Prediction Assignment

Our goal is to predict whether exercises were performed 'correctly' or not. Our data consist of the output of devices worn while individuals performed excercises using dumbbells in various manners.

```
initdf = read.csv('../pml-training.csv')
```

A summary of the data shows there are 160 columns. Part of this summary is shown in the Appendix. The column we are most interested in is *classe*, which details the manner in which an exercise was performed. It is a factor variable, taking five different values.

There are several issues with the dataset. First, the summary reveals some large outliers for some of the variables. Investigation shows that a single row is responsible for many of these outliers, so we remove it.

```
traindf = initdf[initdf$gyros_dumbbell_y<30,]
```

The data in some of the columns also appears to be incomplete or not meaningful. In the summary in the Appendix, these columns have 19216 rows that are blank, with the rest having numeric values or "#DIV/0!". Both the difficulty of understanding these columns and the large percentage of identical values mean these will not be useful for prediction, so we remove them

The first 7 columns also do not contain useful features. These include columns such as the row number, the user name, and the time stamp. We remove these columns as well.

In addition, some columns mostly contain missing values, so we remove those columns.

```
bad_features = c('new_window', 'kurtosis_roll_belt', 'kurtosis_picth_belt', 'kurtosis_yaw_belt', 'skewne
not_predictors = c('X', 'user_name', 'raw_timestamp_part_1', 'raw_timestamp_part_2', 'cvtd_timestamp',
traindf = traindf[,!(names(traindf) %in% c(bad_features, not_predictors))]
traindf = traindf[,colSums(is.na(traindf)) == 0]
```

After paring down the dataset, we are left with 53 columns and 19621 rows. We break this up into a training and testing dataset.

```
set.seed(234256)
inTrain = createDataPartition(traindf$classe, p = 0.6, list = F)
testdf = traindf[-inTrain,]
traindf = traindf[inTrain,]
```

We fit three mdoels to the training set: Random Forest, Stochastic Gradient Boosting, and Linear Discrimant Analysis. Our goal is to stack the results of these models to create an ensemble method.

```
modFit1 = train(classe~., method = 'rf', data = prelim_traindf, preProcess = c('scale', 'center'))
modFit2 = train(classe~., method = 'gbm', data = prelim_traindf, preProcess = c('center', 'scale'), verb
modFit3 = train(classe~., method = 'lda', data = prelim_traindf, preProcess = c('scale', 'center'))
pred1 = predict(modFit1, testdf)
pred2 = predict(modFit2, testdf)
pred3 = predict(modFit3, testdf)
```

After training the models on our training set, we find the predictions for the models on the test set. Using these predictions, we can create an ensemble method by having the three models vote on each prediction. We implement this below, and display the accuracies of the three models and the stacked model.

```
get_acc <- function(pred) sum(pred == testdf$classe)/length(pred)
acc1 = get_acc(pred1); acc2 = get_acc(pred2); acc3 = get_acc(pred3)
sprintf("RF Acc.: %.4f, GBM Acc.: %.4f, LDF Acc.: %.4f", acc1, acc2, acc3)
```

```
## [1] "RF Acc.: 0.9960, GBM Acc.: 0.9692, LDF Acc.: 0.6950"
```

```
most_accurate = 1
if (acc2 > acc1) most_accurate = 2
if (acc3 > acc1 & acc3 > acc2) most_accurate = 3
# Perform stacking, taking most popular prediction for each row, or
# prediction of most accurate model if none agree
stackdf = data.frame(pred1, pred2, pred3)
stack_pred = sapply(1:nrow(stackdf), function(i) Mode(stackdf[i,], returnNull = T, customNull = stackdf
sprintf('Stacked Acc.: %.4f', sum(stack_pred == testdf$classe)/length(stack_pred))
```

```
## [1] "Stacked Acc.: 0.9801"
```

With over 99% accuracy, the random forest method is the most accurate of the three. While stacking creates better predictions that two out of the three models, it actually performs worse than the Random Forest model alone, which is therefore our best model. The 99% accuracy was evaluated on the testing set, separate from where the model was trained, so this represents our estimate of the out-of-sample error rate.

Appendix

```
summary(initdf[,c(1:8,12:13,150:154,159:160)])
```

```
##        X            user_name    raw_timestamp_part_1 raw_timestamp_part_2
##  Min.   :    1   adelmo  :3892   Min.   :1.322e+09    Min.   :   294
##  1st Qu.: 4906   carlitos:3112   1st Qu.:1.323e+09    1st Qu.:252912
##  Median : 9812   charles :3536   Median :1.323e+09    Median :496380
##  Mean   : 9812   eurico  :3070   Mean   :1.323e+09    Mean   :500656
##  3rd Qu.:14717   jeremy  :3402   3rd Qu.:1.323e+09    3rd Qu.:751891
##  Max.   :19622   pedro   :2610   Max.   :1.323e+09    Max.   :998801
##
##          cvtd_timestamp  new_window    num_window      roll_belt
##  28/11/2011 14:14: 1498   no :19216   Min.   :  1.0   Min.   :-28.90
##  05/12/2011 11:24: 1497   yes:  406   1st Qu.:222.0   1st Qu.:  1.10
##  30/11/2011 17:11: 1440               Median :424.0   Median :113.00
##  05/12/2011 11:25: 1425               Mean   :430.6   Mean   : 64.41
##  02/12/2011 14:57: 1380               3rd Qu.:644.0   3rd Qu.:123.00
##  02/12/2011 13:34: 1375               Max.   :864.0   Max.   :162.00
##  (Other)         :11007
##  kurtosis_roll_belt kurtosis_picth_belt var_yaw_forearm
##           :19216             :19216     Min.   :    0.00
##  #DIV/0!  :   10    #DIV/0!  :   32     1st Qu.:    0.27
##  -1.908453:    2    47.000000:    4     Median :  612.21
##  0.000673 :    1    -0.150950:    3     Mean   : 4639.85
##  0.005503 :    1    -0.684748:    3     3rd Qu.: 7368.41
##  -0.016850:    1    11.094417:    3     Max.   :39009.33
##  (Other)  :  391    (Other)  :  361     NA's   :19216
##  gyros_forearm_x   gyros_forearm_y     gyros_forearm_z
##  Min.   :-22.000   Min.   : -7.02000   Min.   : -8.0900
##  1st Qu.: -0.220   1st Qu.: -1.46000   1st Qu.: -0.1800
##  Median :  0.050   Median :  0.03000   Median :  0.0800
##  Mean   :  0.158   Mean   :  0.07517   Mean   :  0.1512
##  3rd Qu.:  0.560   3rd Qu.:  1.62000   3rd Qu.:  0.4900
##  Max.   :  3.970   Max.   :311.00000   Max.   :231.0000
##
##  accel_forearm_x   magnet_forearm_z classe
##  Min.   :-498.00   Min.   :-973.0   A:5580
##  1st Qu.:-178.00   1st Qu.: 191.0   B:3797
##  Median : -57.00   Median : 511.0   C:3422
##  Mean   : -61.65   Mean   : 393.6   D:3216
##  3rd Qu.:  76.00   3rd Qu.: 653.0   E:3607
##  Max.   : 477.00   Max.   :1090.0
##
```