R Notebook

This is an R Markdown (http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

```r
#data import
df<- read.csv(file="c:/Users/zhang/Desktop/6240_r/events_log.csv")



#add new variable "clickthrough" to the data
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
factor <- factor(df$session_id)
groupby <- group_by(df, factor=factor(df$session_id))#group by session id
search_df_order <- groupby[order(factor,groupby$timestamp),] #sort by time in each group
search_df_order$clickthrough <- NA


#find clickthrough searches
for(i in 1:length(search_df_order$action)){
  if(search_df_order$action[i]=="searchResultPage" & search_df_order$action[i+1]!=search_df_order$action[i]){
    search_df_order$clickthrough[i]= TRUE
  }
}


#extract action==searchresultpage data
click<- search_df_order[search_df_order$action == "searchResultPage", ]


#add "Hour"and "Minute"" of search into the dataset ex.20160302162350 YYYYMMDDhhmmss
for (i in 1:length(click$timestamp)){

  click$hour[i] <- substr(toString(click$timestamp[i]),9,10)
  click$minute[i] <- substr(toString(click$timestamp[i]),11,12)

}
```

```
## Warning: Unknown or uninitialised column: 'hour'.
```

```
## Warning: Unknown or uninitialised column: 'minute'.
```

```
#training and testing datasets (random sample from click)
set.seed(136234)
copy_click <- as.data.frame(click)
library(caTools)
copy_click$spl <- sample.split(copy_click$page_id,SplitRatio=0.9) #create new column called spl and assign TRUE or False randomly
test <- subset(copy_click, copy_click$spl==TRUE)
train <- subset(copy_click, copy_click$spl==FALSE)
```

```
#classification problem
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.2
```

```
train_dataframe <- as.data.frame(train)
train_dataframe$clickthrough[is.na(train_dataframe$clickthrough)] <- "FALSE"
train_dataframe$hour <-as.numeric(train_dataframe$hour)
train_dataframe$minute <-as.numeric(train_dataframe$minute)
train_dataframe <- train_dataframe[c(4,8,11,12,13)] #chose variable to fit
train_dataframe <-train_dataframe
for (i in 1:length(train_dataframe$clickthrough)){

  if (train_dataframe$clickthrough[i] == "FALSE"){
    train_dataframe$clickthrough[i] = 0
  }
  if (train_dataframe$clickthrough[i] == "TRUE"){
    train_dataframe$clickthrough[i] = 1
  }

}


# 1. Naive Bayes
# assumption: predictors are independent

naive_fit <- naiveBayes(x=train_dataframe[c(1,2,4,5)],y=factor(train_dataframe$click
through))
predict.click <- predict(naive_fit,newdata=train_dataframe)
table(factor(train_dataframe$clickthrough),predict.click)
```

```
##    predict.click
##         0     1
##   0 10137    34
##   1  3445     8
```

```
# 2. LDA
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lda.fit <- lda(clickthrough~.,data=train_dataframe)
lda.class <- predict(lda.fit)$class
table(train_dataframe$clickthrough,lda.class) #Confusion Matrix
```

```
##    lda.class
##         0     1
##   0 10137    34
##   1  3445     8
```

```
# 3. QDA
qda.fit <- qda(clickthrough~.,data=train_dataframe)
qda.class <- predict(qda.fit)$class
table(train_dataframe$clickthrough,lda.class) #Confusion Matrix
```

```
##    lda.class
##         0     1
##   0 10137    34
##   1  3445     8
```

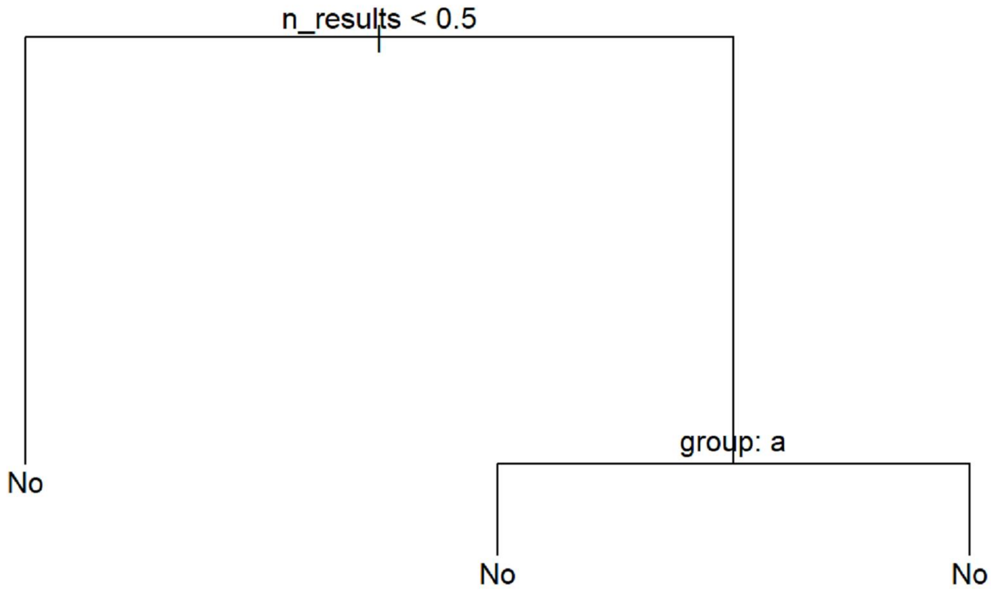
```
# 4. Logistic Regression
train_dataframe_lr<-train_dataframe
for (i in 1:length(train_dataframe_lr$clickthrough)){

  if (train_dataframe_lr$clickthrough[i] == "FALSE"){
    train_dataframe_lr$clickthrough[i] = 0
  }
  if (train_dataframe_lr$clickthrough[i] == "TRUE"){
    train_dataframe_lr$clickthrough[i] = 1
  }

}
train_dataframe_lr$clickthrough<- as.numeric(train_dataframe_lr$clickthrough)
lr.fit <- glm(clickthrough~.,data=train_dataframe_lr,family=binomial)
summary(lr.fit)
```

```
## 
## Call:
## glm(formula = clickthrough ~ ., family = binomial, data = train_dataframe_lr)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -7.4330  -0.8227  -0.5921   1.3819   2.2718  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -1.629452   0.070600 -23.080   <2e-16 ***
## groupb      -0.785870   0.047485 -16.550   <2e-16 ***
## n_results    0.058653   0.002545  23.049   <2e-16 ***
## hour        -0.002635   0.003275  -0.805    0.421    
## minute      -0.001007   0.001180  -0.853    0.393    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 15425  on 13623  degrees of freedom
## Residual deviance: 14454  on 13619  degrees of freedom
## AIC: 14464
## 
## Number of Fisher Scoring iterations: 5
```

```
# 5. Decision Tree without bagging
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.5.2
```

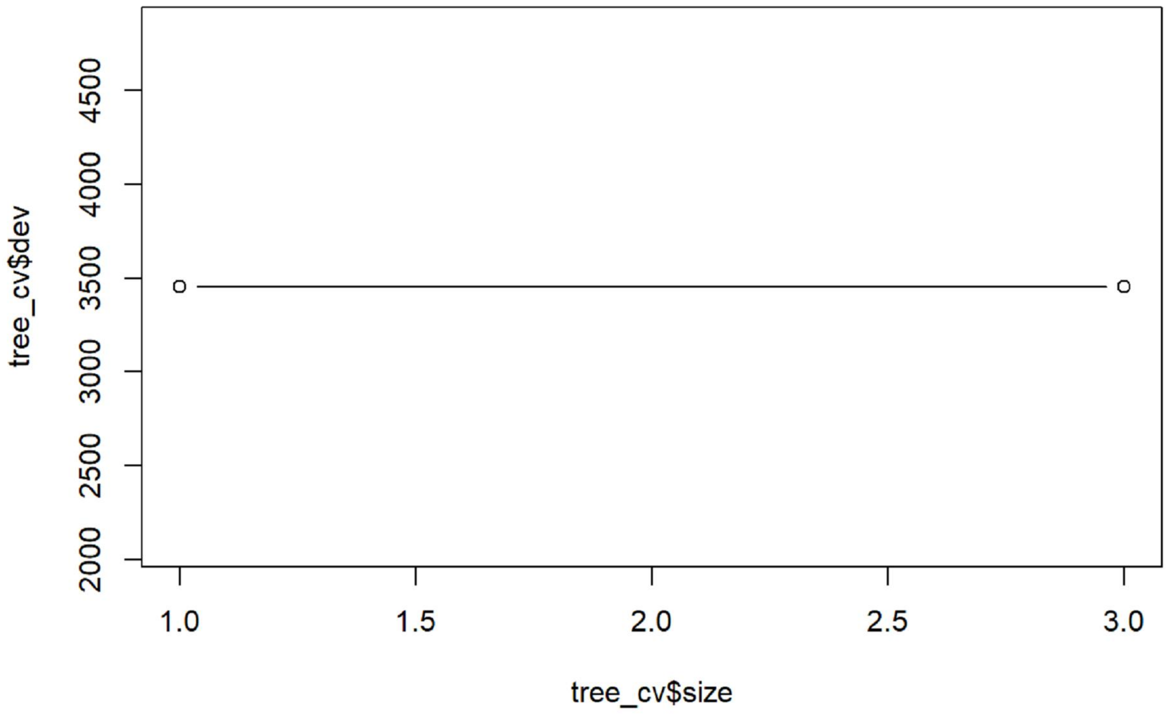
```
library(ISLR)
train_tree <- train_dataframe
train_tree <- train_tree %>% 
  mutate(YES=factor(ifelse(clickthrough==0,"No","Yes")))
tree.fit <- tree(YES~.-clickthrough,train_tree)
plot(tree.fit);text(tree.fit ,pretty =0)
```



```
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = YES ~ . - clickthrough, data = train_tree)
## Variables actually used in tree construction:
## [1] "n_results" "group"
## Number of terminal nodes:  3
## Residual mean deviance:  0.9941 = 13540 / 13620
## Misclassification error rate: 0.2534 = 3453 / 13624
```

```
#CV
tree_cv <- cv.tree(tree.fit ,FUN=prune.misclass )
plot(tree_cv$size ,tree_cv$dev,"b")
```



```
#ROC and AUC
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.5.2
```

```
## Loading required package: gplots
```

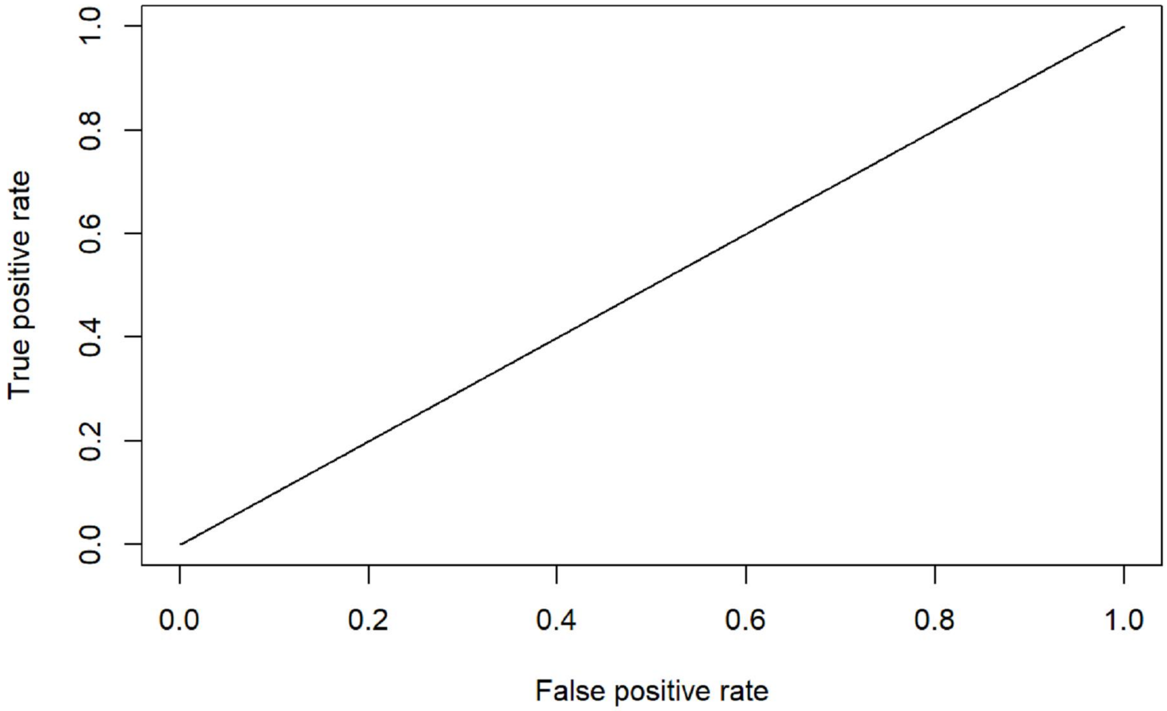
```
##
## Attaching package: 'gplots'
```

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

```
#test data
test_dataframe <- as.data.frame(test)
test_dataframe$clickthrough[is.na(test_dataframe$clickthrough)] <- "FALSE"
test_dataframe$hour <-as.numeric(test_dataframe$hour)
test_dataframe$minute <-as.numeric(test_dataframe$minute)
test_dataframe <- test_dataframe[c(4,8,11,12,13)] #chose variable to fit
for (i in 1:length(test_dataframe$clickthrough)){

  if (test_dataframe$clickthrough[i] == "FALSE"){
    test_dataframe$clickthrough[i] = 0
  }
  if (test_dataframe$clickthrough[i] == "TRUE"){
    test_dataframe$clickthrough[i] = 1
  }

}


#test data predict
#LDA
lda.predictions <- predict(lda.fit,test_dataframe)$class
lda.pred <- as.data.frame(sapply(lda.predictions, as.numeric))
real <- as.data.frame(sapply(test_dataframe$clickthrough, as.numeric))
lda.pred <- prediction(lda.pred,labels=real)

lda.AUC <- performance(lda.pred,"auc")@y.values[[1]] #AUC
lda.AUC
```

```
## [1] 0.499715
```

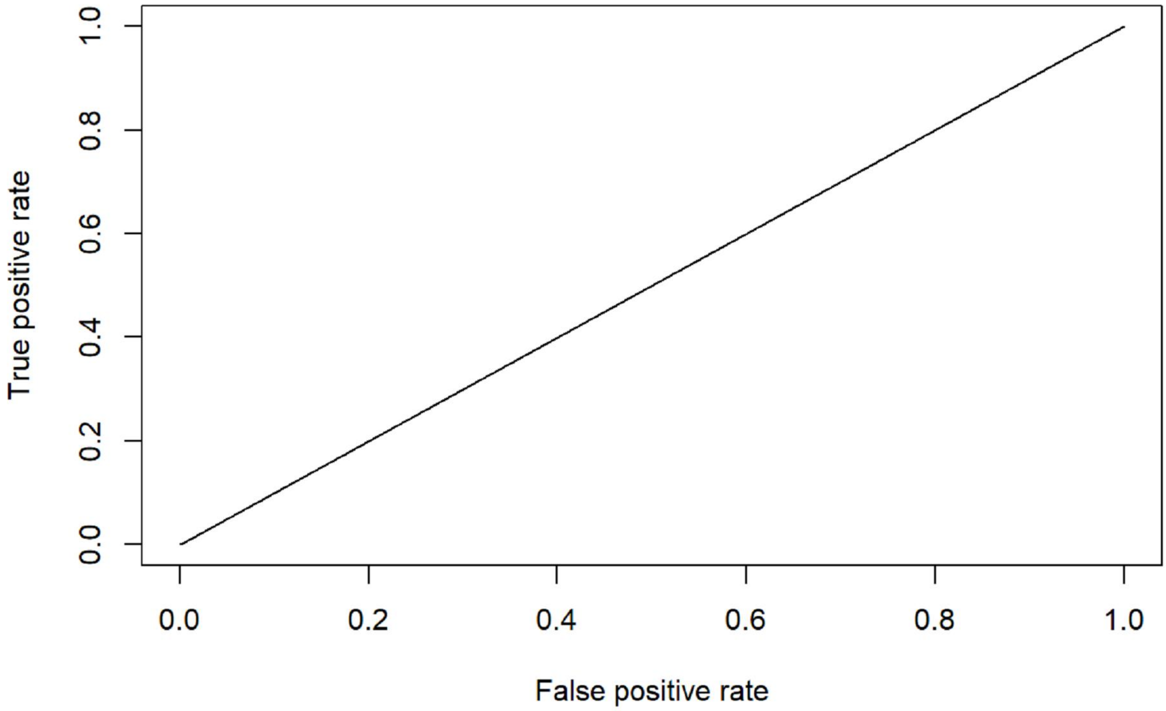
```
lda.ROC <- performance(lda.pred,"tpr","fpr")
plot(lda.ROC)#ROC
```



```
#QDA
qda.predictions <- predict(qda.fit,test_dataframe)$class
qda.pred <- as.data.frame(sapply(qda.predictions, as.numeric))
qda.pred <- prediction(qda.pred,labels=real)
qda.AUC <- performance(qda.pred,"auc")@y.values[[1]] #AUC
qda.AUC
```

```
## [1] 0.4996826
```

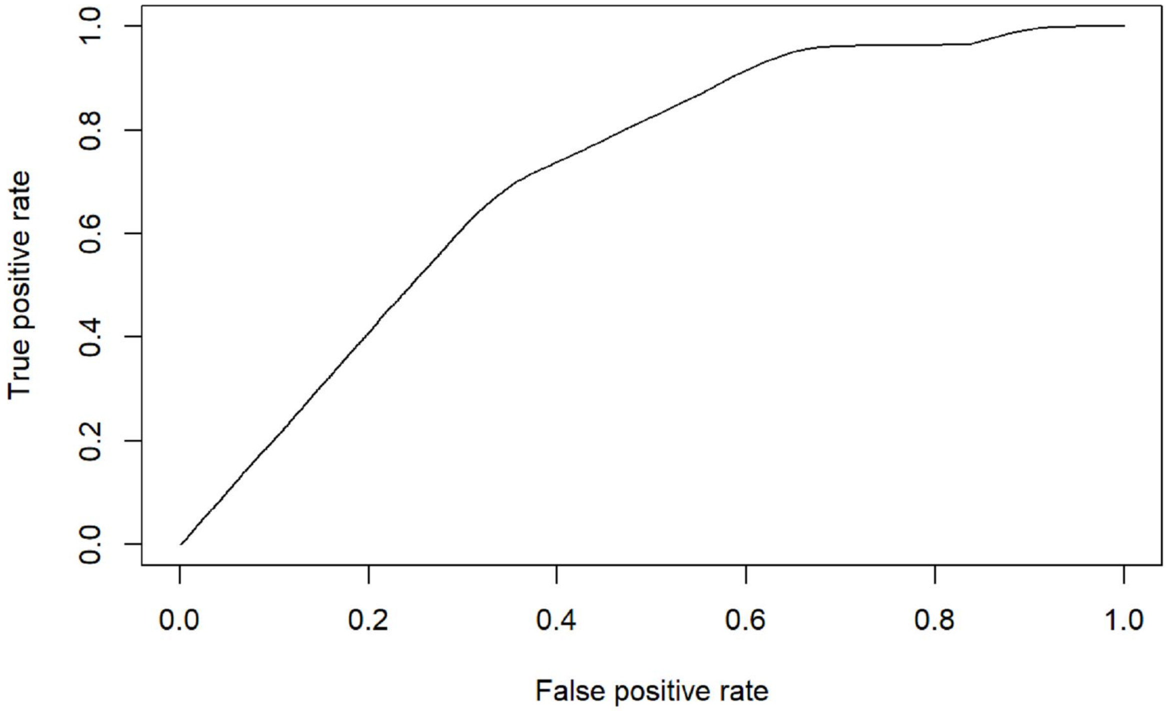
```
qda.ROC <- performance(qda.pred,"tpr","fpr")
plot(qda.ROC)#ROC
```



```
#log
lr.pred <- prediction(predictions=predict(lr.fit,test_dataframe),labels=real)
performance(lr.pred,"auc")@y.values[[1]] #AUC
```

```
## [1] 0.7137257
```

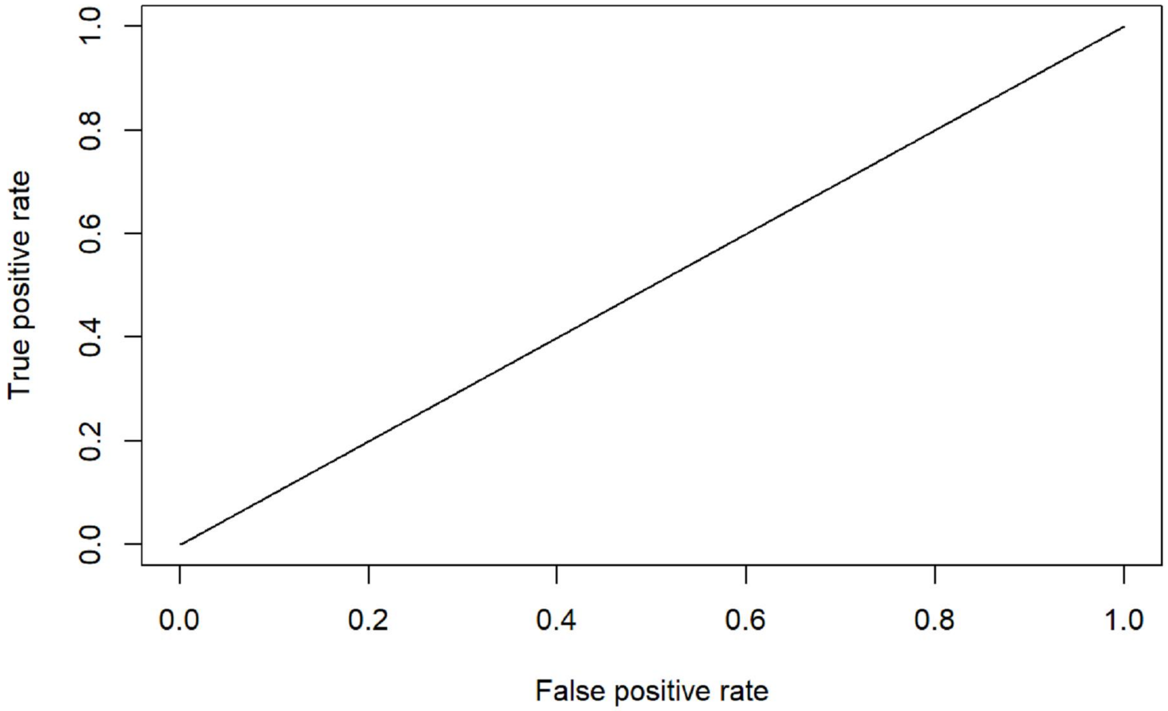
```
lr.ROC <- performance(lr.pred,"tpr","fpr")
plot(lr.ROC) #ROC
```



```
      #train predict
#LDA
lda.predictions1 <- predict(lda.fit,train_dataframe)$class
lda.pred1 <- as.data.frame(sapply(lda.predictions1, as.numeric))
real1 <- as.data.frame(sapply(train_dataframe$clickthrough, as.numeric))
lda.pred1 <- prediction(lda.pred1,labels=real1)

lda.AUC1 <- performance(lda.pred1,"auc")@y.values[[1]] #AUC
lda.AUC1
```

```
## [1] 0.499487
```

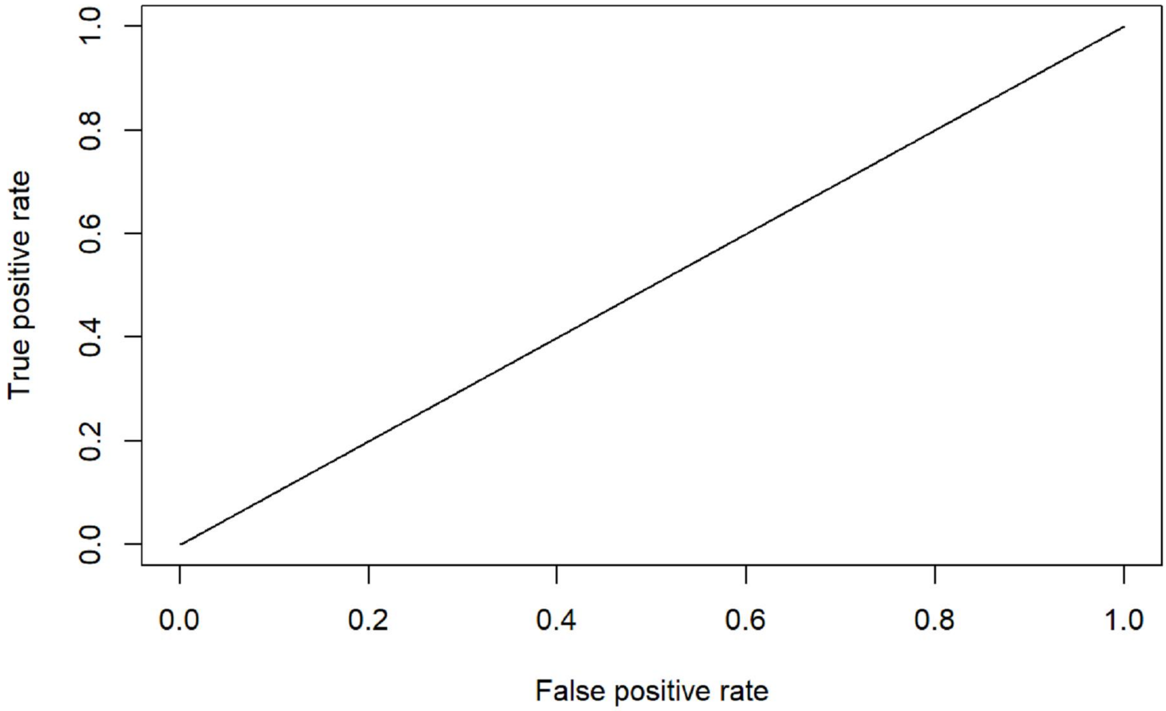
```
lda.ROC1 <- performance(lda.pred1,"tpr","fpr")
plot(lda.ROC1)#ROC
```



```
#QDA
qda.predictions1 <- predict(qda.fit,train_dataframe)$class
qda.pred1 <- as.data.frame(sapply(qda.predictions1, as.numeric))
qda.pred1 <- prediction(qda.pred1,labels=real1)
qda.AUC1 <- performance(qda.pred1,"auc")@y.values[[1]] #AUC
qda.AUC1
```

```
## [1] 0.4994378
```

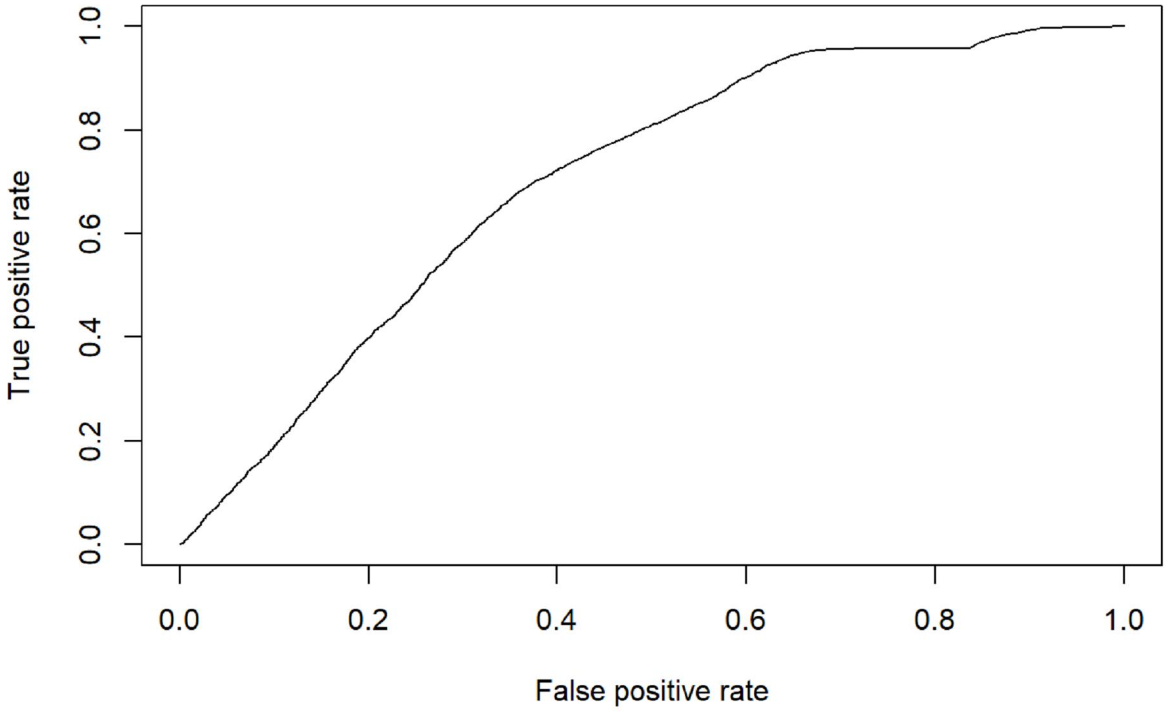
```
qda.ROC1 <- performance(qda.pred,"tpr","fpr")
plot(qda.ROC1)#ROC
```



```
#log
lr.pred1 <- prediction(predictions=predict(lr.fit,train_dataframe),labels=real1)
performance(lr.pred1,"auc")@y.values[[1]] #AUC
```

```
## [1] 0.7028949
```

```
lr.ROC1 <- performance(lr.pred1,"tpr","fpr")
plot(lr.ROC1) #ROC
```


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

```
#add new variable "clickthrough" to the data
library(dplyr)
factor <- factor(df$session_id)
groupby <- group_by(df, factor=factor(df$session_id))#group by session id
search_df_order <- groupby[order(factor,groupby$timestamp),] #sort by time in each g
roup
search_df_order$check30 <- NA


#find check for at least 30 seconds searches
for(i in 1:length(search_df_order$action)){

  if(search_df_order$action[i] == "searchResultPage" & is.na(search_df_order$checkin
[i+4])==FALSE &search_df_order$checkin[i+4]==30){
    search_df_order$check30[i]= TRUE
  }

}

search_df_order$clickthrough <- NA


#find clickthrough searches
for(i in 1:length(search_df_order$action)){
  if(search_df_order$action[i]=="searchResultPage" & search_df_order$action[i+1]!=se
arch_df_order$action[i]){
    search_df_order$clickthrough[i]= TRUE
  }
}




#extract action==searchresultpage data
click<- search_df_order[search_df_order$action == "searchResultPage", ]


#add "Hour"and "Minute"" of search into the dataset ex.20160302162350 YYYYMMDDhhmmss
for (i in 1:length(click$timestamp)){

  click$hour[i] <- substr(toString(click$timestamp[i]),9,10)
  click$minute[i] <- substr(toString(click$timestamp[i]),11,12)

}
```

```
## Warning: Unknown or uninitialised column: 'hour'.
```

```
## Warning: Unknown or uninitialised column: 'minute'.
```

```
#training and testing datasets (random sample from click)
set.seed(136234)
copy_click <- as.data.frame(click)
library(caTools)
copy_click$spl <- sample.split(copy_click$page_id,SplitRatio=0.9) #create new column called spl and assign TRUE or False randomly
test <- subset(copy_click, copy_click$spl==TRUE)
train <- subset(copy_click, copy_click$spl==FALSE)
```

```
#classification problem
library(e1071)
train_dataframe <- as.data.frame(train)
train_dataframe$clickthrough[is.na(train_dataframe$clickthrough)] <- "FALSE"
train_dataframe$check30[is.na(train_dataframe$check30)] <- "FALSE"
train_dataframe$hour <-as.numeric(train_dataframe$hour)
train_dataframe$minute <-as.numeric(train_dataframe$minute)
train_dataframe <- train_dataframe[c(4,8,11,12,13,14)] #chose variable to fit
train_dataframe <-train_dataframe
for (i in 1:length(train_dataframe$clickthrough)){

  if (train_dataframe$clickthrough[i] == "FALSE"){
    train_dataframe$clickthrough[i] = 0
  }
  if (train_dataframe$clickthrough[i] == "TRUE"){
    train_dataframe$clickthrough[i] = 1
  }

}

for (i in 1:length(train_dataframe$check30)){

  if (train_dataframe$check30[i] == "FALSE"){
    train_dataframe$check30[i] = 0
  }
  if (train_dataframe$check30[i] == "TRUE"){
    train_dataframe$check30[i] = 1
  }

}



# 1. Naive Bayes
# assumption: predictors are independent

naive_fit <- naiveBayes(x=train_dataframe[c(1,2,3,5,6)],y=factor(train_dataframe$clickthrough))
predict.click <- predict(naive_fit,newdata=train_dataframe)
table(factor(train_dataframe$clickthrough),predict.click)
```

```
##    predict.click
##         0     1
##   0 10146    25
##   1  1413  2040
```

```
# 2. LDA
library(MASS)
lda.fit <- lda(clickthrough~.,data=train_dataframe)
lda.class <- predict(lda.fit)$class
table(train_dataframe$clickthrough,lda.class) #Confusion Matrix
```

```
##    lda.class
##         0     1
##   0 10159    12
##   1  1415  2038
```

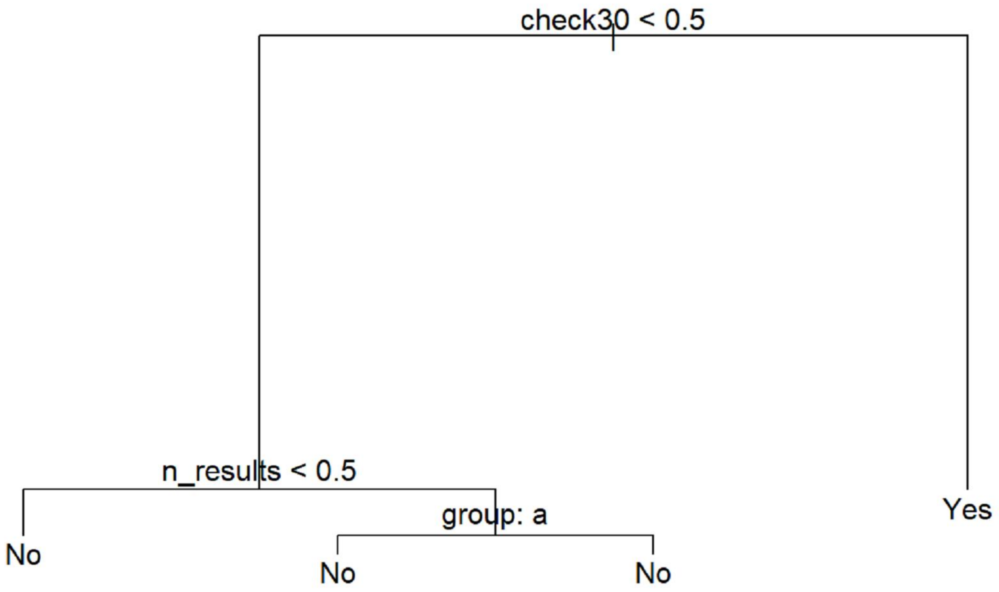
```
# 3. QDA
qda.fit <- qda(clickthrough~.,data=train_dataframe)
qda.class <- predict(qda.fit)$class
table(train_dataframe$clickthrough,lda.class) #Confusion Matrix
```

```
##    lda.class
##         0     1
##   0 10159    12
##   1  1415  2038
```

```
# 4. Logistic Regression
train_dataframe_lr<-train_dataframe
for (i in 1:length(train_dataframe_lr$clickthrough)){

  if (train_dataframe_lr$clickthrough[i] == "FALSE"){
    train_dataframe_lr$clickthrough[i] = 0
  }
  if (train_dataframe_lr$clickthrough[i] == "TRUE"){
    train_dataframe_lr$clickthrough[i] = 1
  }

}
train_dataframe_lr$clickthrough<- as.numeric(train_dataframe_lr$clickthrough)
lr.fit <- glm(clickthrough~.,data=train_dataframe_lr,family=binomial)
summary(lr.fit)
```

```
## 
## Call:
## glm(formula = clickthrough ~ ., family = binomial, data = train_dataframe_lr)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -5.7083  -0.5773  -0.3988   0.0525   2.5818  
## 
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -2.1097823  0.0941457 -22.410   <2e-16 ***
## groupb      -1.0594688  0.0744930 -14.222   <2e-16 ***
## n_results    0.0369571  0.0030066  12.292   <2e-16 ***
## check301     7.9821695  0.4493514  17.764   <2e-16 ***
## hour        -0.0057311  0.0046241  -1.239    0.215    
## minute      -0.0008063  0.0016700  -0.483    0.629    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 15425  on 13623  degrees of freedom
## Residual deviance:  8225  on 13618  degrees of freedom
## AIC: 8237
## 
## Number of Fisher Scoring iterations: 8
```

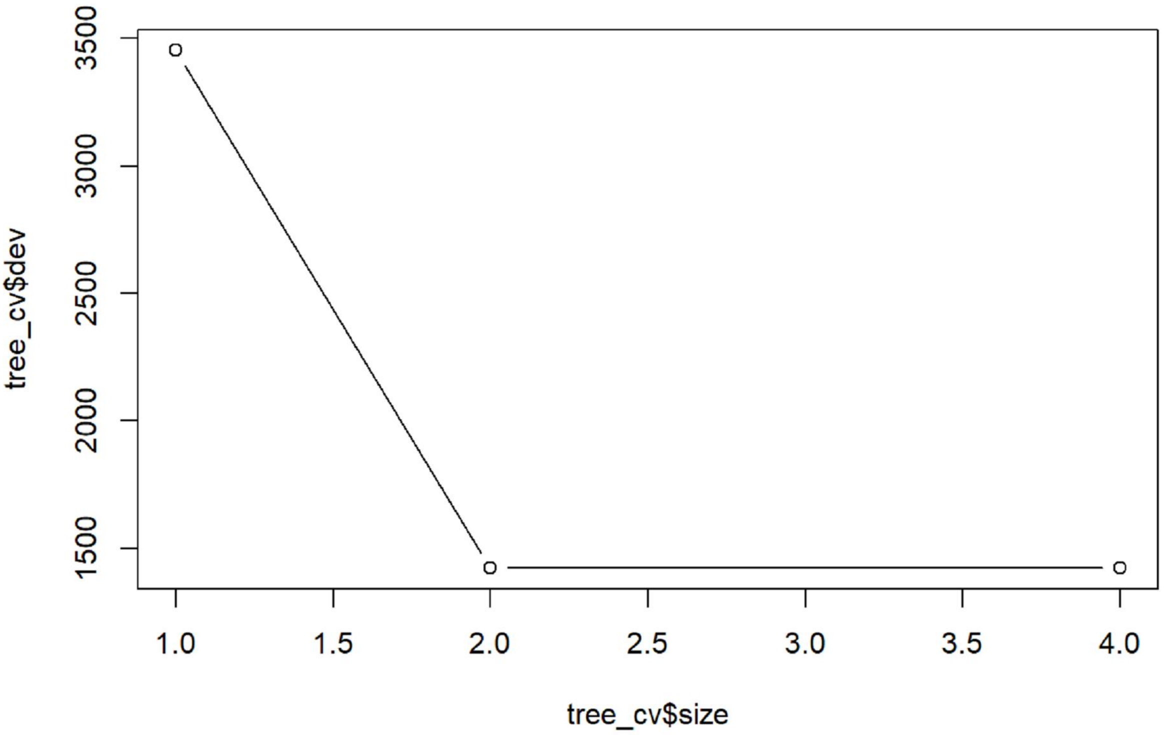
```
# 5. Decision Tree without bagging
library(tree)
library(ISLR)
train_tree <- train_dataframe
train_tree <- train_tree %>%
  mutate(YES=factor(ifelse(clickthrough==0,"No","Yes")))
tree.fit <- tree(YES~.-clickthrough,train_tree)
plot(tree.fit);text(tree.fit ,pretty =0)
```



```
summary(tree.fit)
```

```
##
## Classification tree:
## tree(formula = YES ~ . - clickthrough, data = train_tree)
## Variables actually used in tree construction:
## [1] "check30"   "n_results" "group"
## Number of terminal nodes:  4
## Residual mean deviance:  0.567 = 7722 / 13620
## Misclassification error rate: 0.1044 = 1422 / 13624
```

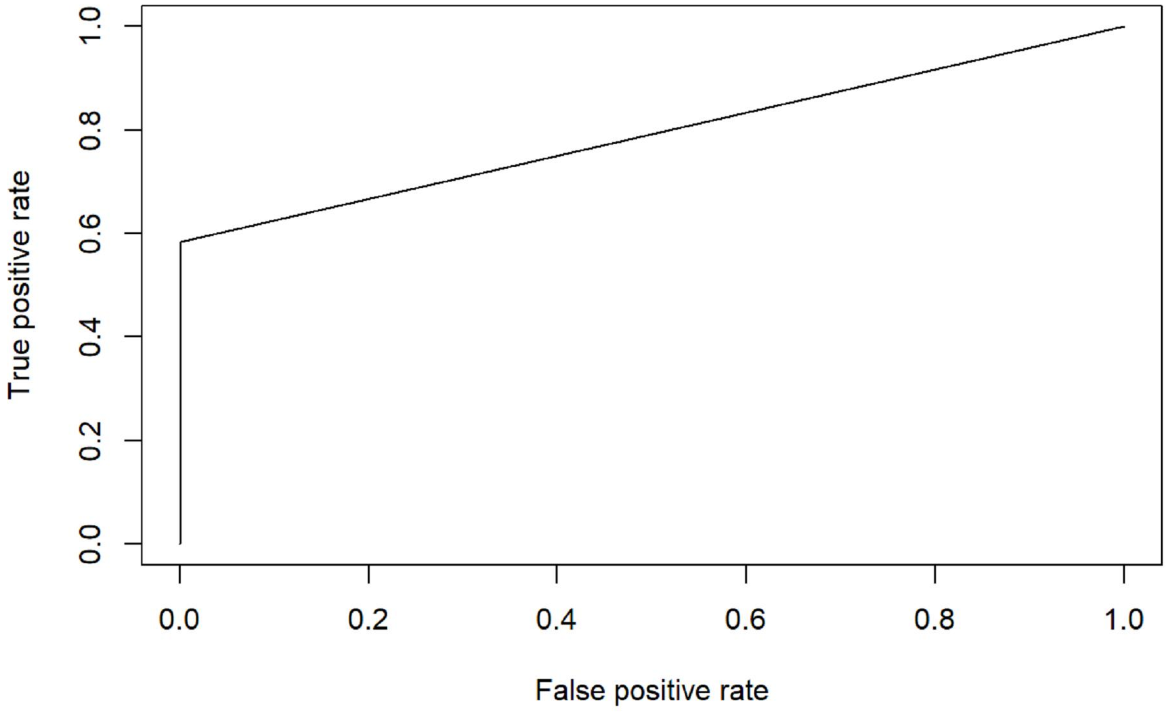
```
#CV
tree_cv <- cv.tree(tree.fit ,FUN=prune.misclass )
plot(tree_cv$size ,tree_cv$dev,"b")
```


tree_cv$dev
3500
3000
2500
2000
1500
1.0
1.5
2.0
2.5
3.0
3.5
4.0
tree_cv$size

```
#ROC and AUC
library(ROCR)
#test data
test_dataframe <- as.data.frame(test)
test_dataframe$clickthrough[is.na(test_dataframe$clickthrough)] <- "FALSE"
test_dataframe$check30[is.na(test_dataframe$check30)] <- "FALSE"
test_dataframe$hour <-as.numeric(test_dataframe$hour)
test_dataframe$minute <-as.numeric(test_dataframe$minute)
test_dataframe <- test_dataframe[c(4,8,11,12,13,14)] #chose variable to fit
for (i in 1:length(test_dataframe$check30)){

  if (test_dataframe$check30[i] == "FALSE"){
    test_dataframe$check30[i] = 0
  }
  if (test_dataframe$check30[i] == "TRUE"){
    test_dataframe$check30[i] = 1
  }

}

for (i in 1:length(test_dataframe$clickthrough)){

  if (test_dataframe$clickthrough[i] == "FALSE"){
    test_dataframe$clickthrough[i] = 0
  }
  if (test_dataframe$clickthrough[i] == "TRUE"){
    test_dataframe$clickthrough[i] = 1
  }

}


#test data predict
#LDA
lda.predictions <- predict(lda.fit,test_dataframe)$class
lda.pred <- as.data.frame(sapply(lda.predictions, as.numeric))
real <- as.data.frame(sapply(test_dataframe$clickthrough, as.numeric))
lda.pred <- prediction(lda.pred,labels=real)

lda.AUC <- performance(lda.pred,"auc")@y.values[[1]] #AUC
lda.AUC
```

```
## [1] 0.7914889
```

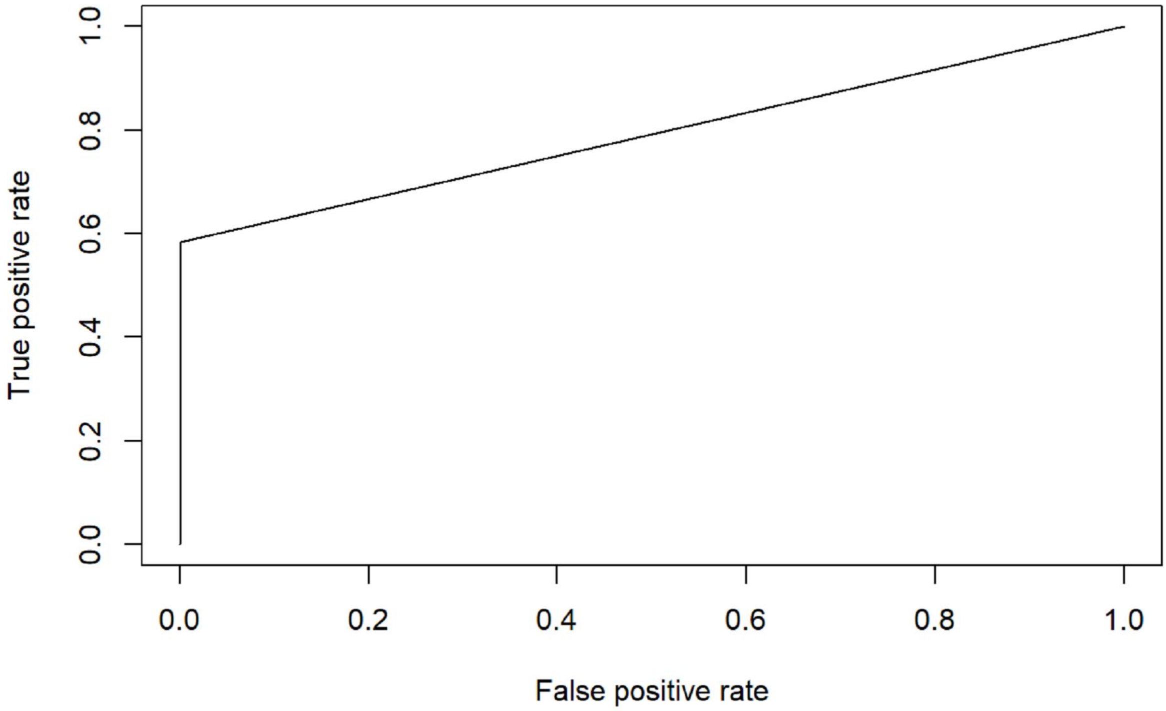
```
lda.ROC <- performance(lda.pred,"tpr","fpr")
plot(lda.ROC)#ROC
```



```
#QDA
qda.predictions <- predict(qda.fit,test_dataframe)$class
qda.pred <- as.data.frame(sapply(qda.predictions, as.numeric))
qda.pred <- prediction(qda.pred,labels=real)
qda.AUC <- performance(qda.pred,"auc")@y.values[[1]] #AUC
qda.AUC
```

```
## [1] 0.7914997
```

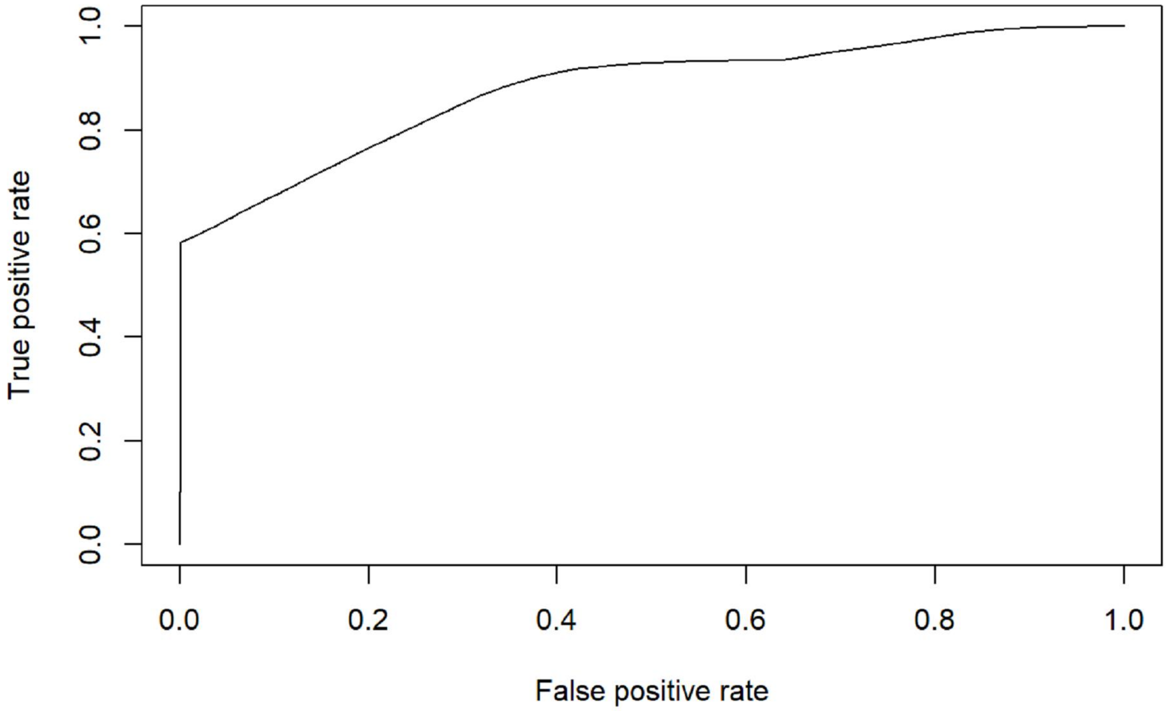
```
qda.ROC <- performance(qda.pred,"tpr","fpr")
plot(qda.ROC)#ROC
```



```
#log
lr.pred <- prediction(predictions=predict(lr.fit,test_dataframe),labels=real)
performance(lr.pred,"auc")@y.values[[1]] #AUC
```

```
## [1] 0.8785645
```

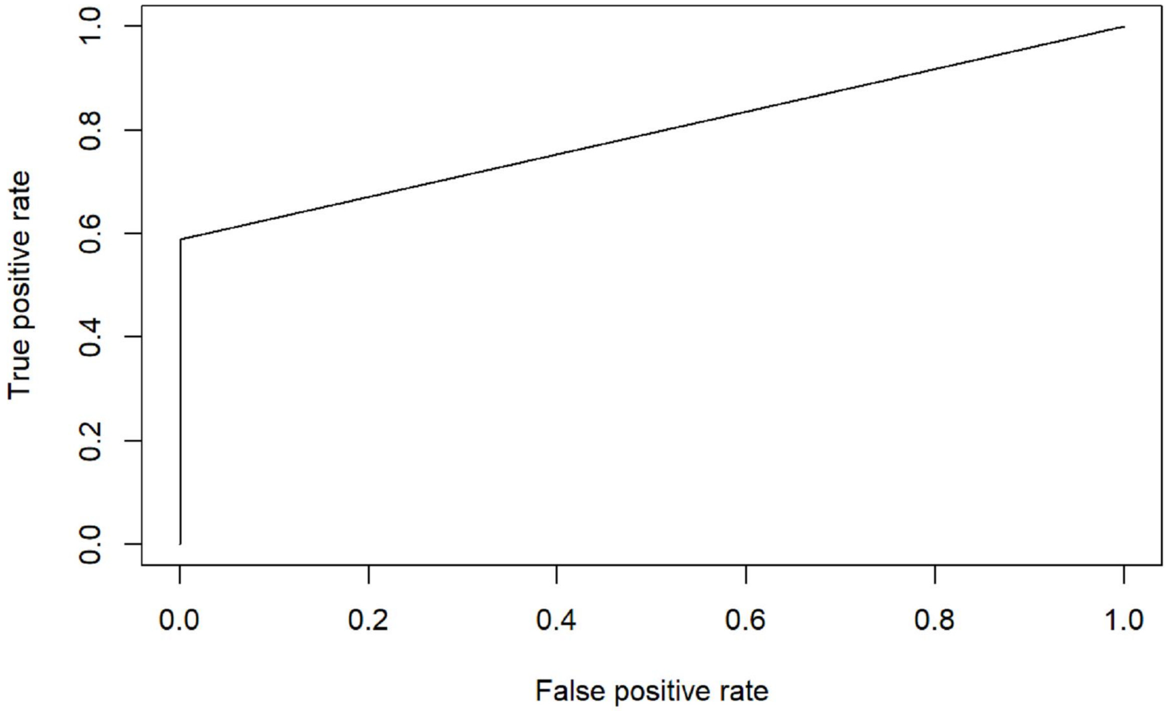
```
lr.ROC <- performance(lr.pred,"tpr","fpr")
plot(lr.ROC) #ROC
```



```
     #train predict
#LDA
lda.predictions1 <- predict(lda.fit,train_dataframe)$class
lda.pred1 <- as.data.frame(sapply(lda.predictions1, as.numeric))
real1 <- as.data.frame(sapply(train_dataframe$clickthrough, as.numeric))
lda.pred1 <- prediction(lda.pred1,labels=real1)

lda.AUC1 <- performance(lda.pred1,"auc")@y.values[[1]] #AUC
lda.AUC1
```

```
## [1] 0.7945158
```

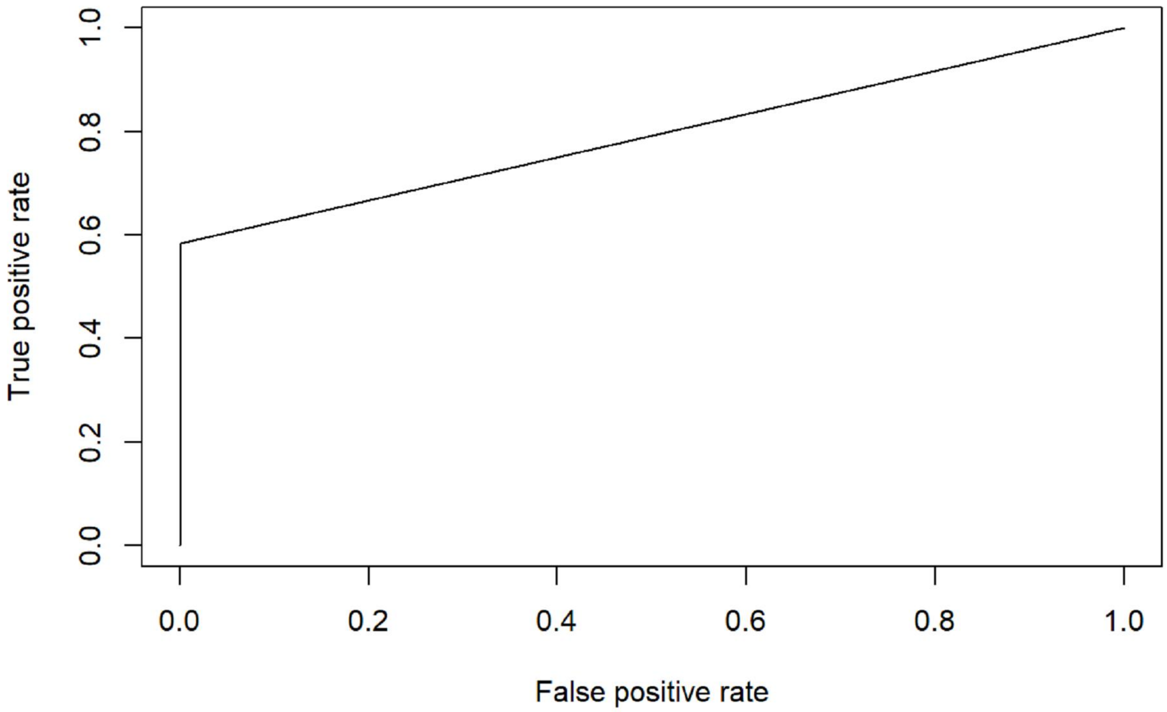
```
lda.ROC1 <- performance(lda.pred1,"tpr","fpr")
plot(lda.ROC1)#ROC
```



```
#QDA
qda.predictions1 <- predict(qda.fit,train_dataframe)$class
qda.pred1 <- as.data.frame(sapply(qda.predictions1, as.numeric))
qda.pred1 <- prediction(qda.pred1,labels=real1)
qda.AUC1 <- performance(qda.pred1,"auc")@y.values[[1]] #AUC
qda.AUC1
```

```
## [1] 0.7944202
```

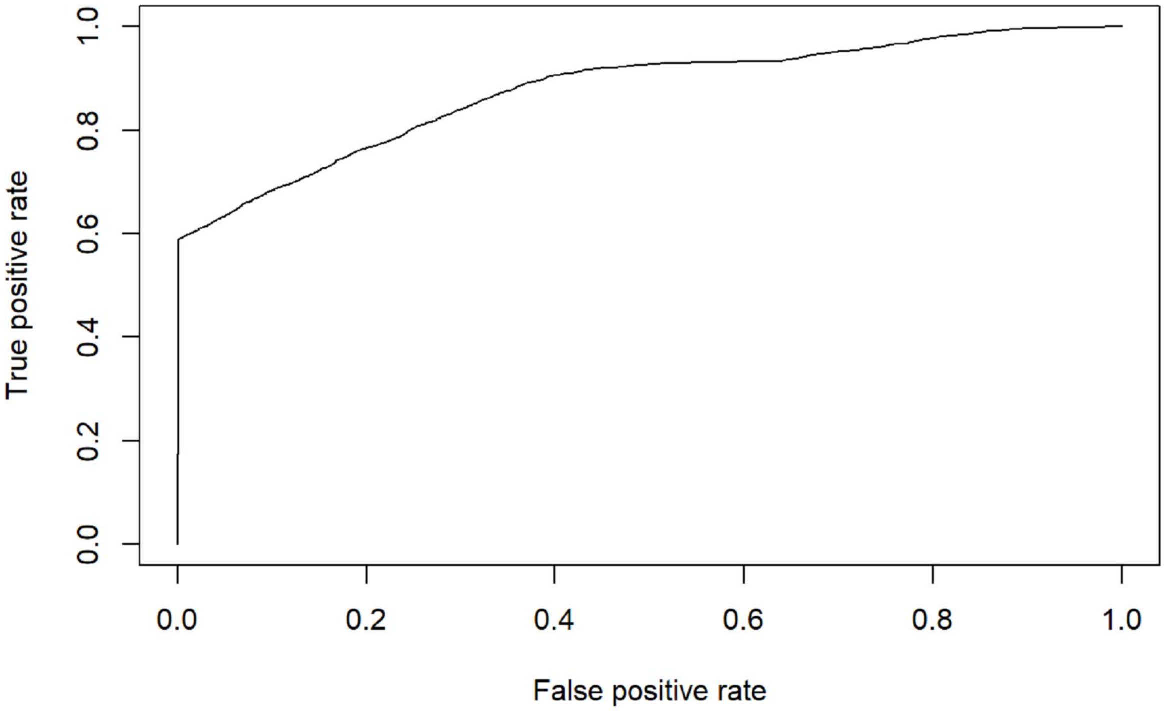
```
qda.ROC1 <- performance(qda.pred,"tpr","fpr")
plot(qda.ROC1)#ROC
```



```
#log
lr.pred1 <- prediction(predictions=predict(lr.fit,train_dataframe),labels=real1)
performance(lr.pred1,"auc")@y.values[[1]] #AUC
```

```
## [1] 0.8772896
```

```
lr.ROC1 <- performance(lr.pred1,"tpr","fpr")
plot(lr.ROC1) #ROC
```


True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0