Practical Machine Learning - Course Project

Sambasiva Andaluri

September 20, 2015

Predicting manner of exercise

Introduction

Personal activity data of an activity from various devices such as JawBone Up, FitBit and FuelBand was collected for several users. This data is now being used for making prediction of the activity based on the numeric data. In this report we will build a machine learning model to make predictions of activity performed using the data. We will explore multiple models and select a model with highest accuracy as the final model.

Load libraries

```
library(caret)
library(dplyr)
library(rpart)
```

Gather data

Download the data once to save time and bandwidth of downloading each time.

```
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",
destfile="pml-training.csv", method="curl", quiet=TRUE)
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",
destfile="pml-testing.csv", method="curl", quiet=TRUE)
```

Data Cleaning

Taking a cursory look at the data using str command, it looks like many columns have a lot of values as NA, #DIV/0! or empty values. While reading data, these values are registered as NA for easy removal.

```
pmlTrainRaw <- read.csv("pml-training.csv", na.strings = c("NA","#DIV/0!", ""))
pmlTestRaw <- read.csv("pml-testing.csv", na.strings = c("NA","#DIV/0!", ""))
```

Many of the columns have a lot of NAs. Some columns have all NAs or have a significant number (>97%) values are NA. These columns will be removed to make tidier data set. We would need to save rmNA to remove the same columns from test data set as well.

```
as.data.frame(sapply(pmlTrainRaw, function(x) mean(is.na(x))))
rmNA <- sapply(pmlTrainRaw, function(x) mean(is.na(x))) > 0.97
pmlTrainFilt <- pmlTrainRaw[,rmNA==FALSE]
pmlTestFilt <- pmlTestRaw[,rmNA==FALSE]
```

We now have 60 variables after initial cleaning. Out of which the first 7 columns does not seem to be of value: X (id), user_name, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp, new_window, num_window. We will remove these columns, and now we should have 53 variables.

```
pmlTrainClean <- select(pmlTrainFilt, roll_belt:classe)
pmlTestClean <- select(pmlTestFilt, roll_belt:problem_id)
```

In the interest of finding a parsimonious model, we should remove redundant features by finding and removing highly correlated variables with a correlation of .75 or more. This should leave us with 32 variables.

```
corTrain <- cor(pmlTrainClean[1:52])
highCorCols <- findCorrelation(corTrain, cutoff = .75)
pmlTrainFinal <- pmlTrainClean[,-highCorCols]
pmlTestFinal <- pmlTestClean[,-highCorCols]
```

We should now test the data for Near Zero Variance and remove any near zero covariates. Output shows there is no NearZeroVariance variables, though many variables have very low percentUnique.

```
nearZeroVar(pmlTrainFinal, saveMetrics=TRUE)
```

```
##                      freqRatio percentUnique zeroVar   nzv
## yaw_belt              1.058480     9.9734991   FALSE FALSE
## gyros_belt_x          1.058651     0.7134849   FALSE FALSE
## gyros_belt_y          1.144000     0.3516461   FALSE FALSE
## gyros_belt_z          1.066214     0.8612782   FALSE FALSE
## magnet_belt_x         1.090141     1.6664968   FALSE FALSE
## magnet_belt_y         1.099688     1.5187035   FALSE FALSE
## roll_arm             52.338462    13.5256345   FALSE FALSE
## pitch_arm            87.256410    15.7323412   FALSE FALSE
## yaw_arm              33.029126    14.6570176   FALSE FALSE
## total_accel_arm       1.024526     0.3363572   FALSE FALSE
## gyros_arm_y           1.454369     1.9162165   FALSE FALSE
## gyros_arm_z           1.110687     1.2638875   FALSE FALSE
## magnet_arm_x          1.000000     6.8239731   FALSE FALSE
## magnet_arm_z          1.036364     6.4468454   FALSE FALSE
## roll_dumbbell         1.022388    84.2065029   FALSE FALSE
## pitch_dumbbell        2.277372    81.7449801   FALSE FALSE
## yaw_dumbbell          1.132231    83.4828254   FALSE FALSE
## total_accel_dumbbell  1.072634     0.2191418   FALSE FALSE
## gyros_dumbbell_y      1.264957     1.4167771   FALSE FALSE
## magnet_dumbbell_z     1.020833     3.4451126   FALSE FALSE
## roll_forearm         11.589286    11.0895933   FALSE FALSE
## pitch_forearm        65.983051    14.8557741   FALSE FALSE
## yaw_forearm          15.322835    10.1467740   FALSE FALSE
## total_accel_forearm   1.128928     0.3567424   FALSE FALSE
## gyros_forearm_x       1.059273     1.5187035   FALSE FALSE
## gyros_forearm_z       1.122917     1.5645704   FALSE FALSE
## accel_forearm_x       1.126437     4.0464784   FALSE FALSE
## accel_forearm_z       1.006250     2.9558659   FALSE FALSE
## magnet_forearm_x      1.012346     7.7667924   FALSE FALSE
## magnet_forearm_y      1.246914     9.5403119   FALSE FALSE
## magnet_forearm_z      1.000000     8.5771073   FALSE FALSE
## classe                1.469581     0.0254816   FALSE FALSE
```

Data Slicing

Though we were given a train and test data sets, we should still split the training data set for testing and evaluating various models before we test our models on the backup test set.

```
pmlInTrain <- createDataPartition(y=pmlTrainFinal$classe,p=0.75, list=FALSE)
pmlTrain <- pmlTrainFinal[pmlInTrain,]
pmlTest <- pmlTrainFinal[-pmlInTrain,]
```

Build Models

We will try 2 models, first a Classification Tree algorithm and then Random Forest algorithm each with a preProces option to center and scale data.

Classification Tree

First lets start with a simple classification tree as we need to use numeric data to predict a factor outcome. However the accuracy of the model is about 0.53 which is no different from a coin toss. So we should discard this model and try Random Forest model.

```
fitCT <- train(pmlTrain$classe ~ ., preProcess=c("center", "scale"), data=pmlTrain, m
ethod="rpart")
predCT <- predict(fitCT, newdata=pmlTest)
cmCT <- confusionMatrix(predCT, pmlTest$classe)
cmCT$overall['Accuracy']
```

```
##  Accuracy
## 0.5254894
```

Random Forest

Since our earlier model did not yield acceptable result, we will try the Random Forest model with 4 fold Cross validation as tuning parameter. With RF model, the accuracy increased to 0.99. This is acceptable for making further predictions on the test data provided for assignment.

```
fitRF <- train(pmlTrain$classe ~ ., preProcess=c("center", "scale"), data=pmlTrain, m
ethod="rf", trControl=trainControl(method = "cv", number = 4))
predRF <- predict(fitRF, newdata=pmlTest)
cmRF <- confusionMatrix(predRF, pmlTest$classe)
cmRF$overall['Accuracy']
```

```
##  Accuracy
## 0.9900082
```

Sample Errors

The in sample error rate for Random Forest was 0.01. Out of sample error is 0.0099918

Conclusion

In conclusion after evaluating two models we found that the random forest model yields the most accurate predictions. Using this model, we were able to predict the activity from the given data with an error rate of 1%. In sample and out of sample error rates are pretty much same, as such there is no indication of overfitting.

Predict held test data (for submission)

```
answers = predict(fitRF, newdata=pmlTestFinal)
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
```

```
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)
  }
}
pml_write_files(answers)
```