Machine Learing Course Project

Cobus Nel

Sunday, March 22, 2015

EXECUTIVE SUMMARY

The purpose of this document is to communicate the workflow that was followed to create the Machine Learning model required to predict the outcome of the twenty provided problem sets.

The data is based on measurements taken during dumbell excercises. The model is required to predict how the excercise was conducted (correct or incorrect). There is five different possible ways to conduct the excercise.

DATA

Data was provided in the form of two CSV files, the first is a training set and the second contain 20 unclassified lines to be used for prediction.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.3
```

```
#raw_data<-read.csv("pml-training.csv")
```

DATA CLEANING.

The data set contain 158 columns of which a number of columns contain invalid data such as zeros or NA. In order to use a practical set, the original data was reduced to the following columns:

```
columns=c(160,8:11,37:49,60:68,84:86,102)
features=raw_data[columns]
```

COLUMNS USED

The list of usable features are as listed below:

```
names(features)
```

```
##  [1] "classe"               "roll_belt"            "pitch_belt"
##  [4] "yaw_belt"             "total_accel_belt"     "gyros_belt_x"
##  [7] "gyros_belt_y"         "gyros_belt_z"         "accel_belt_x"
## [10] "accel_belt_y"         "accel_belt_z"         "magnet_belt_x"
## [13] "magnet_belt_y"        "magnet_belt_z"        "roll_arm"
## [16] "pitch_arm"            "yaw_arm"              "total_accel_arm"
## [19] "gyros_arm_x"          "gyros_arm_y"          "gyros_arm_z"
## [22] "accel_arm_x"          "accel_arm_y"          "accel_arm_z"
## [25] "magnet_arm_x"         "magnet_arm_y"         "magnet_arm_z"
## [28] "roll_dumbbell"        "pitch_dumbbell"       "yaw_dumbbell"
## [31] "total_accel_dumbbell"
```

CORRELATION BETWEEN FEATURES

Since there is such a large number of columns, it make sense to investigate the columns that are highly correlated to other features in the data set in order to reduce pair-wise data sets. The columns printed in the list below could be safely removed.

```
correlation_matrix<-cor(features[,2:31])
higly_correlated<-findCorrelation(correlation_matrix, cutoff=0.75)
print(names(features[,2:31])[higly_correlated])
```

```
##  [1] "yaw_belt"         "accel_belt_z"     "accel_arm_y"
##  [4] "roll_belt"        "total_accel_belt" "magnet_belt_x"
##  [7] "accel_belt_x"     "accel_arm_x"      "accel_arm_z"
## [10] "magnet_arm_y"     "magnet_belt_y"    "gyros_arm_y"
```

In the event, machine learning algorithms were used that select important features as part of the algorithm (Random trees and boosting) so the above listed columns were not removed from the final set.

Exploratory Models with Cross Validation

The 'R' Script prelim.R (As below) were used to create preliminary models and save to file. Since models can take some time to excecute, it made sense to save the models for later processing.

```
#
# Preliminary.R
#
library(caret)
library(parallel)
library(doParallel)

# Enable multi cores
cl <- makeCluster(detectCores())
registerDoParallel(cl)

# Load training data
raw_data<-read.csv("pml-training.csv")
columns=c(160,8:11,37:49,60:68,84:86,102)
features=raw_data[columns]
inTrain<-createDataPartition(y=features$classe, p=0.2, list=FALSE)
training<-features[inTrain,]

# Use Cross Validation
train_control <- trainControl(method="cv", number=5)

# Random Forest
print("Random Forest")
model_rf<-train(classe ~ ., data=training, method="rf")
save(model_rf, file="prelim_model_rf.rdata")

# Random Forest with Principal Component Analysis
print("RCA")
```

```
model_rf_pca<-train(classe ~ ., data=training, method="rf", preProcess="pca")
save(model_rf_pca, file="prelim_model_rf_pca.rdata")

# Boosting
print("Boosting")
model_gbm<-train(classe ~ ., data=training,  trControl=train_control, method="gbm", verbose=FALSE)
save(model_gbm, file="prelim_model_gbm.rdata")
```

Random Forest

The outcome of the random Forest test is shown below:

```r
print(model_rf)
```

```
## Random Forest
##
## 14718 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9757    0.9692  0.002685     0.003395
##   16    0.9761    0.9697  0.003120     0.003945
##   30    0.9676    0.9591  0.004130     0.005218
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 16.
```

Random Forest with Principal Component Analysis

The outcome of the test utilizing random Forest with Principal Component Analysis is shown below:

```r
print(model_rf_pca)
```

```
## Random Forest
##
## 14718 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## Pre-processing: principal component signal extraction, scaled, centered
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
```

```
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9177    0.8959  0.003300     0.004133
##   16    0.8936    0.8655  0.005483     0.006939
##   30    0.8934    0.8652  0.005378     0.006816
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Boosting

The outcome of the test utilizing boosting is shown below:

```
model_gbm
```

```
## Stochastic Gradient Boosting
##
## 3927 samples
## 30 predictor
## 5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 3142, 3142, 3141, 3142, 3141
##
## Resampling results across tuning parameters:
##
##  interaction.depth  n.trees  Accuracy   Kappa      Accuracy SD  Kappa SD
##  1                   50      0.6786422  0.5913466  0.026048668  0.03211281
##  1                  100      0.7435704  0.6746881  0.017150380  0.02144835
##  1                  150      0.7756558  0.7156237  0.015591524  0.01965484
##  2                   50      0.7782039  0.7188334  0.008917498  0.01125014
##  2                  100      0.8370292  0.7935647  0.012182750  0.01530988
##  2                  150      0.8698786  0.8352685  0.013826944  0.01747215
##  3                   50      0.8377906  0.7944716  0.021085969  0.02659330
##  3                  100      0.8792982  0.8471951  0.019321975  0.02445431
##  3                  150      0.9027248  0.8769036  0.015496837  0.01962307
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150, interaction.depth = 3 and shrinkage = 0.1.
```

Selected Candidate

From the above, it would appear that the *Random Forrest* algorithm provide the best model in this instance.

Most Important Features.

The graph below list the most important features as determined by the Random Forrest algorithm.

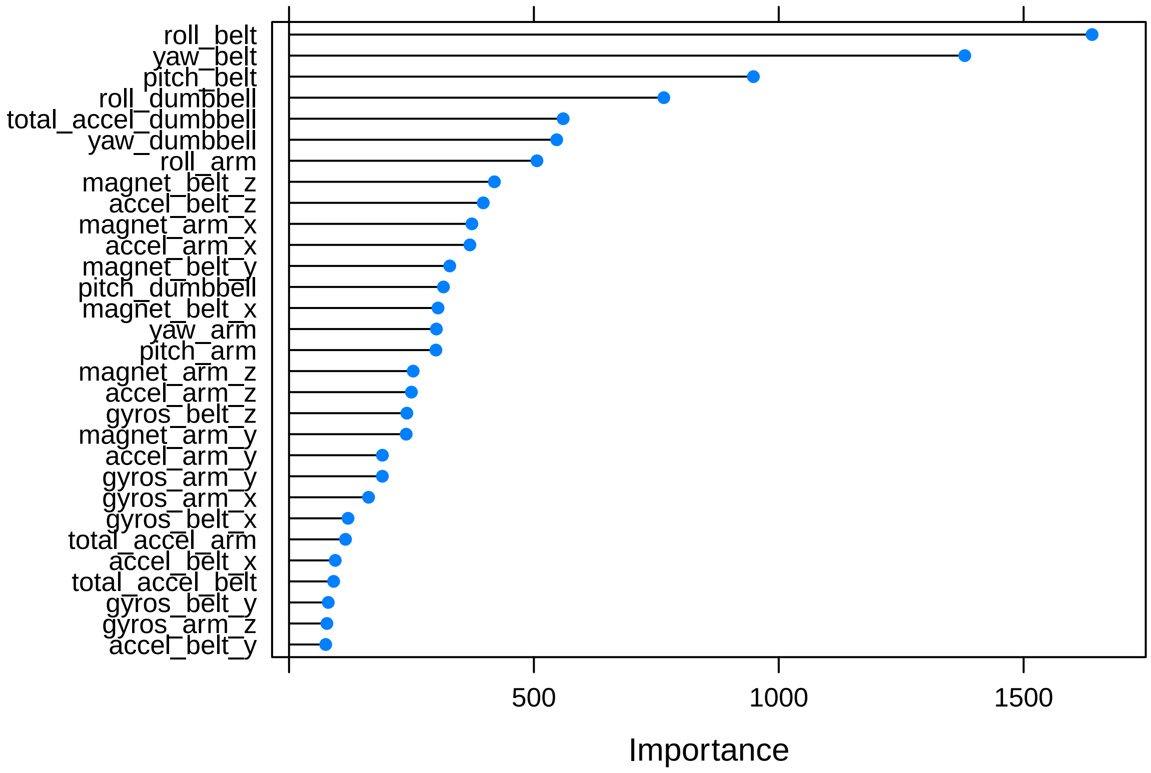
```r
importance<-varImp(model_rf, scale=FALSE)
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.1.3
```

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

```r
plot(importance)
```



Prediction Model

The final prediction model was created using train_model.R

```
#
# train_model.R
#
```

```
library(caret)
library(parallel)
library(doParallel)

cl <- makeCluster(detectCores())
registerDoParallel(cl)

raw_data<-read.csv("pml-training.csv")
columns=c(160,8:11,37:49,60:68,84:86,102)
features=raw_data[columns]
inTrain<-createDataPartition(y=features$classe, p=0.75, list=FALSE)
training<-features[inTrain,]
testing<-features[-inTrain,]

# Important. Save testing data.
save(testing, file="testing.rdata")

# Cross Validatation
train_control <- trainControl(method="cv", number=5)

model_rf<-train(classe ~ ., data=training, method="rf")
save(model_rf, file="model_rf.rdata")
```

This model provided the following output:

```
load(file="model_rf.rdata")
model_rf
```

```
## Random Forest
##
## 14718 samples
##    30 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 14718, 14718, 14718, 14718, 14718, 14718, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9757    0.9692  0.002685     0.003395
##   16    0.9761    0.9697  0.003120     0.003945
##   30    0.9676    0.9591  0.004130     0.005218
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 16.
```

MODEL TESTING AND ACCURACY

The model was tested against the training data with results and confusion matrix, below:

```
prediction<-predict(model_rf, testing)
load(file="testing.rdata")
confusionMatrix(prediction, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1378    9    0    3    0
##          B    8  929    8    2    4
##          C    2    9  841    8    1
##          D    6    2    6  788    5
##          E    1    0    0    3  891
##
## Overall Statistics
##
##                Accuracy : 0.984
##                  95% CI : (0.98, 0.988)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.98
##  Mcnemar's Test P-Value : 0.449
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             0.988    0.979    0.984    0.980    0.989
## Specificity             0.997    0.994    0.995    0.995    0.999
## Pos Pred Value          0.991    0.977    0.977    0.976    0.996
## Neg Pred Value          0.995    0.995    0.997    0.996    0.998
## Prevalence              0.284    0.194    0.174    0.164    0.184
## Detection Rate          0.281    0.189    0.171    0.161    0.182
## Detection Prevalence    0.283    0.194    0.176    0.165    0.183
## Balanced Accuracy       0.992    0.987    0.989    0.988    0.994
```

Based on the above, an accuracy of 98.4% is expected from the model.

In the event, the model accurately predicted 100% of the test cases during submission!!

FINAL TEST CASES

```
#
# train_model.R
#
library(caret)

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.names=FALSE)
  }
}

load("model_rf.rdata")
raw_data<-read.csv("pml-testing.csv")
columns=c(160,8:11,37:49,60:68,84:86,102)
test_cases=raw_data[columns]
answers<-predict(model_rf, test_cases )
pml_write_files(answers)

print(answers)
```

```
##  [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
```