JHU Machine Learning Exercise

Jake Ackman

3/14/2020

Background

Coursera Description

"Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. The goal of this project is to use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants as they perform barbell lifts correctly and incorrectly 5 different ways."

We have been provided with training data to build a predictive model and use the model to measure accuracy and identify the apporpriate type of barbell lift given a set of data.

The first step when receiving data is the clean it and ensure the it is prepared to be part of a preditictive model

Loading and Cleaning the Personal Activity Data

First lets download the data, import it into R, and remove any uncessary columns. We have two versions of the data - the training data that we'll build the model off of, and the test data where we'll try to predict the outcome variable and test our model.

Then we remove all the unnecessary columns by counting the number of NAs, and removing columns with 19216 NAs. We also changed the classe variable to a factor variable. Both the train and test data had the aforementioned modifications.

After that, let's partition the training data into two datasets so we have more data to validate the model.

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'plotly' was built under R version 3.5.3
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

## Warning: package 'e1071' was built under R version 3.5.3
```

```
train <- download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile

test <- download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile =

train_dat <- read.csv("C:/Users/jackman/Desktop/R Files/train.csv")

test_dat <- read.csv("C:/Users/jackman/Desktop/R Files/test.csv")


#set seed#

set.seed(33)

#Remove Near Zero Variance Variables#

nzvcol <- nearZeroVar(train_dat)

train_dat <- train_dat[, -nzvcol]

test_dat <- test_dat[, -nzvcol]

#Remove columns that are all NA#

na_sum <- function(x){sum(is.na(x))}

col_na <- sapply(train_dat, na_sum)

col_na_filter <- col_na == 19216

train_dat <- train_dat[,col_na_filter == FALSE]

test_dat <- test_dat[,col_na_filter == FALSE]

#Removing First 5 Variables as Not Relevant To Analysis#

train_dat <- train_dat[,-c(1:5)]

test_dat <- test_dat[,-c(1:5)]
```

```
#Set Classe Variable to Factor#

train_dat$classe <- as.factor(train_dat$classe)

#Create Data Partition#

train_partition <- createDataPartition(train_dat$classe, p=.70, list = F)

train_initial <- train_dat[train_partition,]

train_validate <- train_dat[-train_partition,]
```

Develop Models

Our approach is going to be to create two separate models and identify which once has the highest accuracy rates. The two model types we will use for this exercise are: GBM (Gradient Boosting Model) and Random Forest. First let's start by creating the respective model objects in R.

```
#mod_gbm <- train(classe ~ ., data=train_initial, method = "gbm")#


mod_rf <- train(classe ~ ., data=train_initial, method = "rf")
```

#GBM Model Validation

Now it's time to see the rate of accuracy for the GBM model. First we create a predict object using the GBM model and the training validation data we partitioned earlier. Once we have the predict object, then we can run a confusion matrix to compare our model's predictions with the actual classe values of the training validation data.

PLEASE NOTE: Running both models the GBM and Random Forest model kept on breaking RMarkdown every time I tried to run it. Therefore, to submit this assignment I had to comment out the GBM model and all downstream objects.

```
#Predict GBM model on validation data#

#pred_gbm <- predict(mod_gbm, train_validate)#

#GBM Model Confusion Matrix#

#cm_gbm <- confusionMatrix(pred_gbm,train_validate$classe)#
```

Wow, this model has an accuracy of 98.7%! That's a very accurate model so this could be a good option. But let's also see what a Random Forest model comes up with for accuracy.

#Random Forest Model Validation

Now it's time to see the rate of accuracy for the Random Forest model. First we create a predict object using the Random Forest model and the training validation data we partitioned earlier. Once we have the predict object, then we can run a confusion matrix to compare our model's predictions with the actual classe values of the training validation data.

```r
#Predict Random Forest model on validation data#

pred_rf <- predict(mod_rf, train_validate)

#Random Forest Confusion Matrix#

cm_rf <- confusionMatrix(pred_rf,train_validate$classe)

print(cm_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    3    0    0    0
##          B    1 1136    1    0    0
##          C    0    0 1025    8    0
##          D    0    0    0  956    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9962, 0.9988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9972
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9974   0.9990   0.9917   1.0000
## Specificity            0.9993   0.9996   0.9984   1.0000   1.0000
## Pos Pred Value         0.9982   0.9982   0.9923   1.0000   1.0000
## Neg Pred Value         0.9998   0.9994   0.9998   0.9984   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1930   0.1742   0.1624   0.1839
## Detection Prevalence   0.2848   0.1934   0.1755   0.1624   0.1839
## Balanced Accuracy      0.9993   0.9985   0.9987   0.9959   1.0000
```

Our Random Forest model had an accuracy rate of 99.7%! This is even better than the GBM model so we have ourselves a winner.

The Random Forest model will be used to predict against the test data.

##Prediction on 20 Test cases

Finally, we'll apply our model to the test cases to once again try to measure accuracy. Let's see how well this model fairs against another set of data by predicting the classe variable and submitting it to our class. After taking the quiz with the predictions below, they're 100% accurate.

```r
model_prediction_test <- predict(mod_rf, test_dat)

model_prediction_test
```

```
##  [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
```