Practical Machine Learning - Course Project: Writeup

Federico Calore

15 Dec 2015

Contents

Introduction

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. 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:

- Class A: exactly according to the specification
- Class B: throwing the elbows to the front
- Class C: lifting the dumbbell only halfway
- Class D: lowering the dumbbell only halfway
- Class E: throwing the hips to the front

More information is available from the website here (see the section on the Weight Lifting Exercise Dataset).

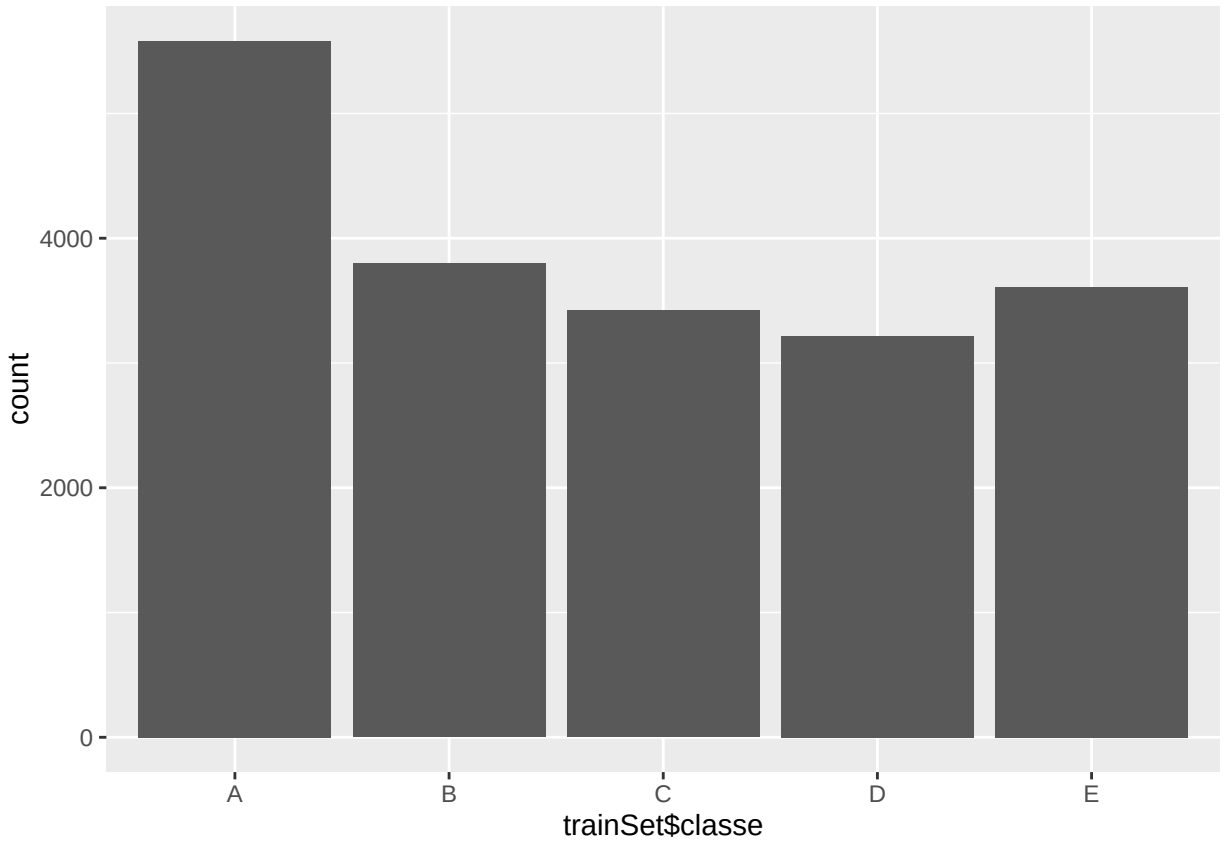
Goal

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with.

Data analysis

As first thing, after downloading the files, we load the dataset in memory. We can check the counts of the outcome levels to get the idea of their distribution.

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



Preprocessing

We preprocess the data to prepare a tidy dataset for modeling:

1. remove variables with **many NAs**
2. identify and remove **unnecessary variables**
3. remove Near **Zero-Variance** predictors
4. **center and scale** all variables
5. split the data into **training and validation**

After a quick exploratory analysis, we found that here variables have either no NAs or empty values, or about 0.98% of them; the latter are summary variables with some statistics for the entire observation window, and happen in combination with the value "yes" for the variable "new_window". Also, performing an analysis for Near Zero-Variance shows that most of these variables only have a handful of significant values, another reason to remove them.

As result of this quick analysis, we will **remove all the (100) colums with summary statistics**, and by this we will get rid of all the NAs in the dataset.

We will also remove the **unnecessary descriptive variables** (serial IDs, timestamps, etc..).

```
rem <- sapply(trainSet, function(x) {sum(is.na(x) | !(x != ""))/length(x)}) # NAs
rem[nearZeroVar(trainSet, saveMetrics = T)[, 4]] <- 1 # flag Near Zero-Variance
rem[1:7] <- 1 # flag timestamp variables
trainSet <- trainSet[, !(rem > 0)] # keep only informative and complete columns
```

We split the training dataset in two parts for **cross-validation**: *70%* will be used for model **training**, and the remaining *30%* will be kept apart as a **validation** dataset, to compare the performance of different algorithms.

Also, we will pre-process the data by **centering and scaling** all the remaining predictor variables.

```
set.seed(19780505) # set seed for reproducibility
inTrain <- createDataPartition(trainSet$classe, p = 0.7, list = FALSE)
trainingS <- trainSet[inTrain, ]
validationS <- trainSet[-inTrain, ]

prePr <- preProcess(trainingS, method = c("center", "scale"))

trainSetPre <- predict(prePr, trainingS)
validSetPre <- predict(prePr, validationS)
```

Modeling

We have readied a smaller dataset with **52 predictors** and 1 categorical outcome.

We build some models with "classe" as the outcome and the other variables left in the dataset as predictors. We want to try different algorithms to compare their performance.

We will leverage the *caret* package to train them by using *2-fold cross-validation* (I would use k = 10 but have no time to compile now!).

```
models <- list()
fitControl <- trainControl(method = "cv",
                           verboseIter = FALSE,
                           number = 2)
# split in different chunks to make caching easier
```

```
models[["lda"]] <-
  train(classe ~ ., data = trainSetPre, method = "lda", trControl = fitControl)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
models[["gbm"]] <-
   train(classe ~ ., data = trainSetPre, method = "gbm",
         trControl = fitControl, verbose = FALSE)
```

```
## Loading required package: gbm
## Loading required package: survival
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
```

```
##
##     cluster
##
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.1
```

```
models[["rpart"]] <-
  train(classe ~ ., data = trainSetPre, method = "rpart", trControl = fitControl)
```

```
## Loading required package: rpart
```

```
models[["ctree"]] <-
  train(classe ~ ., data = trainSetPre, method = "ctree", trControl = fitControl)
```

```
## Loading required package: party
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
##
## Attaching package: 'modeltools'
##
## The following object is masked from 'package:plyr':
##
##     empty
##
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: sandwich
```

```
models[["svm"]] <-
  train(classe ~ ., data = trainSetPre, method = "svmLinear", trControl = fitControl)
```

```
## Loading required package: kernlab
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:modeltools':
##
##     prior
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
models[["rf"]] <-
  train(classe ~ ., data = trainSetPre, method = "rf", trControl = fitControl)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:ggplot2':
##
##     margin
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

We compare the models outcome predicting on the validation dataset and comparing the accuracy of the prediction versus the actual *classe* value. The model that will yield the best overall **accuracy** will be chosen.

```
results <- data.frame(model = names(models), accuracy = NA)
for (i in seq_along(models)) {
  validPred <- predict(models[[i]], validSetPre) # predictions on validation dataset
  results[i, 2]<- postResample(validPred, validSetPre$classe)[1] # accuracy and kappa
}
print(arrange(results, desc(accuracy))) # list models and accuracy in desc order
```

```
##   model  accuracy
## 1    rf 0.9945624
## 2   gbm 0.9646559
## 3 ctree 0.8909091
## 4   svm 0.7811385
## 5   lda 0.6968564
## 6 rpart 0.5029737
```

```
best <- results[results[, 2] == max(results[, 2]),] # best model
```

From the table above, **Random Forest** is the contest winner having achieved an impressive accuracy of 0.9945624 on validation data, which is our estimate for the **out of sample error**.

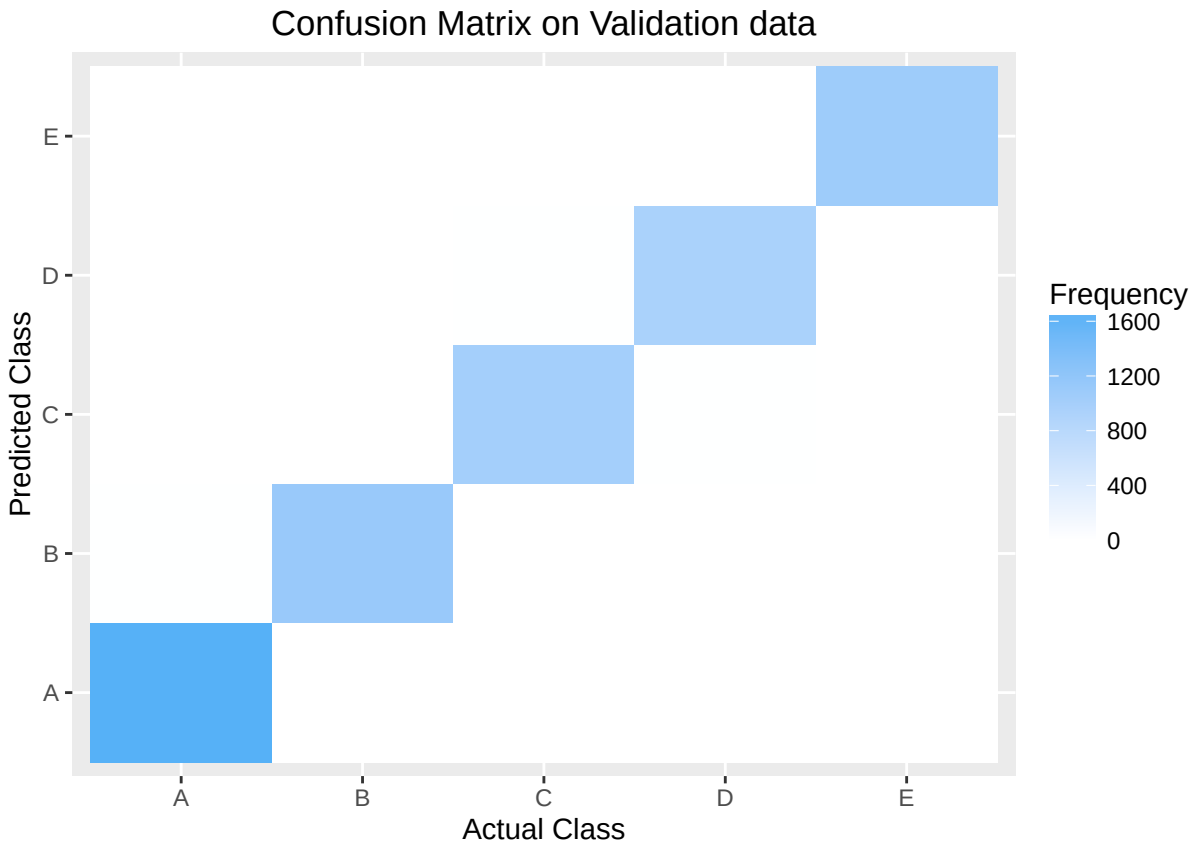
Analysis of the best model

Let's print the other statistics and finally visualize its confusion matrix on the validation dataset.

```
bestModel <- models[[as.numeric(row.names(best))]]
validPred <-
  predict(bestModel, validSetPre) # predictions from the best model
confmx <- confusionMatrix(validPred, validSetPre$classe)
print(confmx) # full confusion matrix and statistics
```

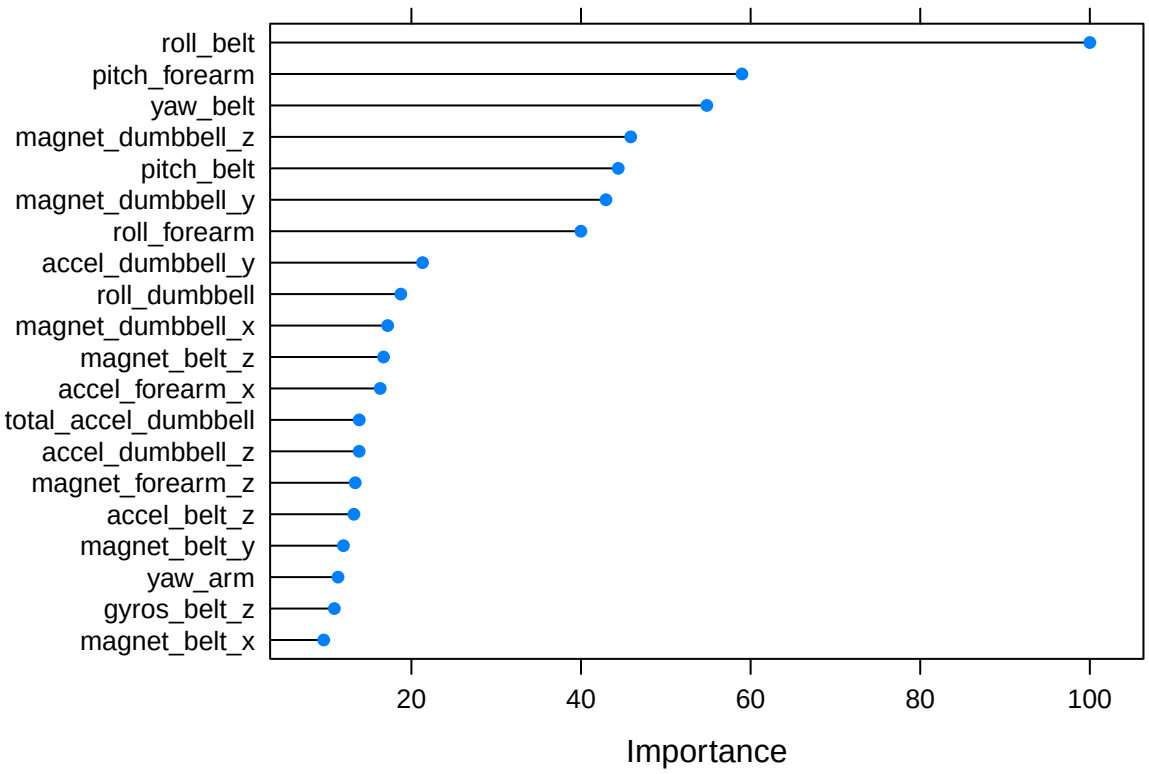
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    7    0    0    0
##          B    1 1131    2    1    0
##          C    1    1 1018    8    1
##          D    0    0    6  955    3
##          E    1    0    0    0 1078
##
## Overall Statistics
##
##                Accuracy : 0.9946
##                  95% CI : (0.9923, 0.9963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9930   0.9922   0.9907   0.9963
## Specificity            0.9983   0.9992   0.9977   0.9982   0.9998
## Pos Pred Value         0.9958   0.9965   0.9893   0.9907   0.9991
## Neg Pred Value         0.9993   0.9983   0.9984   0.9982   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1922   0.1730   0.1623   0.1832
## Detection Prevalence   0.2851   0.1929   0.1749   0.1638   0.1833
## Balanced Accuracy      0.9983   0.9961   0.9950   0.9944   0.9980
```

```
ggplot(as.data.frame(confmx$table), aes(x = Prediction, y = Reference, fill = Freq)) +
  geom_tile() + scale_fill_gradient(low="white") +
  scale_x_discrete(name="Actual Class") + scale_y_discrete(name="Predicted Class") +
  labs(title = "Confusion Matrix on Validation data", fill="Frequency")
```



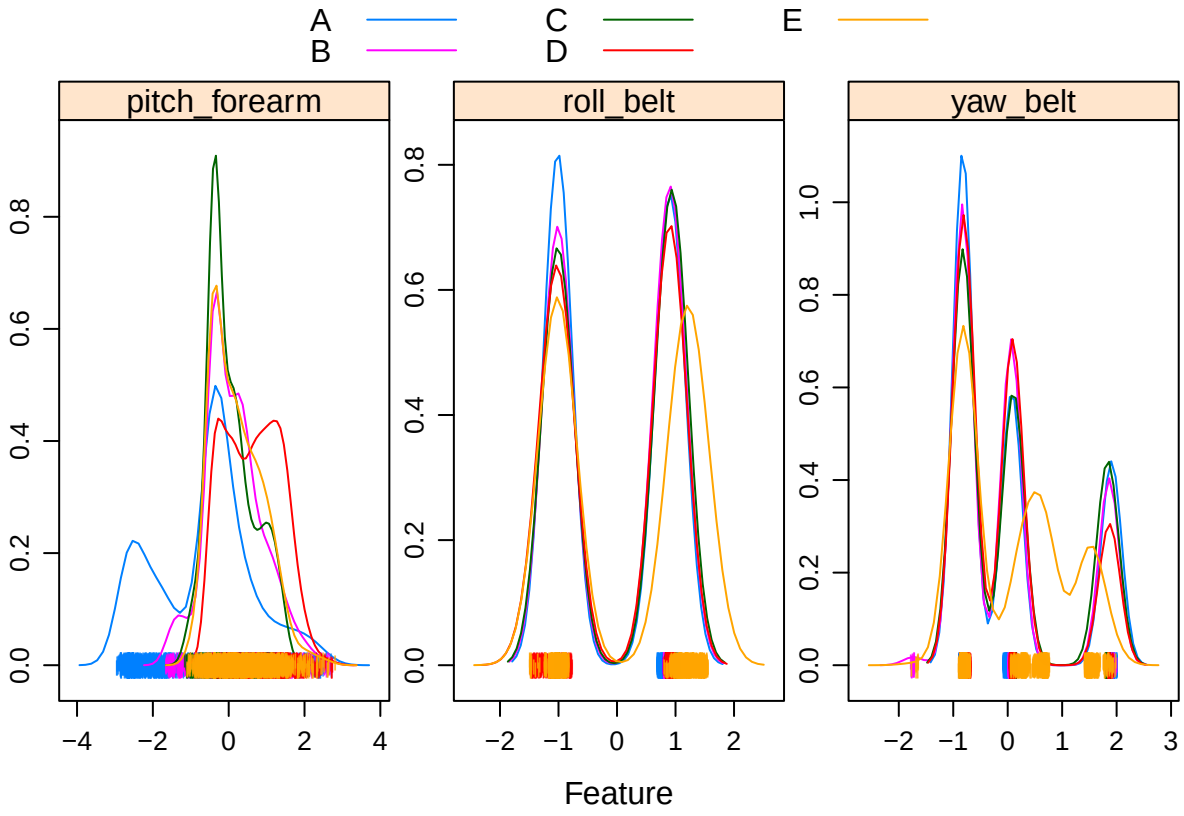
We can extract the order of importance of the variables in the model and visualize their density in relation with the different outcome levels in order to understand their predictive value.

```
plot(varImp(bestModel), top = 20) # variables importance in the final model
```




```
varImp <- varImp(bestModel)$importance
varImp <- varImp[order(varImp[, 1], decreasing = TRUE), , drop = FALSE]

# plot a density curve of the three most important features
featurePlot(x = trainSetPre[, row.names(varImp)[1:3]],
            y = trainSetPre$classe,
            plot = "density",
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            auto.key = list(columns = 3))
```



Final training on all available data

Having selected Random Forest as the best algorithm in this case, we proceed to train it again on the full training dataset, in order to gather the best possible fit based on all the data we have available.

```
# this code won't run to save time when compiling the knitr document
prePr2 <- preProcess(trainSet, method = c("center", "scale"))
trainSetPre2 <- predict(prePr, trainSet)

finalModel <-
  train(classe ~ ., data = trainSetPre2, method = "rf", trControl = fitControl)
```

Appendix

Submission to Coursera

Code to prepare files for the submission to Coursera for the Practical Machine Learning course project.

```
# Please apply the machine learning algorithm you built to each of the 20 test
# cases in the testing data set. For each test case you should submit a text
# file with a single capital letter (A, B, C, D, or E) corresponding to your
# prediction for the corresponding problem in the test data set.

testSet <- read.csv(testDest)
testSet <- testSet[, !(rem > 0)] # remove unnecessary columns
```

```
testSetPre2 <- predict(prePr2, testSet) # apply the same preprocessing to test data
testPred <- predict(finalModel, testSetPre2) # predictions from the tuned model

answers <- as.character(testPred)
pml_write_files <- function(x){
  dir.create("answers")
  for(i in 1:length(x)){
    filename = paste0("./answers/problem_id_",i,".txt")
    write.table(x[i], file=filename,
                quote=FALSE, row.names=FALSE, col.names=FALSE)
  }
}
pml_write_files(answers)
```