PracticalML-Project

Sriram Cheruvu

July 17, 2018

Executive Summary

Based on a dataset provide by HAR http://groupware.les.inf.puc-rio.br/har we will try to predict the manner in which the user did the exercise. The training data set contains the target variable classe, all other variables will be used to predict for it.

The following key steps are performed:

1) Load, Process and Clean the data- I started by cleaning the dataset, removing columns that were not related and readings that were NA."" or DIV/0 values.
2) Explore the data, especially focussing on the paramaters we are interested in
3) Machine Learning model selection, where we try different models to help us answer our questions
4) A Conclusion where we answer the questions based on the data
5) Predicting the classification of the model on test set

For the ML I would be trying 2 models:

a) Fast recursive partitioning model (rpart) and compare this to
b) Random Forest model.

1) Load, Process and Clean the data

```r
train_dataURL = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test_dataURL = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

if (file.exists('pml-training.csv') == FALSE) {
  download.file(train_dataURL, 'pml-training.csv')
}
if (file.exists('pml-testing.csv') == FALSE) {
  download.file(test_dataURL, 'pml-testing.csv')
}
```

Next, read the data into the data frame.

```r
pmlTrainingData <- read.csv('pml-training.csv', na.strings=c("","NA","!DIV/0"))
finalTest <- read.csv('pml-testing.csv', na.strings=c("","NA","!DIV/0"))
```

2) Exploratory Data Analysis

Now we are ready to partition the training and test datasets.Once the partition is ready we will study the correlations of variables with classe.

Create Training & Cross Validation Datasets

The full training dataset it split into a training dataset and a testing dataset. The testing data will be used to cross validate our models.

```r
inTrain <- createDataPartition(pmlTrainingData$classe, p=.7, list=FALSE)
training <- pmlTrainingData[inTrain,]
testing <- pmlTrainingData[-inTrain,]

summary(training$classe)
```

```
##    A    B    C    D    E
## 3906 2658 2396 2252 2525
```

Clean Data

Next, time-related & recording variables and the row index variable X are removed because the purpose of the ML assignment is to make predictions.

```r
training <- training[, -c(1:7)]
testing <- testing[, -c(1:7)]
finalTest <- finalTest[, -c(1:7)]
```

First, I removed variables which contained a majority of missing values. NAs and blank fields were both marked as NA when the CSV was read.

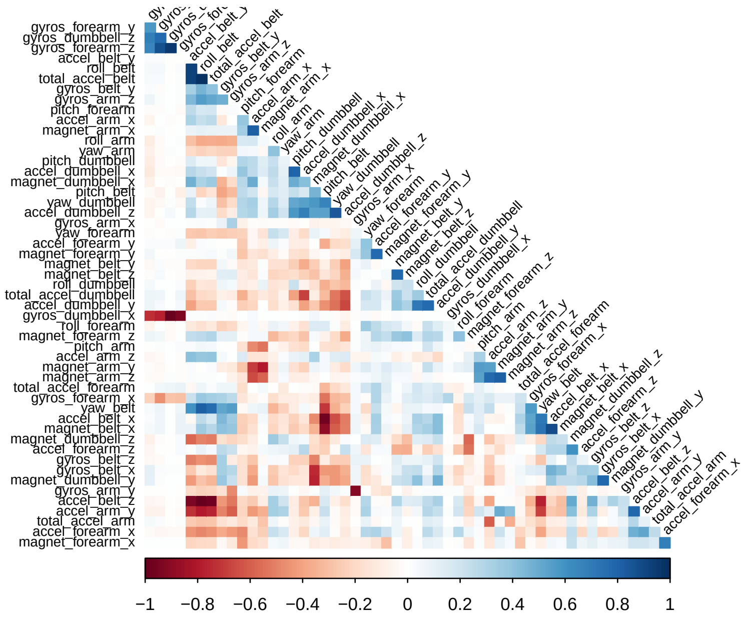
```r
removeNAs <- which(colSums(is.na(training)) > nrow(training)/2)
training <- training[, -removeNAs]
testing <- testing[, -removeNAs]
finalTest <- finalTest[, -removeNAs]
```

Study Correlations

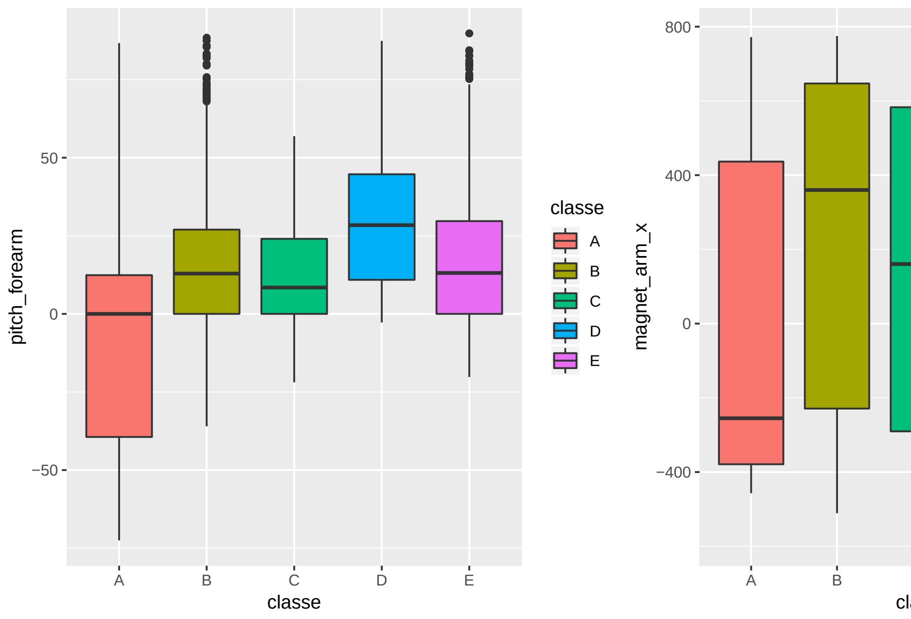
What are some fields that have high correlations with the classe?

```r
classeIndex <- which(names(training) == "classe")
correlations <- cor(training[, -classeIndex], as.numeric(training$classe))
bestCorrelations <- subset(as.data.frame(as.table(correlations)), abs(Freq)>0.3)
bestCorrelations
```

```
##             Var1 Var2      Freq
## 41 pitch_forearm    A 0.3423569
```



We see that there are some features that are quite correlated with each other. We will have a model with these excluded. Even the best correlations with classe are under 0.35. Let's check visually if there is indeed hard to



use these 2 as possible simple linear predictors.

3) Machine Learning

First, lets start with a fast recursive partitioning model (rpart) to start to see if that would produce reasonable predictions.

Recursive partitioning Model

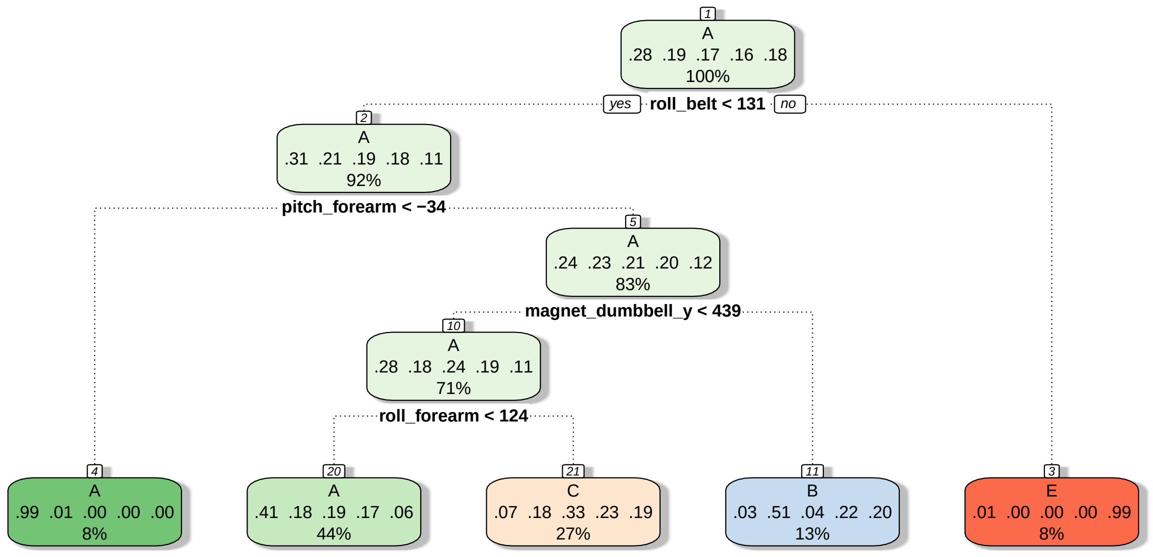
Starting with a simple model. Train the decision tree model

```r
rpModelFit <- train(classe ~ ., method="rpart", data=training, model=TRUE)
rpModelFit$finalModel
```

```
## n= 13737
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##    2) roll_belt< 130.5 12572 8677 A (0.31 0.21 0.19 0.18 0.11)
##      4) pitch_forearm< -33.95 1115    8 A (0.99 0.0072 0 0 0) *
##      5) pitch_forearm>=-33.95 11457 8669 A (0.24 0.23 0.21 0.2 0.12)
##       10) magnet_dumbbell_y< 438.5 9710 6978 A (0.28 0.18 0.24 0.19 0.11)
##         20) roll_forearm< 123.5 6069 3608 A (0.41 0.18 0.19 0.17 0.058) *
##         21) roll_forearm>=123.5 3641 2443 C (0.074 0.18 0.33 0.23 0.19) *
##       11) magnet_dumbbell_y>=438.5 1747  852 B (0.032 0.51 0.037 0.22 0.2) *
##    3) roll_belt>=130.5 1165   11 E (0.0094 0 0 0 0.99) *
```

Next, plot the model

```r
fancyRpartPlot(rpModelFit$finalModel, sub='')
```



Predict `classe` for cross validation dataset

```r
rpPreds <- predict(rpModelFit, newdata=testing)
rpConMatrix <- confusionMatrix(rpPreds, testing$classe)
rpConMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1508  456  453  438  168
##          B   31  398   44  182  149
##          C  132  285  529  344  288
##          D    0    0    0    0    0
##          E    3    0    0    0  477
##
## Overall Statistics
##
##                Accuracy : 0.4948
##                  95% CI : (0.482, 0.5077)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3402
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9008  0.34943  0.51559   0.0000  0.44085
## Specificity            0.6402  0.91445  0.78411   1.0000  0.99938
## Pos Pred Value         0.4988  0.49502  0.33523      NaN  0.99375
## Neg Pred Value         0.9420  0.85416  0.88461   0.8362  0.88807
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2562  0.06763  0.08989   0.0000  0.08105
## Detection Prevalence   0.5137  0.13662  0.26814   0.0000  0.08156
## Balanced Accuracy      0.7705  0.63194  0.64985   0.5000  0.72011
```

We observe a low accuracy with Recursive partitioning model.

```
rpAccuracy = rpConMatrix$overall[[1]]
percent(rpAccuracy)
```

```
## [1] "49.5%"
```

The estimated out of sample error with the cross validation dataset for this model is

```
percent(1.00-rpAccuracy)
```

```
## [1] "50.5%"
```

Unfortunately the estimated out of sample error for the `rpart` model was 51% and too high.

Random Forest Model

Next lets try a random forest model. Using this model, 20 predictions will be made for the test data set.I believe, Random Forests should be a better learning model for our dataset.

```
fitControl <- trainControl(method="cv", number=3, verboseIter=F)
rfModelFit <- train(classe ~., method="rf", data=training, trControl=fitControl)
rfModelFit$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: 2
##
##         OOB estimate of  error rate: 0.74%
## Confusion matrix:
```

```
##      A    B    C    D    E class.error
## A 3902    2    1    0    1 0.001024066
## B   14 2636    8    0    0 0.008276900
## C    0   26 2368    2    0 0.011686144
## D    0    0   35 2215    2 0.016429840
## E    0    0    3    7 2515 0.003960396
```

Predict `classe` for cross validation dataset.

```
rfPredictions <- predict(rfModelFit, newdata=testing)
rfConMatrix <- confusionMatrix(rfPredictions, testing$classe)
rfConMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    7    0    0    0
##          B    0 1127   13    0    0
##          C    0    5 1013   16    0
##          D    0    0    0  948    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9902, 0.9947)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9908
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9895   0.9873   0.9834   0.9982
## Specificity            0.9983   0.9973   0.9957   0.9996   1.0000
## Pos Pred Value         0.9958   0.9886   0.9797   0.9979   1.0000
## Neg Pred Value         1.0000   0.9975   0.9973   0.9968   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1915   0.1721   0.1611   0.1835
## Detection Prevalence   0.2856   0.1937   0.1757   0.1614   0.1835
## Balanced Accuracy      0.9992   0.9934   0.9915   0.9915   0.9991
```

We notice a much higher accuracy with a Random Forest Model. This model performed really well with an estimated out of sample of only 0.7%.

```
rfAccuracy = rfConMatrix$overall[[1]]
percent(rfAccuracy)
```

```
## [1] "99.3%"
```

The estimated out of sample error with the cross validation dataset for this model is

```
percent(1.00-rfAccuracy)
```

```
## [1] "0.731%"
```

4) Conclusion

The Random Forest model outperformed the the Recursive partitioning model by quite a bit. The random forest model was selected for final submissions of the project.

5) Test Data Simulation

The random forest clearly performs better, approaching 99% accuracy for in-sample and out-of-sample error so we will select this model and apply it to the test data set. We use the provided function to classify 20 data points from the test set by the type of lift.

```r
submissionPredictions <- predict(rfModelFit, newdata=finalTest)
submissionPredictions
```

```
##  [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
```