STAT154 Project2

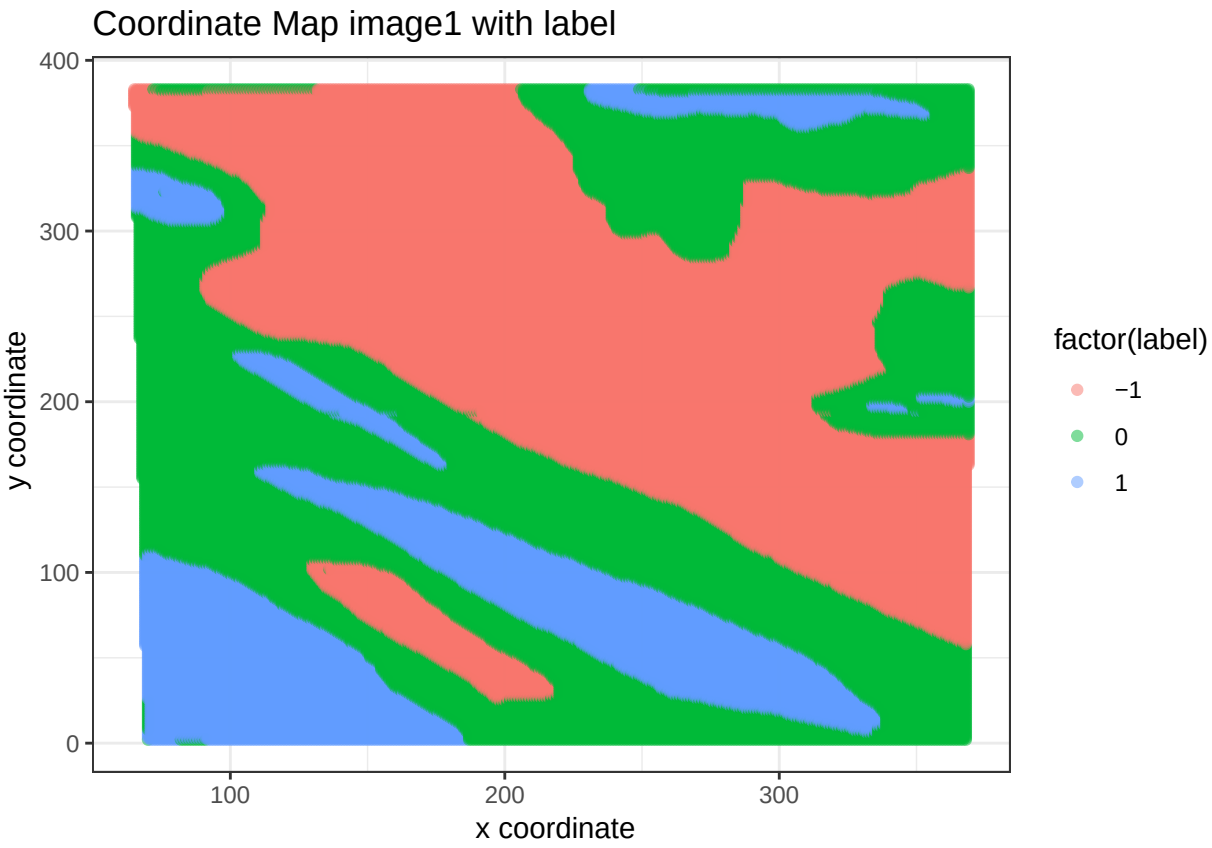
Keqin Cao

5/3/2019

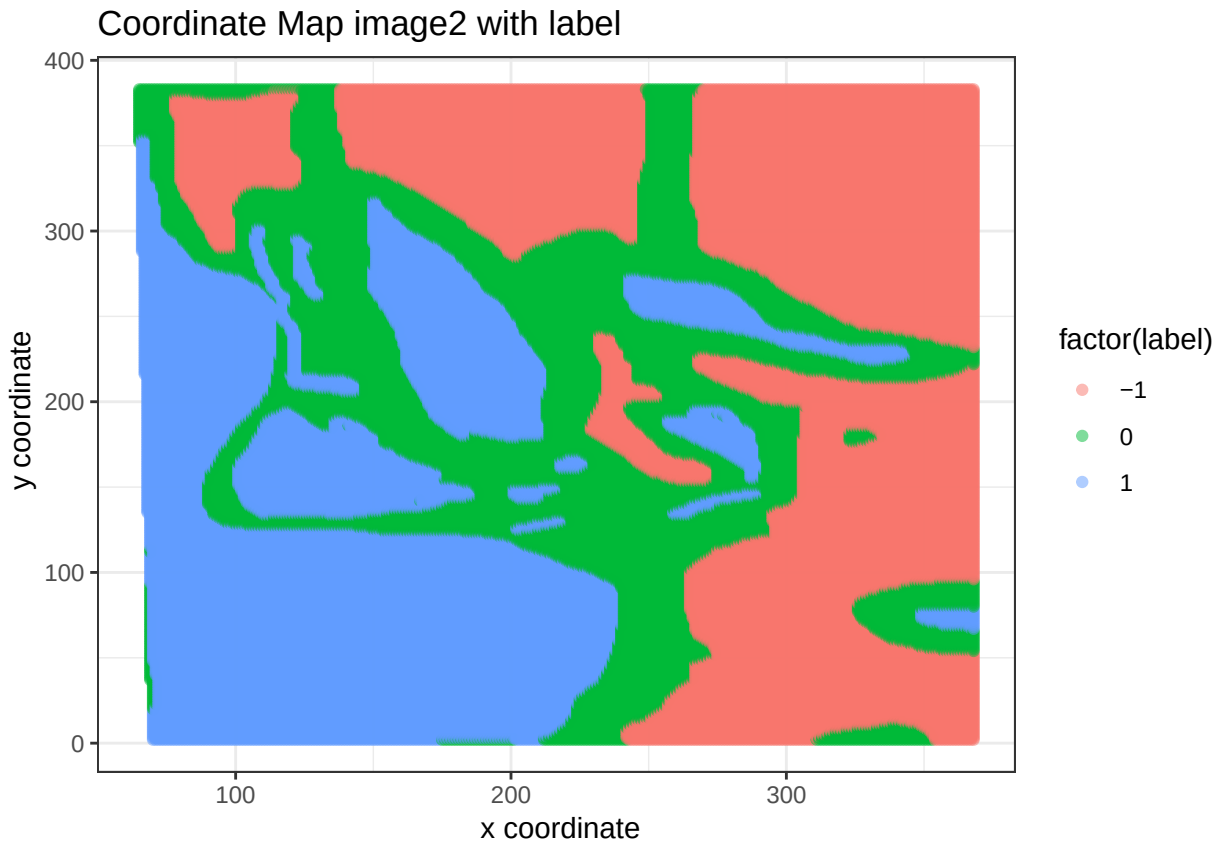
```
# read in the data and rename the column name to align with cloud data in the data dictionaries
image1<- as.data.frame(read.table("image_data/image1.txt"))
image1$picture<- rep("image1", nrow(image1))
colnames(image1) [1:11] <- c("y_coordinate", "x_coordinate", "label", "NDAI", "SD", "CORR", "DF_angle",

image2<- as.data.frame(read.table("image_data/image2.txt"))
image2$picture<- rep("image2", nrow(image2))
colnames(image2) [1:11] <- c("y_coordinate", "x_coordinate", "label", "NDAI", "SD", "CORR", "DF_angle",

image3<- as.data.frame(read.table("image_data/image3.txt"))
image3$picture<- rep("image3", nrow(image3))
colnames(image3) [1:11] <- c("y_coordinate", "x_coordinate", "label", "NDAI", "SD", "CORR", "DF_angle",
```

Problem 1b Summarize Data

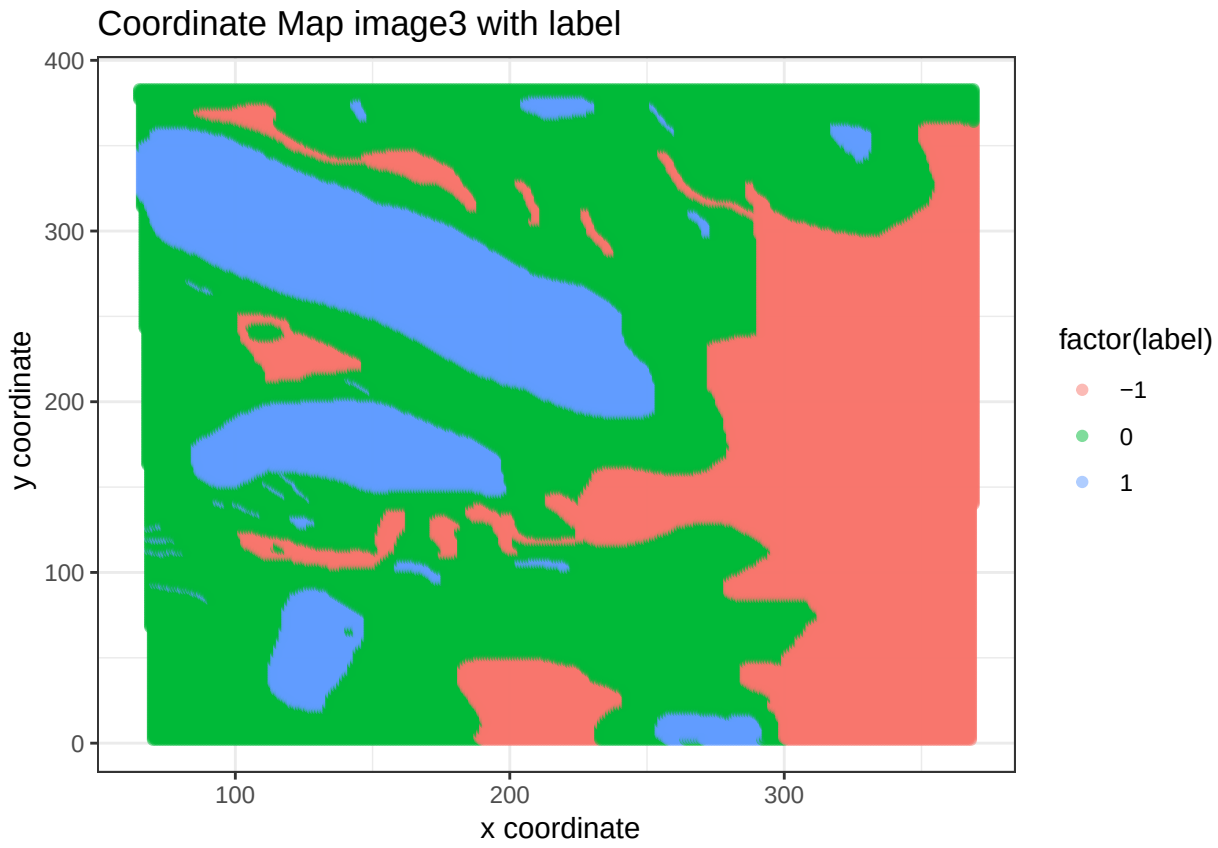
```
total_df <- rbind(image1, image2, image3)

ggplot(image1)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate,
                                            color = factor(label)))+
  xlab(" x coordinate") +
  ylab(" y coordinate") +
  ggtitle("Coordinate Map image1 with label ")+
  theme_bw()
```




```
ggplot(image2)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate,
                                            color = factor(label)))+
  xlab(" x coordinate") +
  ylab(" y coordinate") +
  ggtitle("Coordinate Map image2 with label ")+
  theme_bw()
```




```
ggplot(image3)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate,
                                            color = factor(label)))+
  xlab(" x coordinate") +
  ylab(" y coordinate") +
  ggtitle("Coordinate Map image3 with label ")+
  theme_bw()
```




```
#Check if there's any missing data
sum(is.na(total_df))
```

```
## [1] 0
```

```
#There is no missing value
summary(total_df)
```

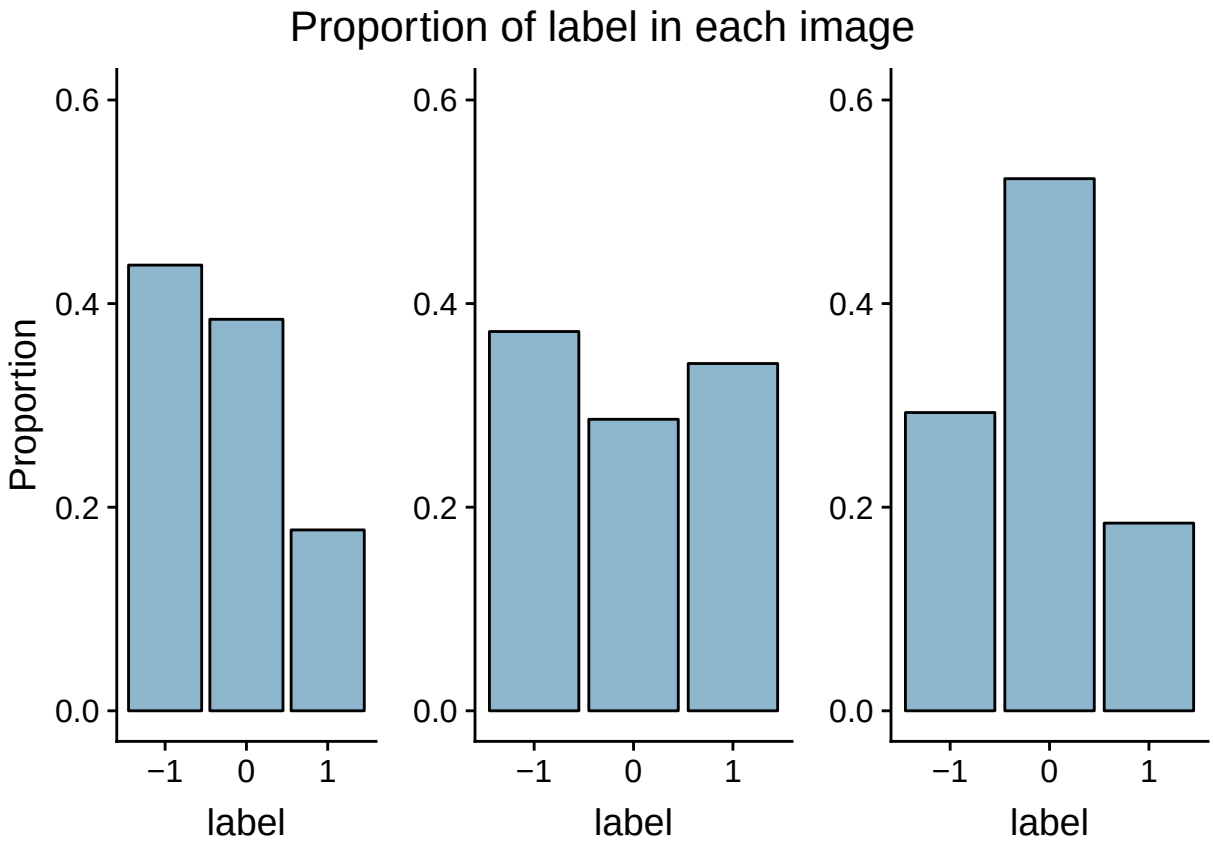
```
##   y_coordinate    x_coordinate       label              NDAI
##  Min.   :  2.0   Min.   : 65.0   Min.   :-1.0000   Min.   :-1.8420
##  1st Qu.: 98.0   1st Qu.:143.0   1st Qu.:-1.0000   1st Qu.:-0.4286
##  Median :193.0   Median :218.0   Median : 0.0000   Median : 1.3476
##  Mean   :193.1   Mean   :218.1   Mean   :-0.1334   Mean   : 1.0847
##  3rd Qu.:289.0   3rd Qu.:294.0   3rd Qu.: 0.0000   3rd Qu.: 2.3142
##  Max.   :383.0   Max.   :369.0   Max.   : 1.0000   Max.   : 4.5639
##        SD               CORR             DF_angle         CF_angle
##  Min.   :  0.1987   Min.   :-0.3872   Min.   : 45.28   Min.   : 31.19
##  1st Qu.:  1.6376   1st Qu.: 0.1253   1st Qu.:244.56   1st Qu.:219.27
##  Median :  4.3095   Median : 0.1603   Median :281.91   Median :259.31
##  Mean   :  8.0633   Mean   : 0.1860   Mean   :271.36   Mean   :246.37
##  3rd Qu.: 10.2264   3rd Qu.: 0.2231   3rd Qu.:300.39   3rd Qu.:279.59
##  Max.   :117.5810   Max.   : 0.8144   Max.   :410.53   Max.   :360.68
##     BF_angle         AF_angle         AN_angle        picture
##  Min.   : 24.49   Min.   : 21.07   Min.   : 20.57   Length:345556
##  1st Qu.:200.79   1st Qu.:185.16   1st Qu.:174.88   Class :character
##  Median :236.17   Median :211.54   Median :197.58   Mode  :character
##  Mean   :224.20   Mean   :201.71   Mean   :188.29
##  3rd Qu.:258.62   3rd Qu.:235.15   3rd Qu.:216.80
```

```
##  Max.   :335.08   Max.   :318.70   Max.   :306.93
```

```
#Calculate the proportion
percentage_label<- total_df %>%
  group_by(picture, label) %>%
  summarise (n = n()) %>%
  mutate(freq = n / sum(n))
percentage_label
```

```
## # A tibble: 9 x 4
## # Groups:   picture [3]
##   picture label     n  freq
##   <chr>   <dbl> <int> <dbl>
## 1 image1     -1 50446 0.438
## 2 image1      0 44312 0.385
## 3 image1      1 20471 0.178
## 4 image2     -1 42882 0.373
## 5 image2      0 32962 0.286
## 6 image2      1 39266 0.341
## 7 image3     -1 33752 0.293
## 8 image3      0 60221 0.523
## 9 image3      1 21244 0.184
```

```
g1 <- ggplot(data=percentage_label[percentage_label$picture=='image1',],
  aes(x=label, y=freq)) +
  geom_bar(stat="identity", color="black", fill="lightskyblue3")+
  ylab("Proportion")+
  ylim(0,0.6)
g2 <- ggplot(data=percentage_label[percentage_label$picture=='image2',],
  aes(x=label, y=freq)) +
  geom_bar(stat="identity", color="black", fill="lightskyblue3")+
  ylab(NULL)+
  ylim(0,0.6)
g3 <- ggplot(data=percentage_label[percentage_label$picture=='image3',],
  aes(x=label, y=freq)) +
  geom_bar(stat="identity", color="black", fill="lightskyblue3")+
  ylab(NULL)+
  ylim(0,0.6)
grid.arrange(g1,g2,g3,ncol=3,top = textGrob("Proportion of label in each image",
                                            gp = gpar(fontsize = 16)))
```



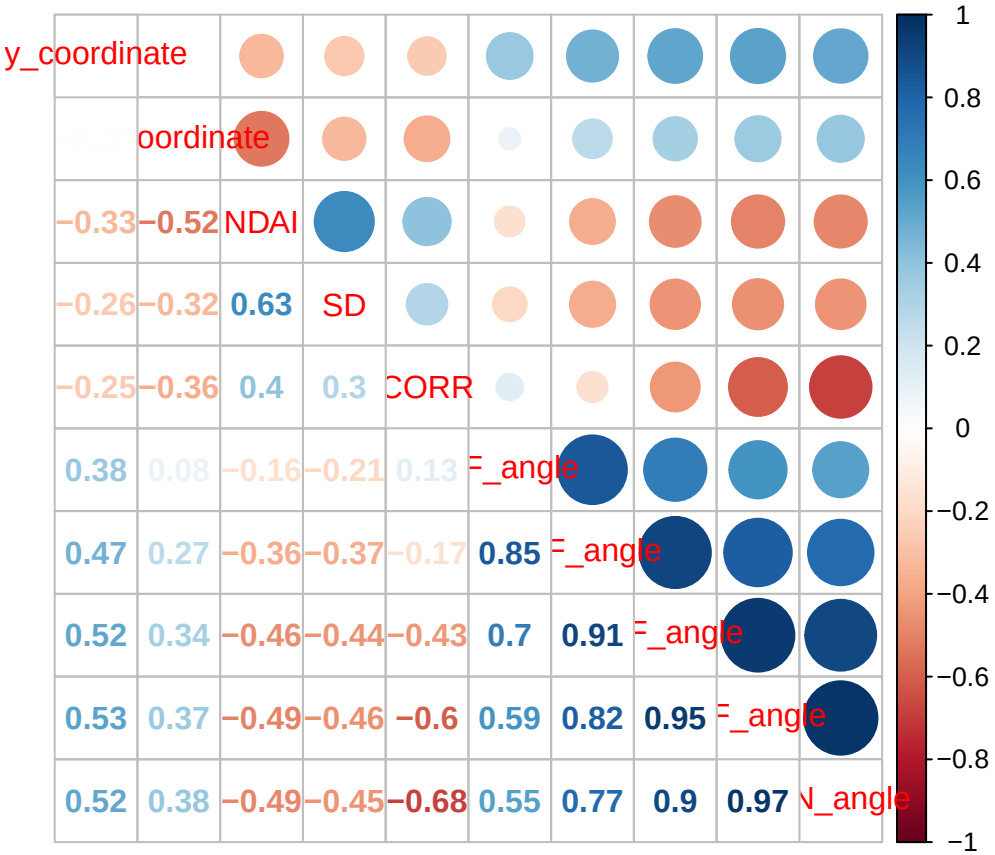
Problem 1c EDA

```
labels<- factor(total_df$label)
cor(total_df[,c(-12)])
```

```
##              y_coordinate x_coordinate        label       NDAI         SD
## y_coordinate  1.000000000 -0.006376002 -0.213372321 -0.3276781 -0.2646997
## x_coordinate -0.006376002  1.000000000 -0.465822566 -0.5231960 -0.3233889
## label        -0.213372321 -0.465822566  1.000000000  0.6169346  0.2954477
## NDAI         -0.327678096 -0.523195958  0.616934624  1.0000000  0.6310626
## SD           -0.264699672 -0.323388894  0.295447745  0.6310626  1.0000000
## CORR         -0.254309102 -0.361793127  0.444059231  0.4034998  0.2968385
## DF_angle      0.378983970  0.081274648  0.006550085 -0.1610916 -0.2061691
## CF_angle      0.472777594  0.269869879 -0.208279170 -0.3622113 -0.3688601
## BF_angle      0.520812966  0.339539816 -0.337948500 -0.4629301 -0.4404404
## AF_angle      0.530441793  0.368160603 -0.389741017 -0.4927484 -0.4555423
## AN_angle      0.515677887  0.383211616 -0.389358825 -0.4895267 -0.4466229
##                    CORR     DF_angle   CF_angle   BF_angle   AF_angle
## y_coordinate -0.2543091  0.378983970  0.4727776  0.5208130  0.5304418
## x_coordinate -0.3617931  0.081274648  0.2698699  0.3395398  0.3681606
## label         0.4440592  0.006550085 -0.2082792 -0.3379485 -0.3897410
## NDAI          0.4034998 -0.161091626 -0.3622113 -0.4629301 -0.4927484
## SD            0.2968385 -0.206169130 -0.3688601 -0.4404404 -0.4555423
## CORR          1.0000000  0.126283481 -0.1660966 -0.4311043 -0.6039353
## DF_angle      0.1262835  1.000000000  0.8495716  0.6991073  0.5910958
## CF_angle     -0.1660966  0.849571562  1.0000000  0.9119430  0.8216176
```

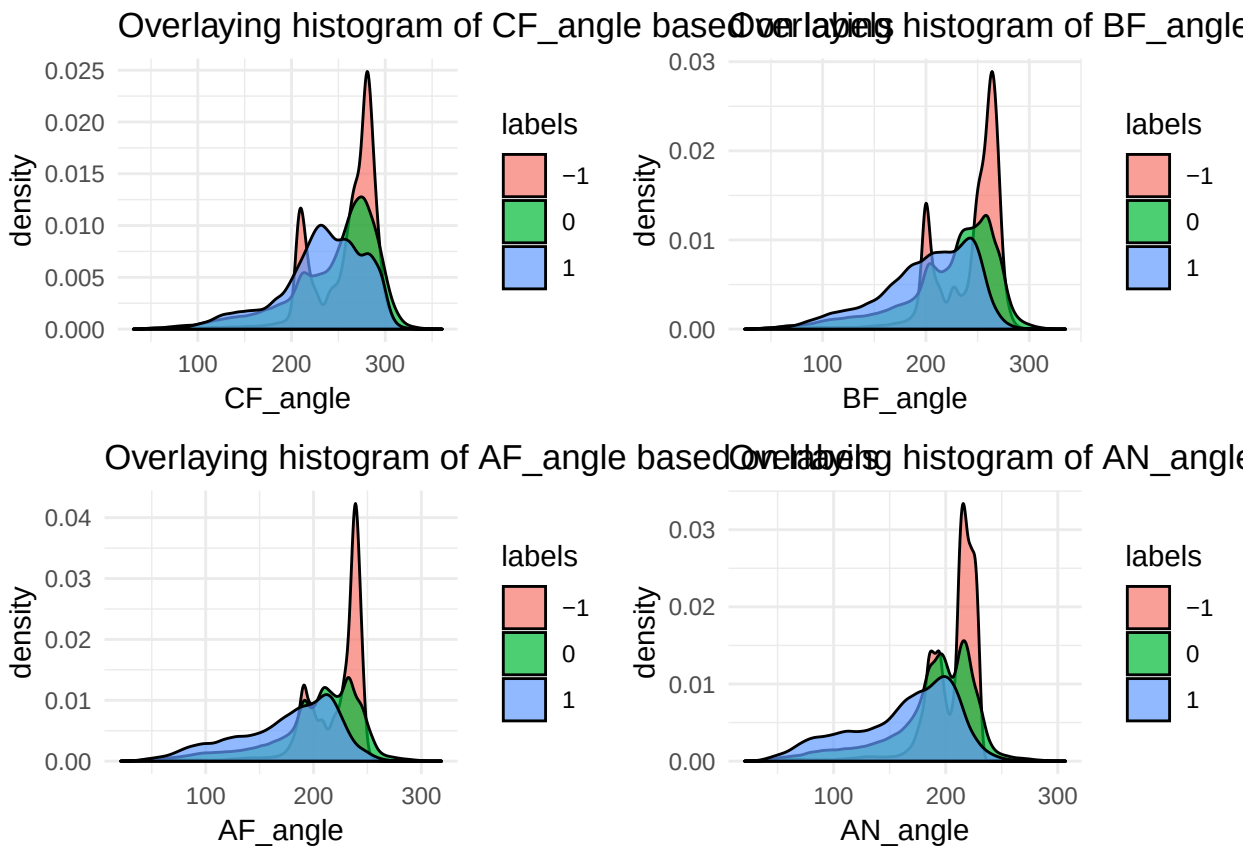
```
## BF_angle     -0.4311043  0.699107255  0.9119430  1.0000000  0.9529627
## AF_angle     -0.6039353  0.591095794  0.8216176  0.9529627  1.0000000
## AN_angle     -0.6820440  0.547609821  0.7727499  0.9043409  0.9706198
##                AN_angle
## y_coordinate  0.5156779
## x_coordinate  0.3832116
## label        -0.3893588
## NDAI         -0.4895267
## SD           -0.4466229
## CORR         -0.6820440
## DF_angle      0.5476098
## CF_angle      0.7727499
## BF_angle      0.9043409
## AF_angle      0.9706198
## AN_angle      1.0000000
```

```
corrplot.mixed(cor(total_df[,c(-3, -12)]))
```

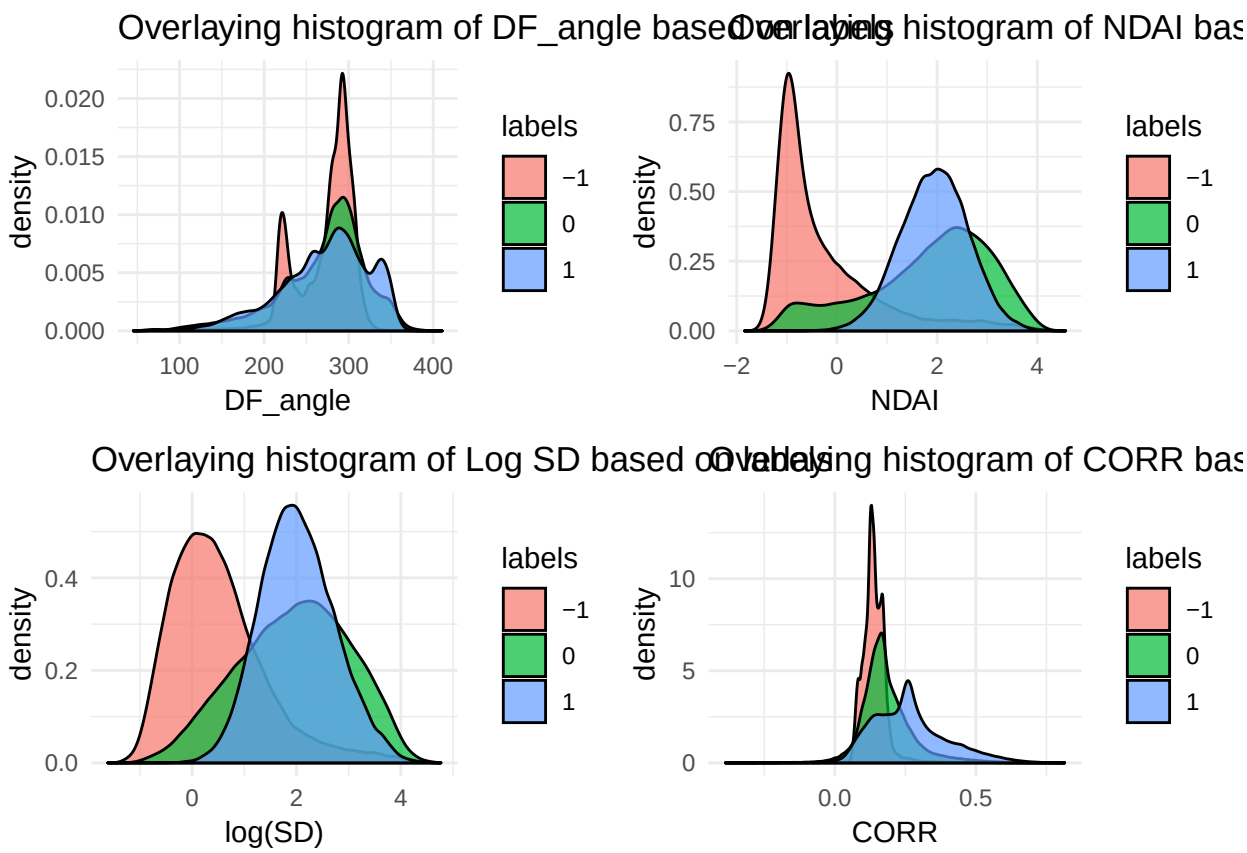



```
G<- ggplot(total_df)+geom_density(aes(x= NDAI, group= label, color=labels, fill= labels), color= "black
  ggtitle("Overlaying histogram of NDAI based on labels")+
  theme_minimal()

H<- ggplot(total_df)+geom_density(aes(x= log(SD), group= label, color=labels, fill= labels), color= "bla
  ggtitle("Overlaying histogram of Log SD based on labels")+
  theme_minimal()

I<- ggplot(total_df)+geom_density(aes(x= CORR, group= label, color=labels, fill= labels), color= "black
```

```
  ggtitle("Overlaying histogram of CORR based on labels")+
  theme_minimal()

A<- ggplot(total_df)+geom_density(aes(x= DF_angle, group= label, color=labels, fill= labels), color= "bl
  ggtitle("Overlaying histogram of DF_angle based on labels")+
  theme_minimal()

B<- ggplot(total_df)+geom_density(aes(x= CF_angle, group= label, color=labels, fill= labels), color= "bl
  ggtitle("Overlaying histogram of CF_angle based on labels")+
  theme_minimal()

C<- ggplot(total_df)+geom_density(aes(x= BF_angle, group= label, color=labels, fill= labels), color= "bl
  ggtitle("Overlaying histogram of BF_angle based on labels")+
  theme_minimal()

D<-ggplot(total_df)+geom_density(aes(x= AF_angle, group= label, color=labels, fill= labels), color= "bla
  ggtitle("Overlaying histogram of AF_angle based on labels")+
  theme_minimal()

E<- ggplot(total_df)+geom_density(aes(x= AN_angle, group= label, color=labels, fill= labels), color= "bl
  ggtitle("Overlaying histogram of AN_angle based on labels")+
  theme_minimal()

plot_grid(B,C,D,E,nrow=2, align="h")
```



```
plot_grid(A,G,H,I,nrow=2, align="h")
```



Problem 2a Data Split

```
# Filter all the undefined label
image1<- image1 %>% filter(label !=0)
image2<- image2 %>% filter(label !=0)
image3<- image3 %>% filter(label !=0)
```

Split method 1 By label

```
traintest_split<- function(data){
  res<-list()
  trainIndex <- createDataPartition(data$label, p = .8,
                                    list = FALSE,
                                    times = 1)
  res$train<- data[ trainIndex,]
  res$test<-  data[-trainIndex,]
  return(res)
}
trainval_split<- function(data){
  res<-list()
  valIndex <- createDataPartition(data$label, p = .2,
                                    list = FALSE,
                                    times = 1)
```

```
  res$val<- data[ valIndex,]
  res$train<-  data[-valIndex,]
  return(res)
}

##image 1
#Test-train split
train1_label_split1<- traintest_split(image1)$train
# get test from train-test split
test_label_split1<- traintest_split(image1)$test

#Train-val split ---get val
val_label_split1<- trainval_split(train1_label_split1)$val
#Get train
train_label_split1<-trainval_split(train1_label_split1)$train

##image 2
train2_label_split2<- traintest_split(image2)$train
# get test from train-test split
test_label_split2<- traintest_split(image2)$test
#Train-val split ---get val
val_label_split2<- trainval_split(train2_label_split2)$val
#Get train
train_label_split2<-trainval_split(train2_label_split2)$train

##image 3
train3_label_split3<- traintest_split(image3)$train
# get test from train-test split
test_label_split3<- traintest_split(image3)$test

#Train-val split ---get val
val_label_split3<- trainval_split(train3_label_split3)$val
#Get train
train_label_split3<-trainval_split(train3_label_split3)$train

method1_train<-rbind(train_label_split1,train_label_split2,train_label_split3)
method1_val<-rbind(val_label_split1,val_label_split2,val_label_split3)
method1_test<- rbind(test_label_split1,test_label_split2,test_label_split3)
method1_train_val<-rbind(method1_train,method1_val)
method1_train_val$label<-factor(method1_train_val$label)
```

Split method 2 By block

```
#Check Duplicates
image1 %>% distinct()
grid_row <- 10
grid_col <- 10
datasplit<- function(grid_row, grid_col, image){
  ret <- list()
  train_data <- data.frame()
  val_data <- data.frame()
  test_data <- data.frame()
  min_y_cor <- min(image$y_coordinate)
```

```
  min_x_cor <- min(image$x_coordinate)
  max_y_cor <- max(image$y_coordinate)
  max_x_cor <- max(image$x_coordinate)
  divide_row <- seq(min_y_cor,max_y_cor+1,(max_y_cor+1-min_y_cor)/(grid_row-1))
  divide_col <- seq(min_x_cor,max_x_cor+1,(max_x_cor+1-min_x_cor)/(grid_col-1))
  for (i in 1:length(divide_row)){
    for (j in 1:length(divide_col)){
      chunk <- image[image$y_coordinate>=divide_row[i] & image$y_coordinate<divide_row[i+1] &
                       image$x_coordinate>=divide_col[j] & image$x_coordinate<divide_col[j+1], ]
      test_and_val <- floor(nrow(chunk)*0.4)
      test_and_val_ind <- sample(seq_len(nrow(chunk)), size = test_and_val)
      val_size <- floor(test_and_val/2)
      val_ind_ind <- sample(length(test_and_val_ind), size = val_size)
      val_ind <- test_and_val_ind[val_ind_ind]
      test_ind <- test_and_val_ind[-val_ind_ind]
      test<- chunk[test_ind, ]
      val<- chunk[val_ind, ]
      train<- chunk[-test_and_val_ind, ]
      train_data = rbind(train_data,train)
      test_data = rbind(test_data,test)
      val_data = rbind(val_data,val)
    }
  }
  ret$training <- train_data
  ret$validation <- val_data
  ret$test <- test_data
  return(ret)
}

split_result_1 <- datasplit(10,10,image1)
split_result_2 <- datasplit(10,10,image2)
split_result_3 <- datasplit(10,10,image3)
train_total<- rbind(split_result_1$training,split_result_2$training,split_result_3$training)
val_total<- rbind(split_result_1$validation,split_result_2$validation,split_result_3$validation)
test_total<- rbind(split_result_1$test,split_result_2$test,split_result_3$test)
train_total$picture<- NULL
val_total$picture<- NULL
test_total$picture<-NULL
train_total$label<-factor(train_total$label)
```

Problem 2b Baseline

```
test_trivial<- test_total
test_trivial$label<--1
train_trivial<- train_total
train_trivial$label<--1
val_trivial<- val_total
val_trivial$label<--1
sum(test_trivial$label ==test_total$label)/length(test_total$label)
```

```
## [1] 0.6103902
```

```
sum(val_trivial$label ==val_total$label)/length(val_trivial$label)
```

```
## [1] 0.6104622
```

```
sum(train_trivial$label ==train_total$label)/length(train_trivial$label)
```

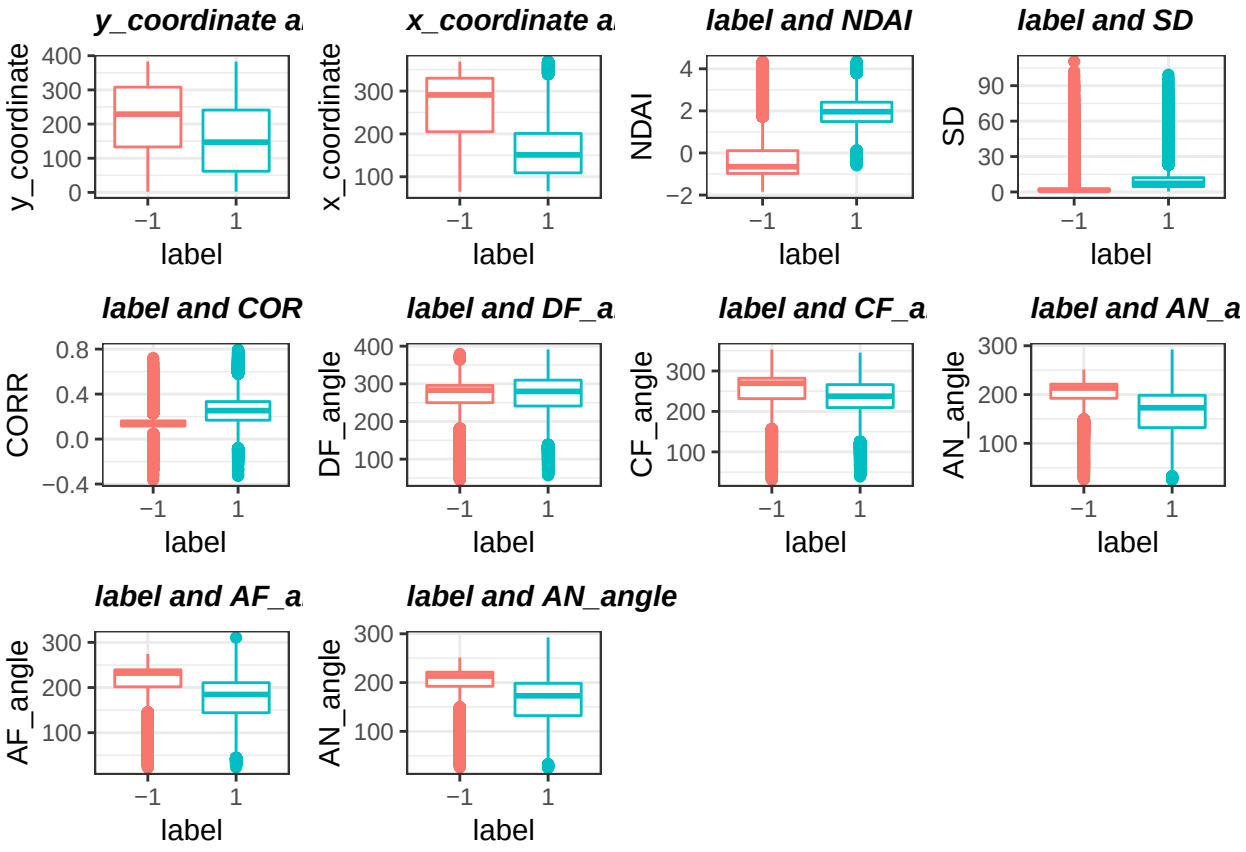
```
## [1] 0.6110293
```

Problem 2c First Order Importance

Visualization

```
train_val<- rbind(train_total, val_total)
train_val$label<- as.factor(train_val$label)
test_total$label<- as.factor(test_total$label)
#visualization by using boxplot
A<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= y_coordinate, color= factor(label)))+ theme_bw
  ggtitle("y_coordinate and label")+xlab("label")+
  theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))+
  theme(legend.position="none")

B<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= x_coordinate,color= factor(label)))+ theme_bw
  ggtitle("x_coordinate and label")+theme(legend.position="none")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

C<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= NDAI,color= factor(label)))+ theme_bw()+
  ggtitle("label and NDAI")+theme(legend.position="none")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

D<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= SD,color= factor(label)))+ theme_bw()+
  ggtitle("label and SD")+theme(legend.position="none")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

E<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= CORR,color= factor(label)))+ theme_bw()+
  ggtitle("label and CORR")+theme(legend.position="none")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

G<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= DF_angle,color= factor(label)))+ theme_bw()+
  theme(legend.position="none")+
  ggtitle("label and DF_angle")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

H<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= CF_angle,color= factor(label)))+ theme_bw()+
  theme(legend.position="none")+
  ggtitle("label and CF_angle")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

I<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= BF_angle,color= factor(label)))+ theme_bw()+
  theme(legend.position="none")+
  ggtitle("label and BF_angle")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

J<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= AF_angle,color= factor(label)))+ theme_bw()+
  theme(legend.position="none")+
```

```
  ggtitle("label and AF_angle")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

K<-I<-ggplot(train_val)+geom_boxplot(aes(x=factor(label), y= AN_angle,color= factor(label)))+ theme_bw()
  ggtitle("label and AN_angle")+theme(legend.position="none")+
  xlab("label")+theme(plot.title = element_text(size = rel(1), vjust = 1.5,face="bold.italic"))

plot_grid(A, B,C,D,E,G,H,I,J,K,nrow=3, align="h")
```



Quantitative

```
summary(lm(as.numeric(label)~NDAI, data = train_val))$r.squared
```

```
## [1] 0.5754858
```

```
summary(lm(as.numeric(label)~CORR, data = train_val))$r.squared
```

```
## [1] 0.3040832
```

```
summary(lm(as.numeric(label)~AF_angle, data = train_val))$r.squared
```

```
## [1] 0.2578005
```

```
summary(lm(as.numeric(label)~x_coordinate, data = train_val))$r.squared
```

```
## [1] 0.3256973
```

```
summary(lm(as.numeric(label)~y_coordinate, data = train_val))$r.squared
```

```
## [1] 0.07886448
```

```
summary(lm(as.numeric(label)~SD, data = train_val))$r.squared
```

```
## [1] 0.1902478
```

```
summary(lm(as.numeric(label)~DF_angle, data = train_val))$r.squared
```

```
## [1] 0.0001123623
```

```
summary(lm(as.numeric(label)~CF_angle, data = train_val))$r.squared
```

```
## [1] 0.07964199
```

```
summary(lm(as.numeric(label)~AN_angle, data = train_val))$r.squared
```

```
## [1] 0.255197
```

Problem 3a

Model 1: Logistic Regression

```
# Compute CV loss
seed = 123
K = 5
source("CVgeneric.R")
cv_result <- CVgeneric("logistic", c("y_coordinate", "x_coordinate", "NDAI", "SD", "CORR", "DF_angle",
                       K, data=train_val, loss= accuracy, seed)
set.seed(cv_result$seed)
cv_result_method_2 <- CVgeneric("logistic", c("y_coordinate", "x_coordinate", "NDAI", "SD", "CORR", "DF
                                K, data=method1_train_val, loss= accuracy, seed)
# Use data based on the best folds
train_data = train_val[-cv_result$index, ]
# Train logistic model
logistic_model<- train(label ~ .,  data=train_data, method="glm", family="binomial")
# Test accuracy
predicted.classes <- logistic_model %>% predict(test_total[,-3])
mean(predicted.classes == test_total$label)
```

```
## [1] 0.8962178
```

```
# Other error metrics
# F1 Score
F1_Score(as.numeric(predicted.classes), as.numeric(test_total$label))
```

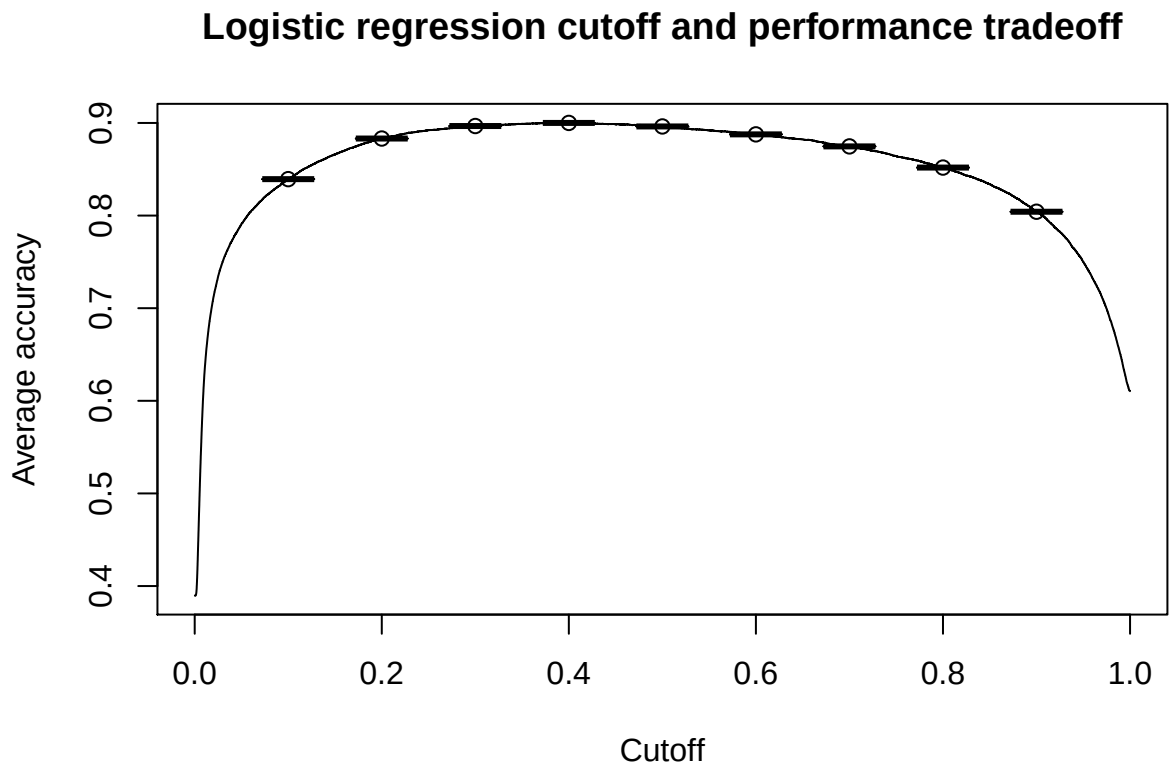
```
## [1] 0.9146999
```

```
# Sensitivity
Sensitivity(as.numeric(predicted.classes), as.numeric(test_total$label))
```

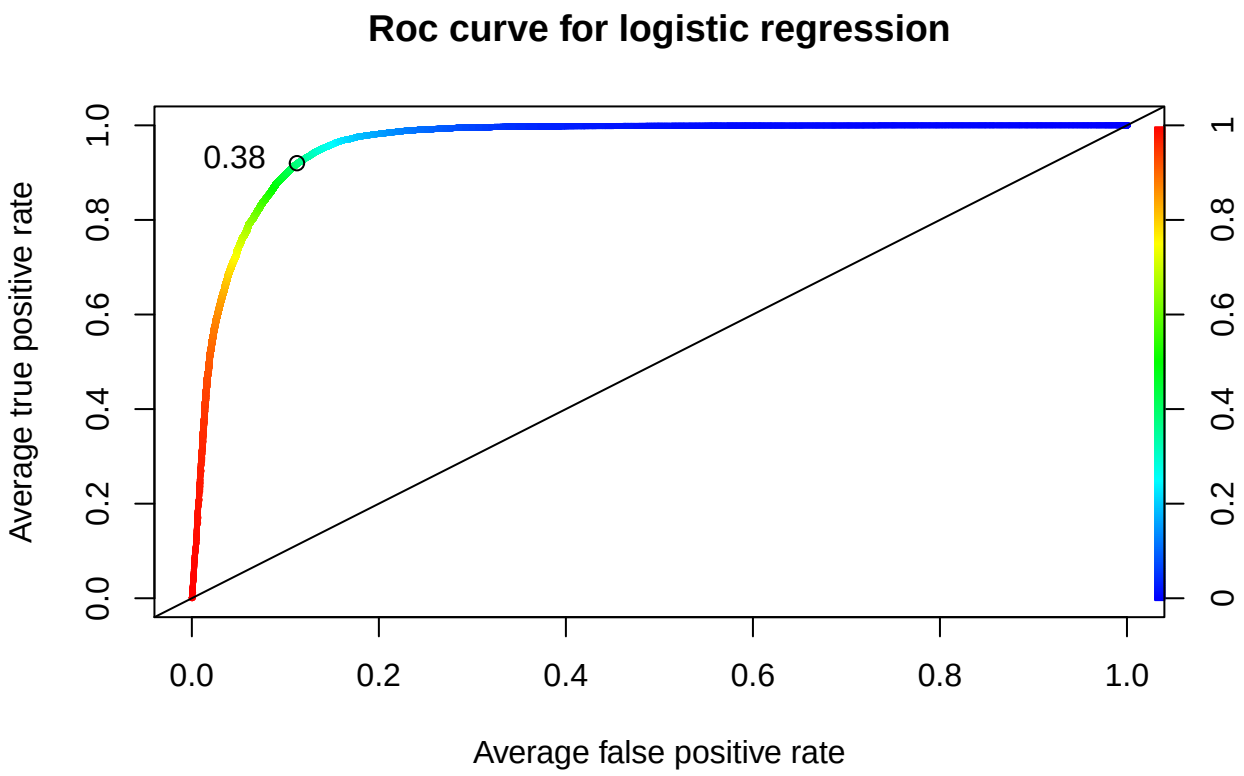
```
## [1] 0.9177995
```

```
# Find the best cut off point
predictions<- data.frame(predict(logistic_model, newdata =test_total[,-3],
                                 type= "prob" ))
colnames(predictions) <- c(-1,1)
pred <- prediction(predictions$`1`, test_total$label)
perf <- performance(pred, "acc")
plot(perf, avg= "vertical", spread.estimate="boxplot",
```

```
     show.spread.at= seq(0.1, 0.9, by=0.1),
     main="Logistic regression cutoff and performance tradeoff")
```




```
index<-which.max(slot(perf, "y.values")[[1]])
max<-slot(perf, "x.values")[[1]][index]

perf <- performance(pred, "tpr", "fpr")
# ROC curve plot
plot(perf, colorize=T, print.cutoffs.at=c(max),
     text.adj=c(1.5,0.2),
     avg="threshold", lwd=3,
     main= "Roc curve for logistic regression")
abline(a=0,b=1)
```



Model 2: LDA

```
# Compute CV loss
cv_result_lda<-CVgeneric("lda", c("y_coordinate", "x_coordinate", "NDAI", "SD", "CORR", "DF_angle", "CF
                         K = 5, train_val, loss = precision, seed)
cv_result_lda_method_2<-CVgeneric("lda", c("y_coordinate", "x_coordinate", "NDAI", "SD", "CORR", "DF_ang
                        K = 5, method1_train_val, loss = precision, seed)
# Use data based on the best folds
train_data = train_val[-cv_result_lda$index, ]
# Fit LDA Model
lda.model = lda(factor(label)~., data=train_val)
lda_pred<-predict(lda.model, newdata=test_total[,-3])
lda_pre2<-predict(lda.model, newdata=test_total[,-3], type=  "prob")

#Other error metrics
#F1 Score
F1_Score(as.numeric(lda_pred$class), as.numeric(test_total$label))
```

```
## [1] 0.9161555
```

```
#Sensitivity
Sensitivity(as.numeric(lda_pred$class), as.numeric(test_total$label))
```

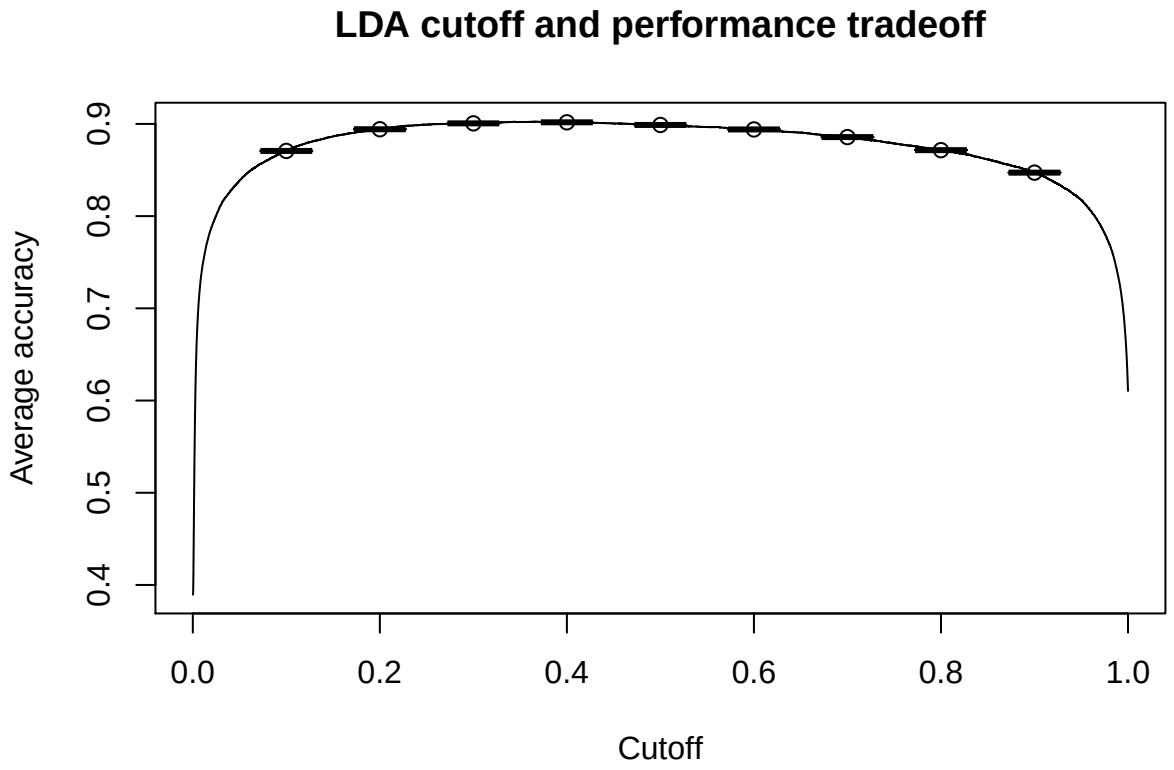
```
## [1] 0.9270859
```

```
#Accuracy of the model
mean(lda_pred$class == test_total$label)
```

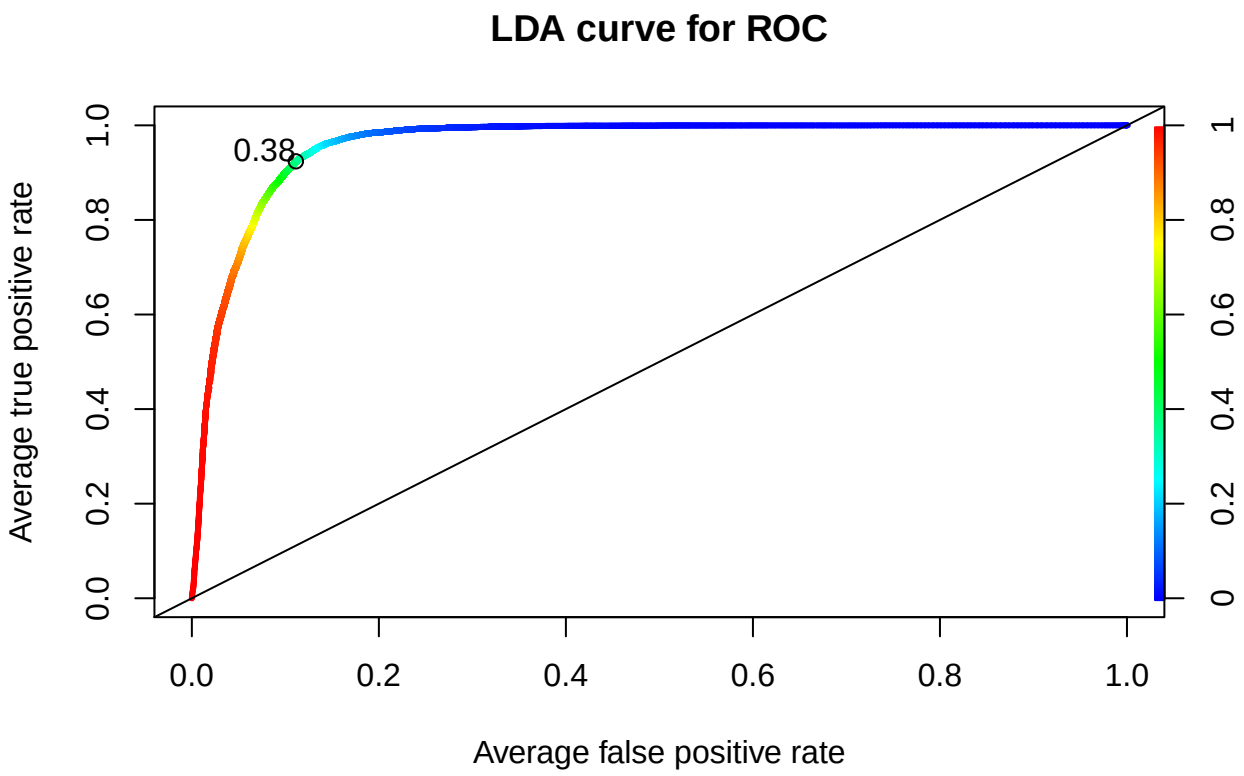
```
## [1] 0.898837
```

```
# Find the best cut off point
predictions<- data.frame(predict(lda.model, newdata =test_total[,-3] ))
colnames(predictions) <- c("label",-1,1)
```

```
pred <- prediction(predictions$`1`, test_total$label)
perf <- performance(pred, "acc")
plot(perf, avg= "vertical", spread.estimate="boxplot", show.spread.at= seq(0.1, 0.9, by=0.1), main="LDA
```

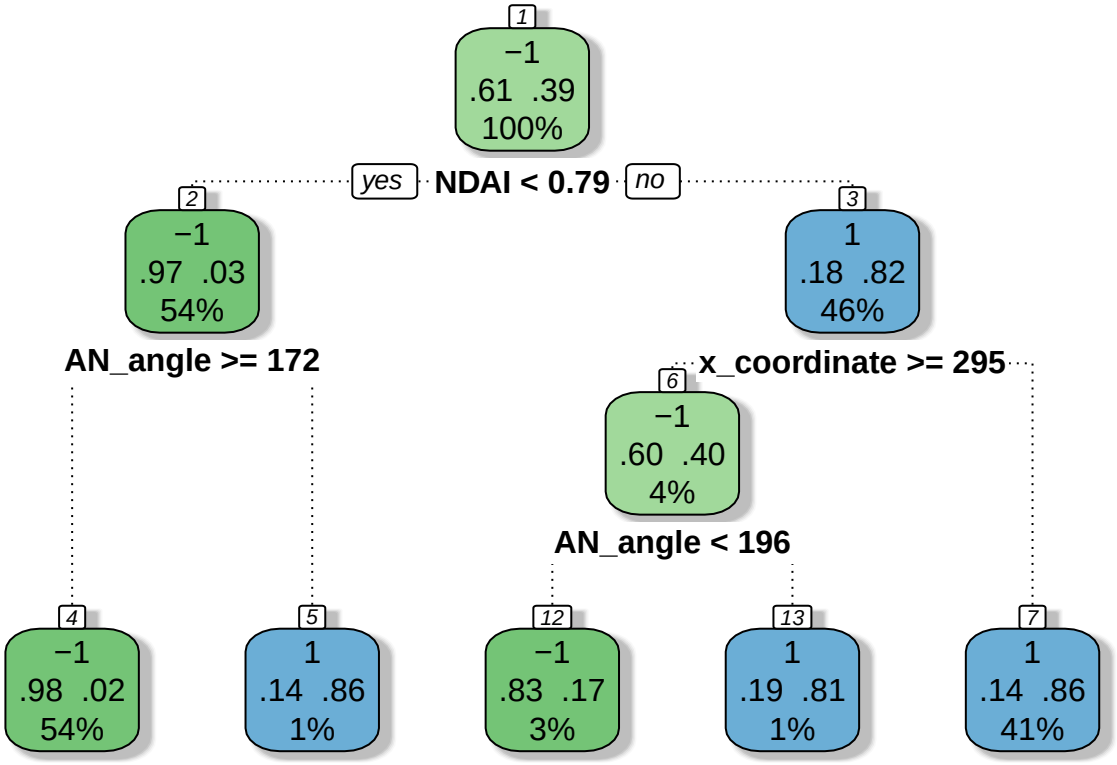



```
index<-which.max(slot(perf, "y.values")[[1]])
max<-slot(perf, "x.values")[[1]][index]
# ROC Curve
perf <- performance(pred, "tpr", "fpr")
plot(perf, colorize=T, print.cutoffs.at=c(max), text.adj=c(1,0), avg="threshold", lwd=3, main= "LDA cur
abline(a=0,b=1)
```



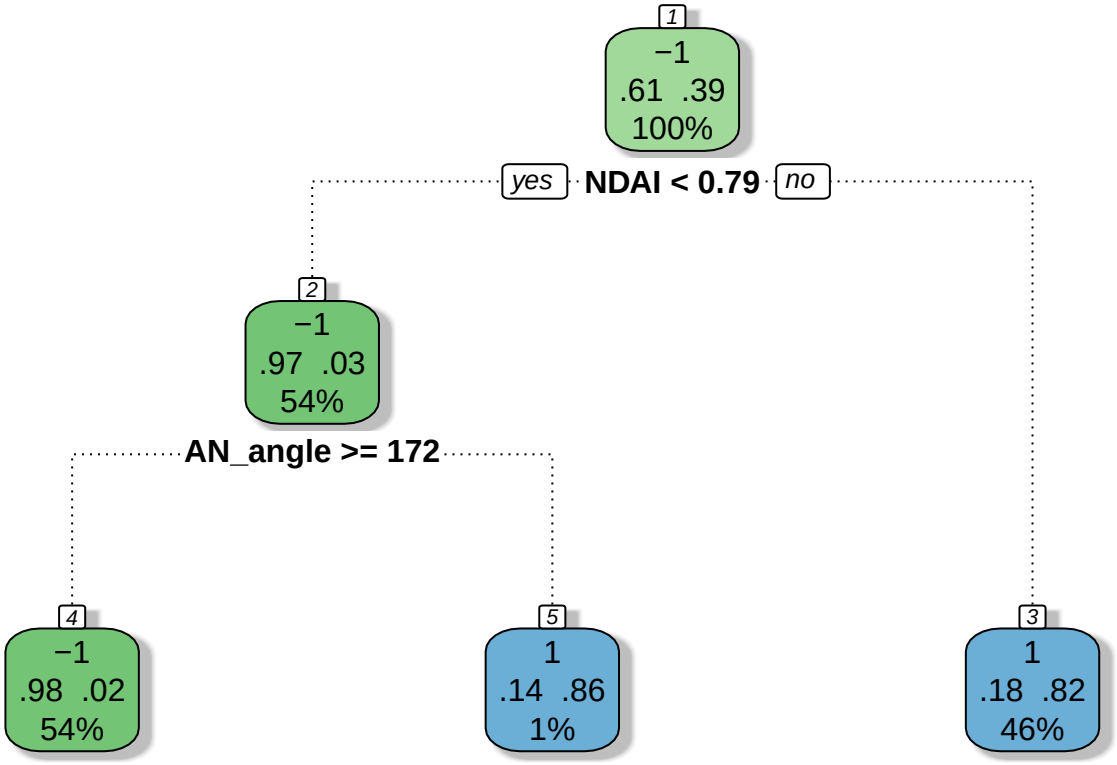
Model 3: Decision Tree

```
# Delete unused column
method1_train_val$picture<-NULL
# Compute CV loss for first method split
seed = 123
K = 5
cv_result_decision_tree <- CVgeneric("decision tree", c("y_coordinate", "x_coordinate", "NDAI", "SD", "
# Use data based on the best folds for the first method split
train_data = train_val[-cv_result_decision_tree$index, ]

# Train the decision tree model
tree = rpart(label ~ ., data=train_data,maxdepth =10, minsplit= 10)
# Visualize the decision tree
fancyRpartPlot(tree)
```



Rattle 2019−May−01 21:34:10 sunsgne

```
# Train the decision tree model without the geographic features
tree_no_xy = rpart(label ~ NDAI+SD+CORR+DF_angle+CF_angle+BF_angle+AF_angle+AN_angle , data=train_data,
# Visualize the decision tree model without the geographic features
fancyRpartPlot(tree_no_xy)
```



Rattle 2019−May−01 21:34:13 sunsgne

```
# Accuracy for decision tree model without the geographic features
tree.pred_no_xy = predict(tree_no_xy, newdata=test_total[,-3],type = 'class')
mean(tree.pred_no_xy==test_total$label)
```

```
## [1] 0.9052768
```

```
# Accuracy for  decision tree model WITH the geographic features
tree.pred = predict(tree, newdata=test_total[,-3],method = 'response')
tree.pred2 = predict(tree, newdata=test_total[,-3],type = 'class')
mean(tree.pred2==test_total$label)
```

```
## [1] 0.9232026
```

```
#Other error metrics
#F1 Score
F1_Score(as.numeric(tree.pred2), as.numeric(test_total$label))
```

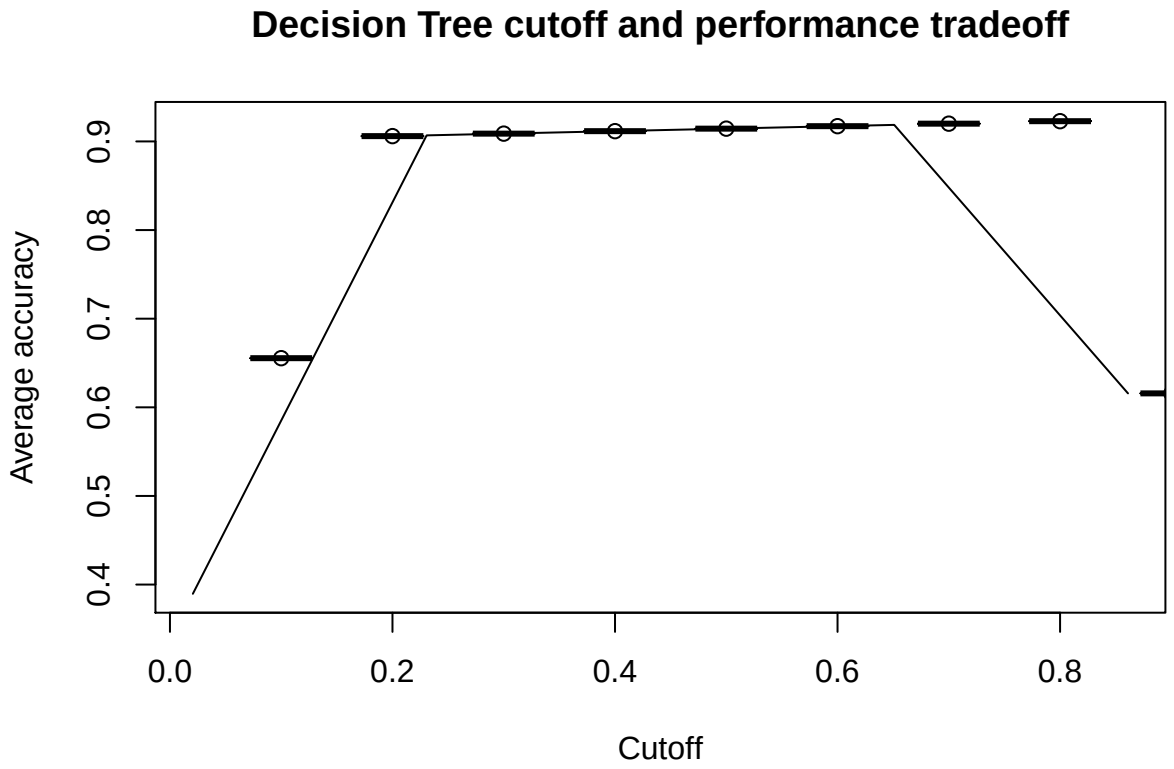
```
## [1] 0.9347036
```

```
#Sensitivity
Sensitivity(as.numeric(tree.pred2), as.numeric(test_total$label))
```

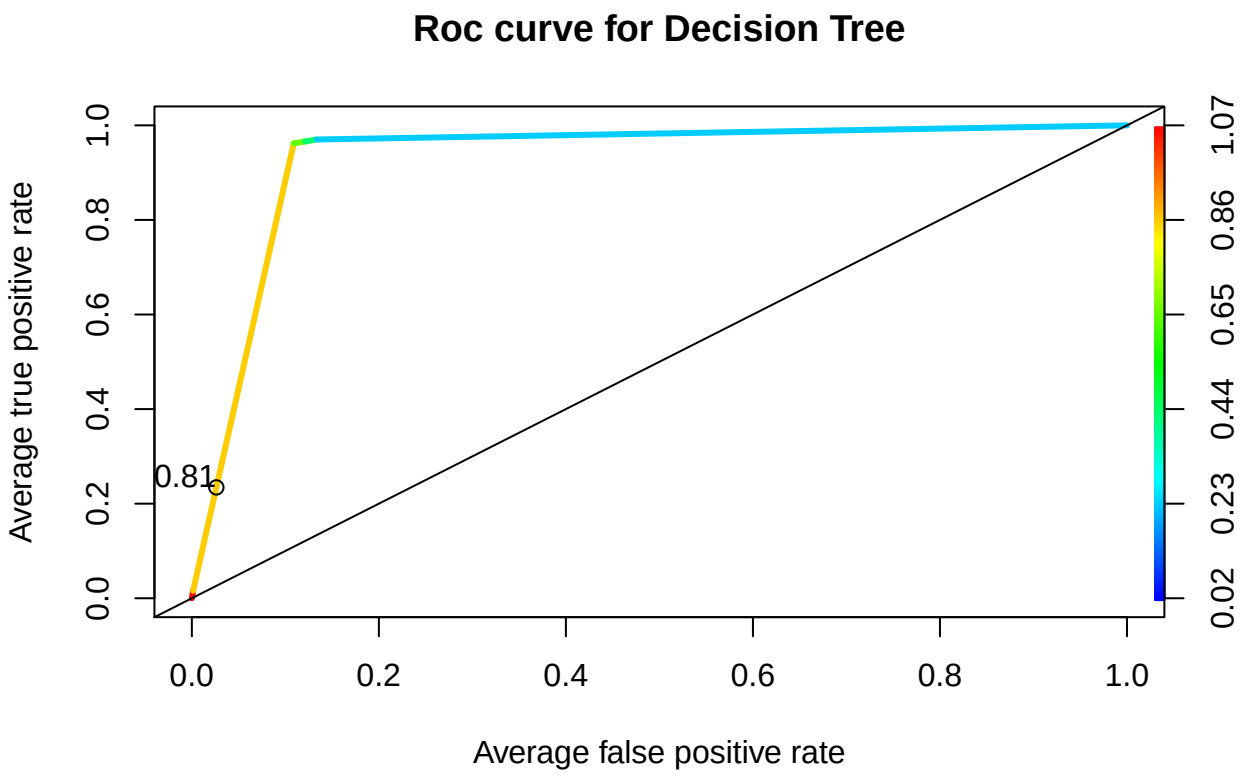
```
## [1] 0.9715851
```

```
# Find the best cut off point
predictions<- data.frame(predict(tree, newdata =test_total[,-3], type= "prob" ))
colnames(predictions) <- c(-1,1)
pred <- prediction(predictions$`1`, test_total$label)
perf <- performance(pred, "acc")
plot(perf, avg= "vertical", spread.estimate="boxplot",
     show.spread.at= seq(0.1, 0.9, by=0.1),
```

```
     main="Decision Tree cutoff and performance tradeoff")
```




```
index<-which.max(slot(perf, "y.values")[[1]])
max<-slot(perf, "x.values")[[1]][index]
perf <- performance(pred, "tpr", "fpr")
# ROC Curve
plot(perf, colorize=T, print.cutoffs.at=max, text.adj=c(1,0), avg="threshold",
     lwd=3,
     main= "Roc curve for Decision Tree")
abline(a=0,b=1)
```



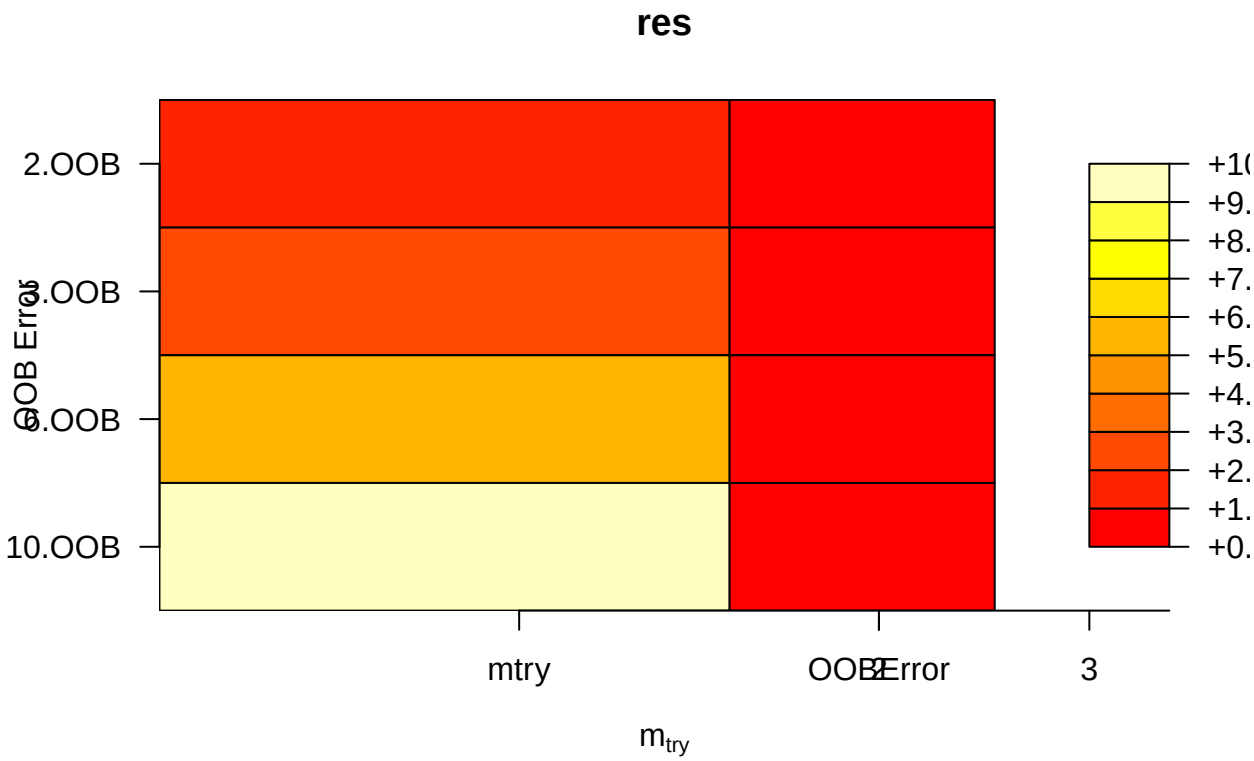

```
# Decision Tree feature importance (4a preparation)
tree_importance<-as.data.frame(varImp(tree))
tree_importance$importance<-rownames(tree_importance)
tree_importance <- tree_importance %>% arrange(desc(Overall))

# Compute CV loss for second method data split
cv_result_decision_tree_method2 <- CVgeneric("decision tree", c("y_coordinate", "x_coordinate", "NDAI",
train_data_tree_2 = method1_train_val[-cv_result_decision_tree_method2$index, ]
tree2 = rpart(label ~ ., data=train_data_tree_2,maxdepth =10, minsplit= 10)
tree.pred_method_2 <- predict(tree2, newdata=test_total[,-3],type = 'class')
```

Model 4:Random Forest

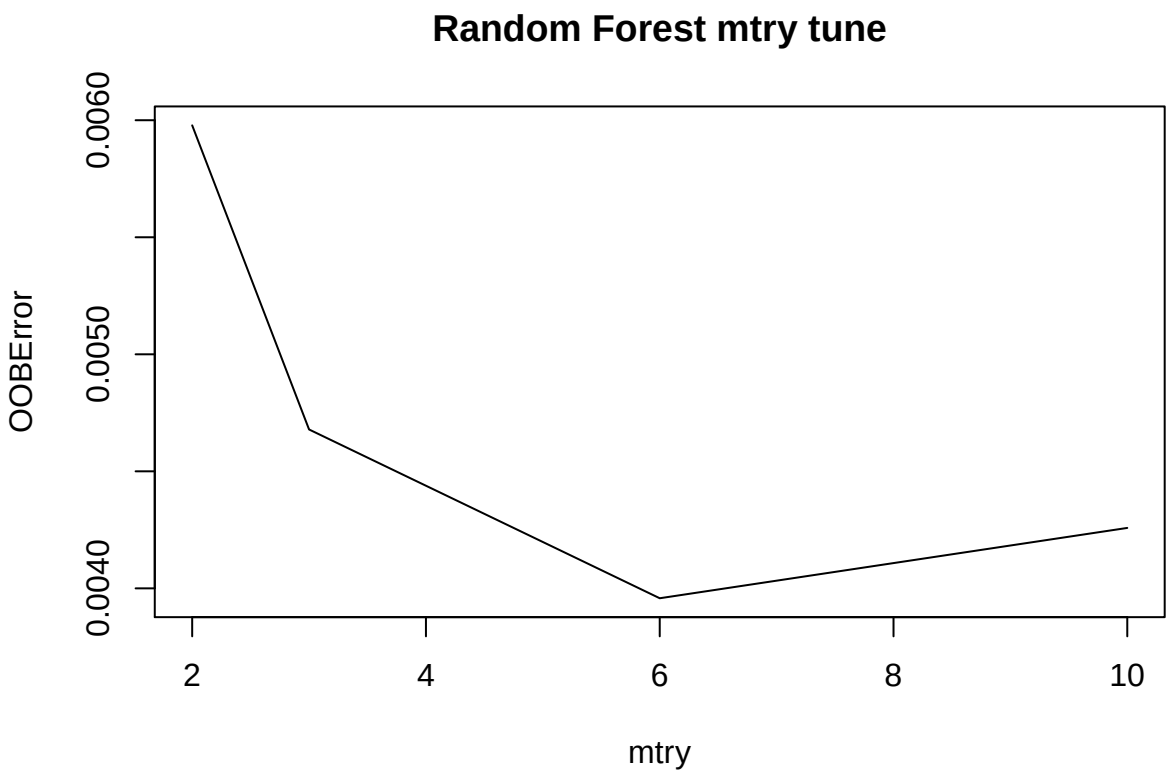
```
seed = 123
K = 5
# Compute CV loss for first method split
cv_result_rf <- CVgeneric("random forest", c("y_coordinate", "x_coordinate", "NDAI", "SD", "CORR", "DF_
# Use the best folds on the training set
train_data_rf = train_val[-cv_result_rf$index, ]
# Choose the best hyperparameter of ntry which is the Number of variables randomly sampled as candidate
Random_forest_hyperparameter_tune<-tuneRF(x =train_data[,-3],
              y = as.factor(train_data[,3]),
              ntreeTry = 50)
```

```
## mtry = 3  OOB error = 0.47%
## Searching left ...
## mtry = 2     OOB error = 0.6%
## -0.2776886 0.05
## Searching right ...
```

```
## mtry = 6     OOB error = 0.4%
## 0.1540931 0.05
## mtry = 10    OOB error = 0.43%
## -0.07590133 0.05
```




```
# Plot the ntry
plot(data.frame(Random_forest_hyperparameter_tune)$mtry,
     data.frame(Random_forest_hyperparameter_tune)$OOBError,
     type = "l",ylab="OOBError",xlab="mtry",
     main = "Random Forest mtry tune")
```




```
#Hyperparameter in R for random Forest
#ntree: Number of trees to grow.

# Fit the random forest model
rf_model <- randomForest(as.factor(label) ~ y_coordinate + x_coordinate +
                           NDAI + SD + CORR + DF_angle + CF_angle + BF_angle +
                           AF_angle + AN_angle, data = train_data_rf,
                         importance = TRUE,
                         ntree=50,
                         ntry=6,
                         maxnodes= 500,
                         nodesize = 10)
# Compute the accuracy of the model
predicted.classes_rf <- rf_model %>% predict(test_total[,-3])
mean(predicted.classes_rf == test_total$label)
```

```
## [1] 0.9863995
```

```
#Other error metrics
#F1 Score
F1_Score(as.numeric(predicted.classes_rf), as.numeric(test_total$label))
```

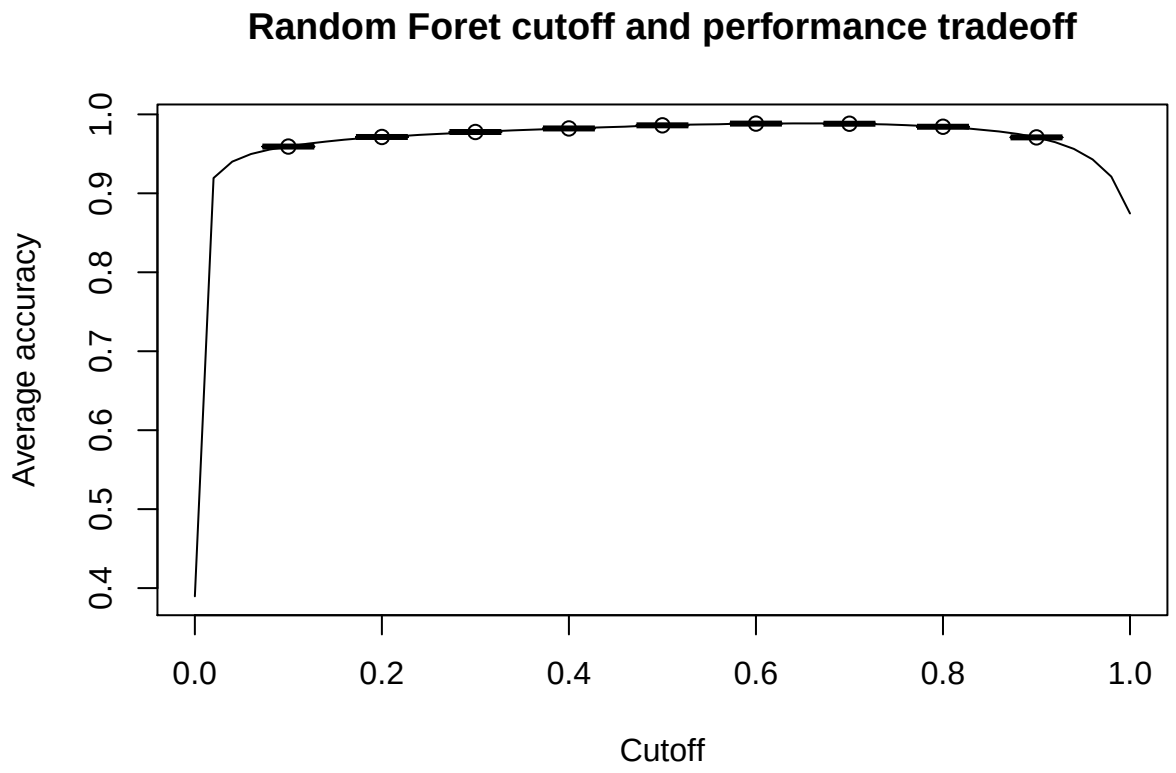
```
## [1] 0.988793
```

```
#Sensitivity
Sensitivity(as.numeric(predicted.classes_rf), as.numeric(test_total$label))
```

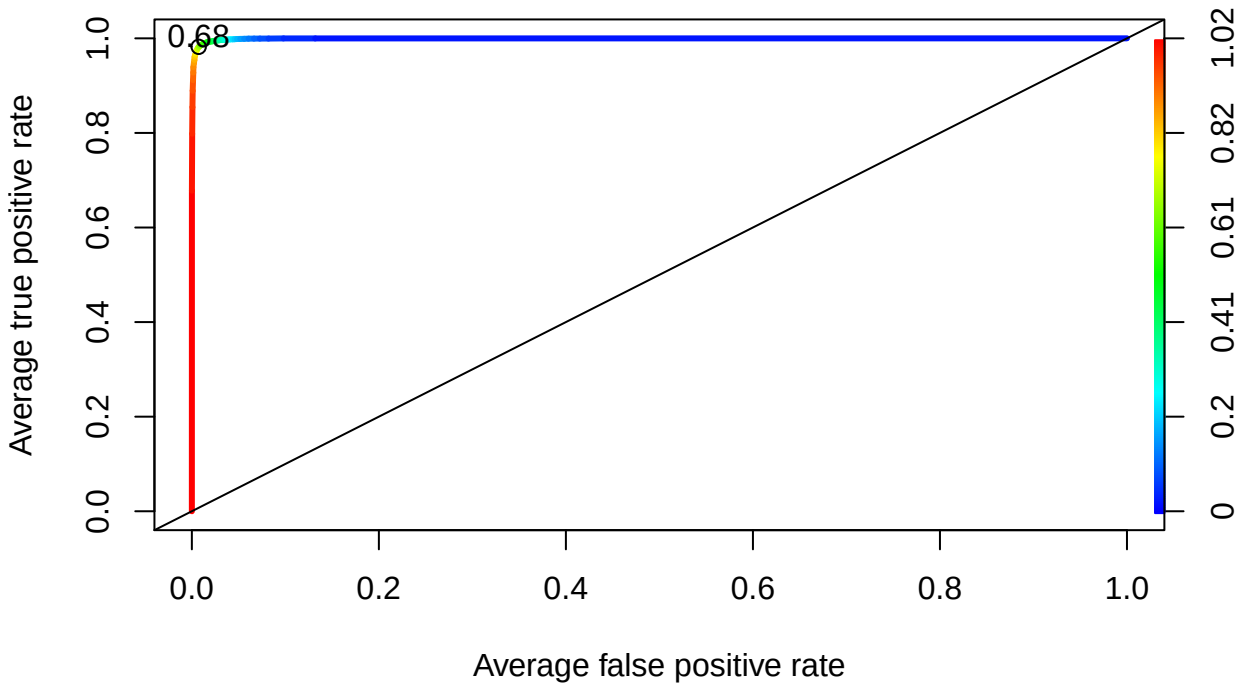
```
## [1] 0.9947016
```

```
# Find the best cut off point
predictions<- data.frame(predict(rf_model, newdata =test_total[,-3], type= "prob" ))
colnames(predictions) <- c(-1,1)
```

```
pred <- prediction(predictions$`1`, test_total$label)
perf <- performance(pred, "acc")
plot(perf, avg= "vertical", spread.estimate="boxplot",
     show.spread.at= seq(0.1, 0.9, by=0.1),
     main="Random Foret cutoff and performance tradeoff")
```




```
index<-which.max(slot(perf, "y.values")[[1]])
max<-slot(perf, "x.values")[[1]][index]
perf <- performance(pred, "tpr", "fpr")
# ROC curve
plot(perf, colorize=T, print.cutoffs.at=c(max), text.adj=c(0.5,0),
     avg="threshold", lwd=3, main= "Roc curve for Random Forest")
abline(a=0,b=1)
```




```
# Feature importance for random forest model
varImp(rf_model)
```

```
##                      -1        1
## y_coordinate 23.376656 23.376656
## x_coordinate 26.100630 26.100630
## NDAI         25.906828 25.906828
## SD            9.313817  9.313817
## CORR         18.226402 18.226402
## DF_angle     11.381089 11.381089
## CF_angle      8.343830  8.343830
## BF_angle      6.510444  6.510444
## AF_angle      7.628433  7.628433
## AN_angle      8.985508  8.985508
```

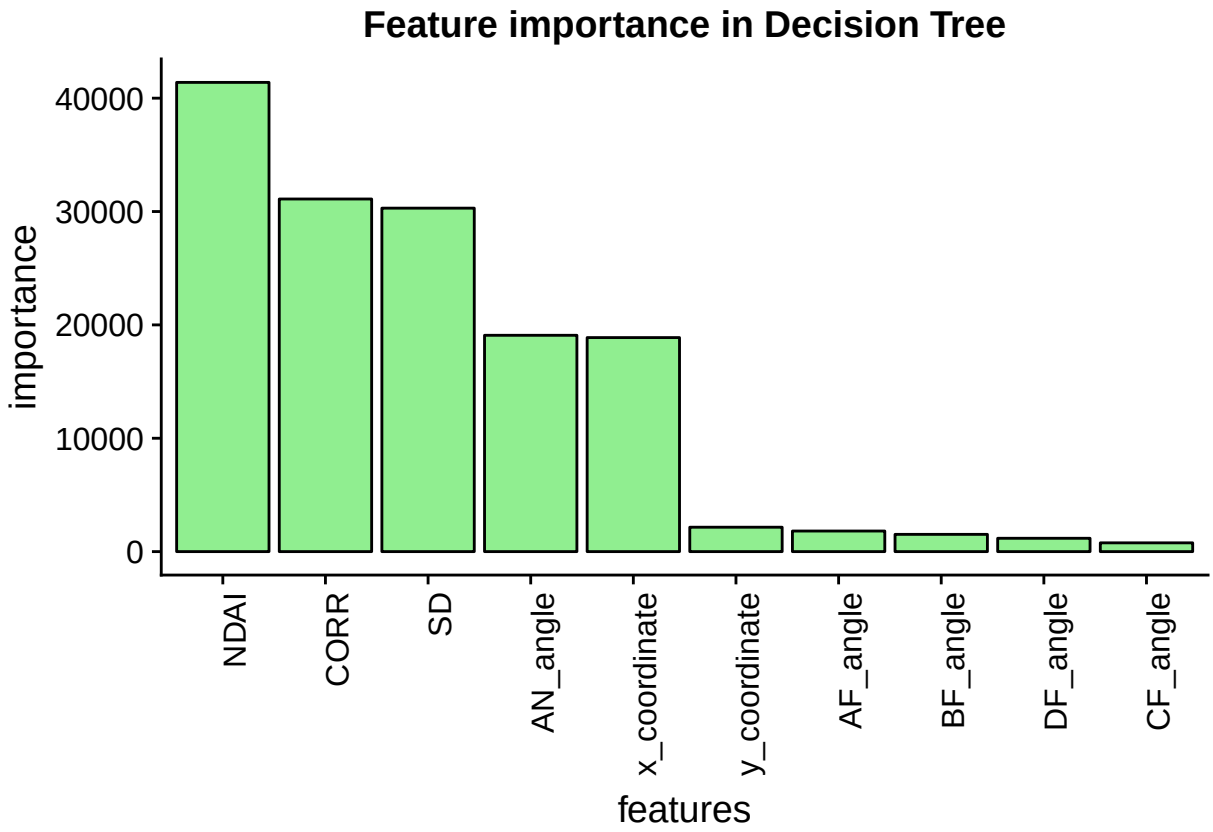
```
#CV method 2
cv_result_rf_method2 <- CVgeneric("random forest",
                                  c("y_coordinate", "x_coordinate", "NDAI", "SD",
                                    "CORR", "DF_angle", "CF_angle", "BF_angle",
                                    "AF_angle", 'AN_angle'),
                                  "label", K, data=method1_train_val,
                                  loss= accuracy,
                                  seed)
#Print accuracy result
#cv_result_rf_method2
#train data by using the best training sets
train_data_rf2 = method1_train_val[-cv_result_rf_method2$index, ]

rf_model_method_2 <- randomForest(as.factor(label) ~ y_coordinate +
                                    x_coordinate + NDAI + SD + CORR +
```

```
                                    DF_angle + CF_angle + BF_angle +AF_angle +
                                    AN_angle,
                                  data = train_data_rf2,
                                  importance = TRUE,ntree=50,
                                  ntry=6,maxnodes= 500, nodesize = 10)

predicted.classes_rf_method_2 <- rf_model_method_2 %>% predict(test_total[,-3])
```

Problem 4 Diagnostics

4a

```
# Decision Tree feature importance
ggplot(data=tree_importance, aes(x=reorder(importance, -Overall), y=Overall)) +
  geom_bar(stat="identity", color="black", fill="lightgreen")+
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  xlab("features")+
  ylab("importance")+
  ggtitle("Feature importance in Decision Tree")
```

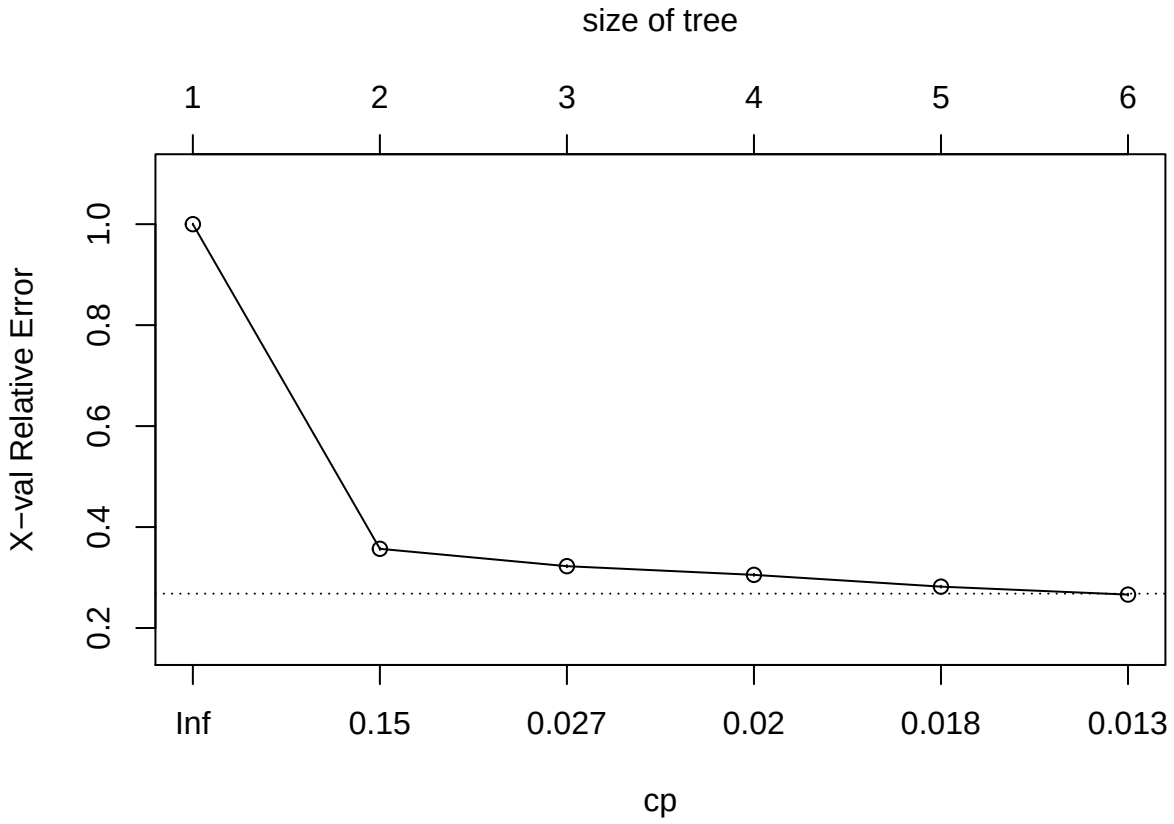



```
# Balance the cost complexity and error
fit <- rpart(label~.,
   method="anova", data=train_val)
printcp(fit) # display the results
```

```
##
## Regression tree:
```

```
## rpart(formula = label ~ ., data = train_val, method = "anova")
##
## Variables actually used in tree construction:
## [1] AN_angle     CORR         NDAI         x_coordinate
##
## Root node error: 39564/166443 = 0.2377
##
## n= 166443
##
##         CP nsplit rel error  xerror      xstd
## 1 0.643582      0   1.00000 1.00001 0.0011150
## 2 0.034698      1   0.35642 0.35694 0.0022024
## 3 0.020428      2   0.32172 0.32251 0.0021648
## 4 0.020398      3   0.30129 0.30532 0.0020456
## 5 0.015657      4   0.28089 0.28198 0.0018871
## 6 0.010000      5   0.26524 0.26619 0.0019180
```

```
plotcp(fit)
```




```
summary(fit)
```

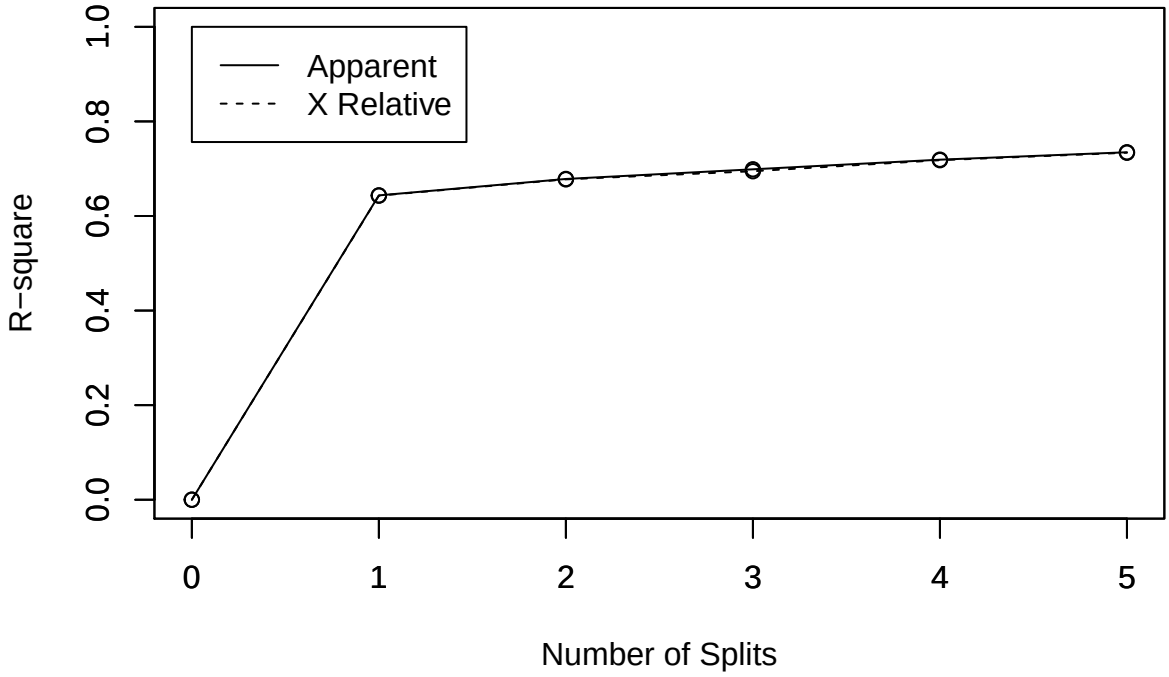
```
## Call:
## rpart(formula = label ~ ., data = train_val, method = "anova")
##   n= 166443
##
##           CP nsplit rel error    xerror        xstd
## 1 0.64358201      0 1.0000000 1.0000063 0.001114980
## 2 0.03469808      1 0.3564180 0.3569367 0.002202353
## 3 0.02042840      2 0.3217199 0.3225058 0.002164792
```

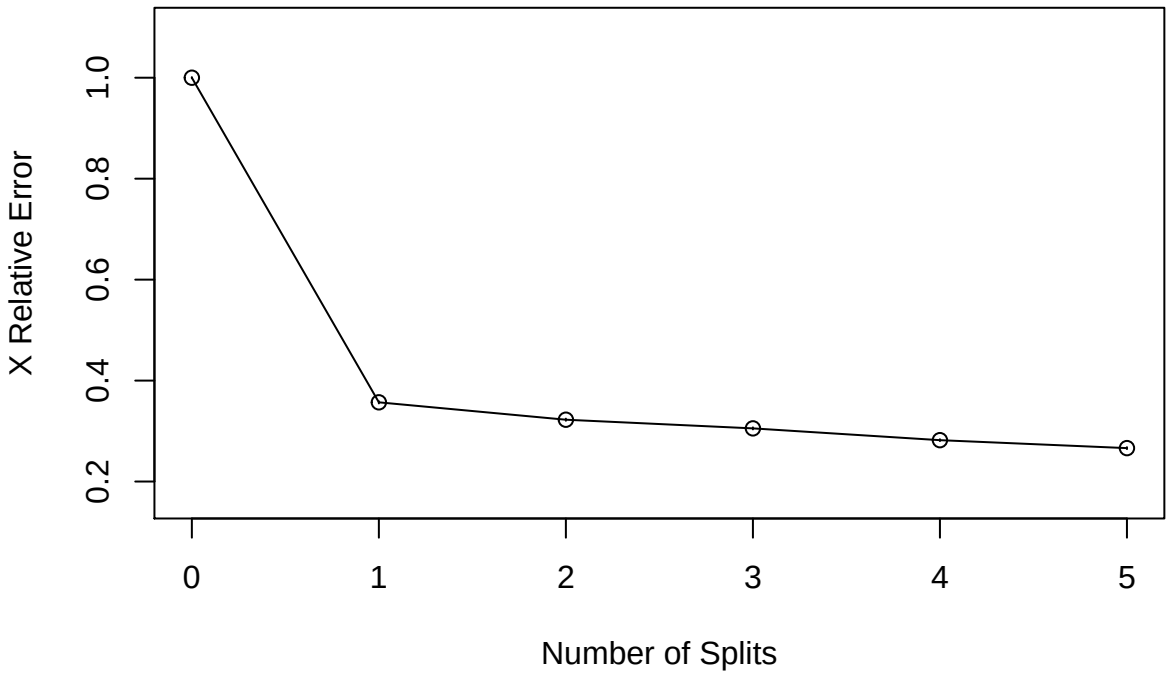
```
## 4 0.02039833      3 0.3012915 0.3053244 0.002045567
## 5 0.01565698      4 0.2808932 0.2819785 0.001887090
## 6 0.01000000      5 0.2652362 0.2661898 0.001918019
##
## Variable importance
##         NDAI           SD         CORR     AN_angle     AF_angle
##           25           18           15           15           13
##     BF_angle x_coordinate     CF_angle     DF_angle
##           12            1            1            1
##
## Node number 1: 166443 observations,    complexity param=0.643582
##   mean=1.389112, MSE=0.2377039
##   left son=2 (90658 obs) right son=3 (75785 obs)
##   Primary splits:
##       NDAI         < 0.7947012  to the left,  improve=0.6435820, (0 missing)
##       SD           < 3.092582   to the left,  improve=0.4652378, (0 missing)
##       CORR         < 0.197311   to the left,  improve=0.4474713, (0 missing)
##       x_coordinate < 229.5      to the right, improve=0.2648212, (0 missing)
##       AN_angle     < 177.6024   to the right, improve=0.2640935, (0 missing)
##   Surrogate splits:
##       SD       < 3.134235   to the left,  agree=0.867, adj=0.708, (0 split)
##       CORR     < 0.1943486  to the left,  agree=0.807, adj=0.576, (0 split)
##       AN_angle < 181.8831   to the right, agree=0.782, adj=0.522, (0 split)
##       AF_angle < 186.9729   to the right, agree=0.762, adj=0.478, (0 split)
##       BF_angle < 244.978    to the right, agree=0.747, adj=0.444, (0 split)
##
## Node number 2: 90658 observations,    complexity param=0.0204284
##   mean=1.031503, MSE=0.03051057
##   left son=4 (89472 obs) right son=5 (1186 obs)
##   Primary splits:
##       AN_angle < 172.2778   to the right, improve=0.2921997, (0 missing)
##       AF_angle < 177.8769   to the right, improve=0.1942544, (0 missing)
##       NDAI     < 0.39333    to the left,  improve=0.1731555, (0 missing)
##       CORR     < 0.2107393  to the left,  improve=0.1664885, (0 missing)
##       BF_angle < 190.153    to the right, improve=0.1453832, (0 missing)
##   Surrogate splits:
##       AF_angle     < 179.1593   to the right, agree=0.996, adj=0.674, (0 split)
##       BF_angle     < 191.1332   to the right, agree=0.993, adj=0.427, (0 split)
##       CF_angle     < 198.8576   to the right, agree=0.990, adj=0.214, (0 split)
##       DF_angle     < 208.4304   to the right, agree=0.988, adj=0.110, (0 split)
##       y_coordinate < 21.5       to the right, agree=0.988, adj=0.066, (0 split)
##
## Node number 3: 75785 observations,    complexity param=0.03469808
##   mean=1.816903, MSE=0.1495724
##   left son=6 (6909 obs) right son=7 (68876 obs)
##   Primary splits:
##       x_coordinate < 295.5      to the right, improve=0.12110790, (0 missing)
##       CORR         < 0.1898805  to the left,  improve=0.11530360, (0 missing)
##       y_coordinate < 45.5       to the left,  improve=0.07630740, (0 missing)
##       DF_angle     < 261.4191   to the left,  improve=0.06399525, (0 missing)
##       SD           < 2.742117   to the left,  improve=0.05182910, (0 missing)
##   Surrogate splits:
##       AN_angle < 263.9707   to the right, agree=0.909, adj=0.003, (0 split)
##       AF_angle < 277.8265   to the right, agree=0.909, adj=0.003, (0 split)
```

```
##       SD       < 0.5655767  to the left,  agree=0.909, adj=0.000, (0 split)
##       CORR     < -0.3280679 to the left,  agree=0.909, adj=0.000, (0 split)
##       BF_angle < 300.493    to the right, agree=0.909, adj=0.000, (0 split)
##
## Node number 4: 89472 observations
##   mean=1.020632, MSE=0.02020647
##
## Node number 5: 1186 observations
##   mean=1.851602, MSE=0.126376
##
## Node number 6: 6909 observations,    complexity param=0.01565698
##   mean=1.391953, MSE=0.2383257
##   left son=12 (4537 obs) right son=13 (2372 obs)
##   Primary splits:
##       AN_angle     < 195.8967   to the left,  improve=0.3762043, (0 missing)
##       AF_angle     < 211.0325   to the left,  improve=0.3423202, (0 missing)
##       BF_angle     < 230.3455   to the left,  improve=0.3188470, (0 missing)
##       y_coordinate < 69.5       to the left,  improve=0.2875034, (0 missing)
##       CF_angle     < 249.0937   to the left,  improve=0.2864887, (0 missing)
##   Surrogate splits:
##       AF_angle     < 208.723    to the left,  agree=0.948, adj=0.847, (0 split)
##       BF_angle     < 230.3455   to the left,  agree=0.910, adj=0.737, (0 split)
##       CF_angle     < 248.7264   to the left,  agree=0.872, adj=0.626, (0 split)
##       y_coordinate < 77.5       to the left,  agree=0.846, adj=0.553, (0 split)
##       DF_angle     < 263.0844   to the left,  agree=0.843, adj=0.544, (0 split)
##
## Node number 7: 68876 observations,    complexity param=0.02039833
##   mean=1.85953, MSE=0.1207381
##   left son=14 (24392 obs) right son=15 (44484 obs)
##   Primary splits:
##       CORR     < 0.1975229  to the left,  improve=0.09704751, (0 missing)
##       SD       < 2.645732   to the left,  improve=0.04213724, (0 missing)
##       DF_angle < 292.9865   to the left,  improve=0.04156833, (0 missing)
##       AF_angle < 226.7572   to the right, improve=0.04012272, (0 missing)
##       NDAI     < 3.325084   to the right, improve=0.03477241, (0 missing)
##   Surrogate splits:
##       AF_angle     < 224.2357   to the right, agree=0.719, adj=0.206, (0 split)
##       AN_angle     < 205.6778   to the right, agree=0.717, adj=0.202, (0 split)
##       SD           < 3.536637   to the left,  agree=0.705, adj=0.166, (0 split)
##       DF_angle     < 280.5912   to the left,  agree=0.690, adj=0.126, (0 split)
##       y_coordinate < 277.5      to the right, agree=0.688, adj=0.120, (0 split)
##
## Node number 12: 4537 observations
##   mean=1.175446, MSE=0.1446649
##
## Node number 13: 2372 observations
##   mean=1.806071, MSE=0.1563206
##
## Node number 14: 24392 observations
##   mean=1.713349, MSE=0.2044824
##
## Node number 15: 44484 observations
##   mean=1.939686, MSE=0.05667606
```

```
rsq.rpart(fit)
```

```
##
## Regression tree:
## rpart(formula = label ~ ., data = train_val, method = "anova")
##
## Variables actually used in tree construction:
## [1] AN_angle     CORR         NDAI         x_coordinate
##
## Root node error: 39564/166443 = 0.2377
##
## n= 166443
##
##         CP nsplit rel error  xerror      xstd
## 1 0.643582      0   1.00000 1.00001 0.0011150
## 2 0.034698      1   0.35642 0.35694 0.0022024
## 3 0.020428      2   0.32172 0.32251 0.0021648
## 4 0.020398      3   0.30129 0.30532 0.0020456
## 5 0.015657      4   0.28089 0.28198 0.0018871
## 6 0.010000      5   0.26524 0.26619 0.0019180
```





Diagnostics 4b

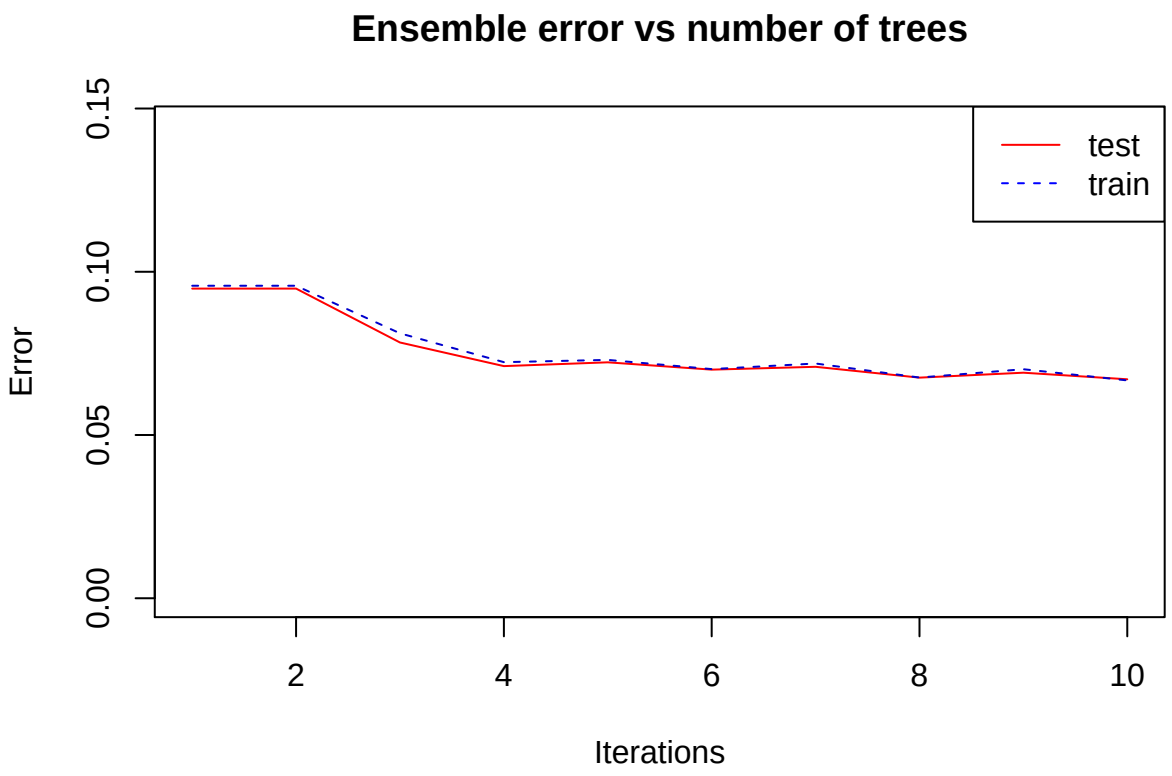
```
#refer section 4d
```

Diagnostics 4c AdaBoost

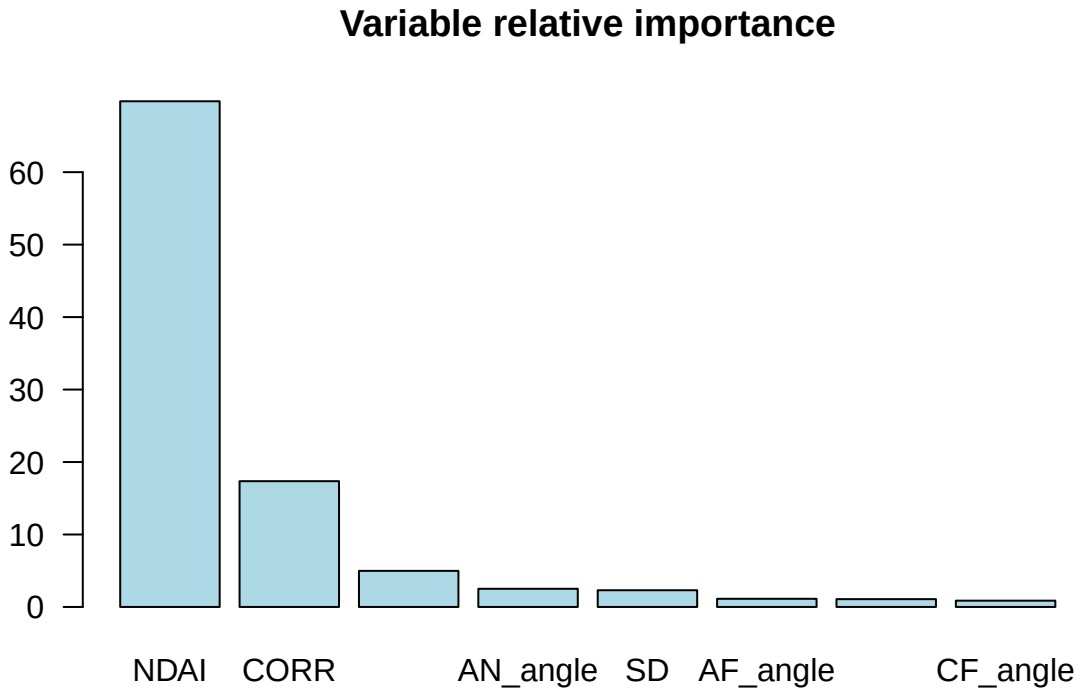
```
# Long running time
# Fit adaboost model
adaboost<- adaboost(label~y_coordinate + x_coordinate  + NDAI + SD + CORR + DF_angle + CF_angle+ BF_angl
# Adaboost model Accuracy
mean(predict(adaboost, newdata =test_total[,-3] )$class == test_total$label)
```

```
## [1] 0.997573
```

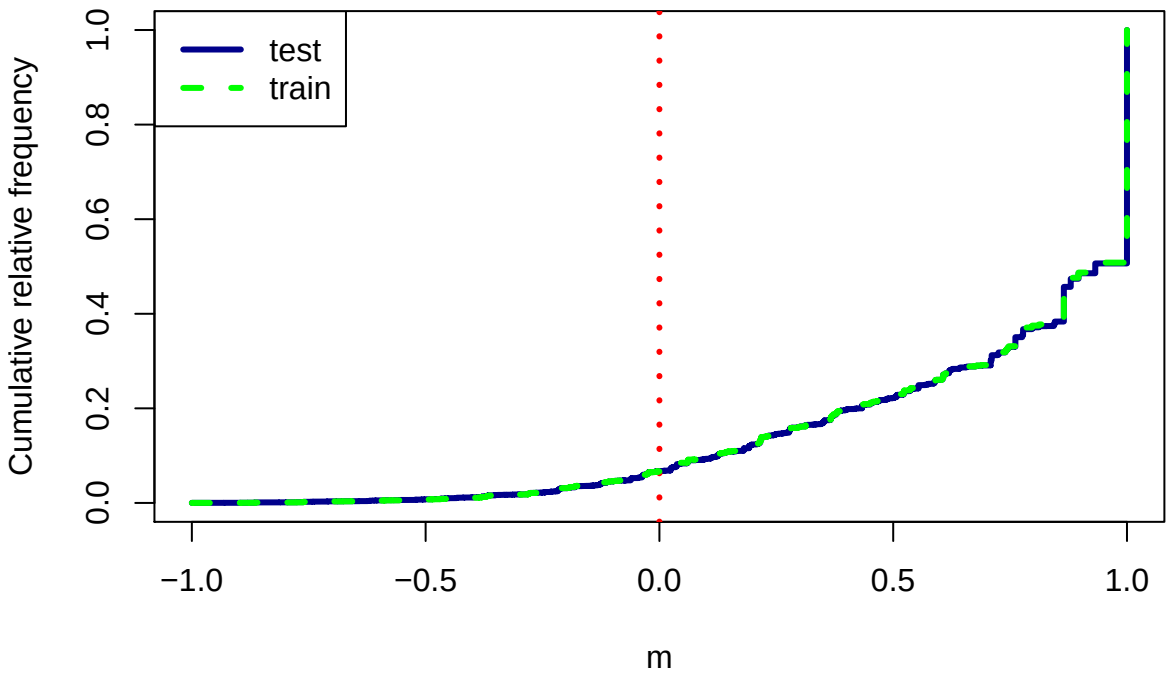
```
# Fit adaboost model without geographic information
boost_no_xy<- boosting(label~NDAI + SD + CORR + DF_angle + CF_angle+ BF_angle + AF_angle + AN_angle ,da

#Error based on number of trees
errorevol(boost_no_xy,train_val[,c(-1,-2)])->evol.train
errorevol(boost_no_xy,test_total[,c(-1,-2)])->evol.test
plot.errorevol(evol.test,evol.train)
```




```
importanceplot(boost_no_xy)
```



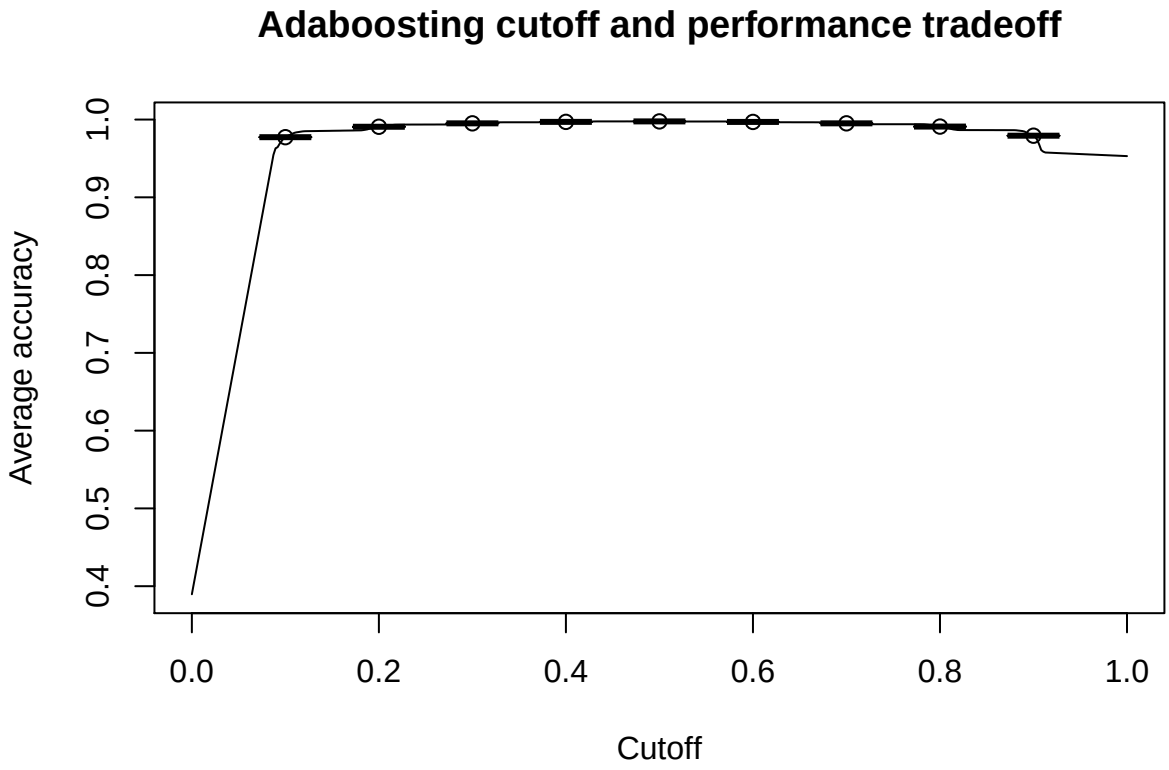

```
# Margins
BC.margins<-margins(boost_no_xy,train_val[,c(-1,-2)]) # training set
BC.adaboost.pred <- predict.boosting(boost_no_xy,newdata=test_total[,c(-1,-2)])

BC.predmargins<-margins(BC.adaboost.pred,test_total[,c(-1,-2)]) # test set
plot.margins(BC.predmargins,BC.margins,alpha=0.3 )
```



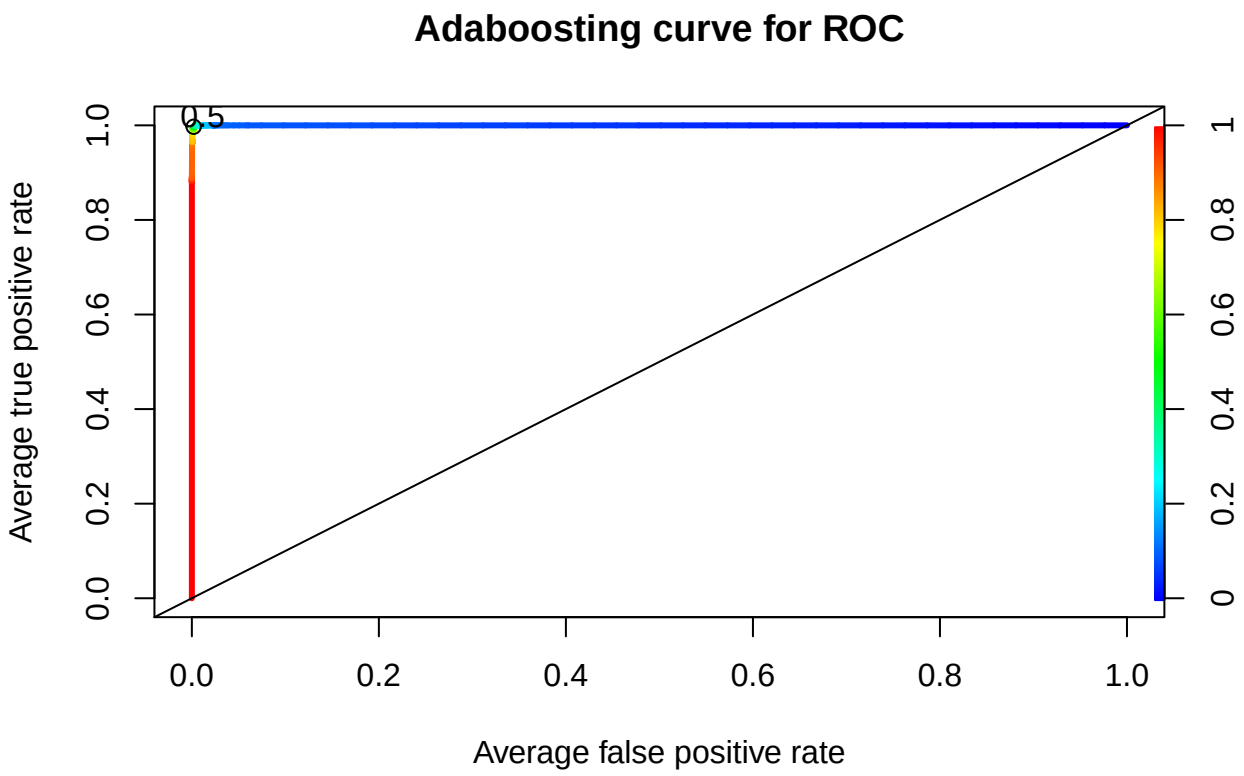

```
# Adaboost model Accuracy without geographic information
mean(predict(boost_no_xy, newdata =test_total[,-3] )$class == test_total$label)
```

```
## [1] 0.9329344
```

```
# Cut off point selection
predictions<- data.frame(predict(adaboost, newdata =test_total[,-3] )$prob)
colnames(predictions) <- c(-1,1)
pred <- prediction(predictions$`1`, test_total$label)
perf <- performance(pred, "acc")
plot(perf, avg= "vertical", spread.estimate="boxplot", show.spread.at= seq(0.1, 0.9, by=0.1), main="Ada
```

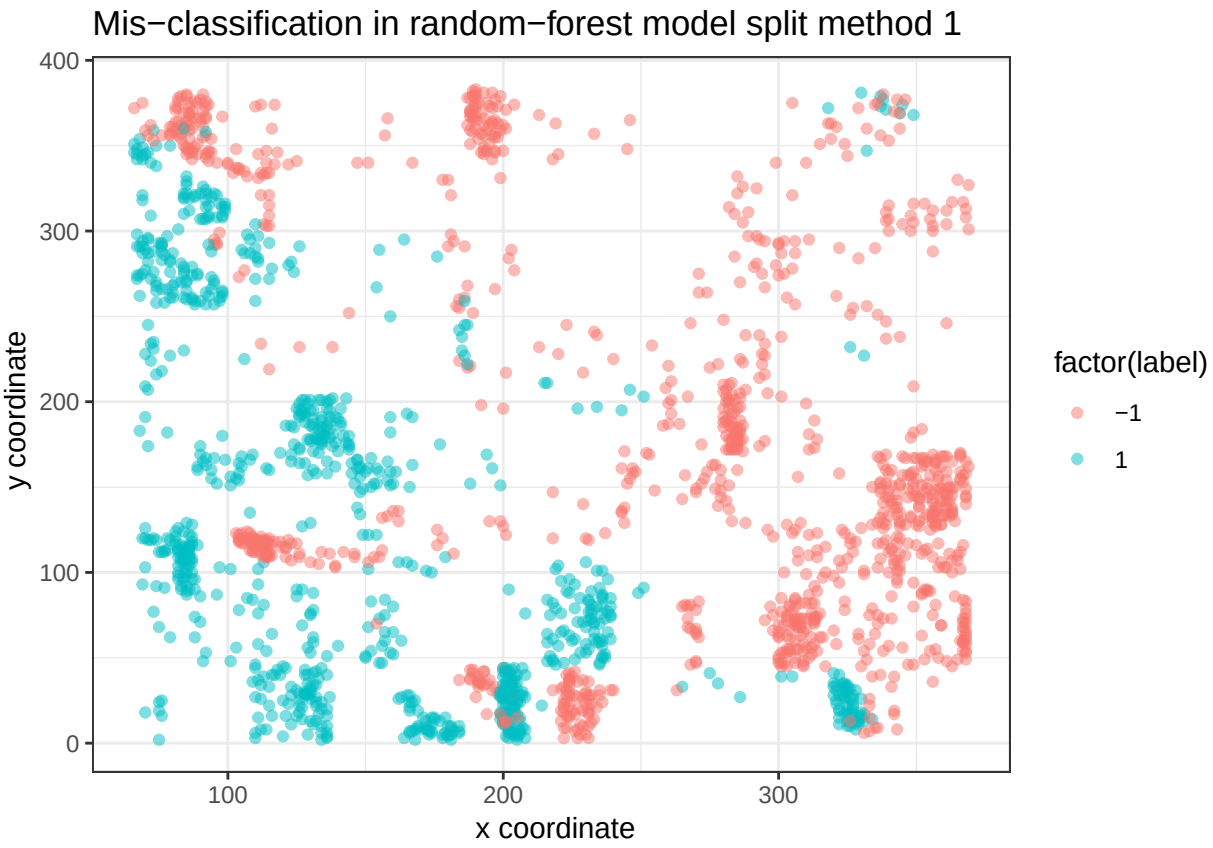



```
index<-which.max(slot(perf, "y.values")[[1]])
max<-slot(perf, "x.values")[[1]][index]
# ROC curve
perf <- performance(pred, "tpr", "fpr")
plot(perf, colorize=T, print.cutoffs.at=c(max), text.adj=c(0.3,0), avg="threshold", lwd=3, main= "Adaboo
abline(a=0,b=1)
```

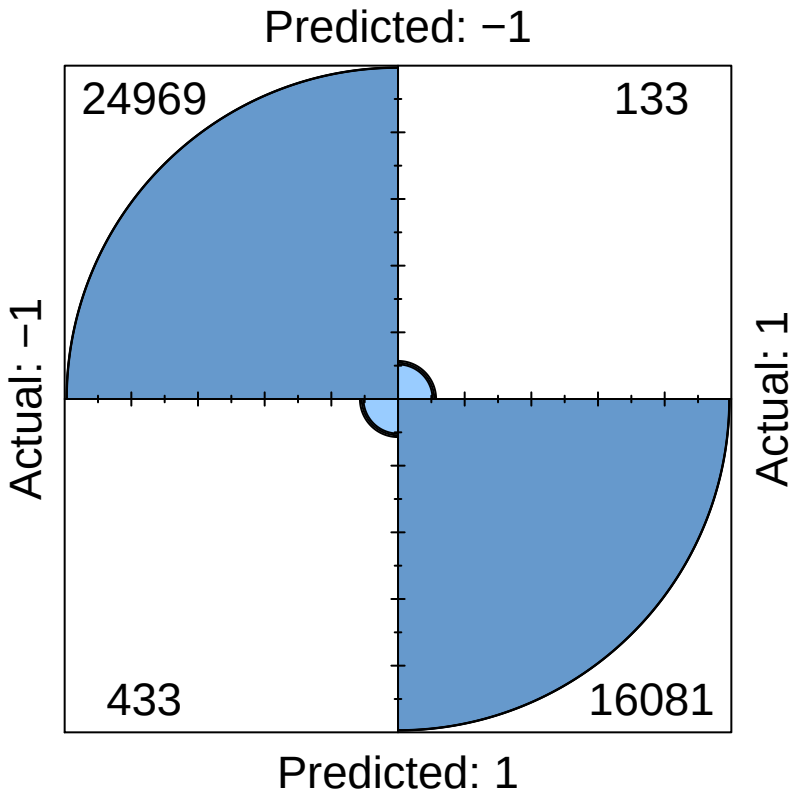


Diagnostics 4d

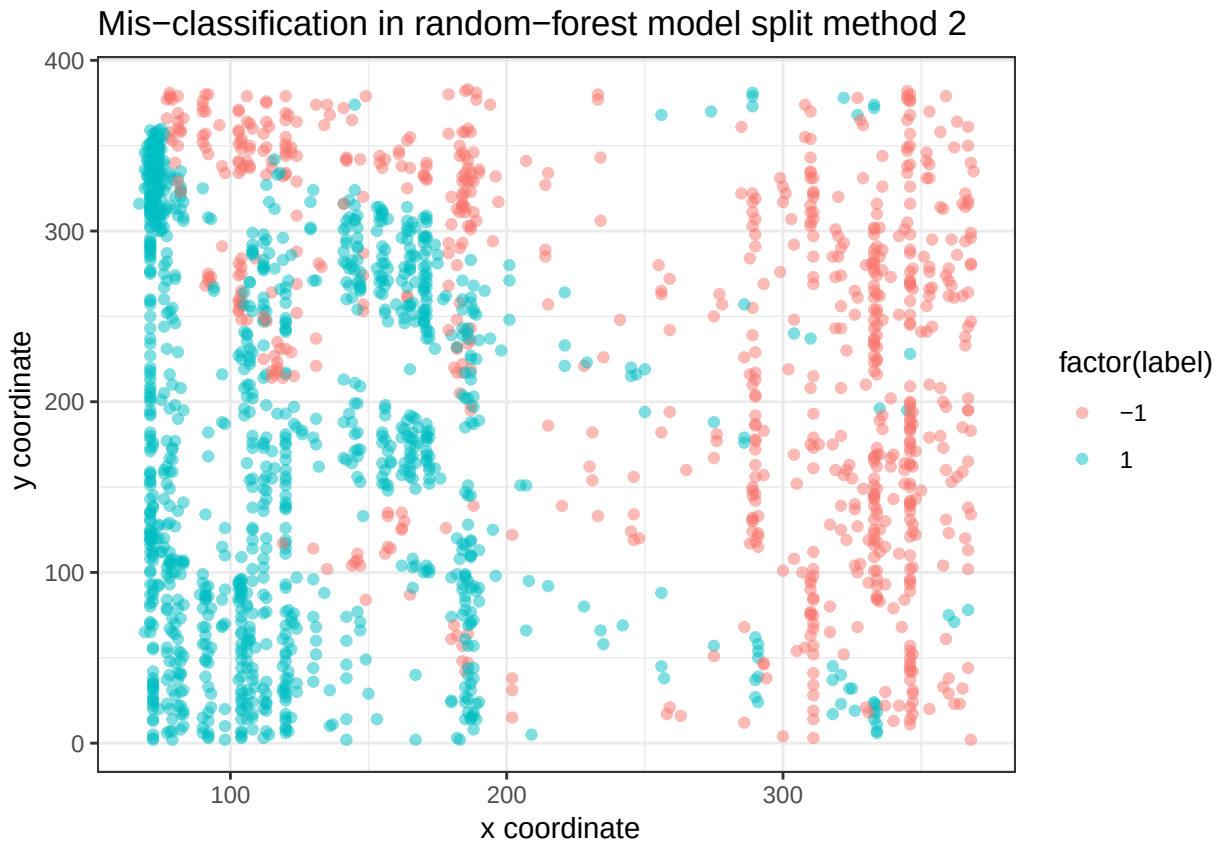
```
#Misclassification pattern for random forest
mis_label_random_forest<- train_data_rf[predicted.classes_rf!=test_total$label,]
ggplot(mis_label_random_forest)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate, color =
  xlab(" x coordinate") + ylab(" y coordinate") + ggtitle("Mis-classification in random-forest model sp
```



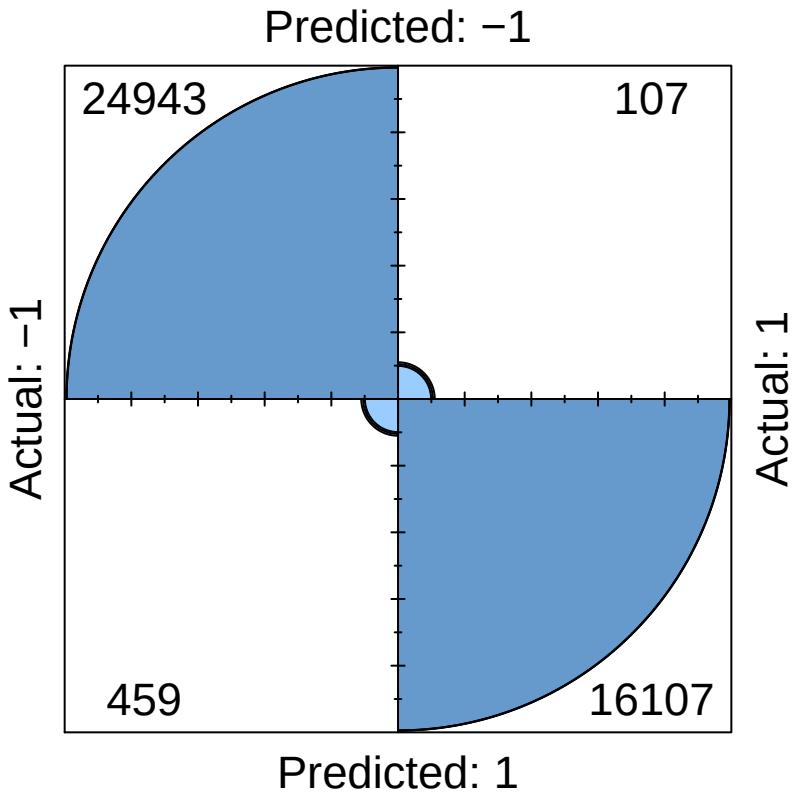

```
Predicted <-predicted.classes_rf
Actual<- test_total$label
fourfoldplot(table(Predicted, Actual))
```



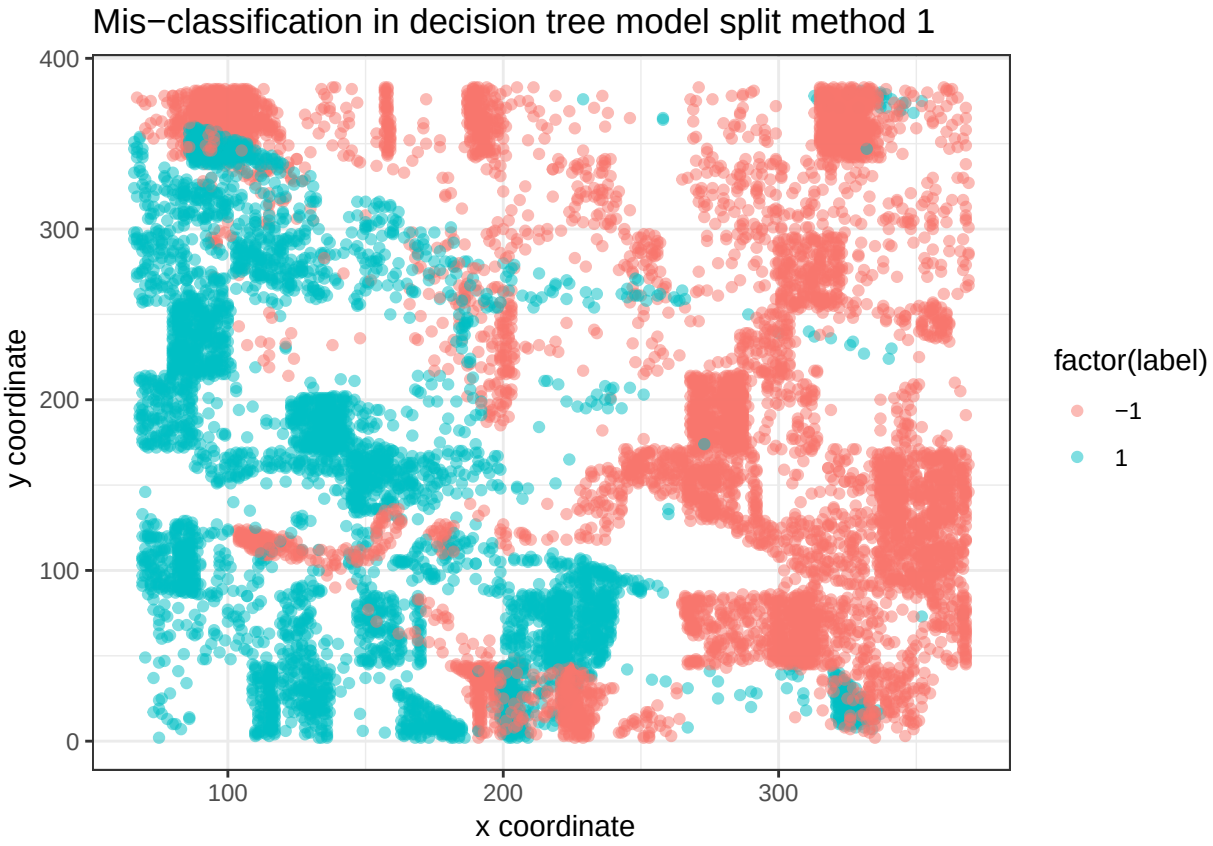

```
#Method 2
mis_label_random_forest<- train_data_rf2[predicted.classes_rf_method_2!=test_total$label,]
ggplot(mis_label_random_forest)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate, color =
  xlab(" x coordinate") + ylab(" y coordinate") + ggtitle("Mis-classification in random-forest model sp
```



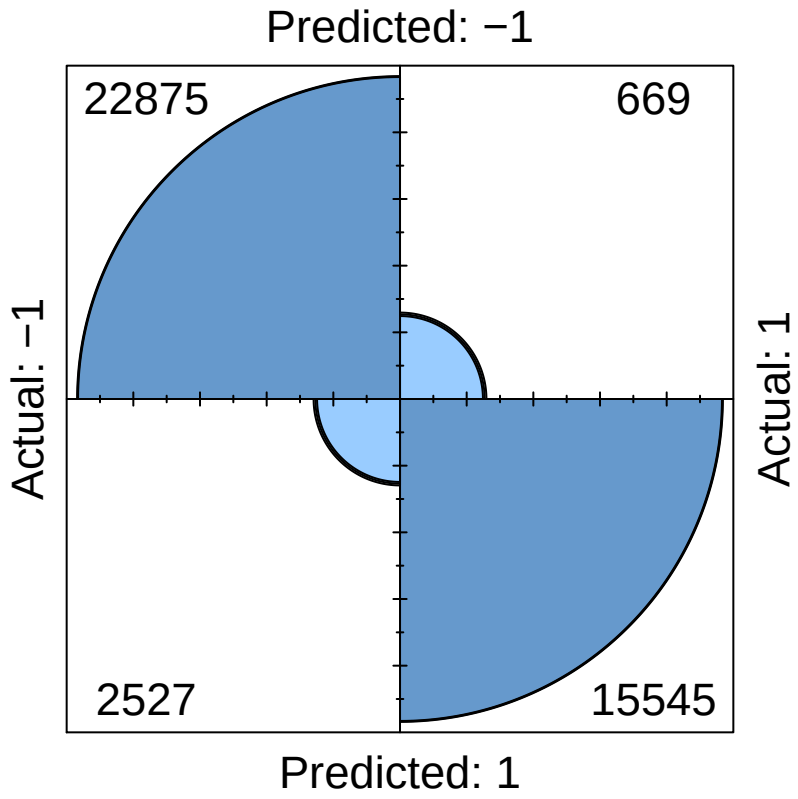

```
Predicted <-predicted.classes_rf_method_2
fourfoldplot(table(Predicted, Actual))
```



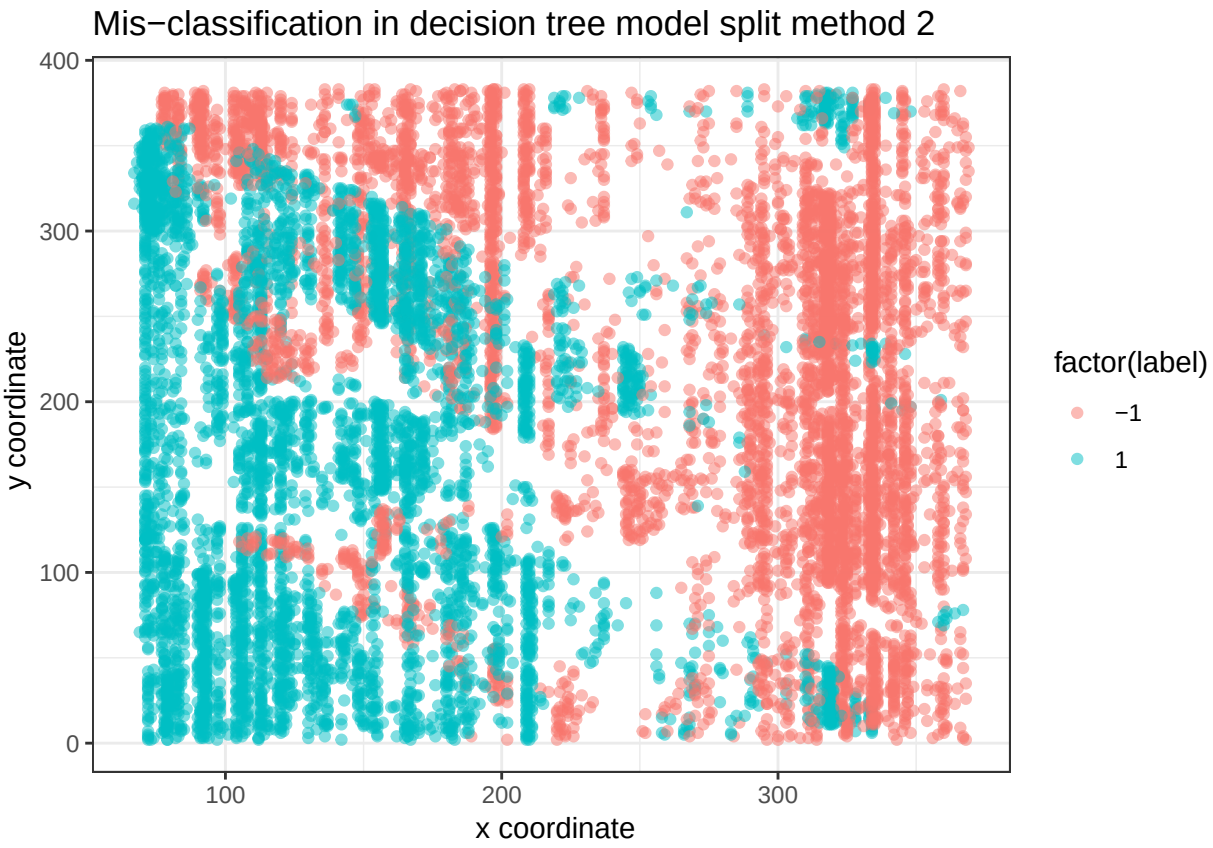

```
#Misclassification pattern for decision tree
mis_label_decision_tree<- train_data[tree.pred2!=test_total$label,]
ggplot(mis_label_decision_tree)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate, color =
  xlab(" x coordinate") + ylab(" y coordinate") + ggtitle("Mis-classification in decision tree model sp
```



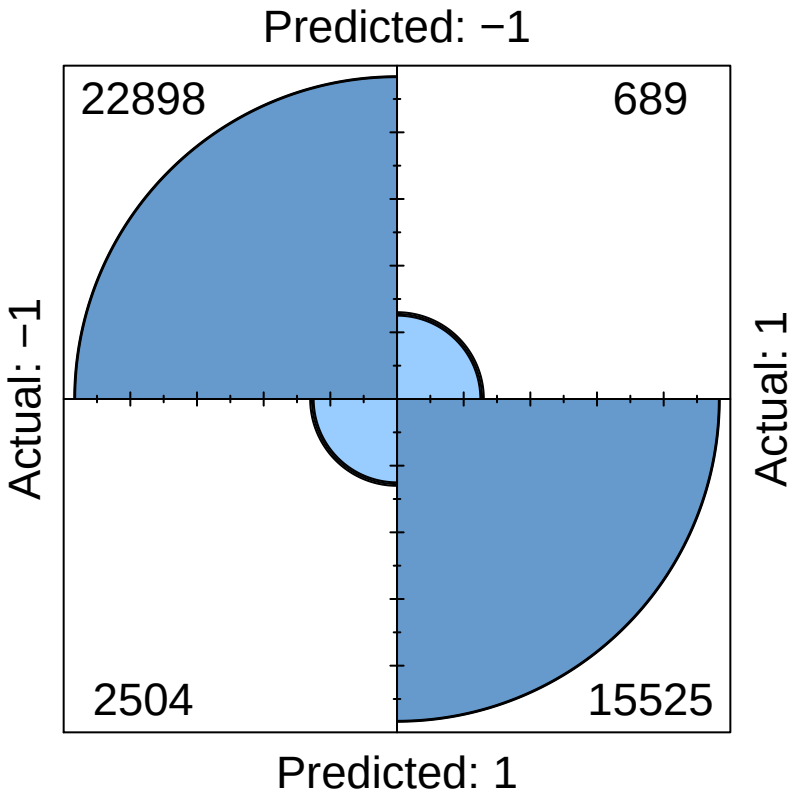

```
Predicted <-tree.pred2
fourfoldplot(table(Predicted, Actual))
```




```
#Method 2
mis_label_decision_tree2<- train_data_tree_2[tree.pred_method_2!=test_total$label,]
ggplot(mis_label_decision_tree2)+ geom_point(alpha = 0.5, aes(x= x_coordinate, y = y_coordinate, color =
  xlab(" x coordinate") + ylab(" y coordinate") + ggtitle("Mis-classification in decision tree model spl
```




```
Predicted <-tree.pred_method_2
fourfoldplot(table(Predicted, Actual))
```


Predicted: −1
22898
689
Actual: −1
Actual: 1
2504
15525
Predicted: 1