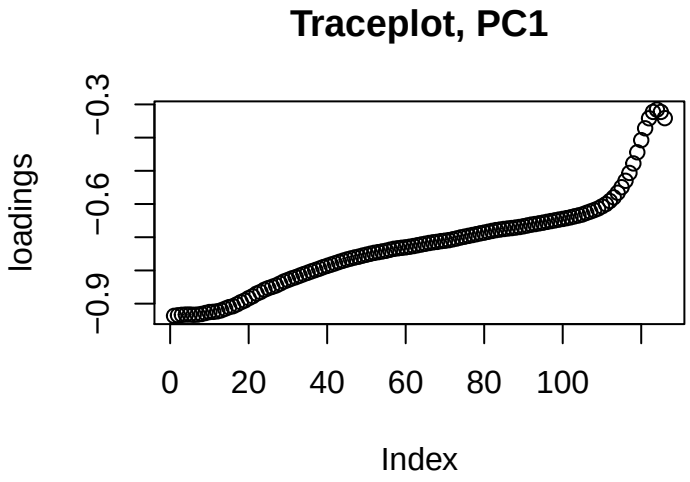


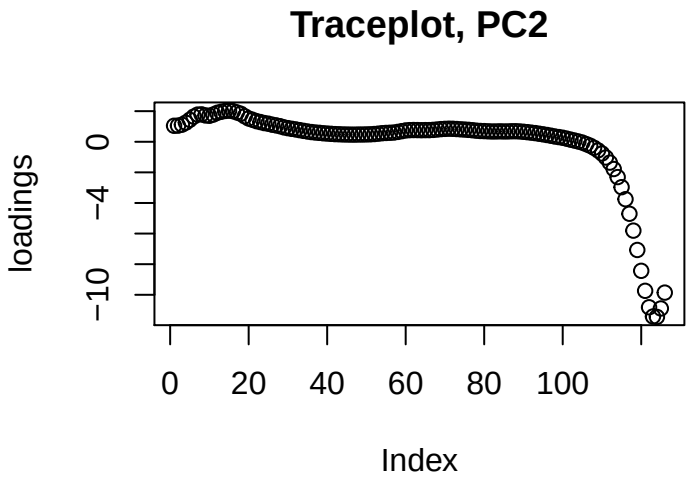


3

a

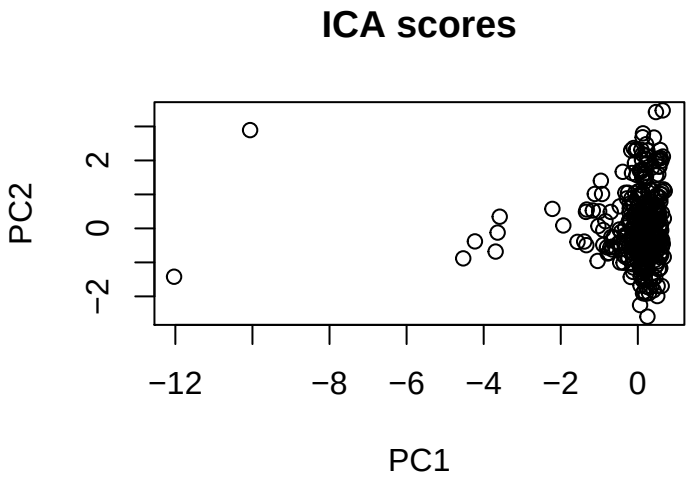
By conducting an ICA method, we can see that result is quite similiar with the plot above. PC1 is explained by most of the feature but PC2 is expained by a few of them.





b

Since PCA does not capture latent factors, eg. rotation invariance, so we can use ICA to choose distribution which is not rotation invariant so that to get unique latent factors. From the below graph, PC1 and PC2 explained quite equally the total variance, which is different from the PCA graph which PC1 is dominantly explained all the variance.



Apprendix

```
## Assignment2
RNGversion("3.5.1")
library(tree)
library(readxl)
```

```r
## Importing and Particianing Data


data <- read_xls("C:/Users/Jooyoung/Documents/ML/Lab2-1/creditscoring.xls")
data$good_bad <- as.factor(data$good_bad)
n <- dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.5))
train <- data[id,]
newdata <- data[-id,]
m <- dim(newdata)[1]
set.seed(12345)
newid <- sample(1:m, floor(m*0.5))
valid <- newdata[newid,]
test <- newdata[-newid,]


## Fitting Decision Trees

fit_dev <- tree(good_bad~., data=train, split="deviance")
plot(fit_dev)
text(fit_dev, pretty=0)
title(main="Decision Tree(Deviation)")
fit_devtr <- predict(fit_dev, newdata=train, type="class")
table_devtr <- as.matrix(table(train$good_bad, fit_devtr))
misclass_devtr <- 1-(sum(diag(table_devtr))/sum(table_devtr))
fit_devte <- predict(fit_dev, newdata = test, type="class")
table_devte <- as.matrix(table(test$good_bad, fit_devte))
misclass_devte <- 1-(sum(diag(table_devte))/sum(table_devte))
misclass_df <- data.frame(Training_Data=misclass_devtr, Test_Data=misclass_devte)

fit_gini <- tree(good_bad~., data=train, split="gini")
plot(fit_gini)
text(fit_gini, pretty=0)
title(main="Decision Tree(Gini)")
fit_ginitr <- predict(fit_gini, newdata=train, type="class")
table_ginitr <- as.matrix(table(train$good_bad, fit_ginitr))
misclass_ginitr <- 1-(sum(diag(table_ginitr))/sum(table_ginitr))
fit_ginite <- predict(fit_gini, newdata = test, type="class")
table_ginite <- as.matrix(table(test$good_bad, fit_ginite))
misclass_ginite <- 1-(sum(diag(table_ginite))/sum(table_ginite))
misclass <- data.frame(Training_Data=misclass_ginitr, Test_Data=misclass_ginite)
misclass_df <- rbind(misclass_df, misclass)
rownames(misclass_df) <- c("Deviance", "Gini")
misclass_df

## Finding Optimal Tree
fit <- tree(good_bad~., data=train, split="deviance")
train_score <- rep(0, 15)
test_score <- rep(0,15)

for (i in 2:15) {
```

```
  pruned_tree <- prune.tree(fit, best=i)
  pred <- predict(pruned_tree, newdata = valid, type="tree")
  train_score[i] <- deviance(pruned_tree)
  test_score[i] <- deviance(pred)
}

plot(x=2:15, y=train_score[2:15], type="b", col="blue", ylim=c(280, 565),
     main="Choosing the Optimal Tree", ylab="Score", xlab="Number of Leaves")
points(2:15, test_score[2:15], type="b", col="red")
legend("left", col=c("red", "blue"), legend=c("Test Score", "Training Score"), lwd=c(1,1))

final_tree <- prune.tree(fit, best=4)
final_predtr <- predict(final_tree, newdata=train, type="class")
final_predte <- predict(final_tree, newdata=test, type="class")
table_finaltr <- as.matrix(table(train$good_bad, final_predtr))
table_finalte <- as.matrix(table(test$good_bad, final_predte))
misclass_finaltr <- 1-(sum(diag(table_finaltr))/sum(table_finaltr))
misclass_finalte <- 1-(sum(diag(table_finalte))/sum(table_finalte))
misclass_finaldf <- data.frame(Training = misclass_finaltr, Test = misclass_finalte)
misclass_finaldf
rownames(misclass_df) <- c("Deviance", "Gini")

## Naive Bayes Model

library(MASS)
library(e1071)

fit <- naiveBayes(good_bad~. , data=train)
fit_tr <- predict(fit, newdata=train, type="class")
table_tr <- table(train$good_bad, fit_tr)
table_tr
table_tr <- as.matrix(table_tr)
misclass_tr <- 1-(sum(diag(table_tr))/sum(table_tr))

fit_te <- predict(fit, newdata=test, type="class")
table_te <- table(test$good_bad, fit_te)
table_te
table_te <- as.matrix(table_te)
misclass_te <- 1-(sum(diag(table_te))/sum(table_te))
new <- data.frame(Training = misclass_tr, Test = misclass_te)
misclass_finaldf <- rbind(misclass_finaldf, new)
rownames(misclass_finaldf) <- c("Decision Tree", "N_Bayes")
misclass_finaldf

## Comparing Models with the Given Principle

prob_tree <- predict(final_tree, newdata=test, type="vector")
prob_nb <- predict(fit, newdata=test, type="raw")
pi <- seq(from=0.05, to=0.95, by=0.05)
ROC_val <- matrix(nrow=length(pi), ncol=5)
colnames(ROC_val) <- c("pi", "tree_tp", "tree_fp", "nb_tp", "nb_fp")
j <- 1
```

```
for (i in pi) {

  #decision tree
  pred_tree <- rep("bad", times=nrow(test))
  goods <- which(prob_tree[,2] > i)
  pred_tree[goods] <- "good"
  mat <- table(test$good_bad, pred_tree)
  if(length(goods) == nrow(test)) {
    tpr_tree <- 1
    fpr_tree <- 1
  }
  else if (length(goods) == 0) {
    tpr_tree <- 0
    fpr_tree <- 0
  }
  else {
    tpr_tree <- mat[2,2]/sum(mat[2,])
    fpr_tree <- mat[1,2]/sum(mat[1,])
  }

  #for naive bayes
  pred_nb <- rep("bad", times=nrow(test))
  goods1 <- which(prob_nb[,2] > i)
  pred_nb[goods1] <- "good"
  mat <- table(test$good_bad, pred_nb)
  if(length(goods1) == nrow(test)) {
    tpr_nb <- 1
    fpr_nb <- 1
  }
  else if (length(goods1) == 0) {
    tpr_nb <- 0
    fpr_nb <- 0
  }
  else {
    tpr_nb <- mat[2,2]/sum(mat[2,])
    fpr_nb <- mat[1,2]/sum(mat[1,])
  }

  ## all calculation is done by now
  ROC_val[j,1] <- i
  ROC_val[j,2] <- tpr_tree
  ROC_val[j,3] <- fpr_tree
  ROC_val[j,4] <- tpr_nb
  ROC_val[j,5] <- fpr_nb
  j <- j+1
}

plot(x=ROC_val[,3], y=ROC_val[,2], main="ROC Curve", xlab="FPR",
     ylab = "TPR", col="red", type="o", xlim=c(0,1), ylim=c(0,1))
points(x=ROC_val[,5], y=ROC_val[,4], col="blue", type="o")
legend("bottomright", legend=c("Decision Tree", "Naive Bayes"),
       col=c("red", "blue"), lty=c(1,1))
```

```r
## Naive Bayes Model with the Given Loss Matrix

##with training data
prob_nb <- predict(fit, newdata=train, type="raw")
res_nb_storetr <- data.frame(Actual=as.character(train$good_bad), stringsAsFactors = FALSE)
resvect <- c()
for(i in 1:nrow(prob_nb)) {
  if ((prob_nb[i,2]/prob_nb[i,1]) > 10) {resele <- "good"}
  else {resele <- "bad"}
  resvect <- append(resvect, resele)
}
res_nb_storetr <- cbind(res_nb_storetr, resvect)
colnames(res_nb_storetr)[2] <- "Train"

##with test data
prob_nb <- predict(fit, newdata=test, type="raw")
res_nb_storete <- data.frame(Actual=as.character(test$good_bad), stringsAsFactors = FALSE)
resvect <- c()
for(i in 1:nrow(prob_nb)) {
  if ((prob_nb[i,2]/prob_nb[i,1]) > 10) {resele <- "good"}
  else {resele <- "bad"}
  resvect <- append(resvect, resele)
}
res_nb_storete <- cbind(res_nb_storete, resvect)
colnames(res_nb_store)[2] <- "Test"

tabletr <- as.matrix(table(res_nb_storetr[,1], res_nb_storetr[,2]))
miscaltr <- 1-(sum(diag(tabletr))/sum(tabletr))
tablete <- as.matrix(table(res_nb_storete[,1], res_nb_storete[,2]))
miscalte <- 1-(sum(diag(tablete))/sum(tablete))
miscal_df <- data.frame(Training=miscaltr, Test=miscalte)

miscal_df

#assignment3

library(tree)
library(knitr)
RNGversion('3.5.1')
set.seed(12345)

data2 <-read.csv2("~/Desktop/State.csv")
data2 <- data2[order(data2$MET),]
plot(data2$MET,data2$EX, xlab = "MET",ylab = "EX",
      main = "MET vs. EX")

## 2

reg_tree <- tree(EX~MET,data = data2,
                 control =tree.control
                 (nrow(data2), minsize = 8))
# use cv.tree to select optimal tree depth
cvreg_tree <- cv.tree(reg_tree)
```

```r
plot(cvreg_tree$size,cvreg_tree$dev,type = "b",
     xlab = "tree size",ylab = "dev of cvtree",
     main = "tree size vs. deviation")
# pruned tree with 3 leaves is the best one
pru_regtree <- prune.tree(reg_tree,best = 3)
plot(pru_regtree)
text(pru_regtree, pretty = 0)
# predict y by pruned tree
    #y_hat <- predict(reg_tree,newdata = data2)
    # predict by original tree
y_hat <- predict(pru_regtree,newdata = data2)
ex_y_residual <- data.frame(original_value=data2$EX,
            fitted_value=y_hat,
            residual=data2$EX-y_hat)
plot(c(1:nrow(data2)),data2$EX,col="red",
       xlab = "observations",ylab = "values of EX",
      main = "origianl value vs. fitted value")
points(c(1:nrow(data2)),y_hat,col="blue")
legend("top",legend = c("original","fitted"),
          col = c("red","blue"),lty = 1:2,cex = 0.4)
ex_residual <- ex_y_residual$residual
hist(ex_residual, prob=TRUE,xlab = "residual",
     main = "Histogram of residual ")
lines(density(ex_y_residual$residual),lwd=3,
      col="red")

## 3

library(boot)

f <- function(data,index){
  data3 <- data[index,]
  reg_tree <- tree(EX~MET,data = data3,
                control =tree.control(nrow(data3),
                                      minsize = 8))
  pru_regtree <- prune.tree(reg_tree,best = 3)
  pre_boot <- predict(pru_regtree,newdata = data2)
  return(pre_boot)
}

res <- boot(data2,f,R=1000)
e <- envelope(res) #compute confidence bands,level=0.95(default)
plot(c(1:nrow(data2)),data2$EX,col="black",
     xlab = "observations",ylab = "values of EX",
     main = "95% confidence band with non-parametric bootstrap")
points(c(1:nrow(data2)),y_hat,col="blue")

# plot 95% confidence bands of regression tree model
# the upper confidence band is stored in e$point[1,]
# the lower confidence band is stored in e$point[2,]
points(c(1:nrow(data2)),e$point[2,],col="pink")
points(c(1:nrow(data2)),e$point[1,],col="green")
lines(c(1:nrow(data2)),e$point[2,],col="pink")
```

```r
lines(c(1:nrow(data2)),e$point[1,],col="green")

## 4

mle <- prune.tree(tree(EX~MET,data = data2,
                       control = tree.control(nrow(data2),
                            minsize = 8)),best = 3)
rng <- function(data,mle){
  data3 <- data.frame(EX=data$EX,MET=data$MET)
  n <- length(data3$EX)
  data3$EX <- rnorm(n, mean=predict(mle,newdata=data3),
               sd(data3$EX-predict(mle,newdata=data3)))
  return(data3)
}

f1 <- function(data3){
  reg_tree <- tree(EX~MET,data = data3,
                   control =tree.control(nrow(data3),
                                         minsize = 8))
  pru_regtree <- prune.tree(reg_tree,best = 3)
  pre_boot <- predict(pru_regtree,newdata = data2)
  return(pre_boot)
}

res2 <- boot(data2,statistic =f1,R=1000,
            mle = mle, ran.gen = rng, sim = "parametric")
e2 <- envelope(res2)
plot(c(1:nrow(data2)),data2$EX,col="black",
     xlab = "observations",ylab = "values of EX ",
     main = "95% confidence band with parametric bootstrap")
points(c(1:nrow(data2)),y_hat,col="blue")
points(c(1:nrow(data2)),e2$point[2,],col="pink")
points(c(1:nrow(data2)),e2$point[1,],col="green")

lines(c(1:nrow(data2)),e2$point[2,],col="pink")
lines(c(1:nrow(data2)),e2$point[1,],col="green")

# prediction interval band
f2 <- function(data3){
  reg_tree <- tree(EX~MET,data = data3,
                   control =tree.control(nrow(data3),
                                         minsize = 8))
  pru_regtree <- prune.tree(reg_tree,best = 3)
  pre_boot <- predict(pru_regtree,newdata = data2)
  n <- length(data2$EX)
  predictEX <- rnorm(n,pre_boot,
                     sd(data2$EX-pre_boot))
   return(predictEX)
}
res3 <- boot(data2, statistic = f2, R=1000 ,mle = mle,
             ran.gen = rng, sim = "parametric")
e3 <- envelope(res3)
#???Warning message:
```

```r
#In envelope(res3) :
#  unable to achieve requested overall error rate

plot(c(1:nrow(data2)),data2$EX,col="black",
     xlab = "observations",ylab = "values of EX",
     ylim = c(130,470),    # the range(e3$point[1,]) and [2,]
     main = "95% prediction band with parametric bootstrap")
points(c(1:nrow(data2)),y_hat,col="blue")
points(c(1:nrow(data2)),e3$point[2,],col="pink")
points(c(1:nrow(data2)),e3$point[1,],col="green")
lines(c(1:nrow(data2)),e3$point[2,],col="pink")
lines(c(1:nrow(data2)),e3$point[1,],col="green")

#assignment4

data <- read.csv2("~/Desktop/NIRSpectra.csv")
set.seed(12345)
#1
data$Viscosity=c()
res=prcomp(data)
lambda = res$sdev^2
#eigenvalues
# lambda

#proportion of variation
sprintf("%2.3f", lambda/sum(lambda)*100)
screeplot(res)

#plot
plot(res$x[,1], res$x[,2], xlab="PC1", ylab="PC2", main="Scores")

#2
U= res$rotation
plot(U[,1], main="Traccplot, PC1", ylab="loadings")
plot(U[,2], main="Traceplot, PC2", ylab="loadings")


#3.

library(fastICA)
w <- fastICA(data,2)
ww <- w$K %*% w$W
plot(ww[,1], main="Traceplot, PC1", ylab="loadings")

plot(ww[,2], main="Traceplot, PC2", ylab="loadings")

plot(w$S[,1], w$S[,2], xlab="PC1", ylab="PC2", main="scores")
```