Data Job Posting Analysis

Introduction

As a job seeker, navigating hundreds or even thousands of job postings can be an arduous process. An applicant can spend 5-8 hours a day searching and applying for positions. As a modern-day computer science student interested in data science, automating the searching process can be a life saver, increase both the efficiency and quality of job searching. In this project, I aim to build a pipeline to collect job postings, glean insights using sophisticated AI tools, clean the data, and analyze them to derive comprehensive insights into the job market. Once built, it will save me a lot of time searching for a perfect match on the job market.

Methods

Data is collected using web scraping techniques, key words of Data Analyst and Data Scientist are used to collect web information of job postings, including, applicationsCount, applyType, applyUrl, benefits, companyId, companyName, companyUrl, contractType, description, experienceLevel, jobId, jobUrl, location, postedTime, posterFullName, posterProfileUrl, published date, salary, sector, title, and workType. Data is saved as .csv file and is processed in Visual Studio 2019 with anaconda environment, python version 3.11. OpenAI API and ChatGPT 4o-mini is called to extract more information from the description column for skills, salary and whether it's a remote work. Even though the salary is already in the scraper response, some job postings don't have that information under the salary section but in the description. Several libraries are adapted to match similar words together, such as K-means clustering from SKlearn and fuzzymatch. Seaborn and Matplotlib are used to plot visualizations.

Data Cleaning and Pre-Processing

A weeks' posting, about 1515 entries of data related job postings are scraped from Linkedin using existing actors on Apify and saved as one .csv file. Due to the nature of online postings and the imperfect of AI generative models, there is a lot to do to clean the dataset. Job titles have all kinds of descriptions in it, salary is missing or have typo. Locations are not standardized with different names for the same location. At last, but not least, the scripts generated by OpenAI have many variations for the same thing. Data cleaning is challenging in this manner.

Data Preprocessing

Description Information Extraction

The information extraction is done by OpenAI API and ChatGPT 4o-mini. A code calling the API, a concatenated prompt with the job description and my customized prompt is send to the server. The server returns a response with a python dictionary data structure with key names of min_years_of_experience, min_hourly_salary, max_hourly_salary, min_yearly_salary, max_yearly_salary, required_degree, remote_work, required_skills. Each row of description is processed this way, and each response is saved into a new column, then each key is separated and saved into a new column.

Due to the variation of job titles, it will be impossible to plot data against job titles. So I simplified them into four categories: Intern, Data Analyst, Data Scientist and Business Intelligence. These four categories consist of the majority of the posting.

Salary Cleaning

First the dataset is randomly sampled with a sample size 20, the salary column is compared with the hourly and yearly salary extract by ChatGPT. If salary is not None but no information is obtained from the description, then use the clean_salary helper defined in utils.ipynb to fill the empty cells. Average hourly salary and yearly salary are calculated from the min and max salary columns. The statistics are described in Table 1 data description of average salary. It's clear that some outliers exist, there should be no hourly salary of $140,000, nor yearly salary of 67.5. A closer look at the outliers shows that there are typos in several entries where hr and yr are misplaced. After these are fixed, the statistics become normal (Table 2 data description of average salary after cleaning). Many entries are empty in both salary and description of the scraped data, there is no way to get this information, so they are left to 'None' data type.

Table 1 data description of average salary

	Average hourly salary	Average yearly salary
count	90	344
mean	1621.71	137006.06
std	14750.69	51086.22
min	12.00	67.50
25%	37.56	100348.00
50%	50.21	132250.00
75%	64.77	169150.00
max	140000.00	300000.00

Table 2 data description of average salary after cleaning

	Average hourly salary	Average yearly salary
count	89	343
mean	53.45	137813.09
std	22.87	50071.85
min	17.00	35040.00
25%	37.72	100918.75
50%	50.43	132500.00
75%	65.00	169150.00
max	130.00	300000.00

Skills Cleaning

Due to the inconsistency between responses, the skill list obtained from ChatGPT often have different names for the same skill, such as SQL vs Advanced SQL, analytics vs analytical, Excel, MS excel or Microsoft Excel. Standardize thousands of inconsistent strings is super labor intensive and challenging in the old times. The advancement of machine learning techniques have made this much easier. Still some tuning and testing are needed to find out the best solution for this problem. In this project, I tested three methods to standardize the skill names: word count, KMeans-clustering and Fuzzymatch. Before applying these methods, the strings are first standardized by converting them to lower cases and removing all special characters.

Word count

Word count is a classic method to deal with inconsistent strings. The phases are break down to single words, counted and displayed. Common words with minimal information are removed, such as ‘skill’, ‘skills’, ‘tools’ and ‘etal’. It’s obvious that some words have similar counts and recombine words like ‘machine’ with ‘learning’, ‘power’ with ‘bi’.

KMeans-clustering

KMeans-clustering group similar elements together by calculating their distance in a map. It can not be used on words directed so the words are first vectorized using TfidfVectorizer library in python, then fitted by KMeans model from SKlearn library, the result is a cluster index label for each skill. The model is tuned by testing nine values of clusters, ranging from the total unique values of the skills to a 10^{th} of that value. The most frequent word in each cluster is used as the representatives in the final top skill list. One problem arising from this test is some simple words like ‘r’ can be randomly grouped into many clusters. For example, ‘r’ is the most frequent word in a cluster, however there are 100 words in it and ‘r’

only counts for 10 but counted as 100 when counting the skill frequency. So, it end up have ridiculous high count in the top skill list. Two adjustments were added to the algorithm to minimize the effect: 1. Context specific qualifiers are added to those simple words like ‘r language’ or ‘C++ language’; 2. A threshold is added to the clusters when counting skill names, only count the skill name when it consists of more than 50% of the cluster.

Fuzzymatch

This method utilizes the Fuzzywuzzy library. A fuzzy match compares words with similarity defined by a threshold. An 80% threshold is used in this project.

Cleaning Result

The cleaning result is shown in Figure 1 Comparison of Three Cleaning Method All three methods generated similar skill list, although the order can be slightly different. Among the three methods, Fuzzymatch works the best with minimum human input while the other two need close examination of the skill list for data preparation and fine tuning. It also loses the least amount of data during the process.

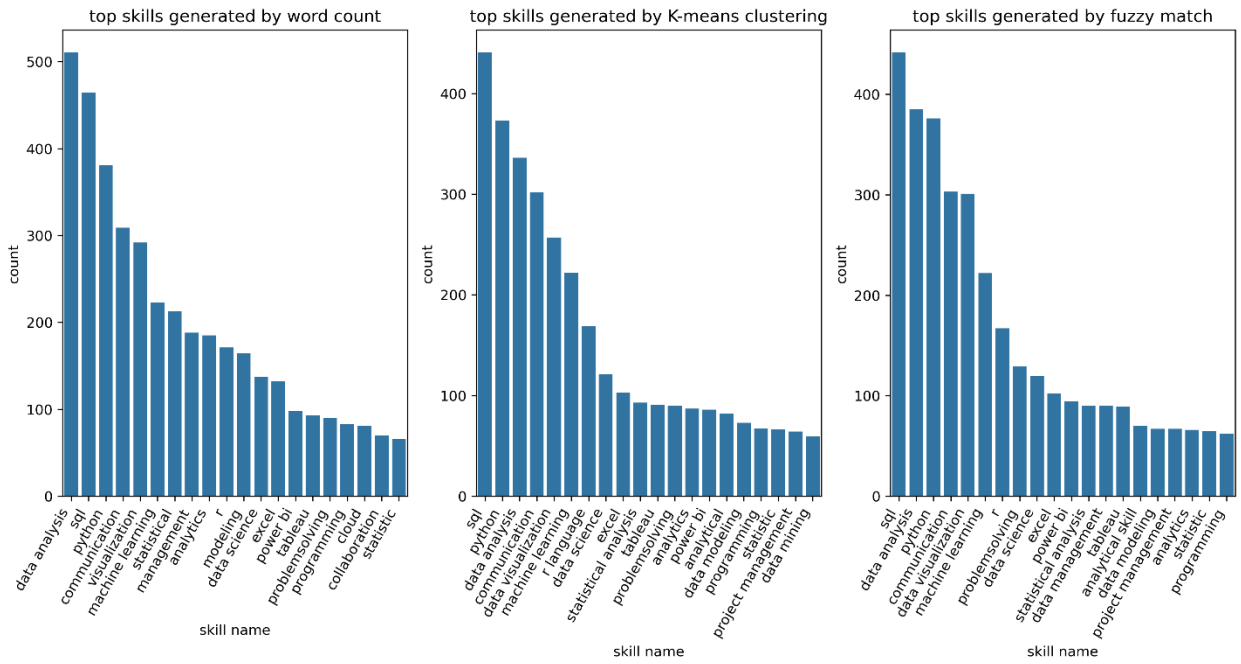


Figure 1 Comparison of Three Cleaning Method

Location Name Cleaning

I faced similar problems with the location data as in the skill cleaning. Many location entries are the same places with different name, like ‘New York, NY’, ‘NY, United State’,

‘New York Metropolitan Area’ and ‘New York City Metropolitan Area’. However, the same techniques produce much worse results. It’s probably due to the long words like ‘Metropolitan Area’ stops clustering and fuzzymatch to work. Fortunately, there are limited variation of the location names, I end up creating a map that mapping variations into one location name.

Suspicious Companies and Duplicates Cleaning

Many companies spam multiple postings for the job. Some are remote jobs created for different locations; some are shady companies try to increase their exposure to attract more applicants. There are two companies that stand out, one is Outlier which is a crowed sourcing company for AI training, the other is Synergistic IT which is not a coach company who will charge applicants for ‘training’, but write their postings like a paying job. These two companies post a lot of jobs at all times, so they need to be removed for any analysis work. In the meantime, other jobs with the same company, title and publish date are also removed except one copy of that job. In the end, 749 job postings were removed, leaving 766 data entries for further analysis.

Analysis

Posting Trend

After removing the non-legit companies and duplications, there are 766 postings left from the original 1515. The majority (>90%) are Data Analyst and Data Scientist jobs, the rest are intern and other Data related work, such as Business Intelligence Analyst. Data Analyst and Scientists opportunities are almost 50-50 (*Figure 2 Pie chart of composition of job titles*). Top company which posted the most are Intuit, Meta and Walmart (Figure 3 Top hiring companies). Figure 4 Job posting number trend and Figure 5 Job posting number by weekdays the job posting number trend over date and weekdays, respectively. There is only one week’s data so no meaningful information can be seen here. One interesting observation is that the postings peak on Tuesday and gradually decline to Monday. The reason is unknow, my theory is HRs need to prepare the postings on Monday, and start to post them start from Tuesday.

Figure 2 Pie chart of composition of job titles

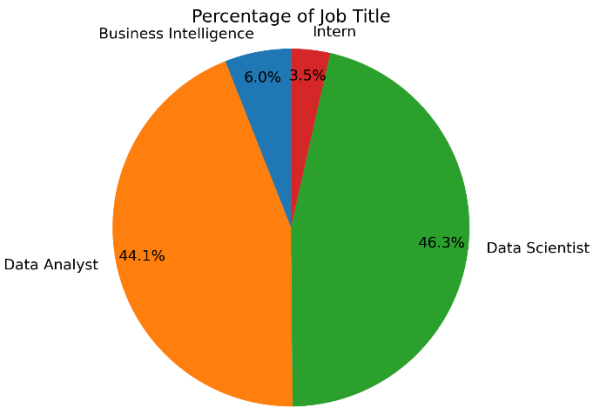


Figure 3 Top hiring companies

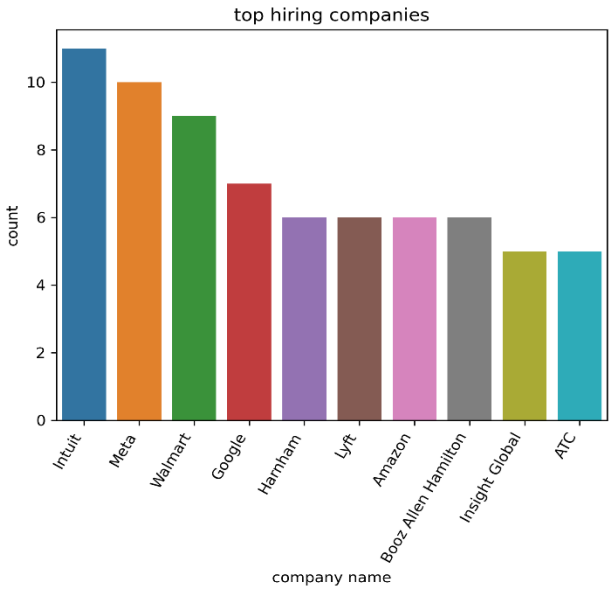


Figure 4 Job posting number trend

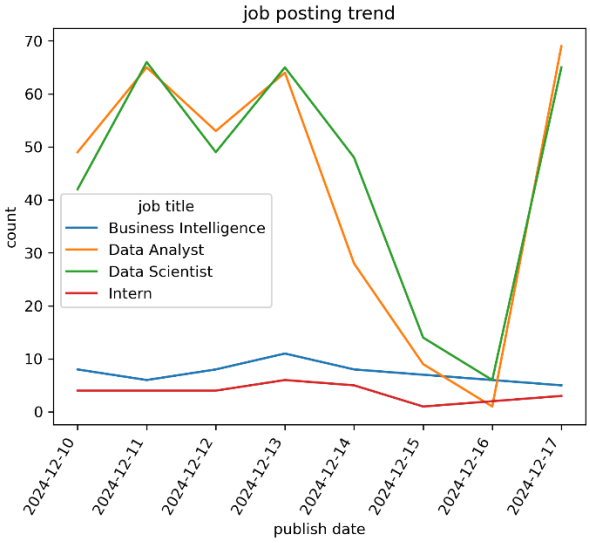
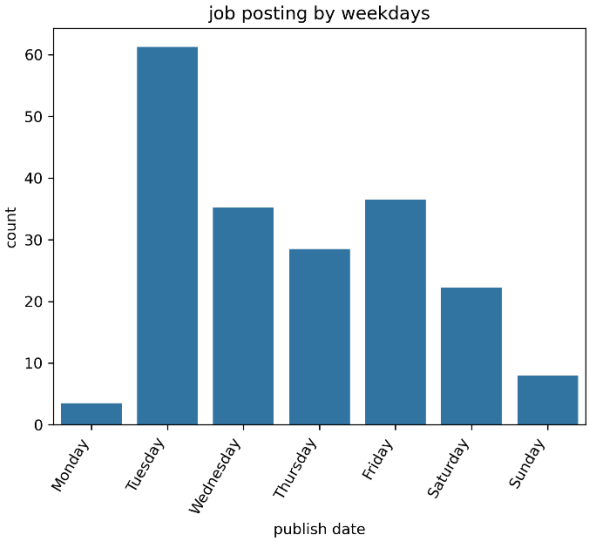


Figure 5 Job posting number by weekdays



Location

As shown in *Figure 6 Remote job percentage* about 39% of the postings are remote job so they don't have a exact location. After location names are cleaned, the postings are grouped based on names and counted. The result is plotted in *Figure 7 Top locations for data jobs* It can be seemed that NYC hosts the most data jobs, far above the rest of cities. San Fransisco is the close second if combined with Mountain View CA which is also near Bay Area. Washington DC and Atlanta holding the third place with above the same number of jobs. Chicago is the only city with decent data job postings in the central US.

Figure 6 Remote job percentage

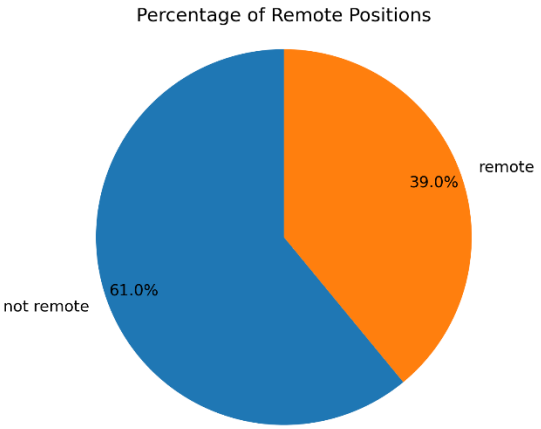
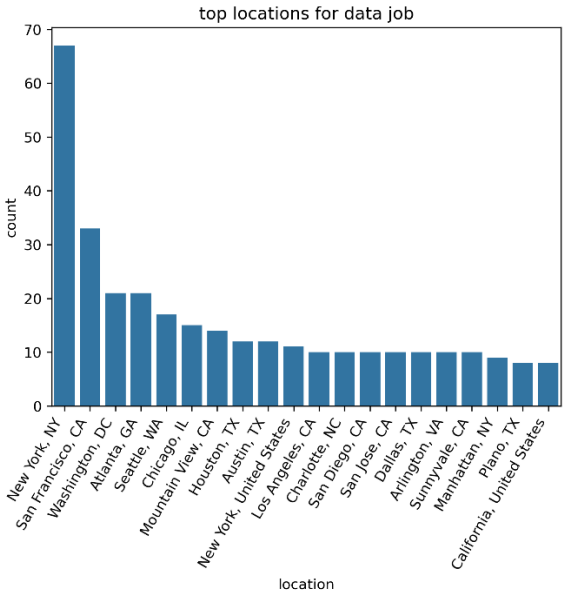


Figure 7 Top locations for data jobs



Experience Requirements

Most data jobs require some level of experience, even for entry level positions. This requirement makes the data field somewhat hard to break in. Figure 8 Experience level and required years of experienceshows the histogram of experience level and years distribution. As we can see, mid-senior level is the most in demand level, followed by entry level. There is almost no job has no experience requirement; most require 1-5 years of experience in the industry. 3 years and 5 years are the most common requirements.

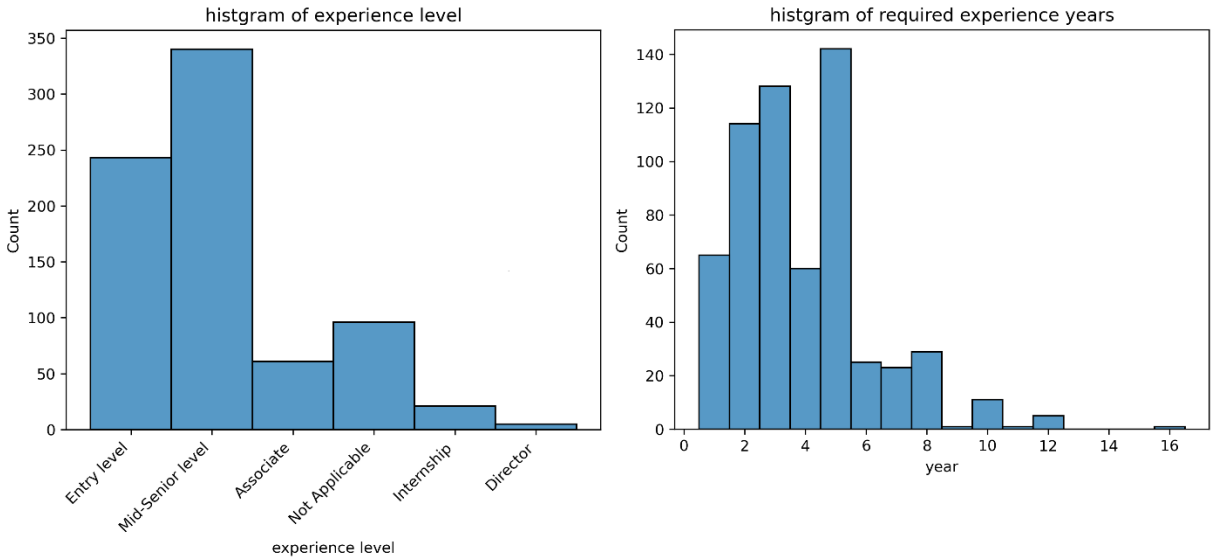


Figure 8 Experience level and required years of experience

Salary Analysis

Table 3 Salary statistics by job title

	Intern	Data Analyst	Data Scientist	Business Intelligence
Count	3	124	200	16
Mean	84680	108173	160419	94907
Std	48522	39143	45263	22388
Min	35040	40000	63000	66500
25% percentile	61020	80000	127010	77625
50% percentile	87000	105000	159150	90225
75% percentile	109500	126625	188625	109375

Max	132000	300000	300000	140000

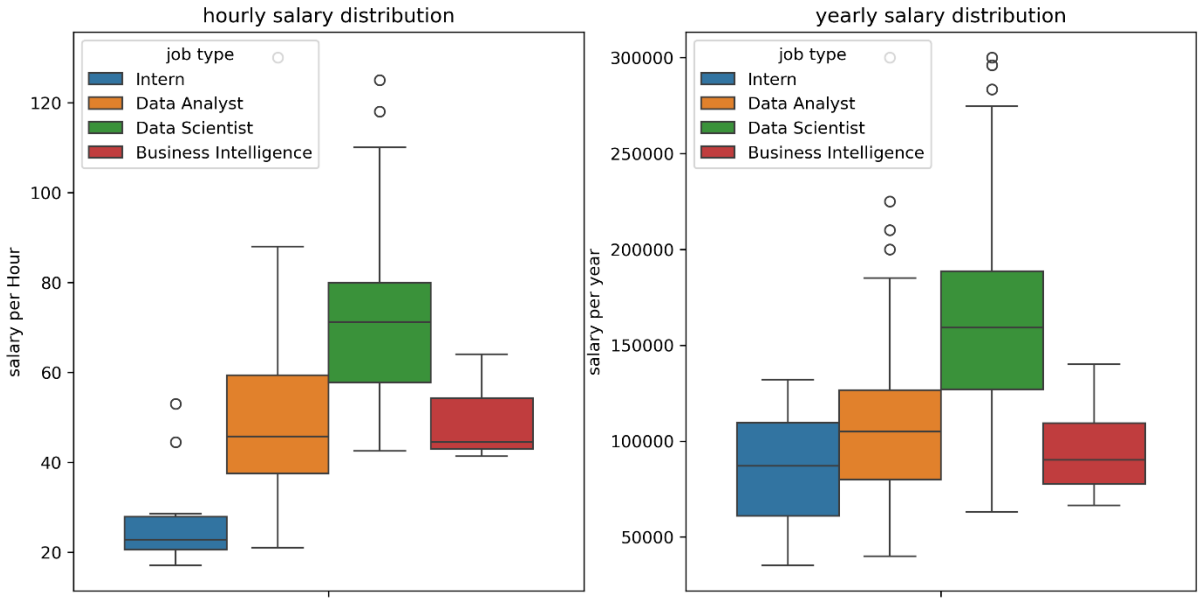


Figure 9 Boxplot of hourly and yearly salary by job title

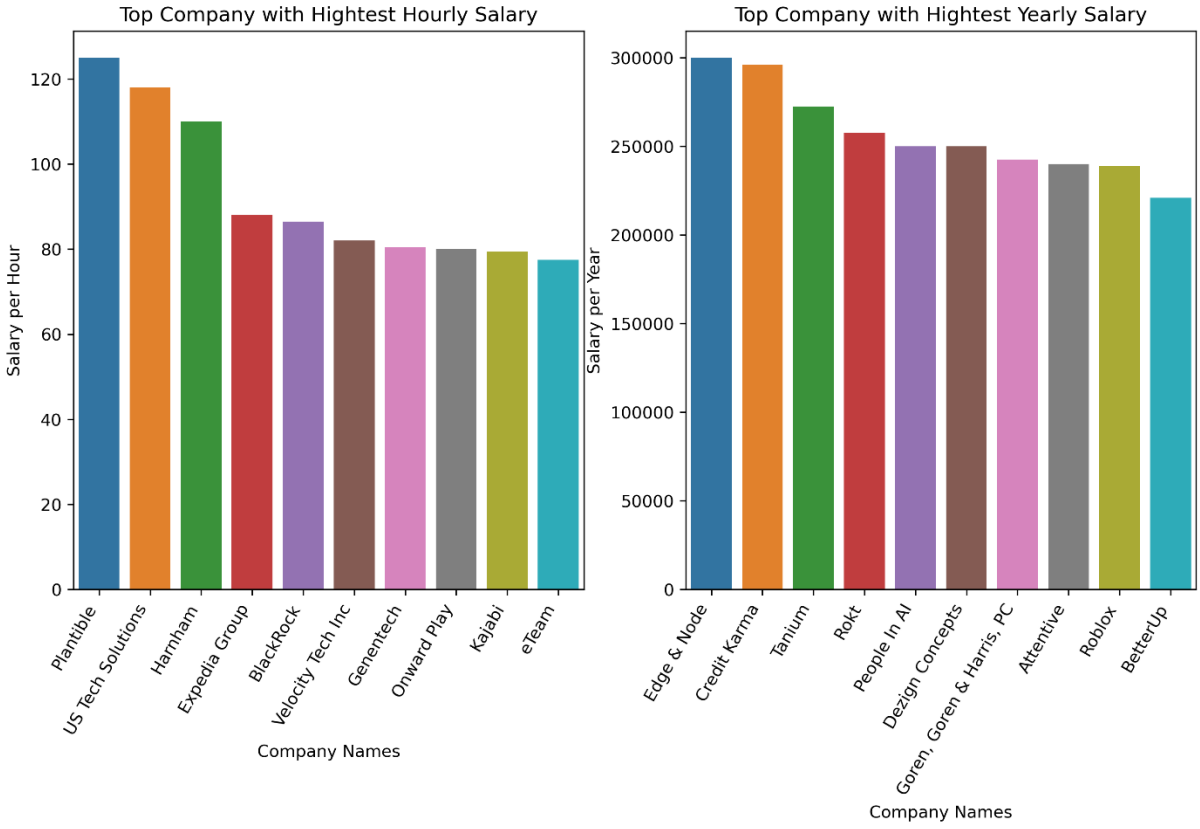


Figure 10 Companies paying the highest salary

Only 432 job postings out of the 766 included salary information, 89 are hourly rate due to the contract nature of the work. The rest are yearly salaries. The distribution is plotted in Figure 9 Boxplot of hourly and yearly salary by job title the numbers are shown in Table 3 Salary statistics by job title. It's clear that the Data Scientist jobs pay the most with an average salary of $160K and max of $300K. Data Analyst and Business Intelligence Analyst have similar salaries of around $100K. However, the highest paid Data Analyst job also pays about $300K. The top paying companies are shown in Figure 10 Companies paying the highest salary for reference.

Top Skills

All three methods mentioned in the data preprocessing section provide similar results. Since the fuzzy match needs the least human input, and less computation power than KMeans, it' s chosen for this analysis. The top skills are listed in Figure 11 top skills by job title The overall skill list is about the same, all jobs have skills like data analysis, SQL, communication, data visualization and Python on the top. However, the importance is slightly different for different titles. Analyst jobs emphasize more data analysis and visualization, while Data Scientists need more advanced modeling techniques like Python and Machine Learning.

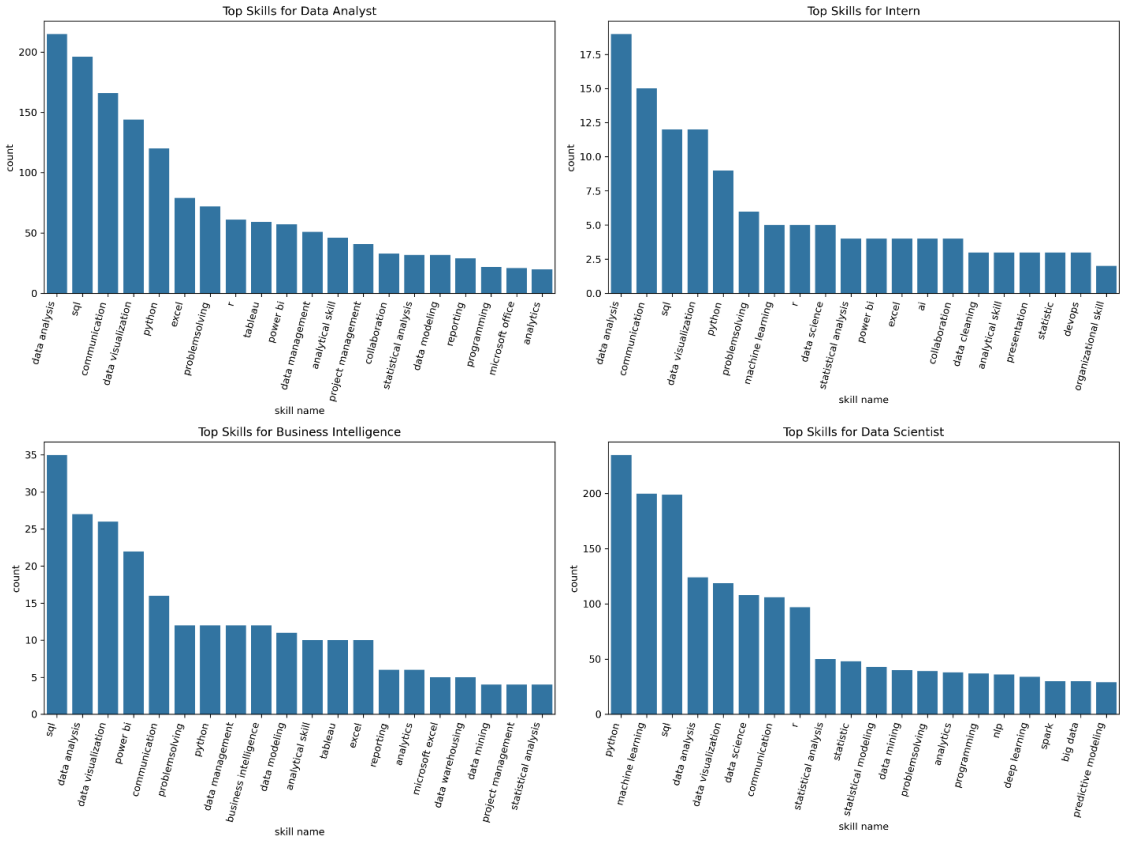


Figure 11 top skills by job title

Conclusion

This project provides valuable insights into the trends in the job market, including job opportunities, locations, salary expectations, and skill to learn. This information will help tremendously in my job search. The key take aways are:

1. Nearly a thousand unique job posted in a near holiday week
2. 39% are remote opportunities, NYC and Bay Area have the best chance of a on-site job.
3. Data Scientist positions provide decent pay, which is about 60% more than Analysts.
4. The most in demand experience level are mid-senior level, and then entry level. Companies usually require 1-5 years of experience.
5. Top skills are Data Analysis, SQL, Python, Communication, and Data Visualization using Tableau and Power BI. If I want to apply for Data Scientist positions, Machine Learning is also a top skill.

Developing advanced AI tools like ChatGPT, unsupervised machine learning tool like KMeans-clutering, and word matching algorithm like fuzzywuzzy are proven to be valuable

assets. They can quickly convert messy web scraped information into insights of the job market to speed up job applications tremendously.

Future Work

This project only analyzed a week's job posting on Linkedin. To understand the long-term trend of the industry, more data is needed. I will keep collecting job posting data while searching and applying for jobs.

I also learned a lot in the project by the healthy cycle of encountering problems, to search and test solutions, and eventually solve the problem. Learning is a continuous process with no end, I will keep exploring more advanced AI and language processing tools to improve this project, both for demonstration my data analysis skills and speeding up my job searching.

Finally, after talking with the administrations of nearby universities, they showed some interest in providing funding for long term data service. I will collect and compile job posting information for their students to use. In the future, this project will not just include data-related work, more field of study like Finance, Accounting or Chemistry will be added. User feedback will also be collected to further improve the quality and usability of this project.