mlPrediction

jbratanov

Tuesday, June 21, 2016

Summary

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. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Data

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions:

- **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

This data is assessed using the **"classe"** variable in the training dataset

Data Processing

```
# load train data
train_df <- read.csv("c:/coursera/machineLearning/project/pml-training.csv", colClasses="character", he
                     stringsAsFactors = FALSE)
# load test data
test_df <- read.csv("c:/coursera/machineLearning/project/pml-testing.csv", colClasses="character", head
                     stringsAsFactors = FALSE)
```

Data Cleansing

- Observe raw data through RStudio and determine columns not useful for prediction exercise
  - Will remove first 7 variables which are related to user, time, etc..
  - Was going to consider timestamp as an option thinking time of day may be an influence, but not enough time span
- Validate variable programmically to see which ones are numeric
- Validate to see which variables have enough useful data to use for prediction. Using 95% as tolerance.

```
# Load packages required for prediction functionality
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.5

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 3.2.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.5
```

```
# remove first 7 vars
train_df[1:7] <- list(NULL)
test_df[1:7] <- list(NULL)
###############################################
# Pre-process training data
#remove columns which are NA, missing values
# or non-numeric using 95% tolerance
###############################################
tolerance <- nrow(train_df) * .05
colCnt <- ncol(train_df)-1
listCnt=1
badColList <- list()
# Loop and build list of columns which don't meet clean data criteria
# Remember not to change or delete "classe" variable
for (i in 1:colCnt)
  {
    if (colSums(is.na(train_df[i]))  > tolerance
       || sum(train_df[i] == "") > tolerance)
      {
        badColList[listCnt] <- colnames(train_df[i])
        listCnt <- listCnt + 1
      }
    else
      {
         # change columns we are keeping into numeric
         train_df[,i] <- as.numeric(as.character(train_df[,i]))
      }
  }
# Drop list of columns not used
train_df <- train_df[, !(names(train_df) %in% badColList)]

###############################################
# Pre-process test data
#remove columns which are NA, missing values
# or non-numeric using 95% tolerance
###############################################
tolerance <- nrow(test_df) * .05
colCnt <- ncol(test_df)-1
testlistCnt=1
testbadColList <- list()
# Loop and build list of columns which don't meet clean data criteria
# Remember not to change or delete "classe" variable
for (i in 1:colCnt)
```

```
  {
    if (colSums(is.na(test_df[i]))  > tolerance
       || sum(test_df[i] == "") > tolerance)
      {
        testbadColList[testlistCnt] <- colnames(test_df[i])
        testlistCnt <- testlistCnt + 1
      }
    else
      {
         # change columns we are keeping into numeric
         test_df[,i] <- as.numeric(as.character(test_df[,i]))
      }
  }
# Drop list of columns not used
test_df <- test_df[, !(names(test_df) %in% testbadColList)]
```

Data to be used for prediction After data cleaning, the following variables will be used.

```
names(train_df)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
names(test_df)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "problem_id"
```

Prediction models Set PCA to reduce the dimension of the data. Get to the important stuff using cross-validation (cv) method Tried “rf”, svm" and “glm” offline. Used “rf” with the high accuracy returns

```
# Set seed to 1.  easy to remember
set.seed(1)
# Create training and cross-validation datasets
inTrain = createDataPartition(train_df$classe, p = 0.7, list=FALSE)
myTrain = train_df[inTrain,]
cvTrain = train_df[-inTrain,]

tcData <- trainControl(method = "cv", number = 5, verboseIter=FALSE , preProcOptions="pca", allowParall
# Using random forest (rf) method as offline pre-processing check validates good accuracy
myTrain_rf <- train(classe ~ ., data = myTrain, method = "rf", trControl=tcData)
```

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

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

Accuracy of training dataset

The acuracy of the training dataset is 100%

```
predTrain <- predict(myTrain_rf, myTrain)
confusionMatrix(predTrain, myTrain$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3906    0    0    0    0
##          B    0 2658    0    0    0
##          C    0    0 2396    0    0
##          D    0    0    0 2252    0
##          E    0    0    0    0 2525
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Accuracy of cross-validation dataset

The acuracy of the CV data 99+ percent

```
predCV <- predict(myTrain_rf, cvTrain)
confusionMatrix(predCV, cvTrain$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    0    0    0
##          B    0 1130    5    0    1
##          C    2    2 1018    4    1
##          D    0    1    3  958    2
##          E    1    0    0    2 1078
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9927, 0.9966)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9921   0.9922   0.9938   0.9963
## Specificity            0.9986   0.9987   0.9981   0.9988   0.9994
## Pos Pred Value         0.9964   0.9947   0.9912   0.9938   0.9972
## Neg Pred Value         0.9993   0.9981   0.9984   0.9988   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1920   0.1730   0.1628   0.1832
## Detection Prevalence   0.2850   0.1930   0.1745   0.1638   0.1837
## Balanced Accuracy      0.9984   0.9954   0.9952   0.9963   0.9978
```

```
predictionValues <- predict(myTrain_rf, test_df)
predictionValues
```

Prediction

```
##  [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
```

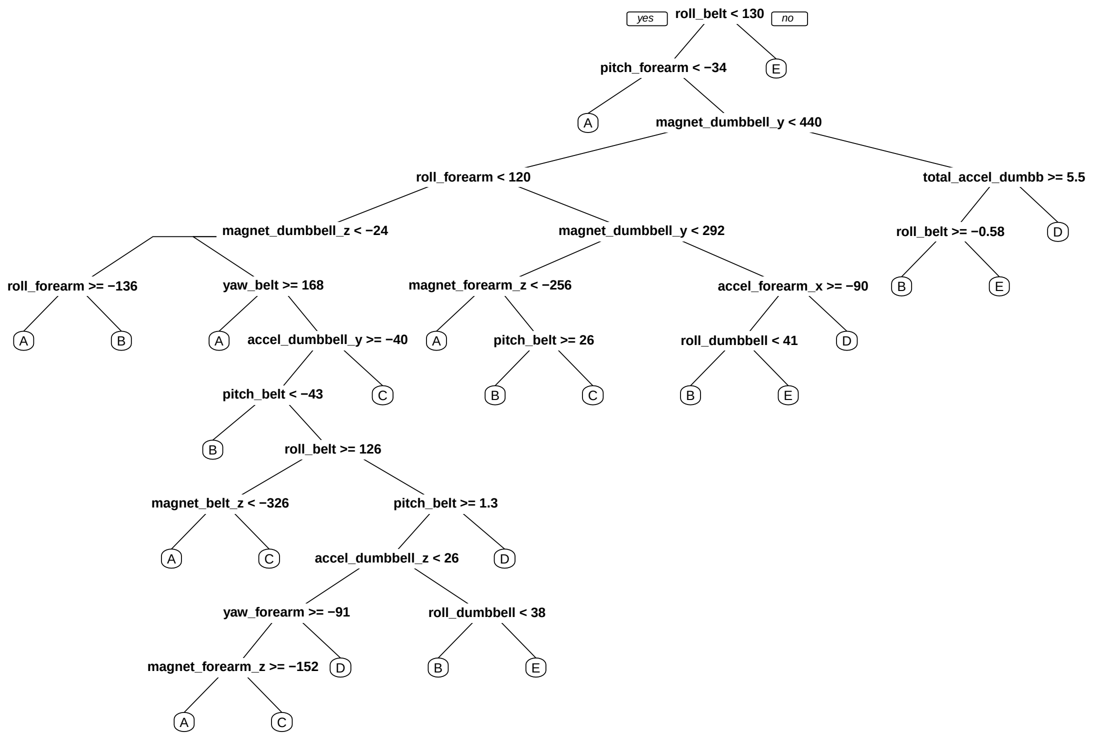
```
# plot data
library(rpart.plot)
```

Plot decision tree for graphic visualization - Plays well with random forest

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

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

```
plotData <- rpart(classe ~ ., data=myTrain, method="class")
prp(plotData)
```



Contributions

Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz4CECF4g6O