Human Activity Recognition (Course Project for Practical Machine Learning)

Author: Yulong Deng, Date:2015-12-24

Abstract:

In this report, we have tried to make human activity recognition based on the Weight Lifting Exercise Dataset. First, the dataset has been cleaned and appropriate variables have been selected as predictors during the pre-processing work. Next, three classic machine learning models have been applied and evaluated, including classification tree (RPART), linear discriminant analysis (LDA) and random forests (RF). Then, the RF model, with the highest accuracy, has been chosen as the prediction model. Based on k-fold cross validation, the assessment of the out of sample error (OSE) of RF model has been made. Last, the best fitted RF model is used to predict the classes of 20 different test cases and the results are given.

1. Pre-Processing for Datasets.

1.1 Load the Datasets.

Note: Three types of items: “NA”, “#DIV/0!”, “” are marked as NA in datasets

```r
library(caret); library(randomForest); library(rpart); library(MASS)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```r
mydata <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
mydata1 <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
```

1.2 Datasets Cleaning.

Notes: the columns in which the number of items marked as NA is greater than 19000 are removed from datasets.

```r
k <- apply(mydata,2,function(x){sum(is.na(x))})
mydata_cleaned <- subset(mydata, select = names(k[k==0]))
k1 <- apply(mydata1,2,function(x){sum(is.na(x))})
mydata1_cleaned <- subset(mydata1, select = names(k1[k1==0]))
```

1.3 Choosing the Predictors and Making the Datasets Ready for Training & Testing.

Notes: Variables: “X”,“user_name”,“raw_timestamp_part_1”,“raw_timestamp_part_2”,“cvtd_timestamp”,“new_window” “num_window” are removed from the predictors because of they are irrelevant to prediction of classes of activity. So We get two final datasets: training_set and testing_set, in which training_set is used for model training and validation, testing_set is set aside for final model testing.

```r
m <- names(mydata_cleaned)[which(!(names(mydata_cleaned) %in%
c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","new_window","nu
m_window")))]
training_set <- subset(mydata_cleaned, select = m)
m1 <- names(mydata1_cleaned)[which(!(names(mydata1_cleaned) %in%
c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","new_window","nu
m_window")))]
testing_set <- subset(mydata1_cleaned, select = m1)
```

2. Making Comparison for Three Machine Learning Models.

Here we fit three classic machine learning models (RPART,LDA and RF), get their accuracy and training time respectively.

2.1 Data Splitting, Using Validation Set Approach.

Notes: The training_set is split to two parts, where 70% of samples are used for training and 30% of samples are used for validation.

```
set.seed(1234)
inTrain <- createDataPartition(training_set$classe, p=0.7, list=FALSE)
training <- training_set[inTrain,]
validation <- training_set[-inTrain,]
```

2.2 Classification Tree (RPART) Model Fitting.

```
modFit1<- train(classe ~ ., method="rpart",data = training)
pred1 <- predict(modFit1, validation)
k1<-confusionMatrix(pred1, validation$classe)
a1 <- as.numeric(k1$overall[1])
t1 <- as.numeric(modFit1$times$everything[1])
```

2.3 Linear Discriminant Analysis (LDA) Model Fitting.

```
modFit2<- train(classe ~ ., method="lda",data = training)
pred2 <- predict(modFit2, validation)
k2<-confusionMatrix(pred2, validation$classe)
a2 <- as.numeric(k2$overall[1])
t2 <- as.numeric(modFit2$times$everything[1])
```

2.4 Random Forests (RF) Model Fitting.

```
t <- system.time({modFit3<- randomForest(classe ~ ., data = training, proximity = TRUE)})
pred3 <- predict(modFit3, validation)
k3<-confusionMatrix(pred3, validation$classe)
a3 <- as.numeric(k3$overall[1])
t3 <- as.numeric(t[1])
```

2.5 Conclusion.

It is illustrated in Fig.1, the accuracy of RF model on validation set is the highest one among three models, however, it also needs quite a long time for training as Fig.2 shows.

```
a_all <- c(a1,a2,a3)
t_all <- c(t1,t2,t3)
par(mfrow = c(1, 2), mar = c(4, 4, 2, 4), oma = c(0, 0, 0, 0))

barplot(a_all, names.arg = c("RPART", "LDA", "RF"), xlab ="Model", ylab ="Accuracy", col="blue",
main = "Fig.1 Accuracy of Models", cex.main = 1.2)
barplot(t_all, names.arg = c("RPART", "LDA", "RF"), xlab ="Model", ylab ="Training Time (second)",
col="blue", main = "Fig.2 Training Time of Models", cex.main = 1.2)
```

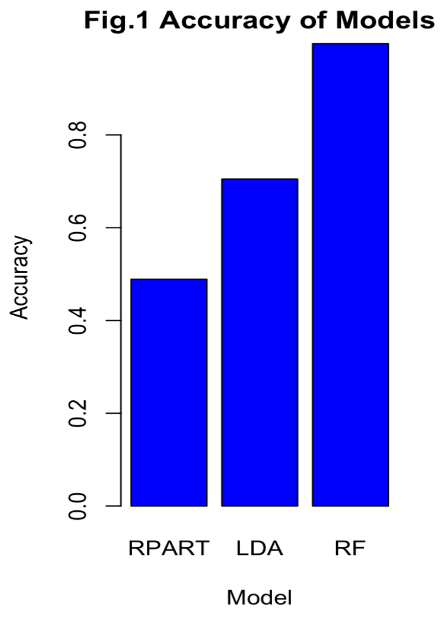


Fig.1 Accuracy of Models

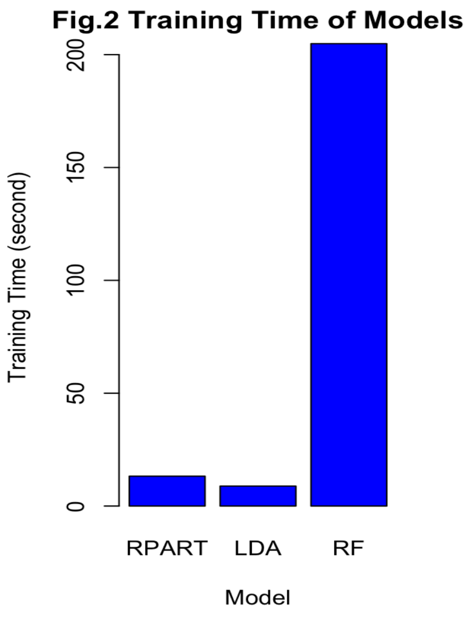


Fig.2 Training Time of Models

3. Fitting Random Forests Models by Cross Validation and Applying it in Prediction.

We give first priority to accuracy of the models in our work, so the RF model is chosen and trained on datasets by cross validation, then the best fitted RF model is selected for prediction on test dataset finally.

3.1 Data Splitting with K-Fold Cross Validation.

Notes: In k-fold cv, k = 5. The training_set is split to five folds, one fold is used for validation and the rest of folds are used for training.

```
set.seed(5678)
set_index <- createFolds(training_set$classe, k = 5, list = TRUE, returnTrain = FALSE)
```

3.2 RF Model Fitting and Prediction.

Notes: RF model will be trained and validated based on datasets by K-Fold CV and applied on the testing_set for prediction. This process is repeated five times, the accuracy value of fitted model on validation set and its prediction result on final test dataset in each time are stored for choosing the best model and prediction.

```
a_five <- NULL
pred_testingset_five <- list(NULL, NULL, NULL, NULL, NULL)
for(i in 1:5){
     all_index <- c(1:19622)
     validation <- training_set[set_index[[i]],]
     training <- training_set[setdiff(all_index, set_index[[i]]),]
     set.seed(6789)
     modFit <- randomForest(classe ~ ., data = training, proximity = TRUE)
     pred <- predict(modFit, validation)
     k<-confusionMatrix(pred,validation$classe)
     a_five <- c(a_five, as.numeric(k$overall[1]))
     pred_testingset <- predict(modFit, testing_set)
     pred_testingset_five[[i]] <- pred_testingset
}
```

3.3 Out of Sample Error (OSE) on Validation Dataset of RF Model in K-Fold CV.

In our experiment, we use accuracy of fitted model on validation dataset to inspect the out of sample error (OSE). The different OSE values in five cross validation and the average of OSE are given as Fig.3. The lowest out of sample error rate and their mean based on validation dataset for the fitted RF model are given as :

```
min(1 - a_five)
```

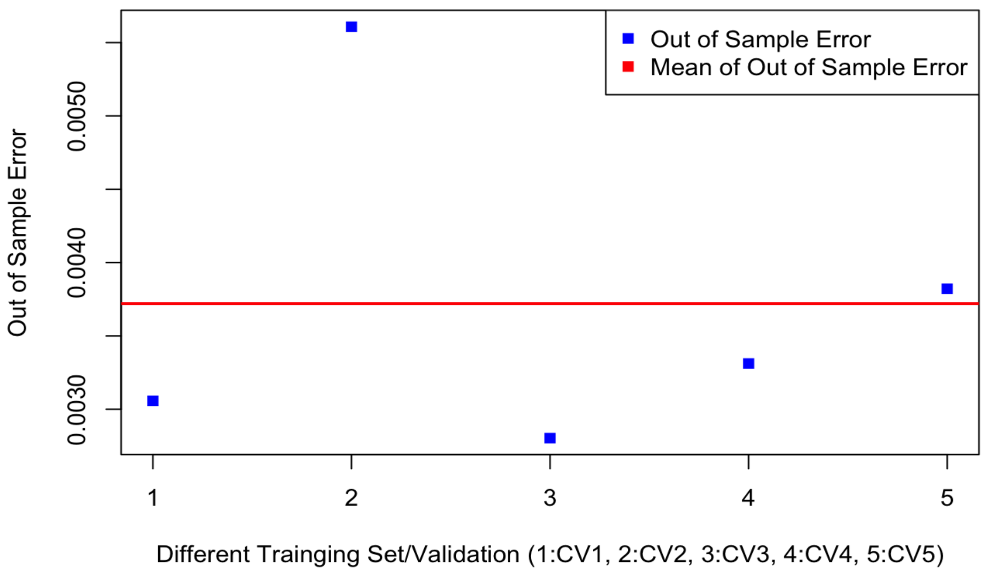
```
## [1] 0.002803262
```

```
mean(1 - a_five)
```

```
## [1] 0.00372046
```

```
plot(x = c(1 ,2 ,3 ,4 ,5), y = 1 - a_five, type="p", pch = 15,  xlab ="Different Trainging
Set/Validation (1:CV1, 2:CV2, 3:CV3, 4:CV4, 5:CV5)", ylab ="Out of Sample Error", col="blue",
main = "Fig.3 Out of Sample Error in K-Fold CV (k=5) for RF model ", cex.main = 1.2)
abline(h = mean(1 - a_five), lwd = 2, col = "red")
legend("topright", pch = 15, col = c("blue","red"), legend = c( "Out of Sample Error","Mean of
Out of Sample Error" ))
```

Fig.3 Out of Sample Error in K-Fold CV (k=5) for RF model



3.4 Prediction Result on Test Dataset and the Out Sample Error.

Based on the five accuracy values of fitted RF model stored before, we choose the fitted model with lowest OSE and its prediction result on final testing_set. So the predictions for activity classes of the 20 test cases is:

```
model_index <- which(1 - a_five == min(1 - a_five))
pred_testingset_five[[model_index]]
```

```
##  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
```

Before applying the fitted model to the final testing_set, the true OSE is unknown, we expect it will be lower than the OSE value gotten on the validation dataset. Then we submit the prediction result in the course project site and hit all of the problems correctly (from program id 1 to 20), so we can give the true out of sample error of our model on the final test dataset, as:

```
answers=as.factor(c("B","A","B","A","A","E","D","B","A","A","B","C","B","A","E","E","A",
"B","B","B"))
k1 <- confusionMatrix(pred_testingset_five[[model_index]],answers)
1 - as.numeric(k1$overall[1])
```

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

Beyond expectations, the true OSE of our model is equal to 0, it shows that the Random Forests model trained by K-Fold CV may be a reasonable choice in the Human Activity Recognition project, if we overlook its shortcoming: very time-consuming.