Assignment3 MAST90138

Muhan Guan 1407870

2023-10-22

Tutor's name: Shangyu Chen

Time of Tutorial class: Thursday 10AM

Problem 1

(a)

(a1)

All in all**, 134321** parameters need to be estimated for QD classifier, based on Bayes theorem and distributional assumption about $\mathbf{X}$ it uses when calculating probability.

Specifically, in QD ,when a new individual in test data with $\mathbf{X}_i = (\mathbf{X}_{i1}, ..., \mathbf{X}_{ip})$ is more likely to be classified as group 1 than group 2 if below equation can be satisfied :

$$-log(\hat{\pi}_1) + log\{det(\hat{\Sigma}_1)\} + (\mathbf{X}_i - \hat{\mu}_1)^T \hat{\Sigma}_1^{-1}(\mathbf{X}_i - \hat{\mu}_1) < -log(\hat{\pi}_2) + log\{det(\hat{\Sigma}_2)\} + (\mathbf{X}_i - \hat{\mu}_2)^T \hat{\Sigma}_2^{-1}(\mathbf{X}_i - \hat{\mu}_2)$$

,where $i = 1, 2$

$\hat{\pi}_i$: The prior probability of each of two groups.**One parameter** need to be estimated and then we calculate the prior probability for another group by $1 - \hat{\pi}_i$

$\hat{\Sigma}_i$: Covariance matrix of the group i . There are **133590 parameters** need to be estimated in total ,because each of two symmetric covariance matrices is 365×365 dimensions and has $\frac{365 \times 366}{2} = 66795$ unique elements.

$\hat{\mu}_i$: Mean vector of the group i. We need to estimate **730** parameters in all, and 365 empirical mean for each of two mean vectors.Number of Parameters in QD=1+133590+730=134321

(a2)

There are **366** parameters need to be estimated for Logistic classifier,based on the formula of Logistic model:

we classify new $\mathbf{X}_i$ in group 1 if $\hat{\beta}_0 + \hat{\beta}^T \mathbf{X}_i > \frac{1}{2}$,where the number of parameters in the $\hat{\beta}$ vector is 365 ,because there would be one parameter for each of the 365 variables in $\mathbf{X}$,and we also need to estimate the $\hat{\beta}_0$,hence the number of parameters to be estimated in Logistic classifier is $365 + 1 = 366$.

(b)

(b1)

```r
#load package and dataset
library(MASS)
```

```
XGtrain<- read.table("/Users/guanmuhan/Downloads/XGtrainRain (1).txt", header = TRUE, sep = ",")
XGtrain$G = as.factor(XGtrain$G)
XGtest <- read.table("/Users/guanmuhan/Downloads/XGtestRain (1).txt", header = TRUE, sep = ",")
XGtest$G = as.factor(XGtest$G)
```

```
#mod_qda0 = qda(G ~ ., data = XGtrain)

# Use tryCatch() to capture errors
tryCatch({
  mod_qda0 = qda(G ~ ., data = XGtrain)
}, error = function(e) {
  # Print the error message
  cat("Error in qda.default():\n")
  cat(conditionMessage(e), "\n")
})
```

```
## Error in qda.default():
## some group is too small for 'qda'
```

When I run the QD classifier model, there is an error message that some group is too small for 'qda' and I fail to run this model, which is because the number of observations is much smaller than that of variables,hence the estimation of covariance matrices would become unreliable and incorrect.

(b2)

```
# Fit a logistic regression model from training data
mod_lr0 = glm(G ~ .,
              data = XGtrain,
              family = binomial(link="logit"),
              control = glm.control(maxit = 100))
```

```
summary(mod_lr0$coefficients)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -81.4440  -3.0966   0.2449  -0.4881   2.5303   9.4901      216
```

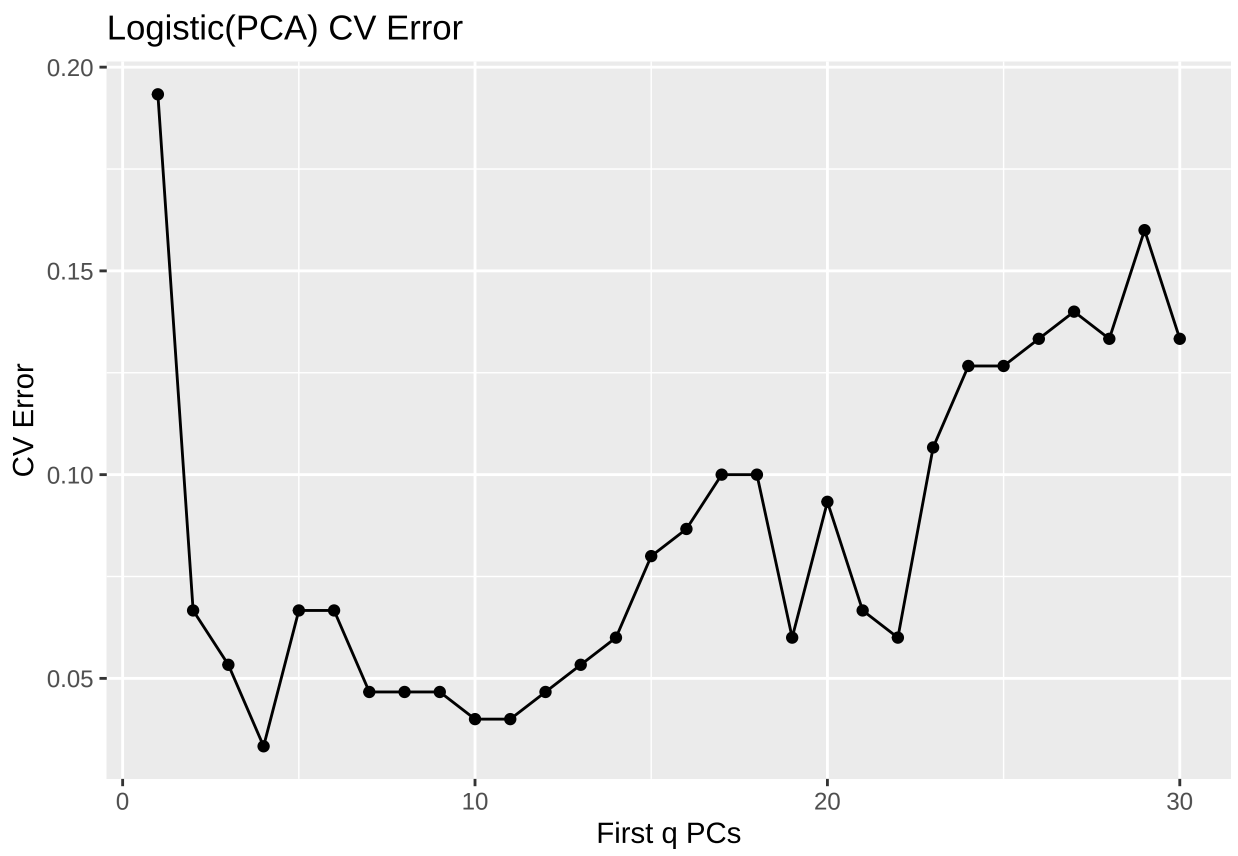
We can notice that there are 216 coefficients are not available.Similarly,the issue arises due to the fact that the number of variables significantly exceeds the number of observations,hence most of the parameters cannot be estimated.

Problem2

(a)

```
Gtrain = XGtrain[, 366]
#scaling the data when using PCA
PCX = prcomp(XGtrain[, -c(366)], retx = T,center = TRUE, scale. = F)
# Train logistic classifier using first q PCA components
CV_error = rep(0, 30)
for (q in 1:30) {
  prediction = c()
  for (i in 1:dim(XGtrain)[1]) {
    GDATACV = Gtrain[-i]

    logistic = suppressWarnings(glm(GDATACV~.,
```

```r
                                         data = as.data.frame(PCX$x[-i, 1:q]),
                                         family = binomial(link = "logit")))
    #amend the type of data
    new_data = as.data.frame(t(PCX$x[i, 1:q]))
    #make sure the column names of test data are the same with that of the train data
    colnames(new_data) = colnames(as.data.frame(PCX$x[-i, 1:q]))
    prediction[i] = ifelse(predict(logistic, newdata = new_data,type='response') > 0.5, 1, 0)
  }
  CV_error[q] = sum(prediction != Gtrain)/dim(XGtrain)[1]
}

# Plotting the CV error
library(ggplot2)
plot_data = data.frame(q = 1:30, CV_error = CV_error)
ggplot(data = plot_data, aes(x = q, y = CV_error)) +
  geom_line() +
  geom_point() +
  labs(x = "First q PCs", y = "CV Error", title = "Logistic(PCA) CV Error")
```



```r
CV_error# all cv error
```

```
##  [1] 0.19333333 0.06666667 0.05333333 0.03333333 0.06666667 0.06666667
##  [7] 0.04666667 0.04666667 0.04666667 0.04000000 0.04000000 0.04666667
## [13] 0.05333333 0.06000000 0.08000000 0.08666667 0.10000000 0.10000000
## [19] 0.06000000 0.09333333 0.06666667 0.06000000 0.10666667 0.12666667
## [25] 0.12666667 0.13333333 0.14000000 0.13333333 0.16000000 0.13333333
```

```r
#the optimal q
which(CV_error == min(CV_error))
```

```
## [1] 4
```

```r
#the cv_error of optimal q
min(CV_error)
```

```
## [1] 0.03333333
```

We can observe that the cross-validation prediction error is minimized when q equals 3, 6, or 7. This means that logistic classifiers using the first 3, 6, or 7 principal components perform best on the training dataset.

(b)

```r
#based on our conclusion,we fit a model with first three(the smallest q) PCs.
logistic_PCA = glm(Gtrain~., data = as.data.frame(PCX$x[, 1:4]),
                     family = binomial(link = "logit"))
summary(logistic_PCA)
```

```
##
## Call:
## glm(formula = Gtrain ~ ., family = binomial(link = "logit"),
##     data = as.data.frame(PCX$x[, 1:4]))
##
## Deviance Residuals:
##       Min          1Q      Median          3Q         Max
## -0.004656   0.000000   0.000000   0.000000   0.004984
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -531.37   14350.44  -0.037    0.970
## PC1            -36.10     531.41  -0.068    0.946
## PC2             50.41     712.66   0.071    0.944
## PC3            -46.44     603.41  -0.077    0.939
## PC4            243.15    3631.14   0.067    0.947
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2.0017e+02  on 149  degrees of freedom
## Residual deviance: 4.7290e-05  on 145  degrees of freedom
## AIC: 10
##
## Number of Fisher Scoring iterations: 25
```

```r
#make sure the test data use the same mean  with that of train data when apply PCA
train_mean = colMeans(XGtrain[, -c(366)])
X_test = scale(XGtest[, -c(366)], center = train_mean, scale = F)
X_test_pca = predict(PCX, newdata =X_test)
PCA_newdata = as.data.frame(X_test_pca[, 1:4])
#predict the label
prediction = ifelse(predict(logistic_PCA, newdata = PCA_newdata,
                             family = binomial(link = "logit"),type = 'response') > 0.5, 1, 0)
#predict the error
LR_PCA_ERROR <- sum(prediction != XGtest[,366])
cat('prediction error for logistic classifier with q=4:',LR_PCA_ERROR,'\n')
```

```
## prediction error for logistic classifier with q=4: 4
```

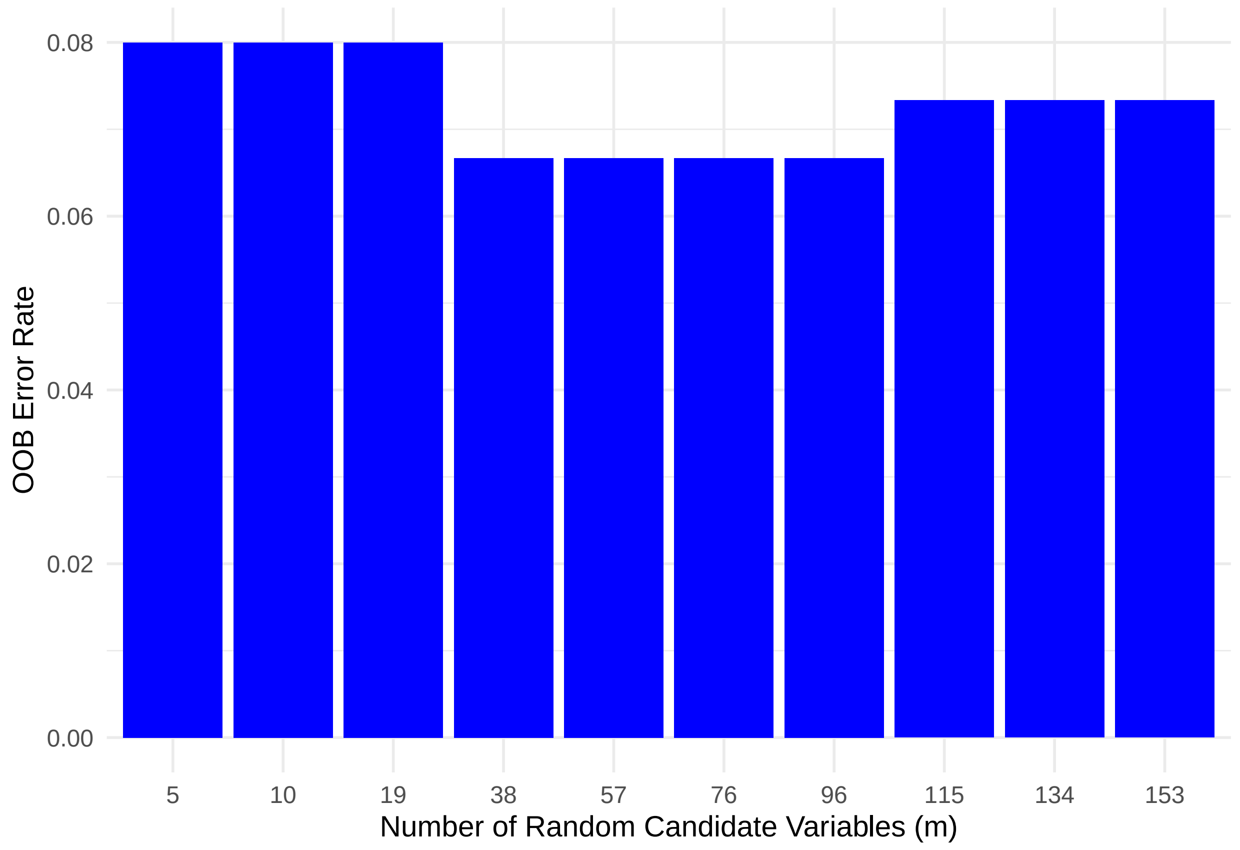
Problem3

(a)

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
#set seed to reproduce the same result
c <- sqrt(365)
#set the grid of m
m_candidates <- round(c * c(1/4, 1/2, 1, 2, 3, 4, 5, 6, 7, 8))
oob_errors <- numeric(length(m_candidates))
set.seed(90139)
for (i in 1:length(m_candidates)) {
  m <- floor(m_candidates[i])
  # Train the random forest model with B = 5000 trees and the current m value
  model <- randomForest(formula=G~., data=XGtrain, ntree = 5000, mtry = m,importance=TRUE)
  # store the OOB error rate for each of the final ten models
  oob_errors[i] <- model$err.rate[nrow(model$err.rate), "OOB"]
}
#visualize the error rate by bar chart
ooberror <- data.frame(m = m_candidates, OOB_Error = oob_errors)
ggplot(data = ooberror, aes(x = factor(m), y = OOB_Error)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "Number of Random Candidate Variables (m)", y = "OOB Error Rate") +
  scale_x_discrete(labels = ooberror$m) +
  theme_minimal()
```



```
#extract the smallest m
best_m <- m_candidates[which.min(oob_errors)]
best_oob_error <- min(oob_errors)
cat('the smallest value of m is :',best_m,'\n')
```

```
## the smallest value of m is : 38
```

```
cat('the OOB classification error for the model with best_m is :',best_oob_error,'\n')
```

```
## the OOB classification error for the model with best_m is : 0.06666667
```

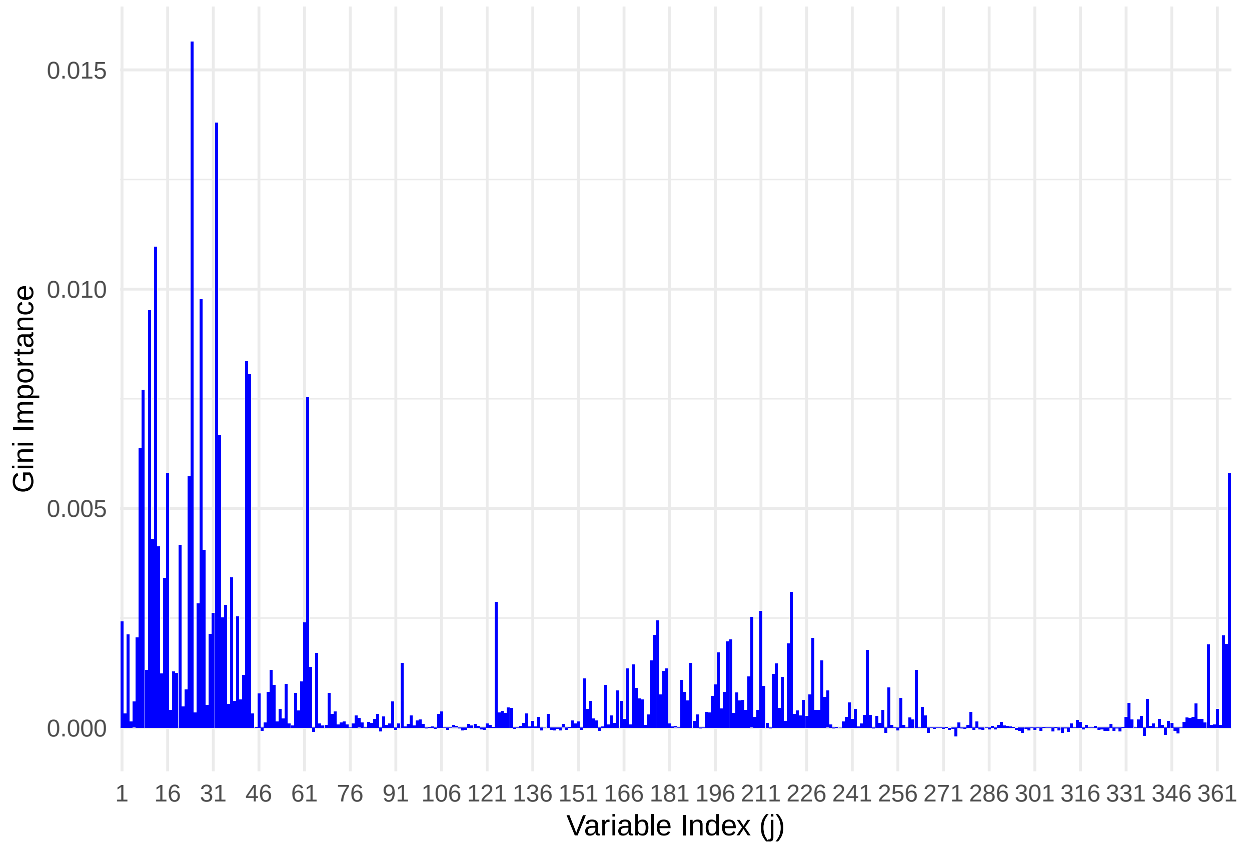
(b)

```

#load the best model
best_model <- randomForest(formula=G~., data=XGtrain, ntree = 5000, mtry =  38  ,importance=TRUE)
summary(best_model)
```

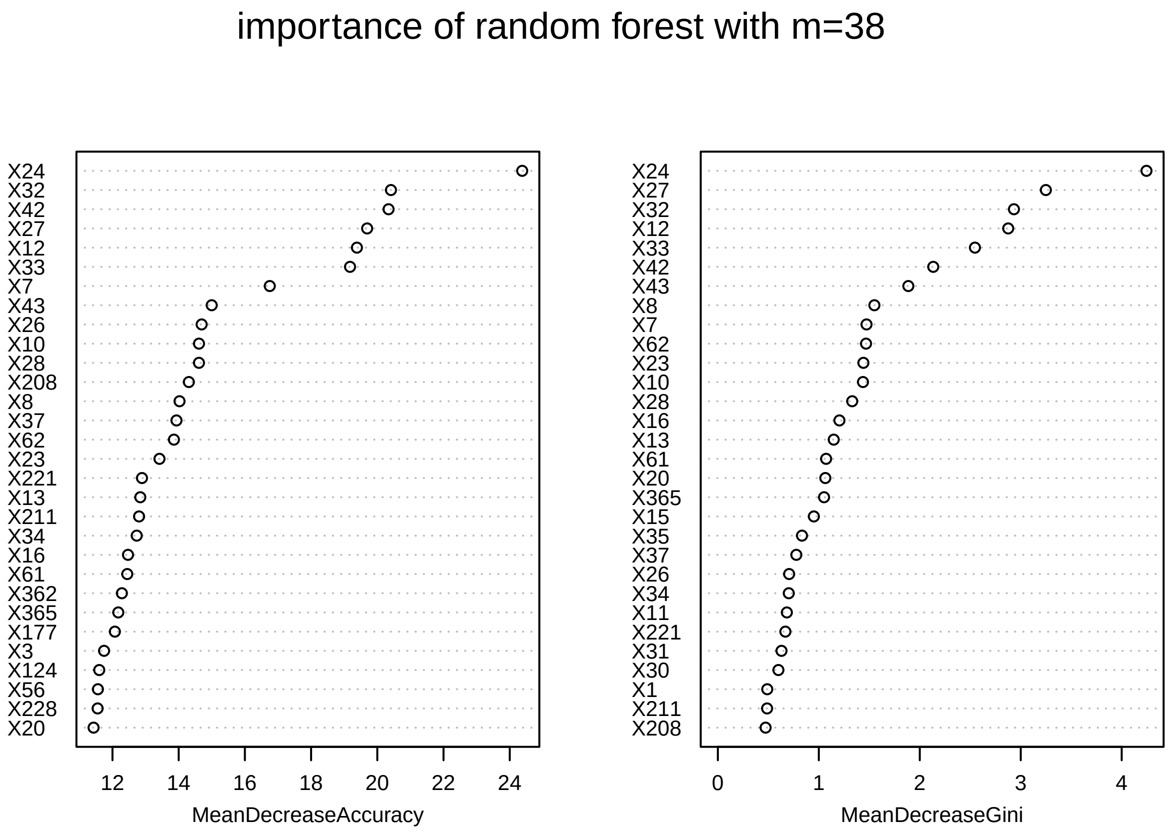
Plot for Gini importance

```
##                 Length Class  Mode
## call                6  -none- call
## type                1  -none- character
## predicted         150  factor numeric
## err.rate        15000  -none- numeric
## confusion           6  -none- numeric
## votes             300  matrix numeric
## oob.times         150  -none- numeric
## classes              2  -none- character
## importance        1460  -none- numeric
## importanceSD      1095  -none- numeric
## localImportance      0  -none- NULL
## proximity            0  -none- NULL
## ntree                1  -none- numeric
## mtry                 1  -none- numeric
## forest              14  -none- list
## y                  150  factor numeric
## test                 0  -none- NULL
## inbag                0  -none- NULL
## terms                3  terms  call
```

```
# Create a data frame containing variable index j and corresponding Gini importance value
importance_values <- best_model$importance
data_df <- data.frame(j = 1:365, GiniImportance = importance_values[1:365, 1])

# Set the step size to control the display density
step <- 15

#Use ggplot2 to create a histogram that displays a label every step j value
ggplot(data = data_df, aes(x = factor(j), y = GiniImportance)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "Variable Index (j)", y = "Gini Importance") +
  scale_x_discrete(breaks = data_df$j[seq(1, nrow(data_df), by = step)], labels = data_df$j[seq(1, nrow
  theme_minimal()
```



```r
#visualization the top thirty most important features
varImpPlot(best_model, cex = 0.7, main = "importance of random forest with m=38 ")
```



The dataset contains variables representing daily rainfall statistics for each year. The Gini importance scores reveal that majority of the top 30 important features are related to rainfall statistics during the summer season. This suggests that there is a significant difference in rainfall between the northern and southern regions of Australia during the summer months. This difference is valuable for our classifier to identify the geographical location of weather stations.(North or South)

```r
# use the model to do prediction
predictions <- predict(best_model, newdata = XGtest,response = "G")
#calculate the number of error
error_count <- sum(predictions != XGtest$G)
cat("Number of prediction errors:", error_count, "\n")
```

Prediction on test data

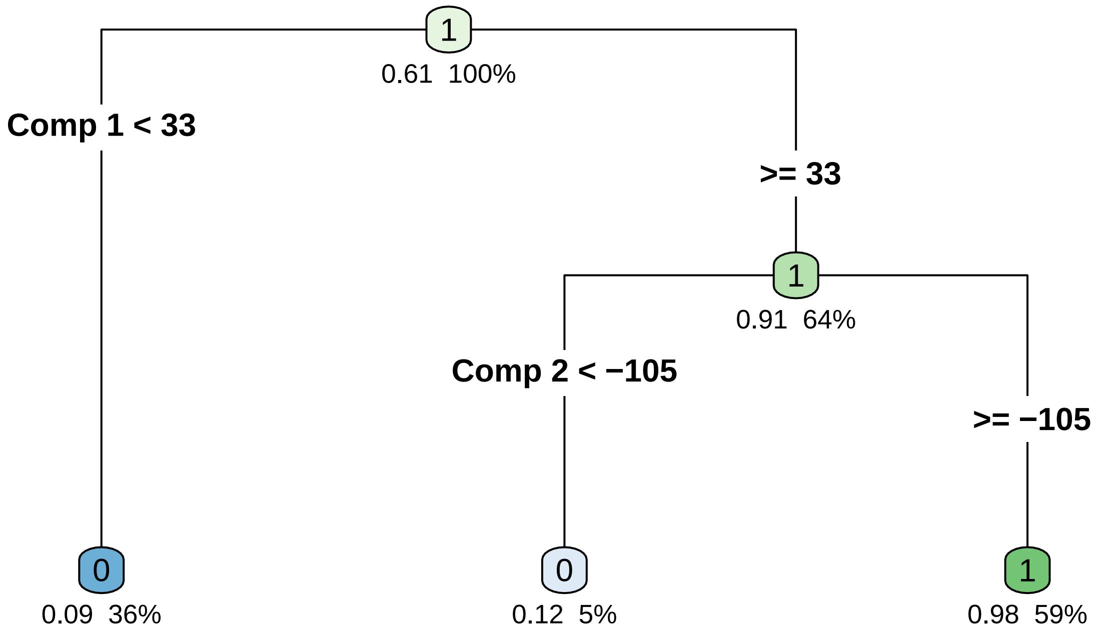
```
## Number of prediction errors: 3
```

(c)

```r
#loading packages
library(rpart)
library(rpart.plot)
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```
# Create a decision tree model
XGtrain<- read.table("/Users/guanmuhan/Downloads/XGtrainRain (1).txt", header = TRUE, sep = ",")
#center the data when using pls
train_mean = colMeans(XGtrain[, -c(366)])
XGtrain[, -c(366)]=scale(XGtrain[, -c(366)], center = train_mean,scale = FALSE)
# PLS using centered data
PLS <- plsr(G ~ ., data = XGtrain, scale = FALSE)
X_PLS_df = as.data.frame(PLS$scores[,1:50])
XGtrain$G=as.factor(XGtrain$G)
#train a tree model
tree = rpart(XGtrain$G~., data=X_PLS_df,method='class')
#plot
rpart.plot(tree, clip.right.labs = T,type=4,under=T)
```



```
summary(tree)
```

```
## Call:
## rpart(formula = XGtrain$G ~ ., data = X_PLS_df, method = "class")
##   n= 150
##
##           CP nsplit rel error    xerror       xstd
## 1 0.7586207      0 1.0000000 1.0000000 0.10283342
## 2 0.1034483      1 0.2413793 0.2931034 0.06693861
## 3 0.0100000      2 0.1379310 0.2413793 0.06142708
##
## Variable importance
##  Comp 1  Comp 3  Comp 5  Comp 7  Comp 6 Comp 11  Comp 2 Comp 10 Comp 12
##      27      19      13      11      11      10       6       2       2
##
## Node number 1: 150 observations,    complexity param=0.7586207
##   predicted class=1  expected loss=0.3866667  P(node) =1
##     class counts:    58    92
##    probabilities: 0.387 0.613
##   left son=2 (54 obs) right son=3 (96 obs)
##   Primary splits:
##       Comp 1 < 33.35609  to the left,  improve=45.76009, (0 missing)
##       Comp 3 < -35.75959 to the left,  improve=31.61485, (0 missing)
##       Comp 5 < -12.6362  to the left,  improve=24.80710, (0 missing)
##       Comp 7 < -10.02891 to the left,  improve=23.09383, (0 missing)
##       Comp 2 < -89.71639 to the left,  improve=16.99730, (0 missing)
##   Surrogate splits:
##       Comp 3  < -22.17146 to the left,  agree=0.893, adj=0.704, (0 split)
##       Comp 5  < -15.02397 to the left,  agree=0.807, adj=0.463, (0 split)
##       Comp 6  < -23.99939 to the left,  agree=0.747, adj=0.296, (0 split)
##       Comp 7  < -10.02891 to the left,  agree=0.740, adj=0.278, (0 split)
##       Comp 11 < 13.96993  to the right, agree=0.733, adj=0.259, (0 split)
##
## Node number 2: 54 observations
##   predicted class=0  expected loss=0.09259259  P(node) =0.36
##     class counts:    49     5
##    probabilities: 0.907 0.093
##
## Node number 3: 96 observations,    complexity param=0.1034483
##   predicted class=1  expected loss=0.09375  P(node) =0.64
##     class counts:     9    87
##    probabilities: 0.094 0.906
##   left son=6 (8 obs) right son=7 (88 obs)
##   Primary splits:
##       Comp 2  < -105.234  to the left,  improve=10.653410, (0 missing)
##       Comp 7  < -12.10875 to the left,  improve=10.653410, (0 missing)
##       Comp 6  < -18.65945 to the left,  improve= 8.205523, (0 missing)
##       Comp 10 < -11.78318 to the left,  improve= 7.315709, (0 missing)
##       Comp 12 < -12.47811 to the left,  improve= 6.519397, (0 missing)
##   Surrogate splits:
##       Comp 7  < -22.11906 to the left,  agree=0.958, adj=0.500, (0 split)
##       Comp 6  < -23.31956 to the left,  agree=0.948, adj=0.375, (0 split)
##       Comp 11 < -18.86167 to the left,  agree=0.948, adj=0.375, (0 split)
##       Comp 10 < -18.88818 to the left,  agree=0.938, adj=0.250, (0 split)
##       Comp 12 < -19.12286 to the left,  agree=0.938, adj=0.250, (0 split)
##
## Node number 6: 8 observations
##   predicted class=0  expected loss=0.125  P(node) =0.05333333
##     class counts:     7     1
##    probabilities: 0.875 0.125
##
```

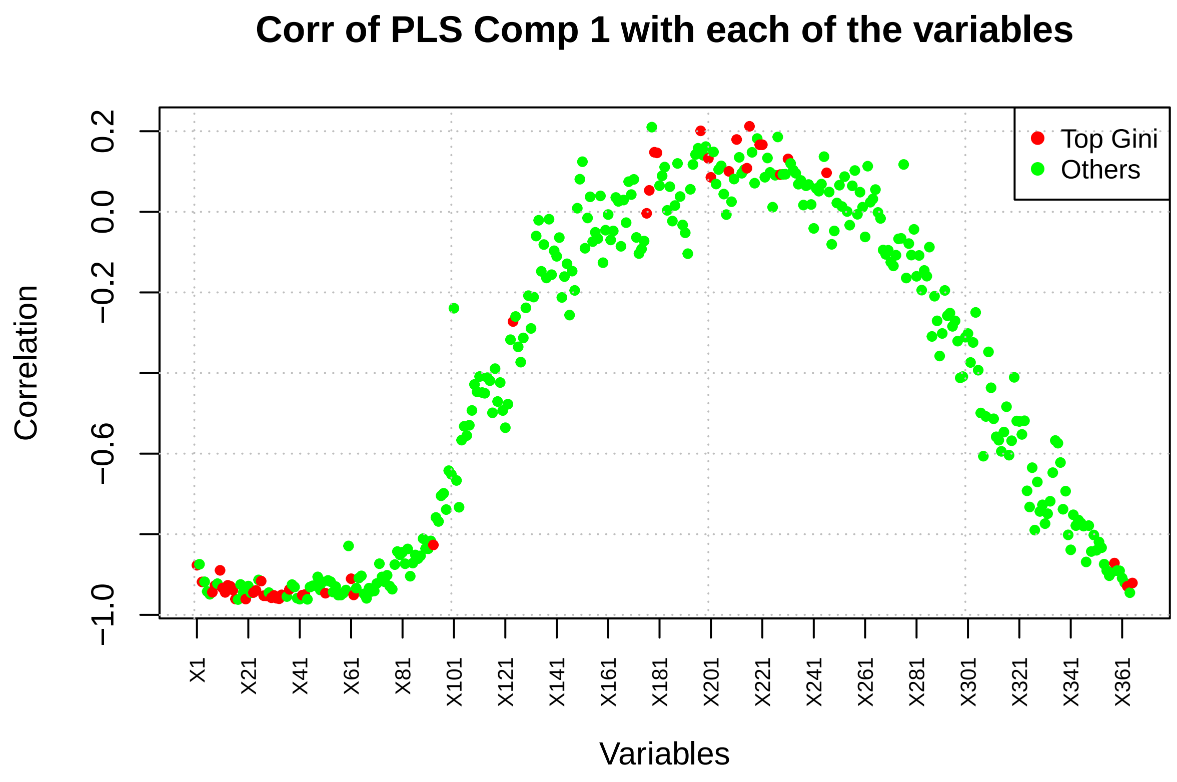
```
## Node number 7: 88 observations
##   predicted class=1  expected loss=0.02272727  P(node) =0.5866667
##     class counts:     2    86
##    probabilities: 0.023 0.977
```

Which PLS components play a role in the tree?

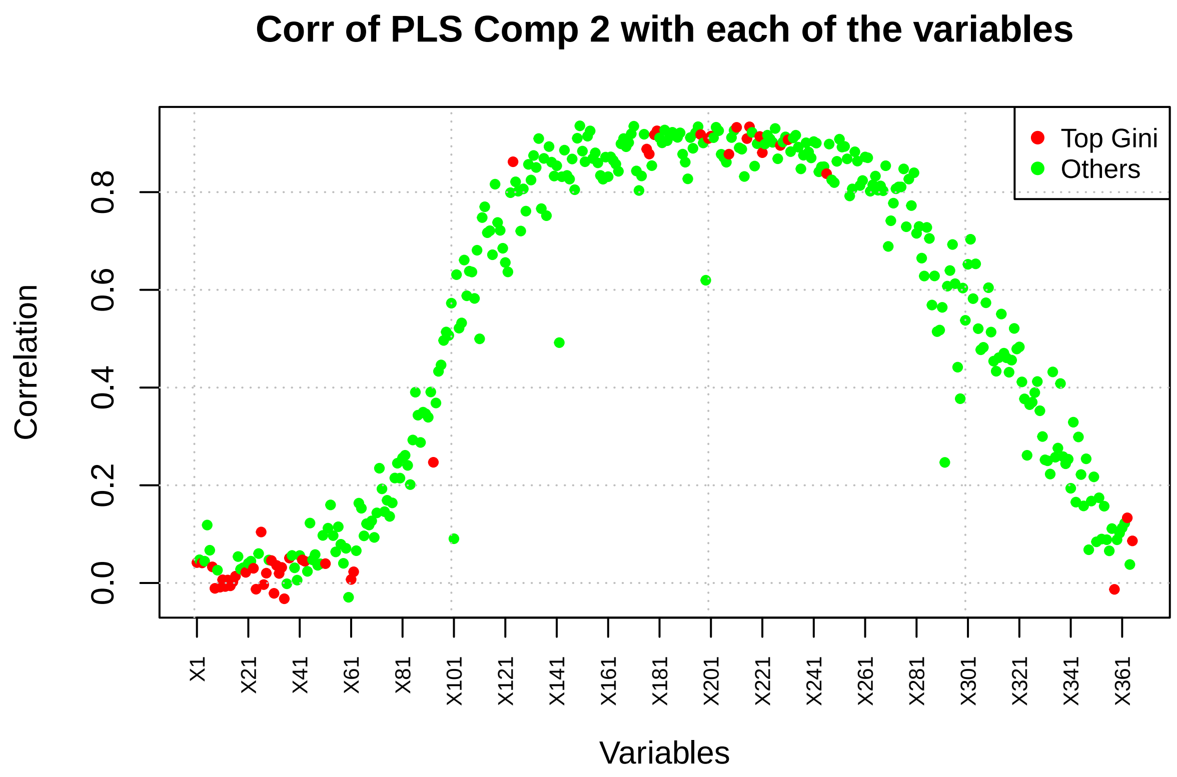
The plot generated by the `rpart.plot` function shows that PLS components *Comp 1 and Comp 2* play crucial roles in the splits at certain nodes. In the `summary` command, we observe that the variable importance rankings are as follows: *Comp 1, Comp 3, Comp 6, Comp 5, Comp 14, Comp 7, Comp 2, Comp 10, Comp 11, Comp 13, and Comp 8.*

```r
plot_correlation_with_gini_dataframe <- function(i, j) {
  # Get the j variable names with the highest Gini Importance
  top_j_importance <- head(sort(best_model$importance[, "MeanDecreaseGini"], decreasing = TRUE), j)
  top_j_variables <- names(top_j_importance)
  data=as.data.frame(cor(XGtrain[, -ncol(XGtrain)],X_PLS_df[, paste0("Comp ", i)]))
  data$x=colnames(XGtrain[, -ncol(XGtrain)])
  data$top_j_variables=ifelse(data$x %in% top_j_variables, "red", "green")

  #scatter plot
  plot(data$V1, pch = 19,
       col = data$top_j_variables,
       main = paste("Corr of PLS Comp", i, "with each of the variables"),
       xlab = "Variables", ylab = "Correlation",
       xaxt = "n", cex = 0.6)
  grid(col = "gray", lty = "dotted")

  axis(1, at = seq(1, nrow(data), by = 20),
       labels = data$x[seq(1, nrow(data), by = 20)],
       cex.axis = 0.7, las = 2)

  # add lengend
  legend("topright", legend = c("Top Gini", "Others"),
         pch = 19, col = c("red", "green"), cex = 0.8)


}

plot_correlation_with_gini_dataframe(1, 50)
```

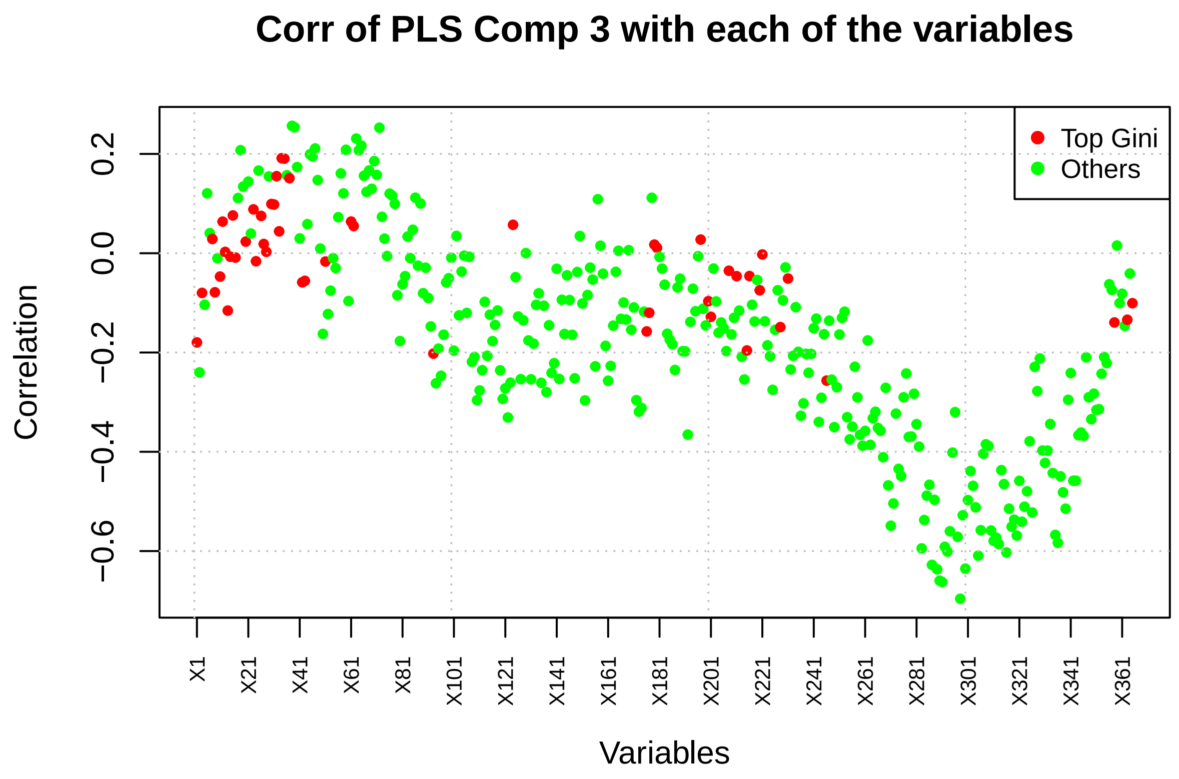
Plot the correlation of PLS component i with top j variables with highest Gini importance score



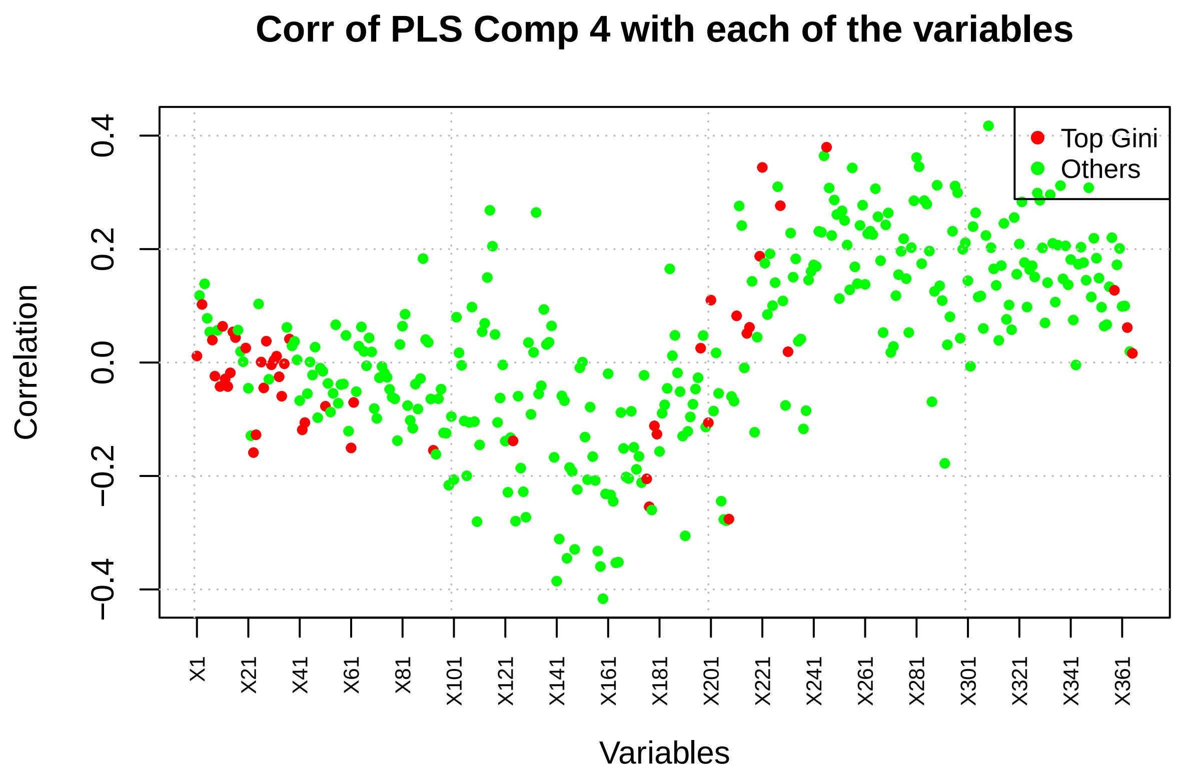
```
plot_correlation_with_gini_dataframe(2, 50)
```



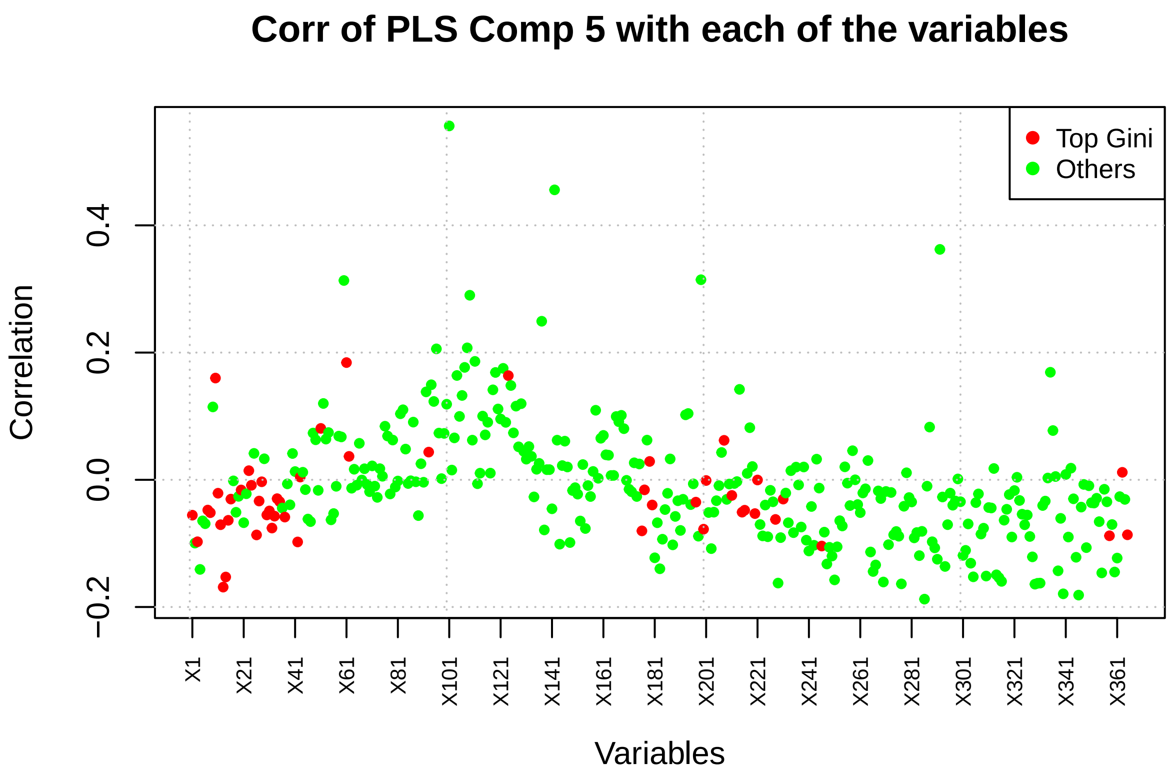
```r
plot_correlation_with_gini_dataframe(3, 50)
```



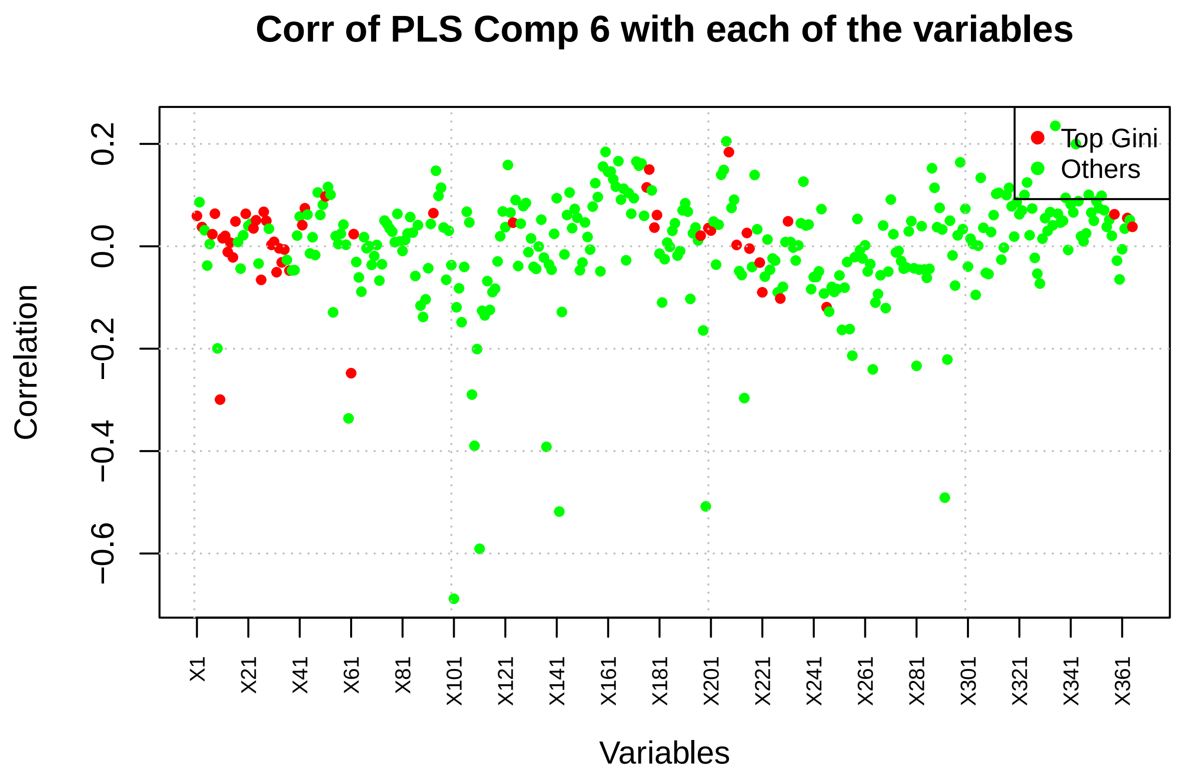
```
plot_correlation_with_gini_dataframe(4, 50)
```



```
plot_correlation_with_gini_dataframe(5, 50)
```



```
plot_correlation_with_gini_dataframe(6, 50)
```



Correlation Analysis Here, we have selected a total of 6 PLS components (i=*1, 2, 3, 4, 5, 6*) that are important for our single tree models. We have plotted the **correlation scatter plot** for each of xis PLS components with the all variables in the random forest classifier, and also **highlight the important variables in part(b) by 'red' color**.

Many of the top important variables(roughly during the `summer period`) generally exhibit a **strong negative correlation** with PLS Comp 1, while their correlation with PLS Comp 2 is relatively weaker.

In comparison,some of the top important variables(roughly during the `winter period`) show a **strong positive correlation** with PLS Comp 2, with a weaker correlation with Comp1.

Furthermore, the remaining 4 components **do not exhibit strong correlations** with the majority of the important original variables.

```
#transform the test data by the same mean and variance with that of the trained data before PLS
X_test = scale(XGtest[, -c(366)], center = train_mean, scale = F)
#apply the same PLS transformation on the test data
X_test_pls=X_test%*%PLS$projection
#dim(X_test_pls)
predictions <- predict(tree, newdata = as.data.frame(X_test_pls[,1:50]),type = 'class')
error_count <- sum(predictions != XGtest$G)
cat("Number of prediction errors:", error_count, "\n")
```

Prediction on test data

```
## Number of prediction errors: 2
```