Machine Learning Course Project

Zelepukin Dmitriy

29 august 2017

The goal of this project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

Load libraries and data sets

```
library(dplyr)
library(rpart)
library(gbm)

# Loading Data

training <- read.csv('pml-training.csv', stringsAsFactors=F, na.strings = c("NA", "", "#DIV/0!"))
testing <- read.csv('pml-testing.csv', stringsAsFactors=F, na.strings = c("NA", "", "#DIV/0!"))

# Data Cleaning

# Remove variables in the training set with too much NAs
goodCol <- colSums(is.na(training)) < 1900
myTraining <- training[ , goodCol][ , ]

# Remove the same columns in the test set
myTesting <- testing[ , goodCol][ , ]

# Remove the first seven columns in both sets
myTraining <- myTraining[ , -(1:7)]
myTesting <- myTesting[ , -(1:7)]
```

Subsetting the training data

In building our model, for a cross validation objective, we subset our training data to a real training set and a test set.

```
# Create inTraining and inTesting
library(caret)
set.seed(4848)
inTrain <- createDataPartition(y = myTraining$classe, p = 0.75, list = FALSE)
inTraining <- myTraining[inTrain, ]
inTesting <- myTraining[-inTrain, ]
```

#Model building with 'randomForest' package:

```
# Train with randomForest
library(randomForest)
```

```
## randomForest 4.6-12
```

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

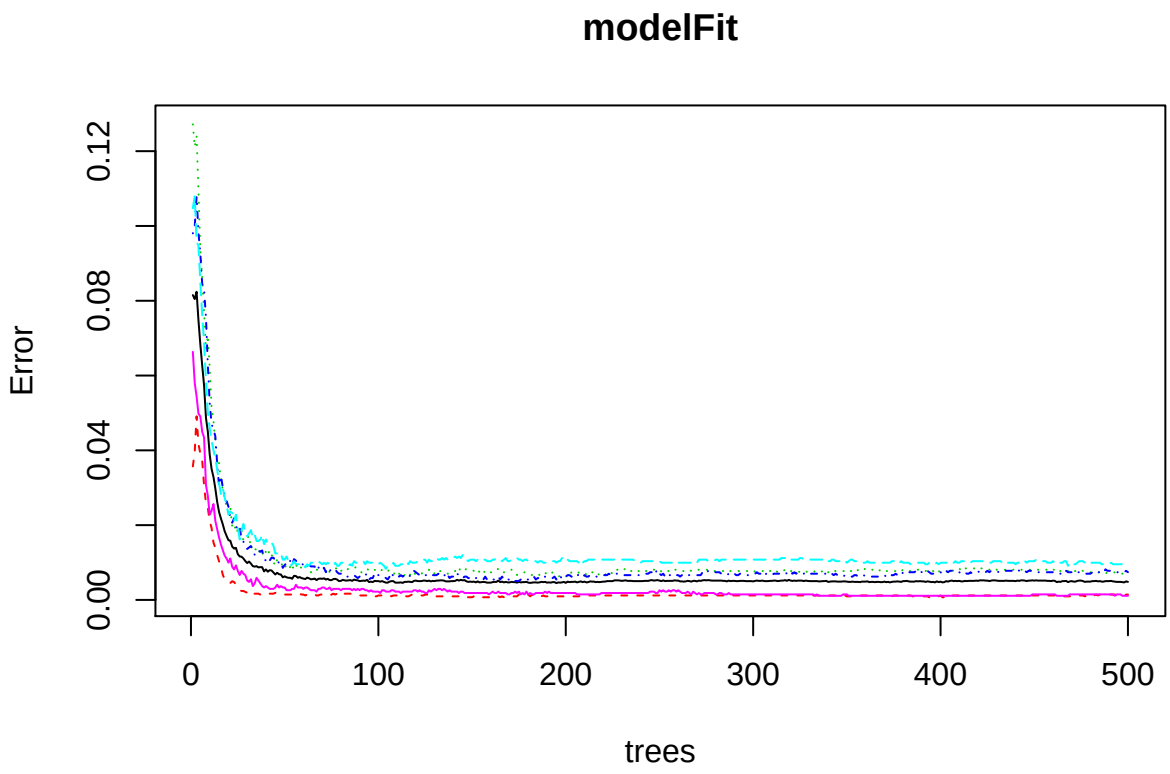
```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
inTraining$classe <- as.factor(inTraining$classe)
set.seed(555)
rfGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                       n.trees = (1:30)*50,
                       shrinkage = 0.1)
modelFit <- randomForest(classe ~., data = inTraining, tuneGrid = rfGrid)
print(modelFit)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = inTraining, tuneGrid = rfGrid)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.48%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4179    4    0    1    1 0.001433692
## B   15 2828    5    0    0 0.007022472
## C    0   17 2548    2    0 0.007401636
## D    0    0   23 2389    0 0.009535655
## E    0    0    1    2 2703 0.001108647
```

```
plot(modelFit)
```



Cross validation

```
# Test "out of sample"
predictions <- predict(modelFit, newdata = inTesting)
confusionMatrix(predictions, inTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    0    0    0    0
##          B    0  949    1    0    0
##          C    1    0  854    1    0
##          D    0    0    0  802    2
##          E    0    0    0    1  899
##
## Overall Statistics
##
##                Accuracy : 0.9988
##                  95% CI : (0.9973, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9985
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   1.0000   0.9988   0.9975   0.9978
## Specificity            1.0000   0.9997   0.9995   0.9995   0.9998
## Pos Pred Value         1.0000   0.9989   0.9977   0.9975   0.9989
## Neg Pred Value         0.9997   1.0000   0.9998   0.9995   0.9995
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1935   0.1741   0.1635   0.1833
## Detection Prevalence   0.2843   0.1937   0.1746   0.1639   0.1835
```

Final validation:

```
# Test validation sample
answers <- predict(modelFit, newdata = myTesting, type = "response")
print(answers)
```

```
##  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
```

Conclusion

Based on the data available, I am able to fit a reasonably sound model with a high degree of accuracy in predicting out of sample observations. One assumption that I used in this work that could be relaxed in future work would be to remove the section of data preparation where I limit features to those that are non-zero in the validation sample. For example, when fitting a model on all training data columns, some features that are all missing in the validation sample do included non-zero items in the training sample and are used in the decision tree models.

The question I'm left with is around the data collection process. Why are there so many features in the validation sample that are missing for all 20 observations, but these have observations in the training sample? Is this just introduced by the Coursera staff for the project to see how students respond? Or is it a genuine aspect of how data is collected from these wearable technologies?

Despite these remaining questions on missing data in the samples, the random forest model with cross-validation produces a surprisingly accurate model that is sufficient for predictive analytics.