Practical Machine Learning: Course Project

Alejandro Osorio

August, 2018

Executive Summary

Model Selection

Given the multiple-outcome-classification problem (more than 2 possible factor-type outcomes), the following model types were tried separately.

1. Penalized Multinomial Logistic Regression
2. Trees
3. Random Forests
4. Boosting with Trees

After obtaining results for each model, a final GAM model based on the combination of two of the previous ones (the ones with best accuracy) was also tested.

Data Splitting and Cross Validation

The following three sets were created, using Random Subsampling:

1. Building Data Set: 70% of the working dataset. This set was sub divided again into:

- Training Set (70%)
- Testing Set (30%)

2. Validation Set: 30% of the working dataset.

The validation set was prepared for the final testing of the GAM based combination model.

For further details on the steps taken and the obtained results, refer to Appendix 2: Data Splitting.

Feature Selection

Feature selection was based on the following 2 criteria:

1. Examining the variables available in the testing set. All variables (100) with mainly NAs (100% each) in the testing set, were also discarded in the training set. The resulting data sets went down to 59 columns each. More details both on this point and on how the final data sets were obtained, in Appendix 1: Getting and Cleaning Data.
2. Based on a p-value analysis on the coefficients of model 1 (Penalized Multinomial Logistic Regression), the final regressors were obtained for the rest of the models. Further details on steps taken and final data set obtained, in Appendix 3: Selection of Regressors Through p-Values Analysis.

Error Type Selection

Given the problem has more than two possible outcomes, plus the objective of the project, the main error type used for each model was Accuracy (below 90% was considered low).

Results Obtained

Performance obtained by each model, with final features selected, was as follows:

1. Multinomial Logistic Regression (Appendix 4.1): 67%
2. Trees (Appendix 4.2): 58%
3. Random Forests (Appendix 4.3): 99%
4. Boosting with Trees (Appendix 4.4): 96%
5. Combination of Predictors with GAM (Appendix 4.5): 48%

Model 5 didn't converge due to one of the predictor variables separating perfectly the values of the dependent variable (more details in Appendix 4.5). Therefore, the final selected model, tested again using the validation set, was Random Forest. Its final attained accuracy was, again, 99% (Appendix 5). Given that result, the quiz was answered using said model, obtaining a 95% final grade.

Appendix 1: Getting and Cleaning Data

Training and testing sets were downloaded using read_csv function, with the 'na' parameter tweaked so that it also included "#DIV/0!" cases (besides the default "" and "NA" cases). Additionally, first columns containing row numbers were eliminated from the obtained datasets.

Next, columns from the testing set that included NAs, where identified and dimensioned the following way:

```
table(colSums((is.na(origtest))))
```

```
##
##   0  20
##  59 100
```

Therefore, as all columns with NAs (100 in this case) had 100% NAs (20 each), those columns were removed both from the testing and training datasets.

```
datacols <- names(which(colSums(is.na(origtest))!=0))
origtest <- select(origtest, -datacols)
origtrain <- select(origtrain, -datacols)
```

Then, a final row check for NAs was performed for the training set

```
which(colSums(is.na(origtrain)) != 0)
```

```
## magnet_dumbbell_z  magnet_forearm_y  magnet_forearm_z
##                45                 57                58
```

```
which(rowSums(is.na(origtrain)) != 0)
```

```
## [1] 5373
```

Finally, after removing the only row with the 3 remaining NAs from the training set, the obtained working set is checked for NAs for the last time:

```
workingset <- origtrain[-which(rowSums(is.na(origtrain)) != 0),]
anyNA(workingset)
```

```
## [1] FALSE
```

The final working dataset, with no NAs, went down to the following dimensions (detailed structure of the obtained dataset can be seen in Appendix 1):

```
dim(workingset)
```

```
## [1] 19621    59
```

Finally, the structure of the obtained training dataset was very similar to that of the testing set, but for the last column ('question' v/s 'classe').

Appendix 2: Data Splitting

Step 1: Creating building and validation sets

```
indbuild <- createDataPartition(workingset$classe, p = 0.7, list = FALSE)
validata <- workingset[-indbuild,]
buildata <- workingset[indbuild,]
```

Step 2: Creating training and testing sets, out of the building set

```
indtrain <- createDataPartition(buildata$classe, p = 0.7, list = FALSE)
training <- buildata[indtrain,]
testing <- buildata[-indtrain,]
```

Appendix 3: Selection of Regressors Through p-Values Analysis

As mentioned earlier, a p-value analysis of a Penalized Multinomial Logistic Regression's coefficients was used in order to determine which potential regressors to discard from further analysis.

Training

Model used:

```
## train.formula(form = classe ~ ., data = training[, -c(1:6)],
##     method = "multinom")
```

Just out of curiosity, the model's overall results with 100% of the potential regressors, was:

```
##           Reference
## Prediction   A   B   C   D   E
##          A 981 117  95  57  62
##          B  63 401  63  11  82
##          C  46 117 469 113  77
##          D  77  39  73 462  88
##          E   4 123  18  32 448
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.704711e-01   5.818925e-01   6.558763e-01   6.848237e-01   2.843613e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   3.099710e-37
```

Even though the obtained accuracy was considered low (below 90%), this model's main objective was to determine the regressors to use with the rest of models. Therefore the following p-value analysis.

P-Values for feature selection

T statistics

Given the fact that null hypotheses consider coefficients equal to cero, the t statistics are just the estimates divided by their standard errors, as follows:

```
modfitmlog_coef <- summary(modfitmlog)$coefficients
modfitmlog_stderr <- summary(modfitmlog)$standard.errors
tBetas <- modfitmlog_coef/modfitmlog_stderr
```

Degrees of freedom

```
modfitmlog_edf <- modfitmlog$finalModel["edf"]
n <- length(training$classe)
df <- n - modfitmlog_edf[[1]]
df
```

```
## [1] 9407
```

P values

```
pBetas <- 2 * data.frame(pt(abs(tBetas), df = df, lower.tail = FALSE))
```

Final feature selection

Features eliminated due to high p-values for all possible outcome factors:

```
lesscoeff <- select(pBetas, contains("gyros"))
names(lesscoeff)
```

```
##  [1] "gyros_belt_x"     "gyros_belt_y"     "gyros_belt_z"
##  [4] "gyros_arm_x"      "gyros_arm_y"      "gyros_arm_z"
##  [7] "gyros_dumbbell_x" "gyros_dumbbell_y" "gyros_dumbbell_z"
## [10] "gyros_forearm_x"  "gyros_forearm_y"  "gyros_forearm_z"
```

Therefore, the final formula to be used with all models, was the following:

```
namelist <- names(training[, -c(1:6, 59)])
lessfeatures <- reformulate(termlabels = namelist[-grep("gyro", namelist, value = FALSE)], response = '
lessfeatures
```

```
## classe ~ roll_belt + pitch_belt + yaw_belt + total_accel_belt +
##     accel_belt_x + accel_belt_y + accel_belt_z + magnet_belt_x +
##     magnet_belt_y + magnet_belt_z + roll_arm + pitch_arm + yaw_arm +
##     total_accel_arm + accel_arm_x + accel_arm_y + accel_arm_z +
##     magnet_arm_x + magnet_arm_y + magnet_arm_z + roll_dumbbell +
##     pitch_dumbbell + yaw_dumbbell + total_accel_dumbbell + accel_dumbbell_x +
##     accel_dumbbell_y + accel_dumbbell_z + magnet_dumbbell_x +
##     magnet_dumbbell_y + magnet_dumbbell_z + roll_forearm + pitch_forearm +
##     yaw_forearm + total_accel_forearm + accel_forearm_x + accel_forearm_y +
##     accel_forearm_z + magnet_forearm_x + magnet_forearm_y + magnet_forearm_z
```

Appendix 4: Model Training and Performance Evaluation

Appendix 4.1: Multinomial Logistic Regression Model

Model trained

```
## train.formula(form = lessfeatures, data = training[, -c(1:6)],
##     method = "multinom")
```

Performance obtained

```
predmlog2 <- predict(modfitmlog2, newdata = testing[,-c(1:6, 59)])
confmatmlog2 <- confusionMatrix(predmlog2, factor(testing$classe))
confmatmlog2$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 968 115  99  53  65
##          B  65 408  60  10  74
##          C  51 120 472 118  84
##          D  82  38  72 465  89
##          E   5 116  15  29 445
```

```
confmatmlog2$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   6.697426e-01   5.811737e-01   6.551403e-01   6.841037e-01   2.843613e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   3.593850e-42
```

Interesting to note that almost the same performance was obtained with 12 less features than the original model. Therefore, the remaining models were trained with these same regressors.

Appendix 4.2: Predicting with Trees

Model trained

```
## train.formula(form = lessfeatures, data = training[, -c(1:6)],
##     method = "rpart")
```

Performance obtained

```
confmattree <- confusionMatrix(predtree, factor(testing$classe))
confmattree$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 733 133  27  37  11
##          B 183 487  71 223 195
##          C 229 167 606 219 180
##          D  10  10  14 196   9
##          E  16   0   0   0 362
```

```
confmattree$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   5.789218e-01   4.702378e-01   5.636690e-01   5.940626e-01   2.843613e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00  1.768228e-204
```

It's worth noting that the Multinomial Logistic Regression model was almost ten percentage points more accurate than this one. Just in case, training and predicting was repeated with this model, including all features and exactly the same result was obtained, thus showing so far a correct p-value analysis for regressor selection.

Appendix 4.3: Random Forest

Model trained

```
## train.formula(form = lessfeatures, data = training[, -c(1:6)],
##     method = "rf")
```

Performance obtained

```
confmatrf <- confusionMatrix(predrf, factor(testing$classe))
confmatrf$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1167    7    0    0    0
##          B    4  783    7    1    2
##          C    0    4  707   15    3
##          D    0    3    4  658    2
##          E    0    0    0    1  750
```

```
confmatrf$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9871297      0.9837194      0.9831987      0.9903448      0.2843613
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Excelent performance, with only the regressors selected post p-value analysis.

Appendix 4.4: Boosting With Trees

Model trained

```
## train.formula(form = lessfeatures, data = training[, -c(1:6)],
##     method = "gbm")
```

Performance obtained

```
confmatgbm <- confusionMatrix(predgbm, factor(testing$classe))
confmatgbm$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1152   32    0    1    3
##          B   15  738   28    7   15
##          C    1   26  680   24    5
##          D    2    1    7  641    6
##          E    1    0    3    2  728
```

```
confmatgbm$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.565323e-01   9.449877e-01   9.498511e-01   9.625556e-01   2.843613e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   1.779706e-05
```

Finally, Random Forest obtained a better accuracy than this model. Just in case, again, the same modelling was carried on with all possible regressors. Almost the same results were obtained. Therefore, p-value analysis was indeed a useful method for selecting final regressors in this problem.

Appendix 4.5: Combining Predictors

Based on the accuracy obtained with each previous model, Random Forests and Boosting with Trees were chosen for this stage.

Training data set

```
dfpredcomb <- data.frame(predrf, predgbm, classe = factor(testing$classe))
```

The following data set for GAM training was obtained from each model's predictions ('rf' and 'gbm') plus the real results in the testing set ('real').

```
## 'data.frame':    4118 obs. of  3 variables:
##  $ predrf : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ predgbm: Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ classe : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Model trained

```
## train.formula(form = classe ~ ., data = dfpredcomb, method = "gam")
```

Performance obtained

```
confmatcomb <- confusionMatrix(predcomb, factor(testing$classe))
confmatcomb$table
```

```
##            Reference
## Prediction    A    B    C    D    E
##          A 1167    7    0    0    0
##          B    4  790  718  675  757
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0    0    0    0
```

```
confmatcomb$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   4.752307e-01   3.277079e-01   4.598789e-01   4.906176e-01   2.843613e-01
## AccuracyPValue  McnemarPValue
##  1.022483e-147            NaN
```

The warning message "fitted probabilities numerically 0 or 1 occurred" was generated for all combinations of predictors, besides de two presented here. Additionally, it happened with methods 'gamLoess' and 'gamSpline'. After some research, it means the function didn't converge due to one of the predictor variables separating perfectly the values of the dependent variable.

Final model used, then, was Random Forest, with an accuracy of almost 99%.

Appendix 5: Final Model's Performance

As said previously, due to non converging GAM combining model, the final model tested with Validation Data, was Random Forest.

Final obtained performance:

```
confmatrf_val <- confusionMatrix(predrf_val, factor(validata$classe))
confmatrf_val$table
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1666   16    0    0    0
##          B    7 1114   19    0    0
##          C    0    8 1003   25    1
##          D    0    1    4  937    5
##          E    0    0    0    2 1076
```

```
confmatrf_val$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9850442      0.9810788      0.9816061      0.9879882      0.2843304
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

Good to see an accuracy of almost 99% was accomplished again. Sweet odds for the quiz!