Project8

Mike

12/26/2016

Executive summary

This project is to create a machine learning model to predict the manner the participants did the exercise. I used SEMMA approach (Sample, Explore, Modify, Model and Assess) to build this predictive model. Explanation of variables are not availabe in original report so I would have to guest their meanings and tried to apply common sense to figure out the real meaning of each variable.

Build a predictive model

Sample

The first step to build a model is to sample the dataset, the original training dataset is divided into 2 new training/testing sets at the ratio of 3:1. Normally, doing the sample before exploration stage help limit the size of data. Howver, I opted for doing this split later as the dataset is not so big and the data processsing codes could run on the dataset once.

```
#Read the dataset
training<-read.csv("/Users/gamelord/Documents/OneDrive/Coursera.org/8. Practical ML/P
roject/pml-training.csv",na.strings = "NA",sep=",",header = TRUE)
testing<-read.csv("/Users/gamelord/Documents/OneDrive/Coursera.org/8. Practical ML/Pr
oject/pml-testing.csv",na.strings = "NA",sep=",",header = TRUE)
```

Explore

According to the report, there are 4 accelerometers in respectively the waist (1), left thigh (2), right ankle (3), and right arm (4). There are some variables in training dataset just derivatives of original data (such as kurtosis, skewness, max/min, var etc.). There are a lot of N/A so these variables will be removed.

```
#Quick look at the variables
str(training); summary(training)
#remove all unnecessary variables which are almost N/A
training1 <- training[colSums(is.na(training)) == 0]
varlist<- data.frame(names(training1))
# new_window variable isn't necessary because it doesn't separate any 'classe' from t
he other because:
training1[training1$new_window=="yes",93]
# remove "skewness", "kurtosis", max/min, amplitude variables
names(varlist)<-"varList"
varlist$ID <- "1"
for (i in 1:nrow(varlist)) {
  if (grepl("skewness",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
  if (grepl("kurtosis",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
  if (grepl("min",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
  if (grepl("max",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
  if (grepl("amplitude",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
  if (grepl("new_window",varlist$varList[i])==TRUE) {varlist$ID[i]=0 }
}
varlist$ID[1:7]<-0
#remove variable 1st to 8th, especially num_window as it's a ridiculus strong indicat
or
varlistID <- which(varlist$ID==1)
training2 <- training1[,varlistID]
summary(training2)
```

Modify

I opted for using original variables without and standardization or normalization to build my model to make sure the coefficients are on the same scale. The correlation table of remaining 52 variables is helpful to determine the relationship between those variables and there are some high correlation between some variables. For instance, roll_belt has high correlation with yaw_bell, total_accel_belt. I opted to remove correlation over 90% to avoid unnecessary duplicated impact.

```
#Correlation matrix
corrtable<-cor(training2[,c(1:52)])
library(caret)
#remove high correlated variables
removal <- findCorrelation(corrtable, cutoff = .90, verbose = TRUE)
training3<-training2[,-removal]
#Create new training and testing dataset
inTrain <- createDataPartition(training3$classe, p = 3/4)[[1]]
training4 <- training3[ inTrain,]
testing4 <- training3[-inTrain,]
allvar<-data.frame("varList"=names(training4)) #All variables in model
dim.training4<-dim(training4)
```

There are 1.471710^{4}, 45 independent variables and 1 target variable left to build the model.

```
#keep necessary variables for testing dataset of 20 observations
vartestlist<-data.frame(names(testing))
names(vartestlist)<-"varList"
vartestlist$ID <- "0"
for (i in 1:nrow(vartestlist)) {
  for (j in 1:nrow(allvar)) {
    if (grepl(vartestlist[i,1],allvar[j,1])==TRUE) {vartestlist[i,2]=1}
  }
}
vartestlistID <- which(vartestlist$ID==1)
testing20 <- testing[,vartestlistID]
```

Model

To build the best model, several packages were used. A gradient boosting model (caret package & h2o package) and a random forest model (randomForest & h2o package)

```
set.seed(2016)
gbm.time<-system.time(gbmmodel<-train(classe~.,data=training4,method='gbm'))
gbmtrain<- predict(gbmmodel,newdata = testing4,method='response')
gbm.accuracy<-confusionMatrix(testing4$classe,gbmtrain)$overall[1]
gbmImp <- varImp(gbmmodel,scale=FALSE)
```

It took forever to run the random forest model with caret package on my MacBook Air 2014 8GB RAM, so I decided try it on ParallelForest. However, due to a compatibility error, I switched to H20 package

```
library(h2o)
localH2O <- h2o.init(nthreads = -1)
#check status
h2o.init()
#data to h2o cluster
train.h2o<-as.h2o(training4)
test.h2o<-as.h2o(testing4)
test20.h2o <- as.h2o(testing20)
#assign ID for variables
y.dep <- 46 #column ID of classe
x.indep <- c(1:45) #all other variables
#Random Forest
rfh2o.time<-system.time(rforest<-h2o.randomForest(y=y.dep,x=x.indep,training_frame=tr
ain.h2o,ntrees=1000,mtries=3,max_depth=4,seed=2016))
h2o.varimp(rforest) #variables' importance
#make prediction with training dataset
rftest<-as.data.frame(h2o.predict(rforest,test.h2o))
rfh2o.accuracy<-confusionMatrix(testing4$classe,rftest$predict)$overall[1]
#Gradient Boosting
gbmh2o.time<-system.time(gbm.model <- h2o.gbm(y=y.dep,x=x.indep,training_frame = trai
n.h2o,ntrees=1000,max_depth=4,learn_rate=0.01,seed=2016))
predict.gbm <- as.data.frame(h2o.predict(gbm.model, test.h2o))
gbmh2o.accuracy<-confusionMatrix(testing4$classe,predict.gbm$predict)$overall[1]
h2o.varimp(gbm.model)

#RandomForest package
library(randomForest)
rf.time<-system.time(rf.training<-randomForest(classe~.,data=training4,ntree=100, imp
ortance=TRUE))
rf.pred<-predict(rf.training,newdata=testing4,type="class")
rf.accuracy<-confusionMatrix(testing4$classe,rf.pred)$overall[1]
```

Assess

To find the best model, I opted to use the Accuracy rate on Testing dataset.

```
#Summary table of system time and accuracy
summary.t <- as.data.frame(rbind("caret","h2o","randomForest","h2o"))
names(summary.t)<-"Package"
summary.t<-cbind(summary.t,"Model"=as.data.frame(rbind("GBM","GBM","Random Forest","R
andom Forest")),Accuracy=as.data.frame(rbind(gbm.accuracy,gbmh2o.accuracy,rf.accuracy
,rfh2o.accuracy)),"System Time"=as.data.frame(rbind(gbm.time,gbmh2o.time,rf.time,rfh2
o.time)))
#Predict results of 20 observations
h2o.init()
gbmh2o.test20 <- as.data.frame(h2o.predict(gbm.model, test20.h2o))
rfh2o.test20 <-as.data.frame(h2o.predict(rforest,test20.h2o))
gbm.test20 <- predict(gbmmodel,newdata = testing20,method='response')
rf.test20 <-predict(rf.training,newdata=testing20)
```

The table below summary time need to run each model, its respective accuracy on testing dataset and system running time

```
##                       Package            V1  Accuracy System Time.user.self
## gbm.accuracy            caret           GBM 0.9651305              1762.770
## gbmh2o.accuracy           h2o           GBM 0.9891925                 1.830
## rf.accuracy      randomForest Random Forest 0.9949021                26.340
## rfh2o.accuracy            h2o Random Forest 0.8174959                 0.618
##                  System Time.sys.self System Time.elapsed
## gbm.accuracy                   23.027            1813.868
## gbmh2o.accuracy                 0.125             163.379
## rf.accuracy                     0.346              27.090
## rfh2o.accuracy                  0.066              34.087
##                  System Time.user.child System Time.sys.child
## gbm.accuracy                          0                     0
## gbmh2o.accuracy                       0                     0
## rf.accuracy                           0                     0
## rfh2o.accuracy                        0                     0
```

randomForest package performs faster than caret package on GBM but the two all lagged behind h2o package in term of speed. However, in term of accuracy rate, randomForest package outperforms h2o package random forest and it’s the best model so far. h2o package gbm model beats caret package gbm in both system running time and accuracy rate. I intended build an ensemble model but as the accuracy of h2o random forest is pretty good. So I opted for random forest model from randomForest package to predict the outcome of 20 observations

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```