Data Mining Report

Khade Pallavi (1719), Gauri Zape (Maths Dept)(17231)

2 October 2019

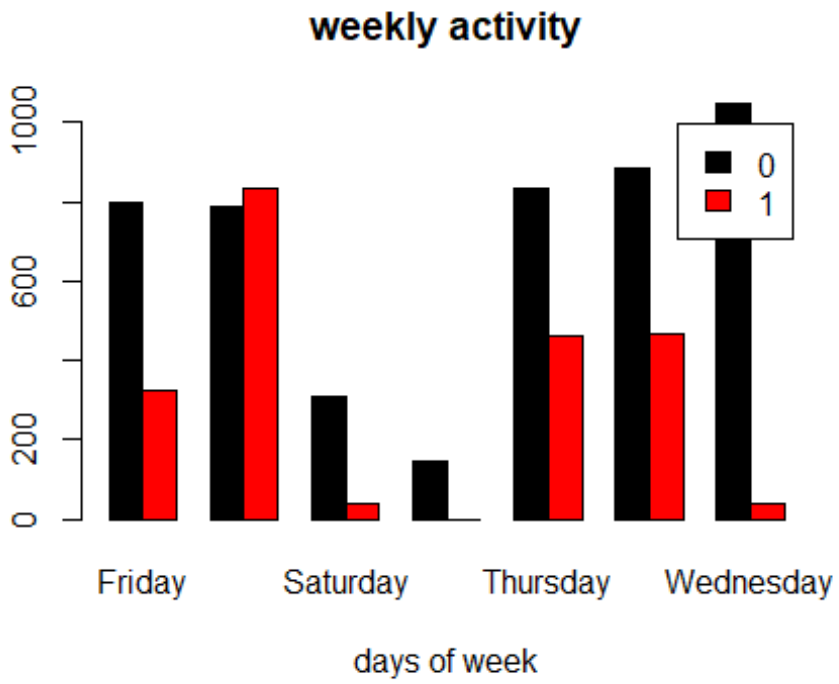
Data Description Description: Web-user identification is one of the important problems in behavioural psy-chology. Here we try to identify a user on the Internet by tracking his/her sequence of visited Web pages. The algorithm to be built will take a webpage session (a sequence of ten webpages visited consecutively by the same person) along with the start time of visit for a webpage and predict whether it belongs to Alice (1) or somebody else (0)t. ### The train_data contains information on user browsing sessions where the features are: site_ - ids of sites in this session. time_ - timestamps of attending the corresponding site target - whether this session belongs to Alice One can use the original data train.zip to form a train set differing from train_sessions.csv. ## Added Time Features days_:Days of Week Binary_Days:Whether days are weekdays or weekends Shift: timestaps of whole days 0:6(hr)- “night” ; 6:11(hr)-“morning” ;11:16(hr)-“aftenoon” 16:19(hr)-“evening” 19:24(hr)-“night1”

```
train_data = read.csv("C://Users//Pooja//Desktop//Train_data.csv")
```

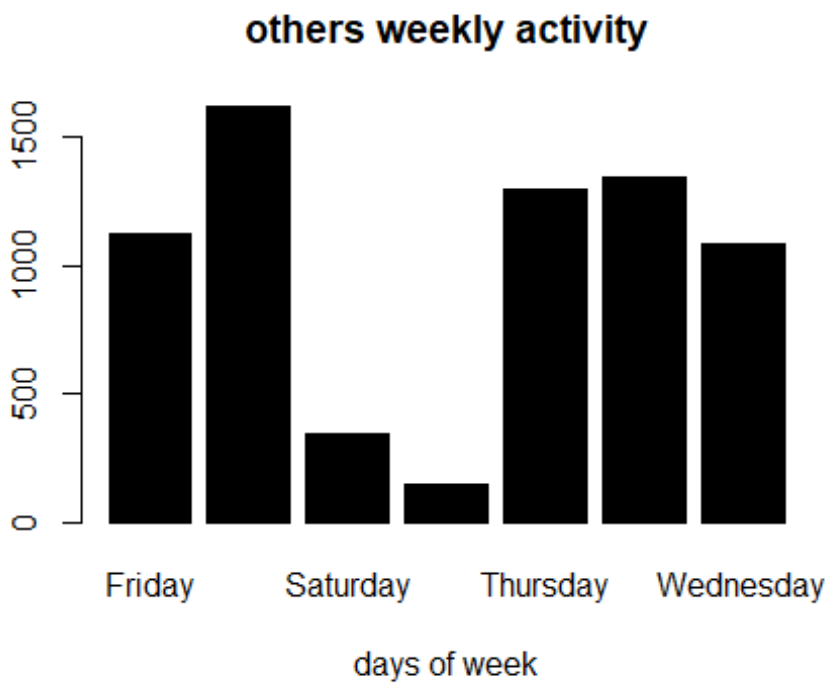
Exploratory Analysis

Let us first observe weekday barplots.. will see that it may fail to give a good idea.

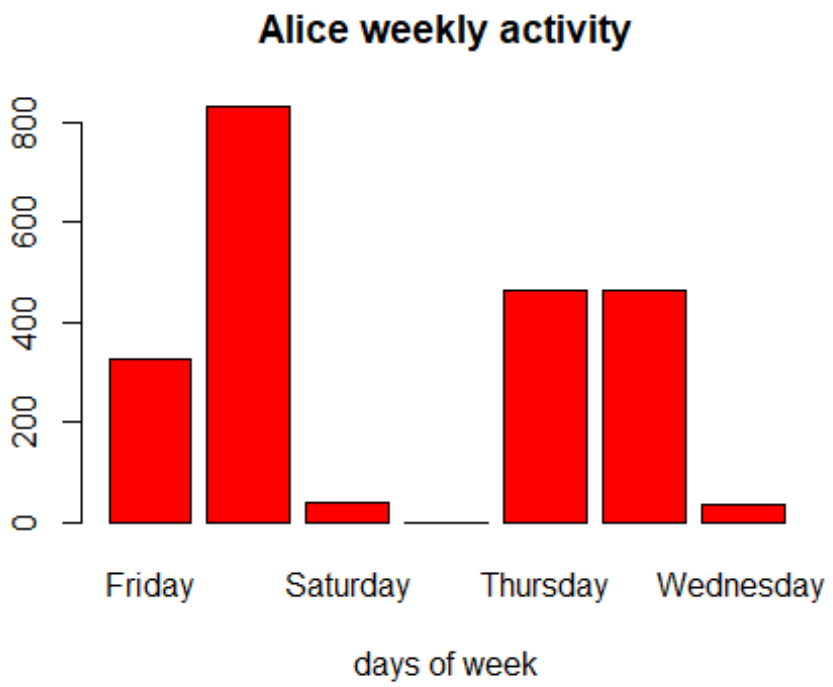
```
dayst<-table(train_data$Days,train_data$target)
f1=barplot(t(dayst),beside = TRUE,col=1:2,legend=TRUE,xlab="days of
week",main="weekly activity")
```




```
s=subset(train_data,target==0)

dayst1<-table(s$Days,s$target)
f2=barplot(t(dayst),col=1,xlab="days of week",main="others weekly activity")
```



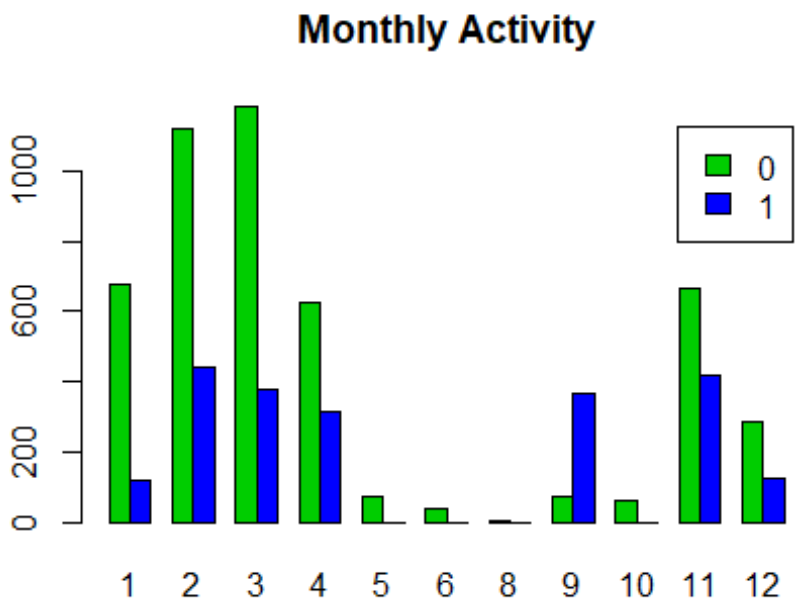

```
alice=subset(train_data,target==1)
alc=table(alice$Days,alice$target)
f3=barplot(t(alc),col=2,xlab="days of week",main="Alice weekly activity")
```



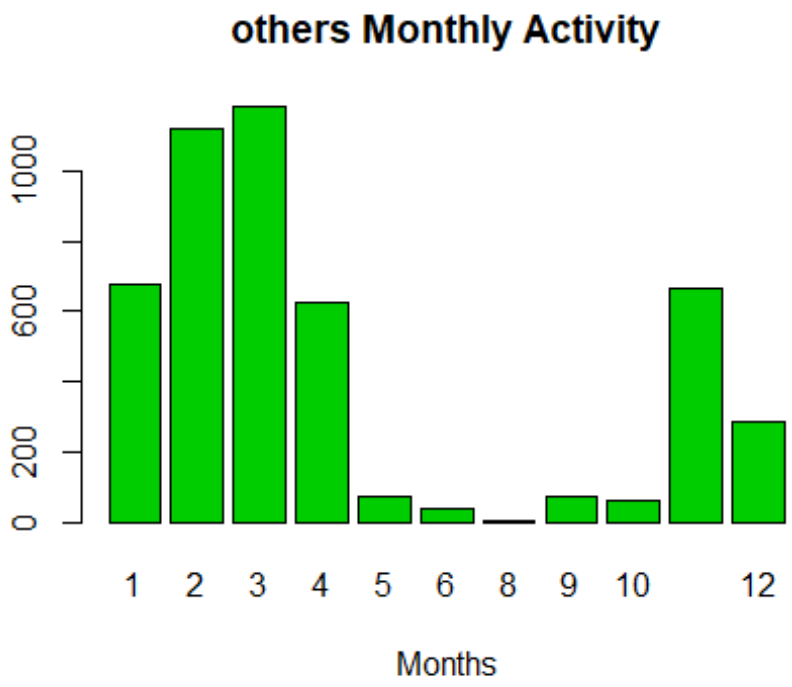
EXcept Wednesday

most oftenly Alice is online on Weekdays

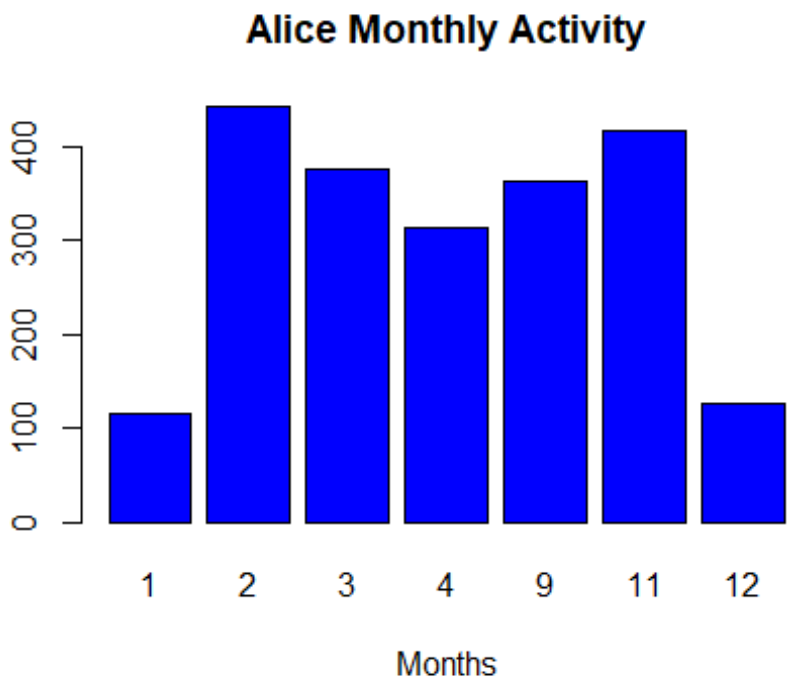
```
monthst<-table(train_data$Month,train_data$target)
f4=barplot(t(monthst),beside = TRUE,col=3:4,legend=TRUE,main="Monthly
Activity")
```




```
s=subset(train_data,target==0)
#View(s)
monthst1<-table(s$Month,s$target)
f5=barplot(t(monthst1),col=3,xlab="Months",main="others Monthly Activity")
```

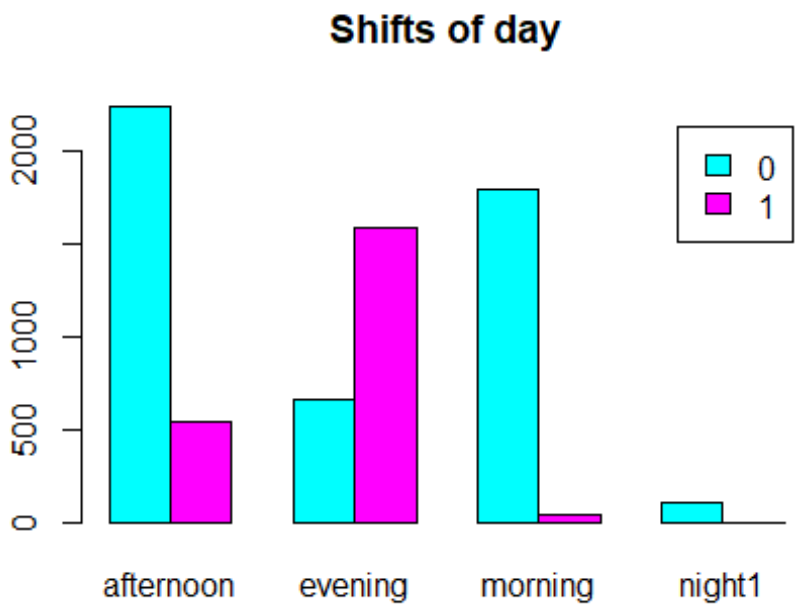



```
d=subset(train_data,target==1)
alice=table(d$Month,d$target)
f6=barplot(t(alice),col=4,xlab="Months",main="Alice Monthly Activity")
```



Alice is active at at spring and some autumn months. She is a student.

```
shift<-table(train_data$Shift,train_data$target)
fig7=barplot(t(shift),beside = TRUE,col=5:6,legend=TRUE,main="Shifts of day")
```



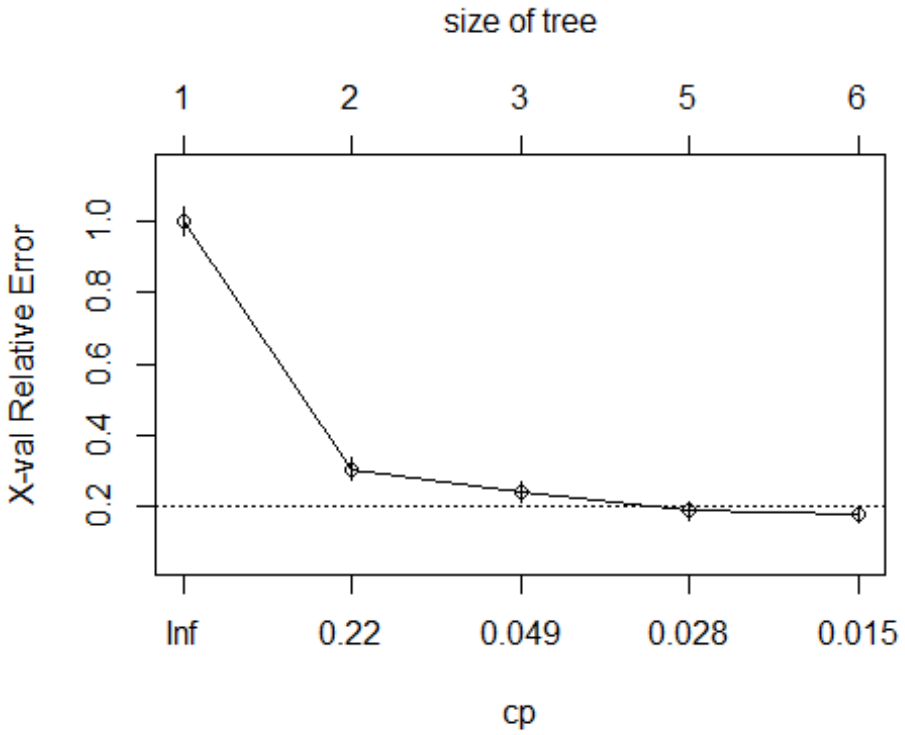
We get idea that Alice is most oftenly online at Evening

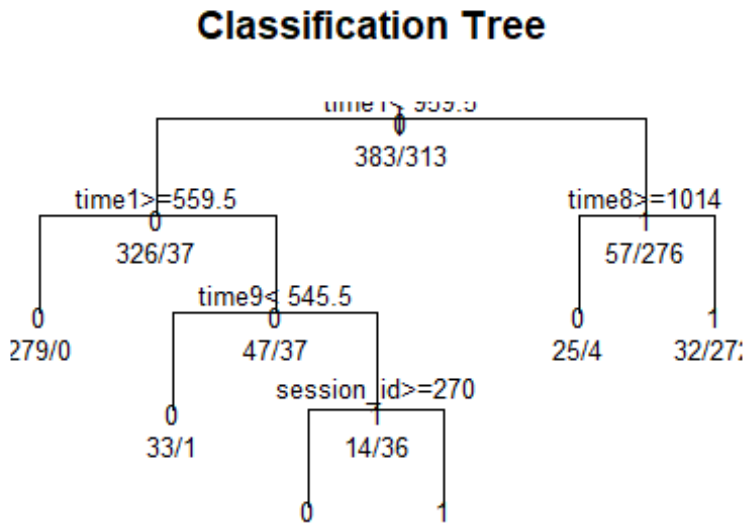
Q.1. Build a classi???er which can identify whether the user is Alice using 10-fold cross validation. Report the cross-validation based error for your ???nal classi???er. Do mention what all classi???ers you tried and justify your choice. Ans: We used Decision Tree, Random Forest, SVM, Logistic Regression.

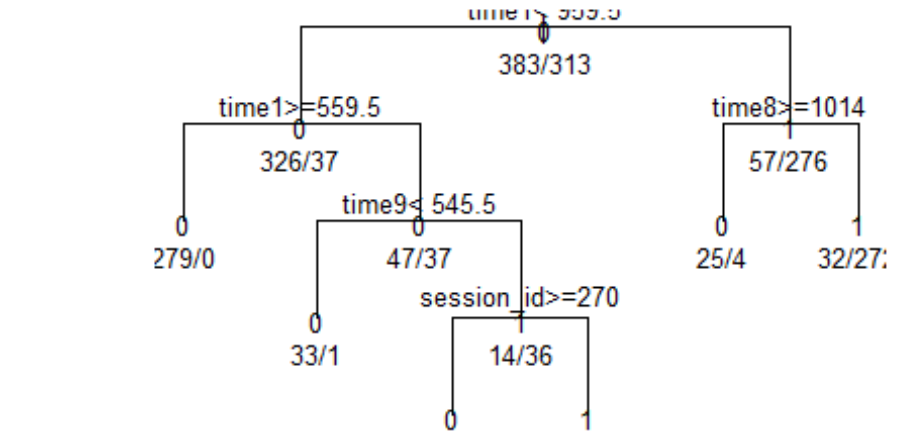
(1) Decision tree

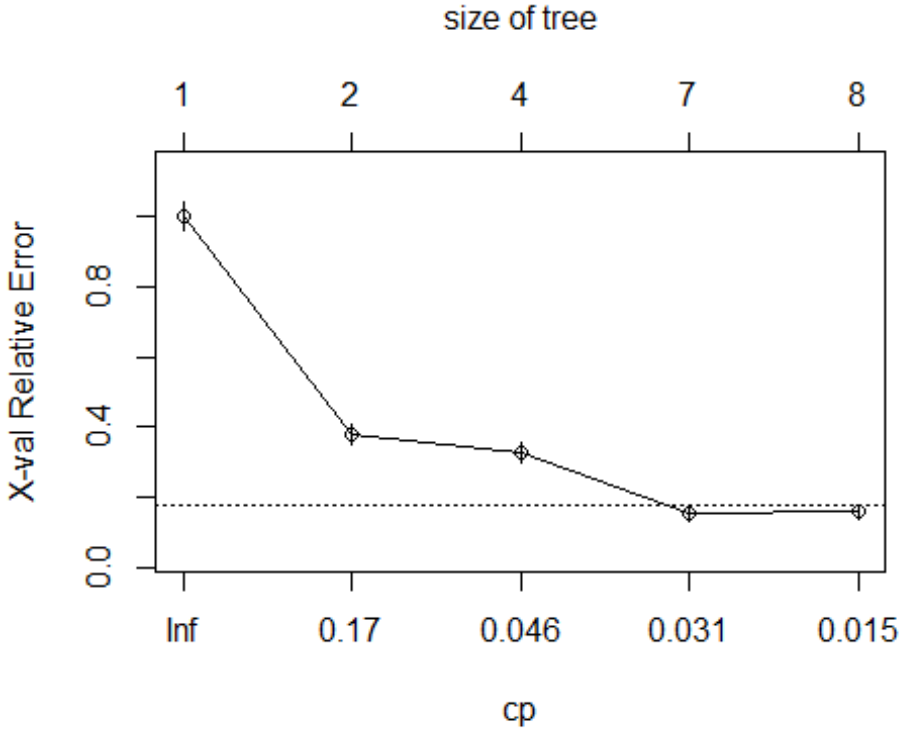
```
######## Tree ######
library(rpart)

## Warning: package 'rpart' was built under R version 3.5.3

library(chron)

## Warning: package 'chron' was built under R version 3.5.3

#timep=train_data$time10.1
#shift=cut(chron::times(timep) , breaks = (1/24) * c(1,5,11,16,19,24),
#    labels = c("night", "morning", "afternoon", "evening", "night1"))
#shift=write.csv(shift,"C:/Users/Pooja/Desktop/shift.csv")
train_data = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")
View(train_data)
folds <- cut(seq(1,nrow(train_data)),breaks=10,labels=FALSE)
ac = c()
for(f in 1:10)
{
  indexes<-which(folds==f,arr.ind=TRUE)
```

```
  train <- train_data[indexes,]
  test <- train_data[-indexes,]
  # tree
  fit<- rpart(target~. , method="class",data=train)

  #printcp(fit) # display the results

  plotcp(fit) # visualize cross-validation results

  # summary(fit) # detailed summary of splits

  # plot tree
  plot(fit, uniform=TRUE, main="Classification Tree ")
  text(fit, use.n=TRUE, all=TRUE, cex=.8)

  # prune the tree
  pfit<- prune(fit, cp= fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])

  #plot the pruned tree
  plot(pfit, uniform=TRUE,main="Pruned Classification Tree ")
  text(pfit, use.n=TRUE, all=TRUE, cex=.8)
  predictions = predict(fit,test,type="class")

  ac[f]<-sum(predictions==test$target)/length(test$target)

}
```

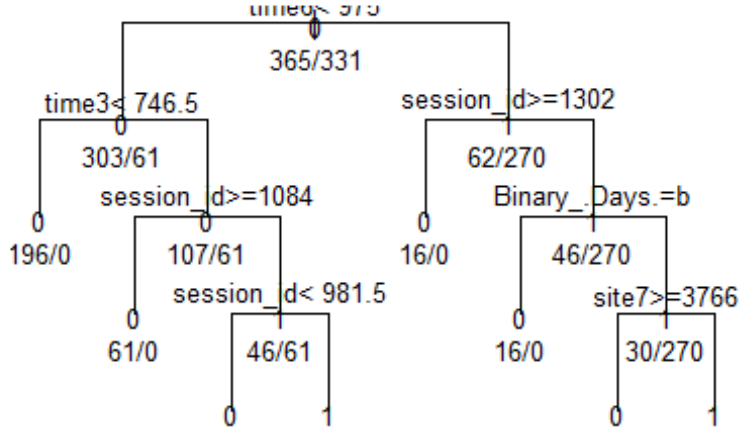

size of tree
1
2
3
5
6
X-val Relative Error
0.2
0.4
0.6
0.8
1.0
Inf
0.22
0.049
0.028
0.015
cp


Classification Tree
383/313
time1>=559.5
0
326/37
time8>=1014
1
57/276
time9< 545.5
0
47/37
0
279/0
0
25/4
1
32/27
session_id>=270
0
33/1
1
14/36
0
1

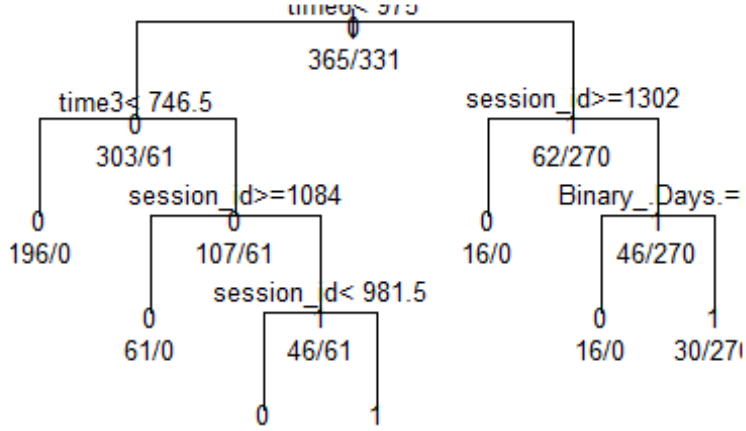

Pruned Classification Tree
383/313
time1>=559.5
0
326/37
time8>=1014
1
57/276
time9< 545.5
0
47/37
0
279/0
0
25/4
1
32/27
session_id>=270
1
14/36
0
33/1
0
1

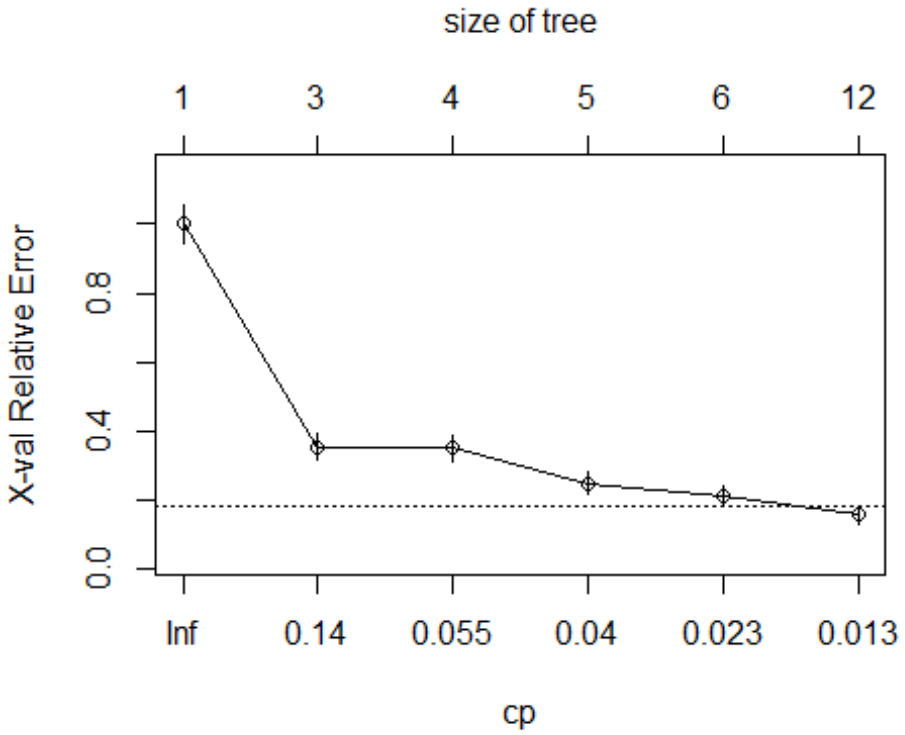

size of tree
1
2
4
7
8
X-val Relative Error
0.0
0.4
0.8
Inf
0.17
0.046
0.031
0.015
cp

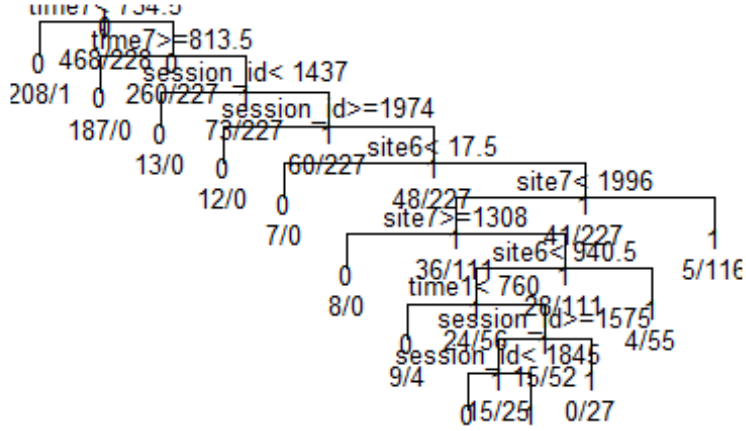
Classification Tree



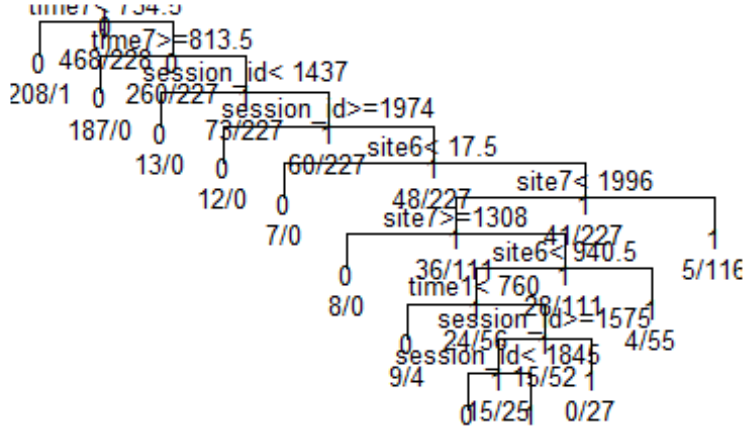
Pruned Classification Tree

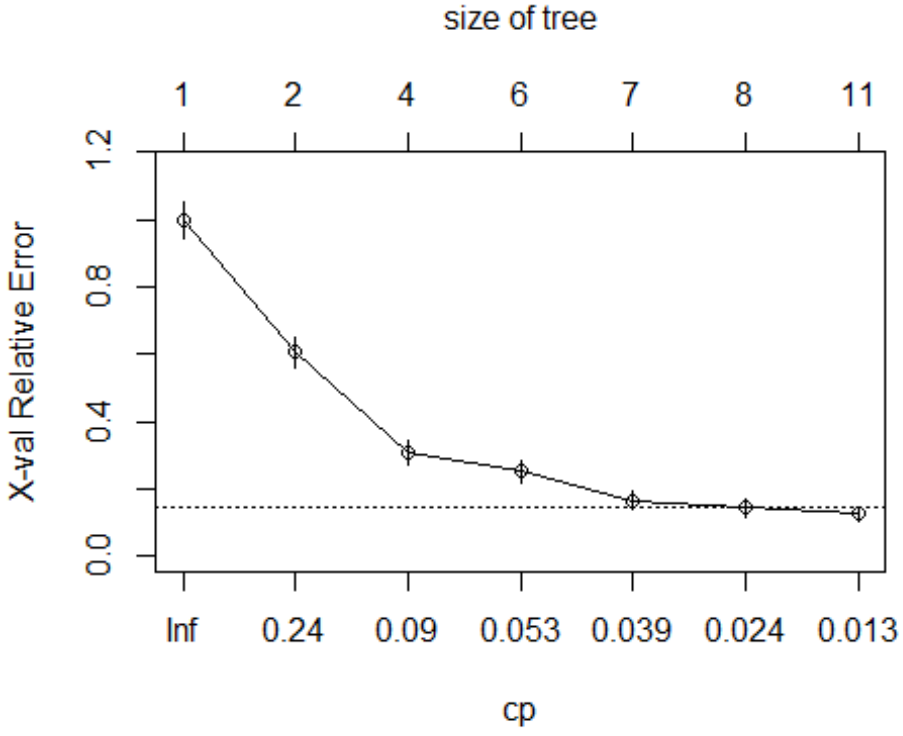



size of tree
1
3
4
5
6
12
X-val Relative Error
0.0
0.4
0.8
Inf
0.14
0.055
0.04
0.023
0.013
cp

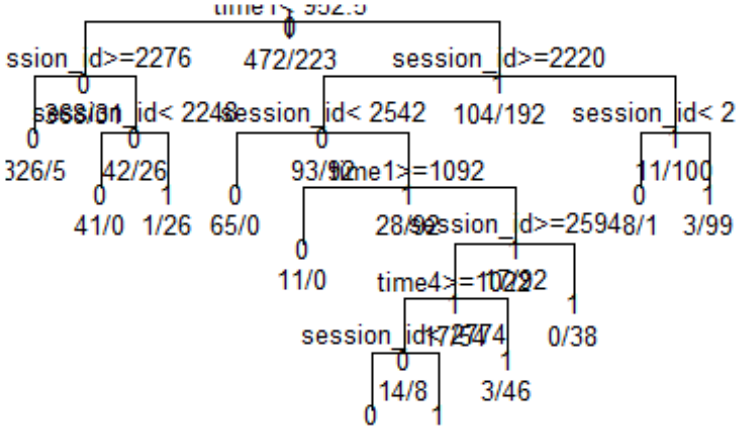

Classification Tree
time7>=813.5
session_id< 1437
session_id>=1974
site6< 17.5
site7< 1996
site7>=1308
site6< 940.5
time1< 760
session_id>=1575
session_id< 1845
0
468/228
208/1
260/227
187/0
73/227
13/0
60/227
12/0
48/227
7/0
41/227
36/111
8/0
28/111
24/56
4/55
9/4
15/52
15/25
0/27
5/116

Pruned Classification Tree

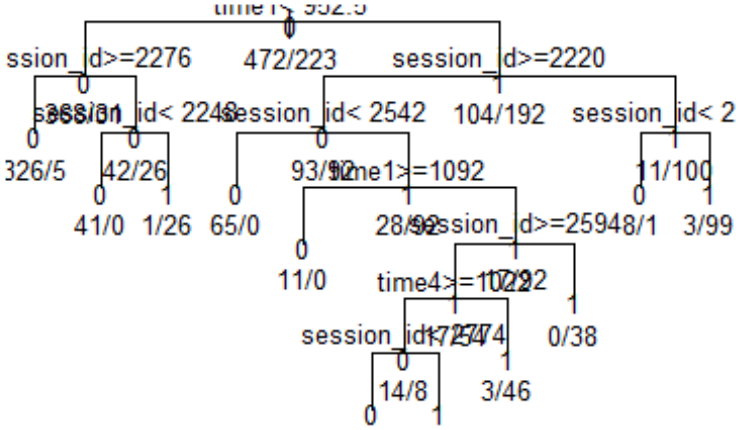


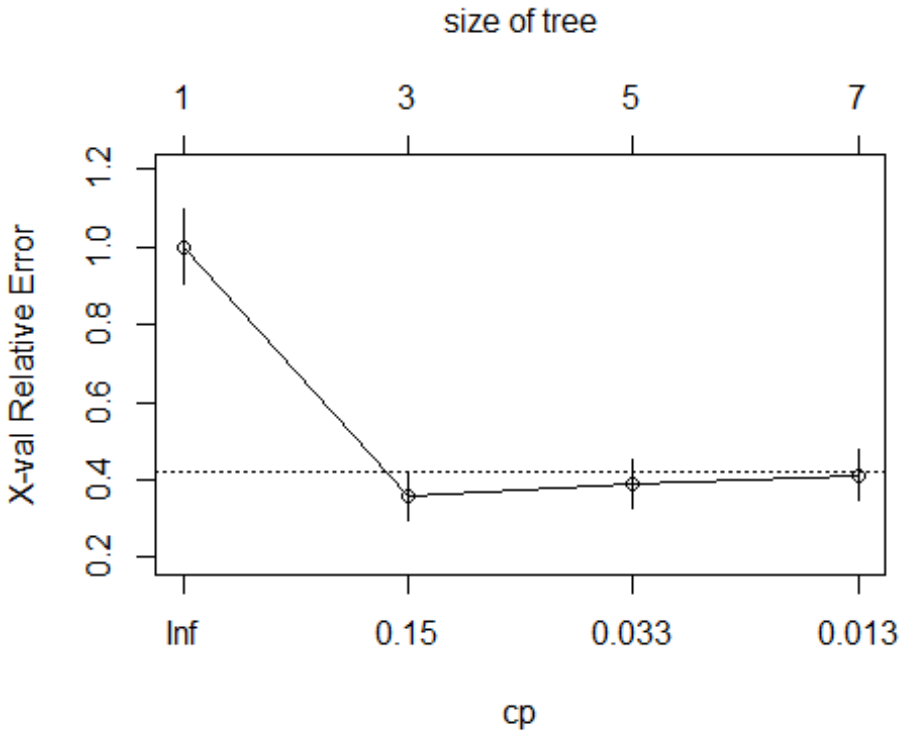


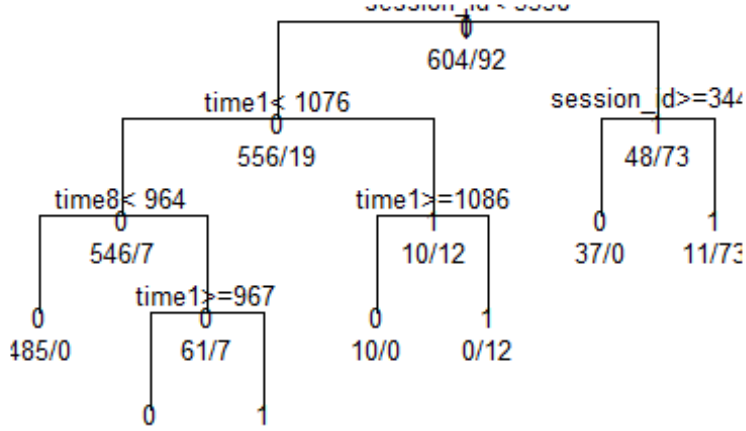
Classification Tree

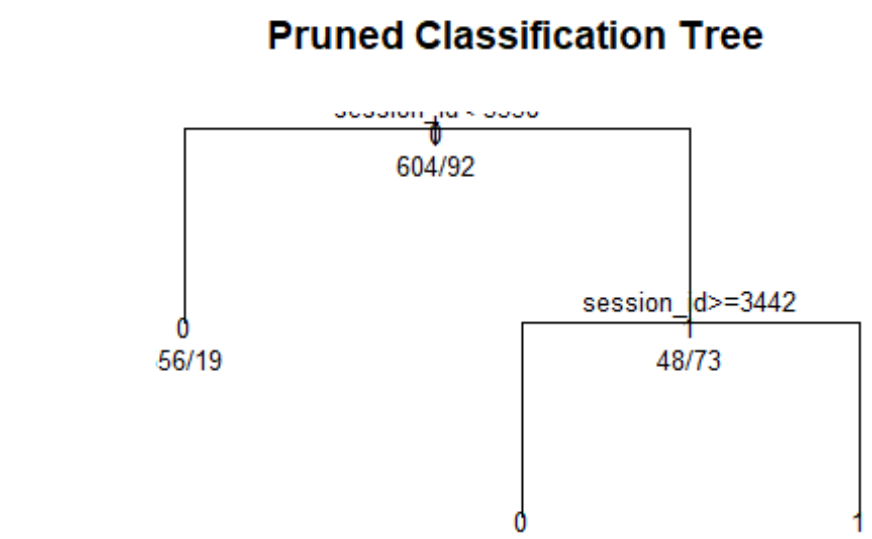


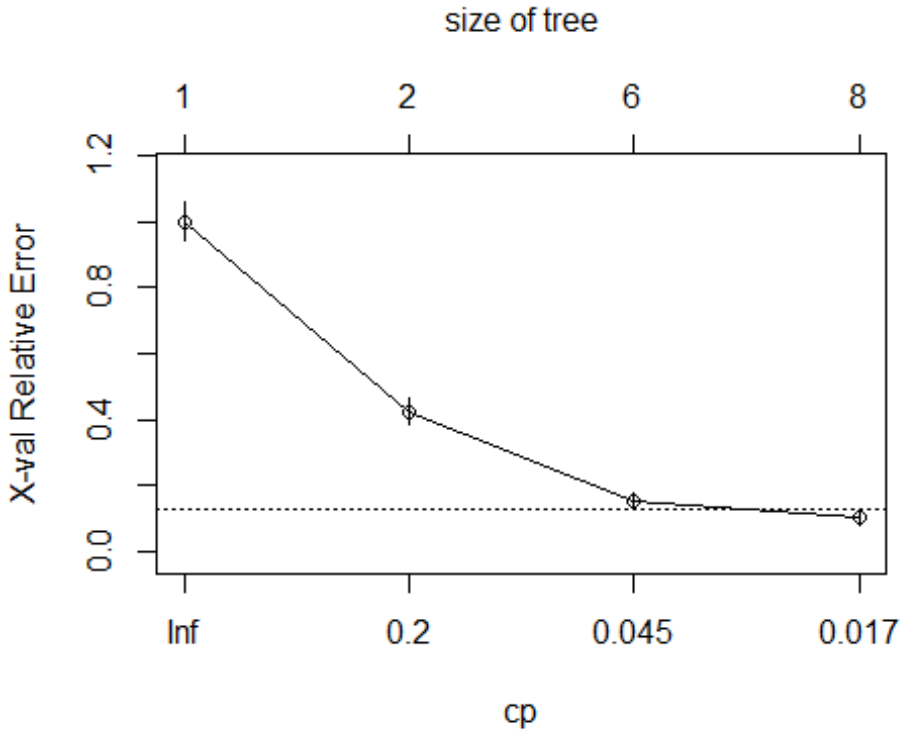
Pruned Classification Tree



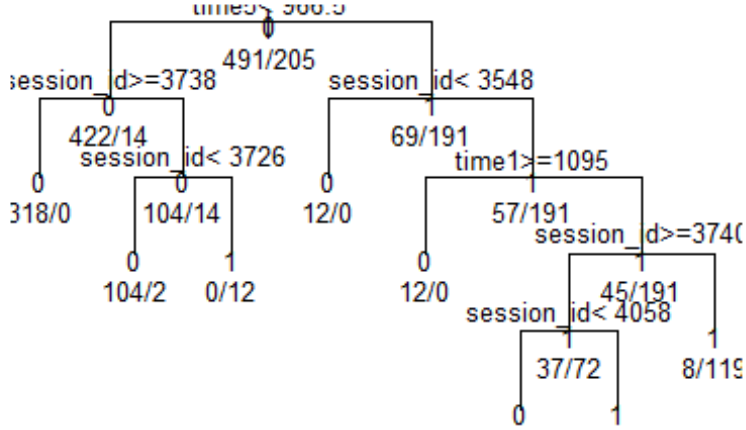

size of tree
1
3
5
7
X-val Relative Error
0.2
0.4
0.6
0.8
1.0
1.2
Inf
0.15
0.033
0.013
cp


Classification Tree
604/92
time1< 1076
0
556/19
session_id>=34
48/73
time8< 964
546/7
time1>=1086
10/12
0
37/0
1
11/73
time1>=967
485/0
61/7
10/0
0/12

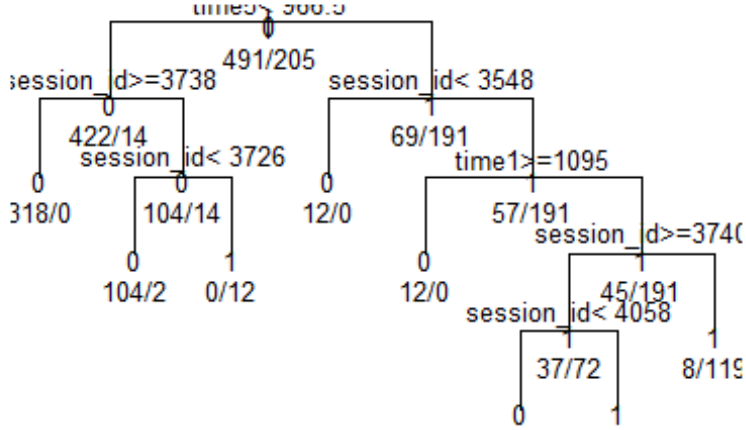

Pruned Classification Tree
604/92
0
56/19
session_id>=3442
48/73
0
1

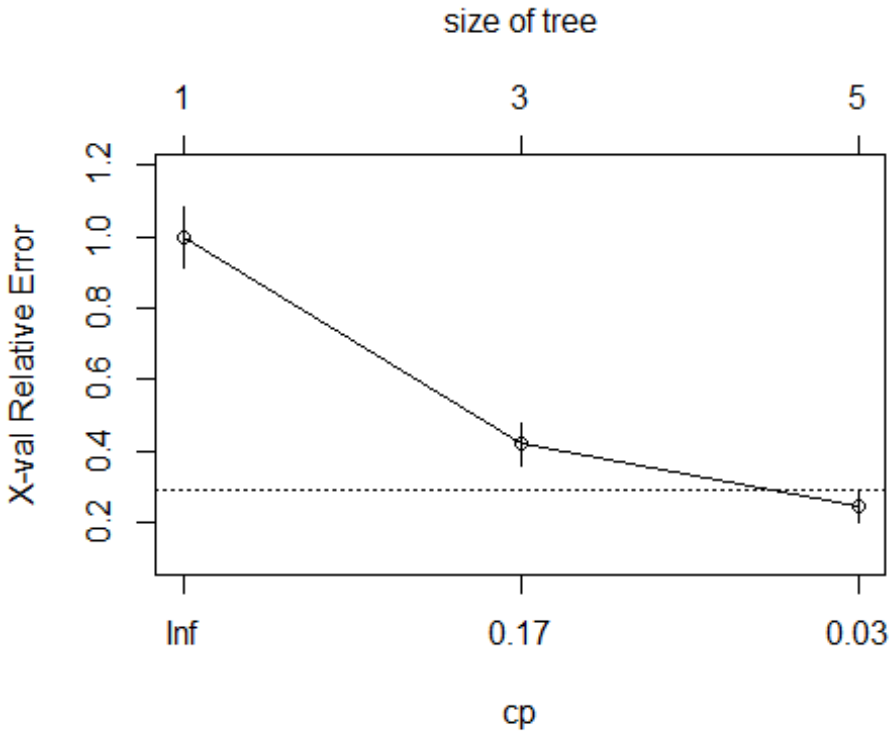

size of tree
1
2
6
8
X-val Relative Error
0.0
0.4
0.8
1.2
Inf
0.2
0.045
0.017
cp

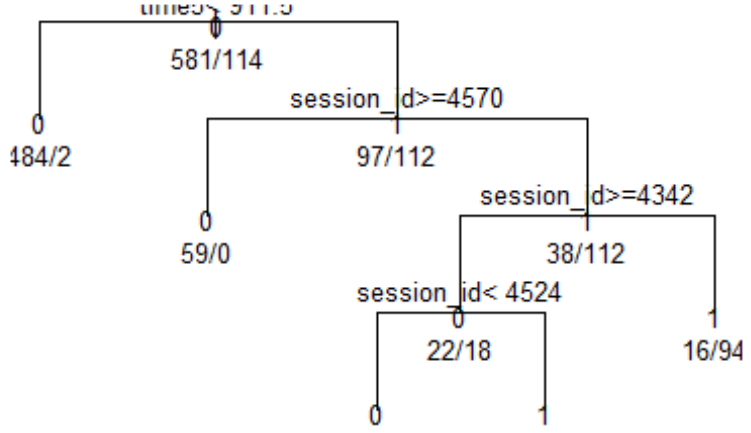
Classification Tree

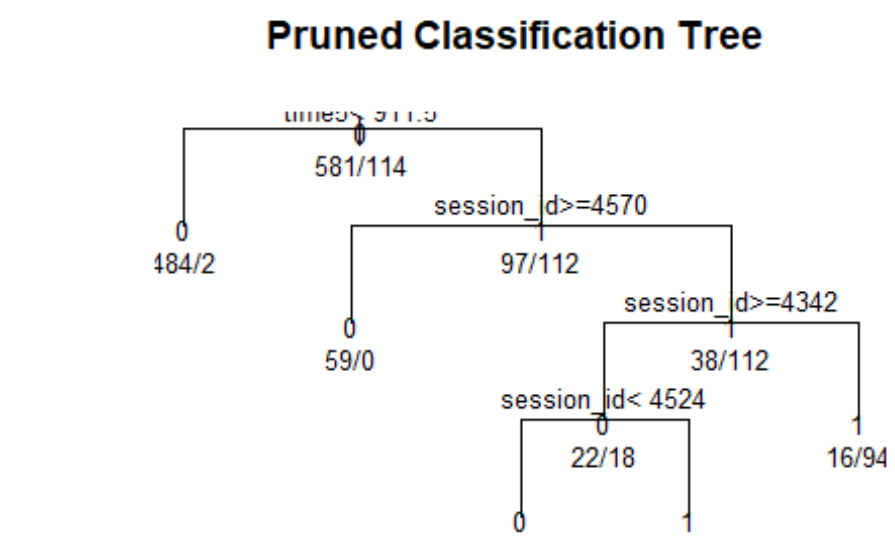


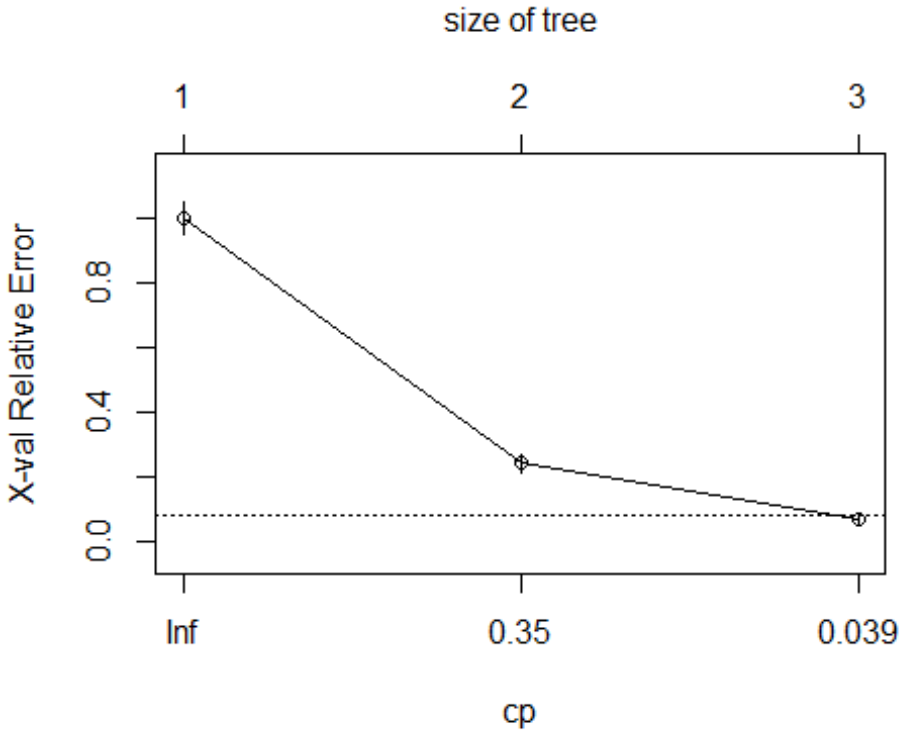
Pruned Classification Tree



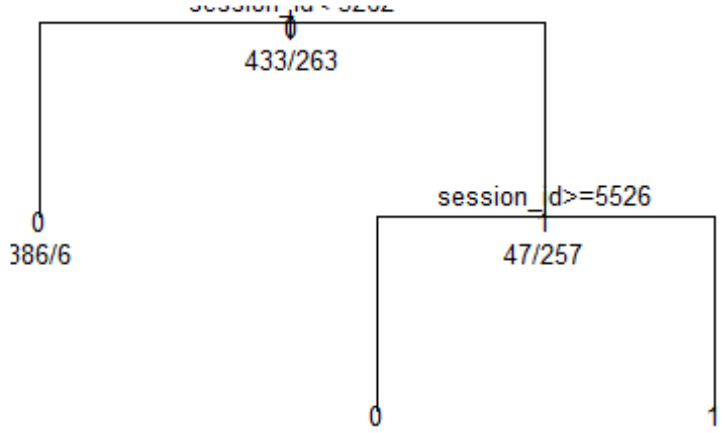

size of tree
1
3
5
X-val Relative Error
0.2
0.4
0.6
0.8
1.0
1.2
Inf
0.17
0.03
cp


Classification Tree
0
581/114
session_id>=4570
0
484/2
97/112
0
59/0
session_id>=4342
38/112
session_id< 4524
0
22/18
1
16/94
0
1

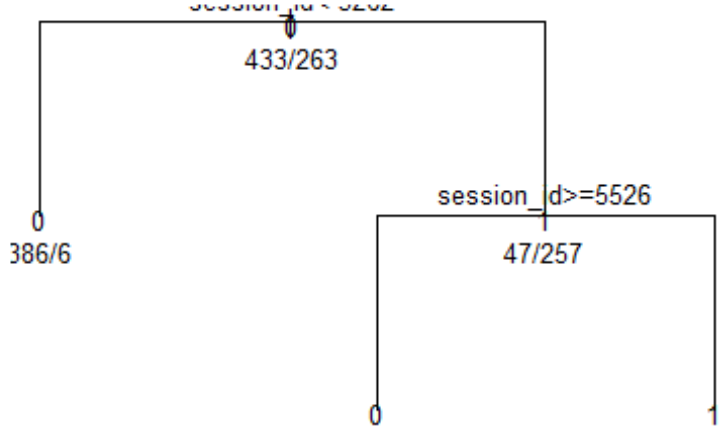

Pruned Classification Tree
581/114
session_id>=4570
0
484/2
97/112
session_id>=4342
0
59/0
38/112
session_id< 4524
0
22/18
1
16/94
0
1

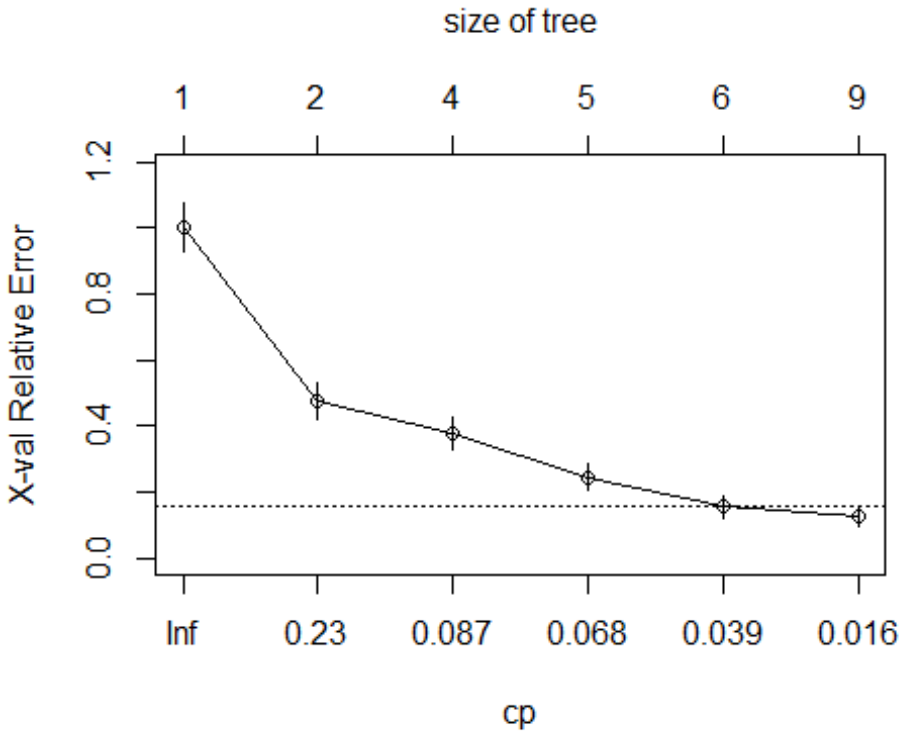

size of tree
1
2
3
X-val Relative Error
0.0
0.4
0.8
Inf
0.35
0.039
cp

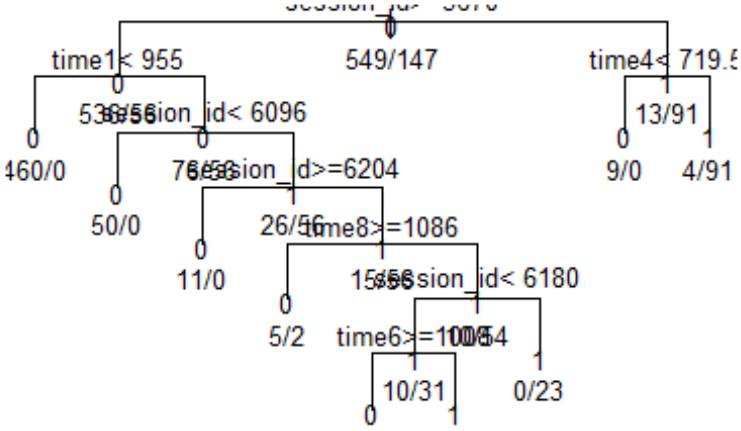
Classification Tree

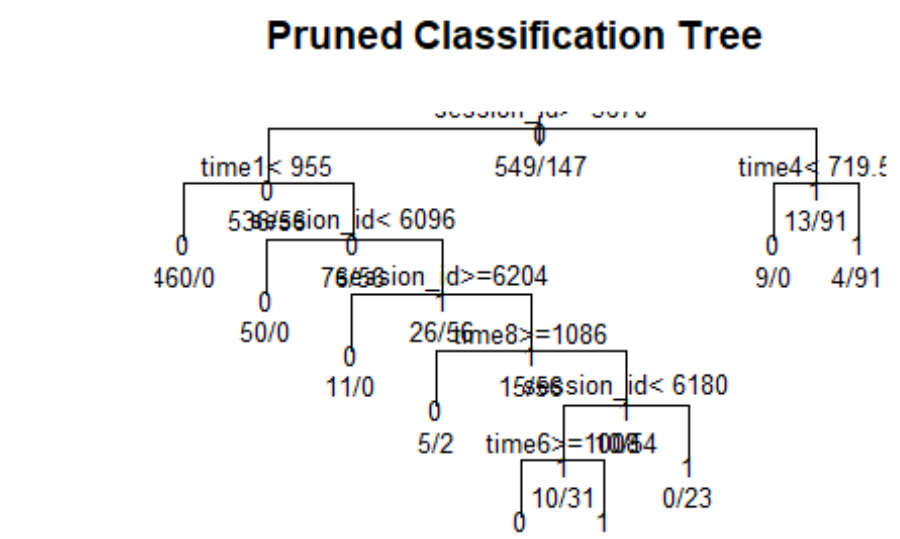


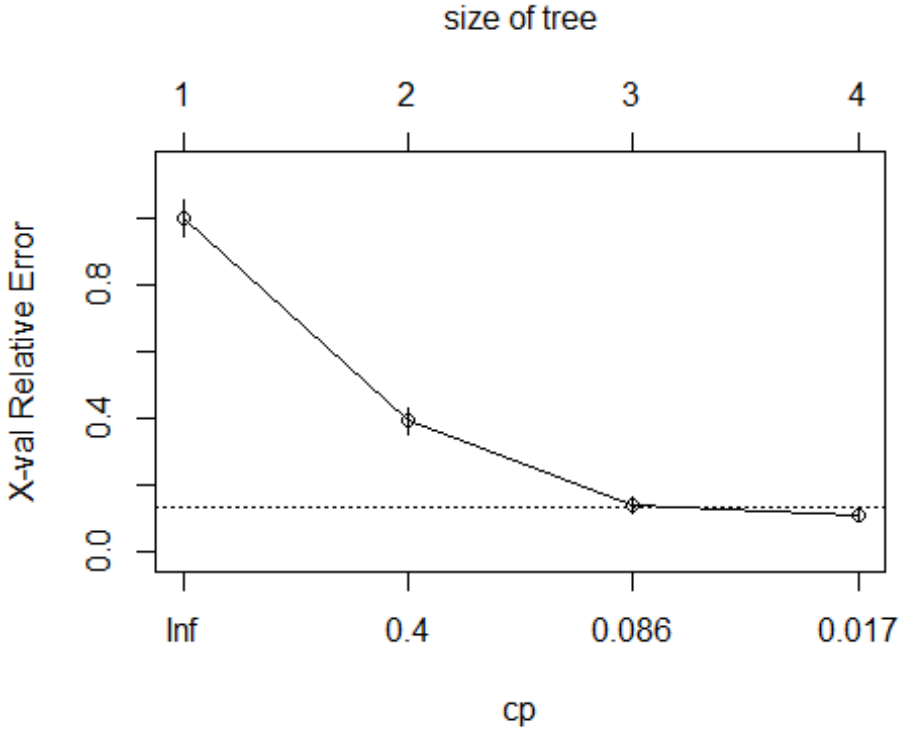
Pruned Classification Tree

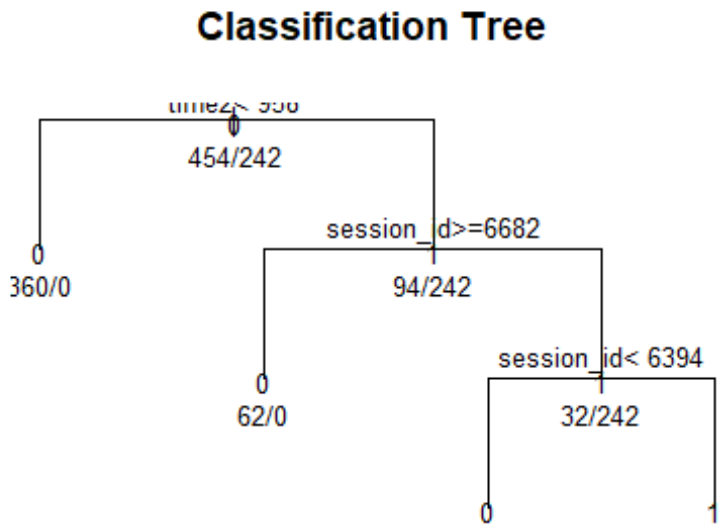


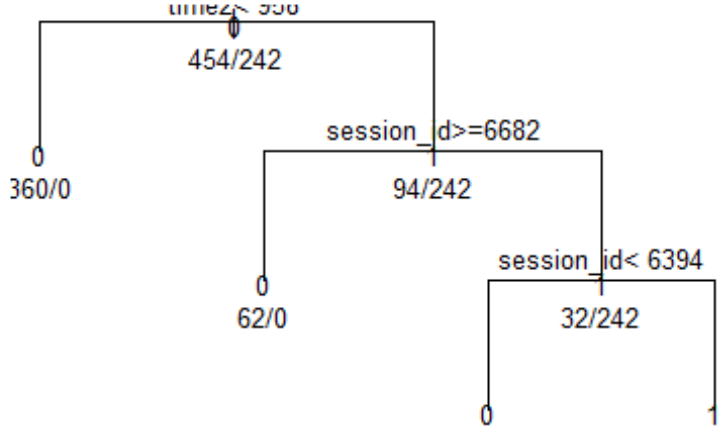

size of tree
1
2
4
5
6
9
X-val Relative Error
0.0
0.4
0.8
1.2
Inf
0.23
0.087
0.068
0.039
0.016
cp


Classification Tree
time1< 955
549/147
time4< 719.5
13/91
0
460/0
50/0
11/0
5/2
9/0
4/91
10/31
0/23


Pruned Classification Tree
time1< 955
549/147
time4< 719.5
13/91
9/0
4/91
460/0
50/0
11/0
5/2
10/31
0/23


size of tree
1
2
3
4
X-val Relative Error
0.0
0.4
0.8
Inf
0.4
0.086
0.017
cp






```
mean(ac)
```

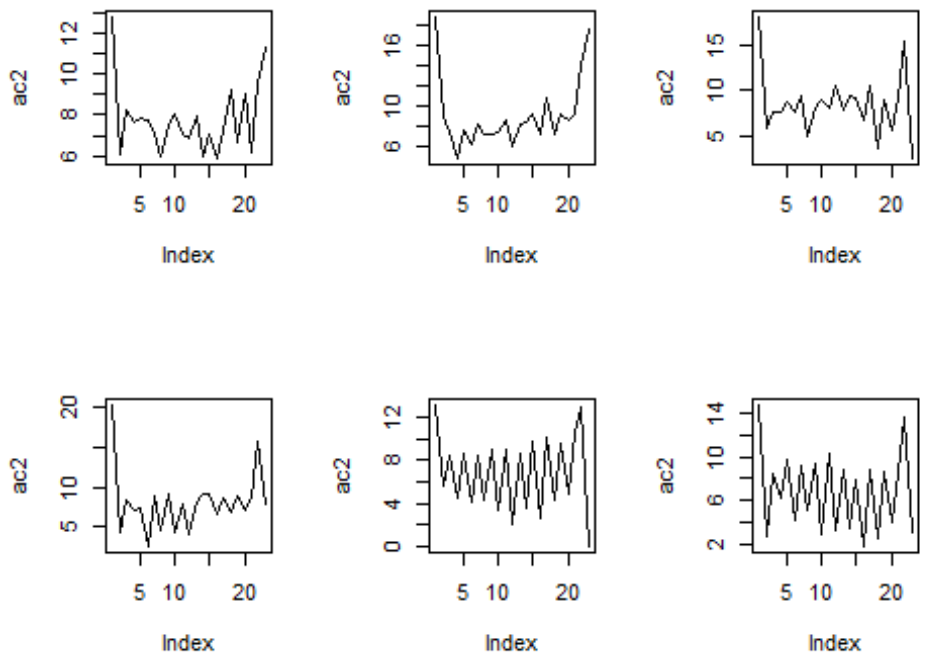
```
## [1] 0.7124963
```

Random Forest

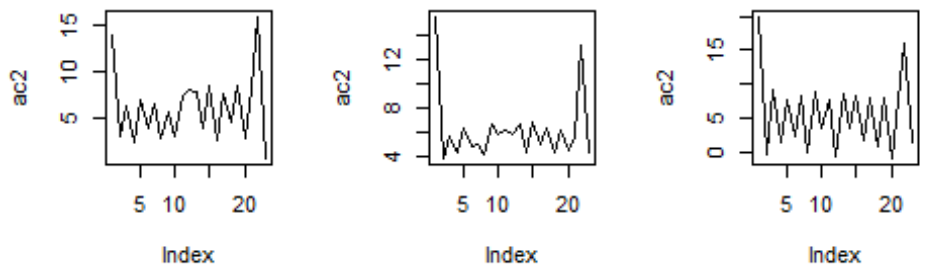
```
####### Random Forest ########

train_data = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")
train_data$target<-as.factor(train_data$target)
#table(train_data$target)
par(mfrow=c(2,3))
folds <- cut(seq(1,nrow(train_data)),breaks=10,labels=FALSE)
ac = c()
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

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

library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

library(e1071)

## Warning: package 'e1071' was built under R version 3.5.3

for(f in 1:10)
{
  indexes<-which(folds==f,arr.ind=TRUE)
  test <- train_data[-indexes,]
  train <- train_data[indexes,]
  #Random Forest
  set.seed(222)
  rf <- randomForest(target~.,data=train,ntree =200, importance =
TRUE,proximity = TRUE)
  #print(rf)
  #attributes(rf)

  #predictions & confusion matrix
  p1<-predict(rf , train)
```

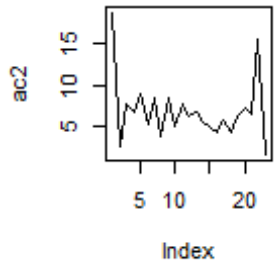
```
  #predictions & confusion matrix - test data
  p2<-predict(rf, test)

  #error rate of random forest
  #plot(rf)

  #tune mtry
  t<-tuneRF(train[,-22],train[,22],stepFactor = 1,plot=FALSE,ntreeTry =
300,trace=TRUE,improve=0.05)

  #variable importance
  imp<-varImp(rf, sort=T, n.var = 10, main="Top 10-variable importance")
  vi <- (importance(rf))
  #print(vi)
  ac2 <- vi[,3]
  plot(ac2, type = "l")
  varUsed(rf)
  q <- sort(ac2)
  ex <- q[-c(1:28)]
  #print(ex)
  ac[f]<-sum(p2==test$target)/length(test$target)

}
```

```
## mtry = 7  OOB error = 134.0791
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 4.484425
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 0.8547782
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 3.040027
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 1.195377
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 0.189229
## Searching left ...
## Searching right ...
```




```
## mtry = 7  OOB error = 0.6591689
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 0.07180887
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 0.1516797
## Searching left ...
## Searching right ...

## mtry = 7  OOB error = 1.077137
## Searching left ...
## Searching right ...

mean(ac)

## [1] 0.7194583
```





SVM

```
######## SVM #####
train_data = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")
folds <- cut(seq(1,nrow(train_data)),breaks=10,labels=FALSE)
ac = c()
library(e1071)
for(f in 1:10)
{
  indexes<-which(folds==f,arr.ind=TRUE)
  test <- train_data[-indexes,]
  train <- train_data[indexes,]
  svmfit <- svm(target ~ ., data = train)
  summary(svmfit)

  pred <- predict(svmfit, test,type="response")
  res2 <- ifelse(pred<=0.5,0,1)
  ta1<-table(res2,test$target)
  ac[f] = sum(res2==test$target)/length(test$target)

}
mean(ac)

## [1] 0.6433513
```

Logistic Regression

```
######### Logistic Regression #######
```

```
train_data = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")

train_data$target<-as.factor(train_data$target)
table(train_data$target)

##
##    0    1
## 4800 2158

par(mfrow=c(2,3))
folds <- cut(seq(1,nrow(train_data)),breaks=10,labels=FALSE)
acc = c()
for(f in 1:10)
{
  indexes<-which(folds==f,arr.ind=TRUE)
  test <- train_data[-indexes,]
  train <- train_data[indexes,]
  train[!complete.cases(train),]
  train<-na.omit(train)
  model <- glm(target~.,data = train , family = "binomial",na.action =
na.pass)
  predictions <- predict(model,newdata=test[,-24],type = "response")
  res1 <- predict(model,train)
  res2 <- ifelse(predictions<=0.5,0,1)
  ta1<-table(res2,test$target)
  acc[f]<-sum(diag(ta1))/sum(ta1)

}

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

mean(acc)

## [1] 0.5789973
```

From Above 4 classifier The accurracy of 1) Decision Tree = 0.7124 2) Random Forest = 0.7194 3) SVM = 0.6433 4) Logistic Regression = 0.5789

The best Classifier is Random Forest Because it have highest accuracy than Decision tree , SVM , and logistic regression

Q.2. Prepare a ???le which gives prediction for the test data. (Your score for this question will be proportional to your prediction accuracy.)

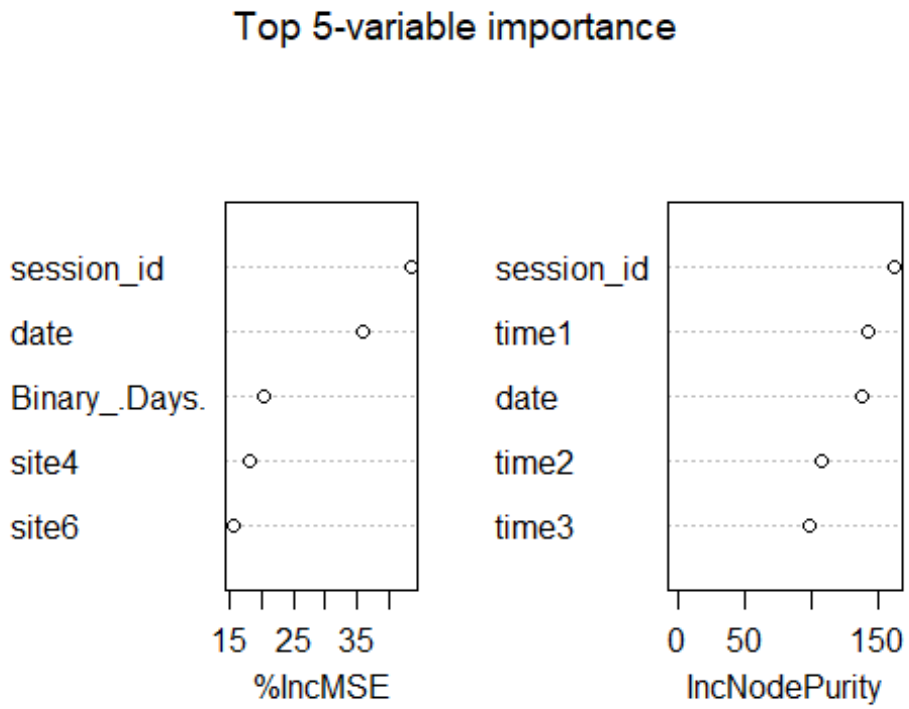
Ans:

```
library(randomForest)
library(caret)
library(e1071)
train = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")
test = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Final_test.csv")

set.seed(222)
#tune mtry
t<-tuneRF(train[,-22],train[,22],stepFactor =
1,plot=FALSE,ntreeTry=300,trace=TRUE,improve=0.05)

## mtry = 7  OOB error = 54.28335
## Searching left ...
## Searching right ...

rf <- randomForest(target~.,data=train,ntree =200, mtry = 7 , importance =
TRUE,proximity = TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

#print(rf)
#attributes(rf)

#predictions & confusion matrix
p1<-predict(rf , test)
#predictions & confusion matrix - test data
a1<-ifelse(p1<=.5,0,1)
a1

##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   0   0   0   1   1   1   1   1   0   1   0   0   0   0   0   0   0   0
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##   0   0   0   0   0   0   0   0   1   1   1   1   1   1   1   1   1   1
##  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##   1   0   0   0   1   0   0   1   0   0   0   0   0   0   0   0   1   1
##  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##   1   1   1   1   1   0   0   0   0   1   0   0   0   1   0   0   0   0
##  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##   0   0   0   0   0   0   0   0   0   0   1   1   1   1   1   1   0   0
##  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
##   0   0   0   0   0   1   1   1   1   0   0   0   0   0   0   0   0   0
## 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
##   0   1   0   0   0   0   0   0   1   1   1   0   0   0   0   0   0   0
## 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
##   0   0   0   0   0   0   0   0   1   0   1   0   0   0   0   0   0   0
## 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   1   1   0   0   0   0   0   0   0   0   0   0   1   1   1   0   0   1
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
```

```
##   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0
## 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
##   0   0   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
## 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
##   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0
## 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
##   0   1   1   0   1   1   0   1   1   1   1   1   1   1   1   1   1   1
## 289 290 291 292 293 294 295 296 297 298 299 300
##   1   0   0   0   0   0   0   0   0   0   0   0

#submission <- write.csv(a1,"C:/Users/Pooja/Desktop/DM Ass
1/submission_file.csv")
```

Q.3. Which variables turn out to be the most useful? Try to justify their usefulness with the help of appropriate parts of classi???cation output and/or exploratory statistical techniques. ANS:

```
train_data = read.csv("C://Users//Pooja//Desktop//DM Ass 1//Train_data.csv")
#Random Forest
  set.seed(222)
  rf <- randomForest(target~.,data=train,ntree =200, importance =
TRUE,proximity = TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

  #print(rf)
  #variable importance
  impvariable<-varImp(rf, sort=T, n.var = 10, main="Top 10-variable
importance")
  varImpPlot(rf, sort=T, n.var = 5, main="Top 5-variable importance")
```




```
vi <- (importance(rf))
```

According to %IncMSE The 5 most Important variables are session_id, date, Binary_Days, site4, site6. According to IncNodePurity the 5 most impportant variables are session_id, time1, date, time2, time3