Practical Machine Learning Project

Executive summary

This anlysis fit an LDA and GBM model to the Weight Lifting Exercise Dataset found here: http://groupware.les.inf.puc-rio.br/har. The anlysis found that the LDA model suffered from collinearity issues and was unable to predict with more than 61% accuracy. The GBM model did not suffer from this issue and was able to predict with approximately 96% accuracy.

Import Data

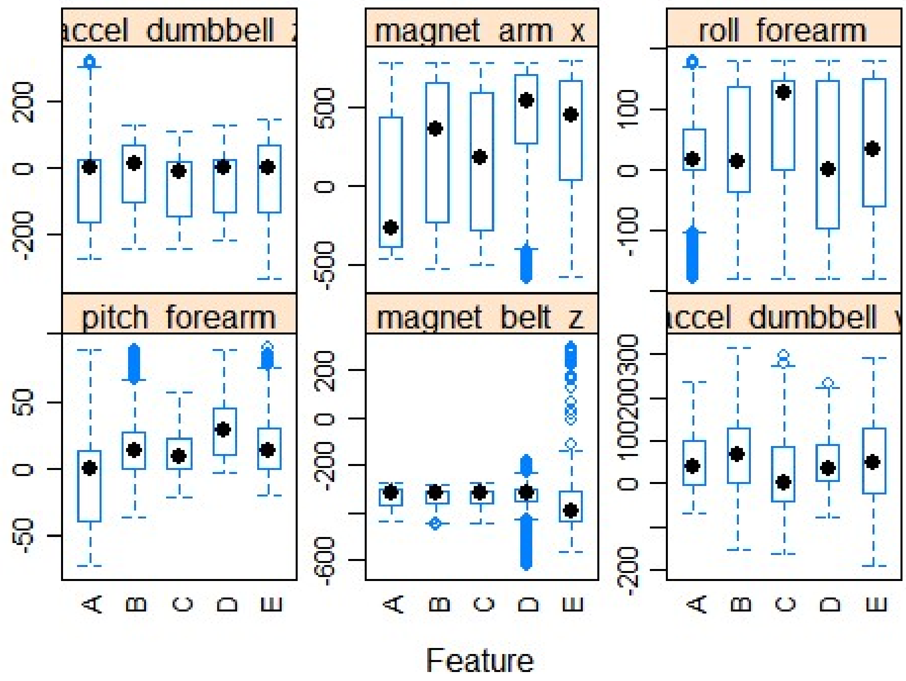
The data can be downloaded directly from the web and read from a temp file using the following code:

```
Temp <- tempfile()
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv",Temp)
OTraining <- read.csv(Temp)
download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv",Temp)
OTesting <- read.csv(Temp)
unlink(Temp)
```

Exploratory Analysis

The featurePlot function found in the Caret package is a useful tool for ispecting the data. A few of the features are plotted below.

```
featurePlot(x = OTraining[,
c("pitch_forearm","magnet_belt_z","accel_dumbbell_y"

,"accel_dumbbell_z","magnet_arm_x","roll_forearm")],
            y = factor(OTraining$classe),
            plot = "box",
            scales = list(y = list(relation="free"),
                          x = list(rot = 90)),
            layout = c(3,2 ))
```



Models ## LDA
Model Since the problem statement is to classify the manner in which exersises were performed and the data set has 5 potential classes and some differences in averages as shown in the exploratory anlysis, an LDA model seemed like a fair place to start. ### Variable Selection

Running:

```
View(OTraining)
```

Allows users to inspect the raw data in R. It is immediately apparent that there are a great deal of missing values and that most of them are in the columns with 'min', 'max', 'kurtosis', and 'skewness' in the variable name. While these provide useful detail about the distributions, there are far to many missing values for them to be useful. For the initial pass, the following predictors were used:

```
##       [,1]                [,2]
##  [1,] "roll_belt"         "roll_forearm"
##  [2,] "pitch_belt"        "pitch_forearm"
##  [3,] "yaw_belt"          "yaw_forearm"
##  [4,] "total_accel_belt" "total_accel_forearm"
##  [5,] "gyros_belt_x"      "gyros_forearm_x"
##  [6,] "gyros_belt_y"      "gyros_forearm_y"
##  [7,] "gyros_belt_z"      "gyros_forearm_z"
##  [8,] "accel_belt_x"      "accel_forearm_x"
##  [9,] "accel_belt_y"      "accel_forearm_y"
## [10,] "accel_belt_z"      "accel_forearm_z"
## [11,] "magnet_belt_x"     "magnet_forearm_x"
```

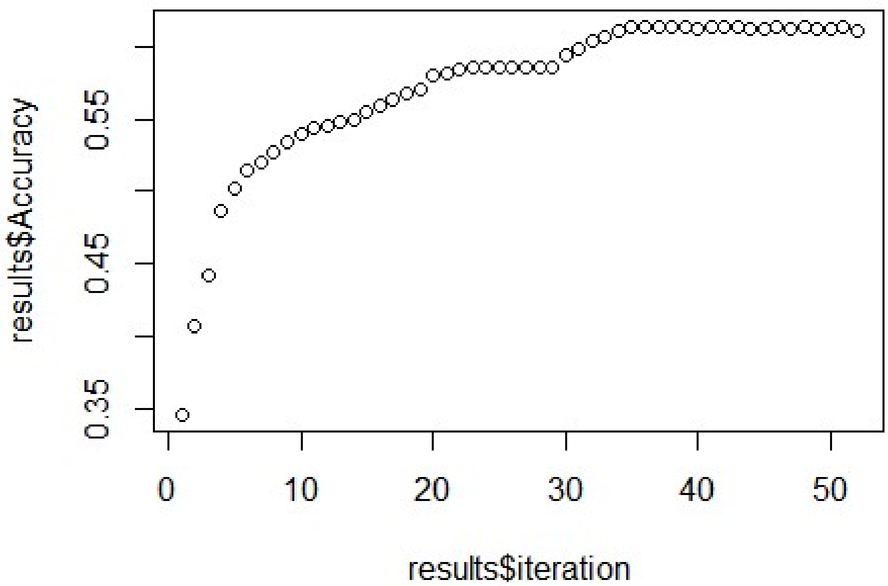
```
## [12,] "magnet_belt_y"    "magnet_forearm_y"
## [13,] "magnet_belt_z"    "magnet_forearm_z"
## [14,] "roll_arm"         "roll_dumbbell"
## [15,] "pitch_arm"        "pitch_dumbbell"
## [16,] "yaw_arm"          "yaw_dumbbell"
## [17,] "total_accel_arm"  "total_accel_dumbbell"
## [18,] "gyros_arm_x"      "gyros_dumbbell_x"
## [19,] "gyros_arm_y"      "gyros_dumbbell_y"
## [20,] "gyros_arm_z"      "gyros_dumbbell_z"
## [21,] "accel_arm_x"      "accel_dumbbell_x"
## [22,] "accel_arm_y"      "accel_dumbbell_y"
## [23,] "accel_arm_z"      "accel_dumbbell_z"
## [24,] "magnet_arm_x"     "magnet_dumbbell_x"
## [25,] "magnet_arm_y"     "magnet_dumbbell_y"
## [26,] "magnet_arm_z"     "magnet_dumbbell_z"
```

Feature selection

The following code was used to perform forward selection using the cross-validation error as the measure for feature selection:

```
{
LDAdata <- data.frame()
for(i in 1:(ncol(Training)-1)){
  dat =data.frame(iteration = rep(i,(ncol(Training)-i)),
                  model = (ncol(Training) - i):1)
  LDAdata  = rbind(LDAdata,dat)

}

predlist <- 1:(ncol(Training)-1)
selected <- c()
results <- data.frame()
for(i in 1:(((ncol(Training)-1)*ncol(Training))/2)){

  ldaFit <- train(classe ~., data =
Training[,c(selected,predlist[LDAdata[i,"model"]],53)]
                  , method = "lda"
                  ,trControl = trainControl(method = "cv"))
  LDAdata[i,"predictor"] = predictors[predlist[LDAdata[i,"model"]]]
  LDAdata[i,"Accuracy"] = ldaFit$results["Accuracy"]

  if (LDAdata[i,"model"] == 1) {
    iteration = LDAdata[i,"iteration"]
    candidates = LDAdata[LDAdata$iteration == iteration,]
    bestPred = candidates[which.max(candidates$Accuracy),"model"]
    selected = c(selected,bestPred)
    results = rbind(results,candidates[which.max(candidates$Accuracy),])
  }
}
```

```r
plot(results$iteration,results$Accuracy)
head(results)
}
```




```
##      X iteration model        predictor  Accuracy
## 1   25         1    28    pitch_forearm 0.3458363
## 2   91         2    13    magnet_belt_z 0.4066356
## 3  106         3    48 accel_dumbbell_y 0.4424615
## 4  154         4    49 accel_dumbbell_z 0.4861891
## 5  227         5    24     magnet_arm_x 0.5022939
## 6  271         6    27     roll_forearm 0.5144746
```

Best LDA Model Results

Looking at the estimated test accuracy as more predictors are added in shows that there is very little gain in accuracy after 34 predictors. Adding additional predictors could lead to overtraining. As a result, the first 34 predictors were used to create an LDA model which had the following results:

```r
predictors = results[1:34,"predictor"]
ldaFit <- train(classe ~., data = Training[,c(predictors,"classe")], method =
"lda"
                ,trControl = trainControl(method = "cv"))
ldaFit

## Linear Discriminant Analysis
##
```

```
## 19622 samples
##    34 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17659, 17658, 17661, 17659, 17661, 17660, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6107435  0.505782
```

```
predictions <- predict(ldaFit,newdata = Training[,c(predictors)])
confusionMatrix(predictions,as.factor(Training$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 4036  511 1033  357  304
##          B  243 2429  496  180  735
##          C  481  473 1604  309  362
##          D  786  233  235 2165  422
##          E   34  151   54  205 1784
##
## Overall Statistics
##
##                Accuracy : 0.6125
##                  95% CI : (0.6056, 0.6193)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.508
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7233   0.6397  0.46873   0.6732  0.49459
## Specificity            0.8430   0.8955  0.89969   0.8978  0.97228
## Pos Pred Value         0.6467   0.5949  0.49675   0.5637  0.80072
## Neg Pred Value         0.8846   0.9120  0.88910   0.9334  0.89519
## Prevalence             0.2844   0.1935  0.17440   0.1639  0.18382
## Detection Rate         0.2057   0.1238  0.08174   0.1103  0.09092
## Detection Prevalence   0.3181   0.2081  0.16456   0.1957  0.11355
## Balanced Accuracy      0.7831   0.7676  0.68421   0.7855  0.73343
```

The Cross-validation accuracy is approximately 61% and the training accuracy is 61%. This is far too low to pass the quiz which requires an 80% or higher. The model does warn that there are collinear variables but the LDA assumptions may not work well for this data set.

GBM Model

Since the LDA model accuracy isn't high enough, a gradient boosting model may yield better results and if not, it could be blended with the LDA and an additional model to achieve the desired accuracy. The gbm model was set using the following code:

```
modelFit <- train(classe ~., data = Training, method = "gbm",
                  trControl = trainControl(method = "cv"))
modelFit
predictions <- predict(modelFit,newdata = Training)
predictions
confusionMatrix(predictions,as.factor(Training$classe))
```

```
## Stochastic Gradient Boosting
##
## 19622 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 17660, 17660, 17660, 17660, 17660, 17660, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7518585  0.6854092
##   1                  100      0.8208644  0.7732484
##   1                  150      0.8548570  0.8163272
##   2                   50      0.8557760  0.8172780
##   2                  100      0.9076056  0.8830819
##   2                  150      0.9328330  0.9150096
##   3                   50      0.8979223  0.8707768
##   3                  100      0.9436868  0.9287508
##   3                  150      0.9634096  0.9537090
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth
=
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction    A    B    C    D    E
##          A 5527   86    0    1    4
##          B   35 3642   88    8   25
##          C   12   69 3298   85   24
##          D    4    0   32 3109   39
##          E    2    0    4   13 3515
##
## Overall Statistics
##
##                Accuracy : 0.9729
##                  95% CI : (0.9706, 0.9752)
##     No Information Rate : 0.2844
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9658
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9905   0.9592   0.9638   0.9667   0.9745
## Specificity            0.9935   0.9901   0.9883   0.9954   0.9988
## Pos Pred Value         0.9838   0.9589   0.9455   0.9764   0.9946
## Neg Pred Value         0.9962   0.9902   0.9923   0.9935   0.9943
## Prevalence             0.2844   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2817   0.1856   0.1681   0.1584   0.1791
## Detection Prevalence   0.2863   0.1936   0.1778   0.1623   0.1801
## Balanced Accuracy      0.9920   0.9747   0.9760   0.9811   0.9867
```

GBM Results

The GBM is leaps and bounds better than the LDA with an estimated accuracy of 96% and a training accuracy of 97%. As a result, this model was applied to the test data using the following code:

```
quiz <- Testing
quiz$prediction <- predict(modelFit,newdata = Testing)
quiz <- quiz[,c("problem_id","prediction")]
```

The prediction results were then entered into the week 4 quiz 2 with passing results.