Practical Machine Learning: Predictor Project

M.Mueller, November 2014 Johns Hopkins/Coursera Data Science Course: Practical Machine Learning

Synopsis:

A random forest predictive model is capable of predicting exercise type (A-E) with a 99.26% accuracy on a validation (out of sample) data set. The random forest accuracy is the higher than those of other types of models, including rpart, lda, and gbm which provides the second highest accuracy. The random forest model is applied to classifying 20 out of 20 exercise types correctly.

Introduction

This project is an exercise for Practical Machine Learning which pertains to analysis of data related to physical exercise prediction. Please reference for more info: http://groupware.les.inf.puc-rio.br/har https://class.coursera.org/predmachlearn-007/human_grading/view/courses/972608/assessments/4/submissions

Data is collected from accelerometers attached to exercising individuals, and five different motions are performed. In this study, a predictor is formed which can distinguish between the five different motions with a high degree of accuracy.

Read Data; Preliminary Data Exploration:

Training and testing data are downloaded from two separate files, into the working directory, then read into dataframes.

The training dataset is comprised of 19622 rows by 160 columns. Since it is the objective to predict “classe” as a function of the other variables, one can plot each of the variables versus classe. Classe is a factor of the levels A, B, C, D and E which each represents a specific exercise. The data is derived from accelermeters.

Code Chunk 1 Read In Data

```
library(kernel); library(caret); library(rpart); library(randomForest)
```

```
## Error in library(kernel): there is no package called 'kernel'
```

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

```
## Warning: package 'randomForest' was built under R version 3.1.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(e1071); library(MASS); library("parallel")
```

```
## Warning: package 'e1071' was built under R version 3.1.2
```

```
#read data
pml_trn <- read.csv("pml-training.csv")
pml_tst <- read.csv("pml-testing.csv")
```

By examining the raw data (through summary, class, and head/tail) we can see that many of the columns are comprised of mostly NA's. Other columns are non-numeric: these contain blanks, DIV/0 and other entries which are not useful for developing a predictive model.

The classe variable can be plotted (y axis) versus the other 159 variables (x axis) to visualize the separation of boxplots for each of the A, B, C, D and E exercise motion types. We can see that the ranges of distributions vary for each of A-E. One such example is shown below. Thus, we can expect that a machine learning model should be able to classify the classe factors A-E based on the information provided in the other variables.

Code Chunk 2 Explore Data

```
#explore raw data
print("explore raw training data as read from files:")
```

```
## [1] "explore raw training data as read from files:"
```

```
dim(pml_trn)
```

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

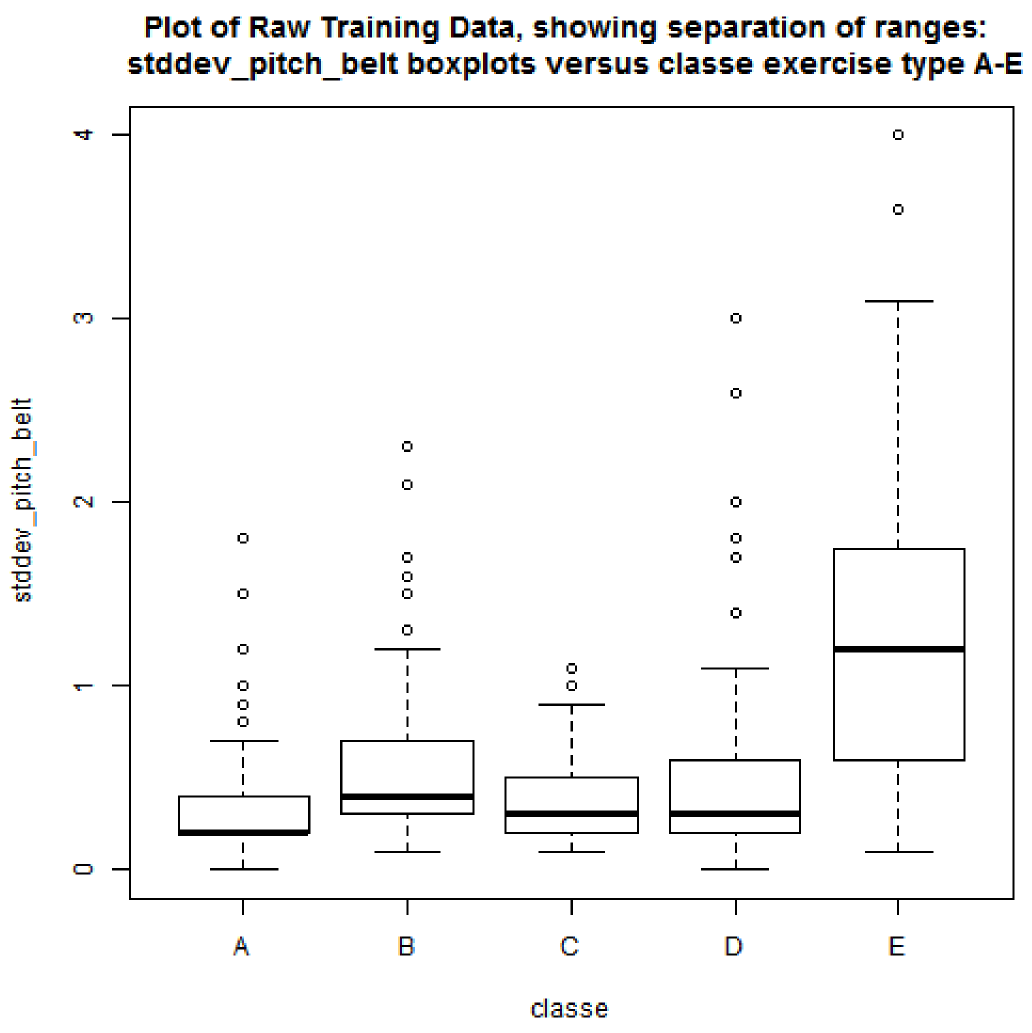
```
dim(pml_tst)  #test set is very small; training large
```

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

```
# colnames(pml_trn); colnames(pml_tst) #same col names in training and
print("note:  column names are identical in the two dataframes, except
```

```
## [1] "note:  column names are identical in the two dataframes, excep
```

```
# plot x=classe, y= everything else (one example shown below), observe
plot(pml_trn$classe,pml_trn[,32],main="Plot of Raw Training Data, show
```



Processing of Data:

The training dataset is divided into training dataframe trn (80%) and a test dataframe tst (20%).

Dataframe trn is subsetted to eliminate some of the original columns in several different ways.

First of all, some of the columns are substantially NA: these columns can be eliminated from a training dataset since these will not be useful in developing a predictive model.

Secondly, the first seven columns are also eliminated. The first column is an index number, while the others are timestamp and user name. These are not expected to be useful for predicting the “classe” exercise type (A-E), especially those derived from different users and different times. Although one can see in the “test” datafile that the user names correspond to those in the training datafile, it is a conscious choice to remove this variable as if one had not first examined the test datafile.

Third, some of the columns contain factor type data. Many of the factor columns contain blanks, DIV/0 and other non-numeric data which are not expected to add predictive value to a predictive model. THese can be eliminated.

In summary, the processed dataframe trn contains 53 columns (including classe). This processed training dataframe contains zero NA's. The dataframe is clean and tidy.

It is worth noting that the test dataframe tst is not processed: columns are not removed. Leaving all columns in a test set does not affect the prediction from the training set (at least in this case). The data from the 20-line pml-test.csv is also left in the raw format.

Code Chunk 3 Data Processing

```
# Divide the (large) training set into training data (80%)and testing
set.seed(3456)
inTrain <- createDataPartition(y=pml_trn$classe,p=0.8,list=FALSE)
trn <- pml_trn[inTrain,]
tst <- pml_trn[-inTrain,]
dim(trn); dim(tst)  #check dimensions
```

```
## [1] 15699   160
```

```
## [1] 3923  160
```

```
# Note:  Numeric data is in columns which are either zero NA's, or alm

# Data Processing:
# Remove columns which are substantially (>60%) NA
na_cols <- sapply(colnames(trn),
    function(x) if(sum(is.na(trn[,x])) > 0.60*nrow(trn)){return(TRUE)}
trn <- trn[,!na_cols]
sum(is.na(trn))  #zero NA's in remaining trn training dataframe, confi
```

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

```
# Remove first seven columns (not needed for predictive model)
trn <- trn[,8:93]

# Remove columns containing factors consisting of mainly blanks, DIV/0
trn_zerovar <- nearZeroVar(trn, saveMetrics = TRUE)
trn <- trn[,!trn_zerovar$nzv]
# remaining data is either "numeric", "integer"; the last column "clas
```

Random Forest Model

A random forest model rf_model is generated using the trn (training) dataframe.

The in-sample accuracy of this model is 0.994, with an accuracy standard deviation of 0.00169. The random forest package performs cross-validation within the algorithm. The best accuracy is obtained for mtry=2, or the number of variables tried at each split = 2.

The out-of-sample accuracy is expected to be slightly worse, but not substantially worse. Because of the (large) size of the training dataset, we should not expect overfitting. In particular, random forest models already contain internal cross-validation, and since the validation set is a (20%) subset of the original dataframe, we can expect similar out-of-sample accuracy in the validation “20% test” set.

Code Chunk 4 Random Forest Model

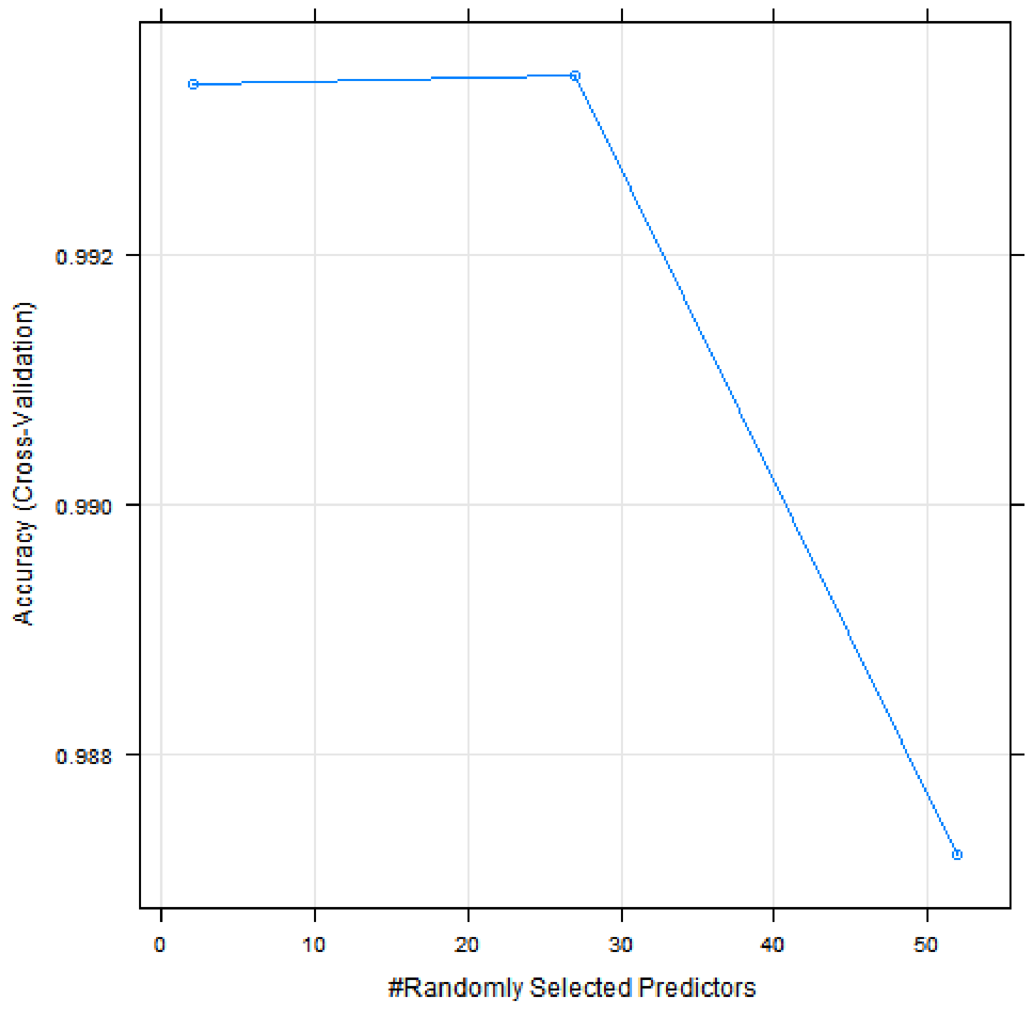
```
# rpart model
#rpart_model <- train(classe~.,data=trn,method="rpart")
#rpart_model
# Random Forests Model
rf_model <- train(classe~.,data=trn,method="rf",importance=TRUE,
    trControl=trainControl(method="cv",number=10))
rf_model
```

```
## Random Forest
##
## 15699 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
##
## Summary of sample sizes: 14128, 14130, 14130, 14130, 14128, 14128,
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa  Accuracy SD  Kappa SD
##    2    0.993     0.992  0.00181      0.00229
##   27    0.993     0.992  0.00202      0.00255
##   52    0.987     0.984  0.00258      0.00326
##
## Accuracy was used to select the optimal model using  the largest va
## The final value used for the model was mtry = 27.
```

```
rf_model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.55%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4460    2    1    0    1 0.0008960573
## B   23 3009    6    0    0 0.0095457538
## C    0   11 2719    8    0 0.0069393718
## D    0    0   22 2551    0 0.0085503304
## E    0    0    4    8 2874 0.0041580042
```

```
plot(rf_model)
```



```
rf_predict <- predict(rf_model,tst)
confusionMatrix(rf_predict,tst$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    3    0    0    0
##          B    0  753    0    0    0
##          C    0    3  682    6    1
##          D    0    0    2  636    1
##          E    0    0    0    1  719
## 
## Overall Statistics
## 
##                Accuracy : 0.9957
##                  95% CI : (0.9931, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9945
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9921   0.9971   0.9891   0.9972
## Specificity            0.9989   1.0000   0.9969   0.9991   0.9997
## Pos Pred Value         0.9973   1.0000   0.9855   0.9953   0.9986
## Neg Pred Value         1.0000   0.9981   0.9994   0.9979   0.9994
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1919   0.1738   0.1621   0.1833
## Detection Prevalence   0.2852   0.1919   0.1764   0.1629   0.1835
## Balanced Accuracy      0.9995   0.9960   0.9970   0.9941   0.9985
```

Results

The predictive (out of sample) validation accuracy for dataframe tst is 0.9926, only slightly worse than the in-sample accuracy. The out-of-sample confidence interval is 0.9894 to 0.995. Thus, it is expected that this random forest model can be applied to predicting the exercise type (A-E) of the 20 observations from the file separate file pml-testing.csv.

Other Models Considered

Models constructed using methods other than random forest method were considered.

An rpart model provided only 0.5 accuracy (in-sample and out-of-sample). With the rpart model, all of the D tst dataframe data was misclassified by the prediction. The lda method resulted in an accuracy of 0.7. For brevity, the code and results for these are not shown here.

A model was trained using the gbm method. This code runs for a while, but produces a model whose in-sample accuracy for dataframe trn is 0.959. Nice, but not as impressive as the random forests model. The out-of-sample accuracy is 0.9526, slightly worse than the in-sample accuracy, with a 95% confidence interval spanning from 0.9561 to 0.9683.

Code Chunk 5 gbm model

```
#code is commented out for faster knitter run... this may take up to 1
#gbm_model <- train(classe~.,data=trn,method="gbm")
#gbm_model
#summary(gbm_model)
#gbm_pred <- predict(gbm_model,tst)
#confusionMatrix(gbm_pred,tst$classe)
# best accuracy is 0.959 (in-sample training accuracy).
# The final values used for the model were n.trees = 150, interaction.
```

Improving Accuracy

The best way to improve accuracy is possibly eliminating NA, blank and DIV/0 entries in the collection of data. If all columns of numeric data were present, an improved random forest model could better predict the exercise type of another dataframe. In this study, only about 1/3 of the columns of the raw data are utilized for generating the random forest predictive model.

Predictive accuracy could be further improved by doing a “deep dive” to consider those data which were falsely categorized into the wrong exercise types. In particular, about half of the misclassified training data, in this work, is the 41 occurances of “D” misclassified as “C”. It may be possible to eliminate outliers, use moving averages to reduce noise, or perhaps determine improved placement of accelerometers or settings for accelerometers. By examining the misclassified data points, it may also be possible to generate new variables which are classifiable with higher accuracy.

##Executive Summary

A random forest predictive model is capable of predicting exercise type (A-E) with a 99.26% accuracy on a validation (out of sample) data set. The random forest accuracy is the higher than those of other types of models, including rpart, lda, and gbm which provides the second highest accuracy. The random forest model is applied to classifying 20 out of 20 exercise types correctly.

Appendix: Predict and Submit Answers to Coursera for pml-testing.csv dataframe pml_tst

```
# code commented out for knitting
#answers <- as.character(predict(rf_model, pml_tst))
#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.na
#  }
#}
#pml_write_files(answers)
```

Please note that the random forest model predicted 20/20 of the exercise motions (A-E) correctly.