Problem- prediction of the number of comments in the upcoming 24 hours on

those blogs, The train data was generated from different base times that may

temporally overlap. Therefore, if you simply split the train into disjoint partitions,

the underlying time intervals may overlap. Therefore, the you should use the

provided, temporally disjoint train and test splits to ensure that the evaluation is

fair.

a. Read the dataset and identify the right features

b. Clean dataset, impute missing values and perform exploratory data analysis.

c. Visualize the dataset and make inferences from that

d. Perform any 3 hypothesis tests using columns of your choice, make conclusions

e. Create a linear regression model to predict the number of comments in the next 24 hours

(relative to basetime)

f. Fine tune the model and represent important features

g. Interpret the summary of the linear model

h. Report the test accuracy vs. the training accuracy

i. Interpret the final model coefficients

j. Plot the model result and compare it with assumptions of the model

#set working directory

path <- "~/December 2016/RF\_Tutorial"

setwd(path)

#load libraries

library(data.table)

library(mlr)

library(h2o)

#set variable names

setcol <- c("age",` `"workclass",` `"fnlwgt",` `"education",` `"education-num",` `"marital-status",` `"occupation",` `"relationship",` `"race",` `"sex",` `"capital-gain",` `"capital-loss",` `"hours-per-week",` `"native-country",` `"target")

#load data

train <- read.table("adultdata.txt",header = F,sep = ",",col.names = setcol,na.strings = c(" ?"),stringsAsFactors = F)

test <- read.table("adulttest.txt",header = F,sep = ",",col.names = setcol,skip = 1, na.strings = c(" ?"),stringsAsFactors = F)

setDT(train)

setDT(test)

dim(train)

dim(test)

str(train)

str(test)

#check missing values

table(is.na(train))

FALSE   TRUE

484153  4262

sapply(train, function(x) sum(is.na(x))/length(x))\*100

table(is.na(test))

FALSE  TRUE

242012 2203

sapply(test, function(x) sum(is.na(x))/length(x))\*100`<br/>

imp1 <- impute(data = train,target = "target",classes = list(integer=imputeMedian(), factor=imputeMode()))

imp2 <- impute(data = test,target = "target",classes = list(integer=imputeMedian(), factor=imputeMode()))

train <- imp1$data

test <- imp2$data

setDT(train)[,.N/nrow(train),target]

target     V1

1: <=50K   0.7591904

2: >50K    0.2408096

setDT(test)[,.N/nrow(test),target]

target     V1

1: <=50K.  0.7637737

2: >50K.   0.2362263

test[,target := substr(target,start = 1,stop = nchar(target)-1)]

library(stringr)

char\_col <- colnames(train)[sapply(train,is.character)]

for(i in char\_col)

set(train,j=i,value = str\_trim(train[[i]],side = "left"))

fact\_col <- colnames(train)[sapply(train,is.character)]

for(i in fact\_col)

set(train,j=i,value = factor(train[[i]]))

for(i in fact\_col)

set(test,j=i,value = factor(test[[i]]))

#create a task

traintask <- makeClassifTask(data = train,target = "target")

testtask <- makeClassifTask(data = test,target = "target")

#create learner

bag <- makeLearner("classif.rpart",predict.type = "response")

bag.lrn <- makeBaggingWrapper(learner = bag,bw.iters = 100,bw.replace = TRUE)

#set 5 fold cross validation

rdesc <- makeResampleDesc("CV",iters=5L)

#set parallel backend (Windows)

library(parallelMap)

library(parallel)

parallelStartSocket(cpus = detectCores())

r <- resample(learner = bag.lrn , task = traintask, resampling = rdesc, measures = list(tpr,fpr,fnr,fpr,acc) ,show.info = T)

#[Resample] Result:

# tpr.test.mean=0.95 #fnr.test.mean=0.0505 #fpr.test.mean=0.487 #acc.test.mean=0.845

#make randomForest learner

rf.lrn <- makeLearner("classif.randomForest")

rf.lrn$par.vals <- list(ntree = 100L, importance=TRUE)

r <- resample(learner = rf.lrn, task = traintask, resampling = rdesc, measures = list(tpr,fpr,fnr,fpr,acc), show.info = T)

# Result:

# tpr.test.mean=0.996 #fpr.test.mean=0.72 #fnr.test.mean=0.0034 #acc.test.mean=0.825

#set cutoff

rf.lrn$par.vals <- list(ntree = 100L, importance=TRUE, cutoff = c(0.75,0.25))

r <- resample(learner = rf.lrn, task = traintask, resampling = rdesc, measures = list(tpr,fpr,fnr,fpr,acc), show.info = T)

#Result: tpr.test.mean=0.934 #fpr.test.mean=0.43 #fnr.test.mean=0.0662 #acc.test.mean=0.846

getParamSet(rf.lrn)

#set parameter space

params <- makeParamSet(makeIntegerParam("mtry",lower = 2,upper = 10),makeIntegerParam("nodesize",lower = 10,upper = 50))

#set validation strategy

rdesc <- makeResampleDesc("CV",iters=5L)

#set optimization technique

ctrl <- makeTuneControlRandom(maxit = 5L)

#start tuning

tune <- tuneParams(learner = rf.lrn, task = traintask, resampling = rdesc, measures = list(acc), par.set = params, control = ctrl, show.info = T)

#[Tune] Result: mtry=2 : nodesize=23 : acc.test.mean=0.858