Practical Machine Learning Project

Background

> Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, your goal will be to use data 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.

Objective

The goal of this project is to predict the manner in which each subject did their respective exercise. This is the `classe` variable in the training set. This report aims to describe how the model was built, explain the method of cross validation used, provide an estimation of out of sample error, and to provide a justification of model choices. The model built on the training set will then be used to predict 20 different test cases.

Data

The training data is available here:

- https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The testing dats is available here:

- https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Note: The data for this project come from this source:

- http://groupware.les.inf.puc-rio.br/har.

If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Methodology

This analysis will attempt to clasify the data by using three types of robust classification algorithm:

- Random Forest
- Boosting
- Support Vector Machine (Both linear and radial kernal)

Theses models have been selected due to their high accuracy and manageable complexity. There are algorithms that could offer higher accuracy in the `caret` package such as `mxnet`, however due to hardware and time limitations, these may be slightly out of the scope of this project.

These models will be built on a 70% partition of the training dataset and benchmarked against a validation set, before performing predictions against 20 unlabeled test exmples.

These models will be cross validated using the following train control function: `trainControl(method = "cv", number = 5)`. The only other modified parameter is the maximum number of trees allocated to the random forest model which has been limited to 80 in order to reduce computational complexity.

```r
# Load Data
if(!dir.exists("MlProject_data")) {
  dir.create("MlProject_data")
}

if(!file.exists("./MlProject_data/training.csv")){
  download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", "./MlProj
}

if(!file.exists("./MlProject_data/testing.csv")){
  download.file(url = "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", "./MlProje
}

training <- read.csv("./MlProject_data/training.csv", header = TRUE, sep = ",")
testing <- read.csv("./MlProject_data/testing.csv", header = TRUE, sep = ",")
dim(training);dim(testing)
```

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

```
## [1]  20 160
```

```r
# Load Packages
library(caret)
library(tidyverse)
library(randomForest)
```

Data Cleaning and Partitioning

```r
# Data cleaning
# Remove first seven cols of useless info
training[, 1:7] <- NULL
testing[, 1:7] <- NULL

# Remove cols containing more than 90% NA
removeColNa <- which(colSums(is.na(training) | training=="")>0.9*dim(training)[1])
training <- training[, -removeColNa]

removeColNa <- which(colSums(is.na(testing) | testing=="")>0.9*dim(testing)[1])
testing <- testing[, -removeColNa]

dim(training); dim(testing)
```

```
## [1] 19622    53
```

```
## [1] 20 53
```

```r
# Create training and validation sets
inTrain <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
  train <- training[inTrain, ]
  val <- training[-inTrain, ]
```

Build Models and Predict on Validation Set

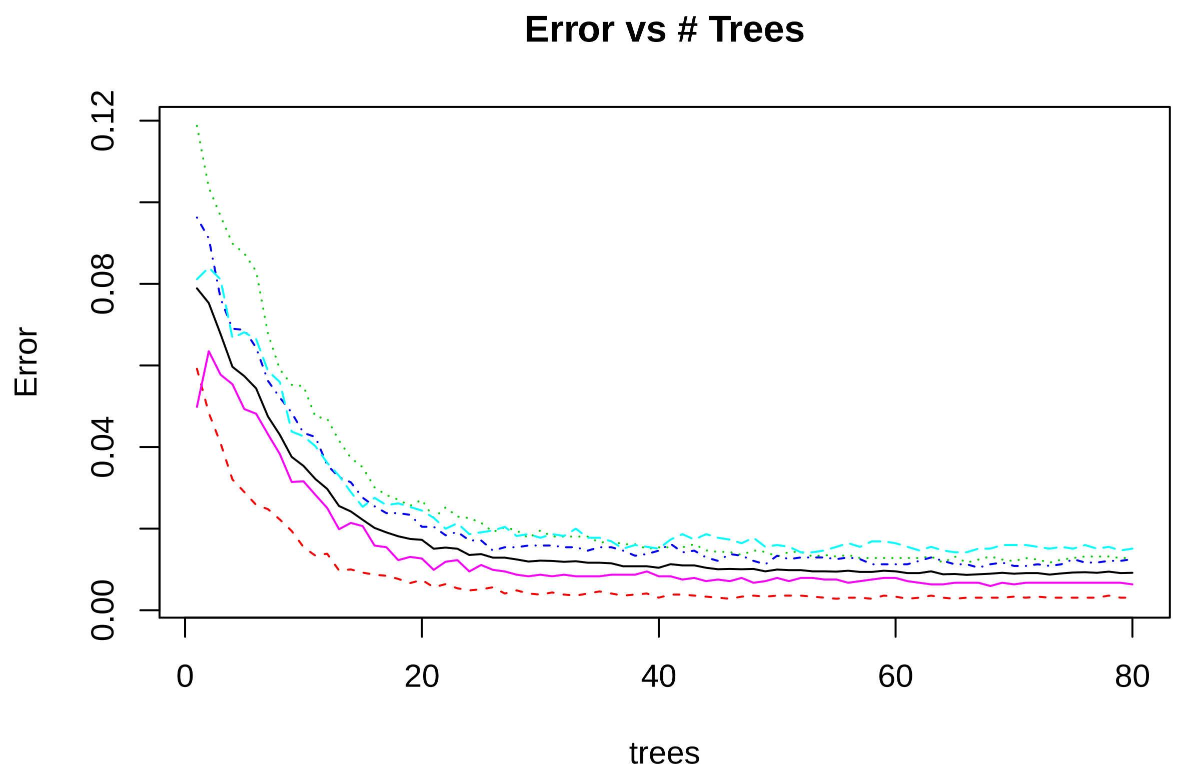
```r
# Specify the use of cross validation
fitControl <- trainControl(method = "cv", number = 5)

# Random forest
mod1 <- train(classe ~., data = train, method = "rf", prox = TRUE, ntree=80, trControl = fitControl)

# Boosting
mod2 <- train(classe ~., data = train, method = "gbm", verbose = FALSE, trControl = fitControl)

# Classification with Linear Support Vector Machine
mod3 <- train(classe ~., data = train, method = "svmLinear", trControl =   fitControl)

# Classification with Radial Support Vector Machine
mod4 <- train(classe ~., data = train, method = "svmRadial", trControl =   fitControl)
```

```r
# prediction for Rf
predRF <- predict(mod1, val)

# Prediction for Gbm
predGbm <- predict(mod2, val)

# Prediction for SVML
predSvmL<- predict(mod3, val)

# Prediction for SVMR
predSvmR <- predict(mod4, val)
```

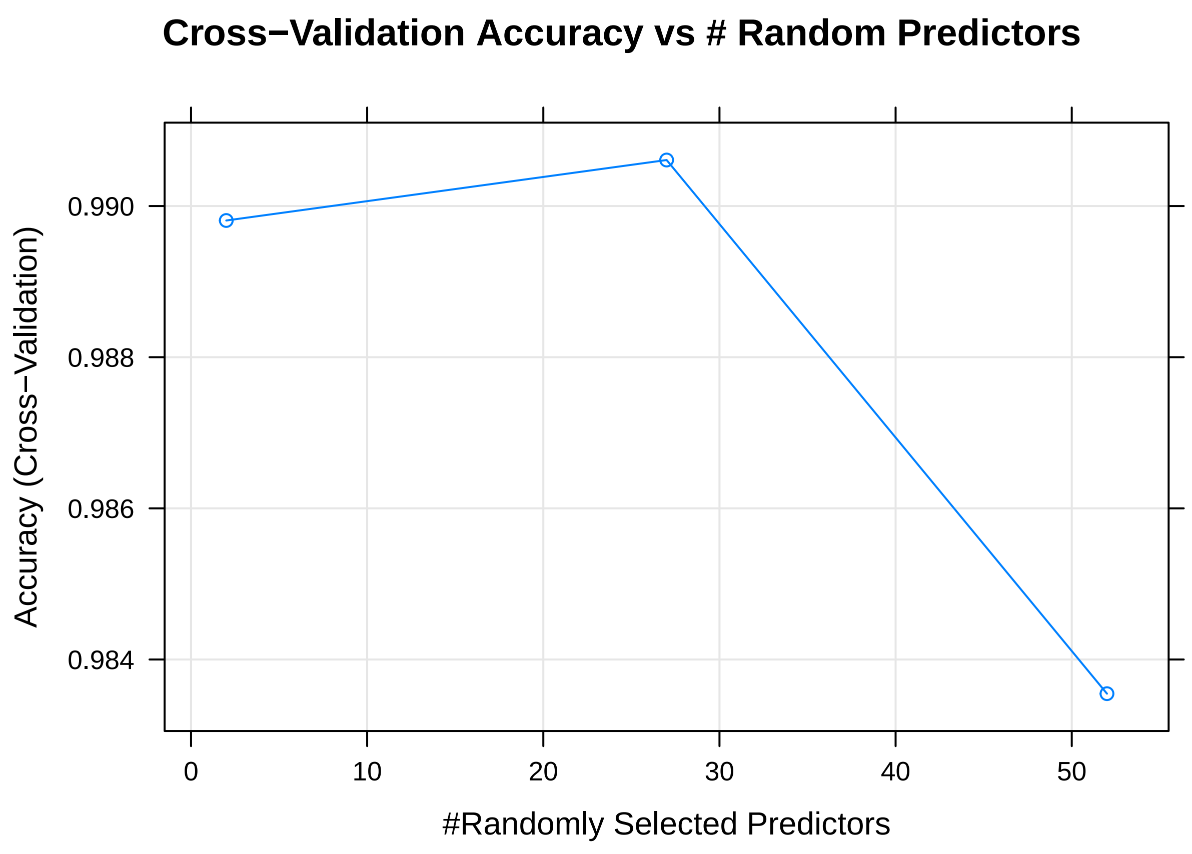
Model Diagnostics

Random Forest Model Diagnostic

```r
# Diagnostics for Random Forest
plot(mod1$finalModel, main = "Error vs # Trees")
```



```r
plot(mod1, main = "Cross-Validation Accuracy vs # Random Predictors")
```



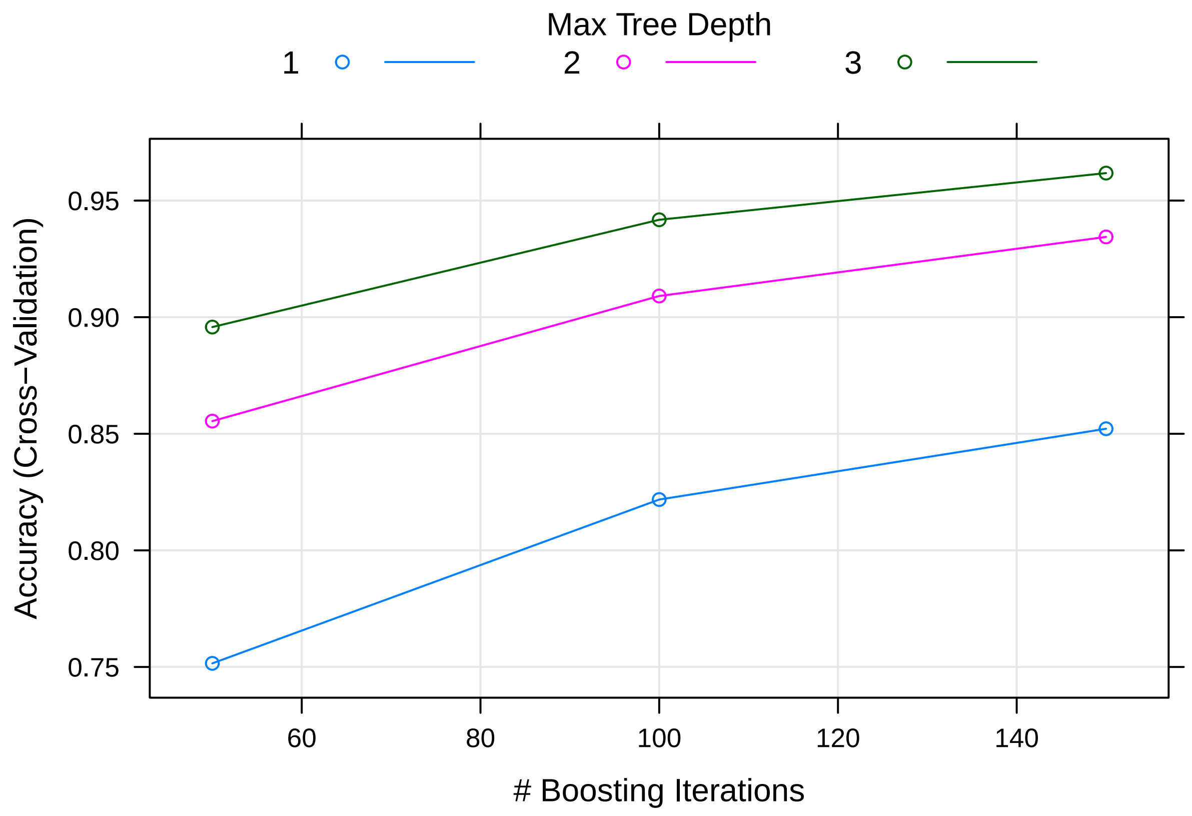
```
confRf <- confusionMatrix(val$classe, predRF)
confRf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    0    2    0    0
##          B    5 1121   12    1    0
##          C    0   12 1011    3    0
##          D    0    1   11  952    0
##          E    0    0    7    4 1071
##
## Overall Statistics
##
##                Accuracy : 0.9901
##                  95% CI : (0.9873, 0.9925)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9885   0.9693   0.9917   1.0000
## Specificity            0.9995   0.9962   0.9969   0.9976   0.9977
## Pos Pred Value         0.9988   0.9842   0.9854   0.9876   0.9898
## Neg Pred Value         0.9988   0.9973   0.9934   0.9984   1.0000
## Prevalence             0.2850   0.1927   0.1772   0.1631   0.1820
## Detection Rate         0.2841   0.1905   0.1718   0.1618   0.1820
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9983   0.9924   0.9831   0.9946   0.9989
```

Boosting Model Diagnostic

```
# Diagnostics for Boosting
plot(mod2)
```



```
confGbm <- confusionMatrix(val$classe, predGbm)
confGbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1648   11   11    3    1
##          B   26 1074   38    1    0
##          C    0   38  971   16    1
##          D    1    3   29  926    5
##          E    4   15   16   15 1032
##
## Overall Statistics
##
##                Accuracy : 0.9602
##                  95% CI : (0.9549, 0.9651)
##     No Information Rate : 0.2853
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9497
##
##  Mcnemar's Test P-Value : 9.138e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9815   0.9413   0.9117   0.9636   0.9933
## Specificity            0.9938   0.9863   0.9886   0.9923   0.9897
## Pos Pred Value         0.9845   0.9429   0.9464   0.9606   0.9538
## Neg Pred Value         0.9926   0.9859   0.9807   0.9929   0.9985
## Prevalence             0.2853   0.1939   0.1810   0.1633   0.1766
## Detection Rate         0.2800   0.1825   0.1650   0.1573   0.1754
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9877   0.9638   0.9502   0.9779   0.9915
```

Support Vector Machine Diagnostic

```
# Diagnostics for Linear Support Vector Machine
mod3$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 7074
##
## Objective Function Value : -1454.691 -1219.319 -1038.813 -624.7436 -1312.259 -877.3315 -1663.805 -11
## Training error : 0.203174
```

```
confSvmL <- confusionMatrix(val$classe, predSvmL)
confSvmL
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1534   33   42   59    6
##          B  128  835   72   26   78
##          C   91   99  764   49   23
##          D   66   35  107  711   45
##          E   67  148   80   59  728
##
## Overall Statistics
##
##                Accuracy : 0.7769
##                  95% CI : (0.766, 0.7875)
##     No Information Rate : 0.3205
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7164
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8134   0.7261   0.7174   0.7865   0.8273
## Specificity            0.9650   0.9358   0.9456   0.9492   0.9293
## Pos Pred Value         0.9164   0.7331   0.7446   0.7376   0.6728
## Neg Pred Value         0.9164   0.9336   0.9381   0.9608   0.9684
## Prevalence             0.3205   0.1954   0.1810   0.1536   0.1495
## Detection Rate         0.2607   0.1419   0.1298   0.1208   0.1237
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8892   0.8309   0.8315   0.8679   0.8783
```

```
# Diagnostics for Radial Support Vector Machine
mod4$finalModel
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0140785508112432
##
## Number of Support Vectors : 6892
##
## Objective Function Value : -1111.004 -822.5836 -737.2596 -438.8847 -1017.047 -578.516 -747.1493 -101
## Training error : 0.06588
```

```
confSvmR <- confusionMatrix(val$classe, predSvmR)
confSvmR
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##           A 1661    6    7    0    0
##           B  101  980   53    3    2
##           C    4   50  951   18    3
##           D    6    1   92  864    1
##           E    2   16   55   26  983
##
## Overall Statistics
##
##                Accuracy : 0.9242
##                  95% CI : (0.9172, 0.9308)
##     No Information Rate : 0.3014
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.904
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9363   0.9307   0.8212   0.9484   0.9939
## Specificity            0.9968   0.9671   0.9841   0.9799   0.9798
## Pos Pred Value         0.9922   0.8604   0.9269   0.8963   0.9085
## Neg Pred Value         0.9732   0.9846   0.9574   0.9904   0.9988
## Prevalence             0.3014   0.1789   0.1968   0.1548   0.1681
## Detection Rate         0.2822   0.1665   0.1616   0.1468   0.1670
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9666   0.9489   0.9027   0.9642   0.9869
```

Prediction on the Test Set

```
# prediction for Rf
predRFT <- predict(mod1, testing)

# Prediction for Gbm
predGbmT <- predict(mod2, testing)

# Prediction for Linear SVM
predSvmL <- predict(mod3, testing)

# Prediction for Radial SVM
predSvmR <- predict(mod4, testing)


predRFT
```

```
##  [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
```

```
predGbmT
```

```
##  [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
```

```
predSvmL
```

```
##  [1] C A B C A E D B A A C A B A E E A B B B
## Levels: A B C D E
```

```
predSvmR
```

```
##  [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

The Kappa values for each model are listed below in rank of accuracy:

1. Random Forest: **Kappa = 0.991**
2. Boosting: **Kappa = 0.957**
3. Support Vector Machine:

- Gaussian Kernal: **Kappa = 0.914**
- Linear Kernal: **Kappa = 0.7353**

As shown above, the most accurate model is the random forest, whereas the least accurate is the support vector machine with the linear kernal.

Both the boosting model and the random forest perfectly agree with eachother when predicting on the testing set, which is not surprising as they are both highly accurate. The support vector machine with the radial kernal also agrees with 80% of the above predictions. From the limited data available in the test set here, the out of sample error can be estimated to be moderately low.

These models could be combined to create an ensemble model, however this will not be necessary as the random forest model already has a large Kappa value of 99.1%.