JOBS RECOMMENDER SYSTEM

**Ayush Jain, Shweta Pathak**

{jain.ayu, pathak.sh} @husky.neu.edu

CSYE 7245, Spring 2019, Northeastern University

***Abstract*— In today’s competitive world, it is very difficult to find the right type of employment which compliments the skills that an individual possesses. Further, due to the amount of job-profiles present in the industry, it is difficult for an individual to understand which profile is most suitable for him. Similarly, companies face the challenge of selecting a candidate best suitable for a specific job from among the list of thousands of participating candidates. Thus, a need to understand the requirements of a job profile present in the company and the skills complimenting it is needed to make this job-search process easier for both the participating company as well as the candidates.**

**In this project, we develop a recommender system to advocate the job-profiles present in a company based on the requirements present in their description and a candidate’s skills mentioned in the resume. For this we have first scrapped the data from *GlassDoor*’s site to get the job postings. We chose few postings with titles such as ‘Data Analyst’, ‘Data Scientist’, ‘Business Intelligence Analyst’ and ‘Data Engineer’ for the states of ‘Massachusetts’, ‘Texas’ and ‘California’. The technology of Selenium with Python was utilized for performing this data scrapping. The next step involved parsing a resume to get the skills that a candidate possesses. For this we parsed sample resumes using python and were able to get the name of the candidates and their respective skill-set. The final step of this project involved curation of the Recommender System in which we used *Cosine Similarity* to find the similarity between the candidate skill-set and the company’s required skills for a specific job-posting.**

This paper contains an introduction to the problem that is faced and the development of our jobs-recommender system as its solution.

**Keywords** Selenium. Scrapping. Parsing. Python. Cosine Similarity. Recommender System.

# **INTRODUCTION**

In today’s competitive world, it is a task for individuals to find jobs which match their skill-set. With the advancement in the technology, there has been hike in the job profiles and the posting platforms. Further, there are too many jobs available with different job-description names and it becomes confusing to a candidate as to which profile matches him perfectly. The company on the other hand receives about 1000+ resumes per

day and it is impossible to go through each of them and find a candidate whose skill-sets match the company’s requirements.

The Internet-based recruiting platforms become a primary recruitment channel in most companies. While such platforms decrease the recruitment time and advertisement cost, they suffer from an inappropriateness of traditional information retrieval techniques like the Boolean search methods.[1]

The main motto behind the implementation of a Recommender System for jobs is to make this process easier by suggesting the correct job-profiles to apply to the candidates. This can be suggested using variety of criterion such as *Salary, Company* and *Data-Skills*. This system aims to ease the process of job applications and providing efficiency to the process. This will enable the matching of a candidate’s skills and personal interests better with the company’s requirements for a specific job-profile. Further the confusion with respect to different job-names having similar skill-requirements will also be solved since the recommender system will recommend based on skills and not just the job-names.

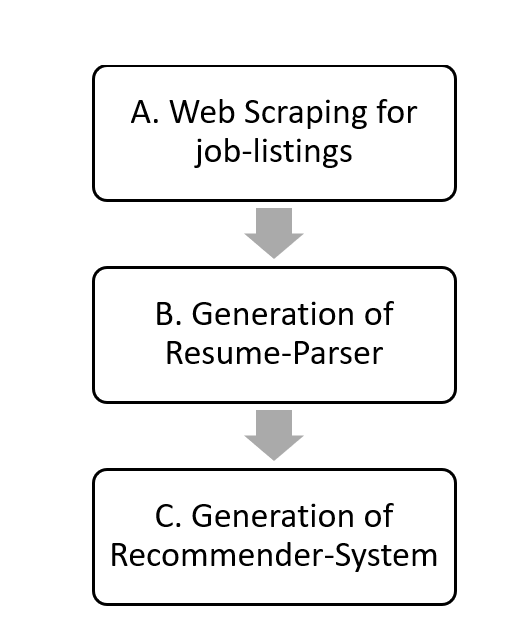
The overall idea of developing this Recommender System for Jobs is to help curb the confusion that exists due to the presence of two different job-names with same requirements, enhance the application system for both the candidate and the company. Consequently, a vast number of candidates miss the opportunity of applying to certain job-profiles due to several reasons ranging from different job-names, not enough information on the required skills or different job-board postings. The Recommender System will help the candidates in this case by providing them with all the job-profiles of various companies which match their personal interests such as Salary requirements, Company name and most importantly their skill-set.

# **METHODOLOGY**

We divided the project into 3 main tasks:

1. Get the Job Listings from a Job-Board
2. Get the Details from a candidate’s Resume
3. Generate a Recommender System for Jobs

These tasks define the working for the generation of the Recommender System. For this project, we curated 3 python notebooks namely, ‘WebScraping\_GlassDoor’, ‘Resume\_Parser’ and ‘Recommender System’. Each of these notebooks contain the above mentioned tasks.

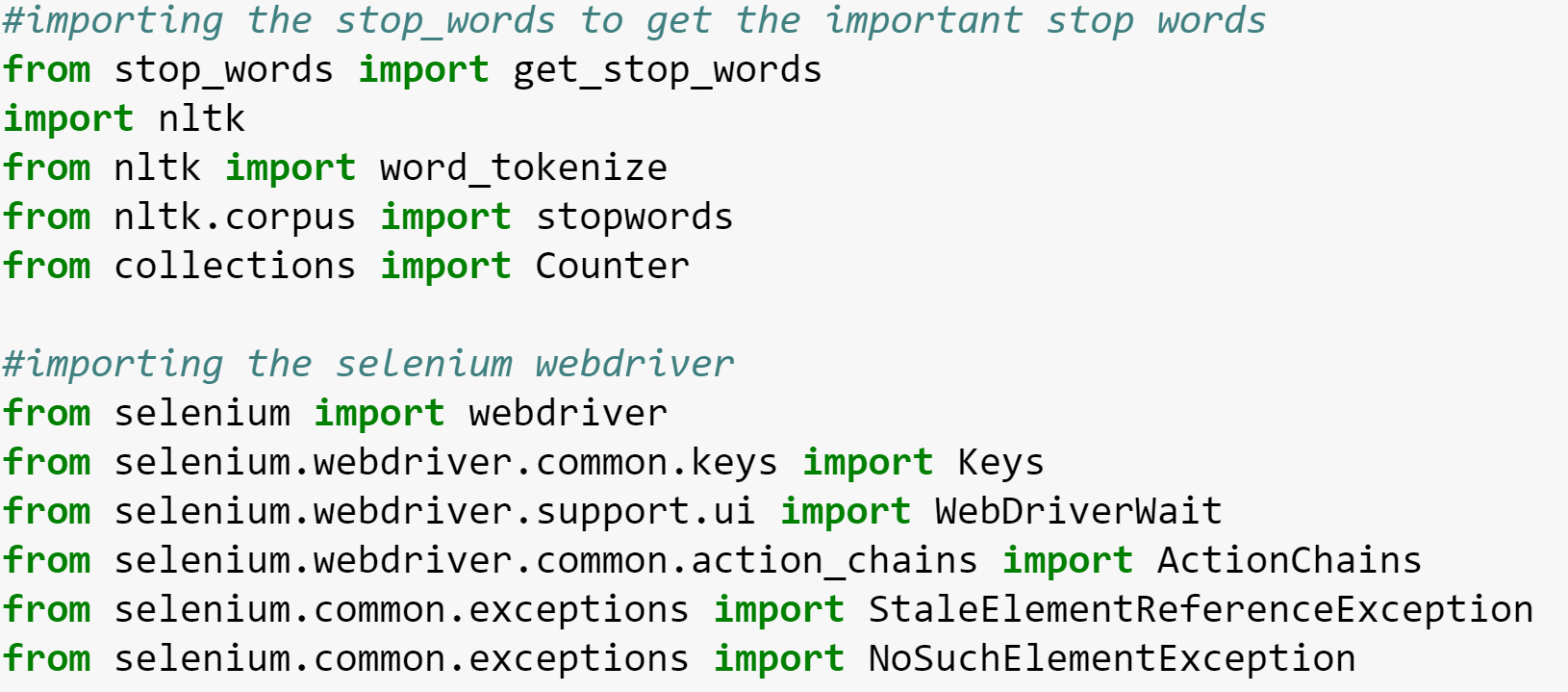


***Fig 1: Work-Flow of the Project***

## **WebScraping\_GlassDoor**

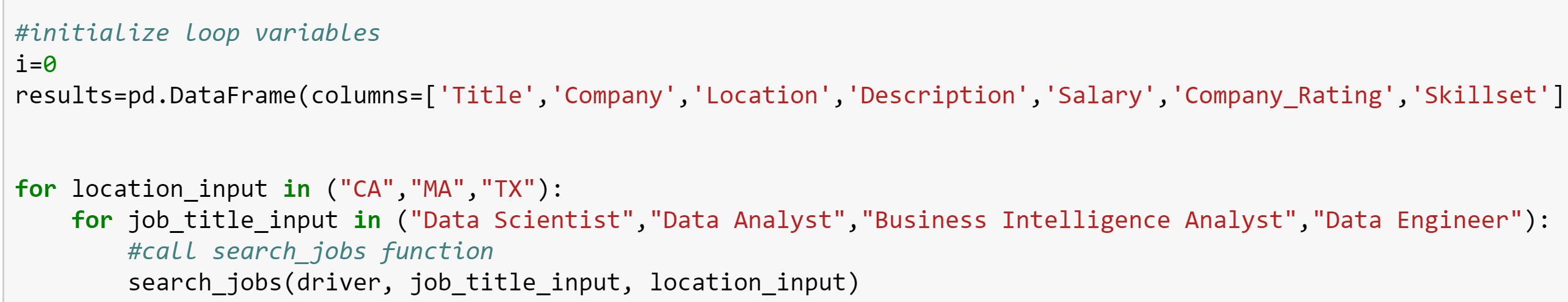
Web Scraping (also termed Screen Scraping, Web Data Extraction, Web Harvesting etc.) is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format.[2]

Web Scraping can be done in python using Beautiful Soup. However, this module does not work for some additions of the websites which are written using JavaScript. Because of this, we used Selenium in combination with Python. The Selenium package is used to automate web browser interaction from Python. With Selenium, programming a Python script to automate a web browser is possible. Afterwards, those pesky JavaScript links are no longer an issue.[3] Hence we imported various drivers for working with Selenium and a web-browser.



***Fig 2: Importing Web drivers and related content for Selenium***

The first step of our project involved getting the job-profiles for various companies using some job-board. For this purpose, we used GlassDoor as our job-board and performed web-scraping on it. This helped us to acquire various job-titles, company names, company-locations, company-glassdoor-ratings, job descriptions and company glassdoor salaries. We did this to get all those criterions from three different states, namely ‘MA’, ‘CA’ and ‘TX’.



***Fig 3: Features Scraped from GlassDoor using Selenium & python***

***Fig 4: Storing the scraped data from GlassDoor***

The next step involves tokenizing and finding the percentage of most common skills mentioned in the scraped job description. Tokenization is “the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.[4] In this process we also remove the stopwords to get more skills and with better frequency. Stop-words are those words in natural language which carry no own meaning and serve the purpose of connecting other words together to create grammatical sentences.[5] Examples of some Stopwords are: ‘and’, ‘is’, ‘it’ etc. This helps us focus on the right words which are skills in our case.

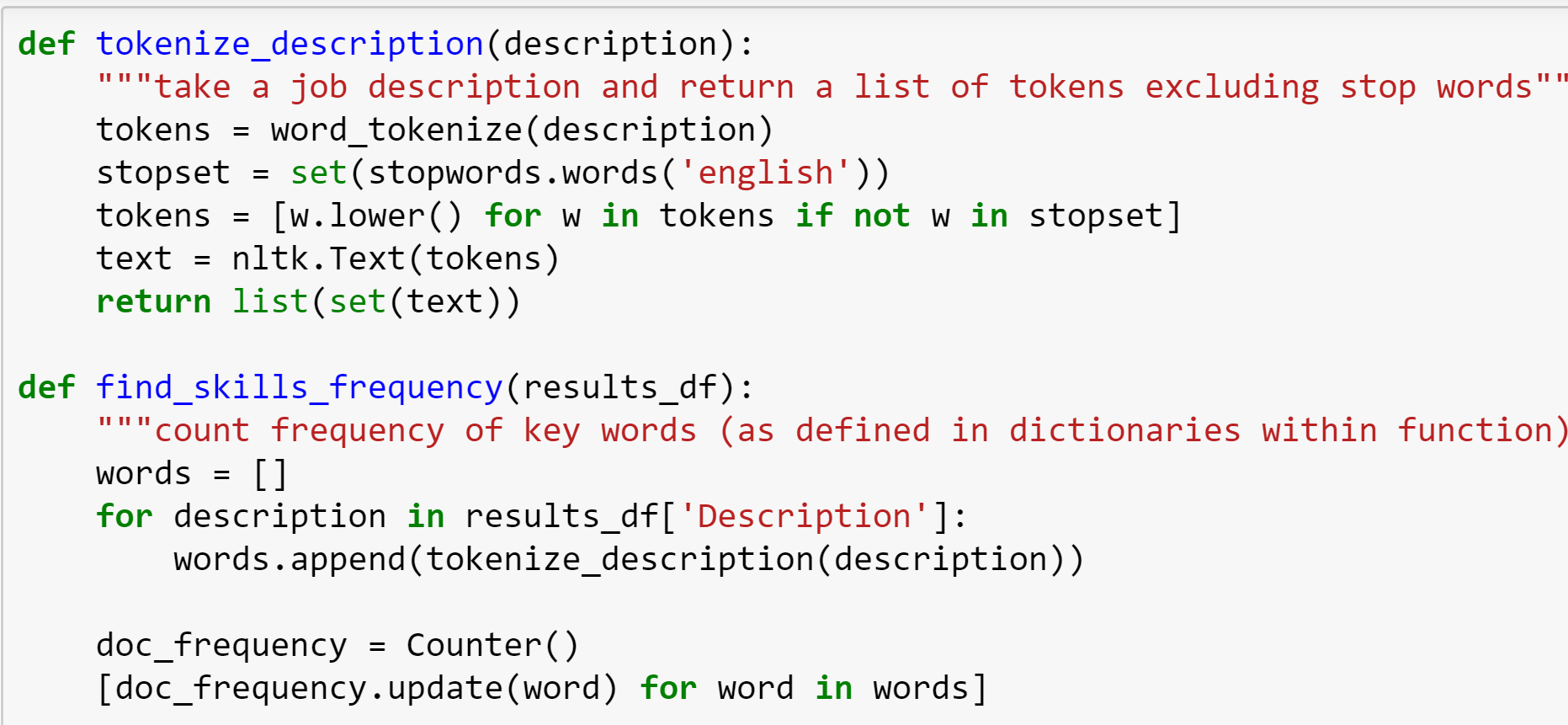
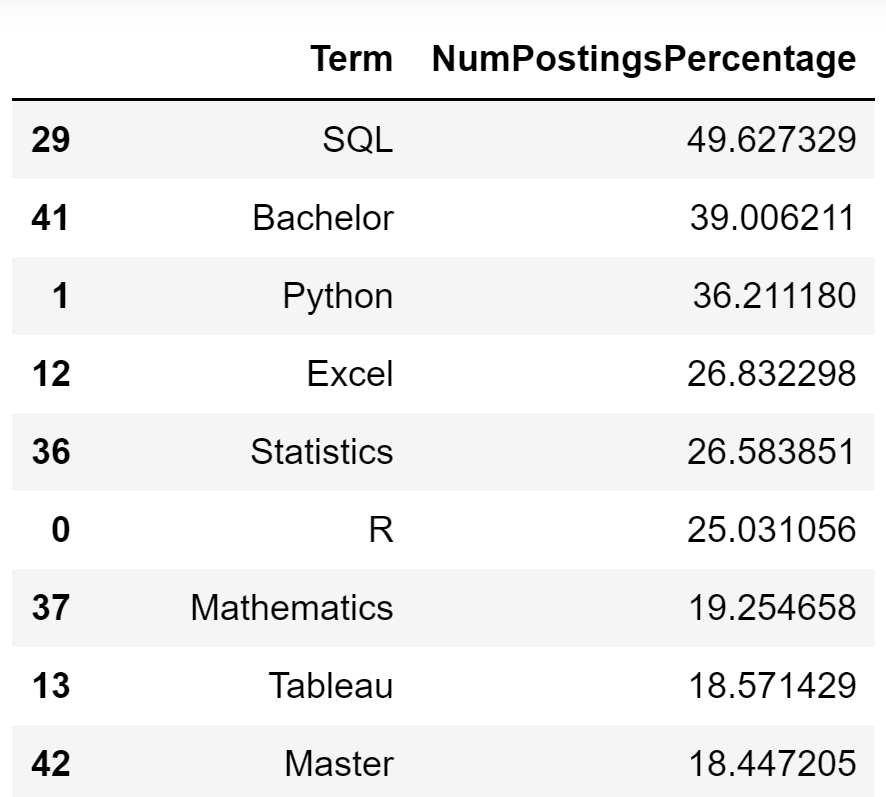
***Fig 5: Tokenizing the description***

Figure 5 displays the method that we created for tokenization which essentially splits the data into common blocks (skills) that is used later for comparison. We found the percentage of occurrence of each skill in the total number of scraped job listings for the purpose of comparison.



***Fig 6: Tokenized Skills and their percentage***

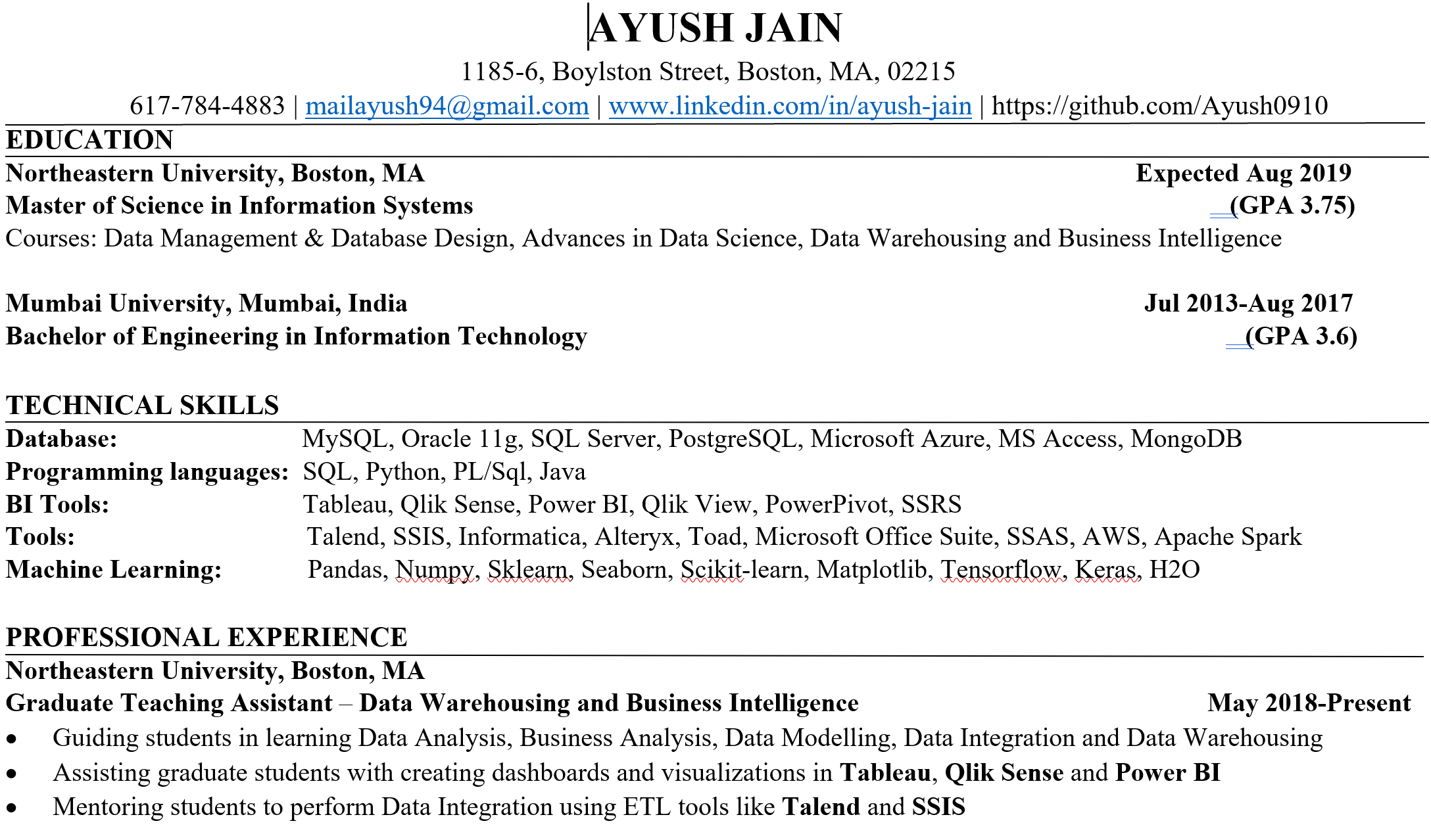
Finally, we found the skill-set present in each of the scraped job descriptions and saved these results in the form of output CSV.



***Fig 7: Output CSV data for Webscraping\_GlassDoor***

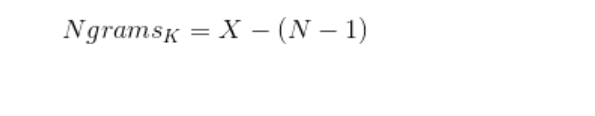
## **Resume\_Parser**

Our second notebook ‘Resume\_Parser’ deals with parsing the candidate’s resume to get his relevant details. Parsing is a method of extracting important and needed data from a document. We use it here for our project to generate a list of skills present in a candidate’s resume.

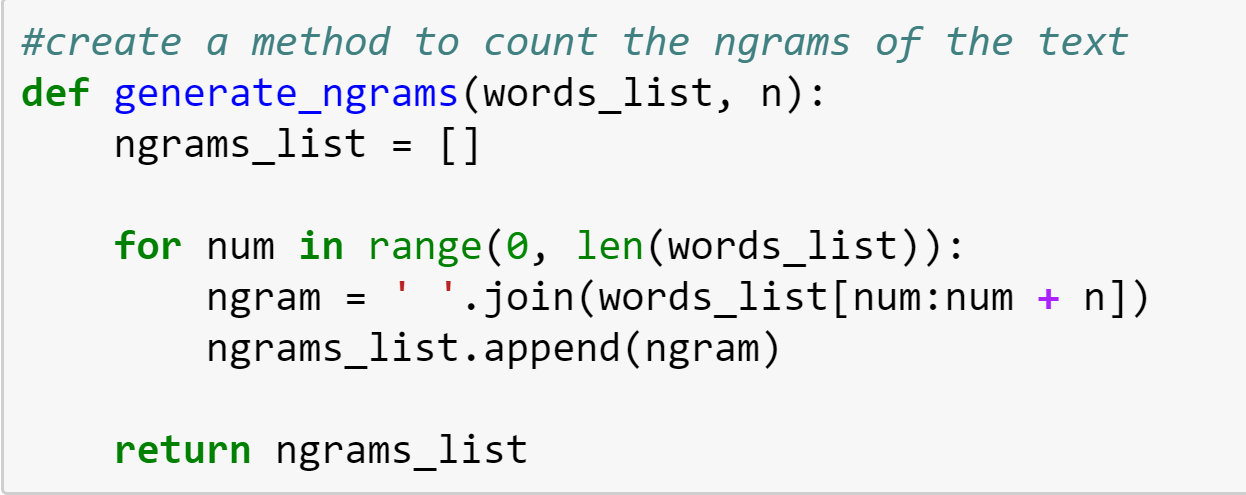
***Fig 8: A Part of Sample Resume***

Initially we define a method to calculate the ngrams of the text present in the resume. Ngrams are basically set of words which occur together, and N can be 1, 2 or any positive integer which can be used for various machine learning algorithms.

Assume a sentence ‘K’, to find Ngrams in that statement we can utilize the formula given below where ‘X’ is the number of words present in that statement.

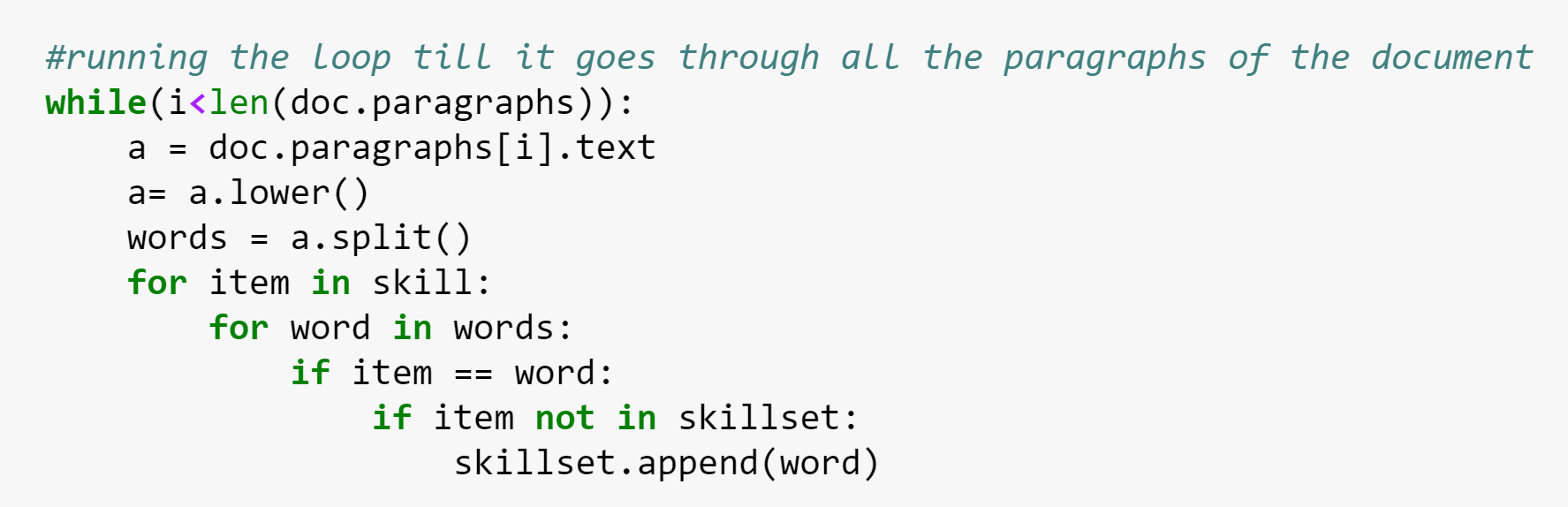


We used ngrams in our project to find important words present in a candidate’s resume. These wordings can be unigrams (N=1), bigrams (N=2), trigrams (N=3) or associated with any positive integer.

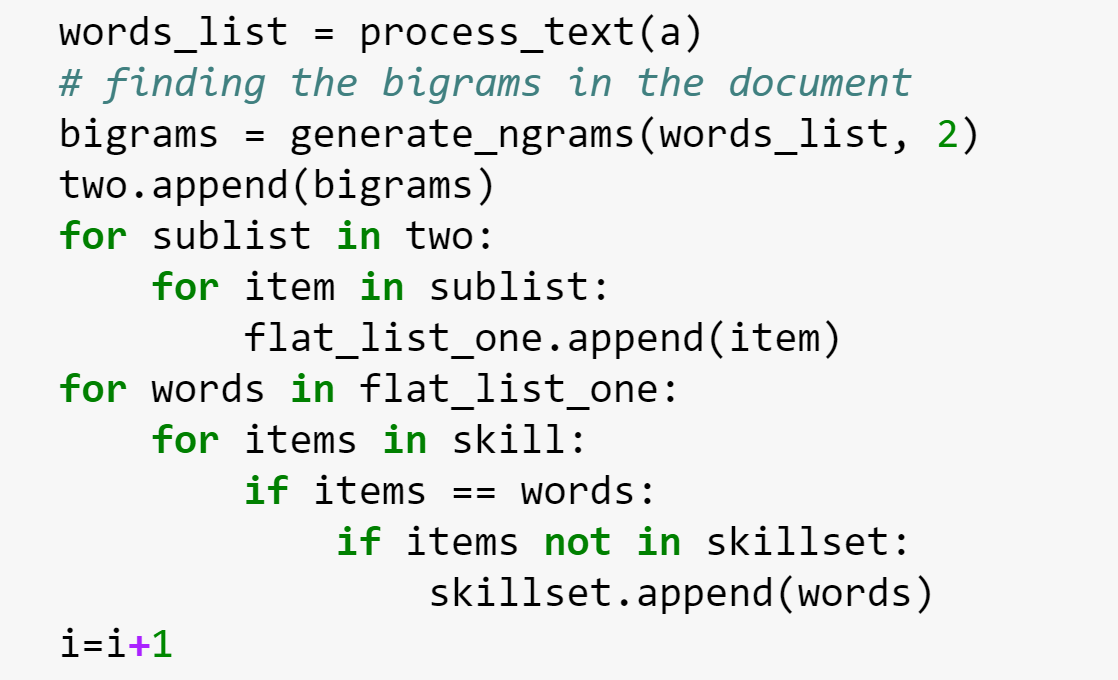


***Fig 9: Method to create Ngrams for the Resume-Parser***

Next, we defined a list of skills which we are trying to find in the candidate’s resume. This involves skills such as ‘Python’, ‘R’, ‘Data Mining’, ‘Tableau’ etc. After defining this list, we run the parser-loop to check if any of the skills mentioned in the above list is present in the candidate’s resume. If a skill is present, it would be appended to a ‘Skill-set’ list which we generated.

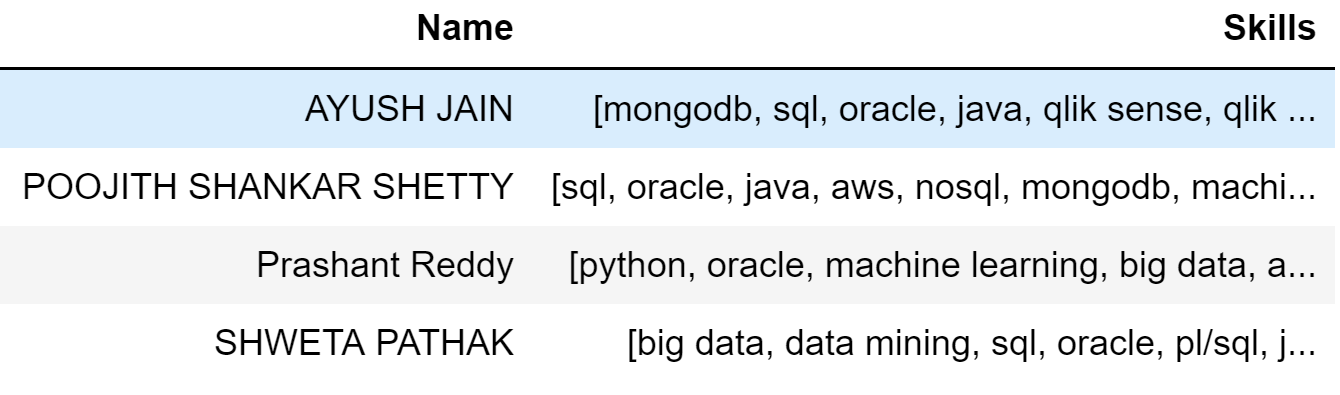
***Fig 10: Parser Code***

We also generated a bigram parser code to be able to capture words such as ‘Data-Mining’, ‘Machine-Learning’ etc.



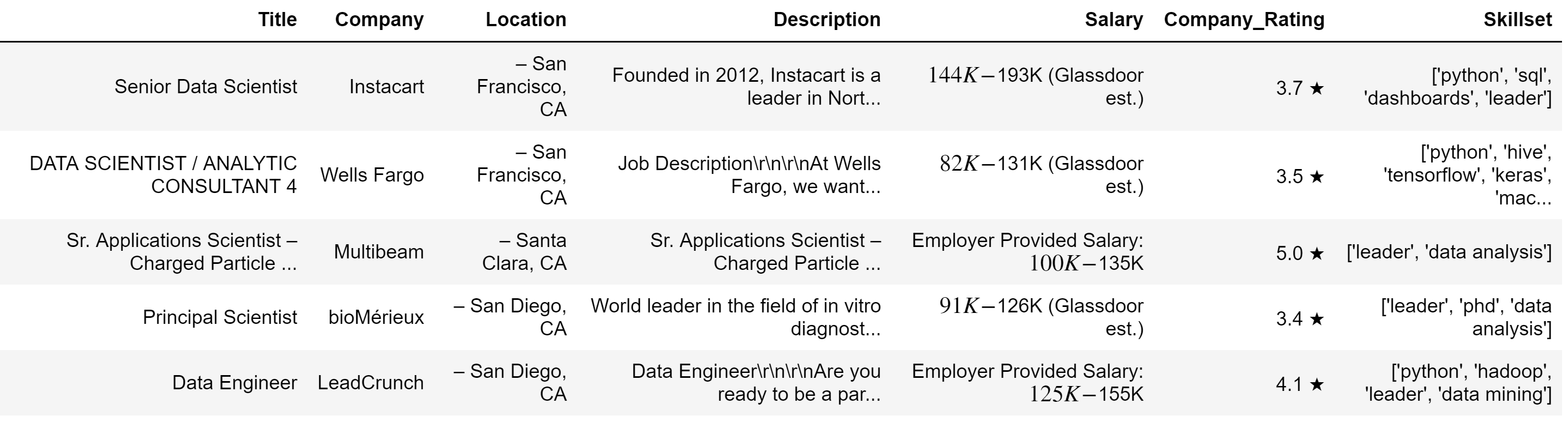
***Fig 11: Bigram Parser Code***

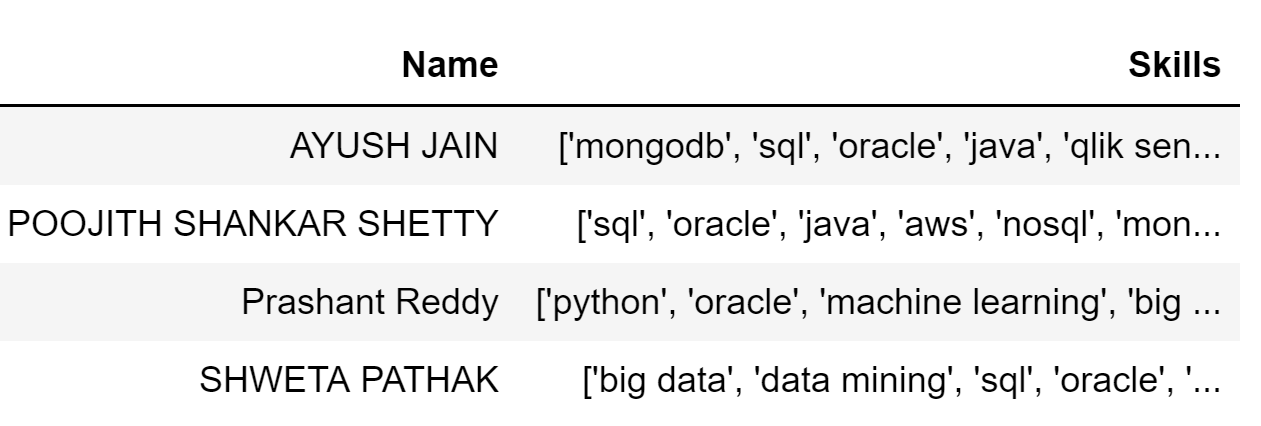
Finally, we extract the name and skills of a candidate using the above methods and parser code. We generate an output csv of the same.

***Fig 12: Output of the Resume\_Parser***

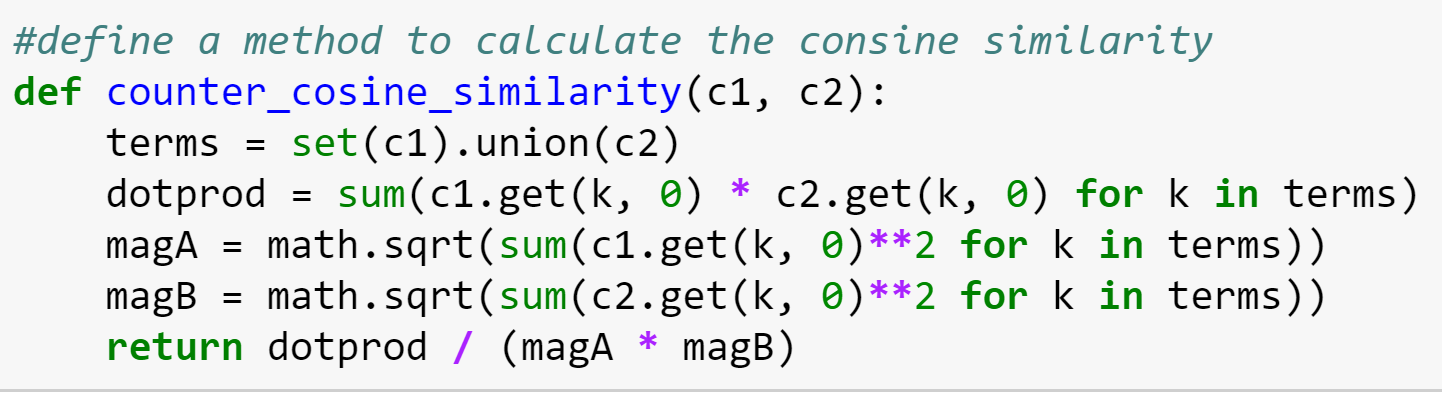
## **Recommender System**

A recommender system or recommendation system (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.[6] Our final python notebook is the curation of the Jobs Recommender System using the earlier mentioned two output CSV files.

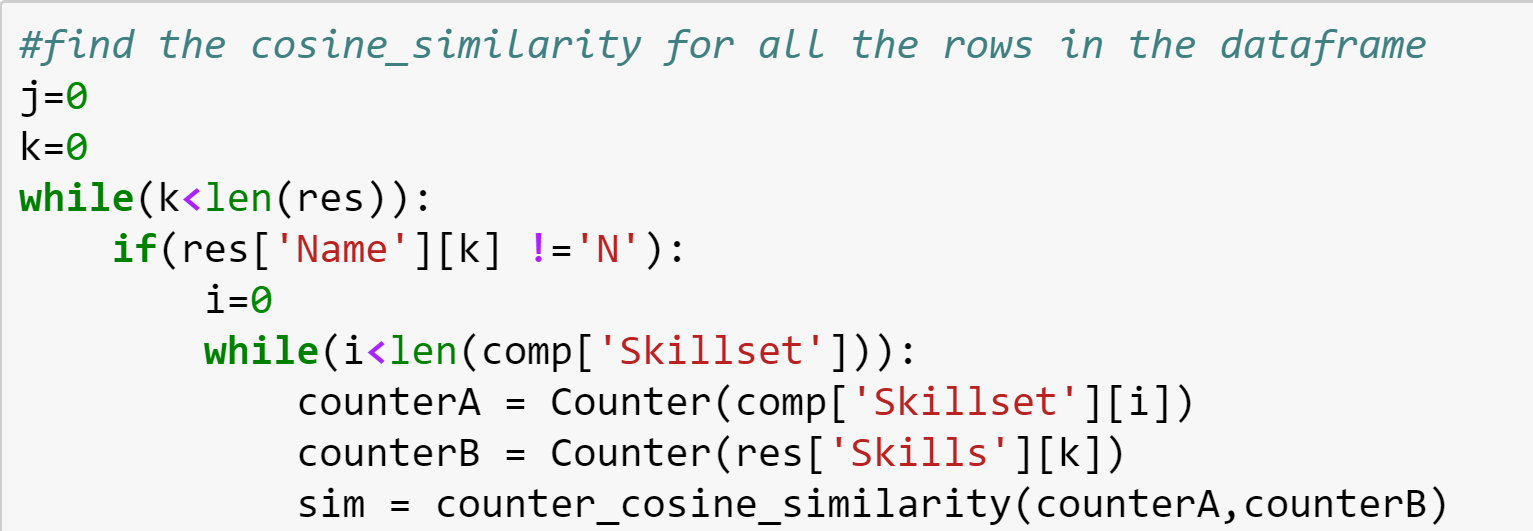
***Fig 13: Input Data of CSV 1 (Web Scraped Data)***

***Fig 14: Input Data of CSV 2 (Resume Parsed Data)***

In order to find the similarity between the skills from the job-postings and that of the candidate’s resume, we used ‘Cosine Similarity’ as it is an efficient way of finding how similar two documents are. Basically, the cosine similarity is the dot product of two vectors divided by the product of the magnitude of each vector. We divide the dot product by the magnitude because we are measuring only angle difference. On the other hand, dot product is taking the angle difference and magnitude into account. If we divide the dot product by the product of each vector’s magnitude, we normalize our data and only measure the angle difference. Dot product is a better measure of similarity if we can ignore magnitude.[7]

***Fig 15: Method defined for Cosine Similarity***

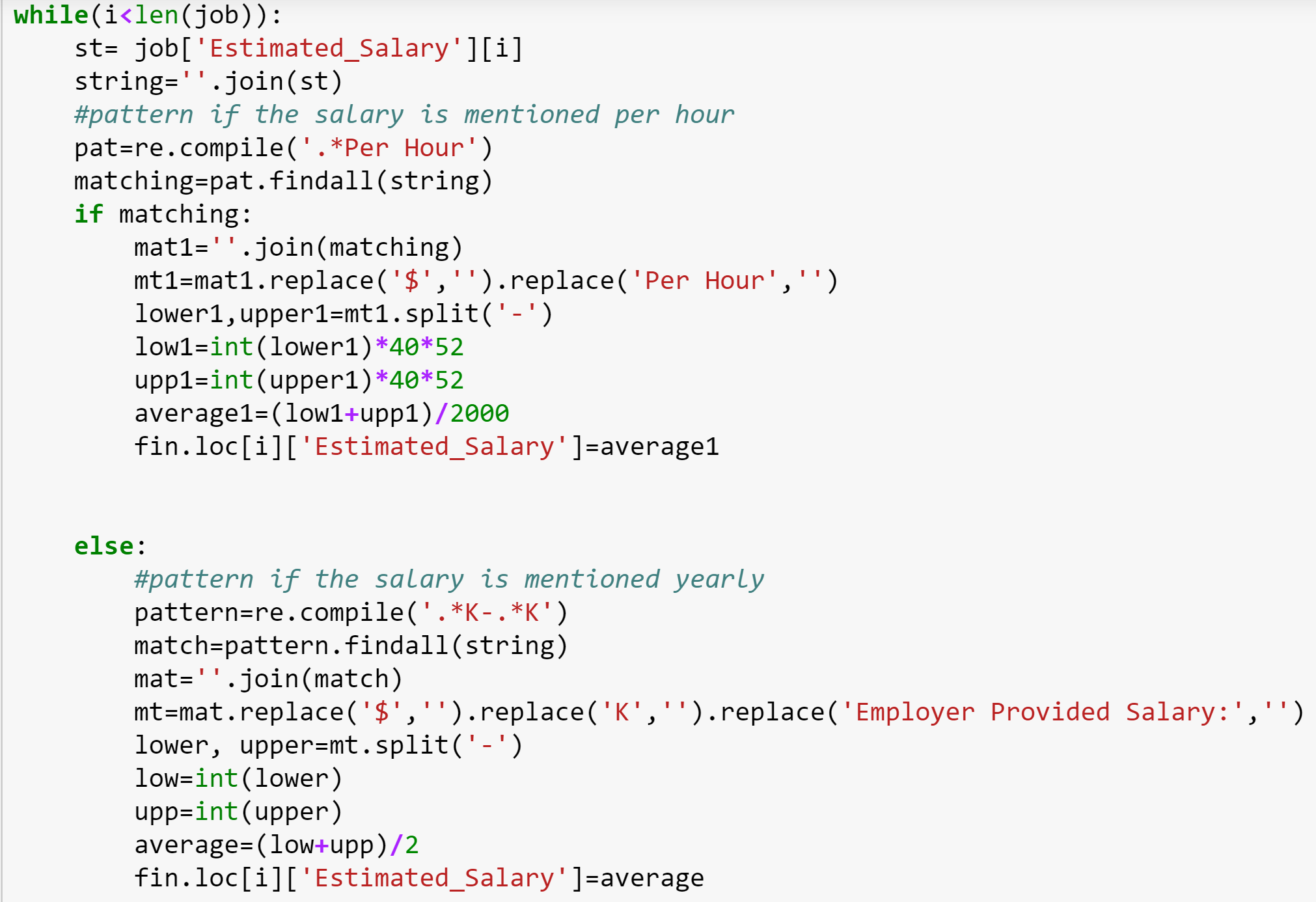
We then run the defined method of Cosine Similarity for each candidate’s parsed data against all the job descriptions scraped from Glassdoor. This is depicted in Fig 16.

***Fig 16: Cosine Similarity Code***

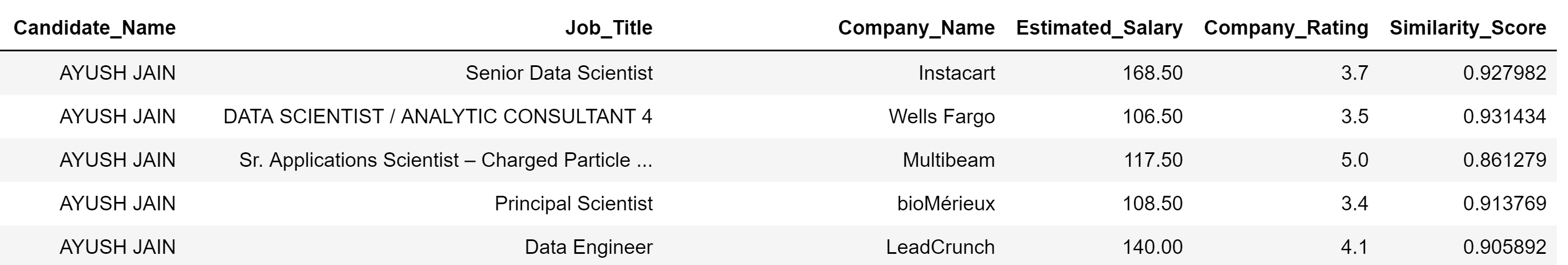
This similarity score is then appended to the data which contains each candidate’s details.

***Fig 17: Appended Similarity Score to the data-frame***

Next, we use pattern-matching to extract the float salary values from all the job listings. There are entries where Salary is mentioned in per-hour scale. We converted it to per-year scale to maintain the consistency of data.

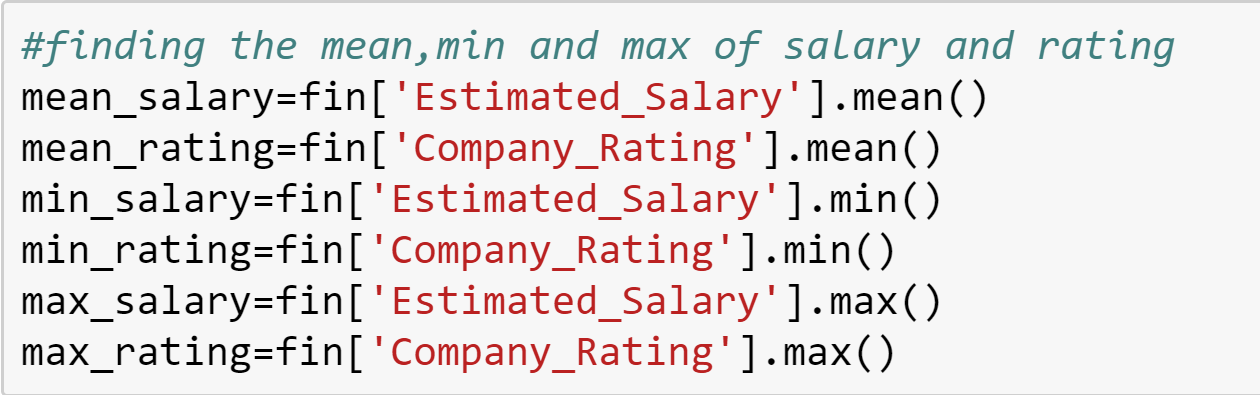
***Fig 18: Use of Pattern-Matching***

We used the same format of code for application of pattern-matching on Ratings extracted.

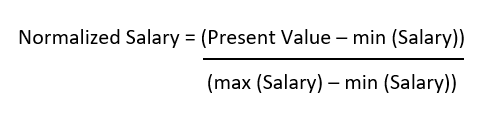


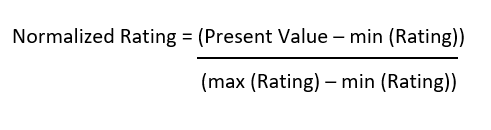
***Fig 19: Salary, Ratings and Similarity Score Displayed***

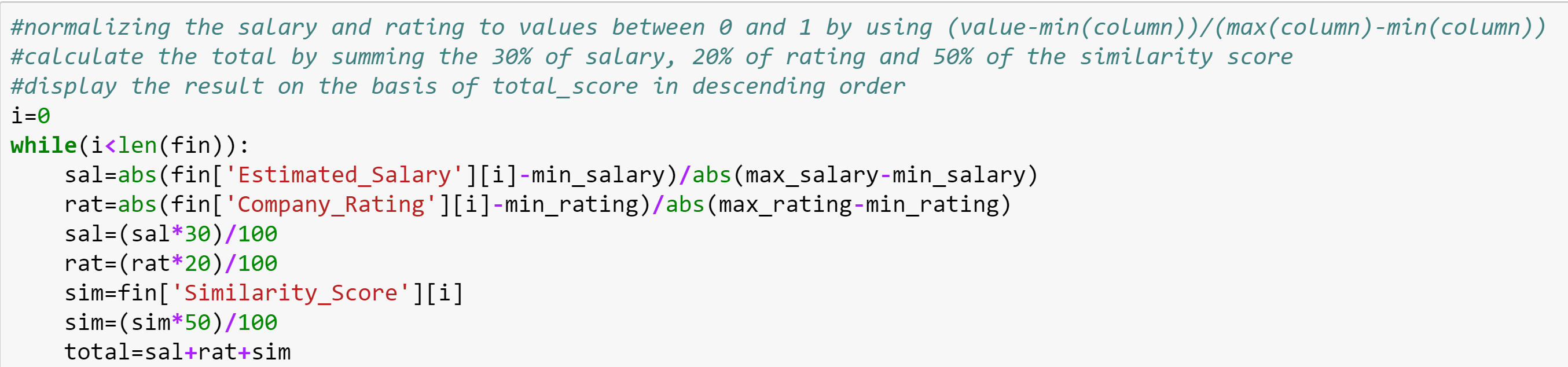
Next, we find the minimum, maximum and mean of the Salary and Ratings score. This is done to normalize these scores.

***Fig 20: Mean, Min, Max of Salary and Ratings***

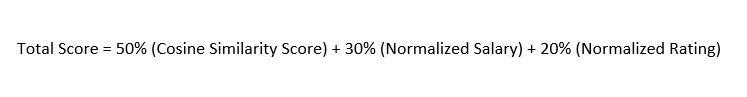
Since our Similarity Score lies between 0 and 1, we needed to normalize the salaries and ratings to generate their scores between 0 and 1. This will help us in assigning the weighted score to those fields.



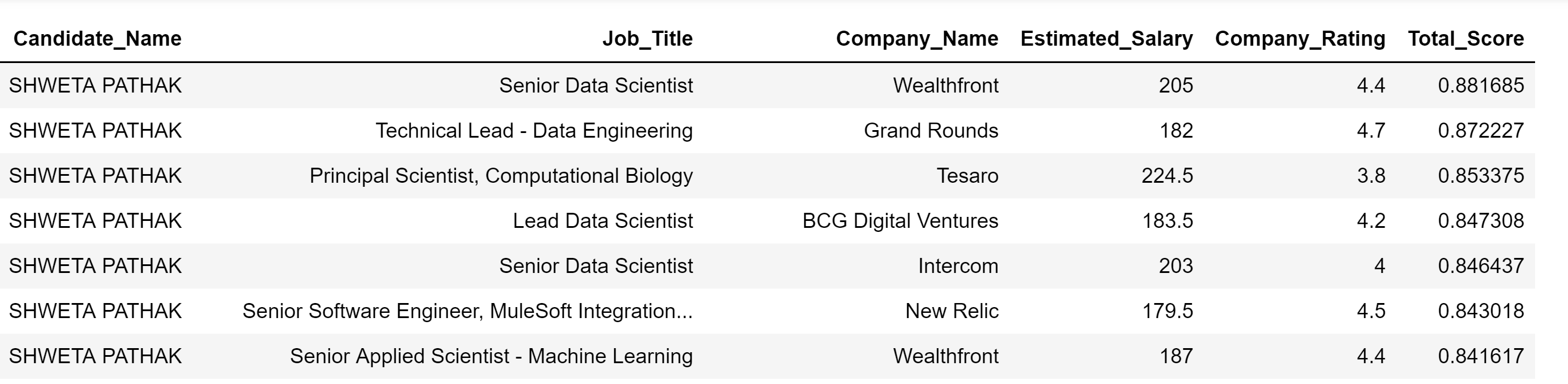


***Fig 21: Normalizing Salary and Ratings***

Our final step is to provide weights to each of these criterions so that we can generate a Total Score which is used to recommend the jobs. Our total score is calculated by summing 30% of salary, 20% of ratings and 50% of the similarity score.



Using this Total Score, we display the recommendation results in a descending order of the score. This means that the first displayed result is the candidate’s most-recommended. We also ensured that for every candidate, the similarity score of the skills should only be displayed if it is above 0.5. This ensures that there is maximum similarity.

***Fig 22: Recommendation System Results for a candidate***

# **RESULTS**

Since we have 3 different Python notebooks for this project, we have displayed the results for each notebook.

*a) Notebook-1 (WebScrapping\_GlassDoor):*

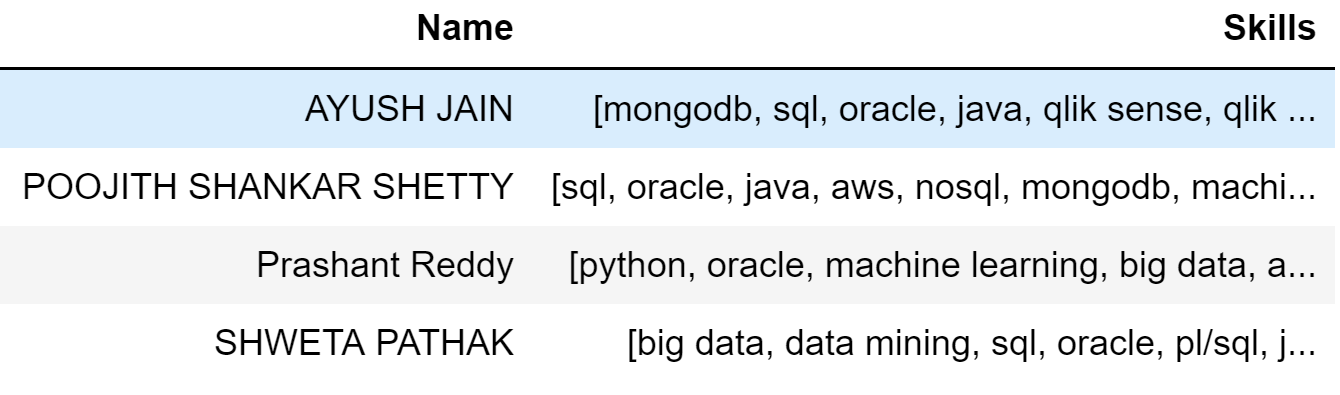
The result of this notebook displays the scraped data from GlassDoor’s website which includes the ‘Title’ of job, ‘Company’ name, ‘Location’ of the company, ‘Description’ included in the job-listing, ‘Salary’, ‘Company\_Rating’ on glassdoor and the ‘Skillset’ mentioned in their job-postings.

This result is posted on python and extracted as a CSV.

***Fig 23: Result of Notebook 1***

*b) Notebook-2 (Resume\_Parser):*

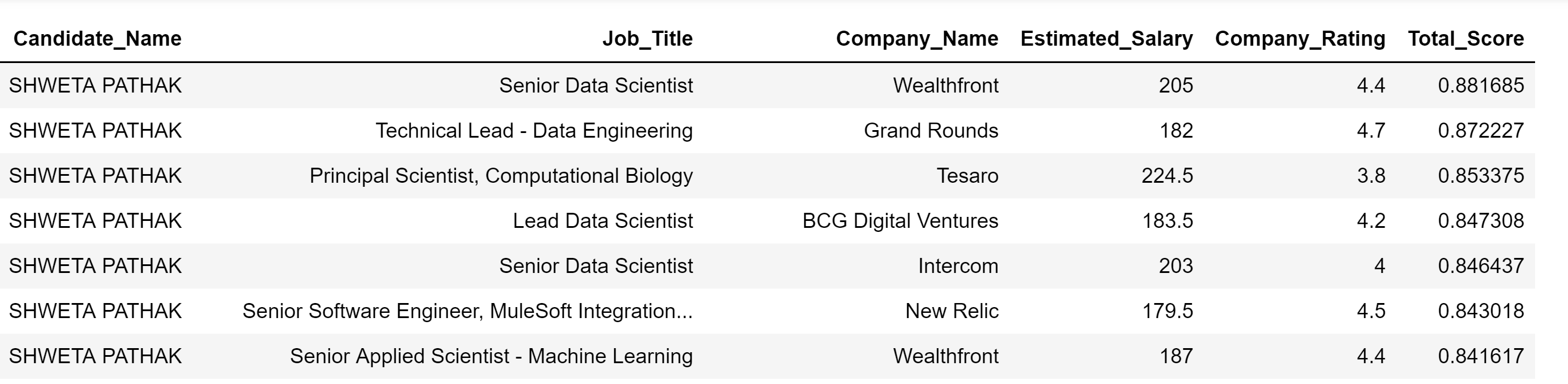
The result of this notebook displays the name of the candidate whose resume was parsed and the skills that he/she has mentioned. This result has also been displayed in python and extracted as a CSV.



***Fig 24: Result of Notebook 2***

*c) Notebook-3 (Recommender System):*

The result of this notebooks displays the recommended job-listings for each of the candidate based on the metrics we mentioned above.

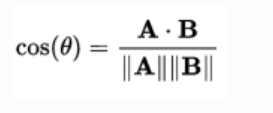
***Fig 25: Result of Notebook 3***

# **METRICS**

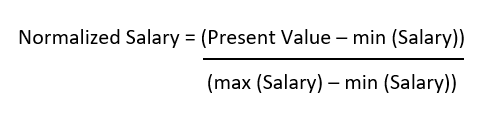
We have used 3 different metrics for this project

1. **Cosine Similarity (Similarity Score):**

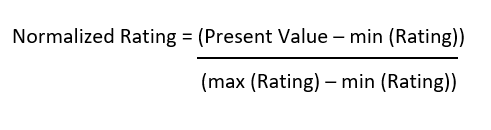
Find this between the job skills requirement and the candidate skillset.



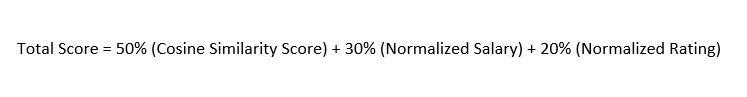
1. **Normalized Salary:**



**Normalized Rating**:



1. **Total Score:**



# **CONCLUSION**

Our project concludes with a Recommendation System which can recommend job-postings posted on a job-board (GlassDoor) to a candidate which the candidate should focus on applying by finding the similarity between the candidate’s skillset and skills in the job description. We also considered Salary and Ratings as a criterion to filter out postings for a candidate. We achieved this Recommender System with the help of Cosine Similarity metric.

We can further extend the scope and working of this Recommender System by utilizing it for more job-boards and postings. We can make use of multiple criterion.

# **REFERENCES**

1. Shaha Alotaibi, “A Survey of Job Recommender Systems”, ResearchGate [Online]

https://www.researchgate.net/publication/272802616\_A\_survey\_of\_job\_recommender\_systems

1. Web Scrapping, WebHarvy [Online]

https://www.webharvy.com/articles/what-is-web-scraping.html

1. WebScraping in Python using Selenium-BeautifulSoup-Pandas, Medium [Online]

https://medium.freecodecamp.org/better-web-scraping-in-python-with-selenium-beautiful-soup-and-pandas-d6390592e251

1. Jeffery Fosset Blog, “Tokenizing Raw Text in Python” [Online]

http://jeffreyfossett.com/2014/04/25/tokenizing-raw-text-in-python.html

1. Martina Pugliese, “English Stopwords and Python Libraries”, GitHub [Online]

https://martinapugliese.github.io/english-stopwords/

1. Recommender System, Wikipedia [Online]

https://en.wikipedia.org/wiki/Recommender\_system

1. Song Recommendation System using Cosine Similarity and Euclidian Distance, Medium [Online]  
   https://medium.com/@mark.rethana/building-a-song-recommendation-system-using-cosine-similarity-and-euclidian-distance-748fdfc832fdW