Practical Machine Learning Project Assigment

Karishma Agarwal

Feb 18, 2018

Synopsis

The document examines data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which these participants exercise. The document describes how the model was built, how was the cross validation used, how was accuracy determined and why some choices were made. Finally, the trained model was used to predict 20 different test cases.

1. Read .csv files required for analysis

```
data_training <- read.csv("Input\ data/pml-training.csv")
data_testing <- read.csv("Input\ data/pml-testing.csv")
```

2. Transform training and test data set to include only features with non-NA values from test data set

```
data_testing <- data_testing[ , colSums(is.na(data_testing)) == 0]
data_training_test <- data_training[names(data_training) %in% names(data_testing)]
# Add the outcome variable
data_training_final <- data.frame(cbind(data_training_test, data_training$classe))
# Change the name of the added column to classe
names(data_training_final)[60] <- 'classe'
```

3. Match factor levels between training and test data set

```
# Remove problem Id [60] from the data_testing
data_testing_new <- data_testing[,-60]
# Remove classe [60] from the data_training_final
data_training_final_new <- data_training_final[,-60]
# Resolving different factor levels issue between training and testing set
data_testing_new$cvtd_timestamp <- as.character(data_testing_new$cvtd_timestamp)
data_testing_new$new_window <- as.character(data_testing_new$new_window)
data_training_final_new$cvtd_timestamp <- as.character(data_training_final_new$cvtd_timestamp)
data_training_final_new$new_window <- as.character(data_training_final_new$new_window)
# Adding isTest identifier
data_testing_new$isTest <- rep(1,nrow(data_testing_new))
data_training_final_new$isTest <- rep(0,nrow(data_training_final_new))
fullSet <- rbind(data_testing_new,data_training_final_new)
fullSet$cvtd_timestamp <- as.factor(fullSet$cvtd_timestamp)
fullSet$new_window <- as.factor(fullSet$new_window)
data_testing_result <- fullSet[fullSet$isTest==1,]
data_training_result <- fullSet[fullSet$isTest==0,]
# Removing isTest identifier
data_testing_result <- data_testing_result[,-60]
data_training_result <- data_training_result[,-60]
# Add the outcome variable
data_training_final_result <- data.frame(cbind(data_training_result, data_training_final$classe))
#Change the name of the added column to classe
names(data_training_final_result)[60] <- 'classe'
```

4. Remove unbalanced class issue form training data set

```
# Training data is skewed on class A, with more than 1700 observations in A as compared to other classes
# To undersample,randomly delete 1,700 rows containing class 'A' from data_training_final_result
new_subset<- subset(data_training_final_result, classe =='A')
# Randomly select 1700 rows from new_subset
new_subset_sample <- new_subset[sample(nrow(new_subset),1700),]
# Delete rows from data_training_final_result which have rows from new_subset_sample
data_training_final_result_sample <- data_training_final_result[!(data_training_final_result$X %in% new_subset_sample$X),]
# Remove first column 'X' does not add value to the model as it is just serial number of records
data_training_final_result_sample <- data_training_final_result_sample[,-1]
data_testing_result <- data_testing_result[,-1]
```

5. Install R packages required for data analysis

```
library(caret)
```

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

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

```
library(e1071)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Using parallel package to improve performance of caret :: train()
library(parallel)
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

6. Prepare training and testing set

```
# Set up training run for x / y syntax because model format performs poorly
x <- data_training_final_result_sample[,-59]
y <- data_training_final_result_sample[,59]
```

7. Configure parallel processing

```
set.seed(95014)
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)
```

8. Configure trainControl object

```
# Changing the resampling method from the default of bootstraping to K-fold cross validation to increase performance
fitControl <- trainControl(method = "cv",
                           number = 5,
                           allowParallel = TRUE)
```

9. Apply random forest method

```
modFit_rf <- train(x,y , method="rf", data=data_training_final_result_sample, trControl = fitControl)
```

10. De-register parallel processing cluster-

```
stopCluster(cluster)
registerDoSEQ()
```

11. Check the model accuracy

```
print(modFit_rf)
```

```
## Random Forest
##
## 17922 samples
##    58 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 14337, 14336, 14339, 14338, 14338
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9973776  0.9967181
##   30    0.9989956  0.9987431
##   58    0.9984934  0.9981146
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 30.
```

```
modFit_rf$resample
```

```
##    Accuracy     Kappa Resample
## 1 0.9988842 0.9986037    Fold1
## 2 0.9991627 0.9989522    Fold3
## 3 0.9991634 0.9989531    Fold2
## 4 0.9983259 0.9979049    Fold5
## 5 0.9994420 0.9993017    Fold4
```

```
confusionMatrix.train(modFit_rf)
```

```
## Cross-Validated (5 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 21.6  0.0  0.0  0.0  0.0
##          B  0.0 21.2  0.0  0.0  0.0
##          C  0.0  0.0 19.1  0.0  0.0
##          D  0.0  0.0  0.0 17.9  0.0
##          E  0.0  0.0  0.0  0.0 20.1
##
##  Accuracy (average) : 0.999
```

12. Apply model to testing data set

```
#Predicting on real test set
prediction_final <- predict(modFit_rf, newdata = data_testing_result)
print(prediction_final)
```

```
##  [1] 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
```