Course 8 - Project Write up

Robert Deprizio

9/30/2018

Preliminaries

Before beginning running the code, I used Len Greski's very helpful tutorial on how to improve the performance of the "train" function in the Caret package. This served to greatly boost the performance when generating the models. (Link below)

https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md

Model Creation - Pre-Preprocessing

A great deal of effort went into pre-processing the data before my final model was chosen (a Random Forest model - listed as "model_1"in my code). I first imported both the test and training data using the read.csv function and accounting for null values such as "#DIV/0!", "NA", and blank spaces by using the "na.strings" setting. This later served as an important step in that such null values can create problems when using the data to build potential models.

I then attempted to look for variables with little or no data, then removed them, creating a new data set that I would work with to create my own training, testing, and validation sets "data_for_model". This left me with 59 potential predictors and the "classe" variable.

Next, I looked for variables considered to be near zero predictors using the "nearZeroVar" function in the Caret package. None of the remaining variables seemed to fit the criteria.

I then stored the variable names which allowed me to keep the variables that my models would use. This helped me prep the raw test data set that would be used for the predictions for the quiz in that it allowed me to extract only the variables my model would be using.

Model Creation - Building the training, testing, and validation sets

Separate testing, training and validation data sets were created from the pre-processed data set "data_for_model" to help build and test for our potential models. 30% of the pre-processed training data "data_for_model" was used as a validation data set to test the potential models and the remaining 70% was used as training and testing sets to build 3 potential models including a stacked model "model_3".

A validation set was created as the stacked model I planned to create was to be trained on predictions given by models 1 and 2 using the testing data. Not creating a separate data set would create an inaccurate representation of the out of sample error when comparing all 3 models at the end.

Model Creation - Actual Building & Selection

3 models were created as potential candidates for the final model: "rf - random forest", "gbm - boosting", and a stacked model using "gam - generalized additive model". The train control was set as a 10-fold cross validation for the models in an attempt to avoid overfitting the data.

At first the RF and GBM models were yielding 100% accuracy according to the ConfusionMatrix function. Additionally, my potential stacked model was not using the validation set for its predictions but kept using the

testing dataset that it was built on using the train function. This was a result of not assigning column names to the original data and also not assigning the new data a name that matched the name of the predictor in the original data.

(A special thank you to the author of the following site for explaining this extremely frustrating error I kept running across while running my predictions for the stacked model.

https://faustusnotes.wordpress.com/2012/02/16/problems-with-out-of-sample-prediction-using-r/)

To combat the apparent overfitting from the models and given the large number of variables left in the data which , I implemented Principal Component Analysis into my models which then gave more realistic accuracy measures.

I selected the RF model as it had the highest accuracy rate at 0.9790994 followed by the GBM model at 0.9362787 and the stacked model which gave 0.4711980 when using the validation data. The RF and GBM models yielded similar values on the testing data as well (0.9810588 & 0.9337057 respectively)

I estimated the out of sample error by subtracting from 1, the total number of instances that the predictions from the final model equaled the values from the validation set (5762) divided by the number of observations in the validation set (5885): 2.090059% or 1 - accuracy rate given by the confusionMatrix function.