Practical ML - Course Project

Himanshu Patil

26th August, 2020

Background :

Human Activity Recognition - HAR - has emerged as a key research area in the last years and is gaining increasing attention by the pervasive computing research community.

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.

Subjects were asked to perform barbell lifts correctly and incorrectly in 5 different ways. 1. Exactly according to the specification **(Class A)** 2. Throwing the elbows to the front **(Class B)** - mistake 3. Lifting the dumbbell only halfway **(Class C)** - mistake 4. Lowering the dumbbell only halfway **(Class D)** - mistake 5. Throwing the hips to the front **(Class E)** - mistake

###Objective :

- In this project,the 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.
- Our training data consists of accelerometer data and a label identifying the quality of the activity the participant was doing. Our testing data consists of accelerometer data without the identifying label. Our goal is to predict the labels for the test set observations.

Algorithm for Prediction:

1.1 Data Preparation

Loading **caret package**

```
library(caret)
```

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

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

Reading **training** and **testing** data

```
train <- read.csv("pml-training.csv")
test <- read.csv("pml-testing.csv")
```

To estimate out-of-sample error. The **training data(train)** is randomly split into two different dataset **trainig set (train1)** and **validation set (train2)**

```
set.seed(10)
inTrain <- createDataPartition(y=train$classe, p=0.6, list = F)
train1 <- train[inTrain,]
train2 <- train[-inTrain,]
```

To reduce the number of features by removing variables with nearly zero variance, variables that are almost always NA, and variables that don’t make intuitive sense for prediction. We can achieve this by examination of dataset **train1** and perform the identical removals on **train2**.

Remove variables with zero variances

```
nzv <- nearZeroVar(train1)
train1 <- train1[, -nzv]
train2 <- train2[, -nzv]
```

Remove variables that are always NA

```
mostlyNA <- sapply(train1, function(x) mean(is.na(x))) > 0.95
train1 <- train1[, mostlyNA==F]
train2 <- train2[, mostlyNA==F]
```

Remove variables that don’t make intuitive sense for prediction X, user_name, raw_timestamp_part_1, raw_timestamp_part_2 and cvtd_timestamp which are first five variables

```
train1 <- train1[, -(1:5)]
train2 <- train2[, -(1:5)]
```

1.2 Data Modelling

Lets start with implementing the **Random Forest Model**, to see if it would have acceptable performance. Fit the model on train1, and instruct the “train” function to use **3-Fold Cross-Validation** to select optimal tuning parameters for the model.

```
# Instruct train to use 3-fold CV to select optimal tuning parameters
fitControl <- trainControl(method="cv", number=3, verboseIter=F)

# Fit model on train1
fit <- train(classe ~ ., data=train1, method="rf", trControl=fitControl)
```

Print final model to see tuning parameters it chose

```
print(fit$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.28%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3348    0    0    0    0 0.000000000
## B    8 2267    4    0    0 0.005265467
## C    0    7 2047    0    0 0.003407984
## D    0    0    6 1924    0 0.003108808
## E    0    1    0    7 2157 0.003695150
```

We can see that model decided to use 500 trees and 27 variable in each split.

1.3 Model Evaluation and Selection

Now, use the fitted model to predict the label **“classe”** on **Validation Set train2**, then show the confusion matrix to compare the predicted versus the actual labels:

```
# Use model to predict classe in validation set (ptrain2)
preds <- predict(fit, newdata = train2)

# Show confusion matrix to get estimate of out-of-sample error
confusionMatrix(factor(train2$classe), factor(preds))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    0    0    0    1
##          B    1 1517    0    0    0
##          C    0    2 1366    0    0
##          D    0    0    7 1279    0
##          E    0    0    0    0 1442
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9975, 0.9993)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9987   0.9949   1.0000   0.9993
## Specificity            0.9998   0.9998   0.9997   0.9989   1.0000
## Pos Pred Value         0.9996   0.9993   0.9985   0.9946   1.0000
## Neg Pred Value         0.9998   0.9997   0.9989   1.0000   0.9998
## Prevalence             0.2845   0.1936   0.1750   0.1630   0.1839
## Detection Rate         0.2843   0.1933   0.1741   0.1630   0.1838
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9997   0.9993   0.9973   0.9995   0.9997
```

The accuracy is 99.8%, thus predicted accuracy for the out-of-sample error is 0.2%.

This is an very good result, so rather than trying additional algorithms, I will use Random Forests to predict on the test set.

1.4 Re-training the Selected Model

Before predicting on the test set, it is important to train the model on the full **training set (train)**, rather than using a model trained on a **reduced training set (train1)**, in order to produce the most accurate predictions. Therefore, I now repeat everything I did above on **train** and **test**:

```
# remove variables with nearly zero variance
nzv <- nearZeroVar(train)
train <- train[, -nzv]
test <- test[, -nzv]

# remove variables that are almost always NA
mostlyNA <- sapply(train, function(x) mean(is.na(x))) > 0.95
train <- train[, mostlyNA==F]
test <- test[, mostlyNA==F]

# remove variables that don't make intuitive sense for prediction (X, user_name, raw_timestamp_part_1, raw_t
imestamp_part_2, cvtd_timestamp), which happen to be the first five variables
train <- train[, -(1:5)]
test <- test[, -(1:5)]

# re-fit model using full training set (ptrain)
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
fit <- train(classe ~ ., data=train, method="rf", trControl=fitControl)
fit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.14%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 5579    0    0    0    1 0.0001792115
## B    6 3787    3    1    0 0.0026336582
## C    0    5 3417    0    0 0.0014611338
## D    0    0    8 3207    1 0.0027985075
## E    0    0    0    3 3604 0.0008317161
```

1.5 Making Test Set Predictions.

Now,use the model fit on ptrain to predict the label for the observations in ptest, and write those predictions to individual files:

```r
# predict on test set
preds <- predict(fit, newdata=test)
#Evaluation
preds
```

```
##  [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
```

1.6 Generating Prediction File

```r
# convert predictions to character vector
preds <- as.character(preds)

# create function to write predictions to files
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=F, row.names=F, col.names=F)
    }
}

# create prediction files to submit
pml_write_files(preds)
```

Reference and Acknowledgement:

- Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz6WFSAJBLN