Excercise performance analysis and prediction

Cesar Lugo

November 24, 2016

Excercise performance analysis and prediction

Executive summary

In this report we analized weight lifting human activity recognition data that was extracted using data collection devices on six healthy persons. This is because this way we can help people in their excercise and training by telling them if they are doing well so they can know when they should correct their excercising techniqes.

Read more: http://groupware.les.inf.puc-rio.br/har#dataset#ixzz4QyMvpgg2 (http://groupware.les.inf.puc-rio.br/har#dataset#ixzz4QyMvpgg2)

We applied data science through regression models and machine learning to create a prediction model for the class of performance achieved. Also, we quantify the expected out of sample error. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

Read more: http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz4QyKULbdM (http://groupware.les.inf.puc-rio.br/har#wle_paper_section#ixzz4QyKULbdM)

Data gathering, pre procesing and cleansing

First we get the data for testing and training, and clean it.

Model fitting for classe outcome

We try here to fit models to identify which one better explains the classe outcome in the training set.

Exploratory data analysis

Let’s take a look at the correlatio coeficcients first for all relevant features as they correlate to the class outcome:

```
# create partition of the testing set for training and testing models
intrain <- createDataPartition(y=pmlTrainingSource$classe, p=0.75, list=FALSE)
pmlTraining <- pmlTrainingSource[intrain, ]
pmlTrainingforTest <- pmlTrainingSource[-intrain, ]

# Obtain all potential predictor features by removing the class outcome from the training set

pmlTrainingAllPredictors <- pmlTraining[,-c(grep("classe", names(pmlTraining)))]
pmlTrainingPredictors <- pmlTraining[,c(grep("magnet_arm_x|accel_arm_x|total_accel_forearm|magnet_dumbbell_z|acce
l_dumbbell_x|pitch_dumbell|roll_arm|pitch_dumbbell|total_accel_belt|accel_dumbbell_z|magnet_dumbbell_x|roll_belt|
yaw_arm|accel_arm_z|roll_dumbbell|gyros_dumbbell_y|roll_forearm|magnet_belt_x|yaw_belt|gyros_belt_x|pitch_belt",
names(pmlTraining)))]

# Calculate the correlation of each potential predictor with the outcome classe sorted to identify features with
 the highest correlation with the classe outcome
pmlTrainingSummaries <-  pmlTrainingPredictors
pmlTrainingSummaries$numClasse <- as.numeric(pmlTraining$classe)

kable(data.frame(sort(cor(pmlTrainingSummaries)[c(grep("numClasse", names(pmlTrainingSummaries))),])))
```

	sort.cor.pmlTrainingSummaries..c.grep..numClasse…names.pmlTrainingSummaries…..
gyros_belt_x	0.0004152
pitch_belt	0.0118314
yaw_belt	0.0137532
magnet_belt_x	0.0157739
roll_forearm	0.0277455
gyros_dumbbell_y	0.0348995
roll_dumbbell	0.0436160
yaw_arm	0.0511691
accel_arm_z	0.0516551
roll_belt	0.0640197

	sort.cor.pmlTrainingSummaries..c.grep..numClasse…names.pmlTrainingSummaries…..
magnet_dumbbell_x	0.0710392
accel_dumbbell_z	0.0723457
total_accel_belt	0.0799490
pitch_dumbbell	0.0829936
roll_arm	0.0831901
accel_dumbbell_x	0.1136070
magnet_dumbbell_z	0.1457660
total_accel_forearm	0.1539886
accel_arm_x	0.2481740
magnet_arm_x	0.3025691
numClasse	1.0000000

We can see that the mostly correlated variables with the classe outcome are magnet_arm_x, accel_arm_x , total_accel_forearm , magnet_dumbbell_z , accel_dumbbell_x , roll_arm , pitch_dumbbell , total_accel_belt , roll_arm, accel_dumbbell_z , magnet_dumbbell_x , roll_belt , yaw_arm , accel_arm_z , roll_dumbbell , gyros_dumbbell_y , roll_forearm , magnet_belt_x , yaw_belt , gyros_belt_x , pitch_belt . We can use these features as predictors in our models.

Fitting our first prediction model

Here we fit a first model using the selected features as predictors:

```
# Fit a model with all features as predictors of the class outcome
pmlFitAll <- train(x = pmlTrainingPredictors, y = pmlTraining$classe, method = "rpart")

max(pmlFitAll$results$Accuracy)
```

```
## [1] 0.4841086
```

Here we can see that the first model has a low accuracy.

Fitting our second prediction model

Now we try to fit another model also using the selected features as predictors:

```
# Fit a model with all features as predictors of the class outcome
fitCtrl <- trainControl(method = "cv", number = 4, allowParallel = TRUE)

pmlFitAll <- train(x = pmlTrainingPredictors, y = pmlTraining$classe, method = "rf", prof = TRUE, trControl = fit
Ctrl)

max(pmlFitAll$results$Accuracy)
```

```
## [1] 0.98811
```

We can see that with this second model we get much better (quite high) accuracy. Hence, we will keep this one as our prediction model.

Cross Validation

Here we use our partition obtained from our original test set to perform cross validation, so we can see how our selected prediction model performs.

```
# Predicting:
predictionRfFitAll <- predict(pmlFitAll, pmlTrainingforTest)

# Test results on TestTrainingSet data set:
confusionMatrix(predictionRfFitAll, pmlTrainingforTest$classe)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1390    2    0    0    0
##          B    4  942    3    0    0
##          C    0    5  846    6    2
##          D    0    0    6  797    2
##          E    1    0    0    1  897
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9935          
##                  95% CI : (0.9908, 0.9955)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9917          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9926   0.9895   0.9913   0.9956
## Specificity            0.9994   0.9982   0.9968   0.9980   0.9995
## Pos Pred Value         0.9986   0.9926   0.9849   0.9901   0.9978
## Neg Pred Value         0.9986   0.9982   0.9978   0.9983   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2834   0.1921   0.1725   0.1625   0.1829
## Detection Prevalence   0.2838   0.1935   0.1752   0.1642   0.1833
## Balanced Accuracy      0.9979   0.9954   0.9931   0.9947   0.9975
```

So we can validate that our model has a high accuracy, and confirm the model selection.

Expected out of sample error

Now we will take a look at the out of sample error.

```
pmlFitAll$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, prof = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 11
##
##         OOB estimate of  error rate: 0.87%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4179    3    1    0    2 0.001433692
## B   14 2804   28    2    0 0.015449438
## C    0   22 2531   14    0 0.014024153
## D    0    1   17 2391    3 0.008706468
## E    0    5    8    8 2685 0.007760532
```

With this analysis we can see that the expected out os sample error os very low, being about .84% .

Conclusions

Our second and selected fitted model using the selected features as predictors fits very well, with an accuracy higher than 99% and an expected error of about .84% .