Edward Leardi

Metis SF Winter ‘18

Using Linear Regression with Ratemyprofessors.com Data

1. Problem – I wanted to scrape some data from ratemyprofessors.com and see how well I could predict the Overall Rating of each professor on the site.

2. Data – I noticed that every school page on RMP has a “[View All Professors](http://www.ratemyprofessors.com/search.jsp?queryBy=schoolId&schoolName=University+of+Connecticut&schoolID=1091&queryoption=TEACHER)” page. My first scraper used Selenium to loop through every school page by incrementing the school id in the URL by one ([ratemyprofessors.com/campusRatings.jsp?sid=***1091***)](http://www.ratemyprofessors.com/campusRatings.jsp?sid=1091)), then clicked the “View All Professors” button, then loaded all the professors on the page (which was a bit tricky because the “Load More” button was located off screen and in a sidebar so I had to figure out how to get Selenium to select the sidebar, and scroll down to bring the button into view), and finally extracted the URLs for every professor’s RMP page. I then used Beautiful Soup to load in every URL and extracted all the features I needed. Before preprocessing the data, I had a dataframe with 22 features and ~70,000 professors.

3. Setup – I noticed right away that my target variable Overall Rating had a bimodal distribution. I tried both deleting out +/-3 std outliers as well as a log transformation on the target but the models only performed worse after this. I deleted out any rows with errors, which was less than 5% of the dataset. I also deleted out the Would You Take Again? feature as it was missing values for about 70% of the rows. Finally, I converted the Hotness feature to a dummy variable.

4. Models – I split the data 90/10. I started with OLS. First I baselined against just one feature, Difficulty. Then I expanded the model to the 7 features that had an |r|>.2 with Overall Rating. Finally, I tried OLS with all 21 features. I noticed the more features I added with OLS, the better the model performed, so I realized regularization likely wouldn’t help the model’s accuracy much. I then utilized 5-fold cross validation to obtain r-squared values for OLS, LASSO, Ridge, and Elastic Net. All models basically performed around the same level of accuracy yielding about 0.45 r-squared values and errors of about .55 (on a 5-star scale). I also played around with the alpha values of LASSO to investigate which variables it would drop if I used a suboptimal alpha value. I noticed that alphas of less than .01 with LASSO saved all 21 features, an alpha of .01 dropped 3 features (and produced an r-squared of ~0.40), and an alpha of .1 dropped 13 features (and produced an r-squared of ~0.33). Digging deeper into the features LASSO ended up dropping, I saw that these typically tended to be the tags that had p-values of greater than .05, or had low correlation (|r|<.1) with the target variable. For both alphas, LASSO kept the best predictors Difficulty and Hotness, and typically kept the tags that had the most responses such as Tough Grader, Respected, and Amazing Lectures, while dropping some of the less popular tags such as Clear Grading Criteria and Get Ready to Read.

5. Test – All four models produced r-squared values of about ~0.48 on the test data, and regression plots (of y-predicted vs. y-actual) that compared very similarly to those using the training data indicating little to no overfitting.

6. Conclusion – All of the models performed with a very similar level of accuracy. This confirmed my hypothesis that regularization likely wouldn’t help improve my model. That said, my digging into LASSO showed that a model could be built that could trade off 0.1 in r-squared but drop half of the predictor variables. So there’s a case to be made for using LASSO if a simpler model is desired. Otherwise OLS is probably what I’d roll with!