STA 141B Final Project:
A Sentiment Analysis Investigation of RateMyProfessors.com Reviews for UC Davis Professors

Riley Adams, Jordan Bowman, Frankie Glaviano

December 7, 2022

1 Introduction

Perceived quality of a professor by a student is important to both the student in terms of their educational experience, and the professor in terms of their career success and teaching effectiveness. As students at UC Davis, we were interested in the student perception of professors and instructors across campus. RateMyProfessors.com is a popular website that provides a platform for and aggregates student reviews for college professors. Two questions were of particular interest to our research in this project:

1. How do departments differ in terms of student-perceived quality and difficulty of their professors?
2. Given a professor's review text, can we classify the reviewer's sentiment as good or bad?

This second question is a problem of sentiment analysis which looks to measure the opinion of the writer of a body text. Sentiment analysis techniques have broad applications in politics and product review analysis to give insight into public opinion. In this project, we work with a relatively simple logistic regression model to perform our sentiment classification.

2 Data Acquisition

To carry out our sentiment analysis, we required data to train and test a classification model, which could then assess the sentiment of each review written about each professor at UC Davis. This data was acquired from RateMyProfessors.com, a website which allows users to review professors, and read the reviews written by other users. Reviewers give a one to five rating on the professor's quality and difficulty, write comments up to 350 characters long, indicate the class they took, whether they would take the professor again, the grade they received, and a few other metrics and qualitative tags. **Figure 1** is an example of one such review. For our classification model, the data we were most interested in were the quality rating score and the comments contained in each review.

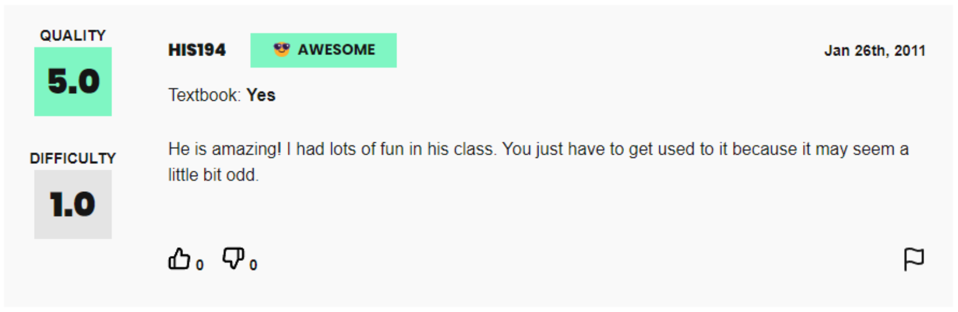

QUALITY
5.0
DIFFICULTY
1.0
HIS194 AWESOME Jan 26th, 2011
Textbook: **Yes**
He is amazing! I had lots of fun in his class. You just have to get used to it because it may seem a little bit odd.
0 0

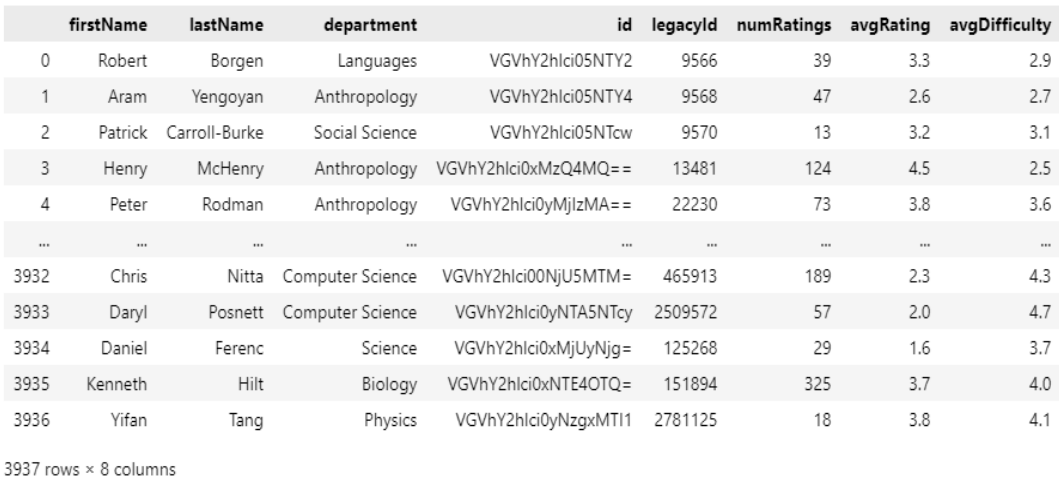
Figure 1: Example of a RateMyProfessors.com review.

2.1 Basic Professor Information (*profs* Dataset)

Reviews for each professor are contained on that particular professor's dedicated webpage. Therefore, we needed a way to access the data displayed on each professor's page, and repeat this for every professor. We decided to get a preliminary dataset which would contain a list of all the professors at UC Davis, which we would then use to navigate to each professor's individual page. We were able to locate a RateMyProfessors.com webpage that displayed all professors at UC Davis and decided this data would be a good starting point.

Collecting this data was not a straightforward task, as the data was not actually contained within the HTML of the webpage. Upon inspection of the network communication, it was discovered that the webpage runs JavaScript to query an internal database via GraphQL. The query to this database only became visible after clicking the "load more" button at the bottom of the professor list. Once the query was located, we extracted the necessary information to make our own queries using the *requests* package in Python.

The data returned by the request was in JSON format, which we then parsed and cleaned and eventually converted into a DataFrame using the pandas package. The DataFrame contains columns *firstName*, *lastName*, *department*, *id*, *legacyID*, *numRatings*, *avgRating*, and *avgDifficulty*, as seen in **Table 1**. Rows with zero ratings were dropped, and the final dataset contained 3,937 rows of professor information to be used in further analysis as well as acquisition of the *reviews* dataset.



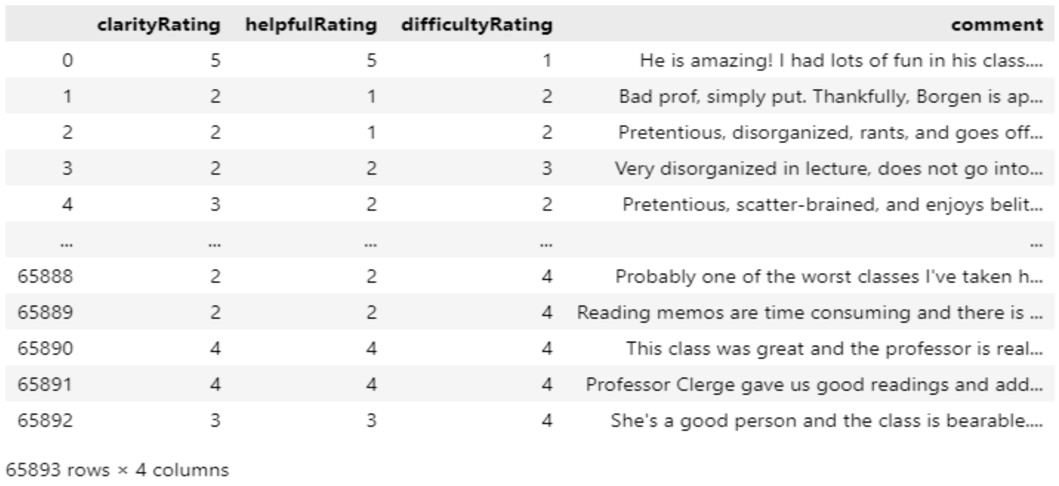
	firstName	lastName	department	id	legacyId	numRatings	avgRating	avgDifficulty
0	Robert	Borgen	Languages	VGVhY2hlci05NTY2	9566	39	3.3	2.9
1	Aram	Yengoyan	Anthropology	VGVhY2hlci05NTY4	9568	47	2.6	2.7
2	Patrick	Carroll-Burke	Social Science	VGVhY2hlci05NTcw	9570	13	3.2	3.1
3	Henry	McHenry	Anthropology	VGVhY2hlci0xMzQ4MQ==	13481	124	4.5	2.5
4	Peter	Rodman	Anthropology	VGVhY2hlci0yMjIzMA==	22230	73	3.8	3.6
...	...	...	...	...	...	...	...	...
3932	Chris	Nitta	Computer Science	VGVhY2hlci00NjU5MTM=	465913	189	2.3	4.3
3933	Daryl	Posnett	Computer Science	VGVhY2hlci0yNTA5NTcy	2509572	57	2.0	4.7
3934	Daniel	Ferenc	Science	VGVhY2hlci0xMjUyNjg=	125268	29	1.6	3.7
3935	Kenneth	Hilt	Biology	VGVhY2hlci0xNTE4OTQ=	151894	325	3.7	4.0
3936	Yifan	Tang	Physics	VGVhY2hlci0yNzgxMTI1	2781125	18	3.8	4.1

3937 rows × 8 columns

Table 1: *profs* data frame (first and last five rows).

2.2 Professor reviews (*reviews* dataset).

The next step in our data acquisition process was to collect the data for every review of each professor in all 3,937 rows of the *profs* dataset. Due to the anticipated scale of the coming dataset, we began by testing our review data acquisition process on the first professor in the list, Robert Borgen. We also used the network communications of Robert Borgen's page to collect the necessary queries for our requests. Once this was implemented successfully, we generalized the process to acquire all the reviews. The process included looping through every row of the *id* column in the *profs* data, to query the website's internal database for data on each professor. The loop also included an if-clause which skipped over any rows which caused an error message to be contained in the returned JSON data. Those requests which did not contain errors were parsed within an inner loop and concatenated as rows in a pandas data frame. The most pertinent columns of the *reviews* DataFrame were later subset and are pictured in **Table 2**.



	clarityRating	helpfulRating	difficultyRating	comment
0	5	5	1	He is amazing! I had lots of fun in his class....
1	2	1	2	Bad prof, simply put. Thankfully, Borgen is ap...
2	2	1	2	Pretentious, disorganized, rants, and goes off...
3	2	2	3	Very disorganized in lecture, does not go into...
4	3	2	2	Pretentious, scatter-brained, and enjoys belit...
...	...	...	...	...
65888	2	2	4	Probably one of the worst classes I've taken h...
65889	2	2	4	Reading memos are time consuming and there is ...
65890	4	4	4	This class was great and the professor is real...
65891	4	4	4	Professor Clerge gave us good readings and add...
65892	3	3	4	She's a good person and the class is bearable....

65893 rows × 4 columns

Table 2: Initial *reviews* DataFrame (first and last five rows).

Running the loop to acquire reviews data was quite computationally intensive, and unfortunately was bottle-necked by the fact that we could not make too many requests too quickly, lest we overload the servers at RateMyProfessors.com and get banned from the site. In our final iteration, the loop ran for approximately an hour, before reaching a timeout error. By this time, we had collected 65,893 rows of reviews, which we deemed sufficient for our analysis due to time constraints.

3 Exploratory Analysis and Cleaning of *profs* Dataset

3.1 Initial Exploration and Cleaning

The next step was to conduct some exploratory analyses of our basic professor information to see what we could uncover about our data. Throughout this process, we discovered numerous aspects of our dataset which could be cleaned up to increase the quality of our information.

First, an initial scatter plot of average rating vs average difficulty revealed that there were a number of points at $(0, 0)$, which was outside of the possible scores of 1 through 5. Upon further inspection, it was discovered that a number of professors were listed on the website, despite never having been reviewed. Thus, they had average ratings of zero. This is what prompted us to go back and locate/remove all of these professors from our data.

Since we had data for the department of each professor, we decided to aggregate our data and see what we could discover about the differences between departments. In doing so, we discovered even more aspects of our data which needed to be tidied.

One major issue was department names. There were instances of repeated names, most often due to formatting issues with the ampersand symbol. For example, we had "Agricultural & Resource Economics," "Agricultural Resource Economics" and "Agricultural amp Resource Economics" all showing up as separate departments. We remedied this using string methods and indexing to filter and edit values in the DataFrame so that all department names were standardized. We also compared department names with those listed on UC Davis affiliated webpages to correct and combine misstated depart names. For example, "French" and "Italian" should both be under the "French & Italian" department and "Women" should be under "Women's Studies." Lastly, there were a few departments listed that were not professor affiliated departments at all, like "Advisor," and "Student Services." We removed them.

3.2 Findings from Cleaned Data

With our data cleaned we were able to conduct a more insightful exploration analysis. We found the mean number of ratings for professors at UC Davis is 21.5. We then plotted the number of ratings for each professor **(Figure 2)**, revealing that the number of ratings is not evenly distributed across all professors.

Most professors have less than 200 reviews, while Andreas Toupadakis of the Chemistry department has upwards of 800 reviews.

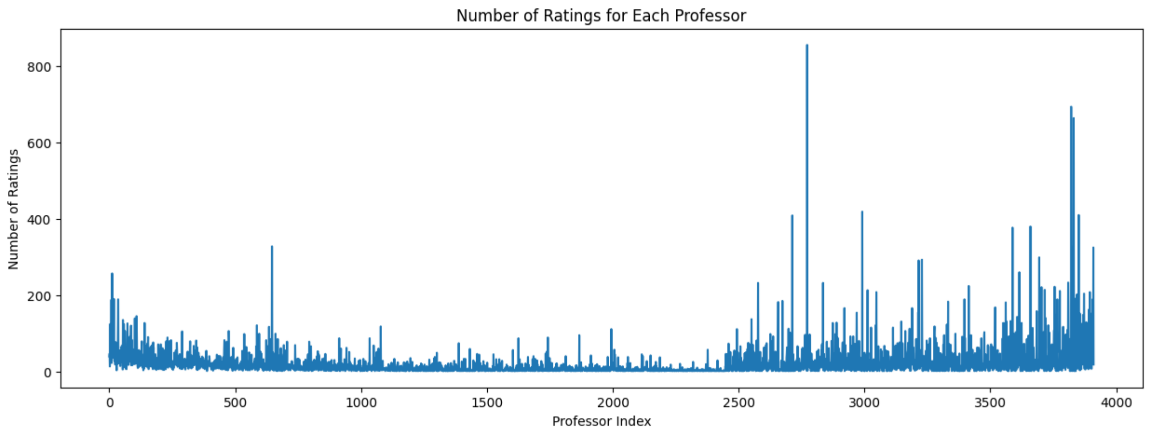


Figure 2: Distribution of number of ratings by professor.

We proceeded our exploration with histograms of the two rating types: average quality, and average difficulty. In **Figure 3**, our histogram of average quality revealed that our data was biased towards positive reviews. Our histogram of average difficulty, on the other hand, is relatively normally distributed, with more values near 3 and less near 1 and 5.

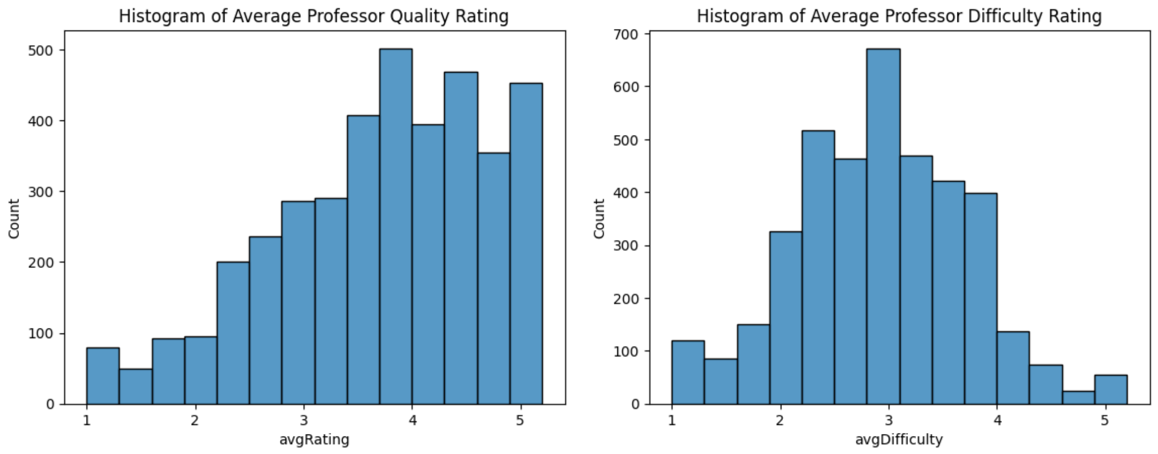


Figure 3: Histograms for quality and difficulty scores.

Next, we examined the relationship between mean quality rating and mean difficulty rating using a scatter plot. As illustrated by the regression line in **Figure 4**, there is a general trend that as perceived difficulty of a professor's class goes up, the lower the quality rating given tends to be. We calculated the correlation between these two variables and found it to be -0.51.

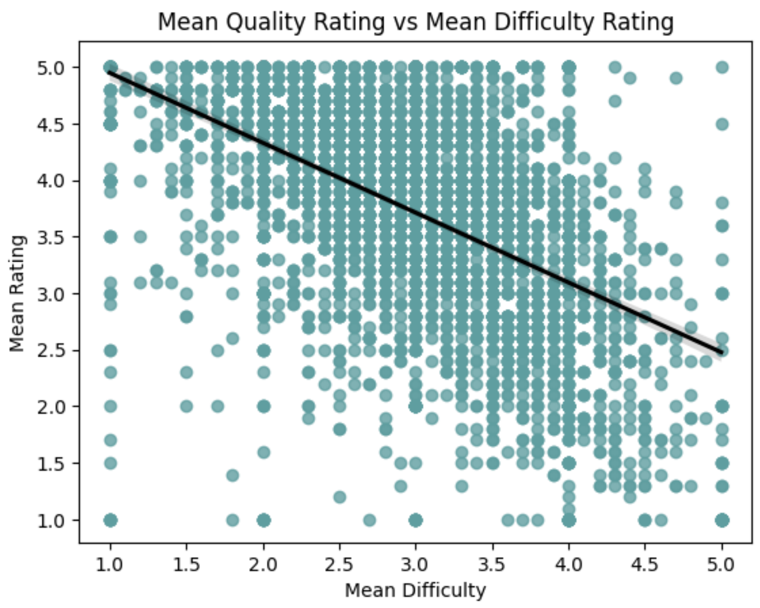


Figure 4: Scatter plot of quality and difficulty scores.

In the final portion of our exploratory analysis, we grouped our data by department so that we could compare by quality, difficulty and number of ratings, to see how they stack up against each other. In doing so, we considered only those departments which had at least 50 reviews.

In **Figure 5** we display the top and bottom ten departments in terms of quality. Our own Statistics Department came in at 74 out 89 departments with a mean quality score of 3.45 and a total of 2163 reviews given.

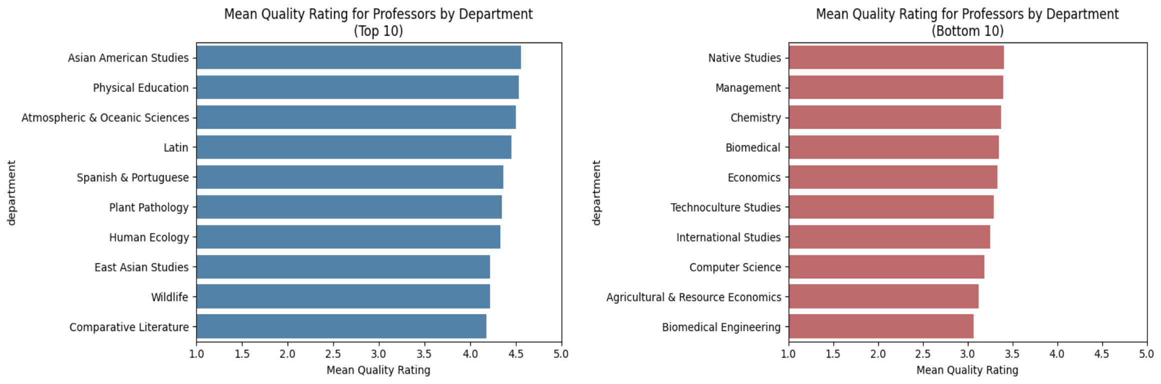


Figure 5: Quality rankings by department.

In **Figure 6** we display the top and bottom ten departments in terms of difficulty. Our own Statistics Department ranked 23rd, so the trend of higher difficulty, lower rating does appear to fit this particular data point.

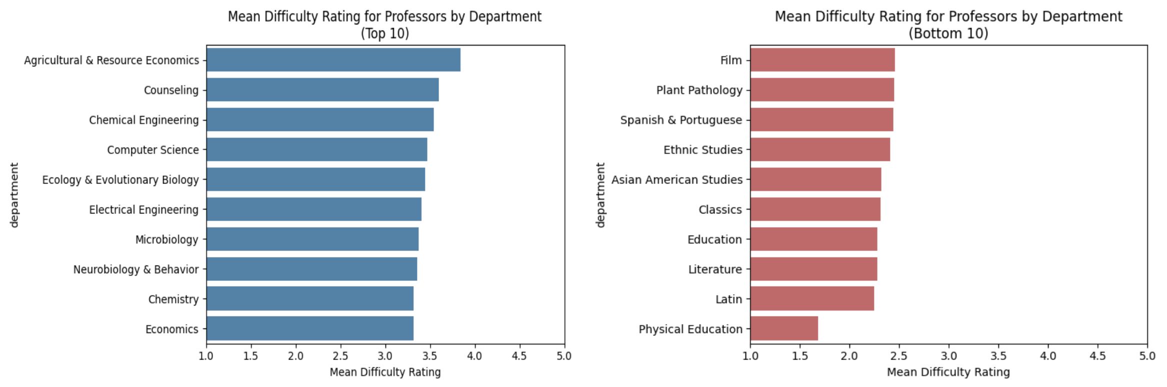


Figure 6: Difficulty rankings by department.

In **Figure 7** we display the top and bottom ten departments in terms of number of ratings. The unequal distribution of ratings by department is apparent here, as those in the top 10 exceed the bottom 10 by orders of magnitude.

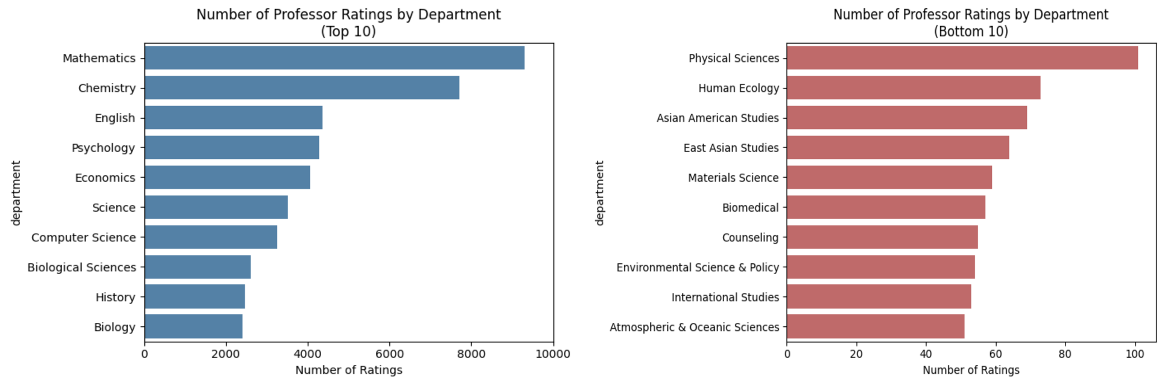


Figure 7: Top and bottom departments by number of reviews.

4 Cleaning the *reviews* Dataset Text

Before we could utilize the *reviews* dataset, we first needed to resolve a particular issue in the way that the overall "quality" metric for a RateMyProfessors.com score is held internally. From **Table 2**, observe that we were not able to locate and subset a score for professor "quality" like that which is presented on a common RateMyProfessors.com review **Figure 1**. Through an evaluation of both the process that a user takes when creating a review and the data that was collected above, we determined that the "quality" score is actually composed of a simple average of two internal values: *clarityRating* and *difficultyRating*. This is in contrast to the difficulty rating which is stored directly. Therefore, to analyze the values a user observes when reading RateMyProfessors.com scores, we had to recreate this column and append it to the inside of our DataFrame.

At this point, the resulting subset DataFrame of the *reviews* dataset was ready to be processed. Foremost in any data processing procedure is to decide on a course of action for missing values. In this case, the proportion of missing values was extremely small in comparison to the overall dataset, so we felt it was most effective to remove them from the dataset entirely. This was principally because any missing text or missing scores would make the sentiment classification analysis we performed later on functionally impossible. Thus, we immediately dropped any rows of this DataFrame that contained NA values, which eliminated 100 rows of the dataset. Through careful inspection we also determined that there were 928 rows that contained "No Comments" in place of the review text. Presumably, this was the default response from the database in any instance where a comment was not provided by the user making their review. Again, these particular reviews were unable to be analyzed for their text content and so we removed them.

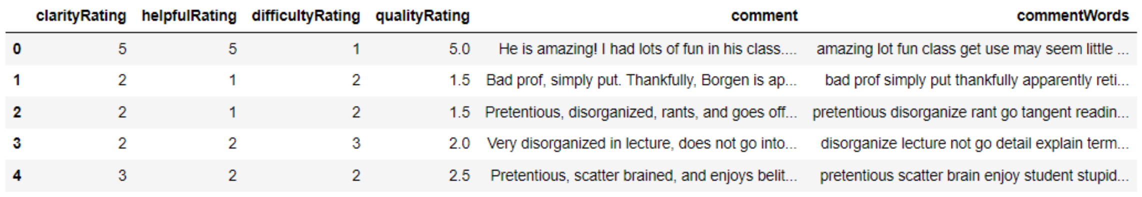
From here, we focused our attention on processing the *comments* column of the new, reduced *reviews* DataFrame. This column is a pandas Series of strings. Ideally, we want to reduce this string to only words that are relevant to the analysis. Words that provide little to no meaning (i.e. stopwords) such as articles and pronouns among other words need to be removed. The NLTK (Natural Language Toolkit) package provides a list of these stopwords (**nltk.corpus.stopwords.words('english')**) that we will utilize. First, however, each string in the *comments* series needed to be tokenized into its constituent words. To achieve this, we first converted all "/" and "-" characters into spaces within the text. In English, these characters are sometimes used for shorthand purposes, however they will interfere with our tokenization. For the tokenization itself, we first converted every character to lowercase and then applied the **nltk.word_tokenize()** function from the NLTK package. Ultimately, we were left with a list of lists of words, where each of the inner lists corresponded to a comment, and therefore a row, in the DataFrame. Although we would manipulate this list independently of the DataFrame, it was critical that the index order of this list did not change prior to the end of our text processing or it would not have been appended properly to the DataFrame.

Crucially, we then lemmatized the words with a **nltk.WordNetLemmatizer** object from the NLTK package. Lemmatization is the process in which a word is reduced to an unconjugated and singular "dictionary form" so that it can be analyzed with other words that share the same base. For instance, "went" and "gone" are forms of the verb "go" and so they would be reduced to "go" during lemmatization. Not only will this improve our ability to interpret sentiment by grouping together words that are the same in meaning but differ in form, but it will also allow us to later use a library of English words for comparison that do not contain all of the varying forms of English words. It is important to note that the **WordNetLemmatizer** requires the part of speech of the word to work effectively and while NLTK provides a function **nltk.pos_tag()** that identifies the most probable part of speech for an input word it returns that part of speech in a format that is unusable by **WordNetLemmatizer**. We created a conversion function to resolve this issue.

Our text processing continued with the elimination of punctuation and the NLTK stopwords mentioned previously with the exception of the word "not." Despite its presence in the list of stopwords, "not" conveys important information about negated sentimental meaning in phrases like "not good" or "not bad."

Lastly, we removed all words that were not in the NLTK corpus of English words derived from **nltk.corpus.words.words()** which eliminates both non-English words and any spelling errors. However, this was a computationally intensive task due to the number of comparisons that had to be performed between the list of English words which contained 236,736 words and every remaining word in our approximately 65,000 comments. To alleviate the computational strain, we first converted our list of lists of words into a single set and then converted that set back into a list. The advantage of using a set as an intermediary is that sets only contain unique elements. Since, there are many, many words that are very frequently used by reviewers, this greatly reduced the number of words to consider against the English corpus. In fact, using a set to remove duplicates reduced the number of words to check from well over a million to approximately 20,000. From here, we generated a list of words that were in our comments, but not in the English word corpus and then removed those from our list of lists of words. All in all, this process of converting from list to set to list took approximately 20 minutes to compute which is significantly less time than direct comparison of our list of lists of words to the English corpus.

The resulting words for each are joined together with spaces to make a list of strings where each string is the processed relevant words of the comment in the row with the corresponding row index. This list is then appended as a new *commentWords* column of the *reviews* DataFrame. Some reviews did not contain a single word at this point because of the cleaning process (i.e. they did not begin with a single relevant English word), so we once again had to remove rows without review text. **Figure 8** shows the head of this DataFrame at the end of our processing.



	clarityRating	helpfulRating	difficultyRating	qualityRating	comment	commentWords
0	5	5	1	5.0	He is amazing! I had lots of fun in his class....	amazing lot fun class get use may seem little ...
1	2	1	2	1.5	Bad prof, simply put. Thankfully, Borgen is ap...	bad prof simply put thankfully apparently reti...
2	2	1	2	1.5	Pretentious, disorganized, rants, and goes off...	pretentious disorganize rant go tangent readin...
3	2	2	3	2.0	Very disorganized in lecture, does not go into...	disorganize lecture not go detail explain term...
4	3	2	2	2.5	Pretentious, scatter brained, and enjoys belit...	pretentious scatter brain enjoy student stupid...

Figure 8: First five rows of cleaned and processed *reviews* DataFrame.

One relevant consideration to our methodology is the occurrence of slang and other colloquialisms in the reviews text. Some words that convey sentimental meaning, such as "halfwit," were not in our corpus of English words. Although they could be interpreted by an English speaker, they were excluded from our analysis here. A possible area of future interest would therefore be to perform additional analysis that includes these slang terms.

5 Logistic Regression Classification

The Towards Data Science article "A Beginner's Guide to Sentiment Analysis with Python" by Natassha Selvaraj formed the basis for our methodology in performing a sentiment analysis in Python. This article involves the classification of positive and negative product reviews like those that would be found on an online retailer's page for a specific product. Similar to the RateMyProfessors.com reviews, the scores given to these products by consumers are on a scale from 1 to 5 (although the system for RateMyProfessors.com allows for half scores as well). We followed Selvaraj's coding methodology, although we will provide our own explanation and analysis below.

Recalling our primary question in this project, we were concerned with determining a model that can classify a given RateMyProfessors.com review for UC Davis professors as positive or negative. As this is a binary classification problem, we constructed a logistic regression model. The process of constructing a logistic regression began with classifying our processed *reviews* data into negative and positive reviews in order to train our model. Reviews with *qualityRating* scores of greater than 3.0 were assigned positive "1" values in a new *sentiment* column in our DataFrame while those with scores lower than 3.0 were assigned negative "-1" values in that column. One immediate problem with our dataset was the presence of neutral 3.0 ratings. However, since 3.0 ratings are neither positive nor negative, these reviews were removed from the regression model as they lie outside the scope of our research interest.

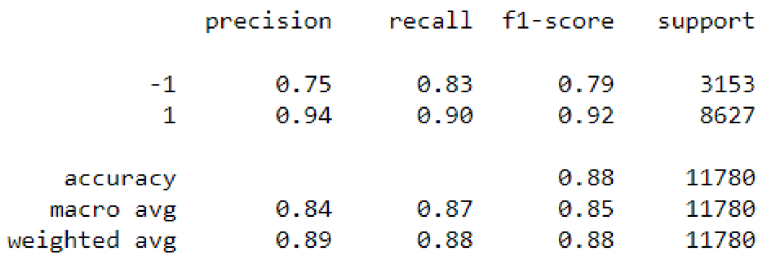
As logistic regression is a supervised statistical learning technique and as we were interested in determining the accuracy of our model, we split our data into a conventional 80% training set and 20% test set ratio. This was accomplished with the use of the **train_test_split()** function imported from **sklearn.model_selection**.

To construct this regression model, we needed to convert the *commentWords* column of the training set into a matrix of words, i.e. a "bag of words" matrix. Within this matrix each column corresponded to a unique word within all of the rows of the training set's *commentWords* and each row of the matrix corresponded to a review within that training set. The number of occurrences of a given word in a given review was recorded in the corresponding row and column. We created this matrix with a **CountVectorizer** object imported from **sklearn.feature_extraction.text** that utilized the **nltk.word_tokenize** tokenizer from NLTK. Additionally, we transformed the test dataset into a matrix that shared the same columns as our training set matrix. This fit the test set to the same features as the training set. Note, that this also made any words present in the test set that were not present in the training set disappear from our analysis as they were not recorded in the matrix. Given the size of our training dataset, this should not have a notable effect on our resulting model.

We then used a **LogisticRegression** object from **sklearn.linear_model** to create a logistic regression of our training data sentiment scores on our training word matrix and used that fitted model to predict the positive or negative sentiment of our test data. We used **confusion_matrix()** and **classification_report()** from **sklearn.metrics** to acquire the information in **Figure 9** and **Figure 10**.

	Predicted Negative	Predicted Positive
True Negative	2631	522
True Positive	894	7733

Figure 9: Confusion matrix for logistic regression.



```
              precision    recall  f1-score   support

          -1       0.75      0.83      0.79      3153
           1       0.94      0.90      0.92      8627

    accuracy                           0.88     11780
   macro avg       0.84      0.87      0.85     11780
weighted avg       0.89      0.88      0.88     11780
```

Figure 10: Classification report for logistic regression.

From **Figure 9** and **Figure 10** we find that our logistic regression model performs with an accuracy of approximately 88%. We have clearly created a model that accurately predicts a RateMyProfessors.com user's sentiment about a professor at UC Davis based upon their review text. Nevertheless, observe that the model is notably worse at predicting negative scores in comparison to positive scores.

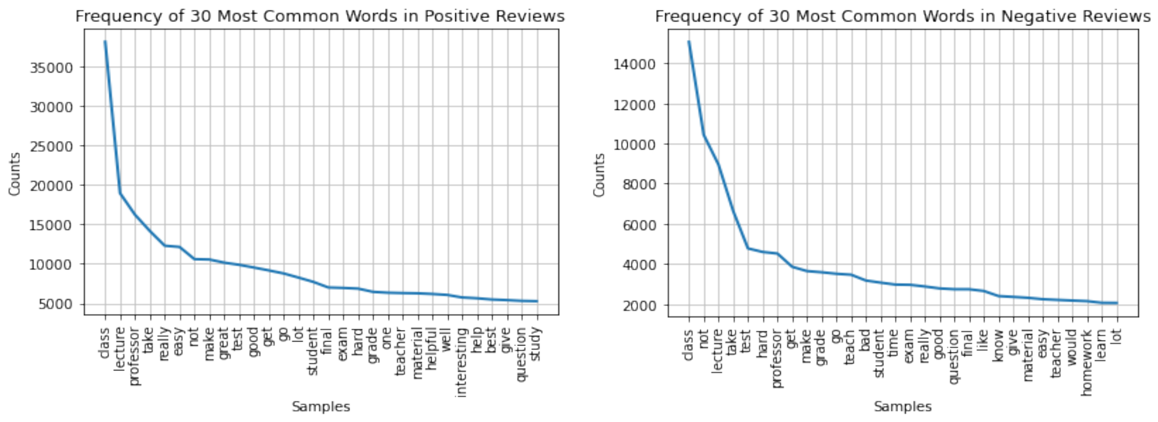


Figure 11: Frequency plots for comment tokens in positive and negative reviews.

In **Figure 11**, we illustrate which words appear most frequently in positive reviews **(left)** and which words appear most frequently in negative reviews **(right)**. It is apparent that a high number of words which are commonly understood to have neutral sentiment appear in both positive and negative reviews with high frequency (e.g. "lecture"). A future iteration of this model may account for these by assigning a "neutral" score. Filtering through these words, we see that some sentimentally significant words which appear most often in positive reviews are "good" and "easy". We also have "hard" occurring fairly frequently in positive reviews, although considerably less frequently than "easy". The reverse is true in negative reviews.

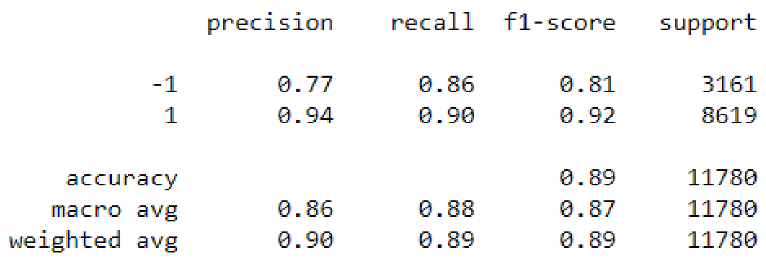
6 Improving Upon the Logistic Regression

6.1 Concatenating "Not"

Consider sentences like the following: "This professsor is not good." Our data processing would have reduced this to a string containing "professor not good" only. Intuitively, we are aware that the "not" preceding "good" negates the meaning of good, however the logistic regression model does not take this into account directly. One possible solution to this issue is to simply concatenate “not” into the word that follows it (i.e. “not good” becomes “notgood” as one word). Although not every case of the word “not” precedes an adjective to negate, we were interested to see if performing this concatenation improved the accuracy of our model. Concatenating “not” and performing a new logistic regression and classification analysis on our testing and training datasets from the previous section yielded the results in **Figure 12** and **Figure 13**.

	Predicted Negative	Predicted Positive
True Negative	2704	457
True Positive	821	7798

Figure 12: Confusion matrix for concatenating “not” logistic regression model.



```
              precision    recall  f1-score   support

          -1       0.77      0.86      0.81      3161
           1       0.94      0.90      0.92      8619

    accuracy                           0.89     11780
   macro avg       0.86      0.88      0.87     11780
weighted avg       0.90      0.89      0.89     11780
```

Figure 13: Classification report for logistic regression.

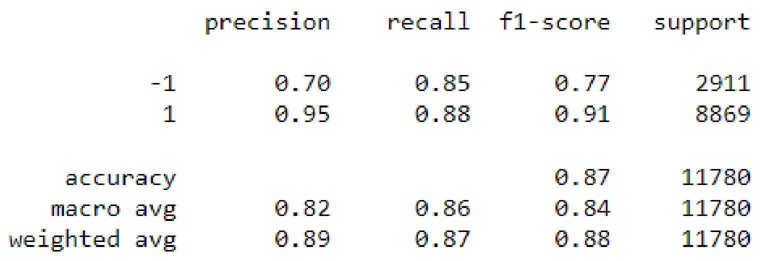
The confusion matrix and classification report above do not show major improvements over our initial logistic regression model. However, we did see minor decreases in false negatives and false positives and an approximately 1% improvement in model accuracy.

6.2 Using Bigrams

We also attempted to address the issue of sentiment interpretations mentioned in the subsection above by using bigrams in our model instead. This yielded the results in **Figure 14** and **Figure 15**.

	Predicted Negative	Predicted Positive
True Negative	2467	444
True Positive	1058	7811

Figure 14: Confusion matrix for bigrams logistic regression model.



```
              precision    recall  f1-score   support

          -1       0.70      0.85      0.77      2911
           1       0.95      0.88      0.91      8869

    accuracy                           0.87     11780
   macro avg       0.82      0.86      0.84     11780
weighted avg       0.89      0.87      0.88     11780
```

Figure 15: Classification report for logistic regression.

Once again, this attempt at improving our model was ultimately unsuccessful. Using bigrams caused only a very slight decrease in overall accuracy.

7 Conclusion

Through our exploratory analysis of the RateMyProfessors data, we were able to touch on some of the relative differences between departments at UC Davis when it comes to perceived quality and difficulty scores. It may be true that, in a general sense, as the perception of a professor's difficulty increases, students are more inclined to give the professor a lower quality score. Future research may employ formal statistical testing on our data to find whether these score differences, and the relationship between difficulty and quality are significant.

Further, we were able to build a classification model which successfully categorizes a RateMyProfessors review sentiment as either positive or negative approximately 88% of the time. Noticeably, this model classifies positive reviews more successfully than negative reviews, and we recognize that this may be due to the bias of our data towards positive reviews which we discovered in our exploratory analysis. Future improvements on our model may take into account non-English words with sentimental significance (i.e. slang), handle negation in a more sophisticated manner (i.e. "not good" or "isn't bad") and introduce a third score for neutral sentiment.

References

1. Selvaraj, Natassha. "A Beginner's Guide to Sentiment Analysis with Python." Medium, Towards Data Science, 12 Sept. 2020. Accessed 7 Dec. 2022. https://towardsdatascience.com/a-beginners-guide-to-sentiment-analysis-in-python-95e354ea84f6.

2. Shung, Koo Ping. "Accuracy, Precision, Recall or F1?" Medium, Towards Data Science, 10 Apr. 2020. Accessed 7 Dec. 2022. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

3. Rate My Professors. https://www.ratemyprofessors.com/.

4. The pandas development team. "Pandas-Dev/Pandas: Pandas." Pandas Documentation, 1.5.2, Zenodo, Feb. 2020, pandas.pydata.org/docs. Accessed 7 Dec. 2022.

5. Perdregosa, F., et al. "Scikit-Learn: Machine Learning in Python." Journal of Machine Learning Research, vol. 12, 2011, pp. 2825–2830.