index

Simon Merino

23 de octubre de 2017

Prediction Assignment

This coursera exercise is intended to category the execution of barbell lifts from the parameters captured by accelerometers on the belt, forearm, arm and dumbbell. The categories are 6, one for the correct execution and 5 extra more for incorrect ones.

Training data is available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

While test data is available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The original training raw data has 19622 samples with 160 variables each, while testing data has 20samples with 160 variables each

Data Cleaning

Not all 160 dimensions contain relevant information, so the next step will be to eliminate not valuable features.

All NAs values

In the training data, 67 variables have all records with NA values. These are directly deleted from both the training data.

```
NAsTraining<-colSums(is.na(training))

# Then remove
training<-training[,NAsTraining==0]
testing<-testing[,NAsTraining==0]
```

Empty and #DIV/0 values

There are 9 features for which the only available values are either empty or #DIV/0. These features are also discarded.

```
training<-subset(training,select=-c(kurtosis_yaw_belt,skewness_yaw_belt,amplitude_yaw_belt,kurtosis_yaw_
amplitude_yaw_dumbbell,
kurtosis_yaw_forearm,
skewness_yaw_forearm,
amplitude_yaw_forearm))

testing<-subset(testing,select=-c(kurtosis_yaw_belt,skewness_yaw_belt,amplitude_yaw_belt,kurtosis_yaw_d
amplitude_yaw_dumbbell,
kurtosis_yaw_forearm,
skewness_yaw_forearm,
amplitude_yaw_forearm))
```

Mostly Empty and some values

There are 24 additional variables which are mainly empty. They have some non-zero values but they are not representative, so these are deleted also.

```
training<-subset(training,select=-c(kurtosis_roll_belt
  ,kurtosis_picth_belt
  ,skewness_roll_belt
  ,skewness_roll_belt.1
  ,max_yaw_belt
  ,min_yaw_belt
  ,kurtosis_roll_arm
  ,kurtosis_picth_arm
  ,kurtosis_yaw_arm
  ,skewness_roll_arm
  ,skewness_pitch_arm
  ,skewness_yaw_arm
  ,kurtosis_roll_dumbbell
  ,kurtosis_picth_dumbbell
  ,skewness_roll_dumbbell
  ,skewness_pitch_dumbbell
  ,max_yaw_dumbbell
  ,min_yaw_dumbbell
  ,kurtosis_roll_forearm
  ,kurtosis_picth_forearm
  ,skewness_roll_forearm
  ,skewness_pitch_forearm
  ,max_yaw_forearm
  ,min_yaw_forearm))


testing<-subset(testing,select=-c(kurtosis_roll_belt
  ,kurtosis_picth_belt
  ,skewness_roll_belt
  ,skewness_roll_belt.1
  ,max_yaw_belt
  ,min_yaw_belt
  ,kurtosis_roll_arm
  ,kurtosis_picth_arm
  ,kurtosis_yaw_arm
  ,skewness_roll_arm
  ,skewness_pitch_arm
  ,skewness_yaw_arm
  ,kurtosis_roll_dumbbell
  ,kurtosis_picth_dumbbell
  ,skewness_roll_dumbbell
  ,skewness_pitch_dumbbell
  ,max_yaw_dumbbell
  ,min_yaw_dumbbell
  ,kurtosis_roll_forearm
  ,kurtosis_picth_forearm
  ,skewness_roll_forearm
  ,skewness_pitch_forearm
  ,max_yaw_forearm
```

```
  ,min_yaw_forearm))
```

Non-info variables

Finally all timestamp and regular ids are removed.

```
training<-subset(training,select=-c(raw_timestamp_part_1,raw_timestamp_part_2,cvtd_timestamp,X,new_windo


testing<-subset(testing,select=-c(raw_timestamp_part_1,raw_timestamp_part_2,cvtd_timestamp,X,new_window
```

Correcting outlier values

Having a look at the distribution of values, for some records there are extreme values that seem like plain wrong. They are not so frequent so will not help in the categorization exercise. Therefore, these entries are corrected using 2 strategies: For variables in which the mean value is around zero, then they are corrected by 0 directly; for variables whose mean is clearly not around zero, then the mean value is entered.

```
for (i in 1:nrow(training)) {
  if(training$gyros_dumbbell_y[i]>50){training$gyros_dumbbell_y[i]=0}
  if(training$gyros_dumbbell_x[i]<(-50)){training$gyros_dumbbell_x[i]=0}
  if(training$gyros_dumbbell_z[i]>50){training$gyros_dumbbell_z[i]=0}
  if(training$gyros_forearm_x[i]<(-10)){training$gyros_forearm_x[i]=0}
  if(training$gyros_forearm_y[i]>50){training$gyros_forearm_y[i]=0}
  if(training$gyros_forearm_z[i]>50){training$gyros_forearm_z[i]=0}
  if(training$magnet_belt_x[i]>350){training$magnet_belt_x[i]=0}
  if(training$magnet_belt_y[i]<300){training$magnet_belt_y[i]=mean(training$magnet_belt_y)}
  if(training$magnet_belt_z[i]>0){training$magnet_belt_z[i]=mean(training$magnet_belt_z)}

  if(training$magnet_dumbbell_y[i]<(-1500)){training$magnet_dumbbell_y[i]=0}
}
```

Cross Validation

In order to test model accuracy, training data is split between 2 parts, an 80% of the data is dedicated to model construction while the 20% remaining is used for model tuning.

```
selectionvector<-createDataPartition(y=training$classe,p=0.80,list=FALSE)
just_for_training<-training[selectionvector,]
just_for_testing<-training[-selectionvector,]
```

Model Construction

As a first try, a *K-Nearest Neighbors* classification algorithm is exercised in the training data.

```
model_knn<-train(classe~.,method="knn", data=just_for_training)
```

The algorithm execution is smooth but doesn´t yield a good enough accuracy. Therefore, it is discarded.

```
confusionMatrix(just_for_testing$classe,predict(model_knn,newdata=just_for_testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1080   12    7   14    3
##          B   38  658   27   22   14
##          C    6   28  627   18    5
##          D    4    4   44  586    5
##          E    4   32   18   26  641
##
## Overall Statistics
##
##                Accuracy : 0.9156
##                  95% CI : (0.9065, 0.9241)
##     No Information Rate : 0.2886
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8933
##  Mcnemar's Test P-Value : 2.491e-11
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9541   0.8965   0.8672   0.8799   0.9596
## Specificity            0.9871   0.9683   0.9822   0.9825   0.9754
## Pos Pred Value         0.9677   0.8669   0.9167   0.9114   0.8890
## Neg Pred Value         0.9815   0.9760   0.9704   0.9756   0.9916
## Prevalence             0.2886   0.1871   0.1843   0.1698   0.1703
## Detection Rate         0.2753   0.1677   0.1598   0.1494   0.1634
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9706   0.9324   0.9247   0.9312   0.9675
```

Finally, a *Random Forest* algorithm is exercised.

```
model_randomForest<-train(classe~.,method="rf", data=just_for_training)
confusionMatrix(just_for_testing$classe,predict(model_randomForest,newdata=just_for_testing))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    0    1    0    1
##          B    3  754    2    0    0
##          C    0    1  678    5    0
##          D    0    0    7  634    2
##          E    1    2    0    2  716
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.99, 0.9955)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9913
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9960   0.9855   0.9891   0.9958
## Specificity            0.9993   0.9984   0.9981   0.9973   0.9984
## Pos Pred Value         0.9982   0.9934   0.9912   0.9860   0.9931
## Neg Pred Value         0.9986   0.9991   0.9969   0.9979   0.9991
## Prevalence             0.2850   0.1930   0.1754   0.1634   0.1833
## Detection Rate         0.2840   0.1922   0.1728   0.1616   0.1825
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9979   0.9972   0.9918   0.9932   0.9971
```

This model yields an accuracy of over 0.99 on the testing data. therefore it looks like a good try for tackling the real testing data.

Variable Importance

Also, an evaluation of the model shows that the most important variables that add value to the model are the following ones

```
varImp(model_randomForest)
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 57)
##
##                      Overall
## roll_belt             100.00
## pitch_forearm          60.29
## yaw_belt               57.37
## pitch_belt             47.24
## magnet_dumbbell_y      44.40
## magnet_dumbbell_z      44.21
## roll_forearm           43.23
## accel_dumbbell_y       22.50
## accel_forearm_x        19.48
## magnet_dumbbell_x      18.93
## roll_dumbbell          17.70
## magnet_belt_z          17.38
## accel_belt_z           17.04
## accel_dumbbell_z       15.31
## magnet_forearm_z       15.04
## total_accel_dumbbell   14.94
## magnet_belt_y          13.79
## gyros_belt_z           13.44
## yaw_arm                12.79
## magnet_belt_x          11.36
```

With the intention of lowering the runtime of the model construction, a couple of new models are constructed with the top 9 and 12 variables in importance.

```
# TOP 9
model_randomForest_9<-train(classe~roll_belt+pitch_forearm+yaw_belt+magnet_dumbbell_y+magnet_dumbbell_z
```

```
top9_matrix<-confusionMatrix(just_for_testing$classe,predict(model_randomForest_9,newdata=just_for_testi

#TOP 12
model_randomForest_12<-train(classe~roll_belt+pitch_forearm+yaw_belt+magnet_dumbbell_y+magnet_dumbbell_z
top12_matrix<-confusionMatrix(just_for_testing$classe,predict(model_randomForest_12,newdata=just_for_tes
```

Corresponding Accuracies for these models are

```
top9_matrix$overall[1]
```

```
##  Accuracy
## 0.9841958
```

```
top12_matrix$overall[1]
```

```
##  Accuracy
## 0.9892939
```

Therefore, the 12 variable Random Forest Model is the one actually picked as a proposed solution to this exercise

Testing Data Estimation

This is the actual outcome which is submitted to the course assignment.

```
predict(model_randomForest_12,newdata=testing)
```

```
##  [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
```