Practical Machine Learning

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.

The goal of your project is to predict the manner in which they did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

Data

Read Data

Reading in the data while transforming all errors, blanks, NAs into a consistent output.

```
trn<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"), na.strings=c("NA","#DIV/0!",""), header=TRUE)
tst<-read.csv(url("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"), na.strings=c("NA","#DIV/0!",""), header=TRUE)
dim(trn)
```

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

```
dim(tst)
```

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

Cleaning data

Removing columns which include the NA outputs. The first 7 columns add no prediction value to the model. Removing these columns to added processing efficiency.

```
TrnC<-trn[, colSums(is.na(trn)) == 0]
TrnC<-TrnC[, -c(1:7)]
TstC<-tst[, colSums(is.na(tst)) == 0]
```

```
TstC<-TstC[, -c(1:7)]
dim(TrnC)

## [1] 19622    53

dim(TstC)

## [1] 20 53
```

Data Segmentation

Separate the training dataset into training and validation set. The validation set will provide a out of sample error rate to fine tune the model before applying to the test set. Applied a 70%/30% training to validation dataset ratio.

```
set.seed(1234)
library(caret)

## Warning: package 'caret' was built under R version 3.4.1

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.1

InTrain<-createDataPartition(y=TrnC$classe, p=.70, list=FALSE)
Training<-TrnC[InTrain,]
Val<-TrnC[-InTrain,]
```

Machine Learning

Cross Validation

The 5-fold cross validation method was used within the decision tree or random forest models.

```
TranFld<-trainControl(method='cv',number=5, savePredictions = TRUE)
```

Decision Tree

```
Moddt<-train(classe~.,data=Training, trControl=TranFld, method="rpart")

## Loading required package: rpart

## Warning: package 'rpart' was built under R version 3.4.1

Moddt

## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
```

```
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10990, 10988, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03549995  0.5214443  0.38010158
##   0.06092971  0.4175484  0.21093618
##   0.11738379  0.3332562  0.07466867
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03549995.
```

```
Preddt<-predict(Moddt, Val)
CMdt<-confusionMatrix(Preddt, Val$classe)
CMdt
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1530  486  493  452  168
##          B   35  379   31  164  145
##          C  105  274  502  348  302
##          D    0    0    0    0    0
##          E    4    0    0    0  467
##
## Overall Statistics
##
##                Accuracy : 0.489
##                  95% CI : (0.4762, 0.5019)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3311
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9140   0.3327   0.4893   0.0000  0.43161
## Specificity            0.6203   0.9210   0.7882   1.0000  0.99917
## Pos Pred Value         0.4890   0.5027   0.3279      NaN  0.99151
## Neg Pred Value         0.9478   0.8519   0.8797   0.8362  0.88641
## Prevalence             0.2845   0.1935   0.1743   0.1638  0.18386
## Detection Rate         0.2600   0.0644   0.0853   0.0000  0.07935
## Detection Prevalence   0.5317   0.1281   0.2602   0.0000  0.08003
## Balanced Accuracy      0.7671   0.6269   0.6388   0.5000  0.71539
```

Low accuracy at 49% leads to the use for random forest approach.

Random Forest

```
ModRmf<-train(classe~.,data=Training, trControl=TranFld, method="rf",
ntree=10)

## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 3.4.1

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

ModRmf

## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10991, 10989, 10989, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9761223  0.9697870
##   27    0.9863144  0.9826862
##   52    0.9826744  0.9780822
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.

PredRmf<-predict(ModRmf, Val)
CMRmf<-confusionMatrix(PredRmf, Val$classe)
CMRmf

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672   19    0    3    0
##          B    0 1110    9    3    5
```

```
##          C    2    5 1012    8    5
##          D    0    3    5  946    4
##          E    0    2    0    4 1068
##
## Overall Statistics
##
##                Accuracy : 0.9869
##                  95% CI : (0.9837, 0.9897)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9834
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9745   0.9864   0.9813   0.9871
## Specificity            0.9948   0.9964   0.9959   0.9976   0.9988
## Pos Pred Value         0.9870   0.9849   0.9806   0.9875   0.9944
## Neg Pred Value         0.9995   0.9939   0.9971   0.9963   0.9971
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1886   0.1720   0.1607   0.1815
## Detection Prevalence   0.2879   0.1915   0.1754   0.1628   0.1825
## Balanced Accuracy      0.9968   0.9855   0.9911   0.9894   0.9929
```

Accuracy has increased to 99% using a 10 tree random forest.

Applied Random Forest on Test set

```
TstRmf<-predict(ModRmf, TstC)
TstRmf
```

```
##  [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
```