Weight Exsercise Prediction

Emmanuel Okyere Darko

4/5/2021

Executive Summary

In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which an exercise is done.

In particular, we will:

Get and clean the data set:

- Train data is from here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)
- Test data is found at here (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Perform Exploratory data analysis to identify patterns

Pre-process data to split and validate, reduce dimentionality with PCA and remoe zero covariates

---

Fit models on different predictors

Assess model metrics

Summary

Getting and Cleaning Data

Data

```
df.train <-read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.cs
        v')

df.test <- read.csv('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
        v')

dim(df.train)
```

```
## [1] 19622   160
```

Data Cleaning

Checking for NAs, it looks like some columns have NAs of more than `50%` the size of the dataset, therefore columns with NAs more than `20%` of the size of the data will be droped, which then reduce the number of columns. Also we will remove timestamp columns since we don't need them as prediction

```
size <- nrow(df.train)
perc <- 20
Na_thresh <- floor(size/100 * perc)

## Drop columns with more than 20% na
dropped_cols <- which(colSums(is.na(df.train) | df.train == "") > Na_thresh)

train_val_set <- df.train[, -c(1, dropped_cols)]
test_set <- df.test[, -c(1, dropped_cols)]

## Drop all columns having timestamp
dropped_time_cols <- grep('timestamp', names(train_val_set))
train_val_set <- train_val_set[, -dropped_time_cols]
test_set <- test_set[, -dropped_time_cols]

## Make classe a factor
train_val_set$classe <- factor(train_val_set$classe)
train_val_set$user_name <- factor(train_val_set$user_name)
test_set$user_name <- factor(test_set$user_name)
```

Split training set to train and validation

`3/4` for training and `1/4` for validation

```
partition <- createDataPartition(y = train_val_set$classe, p = 3/4, list = F)

train_data <- train_val_set[partition, ]
validation_data <- train_val_set[-partition, ]

response <- which(names(train_data) == c("classe"))
```

Exploratory Analysis

Removing zero covariates

zero or near zero covariates predictors will be removed

```
nsv <- nearZeroVar(train_data, saveMetrics=TRUE)

train_data = train_data[, !nsv$nzv]
validation_data = validation_data[, !nsv$nzv]
test_set = test_set[, !nsv$nzv]

nsv
```

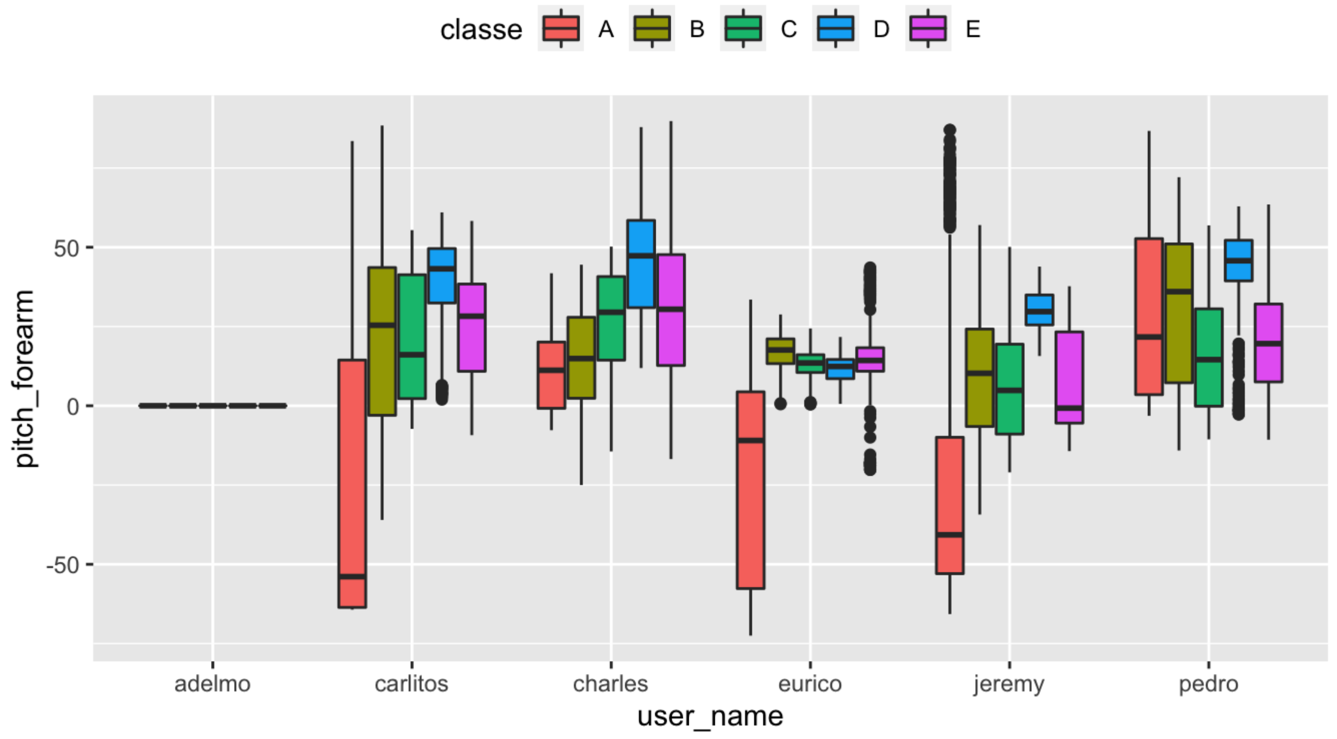
```
##                      freqRatio percentUnique zeroVar   nzv
## user_name             1.104766    0.04076641   FALSE FALSE
## new_window           47.255738    0.01358880   FALSE  TRUE
## num_window            1.034483    5.82280201   FALSE FALSE
## roll_belt             1.110429    7.91547765   FALSE FALSE
## pitch_belt            1.013514   11.80866966   FALSE FALSE
## yaw_belt              1.060686   12.55605381   FALSE FALSE
## total_accel_belt      1.064262    0.19024324   FALSE FALSE
## gyros_belt_x          1.026188    0.90365539   FALSE FALSE
## gyros_belt_y          1.146927    0.44163609   FALSE FALSE
## gyros_belt_z          1.057678    1.11428183   FALSE FALSE
## accel_belt_x          1.119171    1.10748743   FALSE FALSE
## accel_belt_y          1.155203    0.95121620   FALSE FALSE
## accel_belt_z          1.098935    1.94319880   FALSE FALSE
## magnet_belt_x         1.130112    2.08588123   FALSE FALSE
## magnet_belt_y         1.168421    1.95678761   FALSE FALSE
## magnet_belt_z         1.020000    3.00312542   FALSE FALSE
## roll_arm             46.163636   16.68025547   FALSE FALSE
## pitch_arm            79.375000   19.35724963   FALSE FALSE
## yaw_arm              32.974026   18.19540698   FALSE FALSE
## total_accel_arm       1.022422    0.44843049   FALSE FALSE
## gyros_arm_x           1.114441    4.27367849   FALSE FALSE
## gyros_arm_y           1.496021    2.50713412   FALSE FALSE
## gyros_arm_z           1.121588    1.63065634   FALSE FALSE
## accel_arm_x           1.088000    5.22489469   FALSE FALSE
## accel_arm_y           1.200000    3.60782715   FALSE FALSE
## accel_arm_z           1.144330    5.23168909   FALSE FALSE
## magnet_arm_x          1.000000    9.02975948   FALSE FALSE
## magnet_arm_y          1.074627    5.80241881   FALSE FALSE
## magnet_arm_z          1.000000    8.54056258   FALSE FALSE
## roll_dumbbell         1.000000   86.27530915   FALSE FALSE
## pitch_dumbbell        2.382353   83.93124066   FALSE FALSE
## yaw_dumbbell          1.120879   85.58907460   FALSE FALSE
## total_accel_dumbbell  1.098537    0.29215926   FALSE FALSE
## gyros_dumbbell_x      1.015217    1.60347873   FALSE FALSE
## gyros_dumbbell_y      1.287671    1.86166599   FALSE FALSE
## gyros_dumbbell_z      1.055679    1.33170268   FALSE FALSE
## accel_dumbbell_x      1.044355    2.86044299   FALSE FALSE
## accel_dumbbell_y      1.081967    3.09824704   FALSE FALSE
## accel_dumbbell_z      1.171271    2.75173257   FALSE FALSE
## magnet_dumbbell_x     1.058824    7.33795353   FALSE FALSE
## magnet_dumbbell_y     1.192593    5.60538117   FALSE FALSE
## magnet_dumbbell_z     1.027972    4.52507134   FALSE FALSE
## roll_forearm         11.641434   13.20152195   FALSE FALSE
## pitch_forearm        64.911111   18.21579019   FALSE FALSE
## yaw_forearm          15.455026   12.22312814   FALSE FALSE
## total_accel_forearm   1.157780    0.46201930   FALSE FALSE
## gyros_forearm_x       1.062189    1.91602120   FALSE FALSE
## gyros_forearm_y       1.062718    4.93952983   FALSE FALSE
## gyros_forearm_z       1.092643    2.01114282   FALSE FALSE
## accel_forearm_x       1.169231    5.30642750   FALSE FALSE
## accel_forearm_y       1.040000    6.69248539   FALSE FALSE
## accel_forearm_z       1.008403    3.79807039   FALSE FALSE
## magnet_forearm_x      1.064516   10.00815328   FALSE FALSE
## magnet_forearm_y      1.206349   12.44734339   FALSE FALSE
## magnet_forearm_z      1.106383   11.02731349   FALSE FALSE
## classe                1.469452    0.03397201   FALSE FALSE
```

Check for correlation between predictors and response variable

The table below shows all variables have less correlation with the response varible

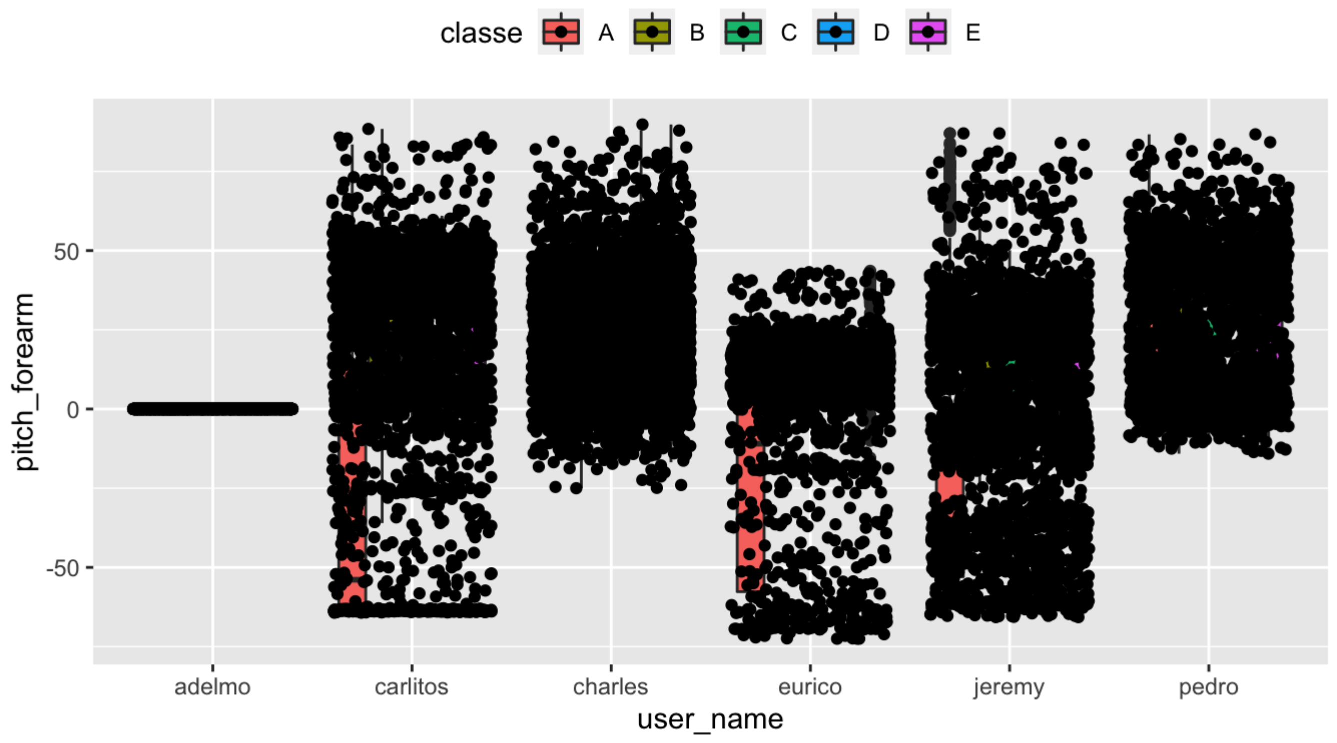
```
## Remove non numeric columns before calculating correlation
df <- train_data %>% select(-c("user_name", "classe"))

corr <- cor(df, as.numeric(train_data$classe))

## Convert to dataframe and arrange in decreasing order
coor_df = data.frame(name= row.names(corr) ,pos_cor = abs(corr))
coor_df[coor_df$pos_cor >0.3,]
```

We can now visualize this for a better pespertive



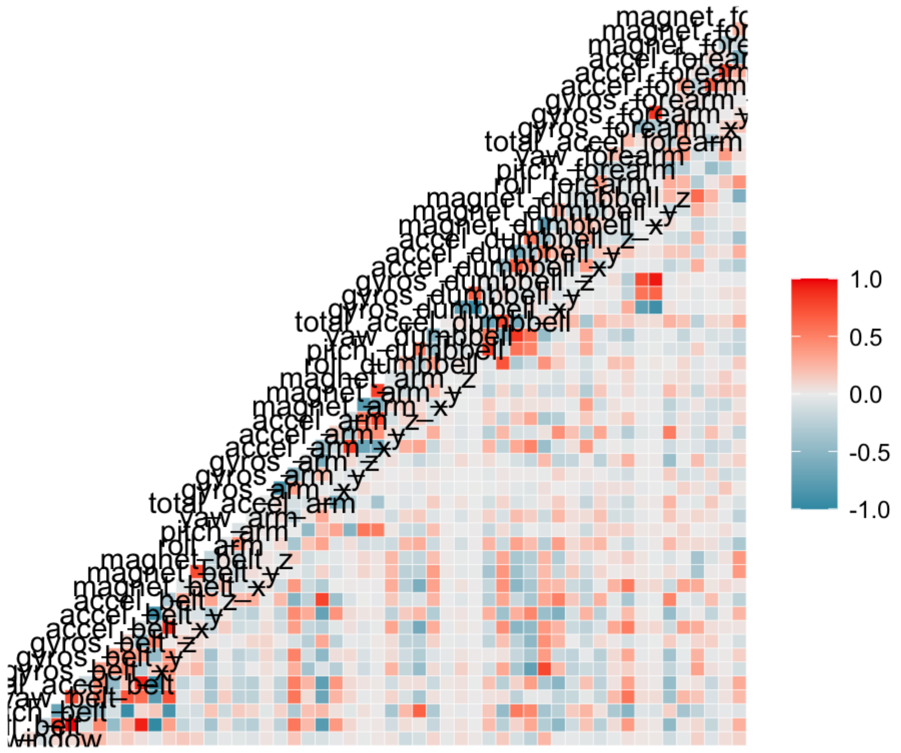
Plot with points overlayed

This plot shows the number of data-points in each category of user-name, it can be inferred that each person is well represented in the dataset



Check for correlation between predictors

This plot is not very informational as we have several variable but we can infer that some variables are strongly correlated with others



Now is a better tim to eliminate highly correlated columns

```
high_cor = findCorrelation(cor(df), cutoff = 0.8)

exclude_cols = c(response, high_cor)
```

Pre-processing for training

To reduce overfitting and also dimentionality, we will use PCA with a thresh of `0.9`

```
## USe pca to reduce highly correlated variables
pca.all <- preProcess(train_data[, -response], method = 'pca', thresh = 0.9)
train_data.pca.all <- predict(pca.all, train_data[, -response])
validation_data.pca.all <- predict(pca.all, validation_data[, -response])
test_set.pca.all <- predict(pca.all, test_set[, -response])

## remove highly correlated columns and fit pca
pca.excluded <- preProcess(train_data[, -exclude_cols], method = 'pca', thresh = 0.9)
train_data.pca.excluded <- predict(pca.excluded, train_data[, -exclude_cols])
validation_data.pca.excluded <- predict(pca.excluded, validation_data[, -exclude_cols])
#test_set.pca.excluded <- predict(pca.excluded, test_set[, -exclude_cols])
```

Model

- Before pca model The next model will be fit on predictors after removing highly correlated variables.
- The train function takes a almost `10x` time to train relative to the specific `randomForest function` but highly efficient.

  user system elapsed 3955.071 67.123 4067.016

```
rf.all <- randomForest::randomForest(x = train_data[, -response], y = train_data$classe,
                                 ntree = 100,
                                  ytest =validation_data$classe, xtest = validation_data
        [, -response])

rf.excluded <- randomForest::randomForest(x = train_data[, -exclude_cols], y = train_dat
        a$classe,
                                 ntree = 100,
                                  ytest =validation_data$classe, xtest = validation_data
        [, -exclude_cols])

rf.pca <- randomForest::randomForest(x = train_data.pca.all, y = train_data$classe,
                                 ntree = 200,
                                 xtest =validation_data.pca.all, ytest =validation_data$
        classe )

rf.pca.excluded <- randomForest::randomForest(x = train_data.pca.excluded, y = train_dat
        a$classe,
                                 ntree = 200,  ytest =validation_data$classe ,
                                 xtest = validation_data.pca.excluded)
```

Metrics

Train Accuracy

```
rf1 = round(1 - sum(rf.all$confusion[, "class.error"]), 3)
rf2 = round(1 - sum(rf.excluded$confusion[, "class.error"]), 3)
rf3 = round(1 - sum(rf.pca$confusion[, "class.error"]), 3)
rf4 = round(1 - sum(rf.pca.excluded$confusion[, "class.error"]), 3)


print(paste("All Predictors acc:", rf1, "Predictors with no high cor acc: ", rf2,
            "PCA acc: ", rf3, "PCA with high corr removed acc:", rf4 ))
```

```
## [1] "All Predictors acc: 1 Predictors with no high cor acc:  1 PCA acc:  1 PCA with h
igh corr removed acc: 1"
```

Validation Accuracy

```
cm.1 <- round(1 - sum(rf.all$test$confusion[, "class.error"]), 3)
cm.2 <- round(1 - sum(rf.excluded$test$confusion[, "class.error"]), 3)
cm.3 = round(1 - sum(rf.pca$test$confusion[, "class.error"]), 3)
cm.4 = round(1 - sum(rf.pca.excluded$test$confusion[, "class.error"]), 3)

#round(confusionMatrix(validation_data$classe, pred.rf.pca.excluded)$overall[1] , 3)


print(paste("All Predictors acc:", cm.1, "Predictors with no high cor acc: ", cm.2,
            "PCA acc: ", cm.3, "PCA with high corr removed acc:", cm.4 ))
```

```
## [1] "All Predictors acc: 1 Predictors with no high cor acc:  0.999 PCA acc:  1 PCA wi
th high corr removed acc: 1"
```

Summary

- PCA reduces computational time and also gives a good parsimonious model
- Train function is over `10x` slower than specific function `randomForest`
- Removing highly correlated predictors before PCA does not change model performance significantly as PCA takes care of the same thing
- An Accuracy of `99.7%` and `87%` was achieved on validation set using all predictors and PCA respectively