Practical Machine Learning Project

Predicting the manner of exercise performance

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Goal

The goal of this project is to predict the manner of how in which they did the exercise.

Data Preprocessing

1. Download dataset provided for this project analysis into the local machine:

- The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

2. Read and clean data as needed

```
setwd("\\Users\\pxc233\\Documents\\Phalkun\\Classes\\Coursera\\Practical Machine Learnin
g\\Project\\Data")

pml_training <- read.csv("pml-training.csv")
pml_testing <- read.csv("pml-testing.csv")
```

Checking train dataset:

```
dim(pml_training)
```

```
## [1] 19622   160
```

```
table(pml_training$classe)
```

```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

To see summary of training dataset:

```
summary(pml_training)
```

Checking test dataset:

```
dim(pml_testing)
```

```
## [1]  20 160
```

To see summary of test dataset:

```
summary(pml_testing)
```

There are total 19622 observations (obs) and 160 variables in the train dataset and 20 observations and 160 variables in the test dataset. Of 19622 obs in the train dataset are classified in to 5 different classifications (‘classe’ variable) including class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes: + Class A (5580 obs): exactly according to the specification, + Class B (3797 obs): throwing the elbows to the front + Class C (3422 obs): lifting the dumbbell only halfway + Class D (3216 obs): lowering the dumbbell only halfway + Class E (3607 obs): throwing the hips to the front

We found that only 406 are complete cases and that some variables have missing values (NA) in the training dataset. Therefore, we cleaned and removed some variables with NA and or some variables that are not important to ‘classe’ variable such as factor variables (X, -timestamp, _window). Finally, there are only 53 variables remaining in the training dataset. We also cleaned and removed some observations with NA, if any, and factor variables (X, -timestamp, _window) from the test dataset and only 53 variables remaining.

Checking variables with complete obs in the training dataset:

```
train_comp_obs <- sum(complete.cases(pml_training))
train_comp_obs # 406 obs
```

```
## [1] 406
```

Checking variables with complete obs in the test dataset:

```
test_comp_obs <- sum(complete.cases(pml_testing))
test_comp_obs # 0 obs
```

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

Removing var or colunm with NA from training dataset:

```
pml_training <- pml_training[, colSums(is.na(pml_training)) == 0]
dim(pml_training) # 19,622 obs. of  93 variables
```

```
## [1] 19622    93
```

names(pml_training) summary(pml_training) # No NA

Removing var that are not important from training dataset:

```
train_remove <- grepl("^X|timestamp|window", names(pml_training))
summary(train_remove) # 6 more var removed
```

```
##    Mode   FALSE    TRUE    NA's
## logical      87       6       0
```

```
pml_training <- pml_training[, !train_remove]
dim(pml_training) # 19622 obs. of  87 variables
```

```
## [1] 19622    87
```

Subset only numeric var in the training dataset:

```
pml_training_final <- pml_training[, sapply(pml_training, is.numeric)]
dim(pml_training_final) # 19622 obs. of  52 variables:
```

```
## [1] 19622    52
```

Since the required ‘classe’ var was removed, we add it back to the final training daset

```
pml_training_final$classe <- pml_training$classe
dim(pml_training_final) # 19622 obs. of  53 variables
```

```
## [1] 19622    53
```

3. Slicing the data

For the purpose of cross validation (CV), we sub-split the training set (final training dataset) into training (70%) and test or validation set (30%). As a result, we got 14,718 obs in the training set and 4,904 obs in the test set.

```
set.seed(32343) # to get the same resampled numbers generated
library(caret)
```

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

Create training set with with 75% of data and validation set

```
inTrain <- createDataPartition(pml_training_final$classe, p=0.75, list=FALSE)
training_data <- pml_training_final[inTrain,]
testing_data <- pml_training_final[-inTrain,]
```

dimension of original training dataset and validation set

```
rbind("original dataset" = dim(training_data),"training set" = dim(testing_data))
```

```
##                   [,1] [,2]
## original dataset 14718   53
## training set      4904   53
```

Data Modeling

We used the test set within training set to do CV, meaning we estimated accuracy of the validation set within the training set and then applied final model in the original test set. Doing so, we would get un-bias measurement of out of sample accuracy. To be more precise in building models, we used 'trainControl' arg of train function of package **caret** and built models on the training set and then assessed performance of the trained model on the validation set.

Out-of-sample error (OOSE):

We calculated OOSE rates using the confusion matrix method of the **Caret** package for the classification model and validation set. We expected that OOSE rates for the validation set should be similar or greater than the calculate out-of-bag (OOB) for the training set. We found that the estimated accuracy is 98.94%, the estimated out-of-sample error (OOSE) is 1.06% and OOB is 0.46%.

```
library(caret)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
# cvControl <- trainControl(## 5-fold CV
#                           method="repeatedcv",
#                           number = 5,
#                           ## repeated 10 times
#                           repeats = 3)

cvControl <- trainControl(method="cv", number=5)
rfmodFit <- train(classe ~ .,data=training_data, method="rf", trControl=cvControl, ntre
e=250)
rfmodFit
```

```
## Random Forest
##
## 14718 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 11775, 11775, 11773, 11775, 11774
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9906239  0.9881388  0.001757647  0.002223266
##   27    0.9919147  0.9897722  0.003070552  0.003884521
##   52    0.9872943  0.9839265  0.004221137  0.005339989
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

Use the fitted model on the test set (testSA)

```
rf_predict <- predict(rfmodFit, testing_data)
```

Use function confusionMatrix to get summary of the results of the model

```
CM_rf <- confusionMatrix(testing_data$classe, rf_predict)
CM_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1393    2    0    0    0
##          B    8  938    3    0    0
##          C    0   15  838    2    0
##          D    0    0    9  793    2
##          E    0    1    3    2  895
##
## Overall Statistics
##
##                Accuracy : 0.9904
##                  95% CI : (0.9873, 0.9929)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9879
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9943   0.9812   0.9824   0.9950   0.9978
## Specificity            0.9994   0.9972   0.9958   0.9973   0.9985
## Pos Pred Value         0.9986   0.9884   0.9801   0.9863   0.9933
## Neg Pred Value         0.9977   0.9954   0.9963   0.9990   0.9995
## Prevalence             0.2857   0.1949   0.1739   0.1625   0.1829
## Detection Rate         0.2841   0.1913   0.1709   0.1617   0.1825
## Detection Prevalence   0.2845   0.1935   0.1743   0.1639   0.1837
## Balanced Accuracy      0.9969   0.9892   0.9891   0.9962   0.9981
```

To calculate Accuracy use postResample function

```
Accuracy <- postResample(rf_predict, testing_data$classe)
Accuracy # Accuracy=0.9893964
```

```
## Accuracy    Kappa
## 0.990416 0.987875
```

To calculate estimated out-of-sample error (OOSE)

```
OOSE <- 1 - Accuracy[1]
OOSE # 0.01060359
```

```
##    Accuracy
## 0.009584013
```

```
# or another way
OOSE_2 <- 1- as.numeric(confusionMatrix(testing_data$classe, rf_predict)$overall[1])
OOSE_2 #  0.01060359
```

```
## [1] 0.009584013
```

To calculate OOB:

```
rfmodFit2 <- randomForest(formula = classe ~ ., data = training_data)
rfmodFit2 # OOB = 0.46%
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training_data)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.45%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4182    2    0    0    1 0.0007168459
## B   10 2834    4    0    0 0.0049157303
## C    0   15 2549    3    0 0.0070120764
## D    0    0   19 2390    3 0.0091210614
## E    0    0    3    6 2697 0.0033259424
```

Prediction for the test dataset

Finally, we used the fitted model to predict for the original test dataset (see Table 1: Results of the prediction).

Removing var or colunm with NA from test dataset:

```
pml_testing <- pml_testing[, colSums(is.na(pml_testing)) == 0]
dim(pml_testing) # 20 obs. of  60 variables
```

```
## [1] 20 60
```

Removing var are not important from test dataset:

```
test_remove <- grepl("^X|timestamp|window", names(pml_testing))
summary(test_remove) # 6 more var removed
```

```
##    Mode   FALSE    TRUE    NA's
## logical      54       6       0
```

```
pml_testing <- pml_testing[, !test_remove]
dim(pml_testing) # 20 obs. of  54 variables
```

```
## [1] 20 54
```

```
pml_testing_final <- pml_testing[, sapply(pml_testing, is.numeric)]
dim(pml_testing_final) # 20 obs and 53 variables
```

```
## [1] 20 53
```

To see all var names:

```
names(pml_testing_final) # 53 variables
```

So there are the same 53 var or features while only 52 in traning dataset

Remove ‘problem_id’ var from the test dataset

```
pml_testing_final <- pml_testing_final[, -53]
dim(pml_testing_final) # 52 var
```

```
## [1] 20 52
```

Now there are the same 52 var or features as in traning dataset

Using the fitted model to Predict for the orginal test dataset

```
predict_test <- predict(rfmodFit, pml_testing_final)
predict_test
```

```
##  [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
```

Visualization

Table 1: Prediction Results for validation data set

Tabulate results

```
table(rf_predict,testing_data$classe)
```

```
##
## rf_predict    A    B    C    D    E
##          A 1393    8    0    0    0
##          B    2  938   15    0    1
##          C    0    3  838    9    3
##          D    0    0    2  793    2
##          E    0    0    0    2  895
```

Figure 1: Decision tree plot

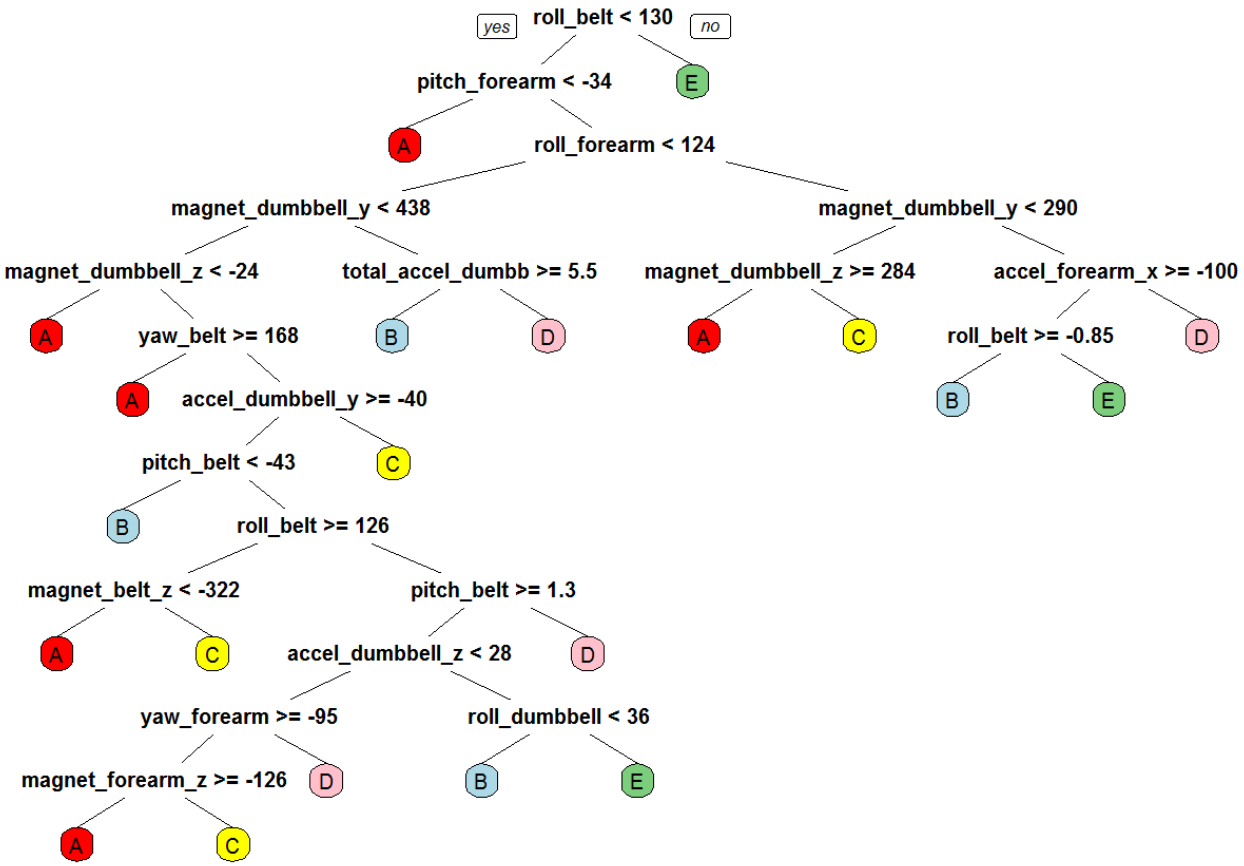
Check overall correct results of prediction by the rf algorithm

```
testing_data$predRight <- rf_predict==testing_data$classe
table(testing_data$predRight)
```

```
##
## FALSE  TRUE
##    47  4857
```

Decision tree plot

```
library(rpart)
library(rpart.plot)
treeplot_model <- rpart(classe ~ ., data=training_data, method="class")
prp(treeplot_model,
    box.col=c("red", "lightblue", "yellow", "pink", "palegreen3")[treeplot_model$frame$yv
al])
```



Creating files for Prediction Assignment Submission

```
answers <- as.vector(predict_test)

pml_write_files = function(x){
        n = length(x)
        for(i in 1:n){
                filename = paste0("problem_id_",i,".txt")
                write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALS
E)
        }
}


pml_write_files(answers)
```

END