Predict Excercise Type

Bishnu Poudel

May 18, 2019

Executive Summary

The goal of our project is to predict the manner in which the subjects did their exercise. This is the "classe" variable in the training set. We use only the readings from the machines to predict our outcome, and we ignore other manufactured variables. I've created a report describing how I built the models. I used cross validation set (named 'testing' in the report) from the the training data itself. We will also use our best prediction model to predict 20 different test cases.

Read the raw datasets

```
rawtraining<- read.csv("training.csv")
rawtesting<- read.csv("testing.csv")
```

check if any of the dependent variable is missing or is na

```
any(is.na(rawtraining$classe)); any( trimws( rawtraining$classe)=='')
```

```
## [1] FALSE
```

```
## [1] FALSE
```

However, we’ve quite some missing values and NAs in the other variables. We will remove all such fields.

Columns with missing data in them.

```
library(DataExplorer)
data.frame( profile_missing(rawtraining) )[data.frame( profile_missing(rawtraining)
 )$num_missing>0, ]
```

```
##                      feature num_missing pct_missing
## 18              max_roll_belt       19216   0.9793089
## 19             max_picth_belt       19216   0.9793089
## 21              min_roll_belt       19216   0.9793089
## 22             min_pitch_belt       19216   0.9793089
## 24        amplitude_roll_belt       19216   0.9793089
## 25       amplitude_pitch_belt       19216   0.9793089
## 27       var_total_accel_belt       19216   0.9793089
## 28              avg_roll_belt       19216   0.9793089
## 29           stddev_roll_belt       19216   0.9793089
## 30              var_roll_belt       19216   0.9793089
## 31             avg_pitch_belt       19216   0.9793089
## 32          stddev_pitch_belt       19216   0.9793089
## 33             var_pitch_belt       19216   0.9793089
## 34               avg_yaw_belt       19216   0.9793089
## 35            stddev_yaw_belt       19216   0.9793089
## 36               var_yaw_belt       19216   0.9793089
## 50              var_accel_arm       19216   0.9793089
## 51               avg_roll_arm       19216   0.9793089
## 52            stddev_roll_arm       19216   0.9793089
## 53               var_roll_arm       19216   0.9793089
## 54              avg_pitch_arm       19216   0.9793089
## 55           stddev_pitch_arm       19216   0.9793089
## 56              var_pitch_arm       19216   0.9793089
## 57                avg_yaw_arm       19216   0.9793089
## 58             stddev_yaw_arm       19216   0.9793089
## 59                var_yaw_arm       19216   0.9793089
## 75               max_roll_arm       19216   0.9793089
## 76              max_picth_arm       19216   0.9793089
## 77                max_yaw_arm       19216   0.9793089
## 78               min_roll_arm       19216   0.9793089
## 79              min_pitch_arm       19216   0.9793089
## 80                min_yaw_arm       19216   0.9793089
## 81         amplitude_roll_arm       19216   0.9793089
## 82        amplitude_pitch_arm       19216   0.9793089
## 83          amplitude_yaw_arm       19216   0.9793089
## 93          max_roll_dumbbell       19216   0.9793089
## 94         max_picth_dumbbell       19216   0.9793089
## 96          min_roll_dumbbell       19216   0.9793089
## 97         min_pitch_dumbbell       19216   0.9793089
## 99    amplitude_roll_dumbbell       19216   0.9793089
## 100 amplitude_pitch_dumbbell       19216   0.9793089
## 103        var_accel_dumbbell       19216   0.9793089
## 104         avg_roll_dumbbell       19216   0.9793089
## 105      stddev_roll_dumbbell       19216   0.9793089
```

```
## 106        var_roll_dumbbell       19216   0.9793089
## 107       avg_pitch_dumbbell       19216   0.9793089
## 108    stddev_pitch_dumbbell       19216   0.9793089
## 109       var_pitch_dumbbell       19216   0.9793089
## 110         avg_yaw_dumbbell       19216   0.9793089
## 111      stddev_yaw_dumbbell       19216   0.9793089
## 112         var_yaw_dumbbell       19216   0.9793089
## 131         max_roll_forearm       19216   0.9793089
## 132        max_picth_forearm       19216   0.9793089
## 134         min_roll_forearm       19216   0.9793089
## 135        min_pitch_forearm       19216   0.9793089
## 137   amplitude_roll_forearm       19216   0.9793089
## 138  amplitude_pitch_forearm       19216   0.9793089
## 141        var_accel_forearm       19216   0.9793089
## 142         avg_roll_forearm       19216   0.9793089
## 143      stddev_roll_forearm       19216   0.9793089
## 144         var_roll_forearm       19216   0.9793089
## 145        avg_pitch_forearm       19216   0.9793089
## 146     stddev_pitch_forearm       19216   0.9793089
## 147        var_pitch_forearm       19216   0.9793089
## 148          avg_yaw_forearm       19216   0.9793089
## 149       stddev_yaw_forearm       19216   0.9793089
## 150          var_yaw_forearm       19216   0.9793089
```

Remove any columns that are empty or are NA

```
nonempty<- data.frame( apply(rawtraining, 2,  function(c) { any(is.na(c)) | any(c==
'')}) )
nonempty$fieldname<- row.names(nonempty);row.names(nonempty)<- NULL
names(nonempty)[1]<- "FLAG"
#list of non empty columns in a dataset
nonEmptyTraining<- rawtraining[, colnames(rawtraining) %in% nonempty[!nonempty$FLA
G,]$fieldname ]
```

We are now left with 60 variables only

However, looking at the documentation, we can infer that only the acceleration, gyroscope and magnet readings are fundamental. We use only those variables to build our models. We will end up with 36 independant and the one dependant variable.

We completey ignored any fields with a missing value This makes sense as they are bad data Additionally, we do have none of the data missing for the acceleration, gyroscope and magnet measurements

```
xyz<- grep("^acc.+|^gyr.+|^mag.+", names(nonEmptyTraining), value=T)
trainingset<- nonEmptyTraining[, names(nonEmptyTraining) %in% xyz | names(nonEmptyTr
aining)=="classe" ]
```

Now we will treat this ‘trainingset’ data as the training data and cross validation data. Note that cross validation data is named testing in the exercises that follow.

Perform Multinomial logistic regression first

```
sample<-createDataPartition(trainingset$classe, p=0.5, list=FALSE)
training<- trainingset[sample,]
testing<- trainingset[-sample,]
modLR<- train( classe~. , data=training , method="multinom", trace=FALSE, trControl=
trainControl(method="cv", number=5) , preProcess = c("center", "scale"))

#Prediction part
predLR<- predict(modLR, newdata= testing )
tableLR<-table(predLR, testing$classe)
tableLR
```

```
##
## predLR    A    B    C    D    E
##      A 2262  300  262  145  139
##      B  137 1140  121  100  298
##      C  184  192 1114  209  133
##      D  153   80  152  972  225
##      E   54  186   62  182 1008
```

```
sum(diag(tableLR))/ sum(tableLR) #accuracy
```

```
## [1] 0.6621814
```

Perform random forest with 25% of the data. More data could not be put into the training set due to memory constraints.

For 10% first

```
sample<-createDataPartition(trainingset$classe, p=0.1, list=FALSE)
training<- trainingset[sample,]
testing<- trainingset[-sample,]
 modRf<- train( classe~ . , data=training , method="rf")
 predRf<- predict(modRf, newdata= testing )
 tableRf<-table(predRf, testing$classe)
 tableRf
```

```
##
## predRf    A    B    C    D    E
##      A 4884  216   49   88   24
##      B   32 3037  145   14  145
##      C   43  159 2862  237   81
##      D   51    2   15 2505   52
##      E   12    3    8   50 2944
```

```
 sum(diag(tableRf))/ sum(tableRf) #accuracy
```

```
## [1] 0.9192434
```

For 25%

```
sample<-createDataPartition(trainingset$classe, p=0.25, list=FALSE)
training<- trainingset[sample,]
testing<- trainingset[-sample,]
 modRf<- train( classe~ . , data=training , method="rf")
 predRf<- predict(modRf, newdata= testing )
 tableRf<-table(predRf, testing$classe)
 tableRf
```

```
##
## predRf    A    B    C    D    E
##      A 4115  112    3   22   12
##      B    8 2637   89    2    8
##      C   22   89 2472  136   35
##      D   40    5    2 2242   26
##      E    0    4    0   10 2624
```

```
 sum(diag(tableRf))/ sum(tableRf) #accuracy
```

```
## [1] 0.9575263
```

Boosting could be done only for 5% of the data provided

```
 sample<-createDataPartition(trainingset$classe, p=0.05, list=FALSE)
training<- trainingset[sample,]
testing<- trainingset[-sample,]
 modBo<- train(classe~., method="gbm", data=training, verbose=F)
 predBo<- predict( modBo, newdata=testing)
 tableBo<-table(predBo , testing$classe)
 tableBo
```

```
##
## predBo    A    B    C    D    E
##      A 5010  368  170  182   62
##      B   64 2674  263   44  261
##      C   78  297 2615  245  146
##      D  113   95  180 2303   86
##      E   36  173   22  281 2871
```

```
 sum(diag(tableBo ))/ sum(tableBo )
```

```
## [1] 0.8301411
```

Conclusion:

Random Forest gives the most accurate form of Prediction in our case. We used just the fundamental measurement varibales to build our model. It gives above 95% accuracy in the cross validation set. We will check its accuracy with the 20 test cases as well.