Midterm Data Exercise #1 - Continuous Outcomes

Kevin McAlister

February 16th, 2022

This is the first midterm data analysis exercise for QTM 385 - Introduction to Statistical Learning. This exercise will use a single data set to ask a few questions related to creating predictive models for a continuous outcome. This data set is larger and messier than those that have been given for the problem set exercises. It is intended to provide you with a "real-world" scenario you might see if attempting to build a predictive model for a question you care about. This assignment is worth 15% of your final grade and your final solutions (a .Rmd file and a corresponding rendered document) should be submitted by 11:59 PM on March 4th. Depending on how you set up your final prediction functions, you may want to submit additional model objects. If you take this route, please submit all of the deliverables (a .Rmd file, a corresponding rendered document, and any additional model objects) as a .zip archive.

Unlike the problem sets, there will be little guidance as to what methods to use and how to interpret results. The idea is that you can treat this like a real analysis exercise - applying different models, seeing what works and doesn't work, presenting the best possible model but being transparent about the downsides of you choice. For each of the three tasks, you should consider a variety of potential methods and choose a single one as your "best" model.

A review of fitting methods that we've covered in this course (thus far):

1. Linear regression
2. Polynomial regression
3. K-nearest neighbors regression
4. Shrinkage Methods (Ridge and LASSO)
5. Splines for univariate relationships
6. Local linear regression (LOESS)
7. Generalized Additive Models (GAMs)
8. Bagged Regression Trees, Random Forests, Boosted Regression Trees

A review of concepts that should be central in your answers:

1. Training Error vs. Test Error
2. Train/Validation/Test Splits
3. Expected Prediction Error
4. Cross-validation
5. Predictor Selection
6. Linear vs. Nonlinear Relationships
7. Non-additive Errors
8. Bagging and Boosting

Data Set Overview

This assignment revolves around a data set of 39,644 articles published by the website mashable.com in 2013 and 2014. Mashable is an international multi-platform media and entertainment company that can be seen as a news aggregator; the site publishes its own articles about current events and topics. In many ways, Mashable was (and still is, but it's under different ownership now) a content aggregator. The goal of Mashable was to provide an aggregate of important daily "news" and to generate ad-revenue via advertising clicks. Therefore, the profitability of the website was directly tied to its ability to attract users to the site. The most common way users were attracted was by shared articles passing through various social media outlets. More shares leads to more ad revenue!

The data set includes article urls, the number of shares on social media via the site's provided sharing mechanisms, and a collection of 59 covariates related to the length and content of the articles. It is important to note that all 59 of these covariates could potentially be measured *prior to publication*. The reason for this can be seen in the original paper that collected and used this data set - "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News" by Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. In this paper, the authors want to demonstrate that predictive models can be used to create actionable suggestions to increase the number of shares that an article receives. In doing so, they first build a predictive model that is expected to perform well on unseen data (articles that have not yet been written) and use this to create a rules set that provides expected improvements for new articles. The proposed IDSS is operationalized over keywords - summary words that are used to categorize an article. However, it is easy to see how a system of this sort could be used to create content suggestions for the writers, themselves, to write the "clickbaitiest" articles possible.

The idea of interpretable and actionable rules sets is central to the study of fairness and bias in machine learning and actionable prediction models. Though most of the models we've discussed in class are of the interpretable variety (KNN regression being the exception) - we can viably see how a predictor increases or decreases the resulting prediction - the resulting prediction surfaces can be very complex. These methods allow "transparent" decision making processes and ensure that we're not making decision based on fragile or unethical logical links (redlining for mortgages, for example). While this data set does not present any avenues for ethically dubious prediction methods (I don't think, at least), you may find associations that are clearly non-causal. You'll discuss these as part of one of the problems.

Many of the covariates presented as processed natural language processing (NLP) metrics that are intended to capture the topic, positive/negative sentiment (polarity), and subjectivity of the article. I'll briefly summarize these concepts here:

1. Topics are presented using LDA dimensions that sum to 1. Latent Dirichlet allocation is a common method of finding common "themes" and word associations across a large corpus of text documents. The authors ran an LDA analysis and settled on 5 "topics" for the entire data set. Each document is assumed to be a mixture of these 5 topics and can be described by a weight that corresponds to how much each document contains each topic. A document with weight 1 on a single topic means that it is wholly decribed by that one topic while weight zero means that there is no part of the document that speaks to the topic. Since this is a mixture, the weights sum to one. There is no guarantee that the LDA topics are coherent, but you can find similarities between articles that are close to each topic. Should you use one of the LDA dimensions in your models (especially your small models), you may want to undertake the task of figuring out what each of them means. **A warning: since the LDA weights for each article sum to 1, placing all 5 into a model that doesn't select away collinear features will run into issues!**

2. Polarity is a measure of the positiveness or negativeness of a document. This is typically computed using a dictionary of positive and negative words and then using associational methods for determining whether words that are not recognized are positive or negative. For this data set, negative polarity means that the document is largely negative while positive polarity means that the document is largely positive. Related measures look for the rate of positive and negative words, title polarity, and conditional polarity for only positive or negative words.

3. Subjectivity is a measure of how "fact-based" an article is. Values of 0 mean that the document is fully fact-based while values of 1 mean that the document is wholly opinion based. The machinery of the subjectivity measure largely mimics that of the polarity measure starting with a dictionary of fact vs. opinion based words and using associations to cover new words.

A final thing to watch out for are two sets of unordered categorical predictors - day of publication and data channel. All of these categorical predictors are already "dummied out", so you don't need to do anything special to make them usable. However, be cautious if building a linear model! The day of the week dummies are also accompanied by `is_weekend` which is just a linear combination of the other predictors. The data channel predictors are binary variables that tell which large topic each article falls under - one of lifestyle, entertainment, business, social media, technology, world news/issues, and viral content. Viral content was dropped from the predictor set. As best I can tell, any article that has a zero on all other data channels falls into the viral content category. You can include this variable in your models if you'd like by adding a column of appropriate indicators.

Overall Goal

For this assignment, you'll take on the role of a data scientist attempting to understand and **predict** what articles are most likely to to garner the most shares. You will have access to a training set of 35,000 observations each corresponding to a specific article. Each observation has 1) the article url, 2) the number of shares, and 3) 59 covariates related to the length and content of the article. Your goal is to build predictive models that will predict the log number of shares for an article on data that was not used to train the model.

Your grade on this assignment will be a combination of:

1. Explaining your model building process
2. Assessing the out-of-sample predictive abilities for your chosen models
3. Interpreting your predictive models (plots and similar exercises)
4. Writing prediction functions and using those on a true hidden test data set

All of the methods we've discussed in class will move quite quickly on data of this size. If you run into problems with the scale of the data, feel free to take a random subset of the training data. You won't be penalized for this reduction in data size, but your models are unlikely to score as well as the large N models in terms of actual out-of-sample predictive ability.

I have a hidden test set of 4,644 articles that were randomly selected from the full data set that will be used to assess the predictive accuracy of your chosen approaches. You will provide three predictor functions that should run in a clean R/Python environment. I've included a prototype test set with 25 observations that can be used to make sure that your final model functions work on for data format that I will pass through it to assess the true predictive accuracy. I've also included 2 example predictor functions.

Question 1 - Building a Big Predictive Model (28 pts.)

Using all available predictors, come up with a predictive approach that maximizes the expected predictive accuracy for the **log number of shares** for articles in the unseen test set. Your model for this problem should use any tools we've covered (or that you know from prerequisite courses) to build the predictive model that **minimizes the expected prediction error on the unseen test set**. It can use any number of predictors and any functional form you can uncover. Obviously, you don't have this test set, so you

can never really know! However, we've covered methods for approximating this value (to the best of our non-omniscient ability).

Once you decide on a final model, write a function called `q1_midterm1_predict` that takes 3 arguments:

1. `test` - a $N_{test} \times P$ matrix, data frame, or other type of reasonable object that includes the new set of 58 predictors (and potentially other covariates) included in your training data. You should be able to pass the prototype test set included with this assignment to `test`.
2. `train` - a $N_{train} \times P$ matrix, data frame, or other type of reasonable object that includes the set of 58 predictors (and potentially other covariates) included in your training data **or** a fitted model object. If `train` is a matrix or data frame, you should be able to pass the entire training data in its first read form (as a data frame, for example) to `train`. If `train` is a model object, please indicate through comments in your code which model object should be passed to the function.
3. `seed` - a random seed that can be used to ensure that the fitting procedure is identical across function calls and computers.

Your function should train the model on `train` and produce N_{Test} predictions for the `test` observations. Your function should only return those predictions as a vector or equivalent data type.

Points for this question will be allocated as follows:

1. 14 points for demonstrating a robust exploration over different possible methods of training the model considering both variable selection and potential nonlinearities.
2. 7 points for a meaningful discussion of how you compared the potential models and what metrics you used to make your final choice.
3. 7 points for a meaningful and transparent discussion of the strengths and weaknesses of your final chosen model (relative to the other methods we've discussed in class).

Question 2 - Interpreting Your Big Predictive Model (10 pts.)

Using your predictive model from question 1, explain how the predictive surface changes as a function of your predictors. Specifically, try to figure out an interpretable set of conditions under which we would expect that an article is shared a lot and when we would expect it to be shared infrequently. This is a high dimensional predictor space, so you won't be able to do this exactly, but do your best to come up with some general observations that dictate the predicted shares for an article.

Question 3 - Building a Predictive Model with a Subset of Predictors (27 pts.)

Though most of the model fitting methods we've discussed in class are interpretable in the sense that we can directly see how each predictor contributes to the overall prediction (sans KNN regression), it is difficult to parse through all of the predictors to derive a coherent set of rules that dictates the predictions. The original purpose of this data set was to show that an automated content analysis system (called the Intelligent Decision Support System by the authors) could be used to improve the "shareability" of articles - translating predictions to actionable directions for maximizing profit. While not as robust as causal analyses, predictive models do provide a method for understanding **associations** in high dimensional data and can be used to think about potential changes that lead to different predictions. You'll discuss the caveats of this method in the next question.

Using the training data, build a predictive model for the log number of shares on an article that minimizes expected prediction error on an unseen data set **using no more than 5 predictors**. Your goal here should

be to both select a good set of predictors and model as much of the variation within the outcome using the small set of predictors - nonlinearities, interactions, and all. Compared to the big prediction model, it should be easier to find these quirks and improve the model to better model them.

Once you decide on a final model, write a function called `q3_midterm1_predict` that takes 4 arguments:

1. `test` - a $N_{test} \times P$ matrix, data frame, or other type of reasonable object that includes the new set of 58 predictors (and potentially other covariates) included in your training data. You should be able to pass the prototype test set included with this assignment to `test`.
2. `train` - a $N_{train} \times P$ matrix, data frame, or other type of reasonable object that includes the set of 58 predictors (and potentially other covariates) included in your training data **or** a fitted model object. If `train` is a matrix or data frame, you should be able to pass the entire training data in its first read form (as a data frame, for example) to `train`. If `train` is a model object, please indicate through comments in your code which model object should be passed to the function.
3. `seed` - a random seed that can be used to ensure that the fitting procedure is identical across function calls and computers.

Your function should train the model on `train` and produce N_{Test} predictions for the `test` observations. Your function should only return those predictions as a vector or equivalent data type.

Points for this question will be allocated as follows:

1. 14 points for demonstrating a robust exploration over different possible methods of training the model considering both variable selection, potential nonlinearities, and complicated dependencies between predictors.
2. 7 points for a meaningful discussion of how you compared the potential models and ultimately chose your set of predictors.
3. 6 points for a meaningful and transparent discussion of the strengths and weaknesses of your final chosen model (relative to the other methods we've discussed in class).

Question 4 - Interpreting the Subset Model (10 pts.)

Using your predictive model from Question 3, explain how the prediction surface changes as a function of your predictors. Since the set of predictors is much more manageable, discuss what types of articles you would predict to have the highest number of shares and what types would have the lowest number of shares. Does the prediction surface over this small subset of predictors look the same as it did in the full model? What changes?

Using your model, come up with a set of actionable rules that you would pass to the Mashable writers to try to maximize the number of shares and discuss why this rule set makes sense in the context of the problem. Then, discuss reasons why we might not expect the rules set to result in the expected increase in shares in the sense of "a change in x results in a change in y". Specifically think about how this method differs from a carefully designed causal analysis (think about confounding!).

Question 5 - Building a Predictive Model with a Single Predictor (15 pts.)

Now, take the idea of creating a small actionable predictive model to the extreme and find the single predictor model that minimizes expected prediction error for log of shares on the unseen test data set. Discuss your process and how you decided on a final model. Create a plot that shows your predictive curve for the training data **and** for a test set that was not used to train the model (a hint about the process you should take for

all of the questions). How does the predictive surface for your single predictor compare to the equivalent impact of the predictor in the subset and full models? Can you intuitively explain this result?

Once you decide on a final model, write a function called `q5_midterm1_predict` that takes 4 arguments:

1. `test` - a $N_{test} \times P$ matrix, data frame, or other type of reasonable object that includes the new set of 58 predictors (and potentially other covariates) included in your training data. You should be able to pass the prototype test set included with this assignment to `test`.
2. `train` - a $N_{train} \times P$ matrix, data frame, or other type of reasonable object that includes the set of 58 predictors (and potentially other covariates) included in your training data **or** a fitted model object. If `train` is a matrix or data frame, you should be able to pass the entire training data in its first read form (as a data frame, for example) to `train`. If `train` is a model object, please indicate through comments in your code which model object should be passed to the function.
3. `seed` - a random seed that can be used to ensure that the fitting procedure is identical across function calls and computers.

Your function should train the model on `train` and produce N_{Test} predictions for the `test` observations. Your function should only return those predictions as a vector or equivalent data type.

Points for this question will be allocated as follows:

1. 7 points for demonstrating a robust exploration over different possible methods of training the model considering both variable selection and potential nonlinearities.
2. 3 points for a meaningful discussion of how you compared the potential models and ultimately chose your single predictor.
3. 5 points for comparison of variable impact across models and discussion of logic for differences.

Question 6 - Do your models perform well on out-of-sample data? (12 pts.)

Finally, let's get everything ready to be run on a true out of sample data set. Please include a code block here that includes `q1_midterm1_predict`, `q3_midterm1_predict`, and `q5_midterm1_predict`. Once the assignments are turned in, I'll use these functions to train your chosen models and create predictions for the out of sample data set. I'll post the data set on the Canvas site after all midterms are turned in.

For each model, I'll compute a root mean squared prediction error. Across all models turned in, points will be allocated as follows:

1. For Q1 and Q3, let RMSE_{Min} be the minimum prediction error over the submitted models. Let RMSE_i be your predictive error. Your prediction points will be computed as:

$$5 \times \frac{\text{RMSE}_{\text{Min}}}{\text{RMSE}_i}$$

2. For Q5 it will be:

$$2 \times \frac{\text{RMSE}_{\text{Min}}}{\text{RMSE}_i}$$

The goal here is to add a low-stakes incentive and fun "competition" to try to optimize your predictive models. I expect that the performance across groups/students will be quite close, so this won't greatly affect your final grade.

If your prediction functions do not run, I will give zero points! I will make every effort to debug on my end, but make sure that your functions can run from a clean R/Python environment. If using libraries/packages, please indicate the packages at the top of your code block/build in a package call that does not require `library()`. Example functions are included below. Code cleanliness and usability is important, so please treat this task with some rigor!

Example Function with train data frame:

```
test_proto <- data.table::fread("midterm1_testProto.csv")
# Train is a data frame and test is a data frame
q1_midterm1_predict <- function(train, test) {
    # Let's train a KNN-regression model with 25 nearest
    # neighbors
    train_x <- train
    train_x$shares <- NULL
    train_x$url <- NULL
    train_x <- as.matrix(train_x)
    train_y <- as.matrix(log(train$shares))
    test_x <- test
    test_x$shares <- NULL
    test_x$url <- NULL
    test_x <- as.matrix(test_x)
    knn_mod <- FNN::knn.reg(train = train_x, test = test_x, y = train_y,
        k = 25)
    return(knn_mod$pred)
}
# It works
q1_midterm_predict(train = train, test = test_proto)
```

Example function with train model object:

```
test_proto <- data.table::fread("midterm1_testProto.csv")
# Let's run a linear regression model
train_proc <- train
train_proc$url <- NULL
train_proc$LDA_04 <- NULL
train_proc$is_weekend <- NULL
train_proc$weekday_is_wednesday <- NULL
lm_mod <- lm(log(shares) ~ ., data = train_proc)
# Save the model object
saveRDS(lm_mod, "q1_midterm_predict.rds")
# We can read the model object back in
pred_mod <- readRDS("q1_midterm_predict.rds")
# You can check that pred_mod is the same as lm_mod! Write a
# function that takes the model object as train
q1_midterm1_predict <- function(train, test) {
    # Using the train model object, make predictions
    return(predict(train, newdata = test))
}
# It works
q1_midterm_predict(train = pred_mod, test = test_proto)
```