AI Career Path Recommender

Project Report

Introduction

In the modern world, there is a wide variety of possible career paths to choose from. For some, this might be a positive thing. However, for many others, this might be a bit overwhelming. In addition, our career plays a serious role in our lifetime. Thus, choosing a career path is a decision with a large impact.

With that in mind, we would like to help these individuals make more informed and data-driven decisions regarding their chosen career path.

In our project, we focus on providing AI-driven features for **everyday end-users** by creating a LinkedIn extension which recommends a career path to users. Moreover, the recommendation will include suggestions for popular jobs within that path. The recommended career path will be one of 10 pre-determined groups, and will be based on the user's data (education, experience, personal, etc.).

In addition, as we know, one of the most important parameters when searching for a job is the salary. Thus, alongside the recommendation, we include the estimated salary of the recommended career path as well as that of the popular jobs within it. This information will also allow the user to come to job interviews with an expected salary in mind.

Our extension provides several benefits to the LinkedIn platform and its users. First and foremost, it enhances the user experience by introducing new and useful features to the platform. This addition might also contribute to user engagement on LinkedIn, as well as add new job-seeking potential users. Last but not least, this may help people all around the world, changing their life for the better. For us, this is the most important benefit.

Data Collection and Integration

Original Datasets

In order to create said platform, we need information regarding the users. Therefore, we chose to use the 'people' dataset (from the original datasets provided). From it, we extracted features which capture the users' background, as well as their career paths. An in-depth description of the process will be provided in the Data Analysis section.

Additional Data

In order to elevate the user experience, when recommending a career path to users, we want to provide important information that does not appear in the given LinkedIn datasets – **estimated salaries**.

In other words, we would like to display a salary estimation for the career path that we recommend, as well as salary estimations for three of the most popular jobs within that path.

In order to gain this information, first, for each group, we want to obtain data regarding different salary listings of three of the most popular jobs within that group. We decided to obtain said information from the "Glassdoor" website (a highly reliable website, which includes relevant and verified salary data regarding different positions). We gathered this data by web scraping using the following tools: Bright Data Scraping Browser, Selenium and Beautiful Soup.

We defined an **item** (in our web scraping process) as the salary of a job title listing. In total, we gathered 757 valid items (roughly 25 for each job title scraped).

Bright Data was an essential tool for us during the scraping process (in order not to get blocked by the website). Due to the budget limitations on the platform, we chose to follow the lower bound that we set for $\alpha(group)$ in the project proposal. That is, we chose to scrape at least $\alpha(group) * 1000 = 700$ records ($\alpha(group) = 0.7$). As we can see, we scraped well above the lower bound.

After scraping the data, we calculated the estimated salary of a scraped job title as the average of all salaries scraped for it. Then, we calculated the estimated salary of a group as the average of all estimated salaries of the jobs associated with it.

As a result, we receive two final datasets: a dataset of estimated annual salaries for each group (10 rows, 1 column), and a dataset of estimated annual salaries for each job (30 rows, 1 column).

For the list of job titles we scraped for each group, an in-depth analysis of the scraping procedure, as well as the code, please refer to the Data Scraping Appendix.

Data Analysis

Analysis Techniques

As mentioned in the previous sections, we need the users' labels (career paths). First, we extracted the job title from the 'position' column (after removing missing values: None, “--"). We would want to use these job titles as the labels for our model. However, the data contains ~1 million unique job titles, and thus it is unfeasible.

While exploring the column, we noticed that the job title frequencies decay very quickly. For example, the 5th most common job title (president) appears ~20k times in the dataset, while the 20th most common one (vice president) appears only ~5k times.

As a result, in order to greatly reduce the variance of job titles, we filtered the data and kept only users with job titles of frequency higher than 3. We chose this threshold by using an elbow graph, which helped us find the “sweet spot” (significantly reducing the number of job titles, without removing too many records from our dataset). For the elbow graph, please refer to Fig 1 in the Plots Appendix.

Note that by using this threshold, we manage to keep over 60% of the data, while reducing the number of distinct job titles by over 95%. However, we still have over 30k distinct job titles. Unfortunately, this is still unfeasible to work with.

Thus, we decided to use an LLM to cluster the job titles into the following 10 general groups: **Management and Leadership, Healthcare and Medical, Education and Teaching, Business and Administration, Legal and Law, Technology and Engineering, Creative and Artistic, Financial and Accounting, Customer Service and Sales, Nonprofit and Social Services.**

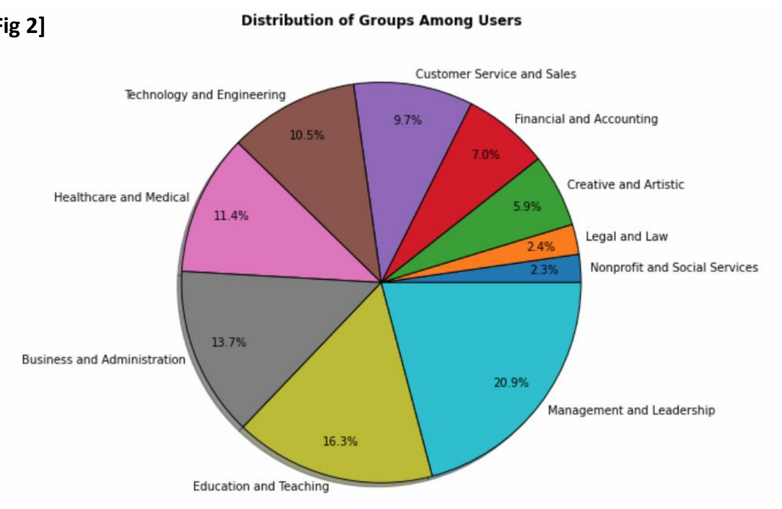
We chose these groups since they are broad (associated with many different job titles) and diverse (quite different from each other).

However, although our groups cover a broad range of job titles, some within our data may not belong in any of them. In order to prevent wrong group assignments, we defined an additional group called "Miscellaneous", and instructed the LLM to assign all said values to this group. For an in-depth description regarding the group assignment process, please refer to the LLM Appendix.

Finally, we filtered the remaining records by removing all users that were assigned to group "Miscellaneous", to receive the final data that we proceed with (**~1.4 million** records).

For the distribution of job title assignments to the ten groups, please see Fig 2 below:

[Fig 2]



As we can see:

- The groups' distribution is not uniform, resulting in quite imbalanced labels.
- The most common group is "Management and Leadership" (about 20% of our data). In addition, the least common ones are "Nonprofit and Social Services" and "Legal and Law" (about 2.5% of our data each).

Feature Selection

We extracted the following initial set of features from the data received in the previous sections:

- Number of degrees (non-negative integer)
- Highest degree type (ordinal, 0-4)
- Accumulated number of education years (non-negative integer)
- Education field cosine similarities (float, [-1, 1])
- Number of past jobs (non-negative integer)
- Accumulated number of experience years (non-negative float)
- Number of recommendations (non-negative integer)
- Number of certifications (non-negative integer)
- Number of spoken languages (non-negative integer)
- Number of volunteer activities (non-negative integer)
- Number of courses (non-negative integer)
- 'About' section embeddings (vector of floats, dimension 128)

A few remarks:

1. We chose this initial set of features since it covers the user's background from a variety of aspects (education, career, personal, etc.).
2. For the exact description and the extraction methods used for the features above, please refer to the Feature Selection Appendix.
3. Note that the final features we selected (among the set above), as well as the selection process, will be described in the following sections.

AI Methodologies

In this section, we conducted numerous tests which lead us to find the model that suits our task the best.

Since our data contains an extremely large number of records (~1.4 million), we performed said tests on 25% of it (by sampling). This allowed us to save valuable computational time, and thus perform as many tests as possible (given the time limit).

Moreover, we divided the sampled data into train and test sets using 80%/20% split. These sets were used in all of the conducted tests.

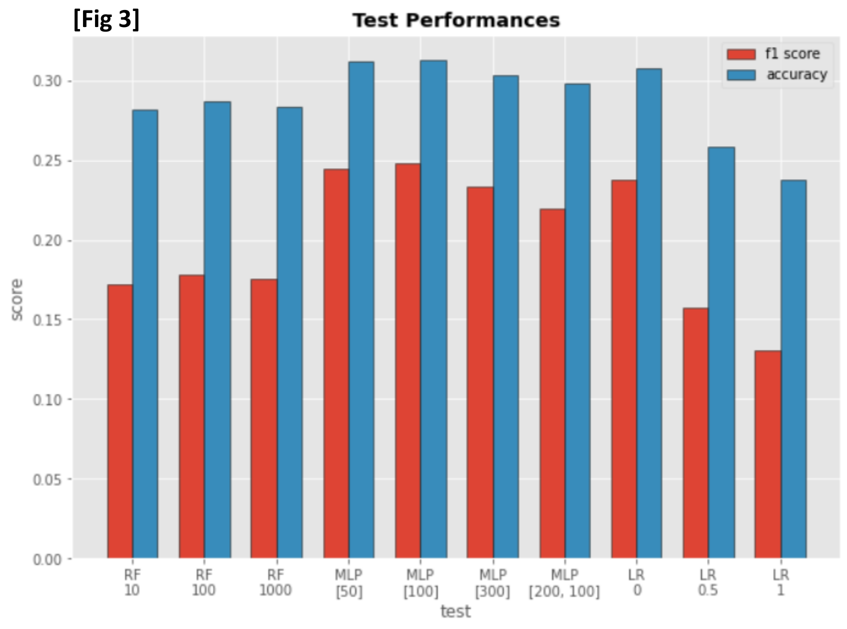
Model Selection

In order to find the most suitable model that we will proceed with, we tested three completely different models. Moreover, to compare the different models more thoroughly, we tried out different architectures for each model as well:

- For Multilayer Perceptron (MLP), we tested different number/size of hidden layers.
- For Random Forest (RF), we tested different number of trees.
- For Multiclass Logistic Regression (LR), we tested different regularization parameters.

For the performances of each model tested (F1 score and accuracy), please see Fig 3 below:

[Fig 3]



The model we chose to proceed with is **MLP, with a single hidden layer of size 100**.

From the graph above, this model achieved the best results. In particular, it received the following scores: F1 score of 0.248 and accuracy of 0.313.

*As seen in Fig 2, the distribution of the labels is imbalanced. Thus, while experimenting, we tried to improve our models' performance with that in mind. We did so by incorporating the inverse probability weights (IPW) of the labels (based on their empirical distribution) during the training process. Unfortunately, this attempt proved to be unsuccessful, and therefore we chose not to incorporate it, nor to display the results we received.

Feature Subset Selection

After choosing the model for the platform, we would like to select the optimal subset of features (among the initial feature set).

In order to achieve that, it is required to check all possible subsets of features. Unfortunately, this process is computationally unfeasible, as it is exponential in the number of features (in our case 2^{12}).

Therefore, to overcome this obstacle, we defined an iterative greedy approach, as follows:

First, we define a feature set that is maintained throughout the process. The starting feature set is defined as the set of all features that we extracted in previous sections.

Then, we iteratively try to remove features from that feature set. In each iteration, we remove a feature from the current feature set, one by one, and check if it improves our model's performance (specifically, the F1 score). While iterating over the features, if removing some feature indeed improves our F1 score, we permanently remove it from our feature set, and move on to the next iteration without it. In the next iteration, we again try to remove each of the features in the current feature set, check whether it improves our results, and so on.

When an iteration ends without any improvements to the F1 score (in other words, removing any feature from the current feature set lowers the score), we return the current feature set, which will be our final one.

From the procedure above, the resulting feature set was **the initial one**. That is, removing any feature from the initial feature set lowers the F1 score.

As a result, our final model is **MLP with a single hidden layer of size 100, trained on the final feature set.**

Evaluation and Results

In the previous sections, we used 25% of the data to find our final model. Now, that we had this model at hand, we evaluated its performance on our entire dataset, using several analysis methods.

General Metrics

Our final model received an F1 score of 0.235 and an accuracy of 0.307 on the test set.

Unfortunately, the model's performance is a bit underwhelming. Moreover, both metrics results were slightly higher when using the sampled dataset. This result is surprising to us, since we expect that training on more data would yield better results. However, the sampled dataset may not fully represent the full one (as it was randomly sampled).

Classwise Analysis

In this section, we performed classwise analysis in order to evaluate our model's performance on each group. In particular, we used classwise F1 scores and confusion matrix.

- From the analysis of the classwise F1 scores, we concluded the following:
    - Our model has quite an impressive predictive power on the “Education and Teaching” and “Management and Leadership” classes.
    - Our model preforms at a mediocre level on the “Technology and Engineering”, “Healthcare and Medical” and “Creative and Artistic” classes.
    - Our model performs quite poorly on the “Business and Administration”, “Financial and Accounting” and “Customer Service and Sales” classes.
    - Our model performs extremely poorly on the “Nonprofit and Social Services” and “Legal and Law” classes with an F1 score of ~0.

  For the F1 scores of each class, please refer to Fig 4 in the Plots Appendix.

- From the confusion matrix analysis, our model's most common errors are as follows:
    - Commonly misclassifying many different groups to be "Management and Leadership".
    - Misclassifying "Business and Administration" as "Management and Leadership" quite often. This insight makes sense to us, as these groups require similar skills and qualities.
    - Never classifying instances as "Nonprofit and Social Services", and rarely classifying instances as "Legal and Law". In our opinion, this result is not surprising, since these classes are extremely underrepresented in our data.

  For the confusion matrix of our model’s predictions, please refer to Fig 5 in the Plots Appendix.

Feature Importance

Let us be reminded that our final model is MLP, of which the predictions are hard to interpret. Still, we wanted to evaluate its "way of thinking". To achieve that, we assessed the importance of each feature.

As we know, the Random Forest model is extremely interpretable, and thus, it is easy to extract each feature's importance from it. In order to utilize this property, we trained an RF model using our final feature subset as the features, and **our model's predictions as the true labels**. Thus, we are “forcing” it to perform similarly to our final model. This process should result in the RF model having similar decision boundaries. After the training process, we used the feature importance of the RF model as an estimation for the feature importance of our model.

From the feature importance analysis, we concluded the following:

- The “Education field cosine similarity” features proved to be the most important for our model. This result is not surprising to us since the field attribute is closely related to a person's career.
- The second most important feature for our model is “About embedding”. This result is to be expected, given the fact that the 'about' section is a concise description of the person's background.
- It seems that the least important features for our model are “Number of languages” and “Number of volunteer activities”, with feature importance of zero for both. However, notice that this is only an estimation, and our final model indeed finds these feature somewhat useful (as seen in the Feature Subset Selection section).

For the feature importance of each feature, please refer to Fig 6 in the Plots Appendix.

Limitations and Reflection

First, the main dataset we worked with in this project (the LinkedIn ‘people’ dataset) is gathered from the real world. As a result, it is challenging to work with: missing values, spam values and hard-to-analyze string values from open text boxes. This issue negatively affects both the quality of the data, as well as the quality of the extractable features.

Moreover, our project includes working with big data (~1.4 million records). As a result, the computation time was extremely high. Although we performed quite a large amount of testing, we still have to compromise on what tests to perform.

Lastly, as seen in Fig 2, the sizes of the groups vary quite drastically (e.g. group “Nonprofit and Social Services” contains of only 2.3% of the data, compared to group “Education and Teaching” which contains above 20%). As a result, the data labels are imbalanced.

All the above constraints and challenges might harm our final model’s performance (each in its own different aspect).

Conclusion

To summarize, we used real life data to create a machine learning system which utilizes users’ background information in order to give data-driven recommendations for a career path.

As seen in the General Metrics section, there is room for improvement for our model’s performance. However, according to the Classwise Analysis section, the performance for a couple of specific classes is quite impressive. Hence, our system could especially help users in these domains, and change their lives for the better.

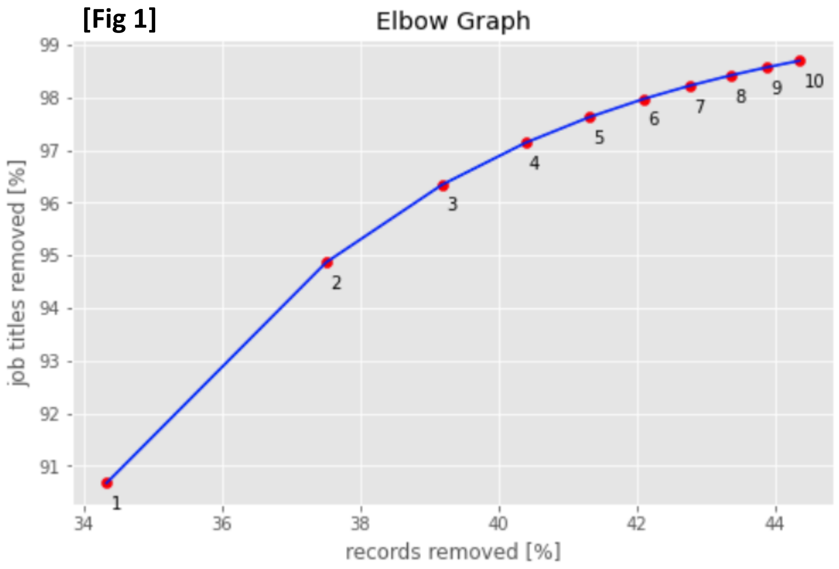
To elevate our project, and to demonstrate our platform’s capabilities, we implemented a prototype for our system, and made it accessible through a simple user interface. This user interface is available in the end of our project’s notebook. A video walkthrough of the user interface is available in the link.

Moreover, a link to the GitHub repository for the project is available in the link.

Plots Appendix

Fig 1: Elbow Plot for Job Title Reduction (Under the Analysis Techniques Section)

[Fig 1]



As we can see, the higher thresholds should better be avoided (losing too many data records, without a justified increase to the number of job titles removed), but there is no specific point where the tradeoff clearly seems to be optimal. Thus, we chose threshold 3, which seems to be in the middle point of the tradeoff.

Fig 4: Classwise F1 Scores (Under the Classwise Analysis Section)

[Fig 4]

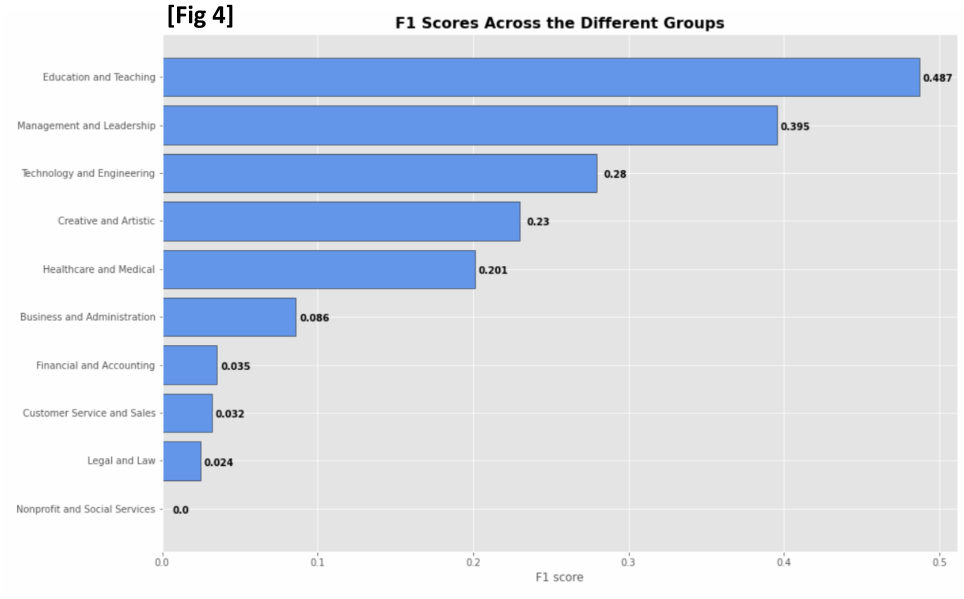
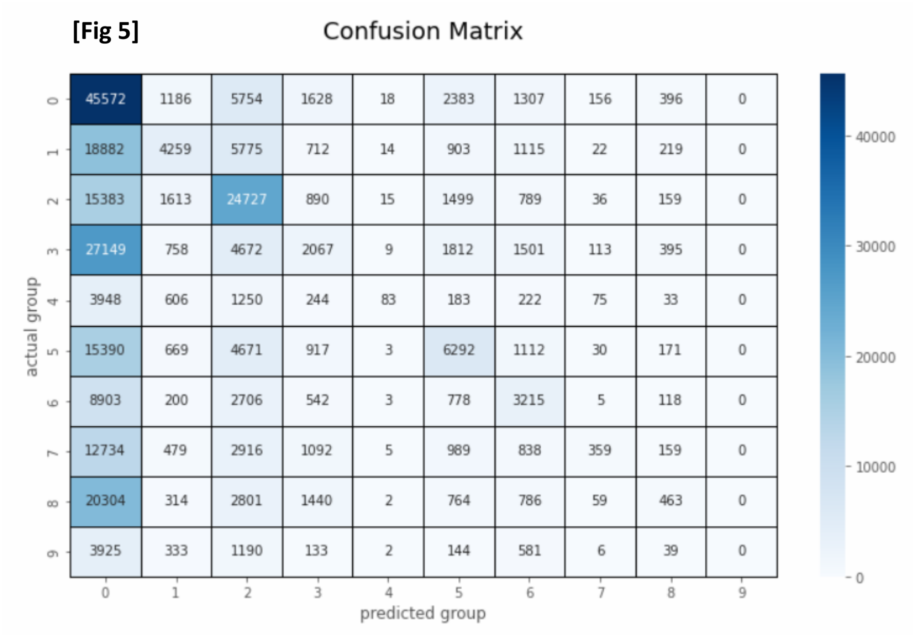


Fig 5: Confusion Matrix of Our Model's Predictions (Under the Classwise Analysis Section)

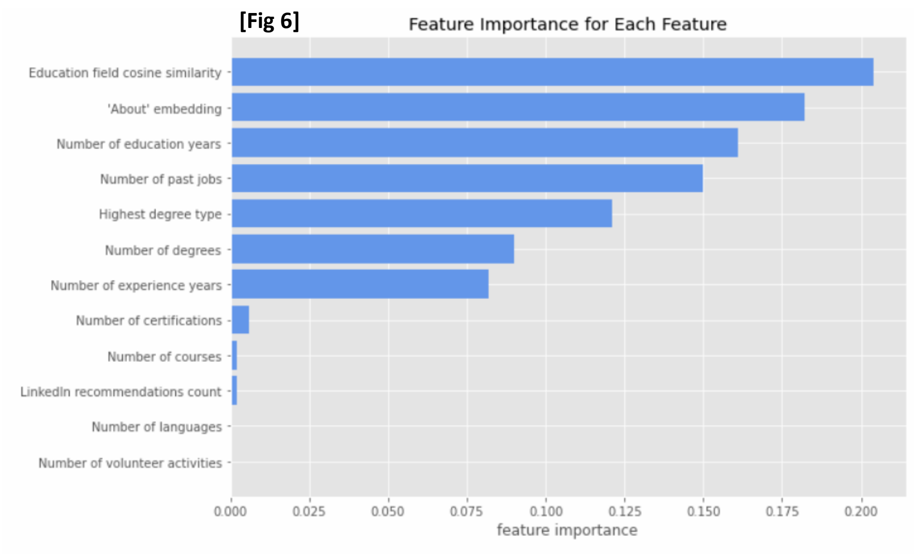


actual \ predicted	0	1	2	3	4	5	6	7	8	9
0	45572	1186	5754	1628	18	2383	1307	156	396	0
1	18882	4259	5775	712	14	903	1115	22	219	0
2	15383	1613	24727	890	15	1499	789	36	159	0
3	27149	758	4672	2067	9	1812	1501	113	395	0
4	3948	606	1250	244	83	183	222	75	33	0
5	15390	669	4671	917	3	6292	1112	30	171	0
6	8903	200	2706	542	3	778	3215	5	118	0
7	12734	479	2916	1092	5	989	838	359	159	0
8	20304	314	2801	1440	2	764	786	59	463	0
9	3925	333	1190	133	2	144	581	6	39	0

Let us be reminded of the encoding of groups to numerical labels:

- Management and Leadership: 0
- Healthcare and Medical: 1
- Education and Teaching: 2
- Business and Administration: 3
- Legal and Law: 4
- Technology and Engineering: 5
- Creative and Artistic: 6
- Financial and Accounting: 7
- Customer Service and Sales: 8
- Nonprofit and Social Services: 9

Fig 6: Feature Importance (Under the Feature Importance Section)



Feature Selection Appendix

Let us elaborate below regarding each of our initially selected features to extract:

From the **Education** column, we extracted the following features:

- Number of degrees – we extracted the number of degrees as the length of the 'education' array. Moreover, for users who do not have any education information (empty array or None), we defined their number of degrees as 0.

- Highest degree type – first, we assigned an ordinal value to each academic education level:
  - Non-academic degrees – 1
  - Bachelor – 2
  - Masters – 3
  - PhD – 4

  Then, we extracted the highest degree type as the maximum among all the 'degree' attributes from the elements under the education column. Moreover, for users who do not have any education information (empty array or None), we defined their highest degree type as 0.

- Accumulated number of education years – extracted as the sum of durations of all elements under the 'education' column (taking into account only elements with both non-null 'start_year' and 'end_year' attributes). Moreover, for users who do not have any education information (empty array or None), we defined their accumulated number of education years as 0.

- Education field cosine similarities – the 'field' attribute (of the elements under the 'education' column) holds information regarding the user's degree domain (e.g. Computer Science). Thus, we would like to extract the similarity between this domain and each of the potential career paths (each of the 10 groups). In order to achieve this goal, first we embedded the users' fields, as well as the group names. Second, we calculated the cosine similarities between each of the fields to each of the group names. Lastly, for each user, we defined their education field cosine similarity to a group name x as the maximum cosine similarity among all cosine similarities (between all the user's fields and x). Note that we chose the maximum aggregation (instead of the "obvious" one, the average). This is because in our eyes, when a user gains knowledge in a certain field which is highly associated with some group, we do not want other less associated fields to decrease this similarity (due to division by the number of fields when averaging). Moreover, for users who do not have any field information, we define their similarity with all groups as 0 (since it does not reflect any information).

From the **Experience** column, we extracted the following features:

- Number of past jobs - we extracted the number of past jobs as the length of the ‘experience’ array. Moreover, for users who do not have any experience information (empty array or None), we defined their number of past jobs as 0.

- Accumulated number of experience years - extracted as the sum of durations of all elements under the ‘experience’ column (taking into account only elements with non-null ‘duration_short’ attributes). Moreover, for users who do not have any experience information (empty array or None), we defined their accumulated number of experience years as 0.

From the **other** columns, we extracted the following features:

- Number of recommendations – the values under the 'recommendations_count' column. Moreover, for users who have a null value in this column, we define their number of recommendations as 0.

- Number of certifications - we extracted the number of certifications as the length of the ‘certifications’ array. Moreover, for users who do not have any certifications information (empty array or None), we defined their number of certifications as 0.

- Number of spoken languages - we extracted the number of spoken languages as the length of the ‘languages’ array. Moreover, for users who do not have any languages information (empty array or None), we defined their number of spoken languages as 1 (assuming all users speak at least one language).

- Number of volunteer activities - we extracted the number of volunteer activities as the length of the ‘volunteer_experience’ array. Moreover, for users who do not have any volunteer activities information (empty array or None), we defined their number of volunteer activities as 0.

- Number of courses - we extracted the number of courses as the length of the ‘courses’ array. Moreover, for users who do not have any courses information (empty array or None), we defined their number of courses as 0.

- 'About' section embeddings – since the values under the ‘about’ column are strings and an input of an open text box, we decided to embed them into vectors using an embedding model.

***Note:** the embedding model we used for feature extraction is the BERT contextualized sentence embedding model. Specifically, we imported the 'sent_small_bert_L2_128' model from the SparkNLP library. This model takes a sentence as input, and returns a contextualized embedding vector of dimension 128 as output. We chose a relatively low embedding size, in order to tackle the high computational cost resulting from our extremely large amounts of data (~1.4 million records).

LLM Appendix

We used **ChatGPT** as our LLM, in order to cluster the large amount of job titles to the mentioned 10 groups. In particular, we used the ‘gpt-3.5-turbo’ model from the OpenAI API.

As mentioned in the report, we have over 30k job titles left after filtering. Therefore, we cannot use ChatGPT to assign groups to all of them in a single query. Thus, we split the job titles into smaller chunks. We assumed that as the chunk size gets larger, ChatGPT's performance will be poorer (the large prompt might be confusing). Therefore, we decided to split the job titles into 1,000 chunks (of size ~35 each). Note that since we defined the groups in advance, the assignment of each job title to a group is independent of that of other job titles.

The prompts we used in the assignment process are as follows:

1. System prompt: used to provide context to the model as preparation for future queries.

Your task is to assign job titles to the distinct group that they belong in. You should assign job titles ONLY to one of the following groups:

1. *Management and Leadership*
2. *Healthcare and Medical*
3. *Education and Teaching*
4. *Business and Administration*
5. *Legal and Law*
6. *Technology and Engineering*
7. *Creative and Artistic*
8. *Financial and Accounting*
9. *Customer Service and Sales*
10. *Nonprofit and Social Services*
11. *Miscellaneous*

2. User prompt: used to query the model.

Hey, I have a list of job titles, and I want you to assign each one into a distinct group. You should assign it only to one of the following groups:

1. *Management and Leadership*
2. *Healthcare and Medical*
3. *Education and Teaching*
4. *Business and Administration*
5. *Legal and Law*
6. *Technology and Engineering*
7. *Creative and Artistic*
8. *Financial and Accounting*
9. *Customer Service and Sales*
10. *Nonprofit and Social Services*
11. *Miscellaneous*

I want your output to be in JSON format, where each key is a job title that I will provide, and the value is the group that you assign it. I remind you that you can assign job titles ONLY to one of the groups that I gave you.

Here is the list of job titles:
{chunk}

As we can see from the user prompt, for each chunk, we asked ChatGPT to provide the output in the format of a JSON file, where the keys of the dictionary are the job titles, and their value is the assigned group.

After receiving the group assignments of each chunk, we converted all JSON files into dictionaries, and combined all of them into a single one. When validating ChatGPT's output, we encountered two main problems:

First, there were 90 job titles in our dataset that were not assigned to a group. It appeared that the vast majority of such missing job titles are spam and invalid values. However, some seem to be valid job titles (with minor typos). We assigned a group to such valid job titles manually, as well as assigned invalid job titles to group "Miscellaneous".

Second, 1,293 job titles were assigned to 212 groups which are not among the 11 groups specified in the prompt. We noticed that some of those groups are different variations of the specified ones. In that case, we reassigned the job titles in those groups to the correct one. After doing so, we were left with 762 job titles that were assigned to 190 groups which are not among the 11 groups (e.g. "Science and Research", "Human Resources", etc.).

Since the number of those groups is high, we cannot reassign them manually. Moreover, we noticed that the vast majority those groups contain only a single job title, and the largest one contains only 50 job titles. Therefore, they do not cover a high enough number of job titles to justify adding them as new labels.

Thus, we would like to reassign the job titles under these groups to group "Miscellaneous". However, this will result in removing the corresponding records later on. Nevertheless, upon examination, these records consist of less than 1% of the data. Hence, we can confidently perform the reassignment mentioned above.

*Since we cannot import the OpenAI python library to Databricks, we ran the corresponding code on our personal computer. The code will be attached in the submission as a separate file.

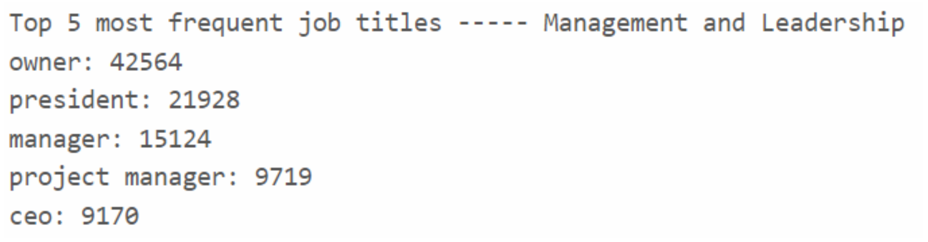
**The full LLM responses to our queries, the full created dictionary, as well as the code, are available in the link.

Data Scraping Appendix

Finding Which Job Titles to Scrape

In order to choose which job titles to scrape for each group, we found the top 5 most common, and chose the top 3 accessible ones among them (job titles that are clearly defined on “Glassdoor”).

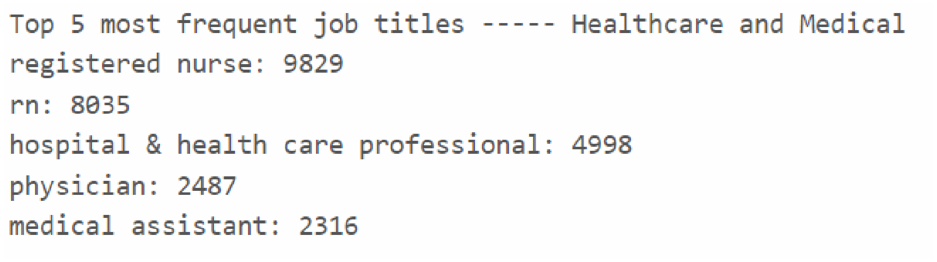
For group **Management and Leadership**:



```
Top 5 most frequent job titles ----- Management and Leadership
owner: 42564
president: 21928
manager: 15124
project manager: 9719
ceo: 9170
```

The job titles we chose to scrape for this group are: **manager, project manager, ceo.**

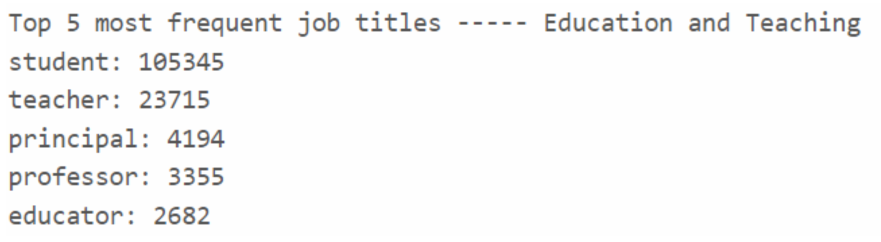
For group **Healthcare and Medical**:



```
Top 5 most frequent job titles ----- Healthcare and Medical
registered nurse: 9829
rn: 8035
hospital & health care professional: 4998
physician: 2487
medical assistant: 2316
```

The job titles we chose to scrape for this group are: **registered nurse, hospital & health care professional, physician.**

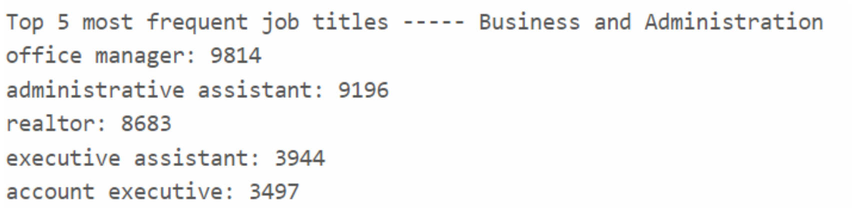
For group **Education and Teaching**:



```
Top 5 most frequent job titles ----- Education and Teaching
student: 105345
teacher: 23715
principal: 4194
professor: 3355
educator: 2682
```

The job titles we chose to scrape for this group are: **teacher, principal, professor.**

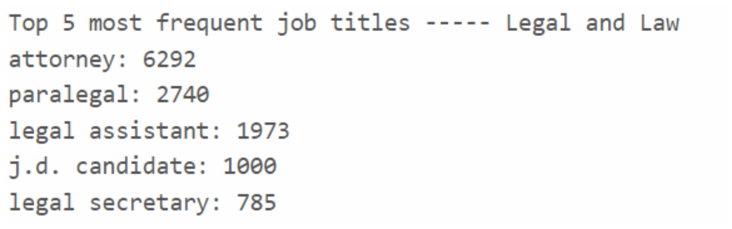
For group **Business and Administration**:



```
Top 5 most frequent job titles ----- Business and Administration
office manager: 9814
administrative assistant: 9196
realtor: 8683
executive assistant: 3944
account executive: 3497
```

The job titles we chose to scrape for this group are: **office manager, administrative assistant, realtor.**

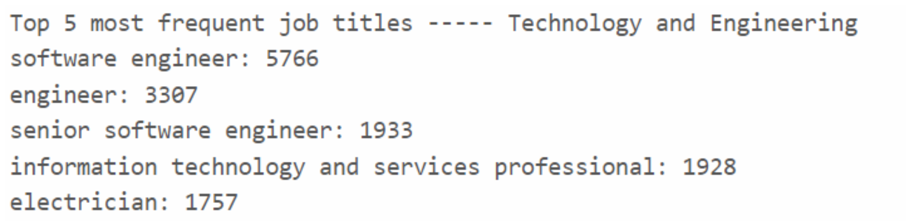
For group **Legal and Law**:



```
Top 5 most frequent job titles ----- Legal and Law
attorney: 6292
paralegal: 2740
legal assistant: 1973
j.d. candidate: 1000
legal secretary: 785
```

The job titles we chose to scrape for this group are: **attorney, paralegal, legal assistant.**

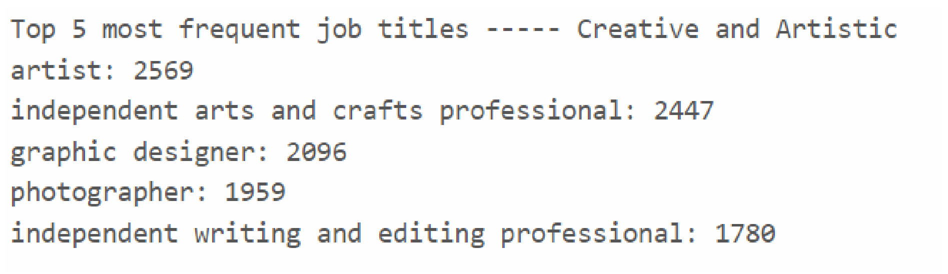
For group **Technology and Engineering**:



```
Top 5 most frequent job titles ----- Technology and Engineering
software engineer: 5766
engineer: 3307
senior software engineer: 1933
information technology and services professional: 1928
electrician: 1757
```

The job titles we chose to scrape for this group are: **software engineer, information technology and services professional, electrician.**

For group **Creative and Artistic**:



```
Top 5 most frequent job titles ----- Creative and Artistic
artist: 2569
independent arts and crafts professional: 2447
graphic designer: 2096
photographer: 1959
independent writing and editing professional: 1780
```

The job titles we chose to scrape for this group are: **graphic designer, photographer, independent writing and editing professional (content writer).**

For group **Financial and Accounting**:

```
Top 5 most frequent job titles ----- Financial and Accounting
accountant: 3991
accounting professional: 2496
controller: 2424
bookkeeper: 2019
financial services professional: 1581
```

The job titles we chose to scrape for this group are: **accountant, controller, bookkeeper.**

For group **Customer Service and Sales**:

```
Top 5 most frequent job titles ----- Customer Service and Sales
sales: 7661
account manager: 4892
sales manager: 4077
sales associate: 3644
retail professional: 3630
```

The job titles we chose to scrape for this group are: **account manager, sales manager, sales associate.**

For group **Nonprofit and Social Services**:

```
Top 5 most frequent job titles ----- Nonprofit and Social Services
pastor: 2476
social worker: 1615
counselor: 1110
program coordinator: 879
volunteer: 746
```

The job titles we chose to scrape for this group are: **pastor, social worker, program coordinator.**

The Scraping Procedure

First, let us mention that all the job title listings were searched in the US (since the records in the LinkedIn datasets are mostly from American users). Moreover, for each job title, we scraped all the visible listings on the job title's page. In addition, we scraped a separate dataset for each job title. Each dataset contains a single column (the scraped salary), and the number of rows is the number of scraped items. Below are the number of scraped items for each of the mentioned job titles, separated by their assigned group:

- Manager: 30
- Project manager: 22
- Ceo: 22

- Registered nurse: 27
- Hospital and healthcare professional: 26
- Physician: 27

- Teacher: 29
- Principal: 30
- Professor: 26

- Office manager: 24
- Administrative assistant: 28
- Realtor: 26

- Attorney: 28
- Paralegal: 28
- Legal assistant: 26

- Software engineer: 26
- IT and services professional: 27
- Electrician: 26

- Graphic designer: 28
- Photographer: 26
- Content writer: 21

- Accountant: 26
- Controller: 26
- Bookkeeper: 23

- Account manager: 15
- Sales manager: 17
- Sales associate: 30

- Pastor: 16
- Social worker: 23
- Program coordinator: 28

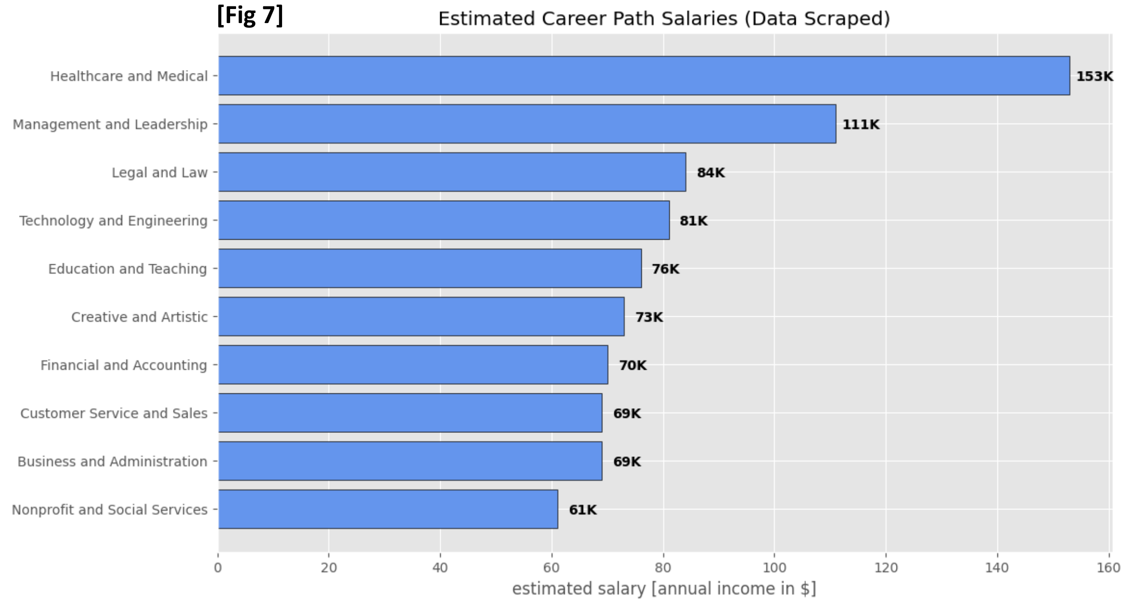
The Data Processing Procedure

Let us describe the scraped data processing:

- The salary values were scraped as strings with extra characters. First, we cleaned the scraped values by removing those characters.
- Some salary values were available in the form of an hourly rate, whereas others were available as an annual income. We chose to proceed with the annual income format, and converted the hourly rates by multiplying them by the standard number of work hours in a work year (2,080).
- Some salary values were scraped as a range. In this case, we chose the middle of the range.

For the estimated salary for each group, please see Fig 7 below:

[Fig 7]

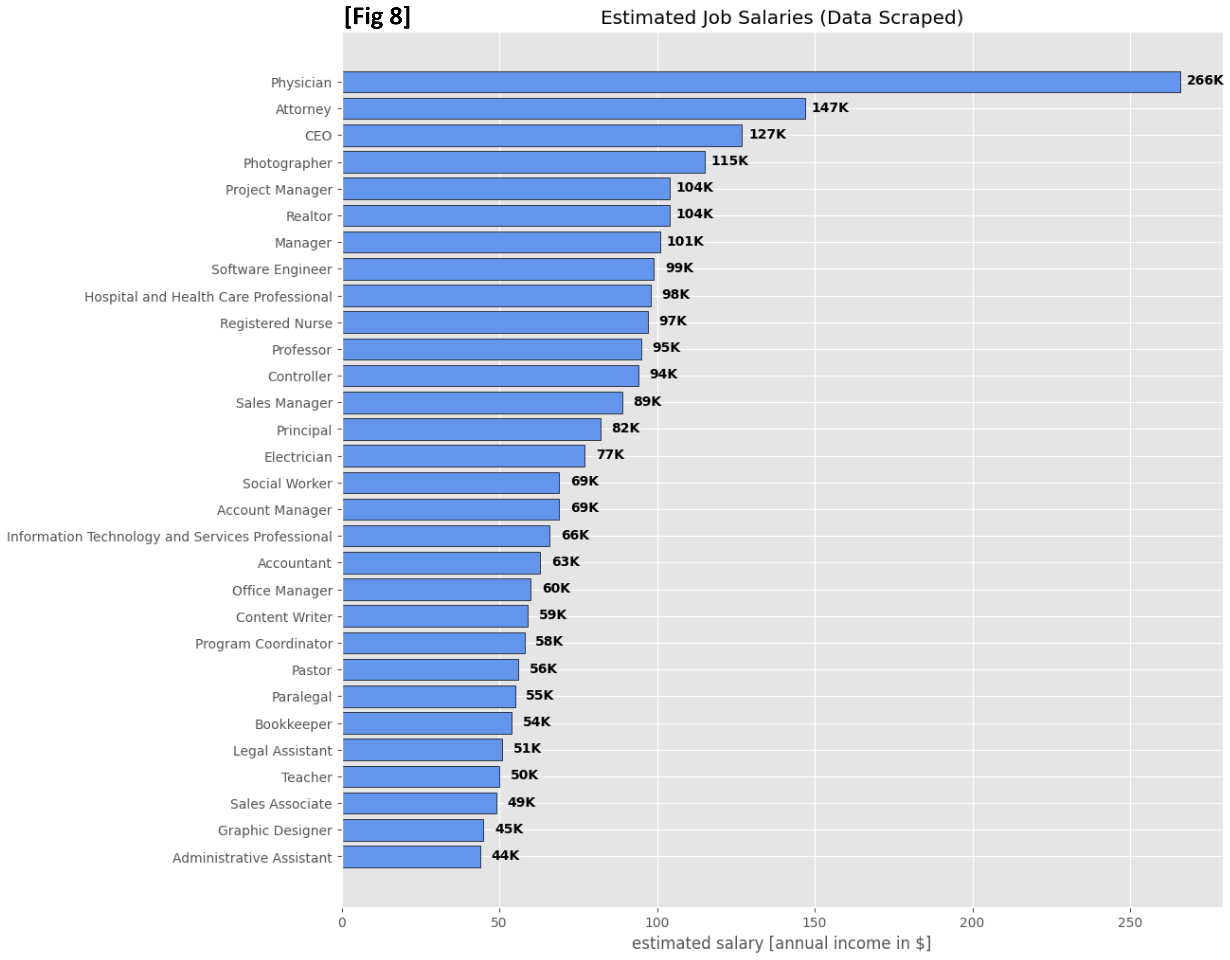


From the graph above, we conclude the following:

- The highest paying career path is “Healthcare and Medical”.
- The lowest paying career path is “Nonprofit and Social Services”.
- The “Technology and Engineering” career path is placed surprisingly low (only 4th place). However, upon inspection, two of the three job titles scraped for this career path (IT and electrician) are not among the group’s “high-end” jobs.

In addition, For the estimated salary for each job title, please see Fig 8 below:

[Fig 8]



From the graph above, we conclude the following:

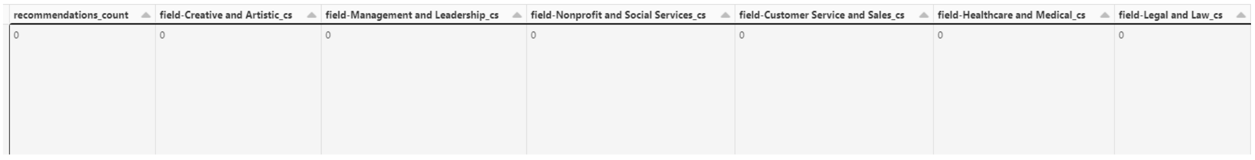
- The highest paying job is “Physician” (which makes sense).
- The lowest paying job is “Administrative Assistant”.

*Since we cannot import the scraping libraries to Databricks, we ran the corresponding code on Google Colab. The notebook will be attached in the submission as a separate file.

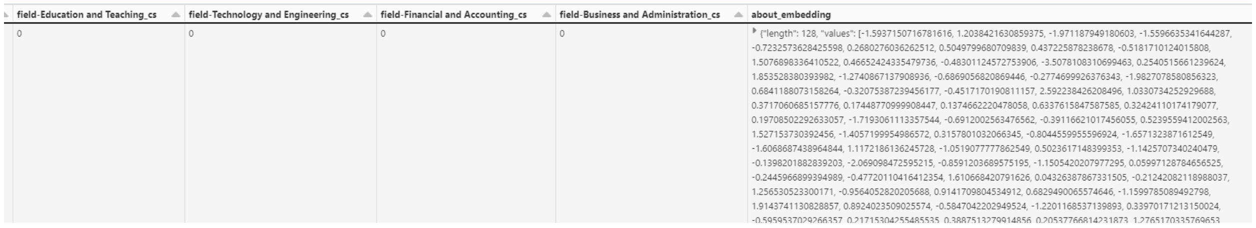
**To view the processed scraped data, as well as the code, click the link.

Datasets Images Appendix

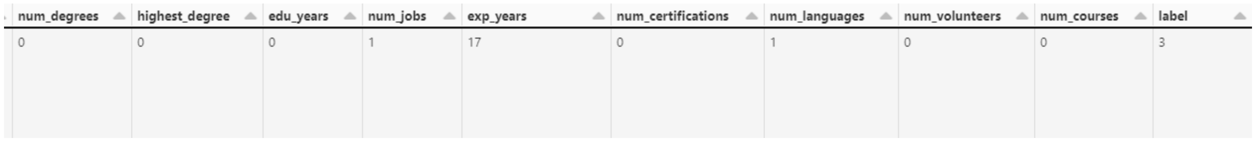
Let us display the dataset of the extracted features and labels:



recommendations_count	field-Creative and Artistic_cs	field-Management and Leadership_cs	field-Nonprofit and Social Services_cs	field-Customer Service and Sales_cs	field-Healthcare and Medical_cs	field-Legal and Law_cs
0	0	0	0	0	0	0



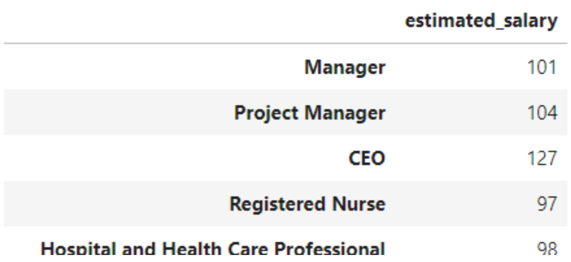
field-Education and Teaching_cs	field-Technology and Engineering_cs	field-Financial and Accounting_cs	field-Business and Administration_cs	about_embedding
0	0	0	0	{"length": 128, "values": [-1.5937150716781616, 1.2038421630859375, -1.971187949180603, -1.559663534164428 7, -0.7232573628425598, 0.2680276036262512, 0.5049799680709839, 0.437225878238678, -0.5181710124015808, 1.507689833641052 2, 0.4665242433547973 6, -0.4830112457275390 6, -3.507810831069946 3, 0.2540515661239624, 1.853528380393982, -1.2740867137908936, -0.6869056820869446, -0.277469992637634 3, -1.9827078580856323, 0.6841188073158264, -0.3207538723945617 7, -0.4517170190811157, 2.592238426208496, 1.033073425292968 8, 0.371706068515777 6, 0.1744877099990844 7, 0.137466222047805 8, 0.633761584758758 5, 0.3242411017417907 7, 0.1970850229263305 7, -1.719306111335754 4, -0.691200256347656 2, -0.391166210174560 55, 0.523955941200256 3, 1.52715373039245 6, -1.405719995498657 2, 0.315780103206634 5, -0.804455995559692 4, -1.657132387161254 9, -1.606868743896484 4, 1.117218613624572 8, -1.051907777786254 9, 0.502361714839935 3, -1.142570734024047 9, -0.139820188283920 3, -2.069098472595215, -0.859120368957519 5, -1.150542020797729 5, 0.0599712878465652 5, -0.244596689939498 9, -0.477201104164123 54, 1.610668420791626, 0.0432638786733150 5, -0.212420821189880 37, 1.256530523300171, -0.956405282020568 8, 0.914170980453491 2, 0.682949006557464 6, -1.159978508949279 8, 1.914374113082885 7, 0.892402350902557 4, -0.584704220294952 4, -1.220116853713989 3, 0.339701712131500 24, -0.5959537029266357, 0.21715304255485535, 0.3887513279914856, 0.20537766814231873, 1.2765170335769653



num_degrees	highest_degree	edu_years	num_jobs	exp_years	num_certifications	num_languages	num_volunteers	num_courses	label
0	0	0	1	17	0	1	0	0	3

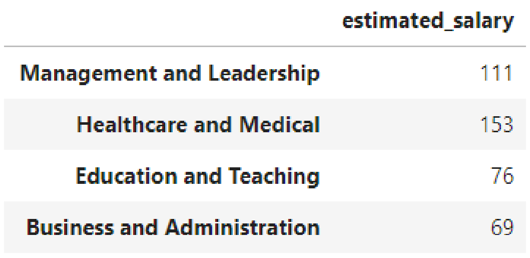
Now, let us display the web scraped datasets that were collected.

Estimated job salaries:



	estimated_salary
Manager	101
Project Manager	104
CEO	127
Registered Nurse	97
Hospital and Health Care Professional	98

Estimated group salaries:



	estimated_salary
Management and Leadership	111
Healthcare and Medical	153
Education and Teaching	76
Business and Administration	69