Machine learning project

Executive Summary

The data is from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The goal was to predict the manner in which they did the exercise. This is the "classe" variable in the training set.

In this report, we used cross validation and evaluated three different predition methods on the training set and chose the model that gave the highest accuracy as the "best model". This model was then applied one time on the test dataset to get the final class prediction.

After comparing three methods - PCA with SVM, Random Forest, and GLM Net, I found that the Random Forest model gave the highest accuracy and chose that as the best model.

Downloading data and installing

```r
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
download.file(trainUrl, destfile = "train.csv",method = "curl")

testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
download.file(testUrl, destfile = "test.csv",method = "curl")

install.packages("AppliedPredictiveModeling")
install.packages("caret", dependencies = TRUE)
```

Read in train and test set

```r
library(AppliedPredictiveModeling)
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

Data cleaning

More information about the data: Each version of the excersise (classe A-E) was repeated 10 times. The researchers devided each collection of measurements for a person and exercise class into timeframes (sliding window approach). The end of a timeframe is marked with "yes" in the "newwindow"" column. In these rows they give the values for the calculated features (like mean, standard deviation, etc.) for the respective window. Each window (timeframe) is identified in the num_window column.

```r
#Since new_window=yes colunms have mean, SD which are derived from original data,
#we don't really care about them. So remove those rows
train1 <- train[(train$new_window=="no"),]
```

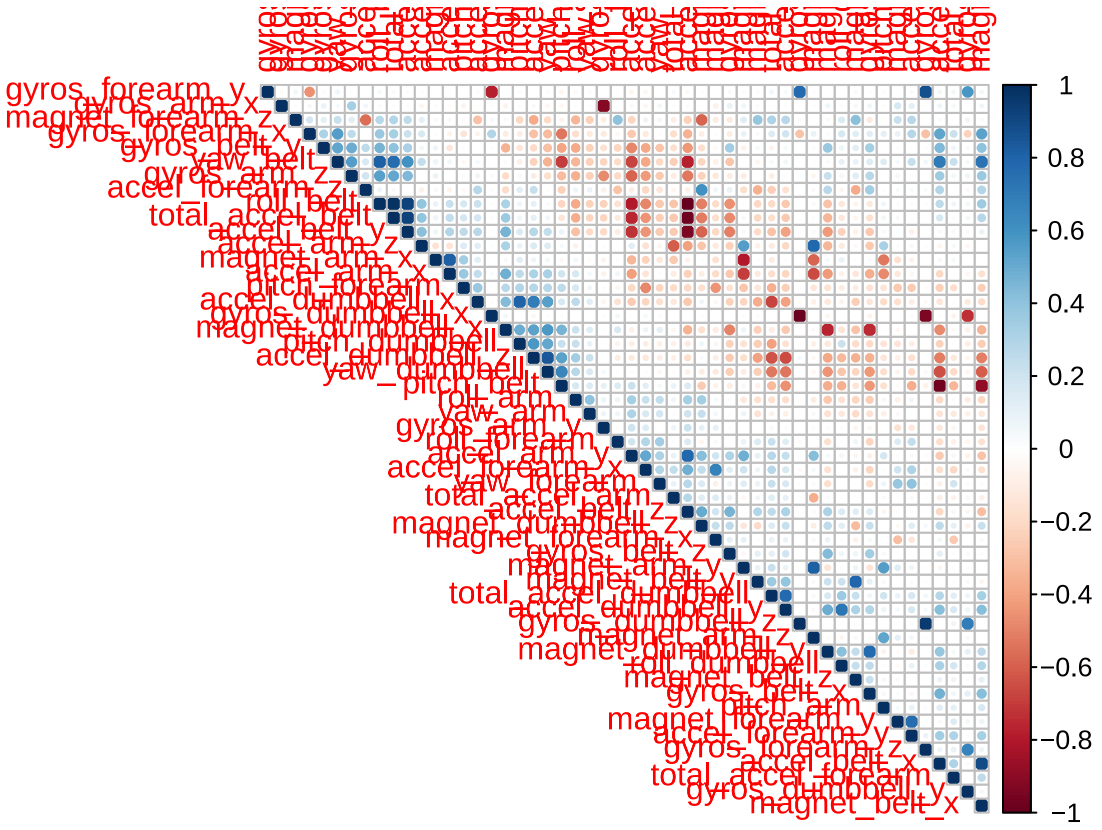
```r
#remove columns that have NA or blank values
col1 <- colSums(is.na(train1)) #Sum of NA values in each column
train2 <- train1[,(col1 == 0)] #remove Columns that have atleast one NA value
col2 <- colSums(train2 == "")
train3 <- train2[, (col2 == 0)] #remove Columns that have atleast one blank value

#creating unique primary key
train3$name <- paste(train3$X, train3$user_name, train3$raw_timestamp_part1,
                     train3$raw_timestamp_part2, sep="|")
row.names(train3) <- train3$name

#The first few columns are un-necessary, so removing them
train4 <- train3[, -(1:7)]
train4 <- train4[, -(54)]

#checking for zero variance -- none of them have zero variance
#nsv <- caret::nearZeroVar(trainClean,saveMetrics = TRUE)
trainClean <- train4 #this is the final cleaned training data

#pull out same columns for test data also
colName1 <- colnames(trainClean)
testClean <- test[, names(test) %in% colName1] #note that this does not have Classe Column

# split cleaned training data into 70-30
trainIndex = createDataPartition(trainClean$classe,p=0.70,list=FALSE)
subTrain = trainClean[trainIndex,]
subTest = trainClean[-trainIndex,]
```

Perform exploratory analysis on cleaned training set

```r
install.packages("corrplot")
```

The corrlelation plot shows a number of highly correlated variables. So we need to do some feature selection and choose only those features that are the most informative.

```r
library(corrplot)
# exploratory analysis -- on train set
corrplot(cor(subTrain[,1:52]), type="upper", order="AOE")
```



Model 1 - Using Principal Component Analyis (PCA) and Support Vector machine(SVM)

Using PCA for dimension reduction, and SVM for model training and testing. We first find PCs that can explain 90% of variability (if I had more RAM I would have set that to 95% variability). Use those principal components generated on training set to build SVM model and apply on training set. Note: For this kind of data, I did not center, scale or log transform data. If all the columns were the same measurements (eg: like the IL* markers in the AlzheimerDisease), I might have used these preprocessing methods.

From this model, we ge an Accuracy of 0.8766 (87%) and a Kappa of 0.8423 (excellent)

The kappa statistic is a measure of how closely the instances classified by the machine learning classifier matched the data labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy. This statistic can shed light into how the classifier itself performed, the kappa statistic for one model is directly comparable to the kappa statistic for any other model used for the same classification task. According to Fleiss, kappas > 0.75 are excellent, 0.40-0.75 as fair to good, and < 0.40 as poor.

```r
#Find the number of features that can explain 90% of variability
preProc1 <- preProcess(subTrain[,1:52], method="pca", thresh=0.90, verbose = FALSE)
#preProc1$rotation #25 PCs - 95% variance. 18PCs for 90% variance

#for parallel processing
library(parallel, quietly=T)
library(doParallel, quietly=T)

#Turn on Parallel Processing (leave at least 1 core free so your machine
#is still usable when doing the calc)
```

```
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

#5 fold cross validation
trainControl5Fold <- trainControl(method="cv", number=5)
# if you have more RAM, do this:
#trainControl5Fold <- trainControl(method="repeatedcv", number=10, repeats = 5)

trainPC <- predict(preProc1, subTrain[,1:52]) #using PCA on train data to get train PCs

# using SVM to create model
set.seed(1)
modelFit1 <- train(subTrain$classe~., data=trainPC, method="svmRadial", trControl=trainControl5Fold) #
```

```
## Loading required package: kernlab
```

```
predictors(modelFit1) #shows the selected 18 PC features
```

```
##  [1] "PC1"  "PC2"  "PC3"  "PC4"  "PC5"  "PC6"  "PC7"  "PC8"  "PC9"  "PC10"
## [11] "PC11" "PC12" "PC13" "PC14" "PC15" "PC16" "PC17" "PC18"
```

```
modelFit1$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0369389033166206
##
## Number of Support Vectors : 7541
##
## Objective Function Value : -1354.929 -1278.501 -1084.357 -660.5552 -1396.921 -857.1512 -1002.307 -10
## Training error : 0.115811
```

```
testPC <- predict(preProc1, subTest[,1:52]) #using train PCA on test data
predictedY1 <- predict(modelFit1, testPC) #predict outcome on test data

confusionMatrix(subTest$classe, predictedY1) #Actual Y from test data vs Predicted Y
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1575   14   37   15    0
##          B  146  880   79    2    8
##          C   25   63  885   31    1
##          D   30   20  110  781    3
##          E   10   27   64   42  915
##
```

```
## Overall Statistics
##
##                Accuracy : 0.8739
##                  95% CI : (0.865, 0.8823)
##     No Information Rate : 0.3099
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.84
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8819   0.8765   0.7532   0.8967   0.9871
## Specificity            0.9834   0.9506   0.9738   0.9667   0.9704
## Pos Pred Value         0.9598   0.7892   0.8806   0.8273   0.8648
## Neg Pred Value         0.9488   0.9733   0.9391   0.9813   0.9974
## Prevalence             0.3099   0.1742   0.2039   0.1511   0.1609
## Detection Rate         0.2733   0.1527   0.1536   0.1355   0.1588
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      0.9326   0.9136   0.8635   0.9317   0.9787
```

```
# Accuracy = 0.8766

## Turn off Parallel Processing
stopCluster(cluster)
registerDoSEQ()
```

Model 2 - Random Forest

Random Forst will automatically also do feature selection. From this model, we get Acuracyof 0.9912 (99%), and OOB estimate of error rate 0.76%

```
# Model 2 -- Random Forst will automatically also do feature selection

# Turn on Parallel Processing (leave at least 1 core free so your machine
#is still usable when doing the calc)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

set.seed(1)
modelFit2 <- train(subTrain$classe~., data=subTrain, method="rf",
                   trControl=trainControl5Fold)
```

```
## Loading required package: randomForest
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
#predictors(modelFit2) #Shows the selected features in the final model
#modelFit2$finalModel
predictedY2 <- predict(modelFit2, subTest[,1:52]) #prediction
confusionMatrix(subTest$classe, predictedY2) #confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1640    1    0    0    0
##          B   14 1096    5    0    0
##          C    0    3  998    4    0
##          D    0    0   10  933    1
##          E    0    0    1    2 1055
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.287
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9915   0.9964   0.9842   0.9936   0.9991
## Specificity            0.9998   0.9959   0.9985   0.9977   0.9994
## Pos Pred Value         0.9994   0.9830   0.9930   0.9883   0.9972
## Neg Pred Value         0.9966   0.9991   0.9966   0.9988   0.9998
## Prevalence             0.2870   0.1909   0.1760   0.1629   0.1832
## Detection Rate         0.2846   0.1902   0.1732   0.1619   0.1831
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      0.9956   0.9961   0.9914   0.9957   0.9992
```

```
#Acuracy = 0.9912
#OOB estimate of  error rate: 0.76%

## Turn off Parallel Processing
stopCluster(cluster)
registerDoSEQ()
```

Model 3 - Generalized linear model (GLM)

When Alpha = 0 is ridge regression, alpha = 1 means it is lasso. Alpha = 0.5 is elastic net. Lambda is the shrinkage parameter: when Lambda=0, no shrinkage is performed, and as Lambda increases, the coefficients are shrunk ever more strongly.

In order to choose from the various models with different lambda values provided by glmnet, we perform cross validation using the cv.glmnet function with misclassification error as the criterion for 5-fold procedure. The lambda value that yields the minimum cross validation error gives the best model. The figure below shows a plot of lambda with mis-classification error. The very small error rate at the optimal lambda is a sign of the model's effectiveness.

We can see that the final GLM model has an accuracy of 0.7321 (73%), with Kappa value of 0.6603 (good).

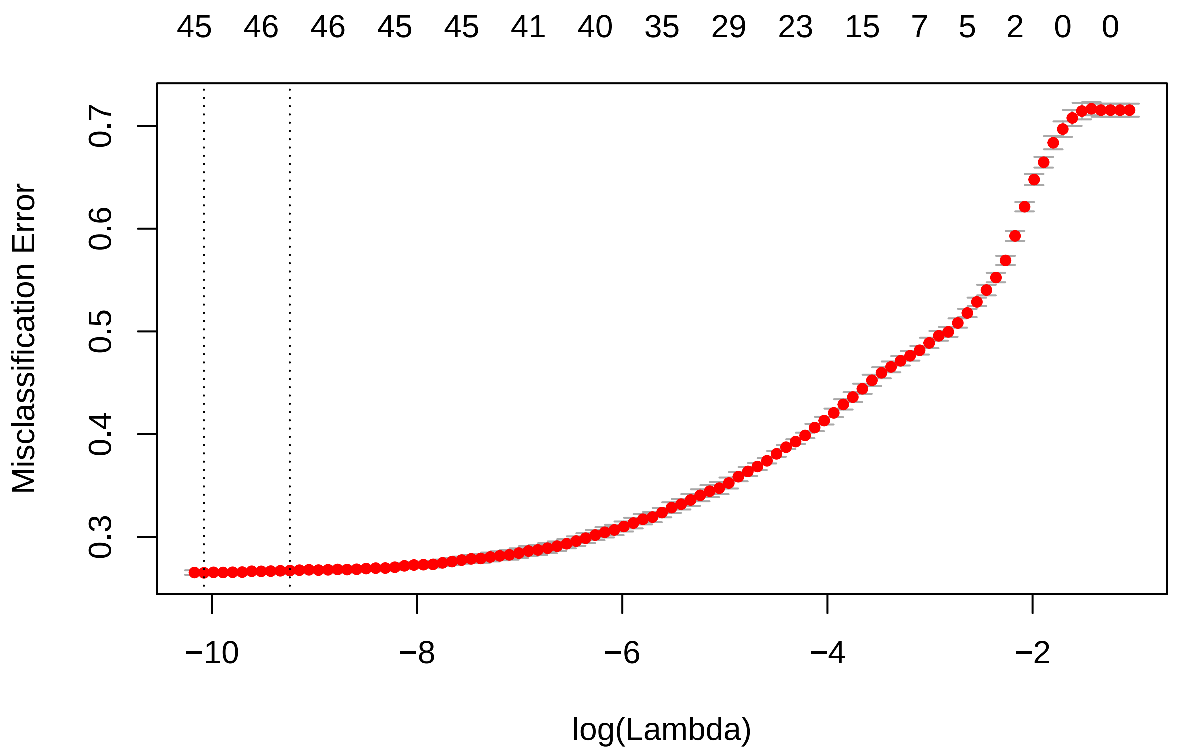
```r
library(glmnet)
```

```
## Loading required package: Matrix
## Loaded glmnet 2.0-2
```

```r
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

# GLM
set.seed(1)
modelFit3 <- glmnet(x = as.matrix(subTrain[, 1:52]), y = subTrain$classe, family = "multinomial", alpha

## OR
# modelFit3 <- train(subTrain$classe~., data=subTrain, method="glmnet",family="multinomial", tuneGrid =

#Model with cross validation.Lambda value that yields the minimum cross validation error is returned
modelFit3.cv <-cv.glmnet(x = as.matrix(subTrain[, 1:52]), y = subTrain$classe,
        family = "multinomial", alpha = 0.5, nfolds = 5, type.measure = "class")
modelFit3.cv$lambda.min
```

```
## [1] 4.196407e-05
```

```r
plot(modelFit3.cv)
```



```r
predictedY3 <- predict(modelFit3.cv, newx = as.matrix(subTest[,1:52]),
                       type = "class", s = "lambda.min")
confusionMatrix(subTest$classe, predictedY3) #accuracy = 0.732
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1439   42   69   74   17
##          B  150  715   95   49  106
##          C   90   89  726   65   35
##          D   66   28  109  702   39
##          E   59  132   75   87  705
##
## Overall Statistics
##
##                Accuracy : 0.7439
##                  95% CI : (0.7324, 0.7551)
##     No Information Rate : 0.313
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.675
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7977   0.7107   0.6760   0.7185   0.7816
## Specificity            0.9490   0.9159   0.9405   0.9494   0.9274
## Pos Pred Value         0.8769   0.6413   0.7224   0.7436   0.6664
## Neg Pred Value         0.9115   0.9374   0.9269   0.9429   0.9581
## Prevalence             0.3130   0.1746   0.1864   0.1695   0.1565
## Detection Rate         0.2497   0.1241   0.1260   0.1218   0.1223
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      0.8733   0.8133   0.8082   0.8340   0.8545
```

```
## Turn off PP
stopCluster(cluster)
registerDoSEQ()
```

Best model

So we can see that Random Forest gave the highest accuracy, so that is the model that we select. Accuracy is 0.9911 and Out of sample error is 0.0088 (0.8%). Ideally it would have been better to use 10 or more fold cross validation to get a more robust model, but was not possible. due to RAM restrictions. If we had a 2 class problem, we would have been able to plot ROC curves. But since this is a multi-class problem, I have not plotted ROC curves

```
#Final RF model
modelFit2$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: 27
```

```
##
##         OOB estimate of  error rate: 0.73%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3824    2    2    0    2 0.001566580
## B   20 2575    8    0    0 0.010756819
## C    0   11 2326   10    0 0.008947593
## D    0    2   27 2172    2 0.014071720
## E    0    0    3    9 2458 0.004858300
```

```
#Accuracy
accuracy2 <- postResample(predictedY2, subTest$classe)
accuracy2
```

```
##  Accuracy     Kappa
## 0.9928856 0.9909983
```

```
# Out of sample error = 1-accuracy
oose2 <- 1 - as.numeric(confusionMatrix(subTest$classe, predictedY2)$overall[1])
oose2
```

```
## [1] 0.00711435
```

Predicting on test data

Now that we have chosen the best model, we will apply this model one time on the test data

```
pred <- predict(modelFit2, testClean)
pred
```

```
##  [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
```

Conclusion

Random Forest model gave the highest accuracy (99.1%) with 5 fold cross validation, so that is the model that we select and applied to the test data and gave all correct classes.