Summary

I am building a model to predict the manner in which the subjects did the exercise based on various data elements from accelerometers.

```
library(caret)
```

Data Prep

First I will prepare the data. I will read in the data, remove the first 6 columns which are not data columns, and then remove any column that has NA values. I will then match the testing dataset columns to the cleaned training dataset columns, excluding the final column as that is the column that we are trying to predict and is not present in the testing data.

```
training<-read.csv("pml-training.csv",na.strings = c("NA","#DIV/0!"),row.names = 1)
testing<-read.csv("pml-testing.csv",na.strings = c("NA","#DIV/0!"),row.names = 1)
training<-training[,-(1:6)]
training<-training[,colSums(is.na(training))==0]
testing<-subset(testing,select = names(training)[1:52])
```

Model Prep

Next I will do some preparations for modeling. I will take the cleaned training data and divide it into a train and test component, and I will also setup my cross validation parameter to use 5-fold cross validation.

```
set.seed(123321)
intrain<-createDataPartition(y=training$classe, p=.7, list=FALSE)
traindata<-training[intrain,]
testdata<-training[-intrain,]
cvRF<-trainControl(method="cv",5)
```

Tree Model

I first build a tree model. The results of this model are not very good as the estimated out of sample accuracy is only .4895.

```
modelTree<-train(classe~., data=traindata, method="rpart", trControl=cvRF)
predictTree<-predict(modelTree, testdata)
confusionMatrix(predictTree, factor(testdata$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1517  456  501  451  139
##          B   30  385   35  176  146
##          C  121  298  490  337  308
##          D    0    0    0    0    0
##          E    6    0    0    0  489
## 
## Overall Statistics
##                                           
##                Accuracy : 0.4895          
##                  95% CI : (0.4767, 0.5024)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.3328          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9062  0.33802  0.47758   0.0000  0.45194
## Specificity            0.6326  0.91846  0.78102   1.0000  0.99875
## Pos Pred Value         0.4951  0.49870  0.31532      NaN  0.98788
## Neg Pred Value         0.9443  0.85253  0.87624   0.8362  0.88998
## Prevalence             0.2845  0.19354  0.17434   0.1638  0.18386
## Detection Rate         0.2578  0.06542  0.08326   0.0000  0.08309
## Detection Prevalence   0.5206  0.13118  0.26406   0.0000  0.08411
## Balanced Accuracy      0.7694  0.62824  0.62930   0.5000  0.72535
```

Random Forest Model

Next I will build a random forest model. The results of this model are much better, with an estimated out of sample accuracy of .9927.

```
modelRF<-train(classe~., data=traindata, method="rf", trControl=cvRF, ntree=200)
predictRF<-predict(modelRF, testdata)
confusionMatrix(predictRF, factor(testdata$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    7    0    0    0
##          B    2 1125    3    0    1
##          C    0    7 1019    7    3
##          D    0    0    4  956    6
##          E    2    0    0    1 1072
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9927          
##                  95% CI : (0.9902, 0.9947)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9908          
##                                           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9877   0.9932   0.9917   0.9908
## Specificity            0.9983   0.9987   0.9965   0.9980   0.9994
## Pos Pred Value         0.9958   0.9947   0.9836   0.9896   0.9972
## Neg Pred Value         0.9990   0.9971   0.9986   0.9984   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2838   0.1912   0.1732   0.1624   0.1822
## Detection Prevalence   0.2850   0.1922   0.1760   0.1641   0.1827
## Balanced Accuracy      0.9980   0.9932   0.9948   0.9948   0.9951
```

Boosting Model

Last I will build a boosting model. The results of this model are pretty good as well with an estimated out of sample accuracy of .9623. Not as good as the random forest but still very good.

```
modelGBM<-train(classe~., data=traindata, method="gbm", trControl=cvRF, verbose=FALSE)
predictGBM<-predict(modelGBM, testdata)
confusionMatrix(predictGBM, factor(testdata$classe))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1640   38    0    0    2
##          B   18 1061   25    4    9
##          C   12   38  992   24   10
##          D    2    0    7  926   17
##          E    2    2    2   10 1044
## 
## Overall Statistics
## 
##                Accuracy : 0.9623
##                  95% CI : (0.9571, 0.967)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9523
## 
##  Mcnemar's Test P-Value : 4.525e-07
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9797   0.9315   0.9669   0.9606   0.9649
## Specificity            0.9905   0.9882   0.9827   0.9947   0.9967
## Pos Pred Value         0.9762   0.9499   0.9219   0.9727   0.9849
## Neg Pred Value         0.9919   0.9836   0.9929   0.9923   0.9921
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2787   0.1803   0.1686   0.1573   0.1774
## Detection Prevalence   0.2855   0.1898   0.1828   0.1618   0.1801
## Balanced Accuracy      0.9851   0.9599   0.9748   0.9776   0.9808
```

Conclusion

The random forest model is best, so we will use that on our data sample that we would like to make predictions on.

```
predictRF_final<-predict(modelRF, testing)
predictRF_final
```

```
##  [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
```