Practical Machine Learning Course Project

Miranda L. Gardner

2020-10-29

1. BACKGROUND

As provided by the coursera.org website and instructions for this assignment:

"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. More information is available from the website here: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset)."

2. DATA

The training data for this report can be accessed at the link below:
Training Data

The testing data for this report can be accessed at the link below:
Testing Data

2.1 Load Libraries for Data Analysis

The following packages are loaded into the R environment for downstream data analysis:

```
library(caret)
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
library(randomForest)
library(corrplot)
library(gbm)
```

2.2 Load and Examine Data

The data files are read into two variables, train and valid, for the training and testing CSVs respectively.

```
train <- read.csv("pml-training.csv")
dim(train)
```

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

```
valid <- read.csv("pml-testing.csv")
dim(valid)
```

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

As seen above, the raw data contains 19622 observations of 160 variables in the training set and 20 observations of 160 variables in the testing set.

2.3 Clean the Data

In this section, we will first remove the variables containing all missing values and additionally omit the first seven variables from analysis as they are more qualitative and will have little influence on how the data is classified.

```
train2 <- train[, colSums(is.na(train)) == 0]
valid2 <- valid[, colSums(is.na(valid)) == 0]

trainData <- train2[, -c(1:7)]
validData <- valid2[, -c(1:7)]
dim(trainData)
```

```
## [1] 19622    86
```

```
dim(validData)
```

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

As seen above, the filtered data contains 19622 observations of 86 variables in the training set and 20 observations of 53 variables in the testing set.

2.4 Partition the Data for Prediction

The training data will be split into 70% train and 30% test to minimize bias and overfitting and to compute the out-of-sample errors

```
set.seed(060301)
inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
trainData <- trainData[inTrain, ]
testData <- trainData[-inTrain, ]
dim(trainData)
```

```
## [1] 13737    86
```

Variables with near-zero variance are removed as they will not be important factors in modeling the data.

```
NZV <- nearZeroVar(trainData)
trainData <- trainData[, -NZV]
testData  <- testData[, -NZV]
dim(trainData)
```

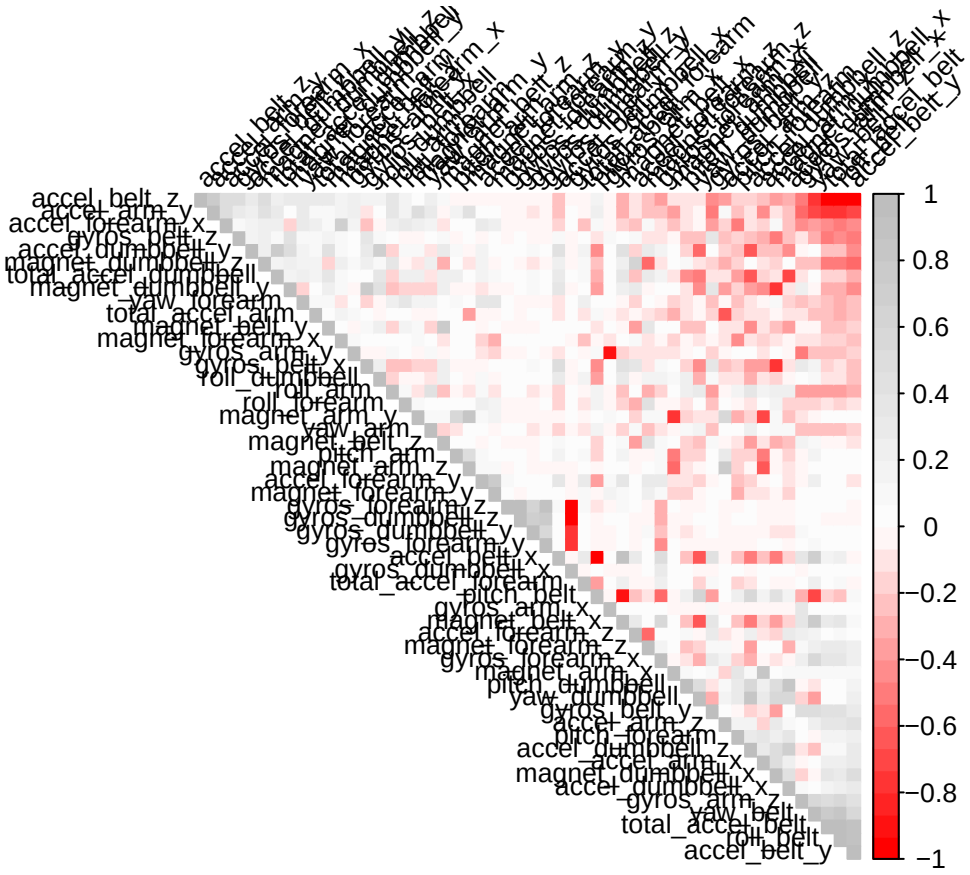
```
## [1] 13737    53
```

As seen above, this results in a final dataset of 53 variables.

The corrplot demonstrates the variables that are most highly correlated (darkest colors).

```
cor_mat <- cor(trainData[, c(-53)])
col<- colorRampPalette(c("red", "white", "gray"))(30)

corrplot(cor_mat, order = "FPC", method = "color", type = "upper",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0), tl.srt=45, col = col)
```



These variables can be filtered out of the data with the following block of code.

```
highlyCorrelated = findCorrelation(cor_mat, cutoff=0.80)
names(trainData)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_dumbbell_z"
##  [5] "accel_belt_x"     "pitch_belt"       "accel_dumbbell_x" "accel_arm_x"
##  [9] "magnet_arm_y"     "gyros_forearm_y"  "gyros_dumbbell_x" "gyros_dumbbell_z"
## [13] "gyros_arm_x"
```

3. MODEL BUILDING

I tested three different models to find the best algorithm for outcome prediction as assessed by the highest accuracy and lowest out-of-sample error. The three models tested were:

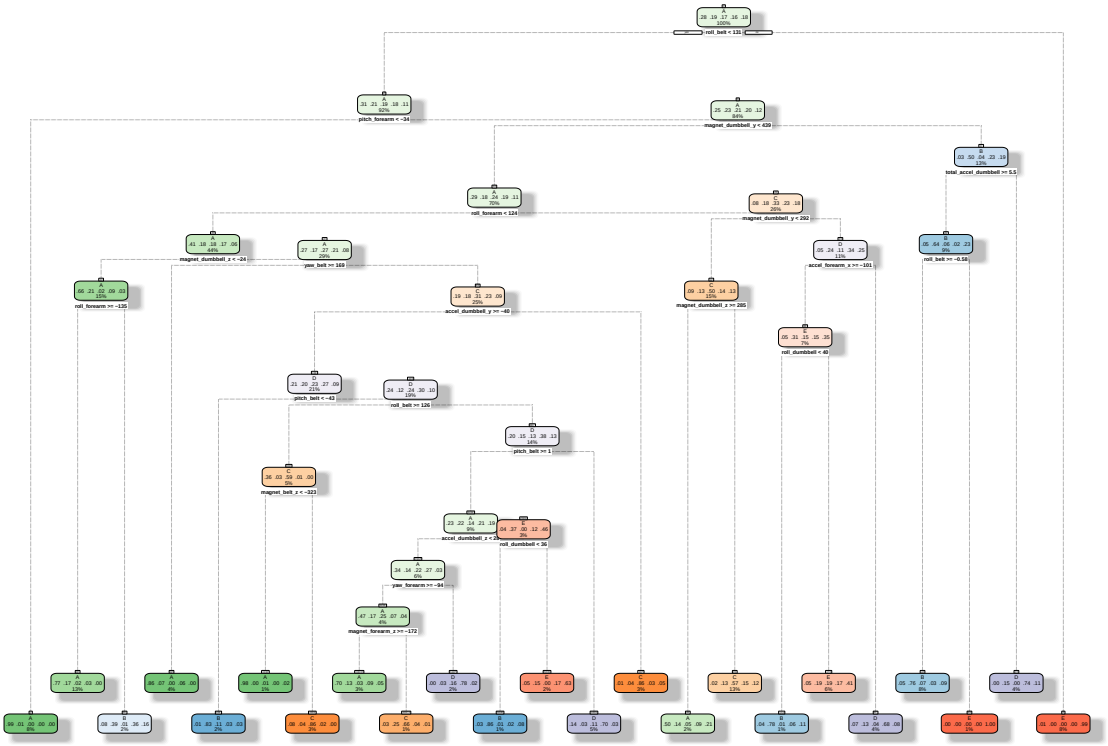
1. Decision trees with rpart
2. Gradient boosting with gbm
3. Random forest with rf

3.1 Prediction with decision trees

Obtain the model and plot as a dendogram.

```
set.seed(120579)
dec_tree <- rpart(classe ~ ., data=trainData, method="class")
fancyRpartPlot(dec_tree)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020−Nov−01 13:11:32 Red

Validate model with testing data and assess performance as measured by accuracy.

```
pred_tree <- predict(dec_tree, testData, type = "class")
cm_tree <- confusionMatrix(pred_tree, testData$classe)
cm_tree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1046  118   19   53   36
##          B   38  456   28   47   52
##          C   22  106  595   89   60
##          D   55   54   35  433   41
##          E   12   58   40   53  580
##
## Overall Statistics
##
##                Accuracy : 0.7538
##                  95% CI : (0.7403, 0.7668)
```

```
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6878
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8917   0.5758   0.8298   0.6415   0.7542
## Specificity            0.9235   0.9505   0.9187   0.9464   0.9514
## Pos Pred Value         0.8223   0.7343   0.6823   0.7006   0.7806
## Neg Pred Value         0.9555   0.9041   0.9625   0.9310   0.9441
## Prevalence             0.2843   0.1920   0.1738   0.1636   0.1864
## Detection Rate         0.2535   0.1105   0.1442   0.1049   0.1406
## Detection Prevalence   0.3083   0.1505   0.2113   0.1498   0.1801
## Balanced Accuracy      0.9076   0.7631   0.8743   0.7939   0.8528
```

The decision tree accuracy is 75.4% while the out-of-sample error is 24.6%.

3.2 Prediction with generalized boosting model

```
set.seed(98765)
ctrl_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
mod_gbm  <- train(classe ~ ., data=trainData, method = "gbm",
                  trControl = ctrl_gbm, verbose = FALSE)
mod_gbm$finalModel
```

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

Validate the model and assess performance with accuracy calculations

```
pred_gbm <- predict(mod_gbm, newdata=testData)
cm_gbm <- confusionMatrix(pred_gbm, testData$classe)
cm_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1165   21    0    2    1
##          B    6  756   14    2    4
##          C    0   15  696   17    5
##          D    1    0    5  650    8
##          E    1    0    2    4  751
##
## Overall Statistics
##
##                Accuracy : 0.9738
##                  95% CI : (0.9685, 0.9785)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.9669
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9932   0.9545   0.9707   0.9630   0.9766
## Specificity            0.9919   0.9922   0.9891   0.9959   0.9979
## Pos Pred Value         0.9798   0.9668   0.9495   0.9789   0.9908
## Neg Pred Value         0.9973   0.9892   0.9938   0.9928   0.9947
## Prevalence             0.2843   0.1920   0.1738   0.1636   0.1864
## Detection Rate         0.2824   0.1832   0.1687   0.1575   0.1820
## Detection Prevalence   0.2882   0.1895   0.1777   0.1609   0.1837
## Balanced Accuracy      0.9925   0.9734   0.9799   0.9795   0.9873
```

The gbm model accuracy is 97.4% while the out-of-sample error is 2.6%.

3.3 Prediction with random forest modeling

```
set.seed(01031975)
ctrl_rf <- trainControl(method="cv", number=3, verboseIter=FALSE)
mod_rf <- train(classe ~ ., data=trainData, method="rf", trControl=ctrl_rf)
mod_rf$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: 0.66%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3900    4    1    0    1 0.001536098
## B   15 2633    8    1    1 0.009405568
## C    0   14 2373    9    0 0.009599332
## D    0    2   20 2228    2 0.010657194
## E    0    0    5    7 2513 0.004752475
```

Validate the model and assess performance with accuracy calculations

```
pred_rf <- predict(mod_rf, newdata=testData)
cm_rf <- confusionMatrix(pred_rf, testData$classe)
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1173    0    0    0    0
##          B    0  792    0    0    0
##          C    0    0  717    0    0
```

```
##          D    0    0    0  675    0
##          E    0    0    0    0  769
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2843
##     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    1.000   1.0000   1.0000   1.0000
## Specificity            1.0000    1.000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000    1.000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000    1.000   1.0000   1.0000   1.0000
## Prevalence             0.2843    0.192   0.1738   0.1636   0.1864
## Detection Rate         0.2843    0.192   0.1738   0.1636   0.1864
## Detection Prevalence   0.2843    0.192   0.1738   0.1636   0.1864
## Balanced Accuracy      1.0000    1.000   1.0000   1.0000   1.0000
```

The random forest model accuracy is 100% while the out-of-sample error is 0%.

4. Conclusions

```
acc_res <- data.frame(
  Model = c('CART', 'GBM', 'RF'),
  Accuracy = rbind(cm_tree$overall[1], cm_gbm$overall[1], cm_rf$overall[1])
)
print(acc_res)
```

```
##   Model  Accuracy
## 1  CART 0.7537567
## 2   GBM 0.9738245
## 3    RF 1.0000000
```

The decision tree has the lowest accuracy at 0.75, followed by gbm at 0.97 and the most accurate model is random forest at 1. This model could be overfitting the data and may warrent more investigation.

5. Apply Model to Validation Data

```
final_val <- predict(mod_rf, newdata=validData)
final_val
```

```
##  [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
```