Practical Machine Learning Project

Alexander M Fisher

November 26 2020

Executive Summary:

In this study we will generate a prediction model using the data from Human Activity Recognition Project at Groupware. Using the data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants, a model will be trained to predict the manner in which they did the exercise. Then it will be applied on the test data to predict the manner of exercise under 20 different test cases. For more information on the data go to Groupware (see the section on the Weight Lifting Exercise Dataset).

Load and Prepare Data:

The first step is to, if not already done, download the training and testing sets, and then load them into R. A preliminary "clean up" will also be preformed. The main steps taken in the code chunk below are,

- download and load data into R
- remove any variable (i.e column) if it isnt 95% complete, that is to say not full of NAs
- remove first 7 columns which are related to identification, not quantitative data.
- split training set into training and validation by 0.7 split.

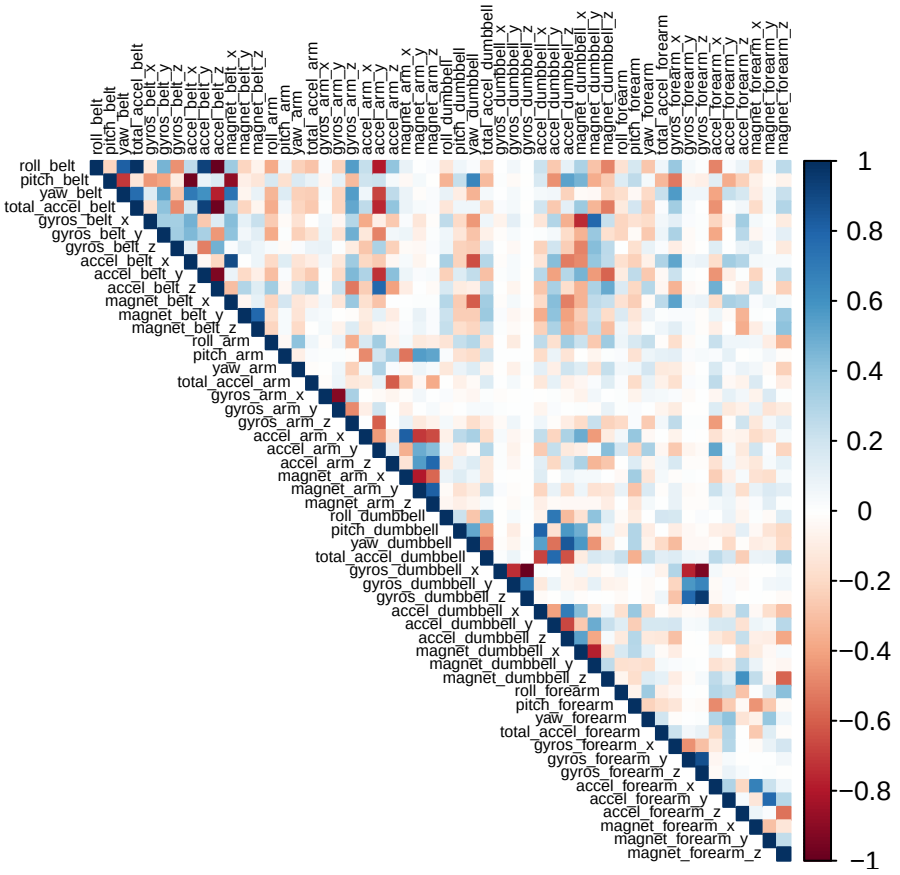
```
test_data_file <- "pml-testing.csv" ; train_data_file <- "pml-training.csv"
test_data_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
train_data_url <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
if (!file.exists(test_data_file)){
        download.file(test_data_url, destfile=test_data_file, method="curl")
}
if (!file.exists(train_data_file)){
        download.file(train_data_url, destfile=train_data_file, method="curl")
}

training_data <- read.csv("pml-training.csv", na.strings= c("NA","","#DIV/0!"))
testing_data <- read.csv("pml-testing.csv", na.strings= c("NA","","#DIV/0!"))
training <- training_data[, !colMeans(is.na(training_data)) > 0.95]
testing <- testing_data[, !colMeans(is.na(training_data)) > 0.95]
training$classe <- as.factor(training$classe)
training <- training[, -(1:7)]
testing <- testing[, -(1:7)]
suppressMessages(library(caret)); set.seed(222)
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
training <- training[inTrain, ]
validation <- training[-inTrain, ]
```

This leaves three data sets. A training, validation, and testing set. All have 53 variables or features. Now to do some exploration. Functions `str`, `summary`, `head` can be used to have a look at the data set (I have not printed them out here to conserve page space).

Exploratory Data Analysis:

This is a quick section making a plot that looks at how the variables are correlated with each other. It can be seen in the resulting graph that there are some variables that are highly correlated with each other (other than the diagonal entries of course which are 1). There are not too many however. PCA may be applied to reduce the feature space, in this analysis as the feature space is relatively small PCA hasn't been applied, it is worth an investigation however, in a deeper analysis. See below the code chunk and resulting graph.

```
correlation_matrix <- cor(training[,-ncol(training)])
corrplot::corrplot(correlation_matrix, method = "color" , type = "upper", tl.cex = 0.5, tl.col = "black
```



Prediction Model Selection:

In this section I will generate three prediction models, namely, a Decision Tree, and Random Forest, and a Generalized Boosted Model. These will be run on the validation set after being trained on the training set, an the best preforming model will be selected to then predict on the test set. I will load the necessary libraries here.

```
library(caret); library(rpart)
```

```
model_dt <- rpart(classe~., training)
predict_dt <- predict(model_dt, validation, type = "class")
confusion_dt <- confusionMatrix(predict_dt, validation$classe); confusion_dt
```

Decision Tree:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1061  172   15   81   46
##          B   20  453   44   27   54
##          C   21   85  581   96   69
##          D   35   66   52  423   35
##          E   34   32   19   48  530
##
## Overall Statistics
##
##                Accuracy : 0.7436
##                  95% CI : (0.7299, 0.7569)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6737
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9061   0.5606   0.8172   0.6267   0.7221
## Specificity            0.8928   0.9559   0.9200   0.9451   0.9605
## Pos Pred Value         0.7716   0.7575   0.6819   0.6923   0.7994
## Neg Pred Value         0.9596   0.8986   0.9600   0.9278   0.9406
## Prevalence             0.2857   0.1971   0.1735   0.1647   0.1791
## Detection Rate         0.2588   0.1105   0.1417   0.1032   0.1293
## Detection Prevalence   0.3354   0.1459   0.2079   0.1491   0.1617
## Balanced Accuracy      0.8994   0.7583   0.8686   0.7859   0.8413
```

A decision tree has been trained and has predicted the classes from the validation set. I have also got a confusion matrix comparing the truth from the predictions. The accuracy of of this decision tree model on the validation set can be seen to be approx. 75%. Now to move onto the next model.

```
control <- trainControl(method = "cv", number = 3, verboseIter=FALSE)
model_rf <- train(classe ~ ., data = training, method = "rf", trControl = control, ntree = 50)
model_rf$finalModel
```

Random Forest:

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 50, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.95%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    7    1    0    0 0.002048131
```

```
## B   29 2613   16    0    0 0.016930023
## C    0   15 2369   12    0 0.011268781
## D    1    4   27 2217    3 0.015541741
## E    0    2    4    9 2510 0.005940594
```

Here a Random Forest has been trained. The final model has been printed out. Again now I will use the model to predict the classe value for the for each of the validation set entries. A confusion matrix is printed out as well, and it can be seen the accuracy of this random forest model is 100%.

```
predict_rf <- predict(model_rf, validation)
confusion_rf <- confusionMatrix(predict_rf, validation$classe); confusion_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1171    0    0    0    0
##          B    0  808    0    0    0
##          C    0    0  711    0    0
##          D    0    0    0  675    0
##          E    0    0    0    0  734
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2857   0.1971   0.1735   0.1647   0.1791
## Detection Rate         0.2857   0.1971   0.1735   0.1647   0.1791
## Detection Prevalence   0.2857   0.1971   0.1735   0.1647   0.1791
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Generalised Boosted Model: This is the final model in this analysis.

```
control <- trainControl(method = "repeatedcv", number = 3, repeats = 1, verboseIter = FALSE)
model_gbm <- train(classe ~ ., data = training, method = "gbm", trControl = control, verbose = FALSE); m
```

```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 52 predictors of which 52 had non-zero influence.
```

Again I will use the model to predict the classe values for the validation set and then check the accuracy. The accuracy of this model on the validation set is approx. 97%.

```
predict_gbm <- predict(model_gbm, validation)
confusion_gbm <- confusionMatrix(predict_gbm, validation$classe); confusion_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1163   18    0    0    0
##          B    5  769   14    1    6
##          C    1   21  688   21    8
##          D    2    0    7  652    6
##          E    0    0    2    1  714
##
## Overall Statistics
##
##                Accuracy : 0.9724
##                  95% CI : (0.9669, 0.9772)
##     No Information Rate : 0.2857
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9651
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9932   0.9517   0.9677   0.9659   0.9728
## Specificity            0.9939   0.9921   0.9849   0.9956   0.9991
## Pos Pred Value         0.9848   0.9673   0.9310   0.9775   0.9958
## Neg Pred Value         0.9973   0.9882   0.9932   0.9933   0.9941
## Prevalence             0.2857   0.1971   0.1735   0.1647   0.1791
## Detection Rate         0.2837   0.1876   0.1678   0.1591   0.1742
## Detection Prevalence   0.2881   0.1939   0.1803   0.1627   0.1749
## Balanced Accuracy      0.9935   0.9719   0.9763   0.9808   0.9859
```

The best model therefore out of the three, is the random forest.

Prediction on Testing Data Set:

The final thing done in this analysis is predicting the classe values in the test set.

```
### Predicting on Test Set
predict_test <- predict(model_rf, testing); predict_test
```

```
##  [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
```