Machine Learning Project

The purpose of this exercise is to build a machine learning algorithm to predict activity quality based on wearable activity monitors. The data used to conduct this analysis is from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har). The scope of the project was to identify the activity quality using a classification algorithm using tools available in R. More specifically, the following analysis borrows heavily on the 'caret' and 'randomForest' package for prediction and evaluation of model results. The following sections details some of the data cleaning, model training, goodness of fit, and supplement codes and charts for reproducibility.

Data Preparation

```
##Prepare Packages
library(caret)
library(ggplot2)
library(lattice)
library(kernlab)
library(randomForest)
```

```
#Data Prep
#Download Files
download.file(url = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfile =
"pml-training.csv", mode='wb')
download.file(url = "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfile =
"pml-testing.csv", mode='wb')
```

```
#Raw Files
trainRaw <- read.csv('pml-training.csv')
testRaw <- read.csv('pml-testing.csv')
```

Upon examining the data, there were several columns removed due to lack of information. Cleaning the data will speed processing time which is a weakness of several machine learning techniques.

```
#Data Cleaning
#Remove near zero variance predictors
remove_nzero <- nearZeroVar(trainRaw)
length(remove_nzero) #60 Vars with near zero variance

trainClean1 <-trainRaw[,-remove_nzero]

#remove columns with over a 90% that have NAs
nasPerColumn <- apply(trainClean1,2,function(x) {sum(is.na(x))});
trainClean2  <- trainClean1[,which(nasPerColumn <  nrow(trainClean1)*0.9)];

#remove additonal attributes
trainClean <- trainClean2[, -which(names(trainClean2) %in% c("X", "user_name", "raw_timestamp_par
t_1", "raw_timestamp_part_2", "cvtd_timestamp", "num_window"))]


dim(trainClean)
#19622rows 53cols

summary(trainClean$classe)
```

For Validation purposes, the data was split 60/40. Since the data set is relatively large, this should be sufficient for training purposes.

```
#Create Train/Validation Data using .60/.40 Split
set.seed(5555)

Split <- createDataPartition(y = trainClean$classe, p=0.6,list=FALSE);

training60 <- trainClean[Split,];
testing40 <- trainClean[-Split,];

#basic housekeeping, remove some datasets
rm(Split, trainClean1, trainClean2, trainClean)
```

The tool of choice for this classification problem is Random Forest (RF). There are several other tools available such as LDA, but RF is typically strong for classification problems. Here we use the Random Forest package to perform the training.

```
#Random Foreset Algorithm
model1 <- randomForest(classe ~ ., data=training60, importance = TRUE, ntrees = 8)
```

Fit on Training Sample

```
#Estimated Error Rate
model1
```

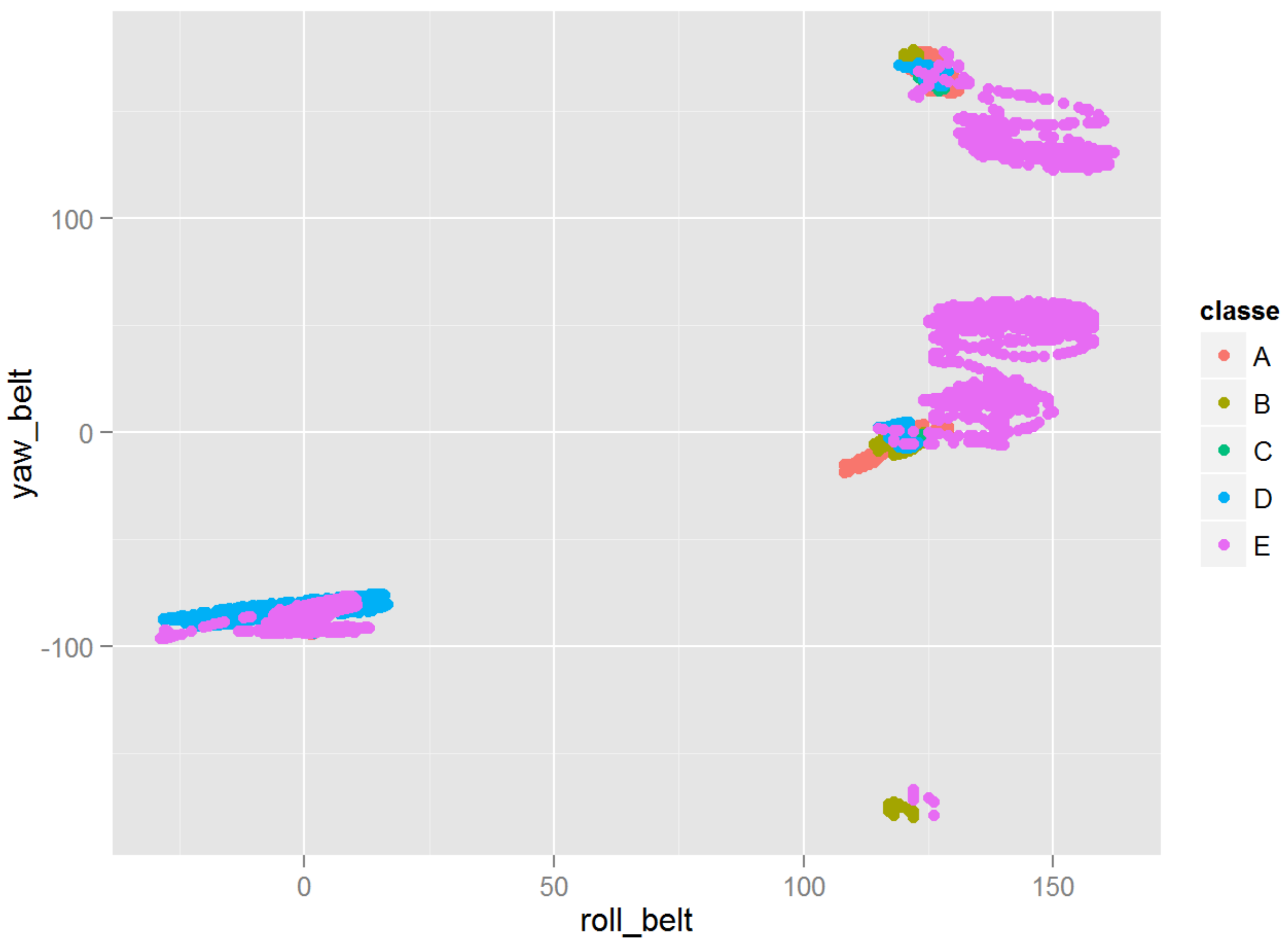
```
##
## Call:
##  randomForest(formula = classe ~ ., data = training60, importance = TRUE,      ntrees = 8)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.6%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3345    2    0    0    1 0.0008960573
## B   12 2259    7    1    0 0.0087757789
## C    0    8 2039    7    0 0.0073028238
## D    0    0   23 1905    2 0.0129533679
## E    0    0    2    6 2157 0.0036951501
```

We see that the estimated error rate is very good at 0.6%. The error rate on the validation sample is shown several paragraphs below.

We examine the variable importance of the RF algorithm as a sanity check and plot two of the top factor.

```
#Var Importance
importance(model1)

#See that roll_belt, pitch_belt, and yaw_belt are top 3
```

```
#Plot some important factors

library(ggplot2)

#p<- qplot(roll_belt, pitch_belt, col=classe, data=trainRaw)
p2<- qplot(roll_belt, yaw_belt, col=classe, data=trainRaw)

#p + geom_point(aes(x=roll_belt, y=pitch_belt, col=classe), data=trainRaw)
p2 + geom_point(aes(x=roll_belt, y=yaw_belt, col=classe), data=trainRaw)
```



```
The chart shows that with these two variables the classes follow a distinct pattern. For example, whe
n rollbelt is less than 50 with yawbelt is approx -100 the two classes present are D and E.
```

Fit on Validation Sample:

```
test40_pr <- predict(model1, testing40)

print(confusionMatrix(test40_pr, testing40$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2223   10    0    0    0
##          B    8 1506    9    0    0
##          C    0    2 1357   14    0
##          D    1    0    2 1271    7
##          E    0    0    0    1 1435
##
## Overall Statistics
##
##                Accuracy : 0.9931
##                  95% CI : (0.991, 0.9948)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9913
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9960   0.9921   0.9920   0.9883   0.9951
## Specificity            0.9982   0.9973   0.9975   0.9985   0.9998
## Pos Pred Value         0.9955   0.9888   0.9883   0.9922   0.9993
## Neg Pred Value         0.9984   0.9981   0.9983   0.9977   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2833   0.1919   0.1730   0.1620   0.1829
## Detection Prevalence   0.2846   0.1941   0.1750   0.1633   0.1830
## Balanced Accuracy      0.9971   0.9947   0.9947   0.9934   0.9975
```

Here we see that the accuracy is approx. 99.31%.

Fit on Test

```
#testRaw_pr <- predict(model1, testRaw)
#testRaw_pr
```