PML_Writeup

Zhongyu

Sunday, April 26, 2015

Introduction

Human Activity Recognition has becoming a hot study field in the past few years. One of the widely used type of approaches for accurately classifying human activity types is machine learning algorithms. This project utilizes machine learning algorithms to build human activity classifiers based on the provided Human Activity Recognition datset.

Explortary Analysis

Split Dataset

Before exploring the basic characteristics of variables, for the purpose of estimating out of sample error after fitting the models, the dataset is split into two smaller datasets: 70% of the original dataset goes into the training set, 30% of the original dataset goes into the testing/cross validation set. Explortary analysis is only performed on the training set.

```
training <- read.csv('pml-training.csv',na.string=c('NA',''));
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
set.seed(1257986)
inTrain <- createDataPartition(y=training$classe,p=0.7,list=FALSE)
trainset <- training[inTrain,]
testset <- training[-inTrain,]
dim(trainset)
```

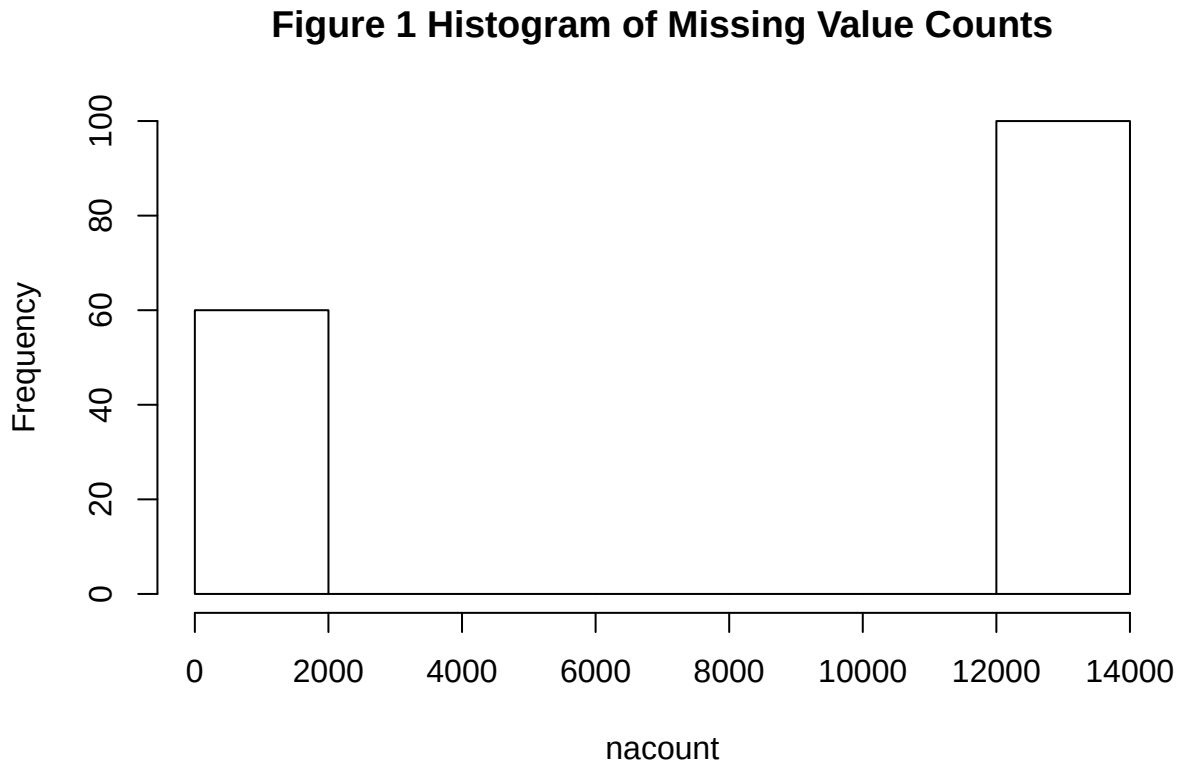
```
## [1] 13737   160
```

Missing Values Investigation

As it is printed out, there are 160 variables and 13737 observations in the training dataset. A quick glance at the training set reveals that for some variables there are a large amount of missing data (NA). Since most of the classificatin/prediction models are not built to deal with missing values, this matter of the dataset needs to be further investigated and dealt with.

A function is written to count the number of missing values of each variables, and a histgram of the obtained missing value counts is plotted in Figure 1.

```
nancount <- function(data){
    varnames  <- colnames(training)
    navar <- vector(mode='numeric', length=length(varnames))
    ind <- 1:length(varnames)
    for (i in ind){
        navar[i]=sum(is.na(data[[varnames[i]]]))
    }
    navar}
nacount <- nancount(trainset)
hist(nacount, main='Figure 1 Histogram of Missing Value Counts')
```




```
selectVarID <- nacount==0
selectTrain <- trainset[,which(selectVarID)]
```

As the plot shown, 100 of all the variables contains 13737 missing observations, and the rest 60 variables have no missing values at all. Therefore only the 60 variables which have no missing values are considered as predictor candidates.

Feature Selection

A further look at the remaining variables indicates that the first five variables are storing information about identities and time stamp of each participant. This information is irrelavant to the outcome variable - classe - human activity types. Therefore the first five variables are not used to be predicting features.

```
str(selectTrain[,1:5]);
```

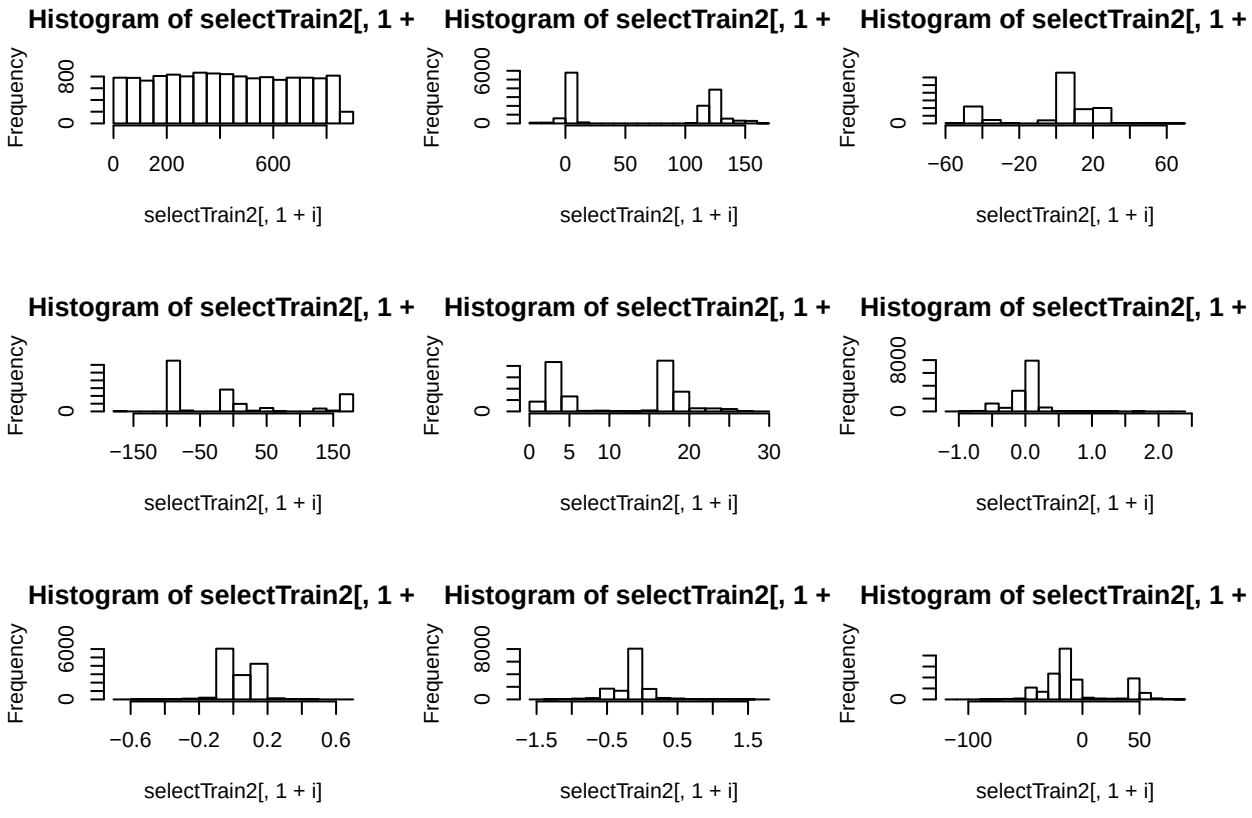
```
## 'data.frame':    13737 obs. of  5 variables:
##  $ X                    : int  2 3 5 6 7 12 14 15 16 17 ...
##  $ user_name            : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084232 1323084232 1323084232 1323084232 1323
##  $ raw_timestamp_part_2: int  808298 820366 196328 304277 368296 528316 576390 604281 644302 692324
##  $ cvtd_timestamp       : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
```

```
nonPredictorID <- c(1,2,3,4,5)
selectTrain2 <- selectTrain[,-nonPredictorID]
```

Data Preprocessing - Standardizing

For the 53 numerical variables of the remaining 55 variables (54 feature predictors and 1 outcome variable), the distribution and scale of each variable is investigated by plotting histgorams. However due to limited space, only histograms of the first 9 variables are shown here.

```
par(mfrow=c(3,3));
for (i in seq(1,9)){
    hist(selectTrain2[,1+i])
}
```



The various histograms illustrate that different variables have their distinct distribtuion characteristics, moreover the scales of different variables vary a lot as well. Therefore the training set is standardized

before fitting models, and the testing/cross validation set is standardized using the same preprocessing object/function.

```
factorID <- c(1,55)
# Preprocess training set
preObj <- preProcess(selectTrain2[,-factorID],method=c('center','scale'))
selectTrainS <- predict(preObj, selectTrain2[,-factorID])
selectTrainS <- cbind(selectTrain2[,1], selectTrainS, selectTrain2[,55])
colnames(selectTrainS)[1] <- 'new_window'
colnames(selectTrainS)[55] <- 'classe'
# Preprocess testing set
selectTest <- testset[,which(selectVarID)]
selectTest2 <- selectTest[,-nonPredictorID]
selectTestS <- predict(preObj, selectTest2[,-factorID])
selectTestS <- cbind(selectTest2[,1], selectTestS, selectTest2[,55])
colnames(selectTestS)[1] <- 'new_window'
colnames(selectTestS)[55] <- 'classe'
```

Building Machine Learning MOdels for Classification

Two different machine learning algorithms are used to train the model: Boosted Logistic Regression and Random Forest. Both of them are commonly used classification algorithms. The final selected model is decided based on trained model perforamce (sepcifically based on estiamted out of sample error/model performance on testing/cross validation set).

Boosted Logistic Regression

Logistic regression method is chosen under the considering the natural of the dataset. Among the 54 predicting features, only one is categorical variable, and the rest 53 are numerical variables; whereas the outcome variable is categorical variables. Logistical regression model is commonly used for utilizing continuous inputs and providing categorical outputs.Comibing with boosting, commonly boosted logsitic regression models provide accurate classification results.

```
set.seed(338007)
modFit1 <- train(classe~.,data=selectTrainS,method='LogitBoost',metric='Accuracy')
```

```
## Loading required package: caTools
```

Random Forest

Random forest is another commonly used classifier with very good performance in terms of classification accuracy. However it is a very computational expensive algorithm. It is capable of fitting very complicated non-linear functions between explainatory variables and response variable, meanwhile it is difficult to interpretate the fitted model, and it needs to be used cautiously to prevend overfitting problem.

```
set.seed(25896)
modFit2 <- train(classe~.,data=selectTrainS,method='rf',metric='Accuracy')
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Model Performances

Boosted Logistic Regression

The first fitted model is employed on the testing/cross validation set to predict the values of classe. And the accuracy (correctly predicted outcome) is computed.

```
modFit1
```

```
## Boosted Logistic Regression
##
## 13737 samples
##    54 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   nIter  Accuracy   Kappa      Accuracy SD  Kappa SD
##   11     0.8423351  0.7955569  0.015092917  0.019857606
##   21     0.9009139  0.8735326  0.007320040  0.009228844
##   31     0.9284061  0.9089008  0.007153868  0.009163720
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was nIter = 31.
```

```
predTest1 <- predict(modFit1,selectTestS)
predTestRight1 <- selectTestS$classe==predTest1
print(paste0('The accuracy of fitted Boosted Logistical Regression Model on testing/cross validation se
             as.character(sum(predTestRight1,na.rm=TRUE)/length(predTest1))))
```

```
## [1] "The accuracy of fitted Boosted Logistical Regression Model on testing/cross validation set is:0
```

As it is printed out, the accuracy of the best fitted boosted logistic regression model is about 92.84%, however the accuracy on testing/cross validation set is only about 83.5%. Therefore the estiamted out of sample error is 1-(accuracy on cross validation set), which is about 16.5%.

Random Forest

The second fitted model is employed on the testing/cross validation set to predict the values of classe. And the accuracy (correctly predicted outcome) is computed.

```
modFit2
```

```
## Random Forest
##
## 13737 samples
##    54 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9919708  0.9898443  0.0015936449  0.002017173
##   28    0.9952688  0.9940163  0.0009147898  0.001156651
##   54    0.9888089  0.9858456  0.0028555878  0.003609319
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 28.
```

```
predTest2 <- predict(modFit2,selectTestS)
predTestRight2 <- selectTestS$classe==predTest2
print(paste0('The accuracy of fitted Random Forest Model on testing/cross validation set is:',
             as.character(sum(predTestRight2)/length(predTest2))))
```

```
## [1] "The accuracy of fitted Random Forest Model on testing/cross validation set is:0.997451146983857
```

As it is printed out, the accuracy of the best fitted boosted logistic regression model is about 99.88%, however the accuracy on testing/cross validation set is about 99.75%. Therefore the estiamted out of sample error is 1-(accuracy on cross validation set), which is about 0.25%.

Summary

According to the comparison of model performances between boosted logistic regression model and random forest model, the constructed random forest model is selected for classifying human activity recoginition on this dataset, since it provides higher accuracy performances in both the training dataset and the testing/cross validation dataset. (* Note: the code and result for 20 test cases are not included in this file)