lab2 block1

Zhixuan_Duan(zhidu838)

12/3/2019

Load packages

```
library(readxl)
library(tree)
library(ggplot2)
library(dplyr)
library(e1071)
library(boot)
library(stats)
library(factoextra)
library(fastICA)
```

Assignment 2. Analysis of credit scoring

1.

Import the data to R and divide into train/validation/test as 50/25/25.

2.

Fit a decision tree by a) Deviance, b) Gini index.

```
##
## Classification tree:
## tree(formula = good_bad ~ ., data = train, split = c("deviance"))
## Variables actually used in tree construction:
## [1] "duration" "history"  "age"      "property" "job"      "savings"
## [7] "amount"
## Number of terminal nodes:  12
## Residual mean deviance:  0.9879 = 476.2 / 482
## Misclassification error rate: 0.247 = 122 / 494
```

```
##
## Classification tree:
## tree(formula = good_bad ~ ., data = train, split = c("gini"))
## Variables actually used in tree construction:
##  [1] "foreign"  "coapp"    "depends"  "existcr"  "telephon" "savings"
##  [7] "history"  "property" "employed" "resident" "marital"  "purpose"
## [13] "duration" "housing"  "installp" "amount"   "job"      "age"
## Number of terminal nodes:  70
## Residual mean deviance:  1.059 = 449.1 / 424
## Misclassification error rate: 0.247 = 122 / 494
```

```
##       pdev
##        bad good
##   bad   31   49
##   good  37  133
```

```
##       pgini
##        bad good
```

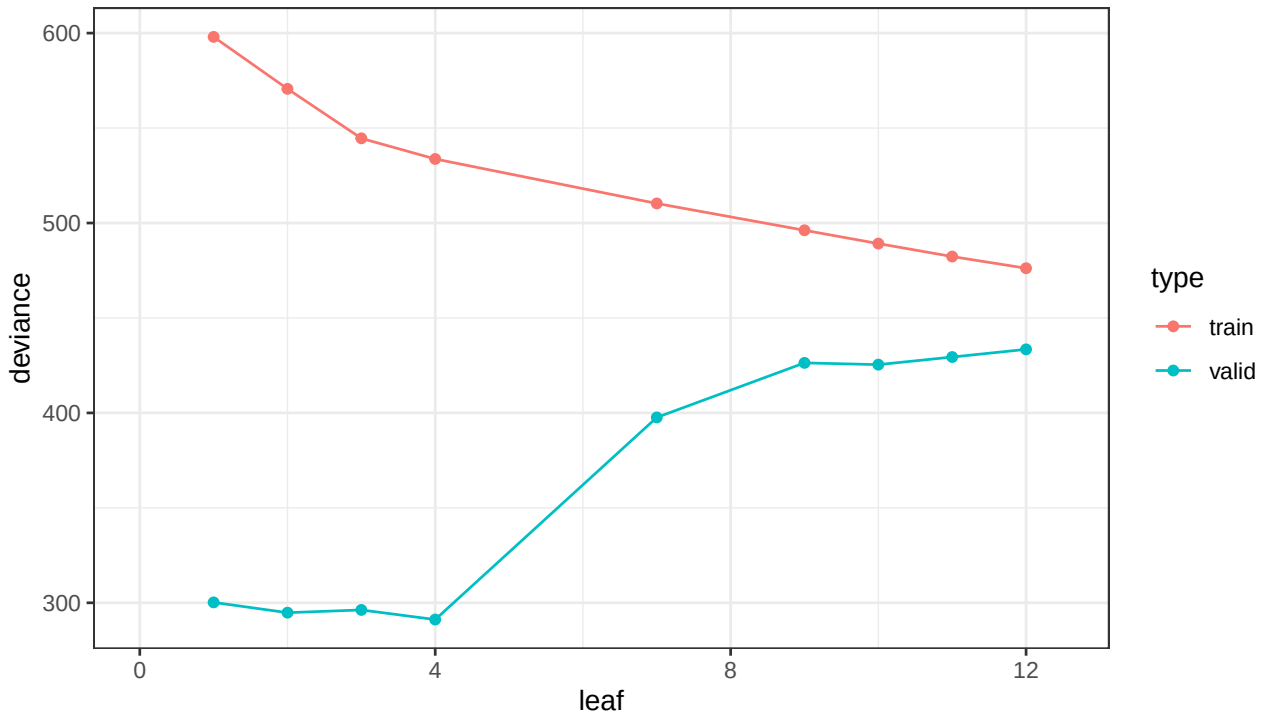
```
##   bad   26   54
##   good  40  130
```

```
## [1] 0.344
```

```
## [1] 0.376
```

The misclassification based on deviance is 0.344, compared to 0.376 of gini index, so we choose Deviance as the measure of impurity.

3.

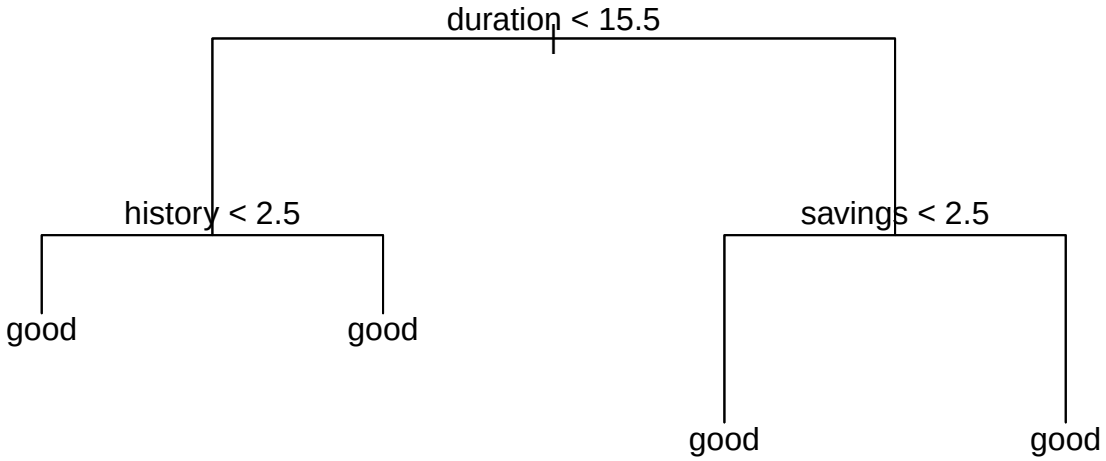


From the plot, the optimal tree depth is when the number of leaves equal to 4.

Based on that, prune the tree to the optimal depth.

```
##
## Classification tree:
## snip.tree(tree = tdev, nodes = c(4L, 7L, 5L, 6L))
## Variables actually used in tree construction:
## [1] "duration" "history"  "savings"
## Number of terminal nodes:  4
## Residual mean deviance:  1.089 = 533.7 / 490
## Misclassification error rate: 0.2935 = 145 / 494
```

From the results, the number of terminal nodes is 4, and it contains 3 variables, which are “duration”, “history” and “saving”.



Here we plot the tree to see its structure, and the misclassification rate for the test data is 0.32.

4.

```
##
## p_nabayes_train bad good
##            bad  103  111
##            good  45  241
```

```
## [1] 0.312
```

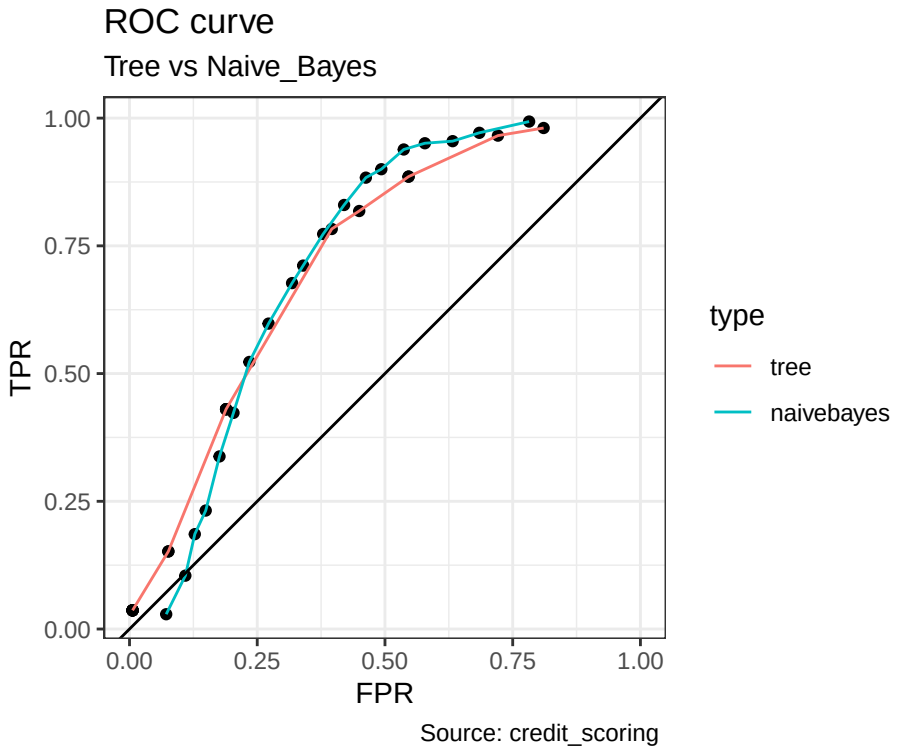
```
##
## p_nabayes_test bad good
##           bad   52   69
##           good  28  101
```

```
## [1] 0.388
```

The misclassification rates for the train data and the test data are 0.312 and 0.388 respectively.

Compared them with the results in 3, we can see the naive_bayes model is better.

5.



From the plot, we can see the naive_bayes curve is closer to the top-left corner, which indicate a better performance.

6.

```
##       pred_good_bad_train
##        bad good
##   bad  145    3
##   good 286   66
```

```
## [1] 0.422
```

```
##       pred_good_bad_test
##        bad good
##   bad   77    3
##   good 136   34
```

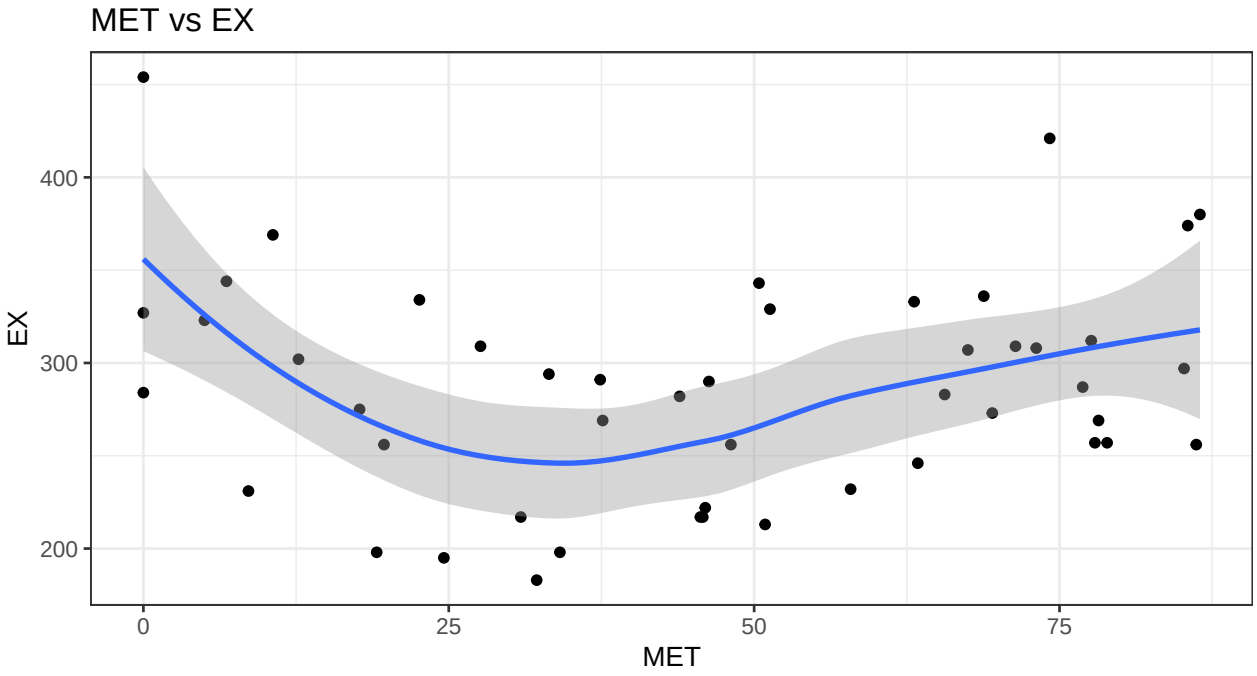
```
## [1] 0.444
```

After adding the loss function to models, the misclassification rate for train and test data are 0.578 and 0.556 respectively.

Assignment 3. Uncertainty estimation

1.

Reorder the data and plot.

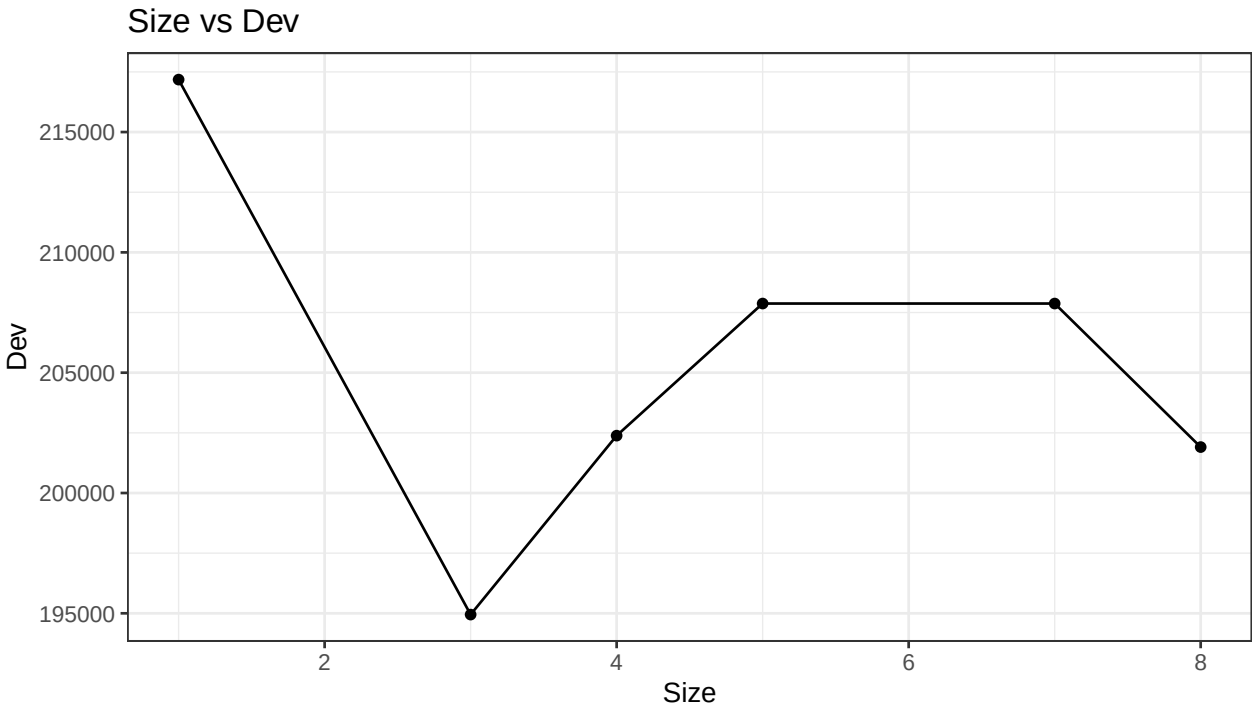


From the plot, this data can be fitted by non-linear model.

2.

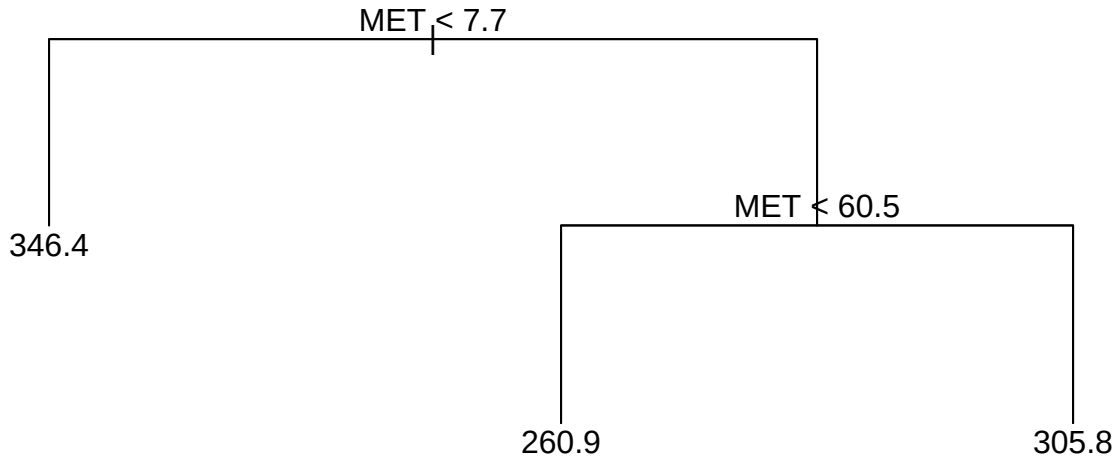
Use tree function and the whole data set, set minimum number of observations(minsize) to 8.

The number of leaves is selected by cross-validation, set K = 10, get the optimal result.

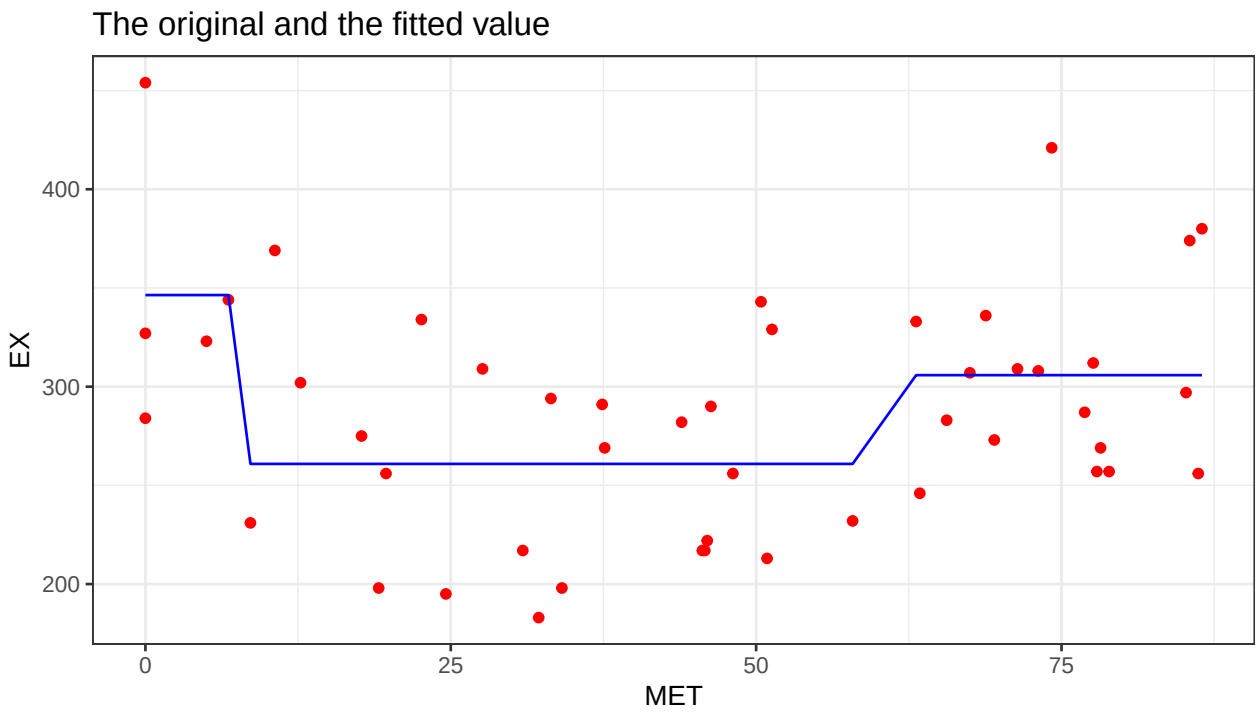


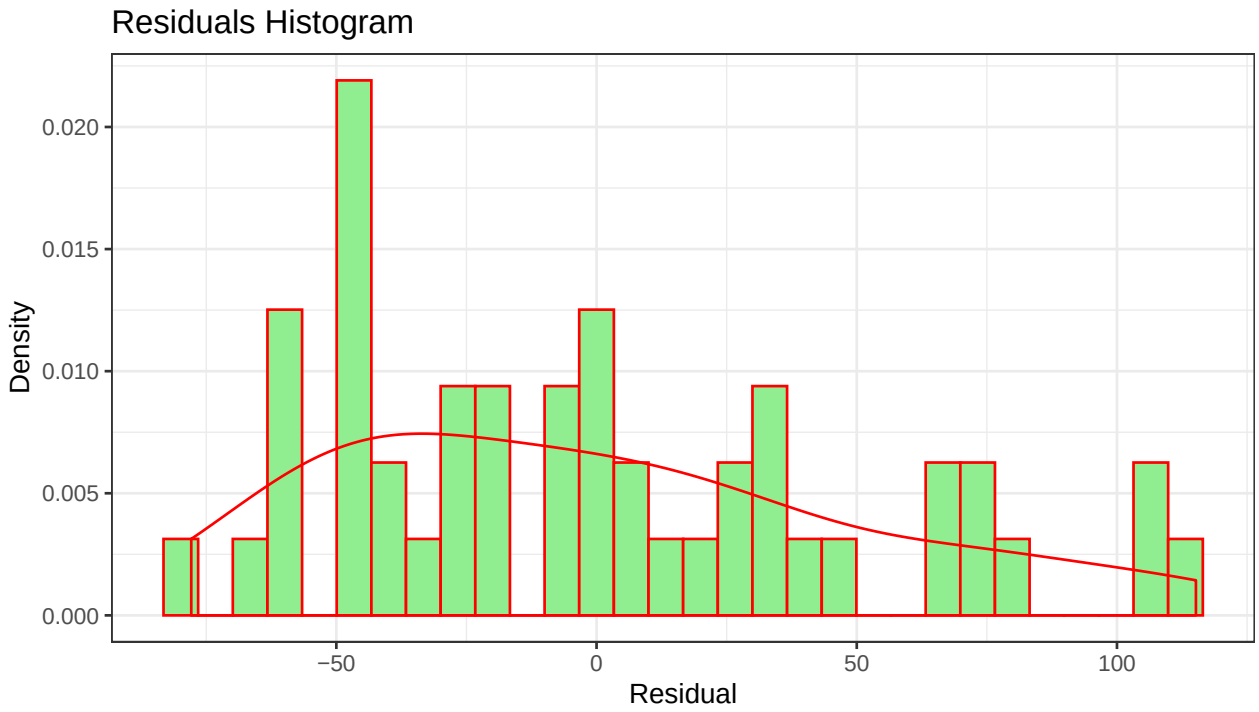
From the plot, the optimal number of leaves(size) is 3.

Plot the pruned tree to see its structure.

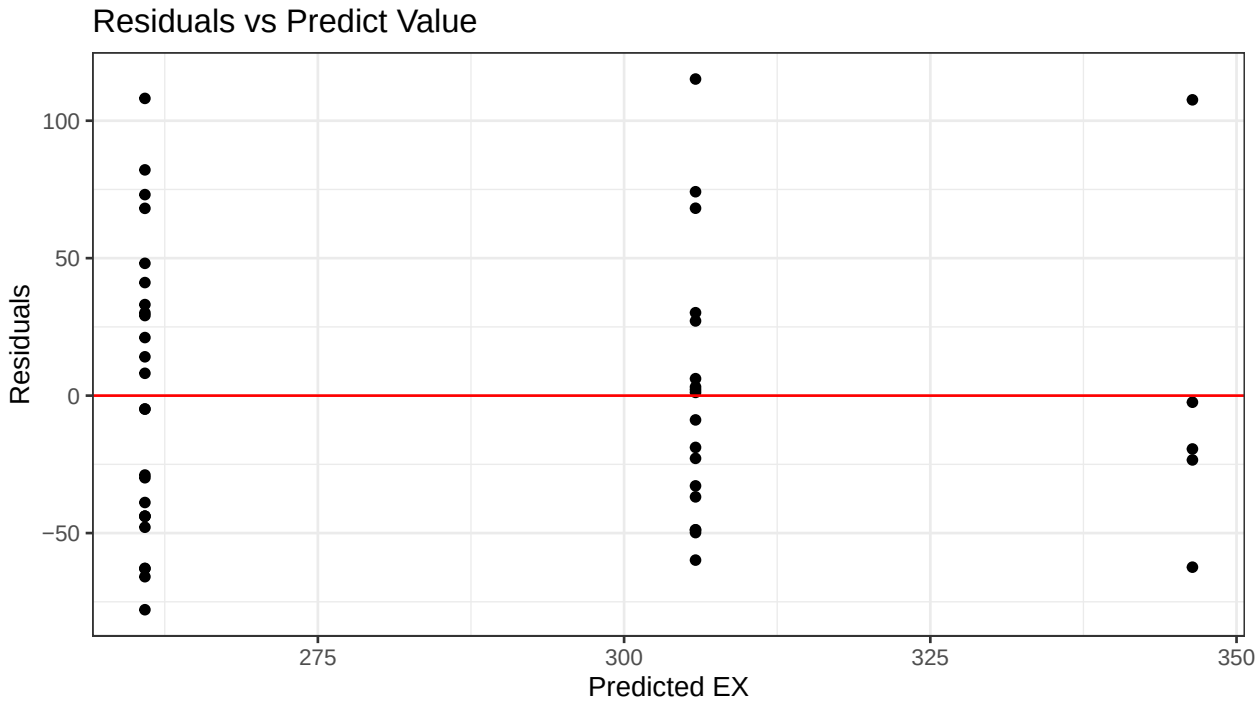


Plot the original and the fitted data and histogram of residuals.



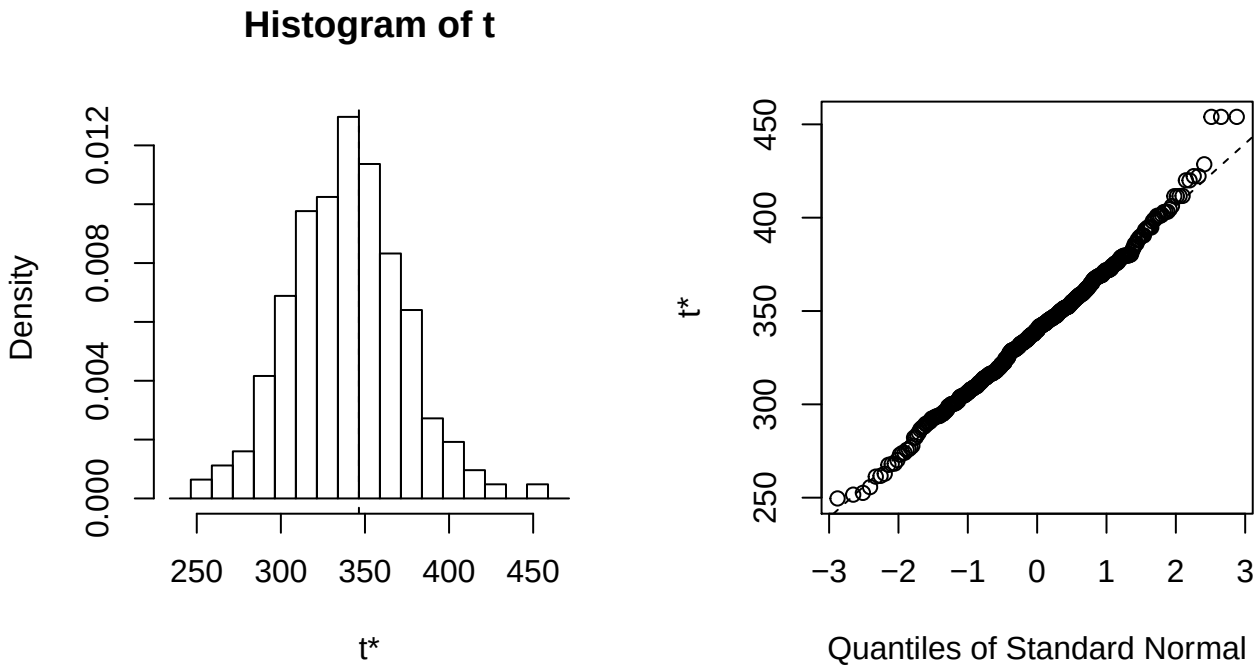


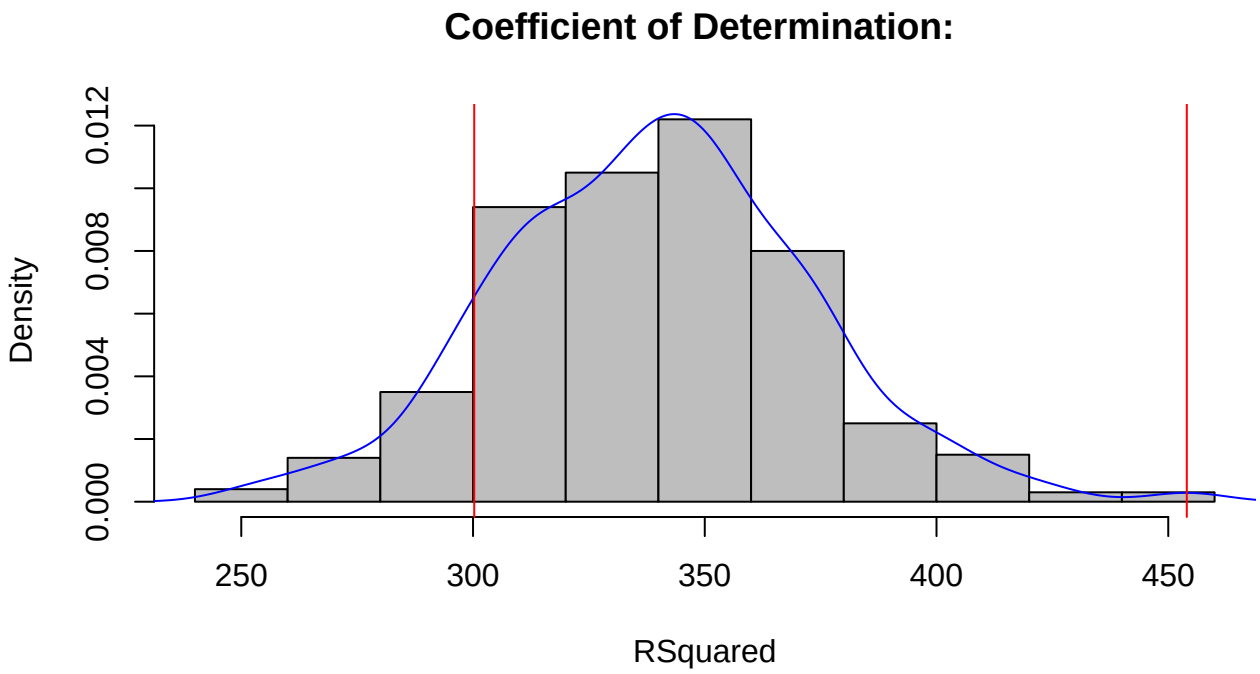
Plot the predict value versus residuals to see its distribution.



From the plot above, we can see there is no bias of this model, the residuals are located both sides of zero.

3.

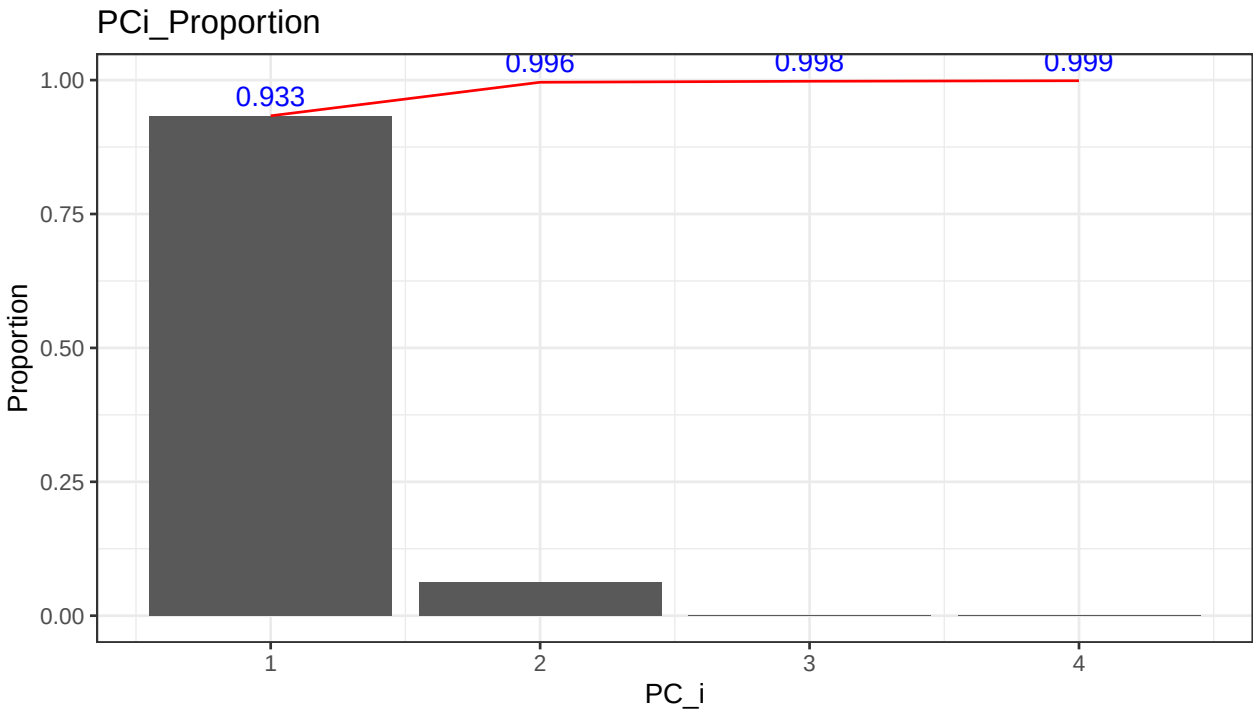



Coefficient of Determination:
Density
0.000
0.004
0.008
0.012
250
300
350
400
450
RSquared

Assignment 4. Principal components

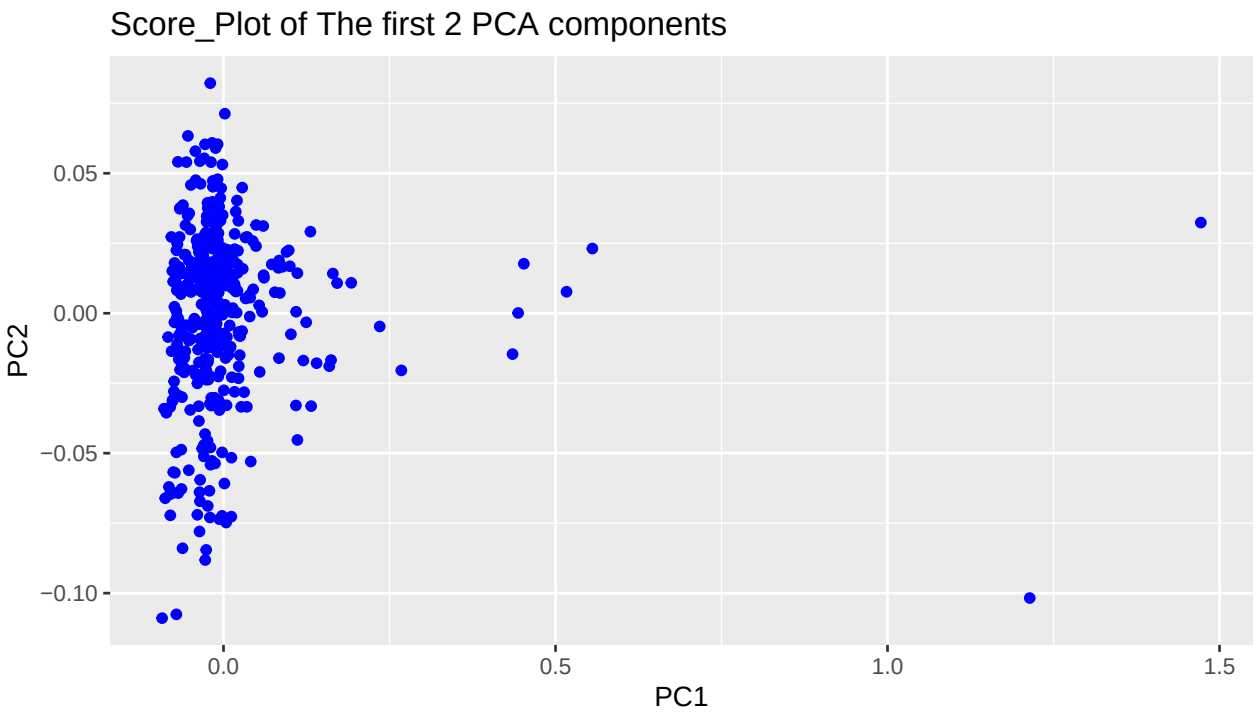
1.

Conduct a standard PCA, and plot.



From the plot above, extract the first 2 components explain over 99% of the total variance.

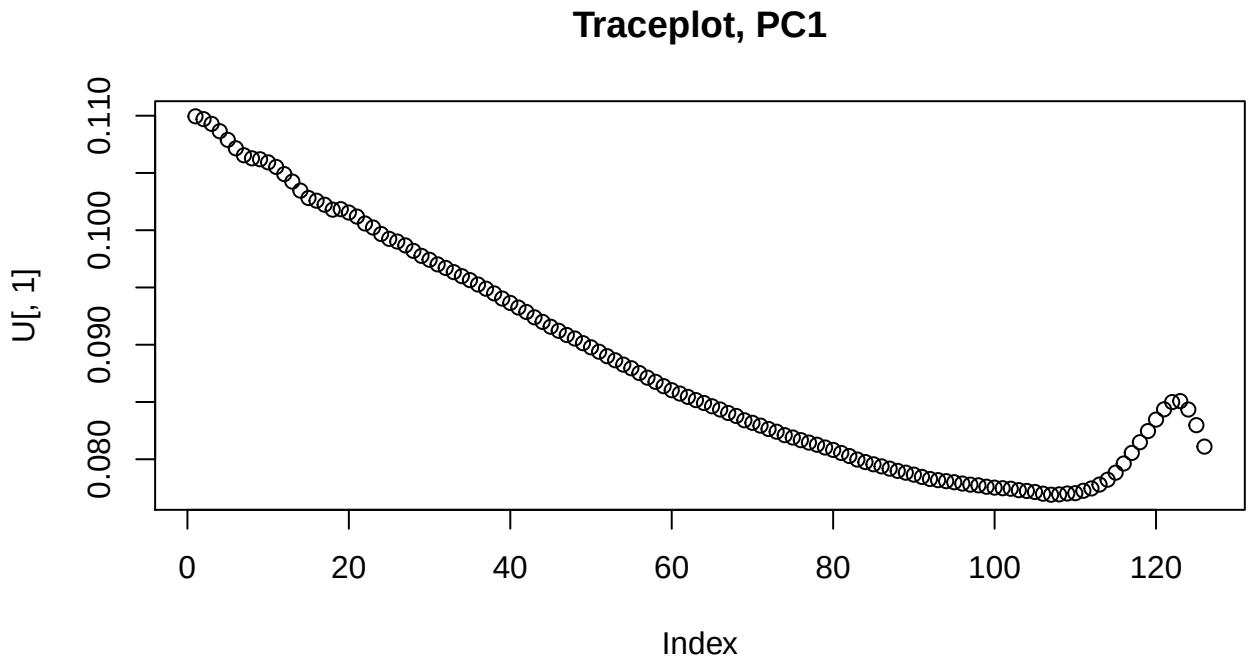
Plot the score_plot of the first 2 PCA components.

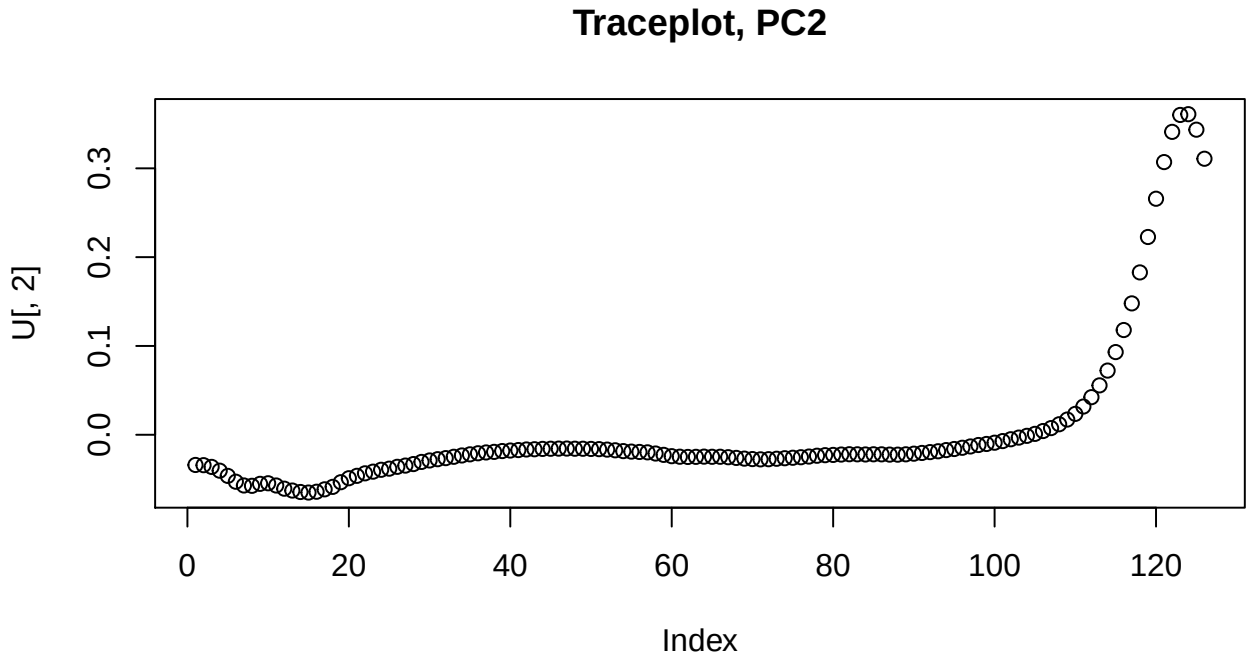


And from the plot, we can see there are some diesel fuels.

2.

Make trace plot.





For PC1, the loadings do not differ very much (max ~ 0.11, mini ~ 0.075).

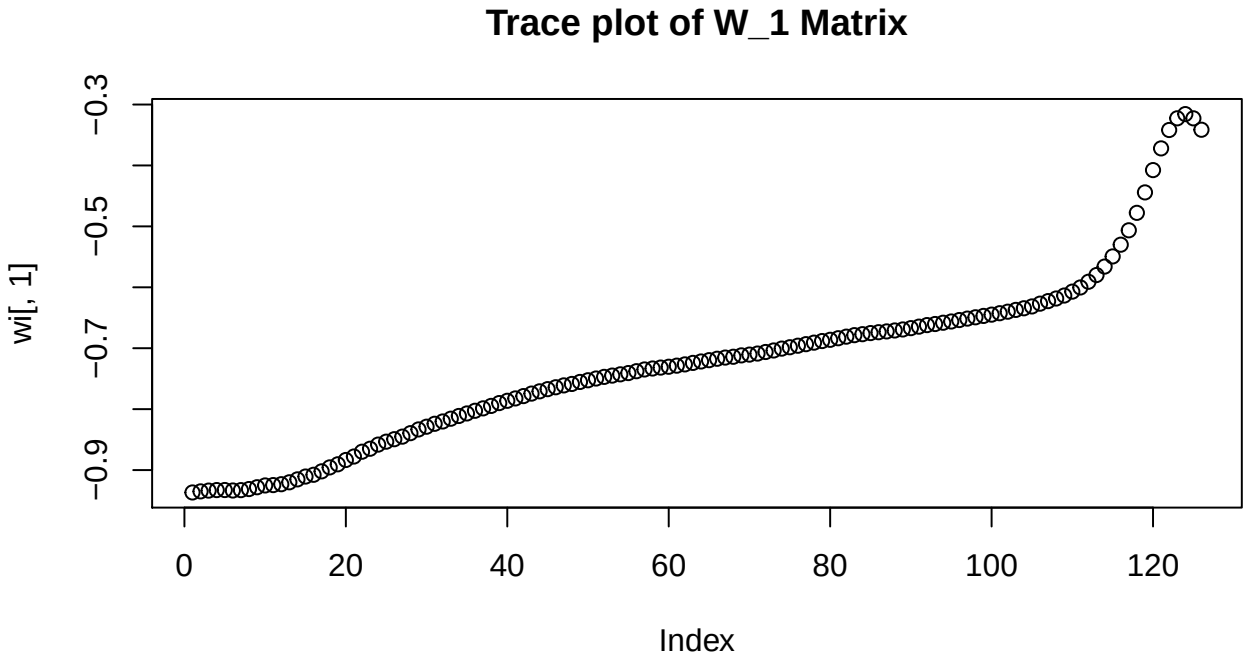
But for PC2 the loadings differ dramatically (max ~ 0.35, mini ~ -0.5), and it's mainly explained by the high index of variables.

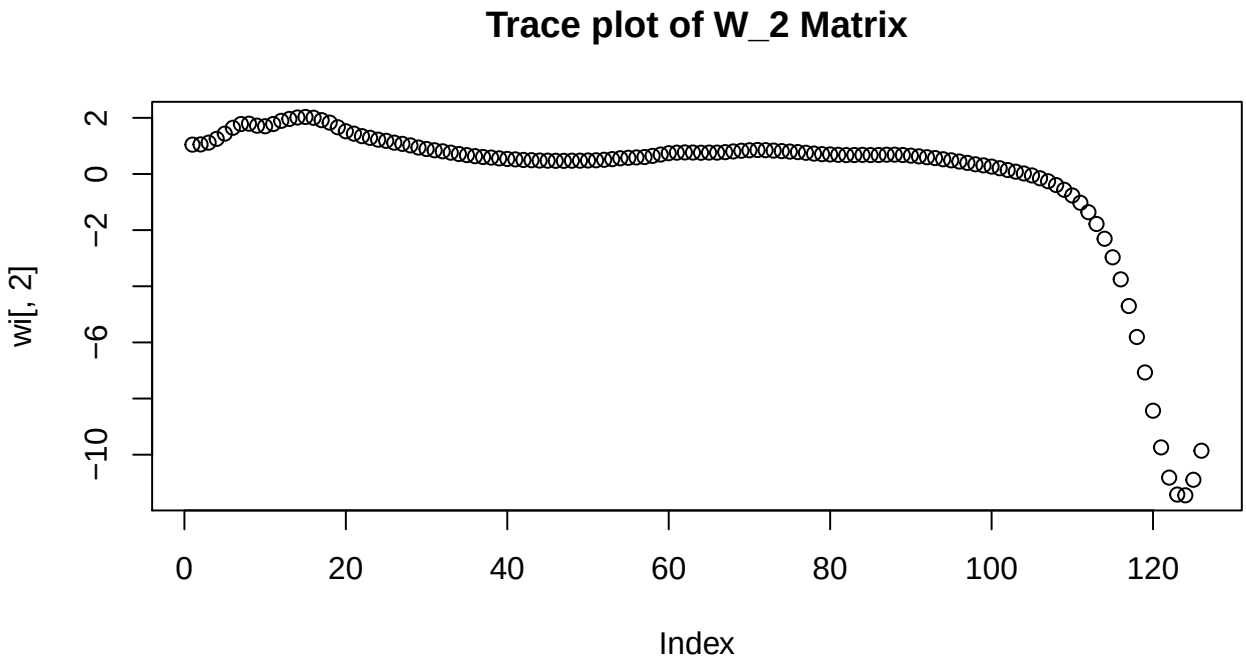
From these two plots, we can see both components are not explained by mainly a few original features.

3.

Perform Independent Component Analysis.

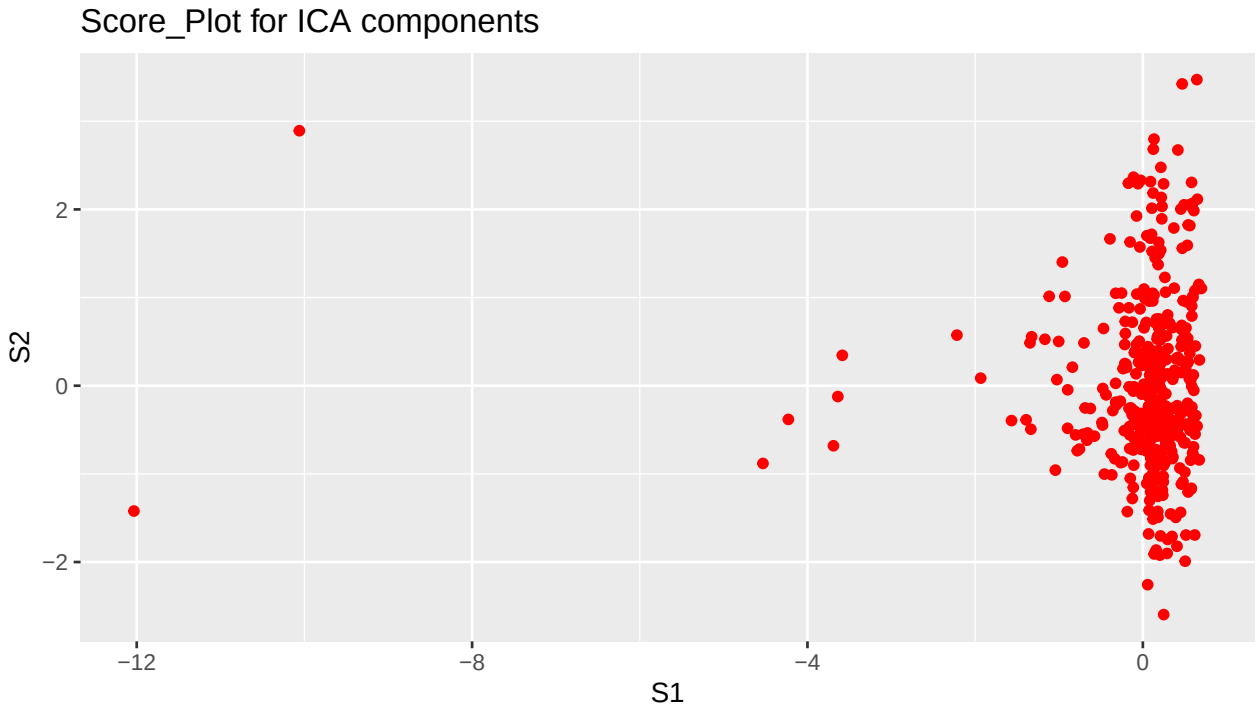
Compute W' and plot.





Compared with the trace plots from step 2, we see that ICA trace plots show a reverse(along the horizonal axis) trend of the PCA trace plots.

The W‘ represents the linear combination of several statistically independent components.



Compared the Score plot from step 1, we find that ICA score plot show a reverse(along the bertical axis) trend of the PCA score plot.

Appendix

```
knitr::opts_chunk$set(echo = FALSE,
                      warning = FALSE,
                      message = FALSE,
                      fig.width = 7,
                      fig.height = 4,
                      fig.align = 'center')
library(readxl)
library(tree)
library(ggplot2)
library(dplyr)
library(e1071)
library(boot)
library(stats)
library(factoextra)
library(fastICA)
# Import dataset
b <- read_excel("/Users/darin/Desktop/creditscoring.xls")
b$good_bad <- as.factor(b$good_bad)

# Divied into 50/25/25
n=dim(b)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=b[id,]

id1=setdiff(1:n, id)
```

```r
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=b[id2,]

id3=setdiff(id1,id2)
test=b[id3,]
tdev <- tree(good_bad~., data=train, split = c("deviance"))
tgini <- tree(good_bad~., data=train, split = c("gini"))
summary(tdev)
summary(tgini)

pdev <- predict(tdev, newdata = test, type = "class")
pgini <- predict(tgini, newdata = test, type = "class")
table(test$good_bad, pdev)
table(test$good_bad, pgini)
a <- ifelse(test$good_bad == pdev, 1, 0)
devmis <- 1-sum(a)/dim(test)[1]
devmis

b <- ifelse(test$good_bad == pgini, 1, 0)
ginimis <- 1-sum(b)/dim(test)[1]
ginimis
set.seed(1234)
tdev <- tree(good_bad ~., data = train, split = c("deviance"))
purnetrain <- prune.tree(tdev, method = "deviance")
purnevalid <- prune.tree(tdev, newdata = valid, method = "deviance")

l_d_t_train <- cbind(purnetrain$size, purnetrain$dev, "train")
l_d_t_valid <- cbind(purnevalid$size, purnevalid$dev, "valid")

ldt <- rbind(l_d_t_train,l_d_t_valid)

colnames(ldt) <- c("leaf", "deviance", "type")

ldt <- as.data.frame(ldt)
ldt$leaf <- as.numeric(as.character(ldt$leaf))
ldt$deviance <- as.numeric(as.character(ldt$deviance))

ggplot(data = ldt,aes(x = leaf, y = deviance, colour = type)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  xlim(c(0,12.5))

prune_tdev <- prune.tree(tdev, best = 4)
summary(prune_tdev)
plot(prune_tdev)
text(prune_tdev)
test_prune <- predict(prune_tdev, newdata = test, type = "class")
c <- table(test$good_bad, test_prune)
c1 <- ifelse(test$good_bad == test_prune, 1, 0)
misrate <- 1-sum(c1)/dim(test)[1]
nabayes <- naiveBayes(good_bad ~ ., data = train)
```

```
p_nabayes_train <- predict(nabayes, newdata = train, type = "class")
p_nabayes_test <- predict(nabayes, newdata = test, type = "class")

d <- table(p_nabayes_train, train$good_bad)
dtemp <- ifelse(p_nabayes_train == train$good_bad, 1, 0)
nabayes_train_misrate <- 1-sum(dtemp)/nrow(train)
d
nabayes_train_misrate

e <- table(p_nabayes_test, test$good_bad)
etemp <- ifelse(p_nabayes_test == test$good_bad, 1, 0)
nabayes_test_misrate <- 1-sum(etemp)/nrow(test)
e
nabayes_test_misrate
set.seed(12345)

t <- tree(good_bad ~ ., data=train, split = c("deviance"))
nb <- naiveBayes(good_bad ~ ., data = train)

pre_t <- predict(t, newdata = test)
pre_nb <- predict(nb, newdata = test, type = "raw")

tprob <- as.data.frame(cbind(pre_t, as.character(test$good_bad), 'tree'))
nbprob <- as.data.frame(cbind(pre_nb, as.character(test$good_bad), 'naivebayes'))

tnb_prob <- rbind(tprob, nbprob)
names(tnb_prob) <- c("good_prob", "bad_prob", "true_good_bad", "type")

tnb_prob$good_prob <- as.numeric(as.character(tnb_prob$good_prob))
tnb_prob$bad_prob <- as.numeric(as.character(tnb_prob$bad_prob))

tempdataset <- tnb_prob[,c(1,3,4)]
tempdataset <- as.data.frame(tempdataset)


temp <- NULL
for (i in seq(0.05, 0.95, 0.05)) {
  tempdataset$threshold <- i
  tempdataset$pred_good_bad <- as.factor(ifelse(tnb_prob$good_prob > i, "good", "bad"))
  temp <- rbind(temp,tempdataset)
}

# x  FPR: FP/(FP + TN)
# y  TPR: TP/(TP + FN)
temp1 <- temp %>%
  group_by(type, threshold) %>%
  summarize(
    fpr = (sum(true_good_bad == "bad" & pred_good_bad == "good") /
                  (sum(true_good_bad == "bad" & pred_good_bad == "good") +
                   sum(true_good_bad == "good" & pred_good_bad == "bad")
                   )),
            tpr = (sum(true_good_bad == "good" & pred_good_bad == "good")/
                   (sum(true_good_bad == "good" & pred_good_bad == "good") +
```

```
                    sum(true_good_bad == "bad" & pred_good_bad == "bad")
                    ))
          )

ggplot(temp1, aes(x = fpr, y = tpr)) +
  geom_point() +
  geom_line(aes(colour = type)) +
  geom_abline(intercept = 0, slope = 1) +
  theme_bw() +
  coord_equal() +
  xlim(c(0,1)) +
  labs(title = "ROC curve",
       subtitle = "Tree vs Naive_Bayes",
       x = "FPR",
       y = "TPR",
       caption = "Source: credit_scoring ")
set.seed(12345)
nabayes <- naiveBayes(good_bad ~ ., data = train)

pred_train <- as.data.frame(predict(nabayes, newdata = train, type = "raw"))
pred_good_bad_train <- ifelse(pred_train$good > 10 * pred_train$bad, "good", "bad")
table(train$good_bad,pred_good_bad_train)
temp_train <- ifelse(train$good_bad == pred_good_bad_train, 1, 0)
nabayes_train_acc <- sum(temp_train)/nrow(train)
nabayes_train_acc

pred_test <- as.data.frame(predict(nabayes, newdata = test, type = "raw"))
pred_good_bad_test <- ifelse(pred_test$good > 10 * pred_test$bad, "good", "bad")
table(test$good_bad,pred_good_bad_test)
temp_test <- ifelse(test$good_bad == pred_good_bad_test, 1, 0)
nabayes_test_acc <- sum(temp_test)/nrow(test)
nabayes_test_acc
a <- read.csv2("/Users/darin/Desktop/State.csv")
# reorder the data to the increase of MET
a <- a[order(a$MET),]
# plot EX versus MET
ggplot(data = a, aes(x=MET, y = EX)) +
  geom_point() +
  geom_smooth() +
  theme_bw() +
  labs(title = "MET vs EX",
       caption = "Source: State")
# use the entire data and set minsize to 8 to fit a tree
set.seed(12345)
t <- tree(data = a, EX~MET, control = tree.control(nobs=nrow(a),
                                                   minsize = 8))
# use cross_validation method to get the best fit model
a_cvtree <- cv.tree(t, K=10)
temp <- as.data.frame(cbind(a_cvtree$size,a_cvtree$dev))
# plot the relationship between size and dev
ggplot(temp, aes(x = V1, y = V2)) +
  geom_point() +
  geom_line() +
```

```
  theme_bw() +
  labs(title = "Size vs Dev",
       x = "Size",
       y = "Dev")
# choose best to 3
prune_t <- prune.tree(t, best = 3)
plot(prune_t)
text(prune_t)
# plot the original and the fitted data
pred_t <- predict(prune_t, newdata = a)
temp1 <- as.data.frame(cbind(pred_t, a$EX, a$MET))
names(temp1) <- c("pred_t", "EX", "MET")
ggplot(temp1, aes(x = MET, y = EX)) +
  geom_point(colour = "red") +
  geom_line(aes(x = MET, y = pred_t), colour = "blue") +
  theme_bw() +
  labs(title = "The original and the fitted value")

# compute the residuals and plot a histgram
res <- a$EX - pred_t
temp2 <- cbind(temp1, res)

ggplot(temp2,aes(res)) +
  geom_histogram(bins = 30,
                 aes(y=..density..),
                 col = "red",
                 fill = "lightgreen") +
  geom_density(col = "red") +
  theme_bw() +
  labs(title = "Residuals Histogram",
       x = "Residual",
       y = "Density")
# plot the predict value versus residuals
ggplot(temp2, aes(x = pred_t, y = res)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 0, colour = "red") +
  theme_bw() +
  labs(title = "Residuals vs Predict Value",
       x = "Predicted EX",
       y = "Residuals")
bootstrap_pred <- function(data, index){
  data1 <- data[index,]
  model <- tree(data = data1, EX~MET, control = tree.control(nobs=nrow(data1),
                                      minsize = 8))
  prune_t <- prune.tree(model, best = 3)
  pred_ex <- predict(prune_t, newdata = data)
  return(pred_ex)
}

result <- boot(data = a, statistic = bootstrap_pred, R = 500)
plot(result)
confidence_interval = boot.ci(result, conf = 0.95, type = 'bca')
ci_H <- confidence_interval$bca[ , c(4, 5)]
```

```r
hist(result$t[,1], main = 'Coefficient of Determination: ', xlab = 'RSquared', col = 'grey', prob = T)
lines(density(result$t[,1]), col = 'blue')
abline(v = ci_H, col = 'red')
set.seed(12345)
# create dataset for pca
nir_data <- read.csv2("/Users/darin/Desktop/NIRSpectra.csv")
pca_data <- nir_data[,c(1:126)]
pca <- prcomp(pca_data)
# compute lambda and draw scree plot
lambda = pca$sdev^2
pca_data1 <- cbind(pci = 1:4,
      pci_proportion = lambda[1:4]/sum(lambda),
      cumm = cumsum(lambda[1:4]/sum(lambda)))
plotData = as.data.frame(pca_data1)

ggplot(data = plotData) +
  geom_col(aes(x = pci, y = pci_proportion)) +
  geom_line(aes(x = pci, y = cumm), colour = "red") +
  geom_text(aes(x = pci, y = cumm, label = round(cumm,3)), colour = "blue", vjust = -0.5) +
  theme_bw() +
  labs(title = "PCi_Proportion",
       x = "PC_i",
       y = "Proportion")
pca_result_data = as.data.frame(cbind(pc1 = pca$x[,1],
                                      pc2 = pca$x[,2]))

ggplot(data = pca_result_data, aes(x = pc1, y = pc2)) +
  geom_point(col = "blue") +
  labs(title = "Score_Plot of The first 2 PCA components",
       x = "PC1",
       y = "PC2")
set.seed(12345)

U <- pca$rotation
plot(U[,1], main="Traceplot, PC1")
plot(U[,2], main="Traceplot, PC2")
set.seed(12345)
# X: a linear combination of non-Gaussian (independent) components
# X = SA
#     S: columns of S contain the independent components
#      A: a linear mixing matrix
#      W: Estimating an un-mixing matrix W where XW= S.

X <- as.matrix(pca_data)

ica <- fastICA(X, 2)

# X -> pre-processed data matrix
# K -> pre-whitening matrix that projects data onto the first n.compprincipal components.
# W -> estimated un-mixing matrix
# A -> estimated mixing matrix
# S -> estimated source matrix
```

```r
wi <- ica$K %*% ica$W
plot(wi[,1], main = "Trace plot of W_1 Matrix")
plot(wi[,2], main = "Trace plot of W_2 Matrix")

ica_data <- as.data.frame(cbind(s1 <- ica$S[,1],
                                s2 <- ica$S[,2],
                                Viscosity <- nir_data$Viscosity))
ggplot(data = ica_data, aes(x = s1, y = s2)) +
  geom_point(col = "red") +
  labs(title = "Score_Plot for ICA components",
       x = "S1",
       y = "S2")
```