Practical Machine Learning - Final Project

The goal of this project is to predict the manner in which barbell lefts will be executed. This is the “classe” variable in the training set, based on the collection of data from devices such as Jawbone UP, Nike Fuel Band, and FitBit. ## Load Packages

```
suppressMessages(library(caret))
suppressMessages(library(rpart))
suppressMessages(library(rpart.plot))
suppressMessages(library(rattle))
suppressMessages(library(randomForest))
set.seed(6824)
```

Load and Cleanse Data

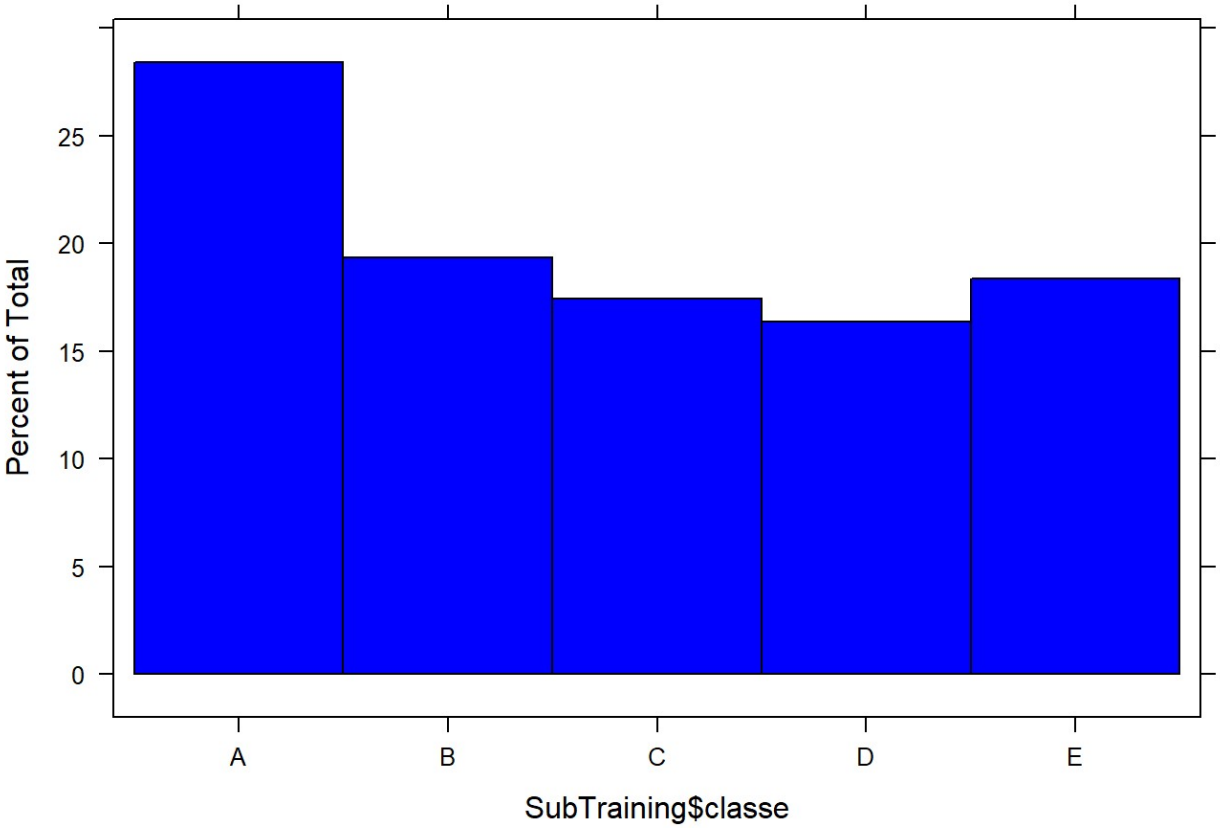
Note, in order to cut down on report length, the data columns and formats were analyzed offline. The tactical cleansing steps are included in the following r code.

```
# Load data
Training <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-train
ing.csv"), na.strings=c("NA","#DIV/0!",""))
Testing <- read.csv(url("http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testin
g.csv"), na.strings=c("NA","#DIV/0!",""))
# remove Columns that will not add value in analysis: columns with no data, overlappin
g columns, non-predective columnns, etc.
Testing = Testing[,colSums(is.na(Training))==0]
Training = Training[,colSums(is.na(Training))==0]
Testing = Testing[,-nearZeroVar(Training)]
Training = Training[,-nearZeroVar(Training)]
Testing = Testing[,-c(1:5)]
Training = Training[,-c(1:5)]
# Particition the data sets
inSubTraining = createDataPartition(y=Training$classe, p=0.6, list=FALSE)
SubTraining = Training[inSubTraining,]
SubTesting = Training[-inSubTraining,]
```

Modeling

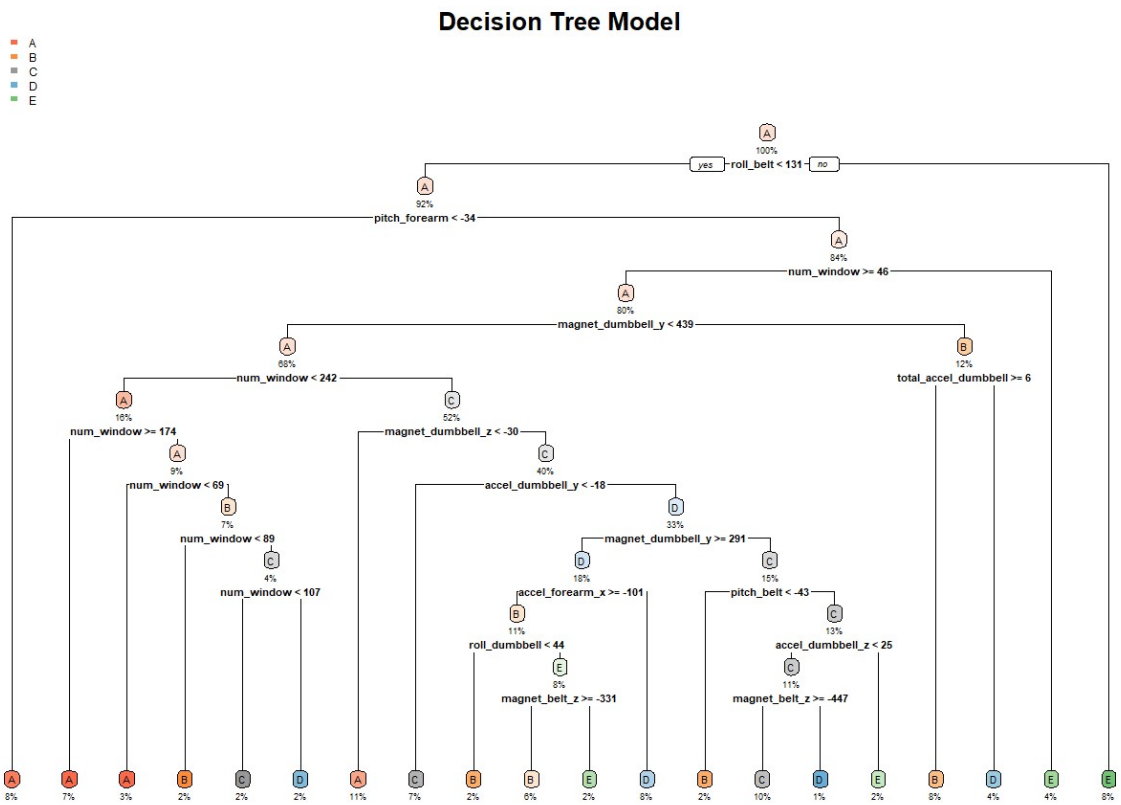
The variable we are predicting is the classe variable. Here is a quick look at the values and distribution:

```
histogram(SubTraining$classe, col="blue")
```



For modeling, we will assess two models: Decision Tree and Random Forest and analyze the fit of each based on confusion matrix results. First, Decision Tree:

```
DTmodel = rpart(classe ~ ., data=SubTraining, method="class")
DTpredict = predict(DTmodel, SubTesting, type="class")
rpart.plot(DTmodel, main="Decision Tree Model", under=TRUE, extra=100)
```




```
confusionMatrix(DTpredict, SubTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1937  177   59   60    9
##          B  146 1023  180  143  163
##          C   42  130 1087  171   52
##          D   89  144   23  779   93
##          E   18   44   19  133 1125
##
## Overall Statistics
##
##                Accuracy : 0.7585
##                  95% CI : (0.7488, 0.7679)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6943
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8678   0.6739   0.7946  0.60575   0.7802
## Specificity            0.9457   0.9001   0.9390  0.94680   0.9666
## Pos Pred Value         0.8640   0.6181   0.7335  0.69060   0.8402
## Neg Pred Value         0.9474   0.9200   0.9558  0.92453   0.9513
## Prevalence             0.2845   0.1935   0.1744  0.16391   0.1838
## Detection Rate         0.2469   0.1304   0.1385  0.09929   0.1434
## Detection Prevalence   0.2858   0.2109   0.1889  0.14377   0.1707
## Balanced Accuracy      0.9068   0.7870   0.8668  0.77628   0.8734
```

Next, Random Forest:

```
RFmodel = randomForest(classe ~ ., data=SubTraining, method="class")
RFpredict = predict(RFmodel, SubTesting, type="class")
confusionMatrix(RFpredict, SubTesting$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    8    0    0    0
##          B    0 1509   10    0    0
##          C    0    1 1357    8    0
##          D    0    0    1 1278    3
##          E    0    0    0    0 1439
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.9944, 0.9973)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9941   0.9920   0.9938   0.9979
## Specificity            0.9986   0.9984   0.9986   0.9994   1.0000
## Pos Pred Value         0.9964   0.9934   0.9934   0.9969   1.0000
## Neg Pred Value         1.0000   0.9986   0.9983   0.9988   0.9995
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1923   0.1730   0.1629   0.1834
## Detection Prevalence   0.2855   0.1936   0.1741   0.1634   0.1834
## Balanced Accuracy      0.9993   0.9962   0.9953   0.9966   0.9990
```

Conclusion

Based on this analysis, we can see that the Random Forest model yields a significantly higher prediction accuracy, 99.6% for Random Forest versus 75.8% for Decision Tree. The out of sample error for the Random Forest model is .004%. Based on this cross-validation test, we will use the Random Forest model to predict the 20 test cases for this assignment, with the expectation that the prediction will yield few or no misclassifications.

Predictions for Test Cases

```
predict(RFmodel, Testing, type="class")
```

```
##  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
```