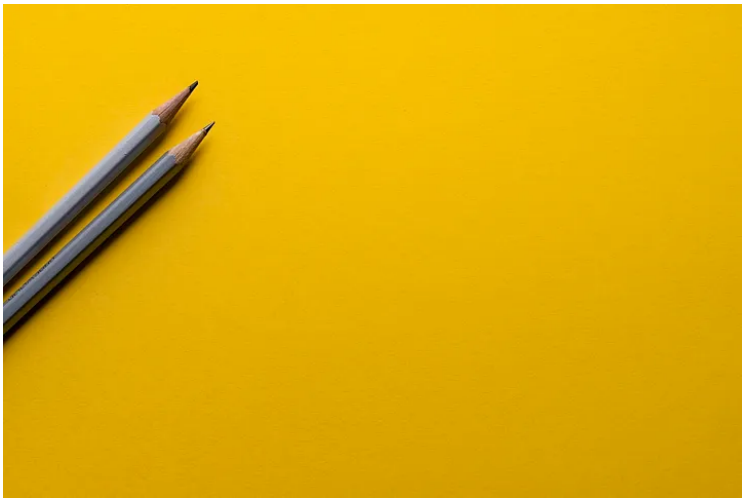


Image from Unsplash

Data Science Job Market Analysis

Ai Zhong · Follow
13 min read · Aug 23, 2021

Project done by: Laiya Lubben, Rachel Wyatt, Ai Zhong

> This is our capstone project for Master of Applied Data Science(MADS) at University of Michigan, with a time span of 2-month.
>
> *Our application: https://capstone-dashboard-app.herokuapp.com/*
>
> *Our github: https://github.com/rachwyatt/capstone-project*

1. Introduction

The goal of this project is to present an analysis of the recent data science related job market. As we approach the end of our MADS degree, we want to utilize our learned skills to help us get a better understanding of the current requirements on the market, meanwhile help everyone prepare themselves better for their dream position.

During the search for data scientist related jobs, we found there are a large range of different directions and position titles, which can be confusing to people fresh to this industry. For example, what are the different skills required for data scientist vs. machine learning engineer, data analyst vs. data engineer etc. Furthermore, different companies may have different position titles for similar functionality. In our view, instead of focusing on position titles, we should focus on required skill sets for different industries/job functions, as they are the true key points employers want to evaluate. Furthermore, we will be utilizing the power of machine learning to cluster job postings and study corresponding skills to explore if there are interesting patterns.

In summary, we scraped data science related job data from major job boards, cleaned them up for modeling, statistical analysis, and visualization. The main focus of our analysis is to find the popular skills required in different companies and domains. Many job sites exist where job posts can be searched using a specific skill as a keyword, but this project's goal is to provide insights about the most in demand skills across various data science domains. Therefore, our analysis can help people understand more about what to expect for a certain domain's data science position.

2. Data

2.1 Data Source Research

In the beginning of our project, we have listed our desired job boards for data scraping, as well as a list of potential APIs suggested by our instruction team, however, after some initial research, most APIs have very limited access on data downloading for free, also, simple requests scripts does not work on popular job boards as they do not allow scraping. The major researched job boards include:

- *Indeed*: API only available for people with 'publisher ID', which requires application, and your site must generate over 10,000 legitimate unique page views per day to qualify. Scraping not allowed with requests.
- *LinkedIn*: No open API available, only for partnership websites.
- *Unicorn hunt*: No open API available; Scraping not allowed with requests;
- *LinkUp*: No open API available; Scraping not allowed with requests;
- *Hackerrank*: No open API available, need to register for 'work account', which requires application.
- *Stackoverflow*: No open API available;
- *Adzuna*: Free API available, but with very limited rate (250 hits/day).
- ***Glassdoor***: No open API available; Scrapable with selenium.

Furthermore, we found some existing free data from the data world website (https://data.world). However, as our focus is data science related positions, after filtering we are left with around 10,000 jobs from various files.

2.2 Data Scraping and Cleaning

In the end, we developed a selenium scraper with selenium to get our main data with enough details. The detailed codes can be found on our github repository.

We have collected a list of data science related job titles for the keyword variable when making requests, in order to reduce the range of scraping.

From raw data, the parsed data fields are: *job_title, company_name, post_date, raw_job_description, job_type, and location*; After getting *raw_job_description*, further processing is needed to extract more details and store them systematically, because each job post have different description styles on the skills they require, as well as the education, experience etc. With the help of regular expression, we were able to extract skills and education requirements for each job description.

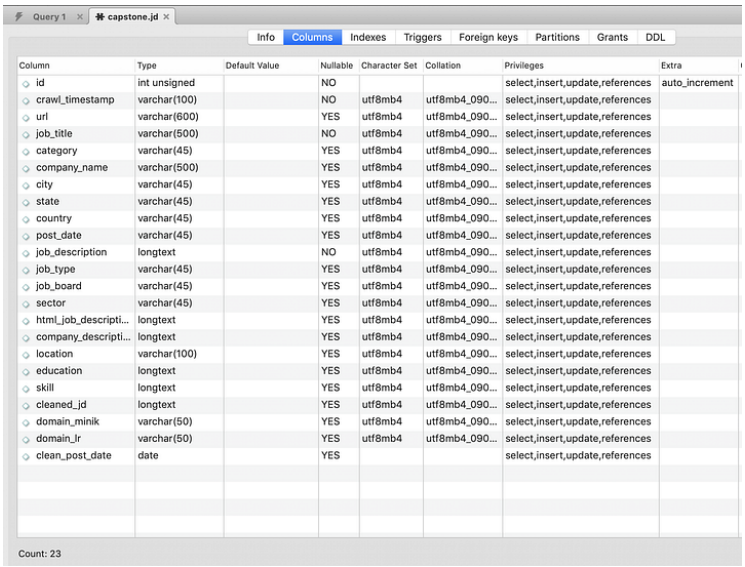
To prepare the data for model training, further cleaning is needed. Job descriptions often contain more than just the text that describes the job content, but also some disclaimers from the company about their policies and benefits etc. Such statements are just noise for our unsupervised models when doing clustering to find potential domains for the job description. Our initial model training shows that with max frequency from 0.5 to 0.6 in the tfidf/countvectorizer model, top words like 'work', 'experience' etc. still appeared in the LDA (or k-means) model's top keywords in some clusters. More details will be illustrated in the following section. The irrelevant sections are then removed with the help of regex.

2.3 Data Storage and Retrieval

As we have data retrieved from different sources with different data formats, and they will finally be processed to a uniform format, it's not ideal to save them as separate files on clouds. We have chosen to use the Amazon RDS (relational database service) as our data storage solution, which can be set up on your AWS.

With mysql RDS set up, we are able to connect the database from python code using packages like pymysql. With this, we can insert, retrieve, and modify data efficiently. Furthermore, we can easily manage duplicates by setting the job_url column to be unique, so when a new job description is being scraped and inserted, the database can automatically check duplicates with all existing data in the database efficiently, if duplicates are found, the insertion is aborted.

The major schema of the job description table is shown below:


Query 1 | capstone.jd

Info | Columns | Indexes | Triggers | Foreign keys | Partitions | Grants | DDL

Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra
id	int unsigned		NO			select,insert,update,references	auto_increment
crawl_timestamp	varchar(100)		NO	utf8mb4	utf8mb4_090...	select,insert,update,references	
url	varchar(600)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
job_title	varchar(500)		NO	utf8mb4	utf8mb4_090...	select,insert,update,references	
category	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
company_name	varchar(500)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
city	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
state	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
country	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
post_date	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
job_description	longtext		NO	utf8mb4	utf8mb4_090...	select,insert,update,references	
job_type	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
job_board	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
sector	varchar(45)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
html_job_descripti...	longtext		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
company_descripti...	longtext		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
location	varchar(100)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
education	longtext		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
skill	longtext		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
cleaned_jd	longtext		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
domain_minik	varchar(50)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
domain_lr	varchar(50)		YES	utf8mb4	utf8mb4_090...	select,insert,update,references	
clean_post_date	date		YES			select,insert,update,references	

Count: 23

The main challenge for us is to set up the RDS properly in AWS as we are new to the process. The official documentation does a good job on explaining the major steps, but there are some details not covered, especially on the inbound and outbound rule settings of security groups. Specifically, the public access should be turned on, and the traffic setting for inbound/outbound should be 'all traffic'. A useful video for reference can be found here.

After the database is set, you can access with pymysql in python as mentioned above, or you can access through some user interface such as

mysql workbench or sequel pro.

3. Modeling

The main objective of training models is to uncover domain information for the job postings data we gathered. One assumption we have at the beginning of this project is that job postings from the same domain should include similar terms in the job descriptions. For that reason, we trained unsupervised learning models to cluster job postings based on job descriptions.

The two primary methods used to vectorize job descriptions for the models:

- **CountVectorizer** — transforms text into a vector of the frequency of each token (TF), which is also known as the bag of words approach.
- **TfidfVectorizer** — transforms text into a term frequency-inverse document frequency vector (TF-IDF) that weights each token frequency by the number of documents that token appears in.

3.1 Unsupervised Learning Model

The one unsupervised learning model we trained is the Mini-Batch K-Means model, which saves a substantial amount of computational time in comparison to the K-Means model. Unlike K-Means models, the Mini-Batch K-Means algorithm only uses small random batches of data at each iteration to update the clusters until convergence. Similar to K-Means models, it is easy to implement and has minimal parameters to tune. The one important parameter required is the number of clusters (k). To find the best number of clusters to train our model, we used the "elbow" method and the average silhouette plot.

One way to find the optimal clusters is by finding the "elbow" point using the SSE plot. SSE is the sum of the squared distance between centroid and each member of the cluster. As the number of clusters increases, SSE decreases and the "elbow point" is the point where SSE starts decreasing in a linear manner. The SSE plot (Fig. A) we generated for the Mini-Batch K-Means models shows fourteen is the optimal number of clusters.

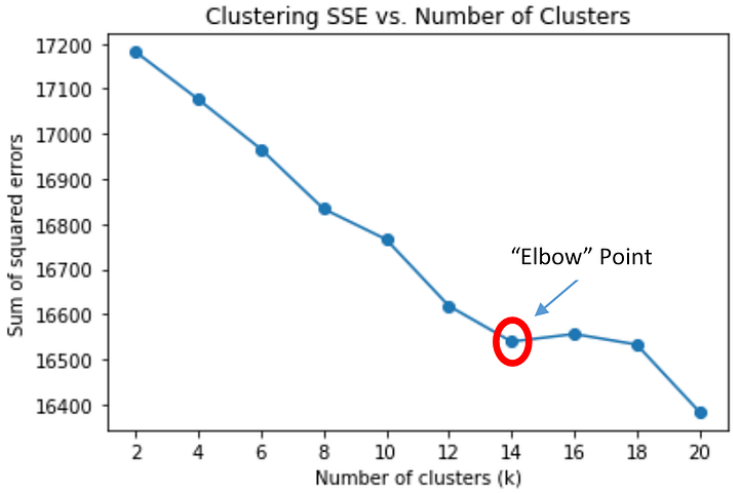


Figure A: SSE plot generated from Mini-Batch K-Means model

The second method we used to evaluate cluster quality is using the average silhouette plot. Silhouette score measures how close each point in one cluster is to points in the neighboring clusters. Data points with silhouette scores close to one indicate the data point is far away from the neighboring clusters; silhouette scores close to zero indicate the data point is on or very close to the decision boundary between two neighboring clusters; negative silhouette scores indicate the data point might have been assigned to the wrong cluster.

The average silhouette plot (Fig. B) generated with the Mini-Batch K-Means model is also showing the same optimal clusters as the SSE plot (Fig. A).

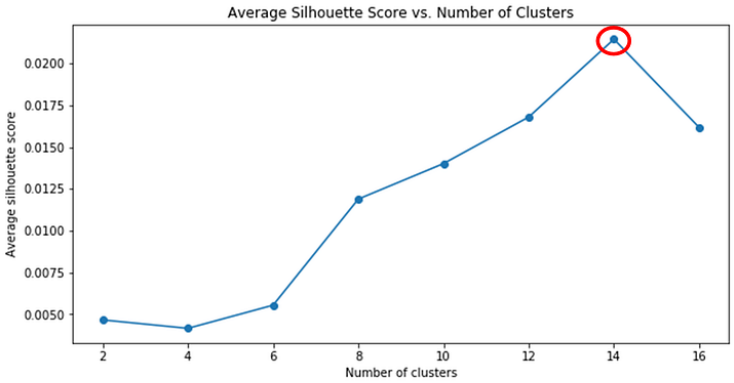


Figure B: Average silhouette scores generated by Mini-Batch K-Means model

Based on the result from the two evaluation methods above, we decided to generate fourteen clusters with the Mini-Batch K-Means model. In order to label each cluster into a specific domain, we plotted a heatmap of the top ten terms for each cluster (Fig. C). The color is ranged from light yellow (low values) to blue (high values), and it is based on the weights of the terms in the given cluster. We labeled the cluster by looking at the distinct terms for each cluster. For example, “clinical”, “patients”, and “hospital” are terms that only belong to cluster 0 so we label the cluster as “healthcare”.

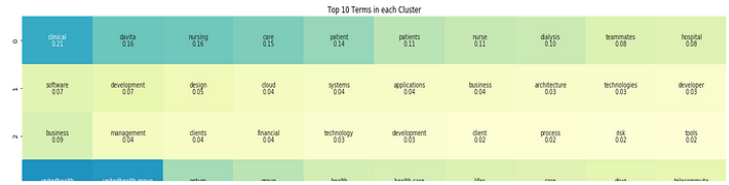


Figure C: Top 10 terms for cluster 0, 1, 2 generated by Mini-Batch K-Means with k = 14

The code for the Mini-Batch K-Means model can be found in the *model_notebooks* folder from our github repository. We also tried other unsupervised learning models such as Gaussian Mixture model and topic models (LDA, NMF). Although we are not including the result from the other unsupervised learning models in this blog, you can still check out the model notebooks in our repository.

3.2 Supervised Learning Model

In addition to the unsupervised learning models, we extracted companies' industry information to use as labels from a [companies dataset] (https://www.kaggle.com/peopledatalabssf/free-7-million-company-dataset) found in Kaggle. Since there are over 100 different industries in the companies dataset, we decided to condense the list of industries into a smaller set of domains. For example, industries such as banking, insurance, financial services, investment management are all categorized as "finance". After we categorized each industry into a specific domain, we ended up with twenty-three different domains. Then we merged the companies dataset with the job postings dataset by company's name and ended up with 10,399 labeled data to use as our training set for supervised learning models.

The distribution of domains within our labeled dataset (Fig. D) shows that our labeled dataset is highly imbalanced. To ensure that both the training set and the test set contain approximately the same percentage of samples of each target domain, we enabled the parameter "*stratify*" from the *train_test_split* function.

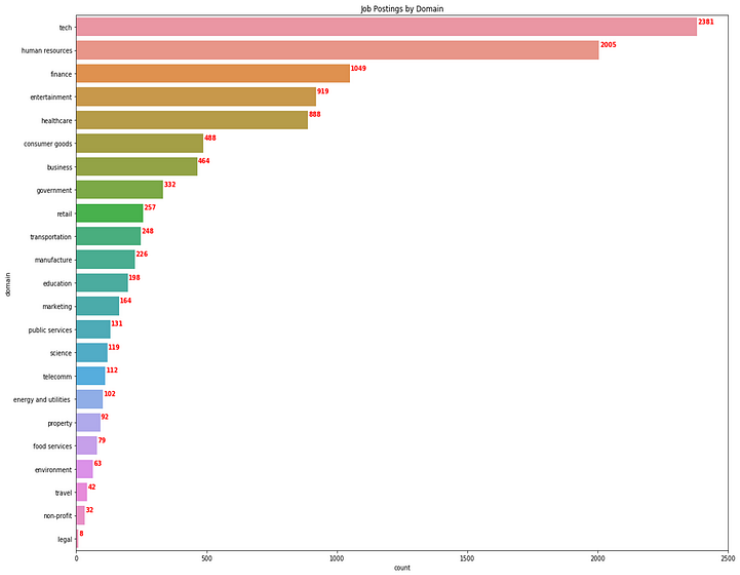


Figure D: Domain distribution of the labeled job dataset

One of the challenges of training supervised learning models is selecting the right model. With the limited time we had, we decided to only compare model performance on the four commonly used models for classification tasks. The four models are Multinomial Naive Bayes (to establish a baseline), Logistic Regression (have the option to add regularization to prevent overfitting), K-Nearest Neighbors (easy to implement and understand), and Random Forest Classifier (generally perform better than other classifiers and can handle dataset with high dimensionality). Since the labeled dataset is highly imbalanced, we used F-1 score to evaluate our models instead of accuracy. F-1 score is the harmonic mean of precision and recall. It is a measure that balances how precise the classifier is, as well as how robust the classifier is. The table below shows that the Random Forest Classifier outperformed the other three models with TF-IDF and the Logistic Regression performed the best with TF.

Models	F-1 Score with TF-IDF	F-1 Score with TF
Multinomial Naive Bayes	0.482	0.658
Logistic Regression	0.659	0.719
K-Nearest Neighbors	0.672	0.523
Random Forest Classifier	0.695	0.702

Based on the result, we used *GridSearchCV* from the sklearn library to tune the model parameters for both the Logistic Regression Model and the Random Forest Classifier. Here is a snippet of the code we used for Logistic Regression:

```
# set up the model and variables for training
model = LogisticRegression()

# define grid search for tuning the parameters
grid_params = {
                'solver':['newton-cg', 'sag', 'lbfgs'],
                'C':[100, 10, 1.0, 0.1]
               }
cv = StratifiedKFold(random_state=RANDOM_SEED)
grid_search = GridSearchCV(model, grid_params, cv=cv, scoring='f1_weighted', n_jobs=-1, verbose=10)
grid_result = grid_search.fit(X_train_tf, y_train_tf)

y_pred = grid_search.predict(X_test_tf)
f1 = f1_score(y_test_tf, y_pred, average='weighted')
```

The final (best) trained model we used to classify our unlabeled dataset is a Logistic Regression model with parameters: C = 0.1 and solver = newton-cg. With this model, we were able to achieve a F-1 score of 0.7227. You can find details of the code from our github repository.

3.3 Unsupervised Learning Model vs. Supervised Learning Model

Since our labeling method for clusters from the Mini-Batch K-Means model is simply based on the top ten terms for each cluster and is a subjective process, we are interested to see whether the labels we chose for the clusters match with the predictions from the Logistic Regression model. To compare the two models, we analyzed the count of job postings in each of the domains from both classifiers (Fig. E).

The first difference we noticed is the list of domains. The Mini-Batch K-Means model only produced eight domains because many of the clusters included terms related to the same specific domains. On the other hand, there are twenty-three domains (labels) used in the Logistic Regression model. Based on the heatmap below, we found that many of the domains for the Mini-Batch K-Means model do not match with the domains predicted from the Logistic Regression model. For example, the domain "*education*" from the Mini-Batch K-Means model includes a large number of job postings from the domain "*human resources*" based on the Logistic Regression.

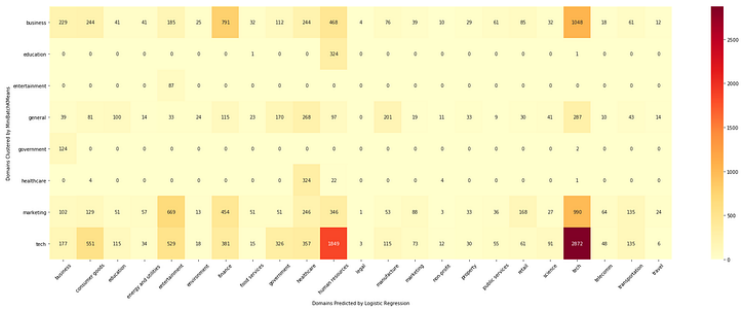
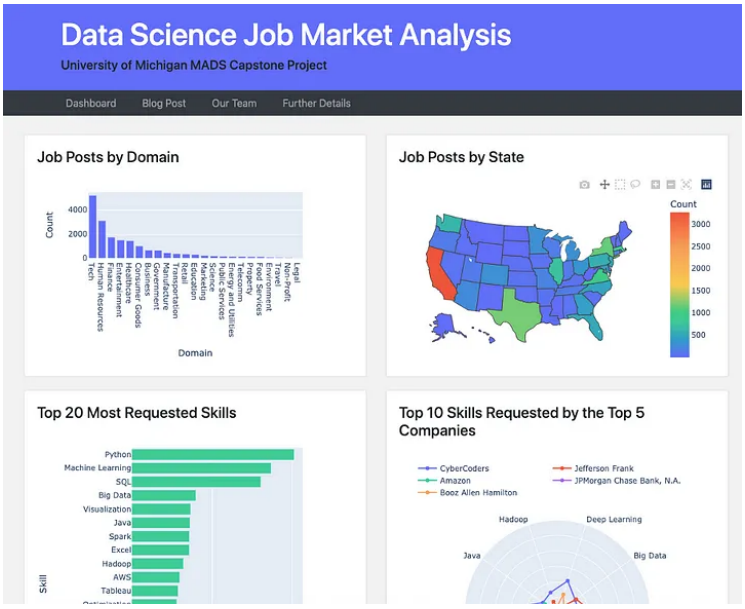


Figure E: Heatmap of MiniBatchKMeans domains and Logistic Regression domains

4. Data Dashboard



Our Dashboard

4.1 Technical Details

To display the results of our models and provide insights for those seeking jobs in data science, our team designed a dashboard using Plotly Dash. In order to provide context for our dashboard, including this blog post, information about our team, and additional project details (link), we built a web application using the Django framework and hosted it on Heroku. To avoid running two Heroku Dyno instances (one for the Dash app and one for the Django app), we used the django-plotly-dash library to run the Dash app within the Django app. Our dashboard also uses the dash-bootstrap-components library to improve the appearance and layout, especially when viewing on mobile devices.

Many tutorials exist for implementing the tools used to publish our project outcomes. Django Girls provides an excellent tutorial for getting started with Django, which was extremely helpful to us. This is also a great tutorial to recommend for anyone just getting started with Python. If you are interested in dabbling in web development as a way to showcase your data science projects, the Django framework is a good place to start because it uses Python, which many data scientists are already experienced with.

Also helpful to our team was a tutorial produced by Mozilla which includes instructions for deploying Django apps to Heroku. It goes into detail about managing environment variables to help you avoid publishing details such as your username and password for your database connection. Heroku is user friendly and integrates directly with GitHub, which is where our project code is stored.

4.2 Dashboard Insights

Exploring the dashboard provides users with many different types of insights to the data science job market. The logistic regression model results provide the default list of job domains, but the user can switch to using the mini-batch k-means model results to determine domains. The logistic regression results were chosen as the default because it provided a

larger selection of domains and exploring the different visualizations in the dashboard showcases a more even distribution of companies across domains.

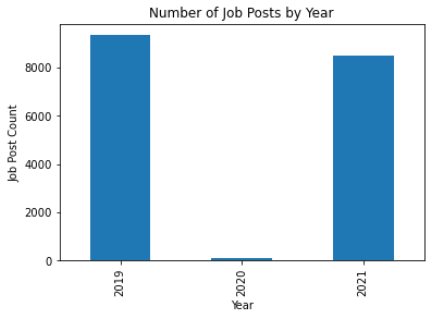
Some of the insights discoverable in our dashboard are not surprising. For example when filtering by domain, the map shows a large number of transportation jobs in Michigan and shows the top 5 companies for the education domain as being primarily major universities. When filtering by state, we see Apple show up in the top five companies in California and transportation is the second largest job domain in Michigan. Also, good news for MADS graduates, Python remains near the top of the list of most in demand data science skills with virtually any combination of filters.

We hope that this dashboard will provide our fellow graduating students with insights as to which additional skills to brush up on depending on what domain, region, or company they are interested in working for in the future.

5. Ethical Considerations

The most important caveat dashboard users should keep in mind is that the **data reflected is frozen in time**. Our data scraping provided a large data set to provide a broad overview of job postings from 2019–2021, but will not be continuously updated with realtime job posting data. Since the job market and most in demand skills are subject to change over time, the insights provided from the dashboard should be taken with a grain of salt. To ensure users are aware of this limitation, an alert appears at the top of the dashboard disclosing this information.

An additional limitation of our data is that there is a very uneven distribution of dates the jobs were posted. This is due to the fact that a large set of data was provided by one of our team members from a previous project and is a little older, and we then scraped for additional more recent data, leaving a large gap missing from the timeline of job postings.



Future possible work would be to find data sources or APIs that would allow the dashboard to be continuously updated to provide more of a steady stream of data and insights to changes in the job market over time.

6. Statement of Work

Our team fell into very natural roles for this project with each member being very well suited for the work they took on. All work is our own, and no outside assistance was received other than bits of advice and guidance from course instructors. Ai took on data collection, cleaning, and database

management, Laiya handled the modeling and performance evaluation, and Rachel developed the Django app and dashboard. All team members pitched in on each other's work as needed providing helpful feedback and suggestions and all worked together to write the content of this blog post, mainly focusing on the sections covering the areas we worked on. We all enjoyed working on this project together and hope our fellow students enjoy exploring our dashboard.

Data Science | Job Market | Machine Learning | Dashboard

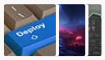
6 min read · Oct 27

--

✦ · 3 min read · Aug 11

-- 46

Lists



Predictive Modeling w/ Python

20 stories · 550 saves

Practical Guides to Machine Learning

10 stories · 629 saves

Natural Language Processing

780 stories · 356 saves

New_Reading_List

174 stories · 170 saves

Anmolgoel

How to Build a Data Science Portfolio

How do you get a job in data science? Knowing enough statistics, machine learning...

18 min read · Jun 7

-- 1

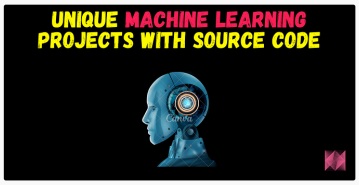
Rick Dronkers in Data to Value

Digital Analytics is Dead? Long Live Digital Analytics

What is the future of Digital Analytics? Is it dead? In this post I dive into the trends...

7 min read · Oct 26

-- 2



Abhishek Sharma

20+ Unique Machine Learning Projects with Source Code

Explore a Collection of 20+ Unique Machine Learning Projects with Source Code. Enhanc...

9 min read · Aug 25

--

Cassie Kozyrkov

What was the hardest thing about quitting my job at Google?

Advice for those who are afraid of leaping into the unknown

✦ · 10 min read · 4 days ago

-- 31

See more recommendations