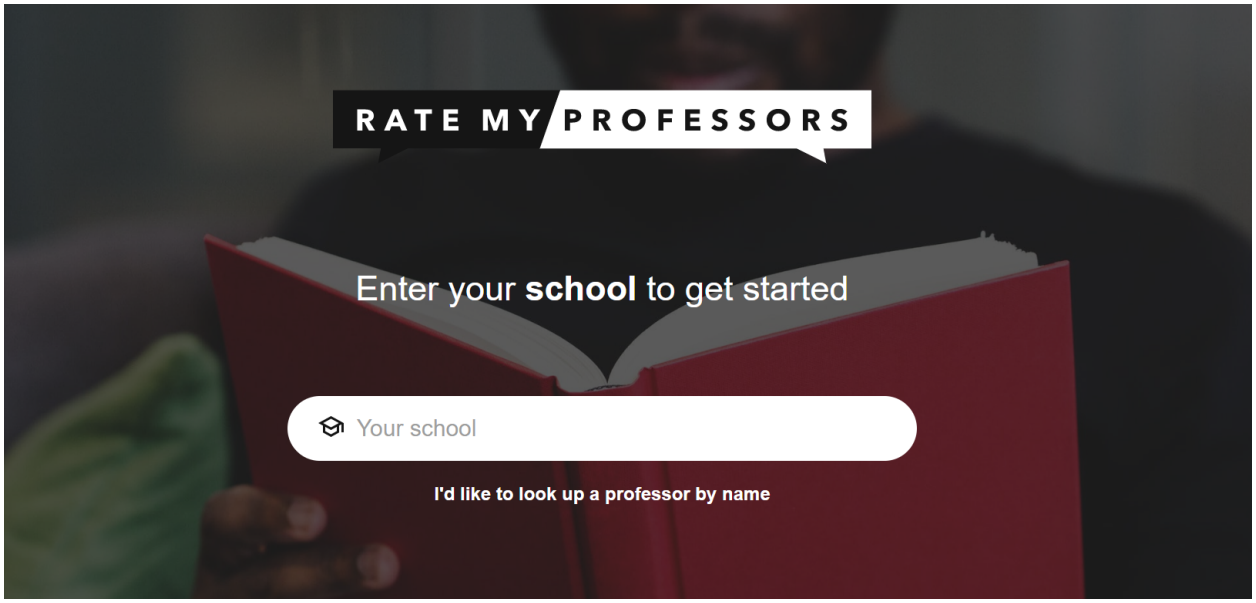
University Selector based on Student Preference

‘Aligns a student to a college and professor of their choice’



By:
Aarushi Sharma
Akshita Khandelwal
Benjamin Rinella
Nithin Raj Rangaraj
Peter Warzecha

Fordham University
Gabelli School of Business

Date: December 21, 2022

Page of Contents

Acknowledgement

Without the able guidance of our mentor Dr. Apostolos Fillipas this project would not have been possible. We would like to express our warm regards and humble gratitude to him for his valuable suggestions and feedback throughout the course of our implementation and analysis phase.

Also we would like to express our gratitude to Fordham University for providing us the opportunity to work on a course project during the Fall semester. This project was very helpful in getting practical exposure to the tools we learnt in class and applying them to work on a meaningful, team oriented project.

Executive Summary

Rate My Professors, an open anonymous platform for students to rate their school and teachers by quality points, has remained the norm for universities.

Student ratings and reviews on Rate My Professors reflect the realistic learning experiences of students regarding their professors and universities. Such information provides a large-scale data source to analyze the ratings by scraping the website.

- ➢ Scraped Rate My Professors website to create a vertical step down approach for students when choosing their colleges and professors for their classes based on preferences, subjects, ratings, and past experiences of previous students
- ➢ Scraped Twitter reviews of Rate My Professors official handle using Twitter API for analyzing student responses towards the website

Introduction

Rate My Professors:

One of the top websites for professor ratings is Rate My Professor. This website now has ratings from around 15 million visitors. Additionally, it offers databases for more than 1.4 million professors and more than 7000 educational institutions. These schools come from the US, UK, and Canada, among other countries. The scores range from 1 to 5, with 5 being the highest.This website was created for college students. Learning from others' experiences encourages interaction between students and helps them determine what is best for themselves. As a result, it is simple to use and better meets the demands of the pupils.

Utilizing Rate My Professors couldn't be easier. Simply enter the institution's name or the professor's name to rate them and discover more about them. This will give you a general evaluation of the lecturer and reveal whether the students plan to enroll in the course again or not on the basis of ratings. Additionally, you can post your ratings.

As a result, it is a fantastic way to improve the educational system and help students and educational institutions alike appreciate esteemed professors. Students are frequently reluctant to provide reviews, but this platform enables them to do so anonymously. Although Rate My Professors is the most popular website for reviewing professors, it is not the only one. Numerous additional websites exist that don't solely focus on ratings. For this reason we not only scraped the ratings of universities and professors from the Rate My Professors but also scraped twitter reviews with their official handle to better understand how students utilize the website.

Business Goal Analysis

Education is one of the most important things in a person's life and often they are anxious about being able to choose the college which will be a right for them.

There are several websites on the internet which can give them a huge load of information about different colleges, comparison tools and rankings based on several sets of parameters. However, we intend to make this experience personalized and customizable according to their priorities in terms of parameters for choosing a college.

For this purpose, we have considered several parameters which are important during university selection and given the user the opportunity to assign a certain weight/importance to them.

The user can list their priorities in the order they prefer and based on that a university rating would be generated. This rating would be based on the user choice and then the user will get the chance to pick the college which they find most suitable for them.

- Currently Rate My Professor uses anonymous feedback to create a rating for each college and each professor

- Our goal was to use this data to create an interface that students could use faster with filters and make it more efficient

    - This led to us to create our own ranking system that created a new overall rating for colleges with personalized weights based on the user's preference

- RMP was one of the only public websites that had documented and detailed personal experiences that could be used to rank colleges and professors

- Currently RMP does not use and filters for searching

    - Relies on the user to manually type in college names and professors

- The project is currently handled to only work in the tri-state area (Connecticut, New York, New Jersey) but has the potential to be used on a national level.

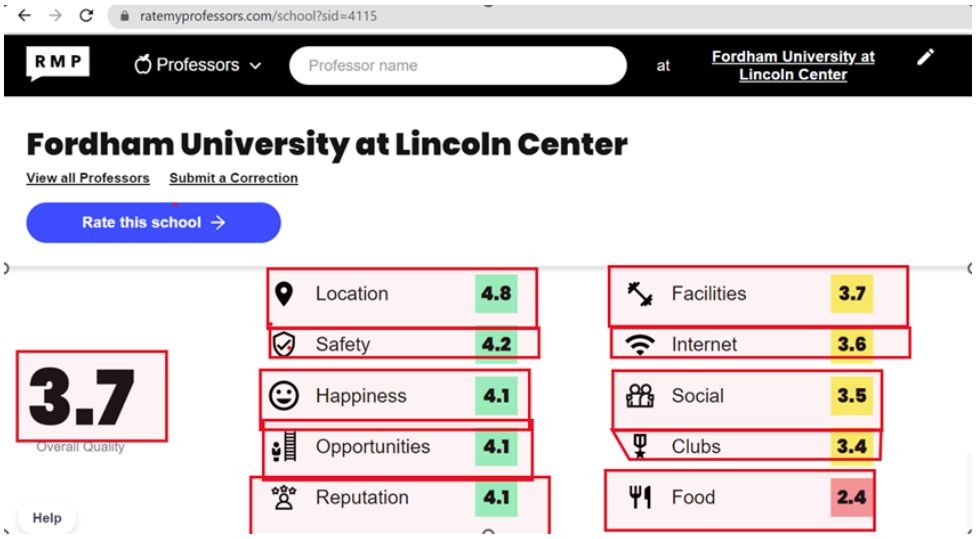
To build a student centric system like this we needed data and with that pursuit in mind we used web scraping and analytics to fetch and retrieve this data to develop the system.

Dataset Description

Three different datasets were obtained during the creation of this project. Two of the datasets were developed to work together to achieve the business goal of assigning a user to a school which matches their desires, and subsequently offering them a series of highly acclaimed professors to which the potential student could be assigned. The final dataset is a series of tweets regarding Rate My Professor with the goal of analyzing the text data within the tweets and getting a better understanding of the public opinion of the website which this project revolves around.

Dataset One: University Rating Data

This first dataset is made up of the 126 schools, hand-picked for this project, in the state of NY and the surrounding NYC area. For each school, 11 features were grabbed, along with the school's name, from their individual webpages. A snapshot of the Fordham Rate My Professor webpage is shown below with the 11 features highlighted.



Subsequently, the .csv file created through the relayed information contained 12 columns: School Name, Overall Rating, Facilities, Reputation, Opportunities, Happiness, Location, Safety, Clubs, Social, Internet, and Food.

As a result of the features being presented on each individual web page in order of highest score to lowest, the text title of each score (i.e., Location, Safety, Happiness …) was required to be scraped first, before being assigned to the correct list in which the corresponding float between 0 and 5 was entered.

In addition to these 12 fields, a 13th has the possibility to be created by the user in accordance with their specified preferences. This new field would exist after the user is asked whether they would like to enter their top five features, ranked from most important to least important. Should the user agree, a new 13th field would be created entitled 'Weighted Overall Rating'.

Weighted Overall Rating:

The weighted overall rating is an amalgamation of the 10 individual features Location, Safety, Happiness, Opportunities, Reputation, Facilities, Internet, Social, Clubs, and Food, weighed in accordance with the user's specifications. The breakdown of the weights is as follows:

The five features ranked by the user:

1. 23%
2. 20%
3. 15%
4. 10%
5. 7%

The remaining 5 features not specified by the user:

6. 5%
7. 5%
8. 5%
9. 5%
10. 5%

Hence, a 13th field containing a score specific to each user for every university.

Dataset Two: University Professors

The second dataset is an ever-changing collection of professors for any specified university existing in the first dataset. Scraped with selenium, which worked to both click past the 3 or more ads which often popped up as a web page loaded and to click the 'show more' button on the bottom of the webpage required to access more professors, this dataset contains 4 fields consisting of the professor's name, their rating as a float ranging from 0.0 – 5.0, the professor's number of ratings, and the subject which the professor teaches.

The total number of professors in this dataset varies by university and ranges anywhere from a few dozen to well over 1000. The code will continue to grab professor names until there are either no professors left to grab, or selenium has clicked the 'show more' button 500 times.

Though the number of professors appearing can vary with each click, some trials showed that the reaching of 500 clicks forced the appearance of around 3000 professors.

A picture of one version of the dataset for professors from Princeton University is presented below.

Out[17]:

	Name	Rating	Number of Ratings	Subject
0	Joshua Katz	5.0	10 ratings	Classics
1	Jennifer Johnson	4.1	5 ratings	Mathematics
2	Morton Kostin	2.8	5 ratings	Engineering
3	Robert Bagley	4.5	5 ratings	Art History
4	Ron Comer	4.3	14 ratings	Psychology
...	...	...	...	...
277	Kaiwen Luo	0.0	0 ratings	Physics
278	Jason Yonover	0.0	0 ratings	Philosophy
279	Nieng Yan	0.0	0 ratings	Biology
280	Frances Lee	0.0	0 ratings	Political Science
281	HV Poor	1.0	1 ratings	Engineering

282 rows × 4 columns

Dataset Three: Rate My Professor Tweets

The third dataset was extracted from Twitter by using the Twitter API; the official twitter handle of Rate My Professor website was used for this purpose. The goal of this was to analyze the reviews of students so that a better understanding of the significance of the Rate My Professor website among students, professors and institutions could be made.

The dataset had the following attributes:

Each extracted tweet is known as a tweet object, and it has a set of features associated with it. The features are as follows:

1. **tweet.user -** gives tweet object

2. **tweet.id -** gives id of the user posting the tweet

3. **tweet.full_text** - gives the actual content of the tweet

4. **tweet.created_at -** gives the date time information

5. **tweet.source -** gives the source of origin for tweet

6. **tweet.retweet_count -** gives the count of number of retweets for a tweet

7. **tweet.favorite_count -** gives the number of likes given to a tweet

	User	UID	Tweet_Text	Tweet_Time	Tweet_Source	Number_of_Retweets	Number_of_Likes
0	User(_api=<tweepy.api.API object at 0x0000015A...	1574183353804201984	RT @justtmythoughtz: All I need is the profess...	2022-09-25 23:45:33+00:00	Twitter Web App	3	0
1	User(_api=<tweepy.api.API object at 0x0000015A...	1567591865930711046	RT @DrunkoffTito: I have to check every teache...	2022-09-07 19:13:19+00:00	Twitter Web App	3	0
2	User(_api=<tweepy.api.API object at 0x0000015A...	1567591779888762881	RT @tmonson23: Biggest advice for incoming col...	2022-09-07 19:12:59+00:00	Twitter Web App	7	0
3	User(_api=<tweepy.api.API object at 0x0000015A...	1567591445053247494	RT @xolopiyotl: This is the funniest rating di...	2022-09-07 19:11:39+00:00	Twitter Web App	3	0
4	User(_api=<tweepy.api.API object at 0x0000015A...	1567591417899270145	RT @suhamalik_: i've watched ppl in two separa...	2022-09-07 19:11:33+00:00	Twitter Web App	4	0
...	...	...	...	...	...	...	...
171	User(_api=<tweepy.api.API object at 0x0000015A...	1346132148345069573	RT @susancsuarez: Honestly @ratemyprofessor is...	2021-01-04 16:31:28+00:00	Twitter Web App	1	0
172	User(_api=<tweepy.api.API object at 0x0000015A...	1343568407837487104	RT @Amreyes7Reyes: I wish someone would have t...	2020-12-28 14:44:05+00:00	Twitter Web App	1	0
173	User(_api=<tweepy.api.API object at 0x0000015A...	1342106676950872065	@KennedyBrackett if you were signed in you can...	2020-12-24 13:55:41+00:00	Twitter Web App	0	0
174	User(_api=<tweepy.api.API object at 0x0000015A...	1342105774810607618	RT @veerose__: My sociology prof emailed us an...	2020-12-24 13:52:06+00:00	Twitter Web App	1	0

Further, the follower information for the Rate My Professor website was also included.

The follower information was displayed as a set of the following parameters:

1. **follower.id -** gives the id of the follower

2. **follower.name -** gives the name of the follower

3. **follower.screen_name -** gives the twitter handle name

4. **follower.location -** gives the location of follower

5. **follower.description -** gives the bio information of follower

6. **follower.url -** gives the twitter url for follower

7. **follower.followers_count -** gives the number of people who follow this person

8. **follower.friends_count -** gives the number of people whom this person follows

9. **follower.favourites_count -** gives the number of likes received on this person's post

Follower_ID	Follower_Name	Follower_Screen_Name	Follower_Location	Follower_Description	Follower_URL	Follower_Count	Followee_Count
1203474909671936001	Jennifer Greene	GenevieveVerte	Montclair, NJ	Children's book editor. Clarionite and Montcla...	None	218	261
1309325273867618304	Hassan Seyfi	hassan_seyfi			None	39	386
1244046454638940162	chris wall	chriswa02239120			None	0	6
1333455136849530881	Clark is looking for custom Barong Tagalog	WorkinRoundCKR		They/He | Student | Blogger | PH through and t...	None	16	100
1382419474146463745	Leena	LeenaLall2		sarcasm is my only defence	None	33	40
...	...	...	...	...	...	...	...
1600164151426433026	Emily Shalom	emilyshalom5			None	1	16
1167959739784974336	MJ ⚡	mjgrayyyy		Burna Boy || Dave || □□□|| IFB	None	2174	2236
1597970008868462594	Aaron	AS013788	Colfax, CA		None	5	27
955785811986976768	Michael Matuch	MatuchMichael	Buffalo, NY	Fordham 24 Go Bills @wfuvsports	https://t.co/yAtmlE2EvZ	85	260
770250281506471941	The Count	virtuebc		"It's necessary to have wished for death in or...	None	2	47

System Design

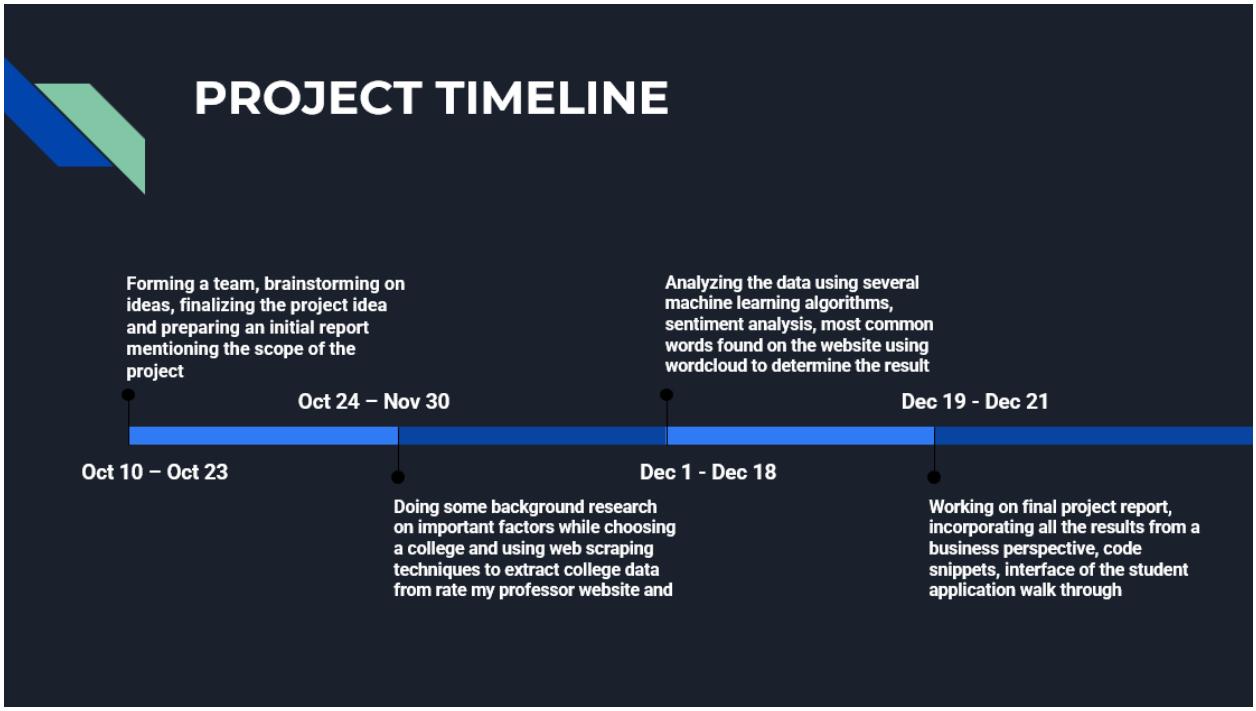
The project was carried out in several phases, each phase was significant towards the completion of the project. The various stages of the project are outlined below:

Stage 1: Ideation
In this phase we brainstormed on several ideas and went ahead with creating a customized student interface that can assist them in selecting universities and professors

Stage 2: Implementation
In this phase we worked on scraping the data from the website and extracted certain features for the colleges and professors.

Stage 3: Analysis
In this phase we used machine learning algorithms, sentiment analysis, wordcloud formation and twitter data to analyze the reviews.

Stage 4: Documentation
In this phase we created a report that highlights the implementation and analysis that was undertaken based on Rate My Professor website data.



System Implementation

The following tools, packages, libraries and modules were used while working on the implementation as well as the analysis of the project. They have been categorized based on the function they served during the course of the project.

The Web Scraping and Data Visualization has been implemented using Python in Jupyter notebook, while the analysis has been done using SPSS Modeler.

A data stream was created in SPSS Modeler which fetched data from the extracted csv file, which was then converted to xlsx format.

Post this the attributes were given their data types and the dependent and independent variables were determined in the type node. Further the partition node was used to split the entire dataset into the training and test component.

The training data was used to build a model and the testing data was used to check the efficiency of the generated model.

A. Web Scraping

- Pandas
- Numpy
- Selenium
- BeautifulSoup
- Time
- Re

B. Analysis using Machine Learning Algorithms

- C&RT Model
- Linear Regression Model
- Multiple Linear Regression
- Artificial Neural Network
- Vader Sentiment
- Wordcloud

C. Data Visualization

- Matplotlib
- Seaborn
- Wordcloud

Evaluation

With the ultimate goal of presenting a user with college choices the code is highly successful. Furthermore, the user is given the opportunity to narrow down their search based on features which are important to them. Once they have selected a school, they are then presented with highly ranked professors which teach classes within the user's major.

The highly specialized search for a school and its professors presents itself through two workbooks of code. The first scrapes Rate My Professor for a select group of schools within the New York state and City area based on a list created by the team. This data is then put into a .csv file to be opened in the second workbook to avoid the running of a time consuming chunk of code each time a search needs to be made.

A portion of the .csv file is shown below, portraying the school name, overall rating, and the ten characteristics which can be rated on Rate My Professor.

	School Nan	Overall Rat	Facilities	Reputation	Opportunit	Happiness	Location	Safety	Clubs	Social	Internet	Food
0	Baruch Coll	3.4	3.2	4	3.8	3.5	4.2	4.1	3.4	2.9	3.2	2.8
1	Fordham U	3.8	4.2	4.3	4.2	4.2	4.1	3.8	3.7	3.7	3	2.5
2	City College	3.2	3.3	3.5	3.3	3.3	3.3	3.2	3.1	2.9	3	2.6
3	Hunter Coll	3	2.8	3.6	3.1	3.1	3.8	3.9	2.8	2.5	3.1	2.4
4	CUNY Quee	3.3	3.6	3.6	3.2	3.4	3.5	4	3.1	2.8	3.1	2.8
5	The New Sc	3.5	3.5	3.9	3.8	3.6	4.7	3.8	2.7	2.7	3.7	3.5
6	Brooklyn C	3.4	3.7	3.7	3.4	3.7	3.7	3.5	3.2	3	3.1	2.9
7	Yeshiva Un	3.8	3.1	4.1	4.3	4	3.8	3.8	3.6	3.6	4	3.4
8	Fashion Ins	3.4	3	4.3	4.1	3.6	4.4	4	3.2	2.7	3.2	2.5
9	John Jay Co	3.7	3.7	4	3.8	3.9	4.1	4.4	3.5	3.2	3.3	3.3
10	Montclair S	3.3	3.5	3.5	3.4	3.5	3.8	4	3.2	3	3.1	2.8
11	New York C	2.8	2.6	2.8	2.8	2.9	3.4	3.7	2.6	2.5	2.5	2.5
12	Drew Unive	3.8	4.1	4	4.1	4.2	4.2	4.3	3.7	3.6	3.5	2.4
13	Manhattan	3.6	3.7	4	3.9	4	3.9	3.6	3.2	3.3	3.4	2.6
14	SUNY Mari	3	3.1	4.1	4.3	2.8	3.3	4.2	2.6	2.1	2	2
15	York Colleg	2.5	2.9	2.9	2.1	3.1	2.1	2.4	2.4	2.7	2.1	2.1
16	Marymoun	3.1	3.1	3.2	3.5	3.3	4.2	3.6	3.2	3	3	2.5
17	Hofstra Uni	3.8	4.1	3.7	3.8	3.9	3.3	3.6	3.8	3.5	3.8	3.3
18	Lehman Co	3.5	3.8	3.6	3.4	3.8	3.7	3.7	3.2	3.1	3.5	3.2
19	Pace Unive	2.9	2.7	3.7	3.4	2.2	3.2	3.6	2.3	2.4	2.3	2
20	Sarah Lawr	3.6	3.6	4.2	3.9	3.9	4.1	3.8	3.3	3	3.6	2.9
21	Vaughn Col	2.7	2.8	2.6	3.2	2.4	2.4	2	3	2.2	2.4	2.2
22	College of S	3.1	3.3	2.9	3.1	3.3	3.3	3.8	2.9	2.8	3	2.6
23	Medgar Eve	3	3	2.9	2.9	3.2	3.2	3.1	2.7	2.8	3.2	2.9
24	Stevens Ins	3.7	3.5	4.1	4.2	3.8	4.8	4.5	3.6	3.4	3.6	3.2
25	Stevens Ins	3.7	3.5	4.1	4.2	3.8	4.8	4.5	3.6	3.4	3.6	3.2
26	Purchase C	3	2.7	3.3	3.2	3.4	3.2	3.3	3.1	3.1	2.8	2.5
27	Kean Unive	3.6	3.9	3.5	3.5	3.7	3.7	4.2	3.4	3.2	3.2	3.2
28	St. Francis (	3.3	3.3	3.3	3.3	3.3	4	4.2	3	2.8	3.4	3.2
29	Molloy Coll	3.5	3.3	4	3.7	3.6	3.6	4.5	3.3	3	3.3	2.8

This is the file which has been used for doing predictive modeling using SPSS modeler.

When the .csv file is opened in the second workbook and converted into a dataframe it appears as it does below, presenting the 126 schools selected by the group.

```
In [40]: school_data_dataframe
```

Out[40]:

	School Name	Overall Rating	Facilities	Reputation	Opportunities	Happiness	Location	Safety	Clubs	Social	Internet	Food
0	Baruch College	3.4	3.2	4.0	3.8	3.5	4.2	4.1	3.4	2.9	3.2	2.8
1	Fordham University - Rose Hill	3.8	4.2	4.3	4.2	4.2	4.1	3.8	3.7	3.7	3.0	2.5
2	City College of New York	3.2	3.3	3.5	3.3	3.3	3.3	3.2	3.1	2.9	3.0	2.6
3	Hunter College	3.0	2.8	3.6	3.1	3.1	3.8	3.9	2.8	2.5	3.1	2.4
4	CUNY Queens College	3.3	3.6	3.6	3.2	3.4	3.5	4.0	3.1	2.8	3.1	2.8
...	...	...	...	...	...	...	...	...	...	...	...	...
121	Bloomfield College	2.9	2.9	3.1	3.0	2.7	3.4	3.9	2.7	2.5	2.6	2.3
122	Eastwick College	3.4	3.8	3.8	3.5	3.5	4.3	3.5	2.3	2.5	4.0	3.3
123	Berkeley College	3.0	3.0	3.0	3.0	3.3	3.3	3.4	2.4	2.7	3.3	2.7
124	Pillar College	3.6	3.4	4.3	3.7	4.3	3.9	4.0	2.9	3.6	4.4	2.7
125	Thomas Edison State University	3.1	3.1	3.4	3.3	3.5	3.7	3.7	2.4	2.5	3.6	2.8

126 rows × 12 columns

User Experience Example:

The easiest way to express the successfulness of the code is through an example. Allow the entrance of John Smith, an incoming undergraduate student looking to study education in New York.

John Smith is looking for a step up in the highly competitive culture which is public education and has certain preferences when looking for a school. When asked what his five most important factors are he states they are opportunities, reputation, location, safety, and facilities, in that order.

John Smith's inputs are expressed in the snapshot of the working code below:

```
Would you like to have a preferred features for your college recommendation?
>>yes
Please enter your top 5 preferences, ranked most important to least important
Here are the feature options:
 ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location', 'Safety', 'Clubs', 'Social', 'Internet', 'Food']

What is your most important feature?
>>Opportunities
Here are the feature options:
 ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location', 'Safety', 'Clubs', 'Social', 'Internet', 'Food']

What is your second most important feature?
>>Reputation
Here are the feature options:
 ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location', 'Safety', 'Clubs', 'Social', 'Internet', 'Food']

What is your third feature?
>>Location
Here are the feature options:
 ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location', 'Safety', 'Clubs', 'Social', 'Internet', 'Food']

What is your fourth feature?
>>Safety
Here are the feature options:
 ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location', 'Safety', 'Clubs', 'Social', 'Internet', 'Food']

What is your last feature?
>>Facilities
```

Hence, based on John Smith's desires, his top three schools are presented, ranked based on a 'Weighted Overall Rating' calculated through his inputted preferences.

```
Here are your top three colleges
```

Out[45]:

	School Name	Overall Rating	Facilities	Reputation	Opportunities	Happiness	Location	Safety	Clubs	Social	Internet	Food	Weighted Overall Rating
109	Princeton University	4.4	4.5	4.7	4.4	4.4	4.2	4.9	4.2	4.2	4.0	4.2	4.437
58	Cornell University	4.4	4.6	4.8	4.5	4.3	3.5	4.4	4.3	4.0	4.4	4.4	4.352
69	St. Lawrence University	4.3	4.4	4.5	4.4	4.5	3.2	4.6	4.1	4.3	4.2	4.1	4.220

When John Smith is forced to choose which school of the three for professors to appear in, he selects St. Lawrence University in Canton, NY.

```
Please enter the school of your choice
>>St. Lawrence University
```

In [47]:
```
######## This is foundation for scraping professor grades
```

Through John's selection of St. Lawrence University, the code finds the school's corresponding url id and scrapes the school's professors for as many as 500 pages, or until there are no professors remaining. A presentation of the dataframe created through such scraping is shown below.

In [48]:
```
Rate_My_Professor_Dataframe
```

Out[48]:

	Name	Rating	Number of Ratings	Subject
0	Kate Susman	4.1	16 ratings	Biology
1	Judith Nichols	4.4	26 ratings	English
2	Fred Chromey	3.5	15 ratings	Astronomy
3	Everett Weedin	3.3	33 ratings	English
4	Peter Pappas	3.3	26 ratings	Mathematics
...	...	...	...	...
426	Ryan Rybka	2.0	1 ratings	Anthropology
427	Nancy Pokrywka	3.3	24 ratings	Biology
428	Bojana Zupan	3.9	25 ratings	Psychology
429	Bryan Van Norden	4.5	64 ratings	Philosophy
430	Patricia Jones	4.1	8 ratings	Economics

431 rows × 4 columns

Finally, John Smith inputs his intended major, narrowing down the professors presented to those who teach within his intended major.

```
Please select your expected major
>> Education
['Business', 'Science', 'Engineering', 'Communications', 'Humanities', 'Religious_Studies', 'Arts', 'Education', 'Political_
Science_and_Law']
```

Out[64]:

	Name	Rating	Number of Ratings	Subject
304	Ah-Young Song	5.0	8 ratings	Education
423	Leonisa Ardizzone	5.0	3 ratings	Education
348	Maria Hantzopoulos	4.9	13 ratings	Education
219	Kimberly Williams Brown	4.8	6 ratings	Education
225	Jaime Del Razo	4.8	4 ratings	Education
111	Erin McCloskey	4.7	13 ratings	Education
61	Chris Bjork	4.6	19 ratings	Education
31	Chris Roellke	4.5	8 ratings	Education
74	Nancy Willard	4.5	7 ratings	Education

Thus, John Smith is presented with a school which meets his desired qualities and highly rated professors within his field of research. His college search has been simplified and hopefully John is satisfied with the results of the search.

This process wherein the user gets to choose the factors that are most important to them for university selection is the most suitable way for them to determine which school is appropriate for them. Education is a big investment in terms of time, money and effort, every person would want to go to a university where they can learn, enjoy and get a return on investment.

The interface developed above will be a means for users to customize the search based on their preferences as well as priorities.

Analysis

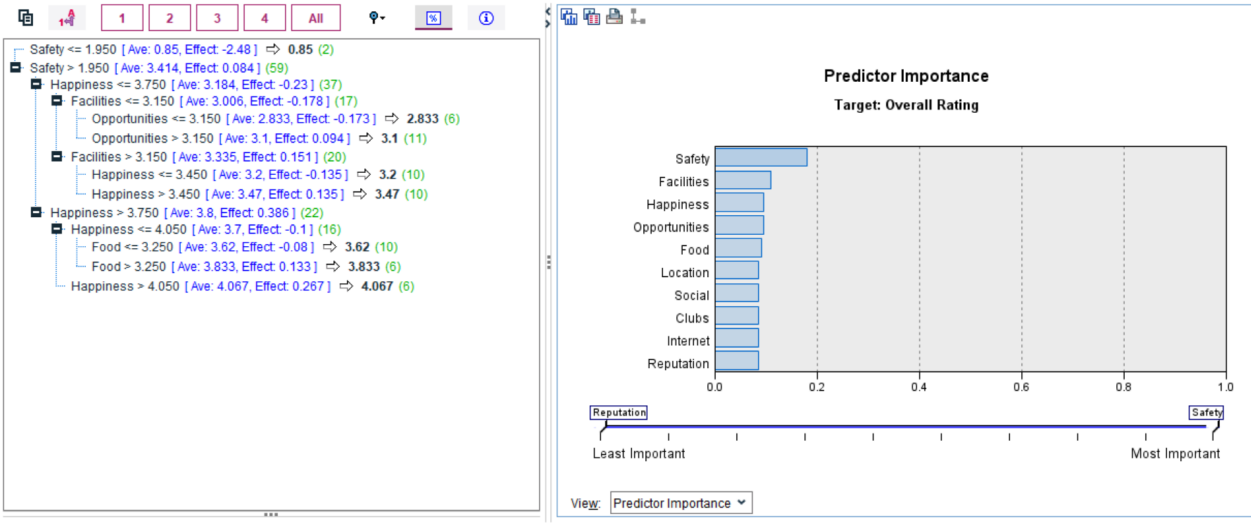
Predictive Model Building

Model Purpose

A huge variety of factors contribute to influencing the overall ratings of the universities. The features are continuous as they directly affect the overall rating and this allows them to build models in order to understand if the model predicts accurate results regarding which feature has more sway over the rating and how it impacts students' decision.
After scraping the data and collecting it into an excel file, various models were created to distinguish which model works best to help us meet our business goal.

C&RT Decision Tree

The number of records in the dataset is 126, we split the total data in 70 (Training) and 30 (Test). The predictor importance graph in the right pane displays the importance of features which impacts students preference. It can be observed that the most important feature is **SAFETY** for students. Moreover, the other features are nearly of equal importance to each other. However, the nodes in the left pane define that Facilities, Happiness, Opportunities and Food have fair importance.



In the decision tree, the overall rating with other independent variables, the main evaluation parameter is Linear Correlation value. The 0.937 of Linear correlation means **93.7%** of the model predicted value is along with the real value in ratings with a mean absolute error of **0.127.**

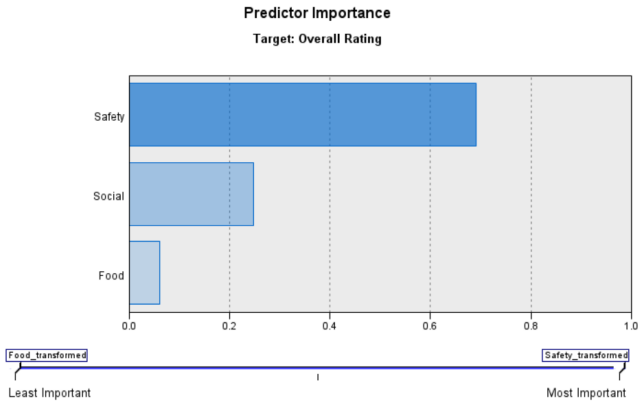
Results for output field Overall Rating

Comparing $R-Overall Rating with Overall Rating

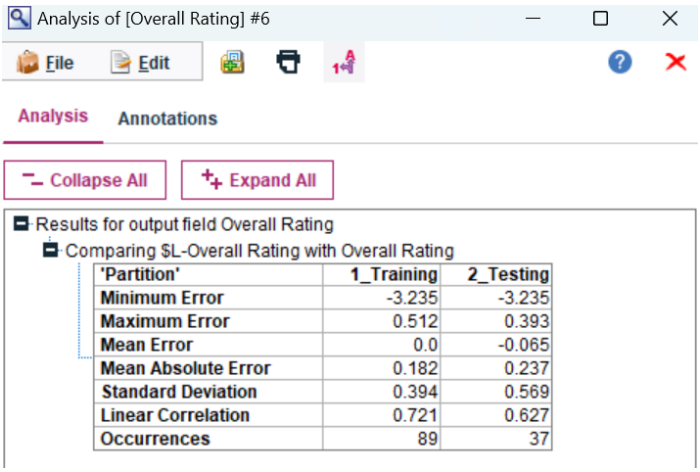
'Partition'	1_Training	2_Testing
Minimum Error	-0.85	-0.85
Maximum Error	0.85	0.333
Mean Error	-0.002	-0.052
Mean Absolute Error	0.127	0.152
Standard Deviation	0.198	0.222
Linear Correlation	0.937	0.964
Occurrences	89	37

Linear Regression

Linear regression models the relationship between the features and the model showed safety as the most concerned factor among students and food as the least bothered.

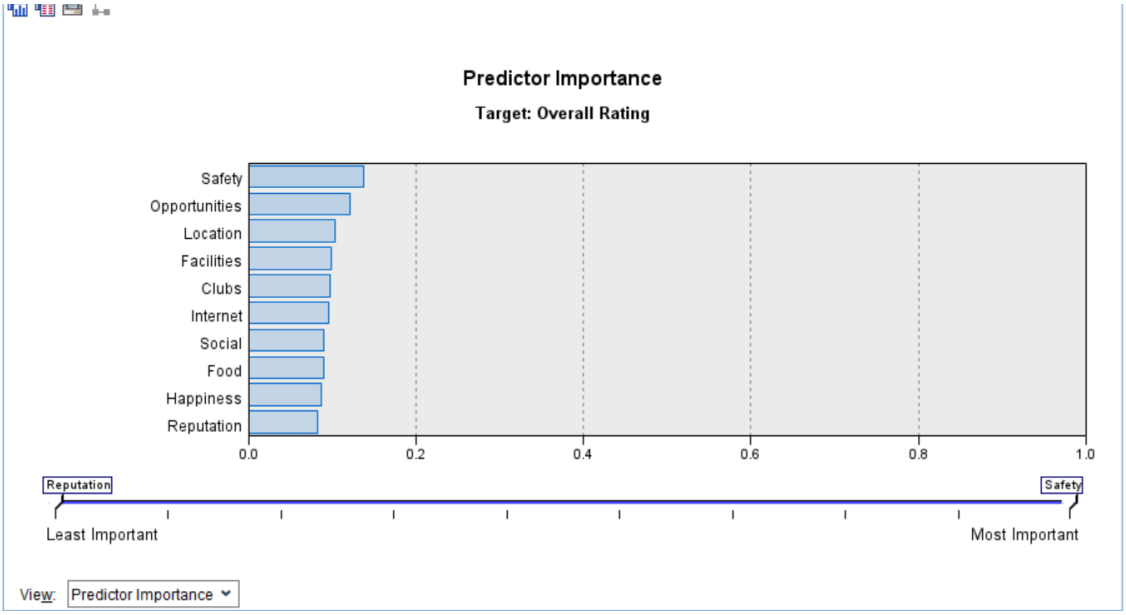


The accuracy of the model is 0.721 i.e **72.1%** which is the least among other models and mean absolute error is **0.182**



Multiple Linear Regression

Moving forward to understand the relationship between a single feature with overall ratings. Safety again as the most important variable and reputation of the university is the least concerned feature for the overall rating of the university.



The model demonstrates the linear correlation accuracy is 0.856 i.e **85.6%** with a mean absolute error of **0.18**

Results for output field Overall Rating

Comparing $E-Overall Rating with Overall Rating

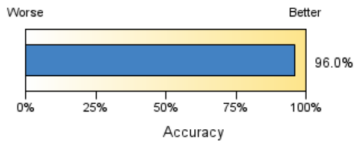
'Partition'	1_Training	2_Testing
Minimum Error	-1.932	-1.932
Maximum Error	0.604	0.74
Mean Error	0.005	-0.077
Mean Absolute Error	0.18	0.235
Standard Deviation	0.294	0.394
Linear Correlation	0.856	0.847
Occurrences	89	37

Artificial Neural Network

A task that a linear model cannot carry out can be carried out by an artificial neural network. Additionally, Artificial Neural Networks (ANN) develop algorithms that can be utilized to simulate complicated patterns and prediction issues by using the brain's processing as a starting point.

Model Summary

Target	Overall Rating
Model	Multilayer Perceptron
Stopping Rule Used	Error cannot be further decreased
Hidden Layer 1 Neurons	5

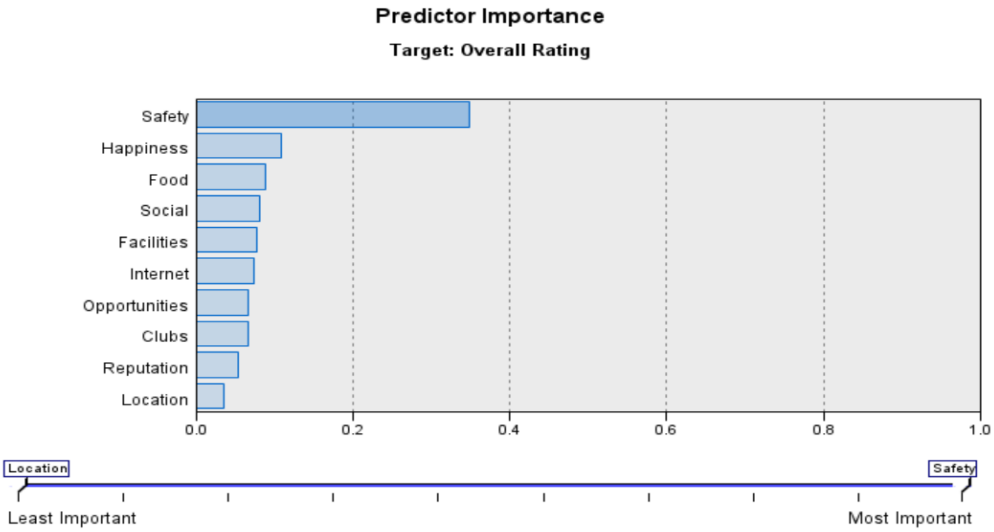


Results for output field Overall Rating

Comparing $N-Overall Rating with Overall Rating

'Partition'	1_Training	2_Testing
Minimum Error	-0.335	-0.335
Maximum Error	0.327	0.279
Mean Error	-0.006	-0.072
Mean Absolute Error	0.085	0.13
Standard Deviation	0.113	0.141
Linear Correlation	0.98	0.982
Occurrences	89	37

The model has an accuracy of **96%** which fits best to successfully predict our business goal. Moreover, the mean absolute error of the model is **0.085**



Overall Analysis to determine most model:

The following table shows the analysis of the 4 machine learning models which were applied to the ratings dataset.

	C&RT Decision Tree	Linear Regression	Multiple Linear Regression	Artificial Nerual Network
Mean Absolute Error	0.152	0.237	0.235	0.13
Standard Deviation	0.222	0.569	0.394	0.141
Linear Correlation	0.964	0.627	0.847	0.982

After analyzing the data with all the four models, it's very clear that **ANN is the best performing model** with the highest Linear Correlation of **98.2%** and with the lowest Mean Absolute Error of **0.13** and Standard deviation of **0.141**

The most important predicting factor of this model is the Safety and location as the least.
The Accuracy of this model is **96%** which is also the highest among every other model.

Sentiment Analysis

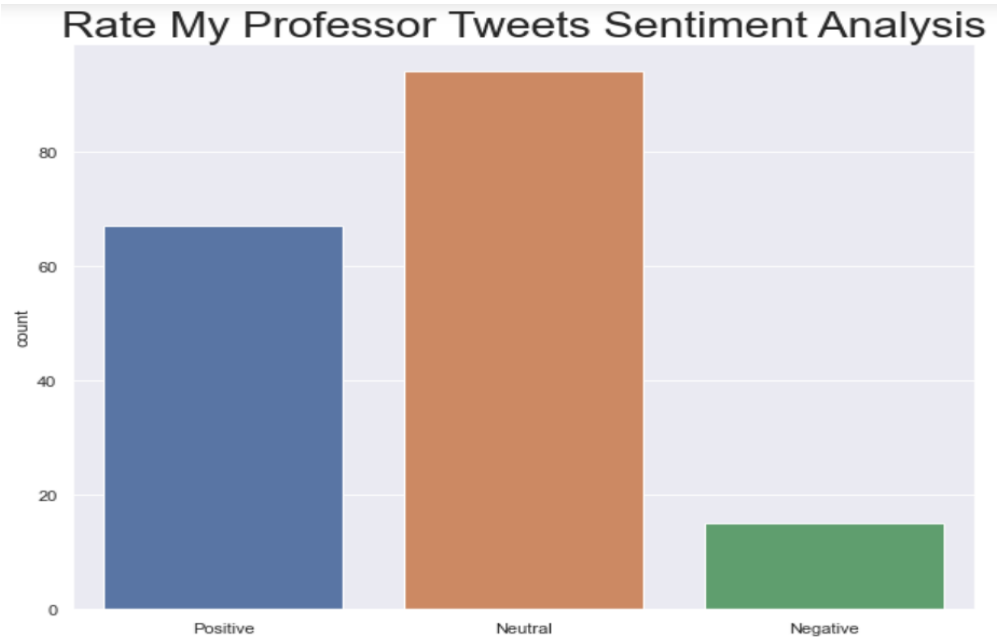
Rate My Professor is a very popular website among students for choosing their courses as well as professors.

Twitter is one of the platforms wherein most of the students are very active and expressive about voicing their opinions regarding a particular course/professor.

The following tasks were carried out in this analysis:

- We used Twitter API to extract the most recent reviews from the rate my professor official handle so as to do sentiment analysis on the kind of tweets which are being posted by students.
- With the help of this we can analyze the significance of these reviews and the kind of sentiment they bring about
- A majority of the reviews are on a neutral level, followed by a relatively decent number of reviews with a positive context and very few reviews that had a negative aspect to it.
- Some of the most common words in the tweets were - college, professor, student, class, time and rated.

Sentiment Score breakdown for Rate My Professor Tweets



100 Most Common words reflecting on Rate My Professor Twitter handle

The rate my professors website is used by most of the students in the US and based on our analysis these are the top 100 words which appeared in the tweets posted on their twitter handle.

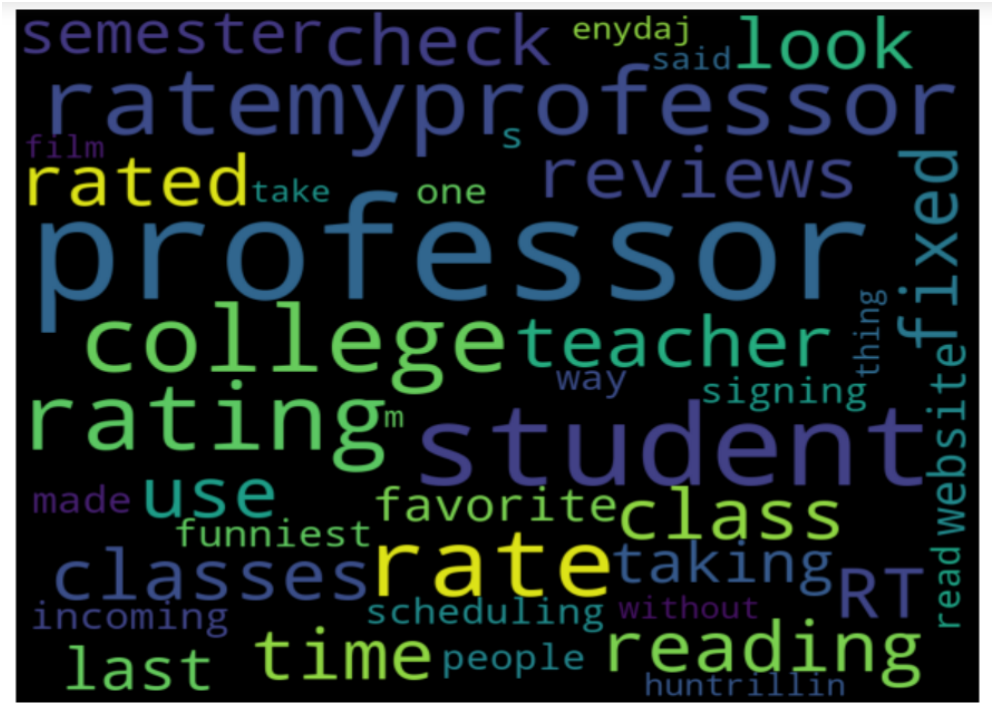
We can see that the emphasis is on professors, teachers, colleges, semester and classes which means that students are interested in sharing their opinions about the experience they had and likewise others are keen on knowing about these topics.

```
['RT', 'my', 'rate', 'professor', 'on', 'I', 'the', 'a', '@ratemyprofessor', 'for', 'to', 'and', 'is', 'reviews', 'professors',
'in', 'Rate', 'of', 'My', 'you', 'classes', 'are', 'that', 'your', 'college', 'from', "we're", 'check', 'students', 'this', 'i
f', 'before', 'time', 'just', 'when', 'how', 'reading', 'do', 'not', 'at', 'up', 'all', 'about', 'class', 'it', 'ratemyprofesso
r', 'so', 'use', 'was', 'fixed!', 'have', 'teacher', 'rating', 'can', 'their', 'rated', 'look', 'student', 'semester', 'by', 'l
ike', 'professor.', 'be', 'last', 'Professor', 'me', 'been', 'taking', 'favorite', 'with', 'All', 'website', 'incoming', 'stude
nts:', 'it's', 'funniest', 'College', 'out', 'made', 'scheduling', 'Professor,', '😊', 'get', 'those', 'people', 'signing', 're
ad', 'one', 'ratings', 'way', '@huntrillin:', 'thing', 'without', 'would', '@enydaj', 'take', 'film', 'said', 'i', 'I'm']
```

Based on this analysis, a few more insights which we get are that students talk about other factors such as favorite which could be for a university, professor or course. Often students are known to go to colleges or take professors that are most liked by other students.

Wordcloud displaying the most common words on Rate My Professor twitter handle

This visual is a good way to analyze the most common words on the RMP official twitter page



Rate My Professor Twitter Follower Analysis

This analysis can be helpful in understanding the audience demographics and their preferences.

Follower_ID	Follower_Name	Follower_Screen_Name	Follower_Location	Follower_Description	Follower_URL	Follower_Count	Followee_Count
1203474909671936001	Jennifer Greene	GenevieveVerte	Montclair, NJ	Children's book editor. Clarionite and Montcla...	None	218	261
1309932573867618304	Hassan Seyfi	hassan_seyfi			None	39	386
1244046454638940162	chris wall	chriswa02239120			None	0	6
1333455136849530881	Clark is looking for custom Barong Tagalog	WorkinRoundCKR		They/He | Student | Blogger | PH through and t...	None	16	100
1382419474146463745	Leena	LeenaLall2		sarcasm is my only defence	None	33	40
...	...	...	...	...	...	...	...
1600164151426433026	Emily Shalom	emilyshalom5			None	1	16
1167959739784974336	MJ	mjgrayyyy		Burna Boy || Dave || □□□|| IFB	None	2174	2236
1597970008868462594	Aaron	AS013788	Colfax, CA		None	5	27
955785811986976768	Michael Matuch	MatuchMichael	Buffalo, NY	Fordham 24 Go Bills @wfuvsports	https://t.co/yAtmiE2EvZ	85	260

Potential Flaws and Setbacks

First Setback:

Just as any beta designed code, our Rate My Professor code had several complications that stopped it from reaching its full potential. One of the biggest setbacks we faced was dealing with how the URL changed for each college listed on Rate My Professor.

When users enter the college they would look to read about, the url encompasses a unique school id instead of the name for that college. Here's an example: student A would like to read feedback from previous students who attended Adelphi University. When they search and click on Adelphi, this is the following url for that Rate My Professor page.

As you can see, the final number in the code is a 6. This six is the unique school id associated with the school Adelphi University. We believe that this list was created alphabetically due to Adelphi and other schools beginning with the letter "A" having lower unique school ids.

Even though we had filled in some blanks, the main issue still existed being that we did not have a full identical list of schools correlated to the unique ids assigned by Rate My Professor. The only solution would be to loop the code for every number between 1 and the amount of schools existed on Rate My Professor.
With Rate My Professor having over ten thousand schools, the code would take days before it could create a full list of all colleges on the website. So as a group, we were left with two options:

1. Make the range for the loop something a little more reasonable like 1,000 and use those schools.

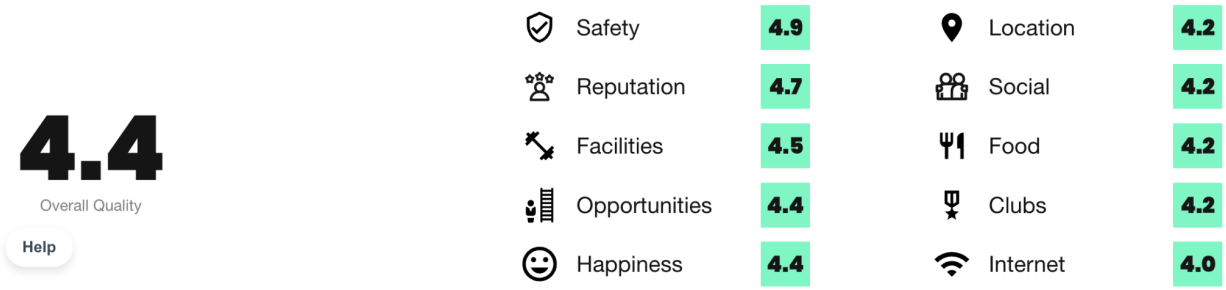
The only problem with this solution is that most of the schools recommended would just be schools starting with the letters A-D.

2. Create a manual list of schools based on a location preference.

We ended up choosing this option because it gave our code more precision when recommending colleges. We were able to encompass over 100 colleges in the tri-state area (New York, New Jersey, Connecticut) and use that as our baseline for the location aspect of our recommendations. With less colleges included in our code to analyze, our project became a lot more scalable.

Second Setback:

This of course led to a different issue, which was the lack of data in our analysis. When recommending colleges, there were a few highly reputable and well known colleges that received very high grades across the boards for all their features listed. For example, we can look at the school Princeton:

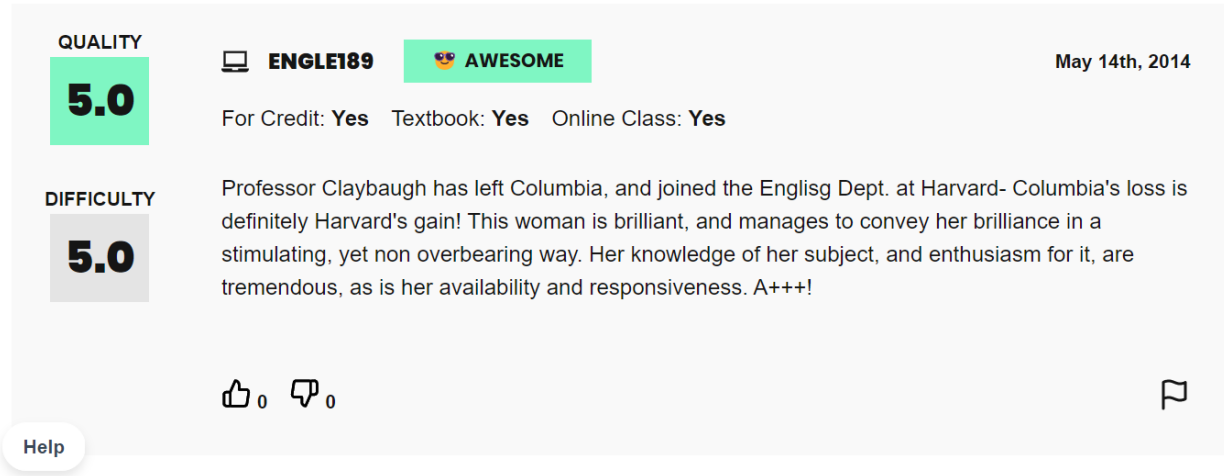


Princeton had above average grades for a majority of its features, and due to the lack of scraped data in our code, would be recommended under most circumstances. This could possibly be avoided if we had scraped more colleges, but the tri state area already had a limited number of colleges. Therefore, we accepted the fact that since schools like Princeton had done an exceptional job at supplying their students with the following features, they were and should always be one of the top schools recommended. If our data was expanded to a national level instead of just the tri state area, this would solve this issue.

Third Setback:

Another potential flaw we ran into in our program was when we scraped Rate My Professor for professors within a school. As mentioned before, the website contains anonymous feedback that is mostly accepted as true personal experiences. But there is nothing that validates these reviews as true and genuine. For all we know, there could be some professors who created accounts on the website and reviewed themselves, giving themselves very high ratings (An issue pointed out by existing users) when they did not deserve high ratings and could/should be rated lower.

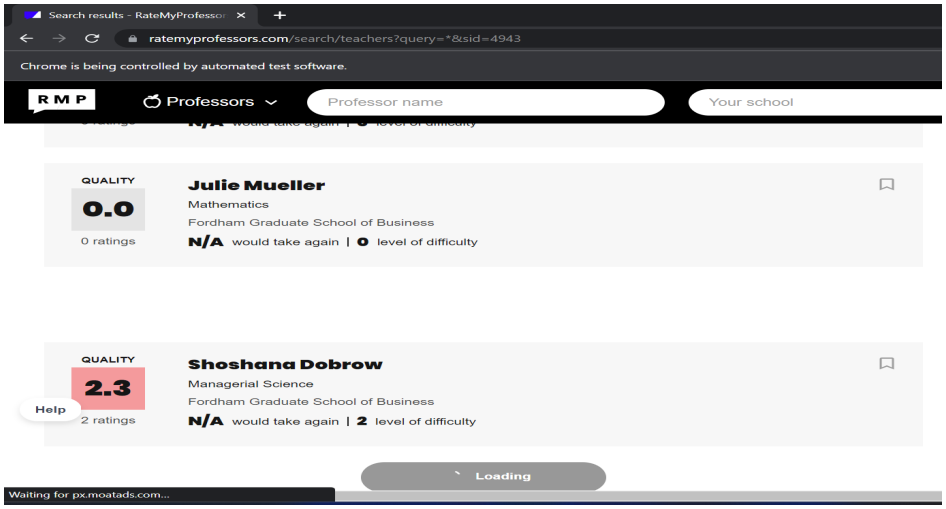
This would ruin the integrity of our code as it fully relies on the reviews being accurate to fulfill its duty of providing some of the highest rated professors. Nonetheless, this was out of our hands as we had to rely on the data we scraped to be factual, and for the most part it seemed that there were very little incidences of collusion.



Final Setback:

The final flaw in our code existed within the ads that consistently popped up during the process of running our code. In order to scrape all professors within a specific school, we needed to use Selenium with the help of ChromeDriver to manually click the "Show More" button to expand the list. Rate My Professor is a free website, and therefore makes a lot of their revenue through advertising. When writing our code, we had to account for three different ads that would pop up and block the "Show More" button and break our code.

Our code would look for the advertisement we programmed it to look for and use Selenium to press the "x" button on the top right of the ad to close it. Unfortunately, the timing of when the ads came and the variation of ads would confuse our program. Therefore, there were times our code ran with no issues and there were times our code would break in the middle of the process due to dynamic changes with the advertisement's shape and size present on the website.



Conclusion and Future Direction

The main idea behind our project was to create a user interface that could filter colleges and professors on Rate My Professor based on preferences of the user. Our goal was to create a code that could execute our initial idea on a smaller scale and see if it was feasible. As seen in our evaluation, the code has been proven to work and can be relied on to perform the necessary analysis. Since our code is viable and can be depended on, the next steps are to increase the dataset that is scraped and ultimately used.

Instead of just using colleges in the tri-state area, we would like to take our code to a national level and include all colleges that have reviews on Rate My Professor. This would give users a more informed recommendation and not rely on a baseline location that we have set ourselves. It would also include a more pruned final list of recommendations that would fit the preferences of the user.

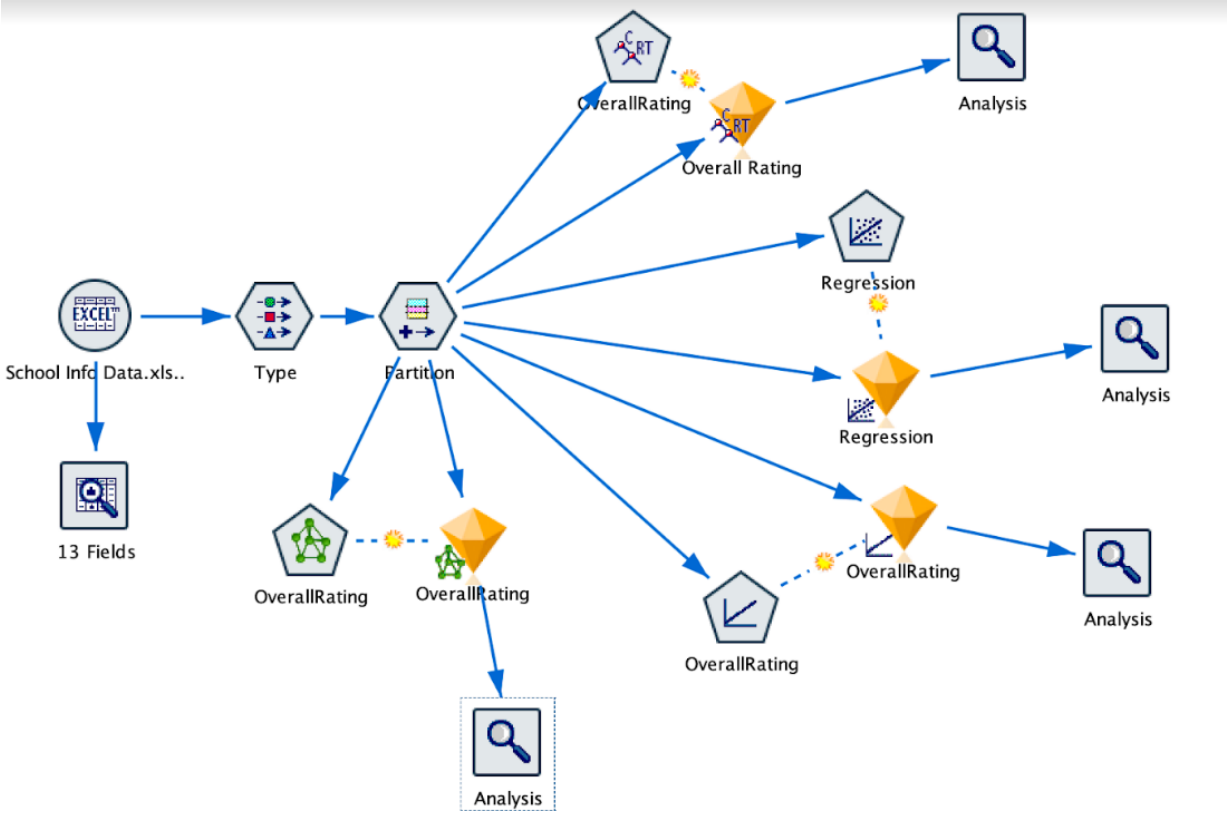
While the colleges we filtered for had 10 distinct features to be measured by, professors did not have many to work with. We used an overall professor rating as our primary numeric value interpreter for how good a professor is, but ideally we would like to expand on that to include more variables. Some current variables that the website already has are level of difficulty, top tags, whether a textbook is required, mandatory attendance, and whether the student would take the class again or not. These could play a huge role in defining our recommendations to be even more accurate. Some features that the website doesn't include but may in the future could be the amount of work the teacher assigns or the style of teaching by the professor. All these would have meaningful impacts on the recommendations provided by our code.

Though having additional features and modified changes can prove to be significant in future iterations of our code, the initial filtering implemented is already more than the website offers and could prove to be useful to a large portion of the American population. As time passes, the data included in the website content linearly grows each month. According to streaming news company Cheddar (Who owns Rate My Professors), "Rate My Professors is being used by more than 6 million college students per month, who write an average of 125,000 new professor and class ratings monthly. Seasonally, these figures peak at 7 million and 300,000, respectively." With this much data on hand, it is imperative for code to exist to help navigate users and make the process for looking for colleges and professors easier and more efficient. Our code does an excellent job of that and has the potential to increase in substantial value with minor future additions.

Our goal was to build an interface that would make the user journey easier and smoother while navigating through the challenging process of university selection. Given that we have an initial prototype, the scope of scalability and innovation is immense and can be explored in the future.

SPSS Modeler Data Stream

This is the data stream which was created in SPSS Modeler to conduct the analysis and predict the accuracy of the results we get.



References

These are the references to any tool, code, or research paper that we used in our project.

- https://www.perfectrec.com/
- https://www.researchgate.net/publication/348928106_Are_Top_School_Students_More_Critical_of_Their_Professors_Mining_Comments_on_RateMyProfessorcom
- Rate My Professors
- https://techcrunch.com/2018/10/25/cheddar-buys-a-user-generated-content-biz-rate-my-professors-from-viacom/

Appendix

 # Scraping Rate My Professor Website and extracting required data features

```
#Grabbing the school data

# Import necessary functions
import time
import re
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC

# Define function to help us click to next page
def check_exists_by_xpath(xpath):
    try:
        browser.find_element(By.XPATH,xpath)
    except NoSuchElementException:
        return False
    return True

# Dictionary of the universities which will become a part of the .csv file
# There is the school name correlated with the corresponding integer which applies to the Rate
# My Professor website url code
# 126 schools
```

school_number_dict = {"Baruch College": 222, "Fordham University": 1325, "City College of New York": 224,

"Hunter College":226, "Queens College of New York": 231, "The New School": 1519,

"Brooklyn College": 223, "Yeshiva University": 1223, "Fashion Institute of Techonology": 975,

"John Jay": 227, "Montclair State University": 630, "New York City College of Technology": 230,

"Drew Univeristy": 1314, "Manhattan College": 557, "SUNY Maritime": 976, "York College": 1224,

"Marymount University": 574, "Hofstra University": 413, "Lehman College": 228,

"Pace University": 12151, "Sarah Lawrence College": 883, "Vaughn College": 15545,

"College of Staten Island": 225, "Medgar Evers College": 229, "Stevens Institute of Technology": 982,

"Wagner College": 1129, "Purchase College": 970, "Kean Univeristy": 478, "St. Francis College": 3975,

"Molloy College": 623, "Plaza College": 2697, "Saint Elizabeth": 4160, "Monmouth University": 625,

"Saint John's Univeristy": 842, "Iona Univeristy": 451, "Saint Thomas Aquinas College": 4409,

"Mercy College": 592, "Monroe College": 2482, "Webb Institute": 1152, "Touro University": 1024,

"Metropolitan College of New York": 5585, "ASA College": 5069, "Manhattanville College": 558,

"New York University": 675, "Stony Brook University": 971, "Seton Hall Sniversity": 892,

"Adelphi University": 6, "Farmingdale State College": 14046, "Wells College": 1157,

"Niagara University": 678, "Columbia University": 278, "Hunter College": 226,

"Long Island University": 527, "SUNY Purchase": 970, "Rutgers": 4939, "New Jersey Institute of Technology": 668,

"Vassar College": 4070, "Fairfield University": 1351, "Seton Hall": 892, "Cornell University": 298,

"Colgate University": 252, "Hamilton College": 389, "University of Rochester": 1331, "Skidmore College": 907,

"Union College": 1044, "Binghamton University": 958, "Syracuse University": 992, "University at Buffalo": 960,

"St. Bonaventure University": 832, "Rochester Institute of Technology": 807, "St. Lawrence University": 847,

"Clarkson University": 240, "D'Youville College": 4638, "St. John Fisher University": 1628, "SUNY New Paltz": 964,

"Marist College": 563, "Le Moyne College": 506, "SUNY Oneonta": 966, "University at Albany": 957,

"Nazareth College": 661, "Ithaca College": 453, "Utica University": 4137, "SUNY Oswego": 967, "SUNY Plattsburgh": 968,

"SUNY Cortland": 961, "SUNY Maritime College": 976, "Canisius College": 175, "Hilbert College": 4499,

"SUNY Brockport": 1549, "Daemen College": 4135, "Siena College": 900, "Boricua College ": 4775,

"SUNY Delhi": 4414, "SUNY Geneseo": 963, "Elmira College": 332, "Hartwick College": 398, "York College": 1224,

"Yale University": 1222, "Wesleyan University": 1161, "Trinity College": 1030, "University of Connecticut": 1091,

"Quinnipiac University": 787, "Sacred Heart University": 827, "University of New Haven": 4159,

"Albertus Magnus College": 16, "University of Hartford": 1103, "University of Bridgeport": 1071,

"Mitchell College": 4485, "Goodwin University": 5260, "The College of Saint Rose": 14032,

"Paier College of Art": 5731, "Princeton University": 780, "The College of New Jersey": 256, "Rowan University": 822,

"Stockton University": 800, "Rider University": 801, "Ramapo College of New Jersey": 13744, "Caldwell University": 144,

"New Jersey City University": 4527, "Centenary University": 1627, "Cazenovia College": 190,
"Felician University": 4507, "Saint Peter's University": 864, "Bloomfield College": 4478,
"Eastwick College": 5574, "Berkeley College": 1723, "Pillar College": 4300, "Thomas Edison State University": 1016}

```
# The empty lists which will be filled with the information and ratings for each university

# Name of the School
schoolname = []

# Rate my professor created overall rating
overallratings = []

# The next ten are the features of the universities which can be rated on the website
reputation = []
opportunities = []
happiness = []
facilities = []
location = []
safety = []
clubs =[]
social =[]
internet = []
food = []

# The initial for code which will run through the dictionary of university names
for school in school_number_dict:

    # Grabs the number in the dictionary which is necessary for the rate my professor url for each school
```

```
    school_id = school_number_dict[school]
    # Turns the code into a string so it can be concatenated
    school_id = str(school_id)

    # This is my path to the chromedriver
    path = "C:\\Users\\benri\\Downloads\\chromedriver_win32 (4)\\chromedriver"
    s=Service(path)
    browser = webdriver.Chrome(service=s)

    # The url for each school is created here
    link = "https://www.ratemyprofessors.com/school?sid=" + school_id
    browser.get(link)
    time.sleep(1)

    # Close out the box talking about cookies
    try:
        if (check_exists_by_xpath('//img[@class="FullPageModal__StyledCloseIcon-sc-1tziext-0
eJtQsN"]')):

browser.find_element(By.XPATH,'//img[@class="FullPageModal__StyledCloseIcon-sc-1tziext-
0 eJtQsN"]').click()
            time.sleep(1)
    except:
        print("could not click")
        pass

    # Skip ad popup if present
    try:
        if (check_exists_by_xpath('//a[@class="bx-close bx-close-link bx-close-inside"]')):
            browser.find_element(By.XPATH,'//a[@class="bx-close bx-close-link
bx-close-inside"]').click()
```

```
        time.sleep(1)
    except:
        print("could not click")
        pass

    # Skip ad popup if present
    try:
        if (check_exists_by_xpath('//div[@class="IL_BASE IL_SR_BG"]')):
            browser.find_element(By.XPATH,'//div[@class="IL_BASE IL_SR_BG"]').click()
            time.sleep(1)
    except:
        print("could not click")
        pass



    # Creating a list of the html which contained the rankings of each of the school based on their features

    page_source = browser.page_source
    clam_chowder = BeautifulSoup(page_source, 'lxml')
    category_content = clam_chowder.find_all('div', class_ = 'CategoryGrade__CategoryGradeContainer-sc-17vzv7e-0 ivOAGg')

    # There was an extra two at the end of the list with the same class as above but which did not hold any ratings
    category_content.pop()
    category_content.pop()

    # The for loop grabbing all of the ratings for the 10 features
    for review in category_content:
```

```
        # The ratings all had the same class title for their respective titles so we had to get the name of the rating first
        # Each rating had a color background corresponding with their rating ie: 1/5 was red, 5/5 was green
        # Once we had the name of the category we then would approach each of the three corresponding classes
        # one for green, yellow, and red
        # And then grab the score depending on which class it was
        category = review.find('div', {'class':re.compile("CategoryGrade__CategoryTitle-sc-17vzv7e-1 XKroK")}).text.strip()

        # This is an example of the if statements for each of the 10 features
        if category == 'Facilities':

            # If the background is green
            if review.find('div', {'class':re.compile("GradeSquare__ coloredSquare-sc-6d97x2-0 fGdpBh")}):
                fac1 = review.find('div', {'class':re.compile("GradeSquare__ coloredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()

                # No longer a string
                # Adds it to its respective list
                facilities.append(float(fac1))

            # If the background is yellow
            elif review.find('div', {'class':re.compile("GradeSquare__ coloredSquare-sc-6d97x2-0 bwfltG")}):
                fac2 = review.find('div', {'class':re.compile("GradeSquare__ coloredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
                facilities.append(float(fac2))
```

```
            # If the background is red
            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
                fac3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
                facilities.append(float(fac3))

            # Elsewhere, though this does not often happen
            else:
                facilities.append(float(0.0))

        # The same applies for the next 9 categories
        if category == 'Reputation':
            if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
                rep1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
                reputation.append(float(rep1))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
                rep2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
                reputation.append(float(rep2))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
                rep3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
                reputation.append(float(rep3))
```

```
            else:
                reputation.appen(float(0.0))

        if category == 'Opportunities':
            if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
                opp1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
                opportunities.append(float(opp1))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
                opp2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
                opportunities.append(float(opp2))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
                opp3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
                opportunities.append(float(opp3))

            else:
                opportunities.append(float(0.0))

        if category == 'Happiness':
            if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
                hap1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
                happiness.append(float(hap1))
```

```
        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
            hap2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            happiness.append(float(hap2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
            hap3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            happiness.append(float(hap3))

        else:
            happiness.append(float(0.0))

    if category == 'Location':
        if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
            loc1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
            location.append(float(loc1))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
            loc2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            location.append(float(loc2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
```

```
            loc3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            location.append(float(loc3))

        else:
            location.append(float(0.0))

    if category == 'Safety':
        if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
            saf1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
            safety.append(float(saf1))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
            saf2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            safety.append(float(saf2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
            saf3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            safety.append(float(saf3))

        else:
            safety.append(float(0.0))

    if category == 'Clubs':
```

```
        if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
            clu1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
            clubs.append(float(clu1))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
            clu2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            clubs.append(float(clu2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
            clu3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            clubs.append(float(clu3))

        else:
            clubs.append(float(0.0))

    if category == 'Social':
        if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
            soc1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
            social.append(float(soc1))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
```

```
            soc2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            social.append(float(soc2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
            soc3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            social.append(float(soc3))

        else:
            social.append(float(0.0))

    if category == 'Internet':
        if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
            int1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
            internet.append(float(int1))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
            int2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
            internet.append(float(int2))

        elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
            int3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
            internet.append(float(int3))
```

```
            else:
                internet.append(float(0.0))

        if category == 'Food':
            if review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
fGdpBh")}):
                foo1 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 fGdpBh")}).text.strip()
                food.append(float(foo1))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
bwfltG")}):
                foo2 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 bwfltG")}).text.strip()
                food.append(float(foo2))

            elif review.find('div', {'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0
jklQNe")}):
                foo3 = review.find('div',
{'class':re.compile("GradeSquare__ColoredSquare-sc-6d97x2-0 jklQNe")}).text.strip()
                food.append(float(foo3))

            else:
                food.append(float(0.0))

    #Getting the Overall score

    page_source = browser.page_source
    chicken_soup = BeautifulSoup(page_source, 'lxml')
```

```
    overall_score = chicken_soup.find('div', class_ = 'OverallRating__Number-y66epv-3
dXoyqn').text.strip()

    # Make it a float from a string
    overall_score = float(overall_score)

    # Adds the overall rating to the overall rating list
    overallratings.append(overall_score)

    # Getting the school name from the website
    school_name = chicken_soup.find('div', class_ =
'HeaderDescription__StyledTitleName-sc-1lt205f-1 eNxccF').text.strip()

    # Adding it to the list
    schoolname.append(school_name)

    # Repeats for all 126 schools

# Running this code took around 30 mins in my apartment

# Each of the lists are being put into a dictionary with keys so they can be converted into a
dataframe
school_data_dict = {"School Name": schoolname, "Overall Rating": overallratings, "Facilities":
facilities, "Reputation": reputation, "Opportunities": opportunities, "Happiness": happiness,
"Location": location, "Safety": safety, "Clubs": clubs, "Social": social, "Internet": internet,
"Food": food}

# The creation of said dataframe
school_data_dataframe = pd.DataFrame(school_data_dict)

# Moving the dataframe into a .csv so that this does not need to be run again
```

```
school_data_dataframe.to_csv('School Info Data.csv')
```

 # Running the .csv file (new workbook); beginning of user experience

```
# Import necessary functions
import time
import re
import pandas as pd
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC

# Define function to help us click to next page
def check_exists_by_xpath(xpath):
    try:
        browser.find_element(By.XPATH,xpath)
    except NoSuchElementException:
        return False
    return True

# Dictionary of the universities which will become a part of the .csv file
# There is the school name correlated with the corresponding integer which applies to the rate
my professor website url code
# 126 schools
```

school_number_dict = {"Baruch College": 222, "Fordham University": 1325, "City College of New York": 224,

"Hunter College":226, "Queens College of New York": 231, "The New School": 1519,

"Brooklyn College": 223, "Yeshiva University": 1223, "Fashion Institute of Techonology": 975,

"John Jay": 227, "Montclair State University": 630, "New York City College of Technology": 230,

"Drew Univeristy": 1314, "Manhattan College": 557, "SUNY Maritime": 976, "York College": 1224,

"Marymount University": 574, "Hofstra University": 413, "Lehman College": 228,

"Pace University": 12151, "Sarah Lawrence College": 883, "Vaughn College": 15545,

"College of Staten Island": 225, "Medgar Evers College": 229, "Stevens Institute of Technology": 982,

"Wagner College": 1129, "Purchase College": 970, "Kean Univeristy": 478, "St. Francis College": 3975,

"Molloy College": 623, "Plaza College": 2697, "Saint Elizabeth": 4160, "Monmouth University": 625,

"Saint John's Univeristy": 842, "Iona Univeristy": 451, "Saint Thomas Aquinas College": 4409,

"Mercy College": 592, "Monroe College": 2482, "Webb Institute": 1152, "Touro University": 1024,

"Metropolitan College of New York": 5585, "ASA College": 5069, "Manhattanville College": 558,

"New York University": 675, "Stony Brook University": 971, "Seton Hall Sniversity": 892,

"Adelphi University": 6, "Farmingdale State College": 14046, "Wells College": 1157,

"Niagara University": 678, "Columbia University": 278, "Hunter College": 226,

"Long Island University": 527, "SUNY Purchase": 970, "Rutgers": 4939, "New Jersey Institute of Technology": 668,

"Vassar College": 4070, "Fairfield University": 1351, "Seton Hall": 892, "Cornell University": 298,

"Colgate University": 252, "Hamilton College": 389, "University of Rochester": 1331, "Skidmore College": 907,

"Union College": 1044, "Binghamton University": 958, "Syracuse University": 992, "University at Buffalo": 960,

"St. Bonaventure University": 832, "Rochester Institute of Technology": 807, "St. Lawrence University": 847,

"Clarkson University": 240, "D'Youville College": 4638, "St. John Fisher University": 1628, "SUNY New Paltz": 964,

"Marist College": 563, "Le Moyne College": 506, "SUNY Oneonta": 966, "University at Albany": 957,

"Nazareth College": 661, "Ithaca College": 453, "Utica University": 4137, "SUNY Oswego": 967, "SUNY Plattsburgh": 968,

"SUNY Cortland": 961, "SUNY Maritime College": 976, "Canisius College": 175, "Hilbert College": 4499,

"SUNY Brockport": 1549, "Daemen College": 4135, "Siena College": 900, "Boricua College ": 4775,

"SUNY Delhi": 4414, "SUNY Geneseo": 963, "Elmira College": 332, "Hartwick College": 398, "York College": 1224,

"Yale University": 1222, "Wesleyan University": 1161, "Trinity College": 1030, "University of Connecticut": 1091,

"Quinnipiac University": 787, "Sacred Heart University": 827, "University of New Haven": 4159,

"Albertus Magnus College": 16, "University of Hartford": 1103, "University of Bridgeport": 1071,

"Mitchell College": 4485, "Goodwin University": 5260, "The College of Saint Rose": 14032,

"Paier College of Art": 5731, "Princeton University": 780, "The College of New Jersey": 256, "Rowan University": 822,

"Stockton University": 800, "Rider University": 801, "Ramapo College of New Jersey": 13744, "Caldwell University": 144,

"New Jersey City University": 4527, "Centenary University": 1627, "Cazenovia College": 190,
"Felician University": 4507, "Saint Peter's University": 864, "Bloomfield College": 4478,
"Eastwick College": 5574, "Berkeley College": 1723, "Pillar College": 4300, "Thomas Edison State University": 1016}

```
# Moving the .csv file with the dataframe scraped previously into a variable
data_frame_name = 'School Info Data.csv'

# Putting the .csv file into a new dataframe with the indication that the first column is an index column
school_data_dataframe = pd.read_csv(data_frame_name, index_col = 0)



# A list of the ten features to reference back to and check for existence
feature_list = ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location',
                'Safety', 'Clubs', 'Social', 'Internet', 'Food']

# The user is asked whether they want to create an overall rating based on their categorical preferences
weight_ask = input('Would you like to have a preferred features for your college recommendation?\n>>')
# Making it lowercase to save an if statement
weight_ask = weight_ask.lower()
# for the while statement to come
while_variable = 0

# Should the user want to make their weighted ranking, assuming they entered 'yes', the following if statement would run
if weight_ask == 'yes':
    # The user will select 5 different features
```

```
    print('Please enter your top 5 preferences, ranked most important to least important')

    while while_variable == 0:
        # Presents options and then asks the user to enter from the list
        print('Here are the feature options:\n', feature_list)
        weight_one = input('\nWhat is your most important feature?\n>>')
        # Checking if the input is in the feature list
        if weight_one in feature_list:
            # Telling it to break the while loop if so
            while_variable += 1

        # If not the user will be told and be forced to try again until successful
        else:
            print('\n\n\n\nYour choice was not in the feature list.\nPlease try again, copied exactly
from the list\n')

    # New while loop, new while_variable
    while_variable = 0

    # Second feature to choose from
    while while_variable == 0:
        # Options presented once again
        print('Here are the feature options:\n', feature_list)
        # the entering of the second choice
        weight_two = input('\nWhat is your second most important feature?\n>>')
        # Making sure a different feature is chosen from the first one
        if weight_one == weight_two:
            print('Please select a different feature')
        # breaking of the loop should that be done
        elif weight_two in feature_list:
            while_variable += 1
```

```
        # Once again, the user will be forced to enter until they choose a new and correctly typed feature
        else:
            print('\n\n\n\nYour choice was not in the feature list.\nPlease try again, copied exactly from the list\n')

    # third while_variable
    while_variable = 0

    # Third feature
    while while_variable == 0:
        # again...
        print('Here are the feature options:\n', feature_list)
        weight_three = input('\nWhat is your third feature?\n>>')

        # More opportunity for same features so more if statements
        if weight_one == weight_three:
            print('\n\n\n\nPlease select a different feature')
        elif weight_three == weight_two:
            print('\n\n\n\nPlease select a different feature')
        # Break the loop
        elif weight_three in feature_list:
            while_variable += 1
        # or don't
        else:
            print('\n\n\n\nYour choice was not in the feature list.\nPlease try again, copied exactly from the list\n')

    # fourth while_variable
    while_variable = 0
```

```
    # Fourth feature
    while while_variable == 0:
        # Options presented
        print('Here are the feature options:\n', feature_list)
        weight_four = input('\nWhat is your fourth feature?\n>>')

        # Even more opportunity for failure
        if weight_one == weight_four:
            print('\n\n\n\nPlease select a different feature')
        elif weight_three == weight_four:
            print('\n\n\n\nPlease select a different feature')
        elif weight_two == weight_four:
            print('\n\n\n\nPlease select a different feature')

        # Be gone loop
        elif weight_four in feature_list:
            while_variable += 1
        # Remain loop
        else:
            print('\n\n\n\nYour choice was not in the feature list.\nPlease try again, copied exactly
from the list\n')

    # fifth while_variable
    while_variable = 0

    # Final feature
    while while_variable == 0:

        # Final selection
        print('Here are the feature options:\n', feature_list)
        weight_five = input('\nWhat is your last feature?\n>>')
```

```
        # Final check for repeats
        if weight_one == weight_five:
            print('\n\n\n\nPlease select a different feature')
        elif weight_three == weight_five:
            print('\n\n\n\nPlease select a different feature')
        elif weight_two == weight_five:
            print('\n\n\n\nPlease select a different feature')
        elif weight_four == weight_five:
            print('\n\n\n\nPlease select a different feature')

        # Final break of the loop
        elif weight_five in feature_list:
            while_variable += 1

        # Final chance for failure
        else:
            print('\n\n\n\nYour choice was not in the feature list.\nPlease try again, copied exactly from the list\n')

    # A list which will hold the weighted rankings based on the user's choices
    weighted_list = []

    # For loop which runs x times based on the number of schools in our list
    # This loop's purpose is to create a new column with a weighted overall rating for each school based on the user's preferences

    for school in range(0,len(school_data_dataframe)):

        # Feature list presented in the for loop so it can be changed and reappear for each school
        feature_list = ['Facilities', 'Reputation', 'Opportunities', 'Happiness', 'Location',
                'Safety', 'Clubs', 'Social', 'Internet', 'Food']
```

```
# The number one choice of feature given the highest weight of .23 or 23%
# It is grabbing the actual score from the dataframe and then applying its weight
one = (school_data_dataframe[weight_one][school]*0.23)
# Specified feature removed from the list
feature_list.remove(weight_one)

# Second choice feature, second highest weight: 20%
two = (school_data_dataframe[weight_two][school]*0.20)
# Specified feature removed from the list
feature_list.remove(weight_two)

# Third feature: 15%
three = (school_data_dataframe[weight_three][school]*0.15)
# Specified feature removed from the list
feature_list.remove(weight_three)

# Fourth: 10%
four = (school_data_dataframe[weight_four][school]*0.10)
# Specified feature removed from the list
feature_list.remove(weight_four)

# Fifth: 7%
five = (school_data_dataframe[weight_five][school]*0.07)
# Specified feature removed from the list
feature_list.remove(weight_five)

# Creating a variable to use
more = 0
```

```
        # Running the remaining five features which are left after the user selected five were
removed
        for item in feature_list:

            # Each remaining feature is given an equal weight of 5%
            more = more + (school_data_dataframe[item][school]*0.05)

        # Hence, the final weighted ranking of the school is created
        # 100% based on the user's choices
        total = one + two + three + four + five + more

        # The weighted score is added to the list for all schools
        weighted_list.append(total)

    # Adding this column to the dataframe
    school_data_dataframe['Weighted Overall Rating'] = weighted_list

    # Creating a dataframe which has the new rating and ranks the schools accordingly
    final_df = school_data_dataframe.sort_values(by=['Weighted Overall Rating'],
ascending=False)

    # Print statement for the following showing
    print('\n\n\n\nHere are your top three colleges')

# For those who did not want to add their own weights
else:
    # final_df is still created though weighted overall rating is not in it
    print('It\'s ok, the original list is fine')
    # Rank is by rate my professor overall rating instead
    final_df = school_data_dataframe.sort_values(by = ['Overall Rating'], ascending = False)
    print('\n\n\n\nHere are the top three colleges based on Rate My Professors overall rating')
```

```
# Top three colleges
top_three_df = final_df.head(3)

# Top three shown
top_three_df
```

 # The user will now select a school from the list and then get professors from rate my professor

```
# Reprise of while_variable
while_variable = 0

# The user must now select one of the schools appearing in their top three
while while_variable == 0:
    # User input of school
    school_choice = input('\nPlease enter the school of your choice\n>>')

    # If they spelled the school correctly
    if school_choice in school_number_dict:
        while_variable += 1

    # If not
    else:
        print('\n\n\n\nYour selection was not in the college list, please try again\nThe list is case sensitive')

# In order to create the url once again, now for the professors
school_id = school_number_dict[school_choice]
```

```
# Int to string
school_id = str(school_id)



######## This is foundation for scraping professor grades



# My chromedriver path
path = "C:\\Users\\benri\\Downloads\\chromedriver_win32 (4)\\chromedriver"
s=Service(path)
browser = webdriver.Chrome(service=s)

# Creates the url for the school using the school id of the selected university
link = "https://www.ratemyprofessors.com/search/teachers?query=*&sid=" + school_id
browser.get(link)

# Waits 20 seconds after the page has loaded so that all of the ads pop up
time.sleep(20)

# Close out the box talking about cookies
try:
    if (check_exists_by_xpath('//img[@class="FullPageModal__StyledCloseIcon-sc-1tziext-0
eJtQsN"]')):

browser.find_element(By.XPATH,'//img[@class="FullPageModal__StyledCloseIcon-sc-1tziext-
0 eJtQsN"]').click()
        time.sleep(1)
except:
    print("could not click")
    pass
```

```
# Skip ad popup if present
try:
    if (check_exists_by_xpath('//a[@class="bx-close bx-close-link bx-close-inside"]')):
        browser.find_element(By.XPATH,'//a[@class="bx-close bx-close-link
bx-close-inside"]').click()
        time.sleep(1)
except:
    print("could not click")
    pass

# Skip ad popup if present
try:
    if (check_exists_by_xpath('//div[@class="IL_BASE IL_SR_BG"]')):
        browser.find_element(By.XPATH,'//div[@class="IL_BASE IL_SR_BG"]').click()
        time.sleep(1)
except:
    print("could not click")
    pass

# Empty list of the characteristics of the professor
names = []
ratings = []
subject = []
school = []
number_of_rating = []


# use selenium to go to the next page
# Max at 500 pages
for item in range(0,500):
```

```
    if (check_exists_by_xpath('//button[@class="Buttons__Button-sc-19xdot-1
PaginationButton__StyledPaginationButton-txi1dr-1 gjQZal"]')):
        browser.find_element(By.XPATH,'//button[@class="Buttons__Button-sc-19xdot-1
PaginationButton__StyledPaginationButton-txi1dr-1 gjQZal"]').click()
        time.sleep(1)
    else: break

    # parse to a soup
page_source = browser.page_source
# Grabbing the necessary html
soup = BeautifulSoup(page_source, 'lxml')
reviews_content = soup.find_all('a', class_='TeacherCard__StyledTeacherCard-syjs0d-0 dLJIlx')

# For each group of html containing each professor
for review in reviews_content:

    # Grab the name of the professor
    name = review.find('div', class_='CardName__StyledCardName-sc-1gyrgim-0 cJdVEK').text

    # When grabbing the rating, each class is associated to its color background based on its score
    # This is the same as the scores in the ratings of the schools
    # Hence, there are if statements for each situation
    if
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
gcFhmN")}):
        rating1 =
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
gcFhmN")}).text.strip()
        # Adding each one to the list and making them floats
        ratings.append(float(rating1))
```

```
    elif
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
icXUyq")}):
        rating2 =
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
icXUyq")}).text.strip()
        ratings.append(float(rating2))

    elif
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
bUneqk")}):
        rating3 =
review.find('div',{'class':re.compile("CardNumRating__CardNumRatingNumber-sc-17t4b9u-2
bUneqk")}).text.strip()
        ratings.append(float(rating3))

    # in the case that the rating does not exist
    else:
        ratings.append(float(0.0))

    # making a list of html containing the number of ratings which the professor has received
    number_of_ratings = review.find("div",
{"class":re.compile("CardNumRating__CardNumRatingCount-sc-17t4b9u-3 jMRwbg")})

    # Grabbing the number of ratings which the professor has received
    if number_of_ratings:
        number_of_ratings = number_of_ratings.text.strip()

    # Grabbing the subject taught of the professor
    subjects = review.find('div', class_='CardSchool__Department-sc-19lmz2k-0 haUIRO').text
```

```
    # append to our accumulative lists
    names.append(name)
    number_of_rating.append(number_of_ratings)
    subject.append(subjects)

# In the case that there is no rating
new_ratings = []
for none in ratings:
    # So that there is always a float to work with
    if none is None:
        none = float(0.0)
        new_ratings.append(none)
    else:
        new_ratings.append(float(none))

# Dictionary of the professors stats
data = {"Name": names, "Rating": new_ratings, "Number of Ratings": number_of_rating,
"Subject": subject}
Rate_My_Professor_Dict = {"Name": names, "Rating": new_ratings, "Number of Ratings":
number_of_rating, "Subject": subject}

# Creation of the professors dataframe
Rate_My_Professor_Dataframe = pd.DataFrame(data)

# Creating a list of possible majors for the user to choose from
major_list = ['Business', 'Science', 'Engineering', 'Communications', 'Humanities',
'Religious_Studies', 'Arts', 'Education', 'Political_Science_and_Law']

# Creating classes which professors could teach based on their respective major
Business = ['Finance', 'Marketing', 'Accounting', 'Business', 'Administration', 'Economics',
'Management', 'Statistics']
```

```
Science = ['Biology', 'Chemistry', 'Science', 'Agriculture', 'Mathematics', 'Physics', 'Psychology', 'Statistics']
Engineering = ['Architecture', 'Computer Science', 'Engineering', 'Information Science', 'Mathematics']
Communications = ['Communication', 'Hospitality', 'Journalism', 'Writing']
Humanities = ['English', 'Humanities', 'Languages', 'Classical Studies', 'Classics', 'Ethnic Studies', 'Literature', 'Modern Languages', 'Philosophy', 'Woman\'s Studies', 'Writing']

# Some classes apply to multiple majors
Religious_Studies = ['Religion', 'Religious Studies', 'Theology']
Arts = ['Art History', 'Design', 'Film', 'Fine Arts', 'Graphic Arts', 'Music', 'Theater']
Education = ['Education']
Political_Science_and_Law = ['Anthropology', 'Criminal Justice', 'Ethnic Studies', 'History', 'International Studies', 'Latin American Studies', 'Law', 'Political Science', 'Social Science', 'Social Work', 'Sociology']

# A list of lists in the same order as the list of majors so that they have the same indexes
major_list_list = [Business, Science, Engineering, Communications, Humanities, Religious_Studies, Arts, Education, Political_Science_and_Law]

# while_variable back again
while_variable = 0

while while_variable == 0:

    # Showing and asking the user to input a major
    print(major_list)
    major = input('Please select your expected major\n>>')

    # Checking if a major was selected
    if major in major_list:
```

```
        # Grabbing the corresponding list
        major_index = major_list.index(major)
        major_choice = major_list_list[major_index]
        # Breaking the while loop
        while_variable += 1
    # Telling the user to type correctly
    else:
        print('\n\n\n\nPlease try again, the majors are case sensitive')

# New dataframe with professors greater than a certain rating within the selected major field
New_df = Rate_My_Professor_Dataframe.loc[(Rate_My_Professor_Dataframe['Rating'] >= 4.5)
&
          Rate_My_Professor_Dataframe['Subject'].isin(major_choice)]

# In the order desired
New_df = New_df.sort_values(by = ['Rating'], ascending = False)

# Presentation of professors
New_df
```

 # Extracting tweets from Twitter API for Sentiment Analysis:

```
# importing necessary libraries and modules
import pandas as pd
import json
from pandas import json_normalize

# installing and importing tweepy to extract tweets using twitter api
pip install tweepy
import tweepy
```

```
# authenticating with user credentials
consumer_key    = "w2JwpEsHNrmKzygM9oJiy2Fax"
consumer_secret = "oZOwrqrQIegQCJCpF9nh0snB9JRyU0kGgPvNRqy3C7PFDkcRjO"
access_key      = "1578756663879897088-UUNTqnKpEKhNk8rWrdAOfmO377cQHk"
access_secret   = "GO0QmMNExlObdQm6Bw3NzvlM7gJSfrGuzF7m57KDLdzkx"

# calling the function to retrieve data from twitter api developer account
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth)

# script to view the reviews for the website based on a specific set of features
user = "ratemyprofessor"
limit = 1000
tweets = api.user_timeline(screen_name = user, count = limit, tweet_mode = 'extended')
columns = ['User','UID', 'Tweet_Text', 'Tweet_Time', 'Tweet_Source', 'Number_of_Retweets',
'Number_of_Likes']
data = []
# creating a for loop to iterate through all the extracted tweets and store it in the list
for tweet in tweets:
    data.append([tweet.user,tweet.id, tweet.full_text, tweet.created_at, tweet.source,
tweet.retweet_count, tweet.favorite_count])
df = pd.DataFrame(data,columns = columns)
df

# script to display tweet reviews as a csv file for data mining analysis
user = "ratemyprofessor"
limit = 100
tweets = api.user_timeline(screen_name = user, count = limit, tweet_mode = 'extended')
```

```
columns = ['User','UID', 'Tweet_Text', 'Tweet_Time', 'Tweet_Source', 'Number_of_Retweets',
'Number_of_Likes']
data = []
for tweet in tweets:
    data.append([tweet.user,tweet.id, tweet.full_text, tweet.created_at, tweet.source,
tweet.retweet_count, tweet.favorite_count])
# converting dataframe to csv format
df = pd.DataFrame(data,columns = columns)
df.to_csv('file.csv')



# getting above mentioned features for user tweets and displaying the result in a dataframe
user = "ratemyprofessor"
limit = 100
tweets = api.user_timeline(screen_name = user, count = limit, tweet_mode = 'extended')



# appending the above attributes as the values of a dictionary for each tweet to a list called
"tweet_list"
tweet_list1 =
[{"UID":tweet.id,"Tweet_Text":tweet.full_text,"Tweet_Time":tweet.created_at,"Tweet_Source":t
weet.source,"Number_of_Retweets":tweet.retweet_count,"Number_of_Likes":tweet.favorite_co
unt} for tweet in tweets]
tweet_list1

# importing necessary libraries and modules
import re
import tweepy
from tweepy import OAuthHandler
!pip install textblob
import textblob
```

```
from textblob import TextBlob

# installing vader sentiment for sentiment analysis of tweets
!pip install vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# script to store reviews as a dataframe
user = "ratemyprofessor"
limit = 1000
tweets = api.user_timeline(screen_name = user, count = limit, tweet_mode = 'extended')
columns = ['Tweet']
# creating an empty list to store reviews
data = []
for tweet in tweets:
    data.append([tweet.full_text])
ratings = pd.DataFrame(data,columns = columns)
ratings



# define a function to calculate the sentiment score for each tweet
def sentiment_analyzer_scores(data):
    analyzer = SentimentIntensityAnalyzer()
    score = analyzer.polarity_scores(data)
    return score

# apply the function to create the score column which will have the sentiment scores
ratings['score'] = ratings['Tweet'].apply(lambda x: sentiment_analyzer_scores(x))
ratings['score'][0]
# storing the sentiment score for the tweets in the list
ratings
a = ratings['score']
```

```
for item in range(0, 176):
    b = a[item]['compound']
    print(b)
compound_list = []

# using a for loop to iterate over all the tweets
for item in range(0, 176):
    b = a[item]['compound']
    b = float(b)
    compound_list.append(b)

print(compound_list)

# determining the sentiment for a review based on the score values obtained
lean = []

# iterating over the for loop to categorize sentiment scores into positive, negative and neutral
for item in compound_list:
    if item > 0:
        c = 'Positive'
    elif item == 0:
        c = 'Neutral'
    else:
        c = 'Negative'
    lean.append(c)

print(lean)

#  mapping sentiment score to the sentiment value
sentiment_dict = {'Lean':lean, 'Score':compound_list}
```

```
new = pd.DataFrame.from_dict(data)

# displaying the sentiment analysis in a dataframe
sentiment_data_frame = pd.DataFrame.from_dict(sentiment_dict)
sentiment_data_frame

# importing seaborn to plot histogram
import seaborn as sns

import matplotlib.pyplot as plt
%matplotlib inline

# creating the histogram for displaying the tweet sentiment analysis breakdown
plt.figure(figsize=(10, 8))
sns.set(style="darkgrid")
sns.countplot(x=sentiment_data_frame['Lean'], order=['Positive', 'Neutral',
'Negative']).set_title('Rate My Professor Tweets Sentiment Analysis', fontsize = 28)



# extracting 100 tweets from Rate My Professor's twitter account
user = "ratemyprofessor"
limit = 100
tweets = api.user_timeline(screen_name = user,count = limit)

# declaring an empty list to store tweets
data16 = []

# using a for loop to iterate over the extracted tweets and extend it to the list created
for tweet in tweets:
    data16.extend(tweet.text.split())
```

```
# using this to display the output list
print(tweets)



# finding 30 most common words used on Fordham twitter account
from collections import Counter
counts = Counter(data16)

# creating an empty list
data17 = []

# using a for loop to iterate over the common words and to append it to the list created
for i in counts.most_common(100):
    data17.append(i[0])

# using this to display the output list
print(data17)
# scraping information for followers of user
user = 'ratemyprofessor'
followers =
[[follower.id,follower.name,follower.screen_name,follower.location,follower.description,followe
r.url,follower.followers_count,follower.friends_count,follower.favourites_count] for follower in
api.get_followers(screen_name = user, count = 100)]

# storing scraped follower data in a dataframe and displaying it
columns = ['Follower_ID', 'Follower_Name', 'Follower_Screen_Name', 'Follower_Location',
'Follower_Description', 'Follower_URL', 'Follower_Count', 'Followee_Count',
'Favourites_Count']
df1 = pd.DataFrame(followers,columns = columns)
df1
```