Practical Machine Learning - Course Project: Fitness Data

Anonymous

4 September 2017

Dataset & Aim

The dataset for our project can be found from the source here. The training & testing data can be directly found here & here respectively.

The data comes from volunteers performing exercises whilst wearing fitness tracking devices. These devices generate data (such as from the accelerometers). THey are worn on the belt, forearms,arm and dumbell of 6 participants. Volunteers were prompted to perform barbell lifts correctly and incorrectly in 5 different ways.

The goal of the project is to predict the manner in which the 6 volunteers did the exercise. This is the "classe" variable in the training set. To do this we we will look at any of the other variables to build our prediction model.

Load Packages

```
library(randomForest)
library(caret)
library(rpart)
library(rattle)
library(e1071)
```

Load Dataset

We will be loading the two datasets directly from the url's provided in the project brief.

```
#set the seed so the following code will be reproducible.
set.seed(12345)

#Load in training and test data
training.data <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"),na
testing.data <- read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"),na.s

#Take a look at the structure of training data.
str(training.data)
```

We can see from the structure of our data that we have many columns with NA's as well as a few columns at the beginning of the dataset that will not aid in the prediction of the 'classe' variable. We will set about removing these in the next step.

Data Cleaning

```r
#First let's remove the columns that will not aid in our prediction
colremove <- c("X", "user_name", "raw_timestamp_part_1", "raw_timestamp_part_2",
                "cvtd_timestamp", "new_window", "num_window")
#These also happen to be our first 7 columns so we can remove in a much easier way
training.data<-training.data[,-c(1:7)]

#Next we want to remove any columns with all missing values/NA values.
training.data <- training.data[, colSums(is.na(training.data)) == 0]

#What we have left is 53 columns of data left for training.
dim(training.data)
```

```
## [1] 19622    53
```

The above process will now be applied to the testing data set as well.

```r
testing.data<-testing.data[,-c(1:7)]

testing.data <- testing.data[, colSums(is.na(testing.data)) == 0]
#What we have left is 53 columns of data left for testing.
dim(testing.data)
```

```
## [1] 20 53
```

Data Splitting

We will further split our training data into the training set and validation set. We will put 70% in our training set and 30% into our validation set.

```r
split<-createDataPartition(y=training.data$classe, p=0.7, list=F)
train.set<-training.data[split,]
validate.set<-training.data[-split,]
```

Build Model(s)

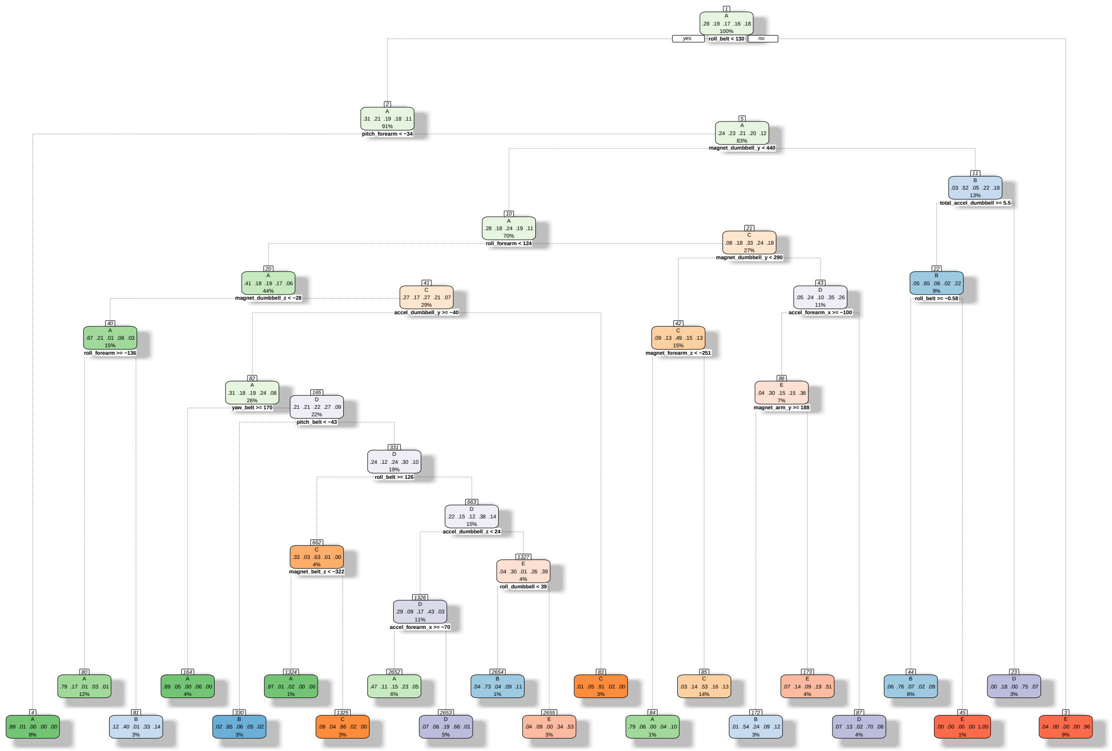
I will approach this with three model building methods.

- Classification Trees
- Random Forest
- Support Vector Machine

Classification Trees

```r
# Fit the model
ct.mod<-rpart(classe ~., data=train.set,method="class")

#Take a look at the tree diagram
fancyRpartPlot(ct.mod)
```



Rattle 2017−Sep−06 22:50:07 Nicholas

```
#Predict Output
prediction.ct<-predict(ct.mod,validate.set,type="class")

#Produce Confusion Matrix
confmatrix<-confusionMatrix(prediction.ct,validate.set$classe)

#Get overrall model accuracy
ct.accuracy<-confmatrix$overall['Accuracy']

#Get expected out-of-sample error
ct.error<-1-ct.accuracy
```

Random Forest

```
#Fit the model
rf.mod<-randomForest(classe ~., data=train.set)

#Predict Output
prediction.rf<-predict(rf.mod,validate.set,type="class")

#Produce Confusion Matrix
confmatrix<-confusionMatrix(prediction.rf,validate.set$classe)

#Get overrall model accuracy
rf.accuracy<-confmatrix$overall['Accuracy']
```

```r
#Get expected out-of-sample error
rf.error<-1-rf.accuracy
```

Support Vector Machine (SVM)

```r
#Fit the model
svm.mod<-svm(classe ~., data=train.set)

#Predict Output
prediction.svm<-predict(svm.mod,validate.set,type="class")

#Produce Confusion Matrix
confmatrix<-confusionMatrix(prediction.svm,validate.set$classe)

#Get overrall model accuracy
svm.accuracy<-confmatrix$overall['Accuracy']

#Get expected out-of-sample error
svm.error<-1-svm.accuracy
```

Training Results

Let's now take a look at all of our Accuracy and out-of-sample errors.

```
##                     Classification.Trees Random.Forest        SVM
## Model Accuracy                 0.7220051    0.99184367 0.93746814
## Out-of-sample-error            0.2779949    0.00815633 0.06253186
```

We can see our best result was the Random Forest Model with a model accuracy of 0.99184. However the SVM scored well with 0.94732 so I am interested to use this as well and compare on our test data set.

Retrain Models

We can now combine our training & validation data to wholistically retrain our model before we jump over to the test dataset. This is useful as we have more data to train on, so we would expect to get a better result (Assuming overtraining doesn't occur)

```r
#Retrain rf model
rf.mod<-randomForest(classe ~., data=training.data)

#Retrain svm model
svm.mod<-svm(classe ~., data=training.data)
```

Predicting Test Data

After all our hard work it's the moment of truth. I will be running the test data through the random forest model to generate the predicted values for our exercises.

```r
prediction.rf<-predict(rf.mod,testing.data,type="class")

# Use the output to predict the exercises in answer to the Course Project Quiz.
```