Practical Machine Learning Project

Deepak Trivedi

February 7, 2016

```
#install.packages("caret")
require(caret)
#install.packages("rpart")
require(rpart)
#install.packages("rpart.plot")
require(rpart.plot)
#install.packages("e1071")
require(e1071)
#install.packages("randomForest")
require(randomForest)
```

Initialize analysis and clean up data

Set seed for the project

```
set.seed(1984)
```

Load data into R

```
# Loading the training data set into my R session replacing all missing with "NA"
trnset <- read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-trainin
g.csv", na.strings=c("","NA","#DIV/0!"))

# Loading the testing data set
tstset <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing
.csv', na.strings=c("","NA","#DIV/0!"))
```

Look at all the column names. I have not echoed it here because the list is long.

```
names(trnset)
```

We need to remove certain columns from the data set, like time, serial number etc.

```
trnset   <-trnset[,-c(1:7)]
tstset <-tstset[,-c(1:7)]
```

Next, we remove columns with no data, since these will not help with our predictions in any way.

```
trnset<-trnset[,colSums(is.na(trnset)) == 0]
tstset <-tstset[,colSums(is.na(tstset)) == 0]
```

This leaves us with 53 columns. The algorithm needs to predict the output variable classe, which describes the manner in which the dumbbell was lifted: (Class A), throwing the elbows to the front (Class B), lifting the dumbbell only halfway (Class C), lowering the dumbbell only halfway (Class D) and throwing the hips to the front (Class E).

Using the trnset, let's make a cross-validation set.

```
ssindex <- createDataPartition(y=trnset$classe, p=0.80, list=FALSE)
trnsub <- trnset[ssindex, ]
tstsub <- trnset[-ssindex, ]
```

Looking at the heads of the training and test set. Not echoing here since it is long.

```
head(trnsub)
head(tstsub)
```

Check the distribution of data in the various sets

```
summary(trnset$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
summary(tstsub$classe)
```

```
##    A    B    C    D    E
## 1116  759  684  643  721
```

```
summary(trnsub$classe)
```

```
##    A    B    C    D    E
## 4464 3038 2738 2573 2886
```

In all three datasets, all the five classes (A,B,C,D,E) are fairly represented.

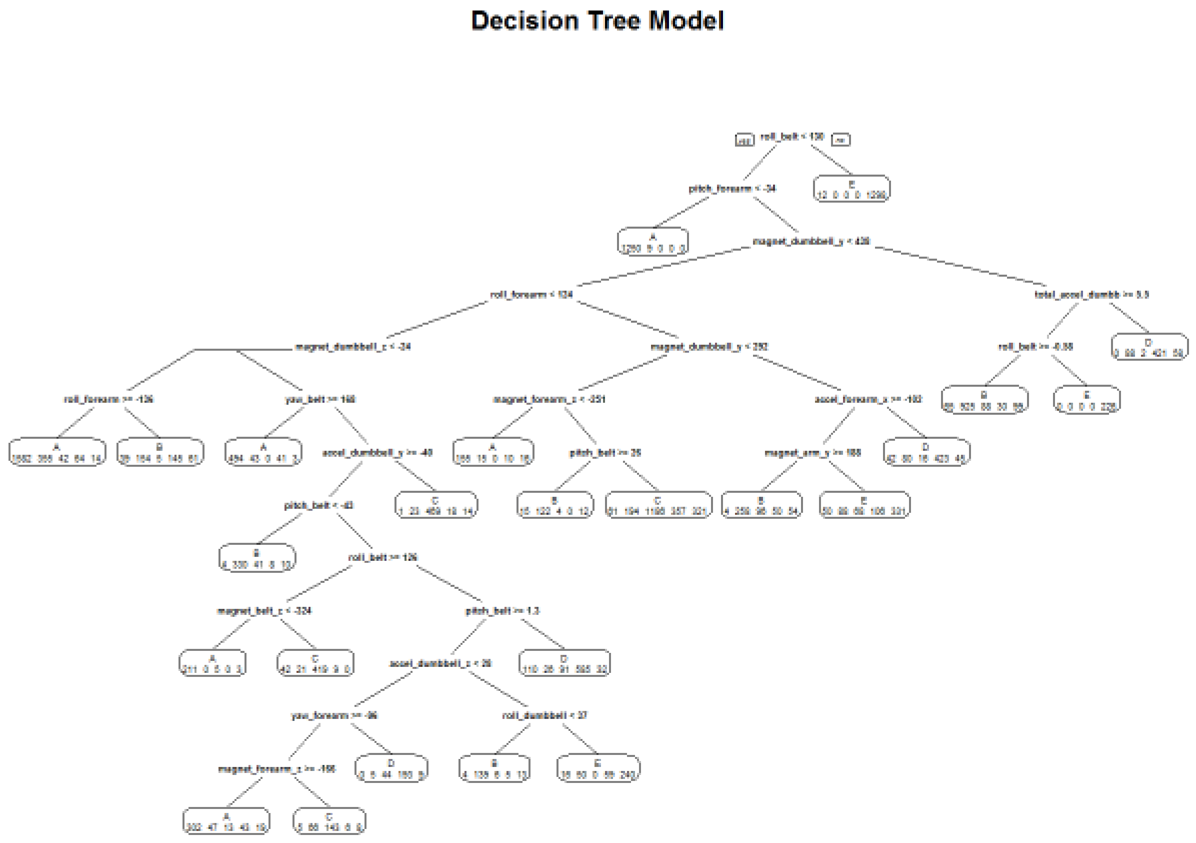
Decision Tree Model

This is the first model to try. We develop a decision tree and look at its performance.

```
dtreemodel <- rpart(classe ~ . , data=trnsub, method="class")
dtreeprediction <- predict(dtreemodel, tstsub, type = "class")
```

Plot the outcome of this model:

```
rpart.plot(dtreemodel, main="Decision Tree Model",extra=1)
```



How good is this model? Let’s look at the confusion matrix:

```
cm1<-confusionMatrix(dtreeprediction, tstsub$classe)
cm1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 990 120  16  35   8
##          B  45 478  65  48  67
##          C  28  66 540  94 109
##          D  37  52  44 415  39
##          E  16  43  19  51 498
##
## Overall Statistics
##
##                Accuracy : 0.7446
##                  95% CI : (0.7306, 0.7582)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6765
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8871   0.6298   0.7895   0.6454   0.6907
## Specificity            0.9362   0.9289   0.9083   0.9476   0.9597
## Pos Pred Value         0.8469   0.6799   0.6452   0.7070   0.7943
## Neg Pred Value         0.9542   0.9127   0.9533   0.9317   0.9323
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2524   0.1218   0.1376   0.1058   0.1269
## Detection Prevalence   0.2980   0.1792   0.2134   0.1496   0.1598
## Balanced Accuracy      0.9117   0.7793   0.8489   0.7965   0.8252
```

Random forest model

The decision tree model doesn’t look very accurate. Next we try a random forest model.

```
rfmodel <- randomForest(classe ~ . , data=trnsub, method="class")
rfprediction <- predict(rfmodel, tstsub, type = "class")
```

How good is this model? Let’s look at the confusion matrix

```
cm2<-confusionMatrix(rfprediction, tstsub$classe)
cm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1114    3    0    0    0
##          B    2  756    4    0    0
##          C    0    0  680    9    0
##          D    0    0    0  632    0
##          E    0    0    0    2  721
##
## Overall Statistics
##
##                Accuracy : 0.9949
##                  95% CI : (0.9921, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9960   0.9942   0.9829   1.0000
## Specificity            0.9989   0.9981   0.9972   1.0000   0.9994
## Pos Pred Value         0.9973   0.9921   0.9869   1.0000   0.9972
## Neg Pred Value         0.9993   0.9991   0.9988   0.9967   1.0000
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2840   0.1927   0.1733   0.1611   0.1838
## Detection Prevalence   0.2847   0.1942   0.1756   0.1611   0.1843
## Balanced Accuracy      0.9986   0.9971   0.9957   0.9914   0.9997
```

Random forest model (Accuracy ~ `0.9949019`) is much better than the decision tree model (Accuracy ~ `0.7445832` .)

Expected out of sample error for the random forest model is `0.0050981` . This looks pretty good. So, we will use this model to make predictions on the test data.

Predictions on test data

Use the predict function with this model and then look at the answers.

```
assignment_answer <- predict(rfmodel, tstset, type="class")
assignment_answer
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  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
```

Conclusion

For this type of data, the random forest model seems to be better at predicting the outcome. The predictions listed above were used to answer the quiz questions, and all responses were found to be correct!