Barbell Lift Prediction

Executive Summary

We would like to predict the manner in which they did the exercise as documented in the 'classe' variable. First we will remove the columns that will not be good predictors. We then subsetted the data into a training and cross validation portion and then constructed a random forest model on the training data. Our model appears to predict with 99% accuracy and less than 1% OOB error.

Model Building

We reduced the initial set of predictors by removing the columns that show little variability (and thus will be poor predictors), columns that are summaries of other columns (marked with avg, var, etc.), and columns that have majority of NAs.

After we reduced the columns we are interested in, we **split the data into a training (70%) and cross validation (30%) portion**. Then we train the training portion of the data using K-fold cross validation with K=10 and the random forest method.

```
# Split the data into a training portion and a cross validation portion
training_part <- createDataPartition(pml.training2$classe, p = 0.7, list = FALSE)
training <- pml.training2[training_part, ]
crossval <- pml.training2[-training_part, ]

# Use K-Fold cross validation with K=10
ctrl <- trainControl(method = "cv", number = 10, allowParallel = TRUE)
set.seed(1234)
modrf <- train(classe ~., data = training, method = "rf", trControl = ctrl)
modrf
```

```
## Random Forest
##
## 13737 samples
##    33 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12363, 12364, 12362, 12364, 12363, 12363, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9909735  0.9885817
##   17    0.9929393  0.9910689
##   33    0.9880619  0.9848989
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 17.
```

```
modrf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 17
##
##         OOB estimate of  error rate: 0.7%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3901    4    1    0    0 0.001280082
## B   19 2629    8    1    1 0.010910459
## C    0   10 2378    8    0 0.007512521
## D    0    2   15 2233    2 0.008436945
## E    0    6    7   12 2500 0.009900990
```

```
# The top 5 variables in terms of importance
head(varImp(modrf)$importance, 5)
```

```
##                 Overall
## roll_belt     100.000000
## pitch_belt     44.412869
## yaw_belt       51.880687
## magnet_belt_z  20.908605
## roll_arm        6.993474
```

The resulting random forest created 500 trees and had an **out of bounds error rate of < 1%**.

Cross Validation

Now we can use the cross validation portion that we kept out of the training set to see how our model does. The cross validation portion had an accuracy rate of > 99%.

```
crossvalpred <- predict(modrf, crossval)
confusionMatrix(table(crossvalpred, crossval$classe))
```

```
## Confusion Matrix and Statistics
##
##
## crossvalpred    A    B    C    D    E
##            A 1673    2    0    0    0
##            B    0 1135    6    1    0
##            C    0    2 1018    6    4
##            D    0    0    2  957    2
##            E    1    0    0    0 1076
##
## Overall Statistics
##
##                Accuracy : 0.9956
```

```
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9965   0.9922   0.9927   0.9945
## Specificity            0.9995   0.9985   0.9975   0.9992   0.9998
## Pos Pred Value         0.9988   0.9939   0.9883   0.9958   0.9991
## Neg Pred Value         0.9998   0.9992   0.9984   0.9986   0.9988
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1929   0.1730   0.1626   0.1828
## Detection Prevalence   0.2846   0.1941   0.1750   0.1633   0.1830
## Balanced Accuracy      0.9995   0.9975   0.9949   0.9960   0.9971
```

Test Set Prediction

Finally we can use the model that we created on a new set of testing data. First we remove the columns that we removed on the training set. Then we predict using our model to determine the 20 predicted classes.

```
# Predict the testing data based on our first model
testpred <- predict(modrf, newdata = pml.testing2)
testpred
```

```
##  [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
```