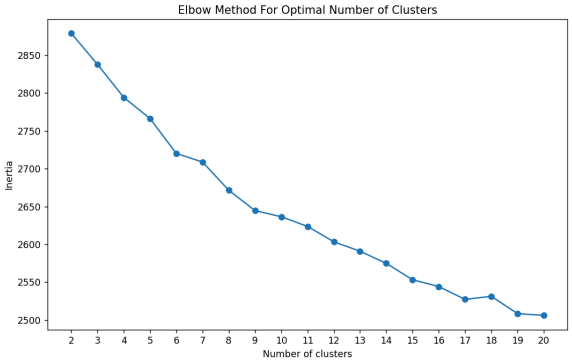
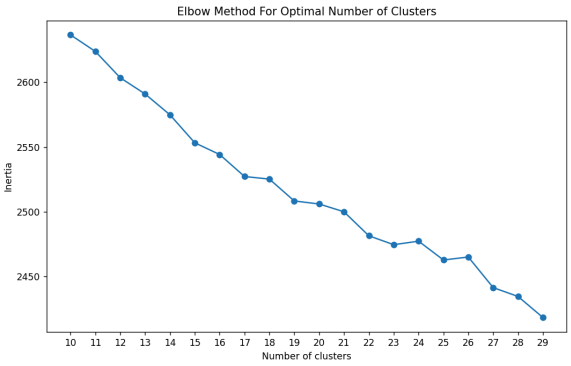
Applicant

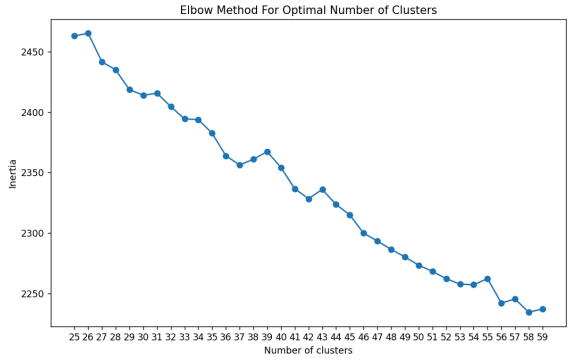
“Elbow Method”



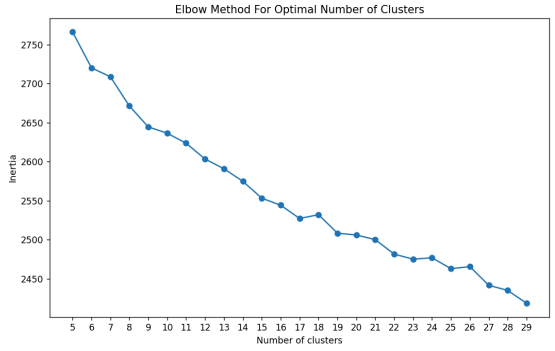
# of clusters: 2 to 20



# of clusters: 10 to 30

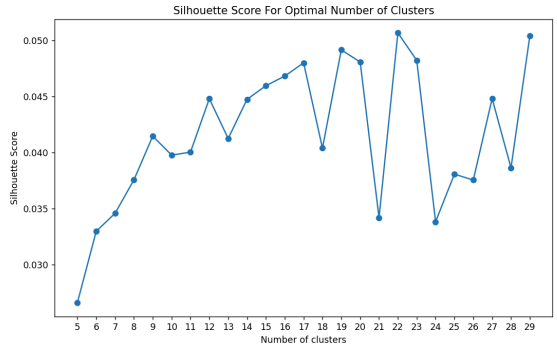


# of clusters: 30 to 60

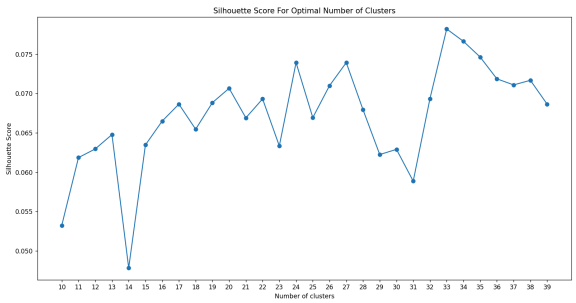


# of clusters: 5 to 30

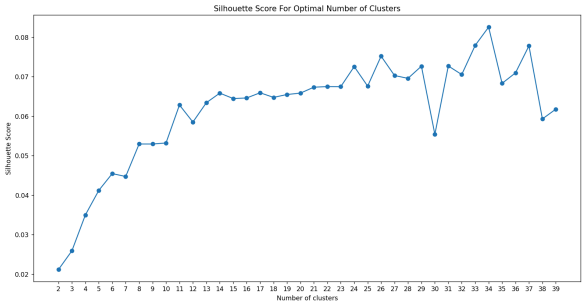
Silhouette Score



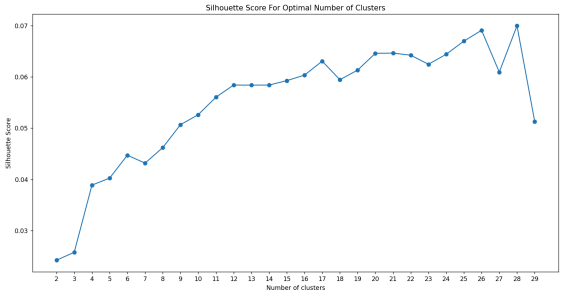
# of clusters: 5 to 30



New data, skills+job title

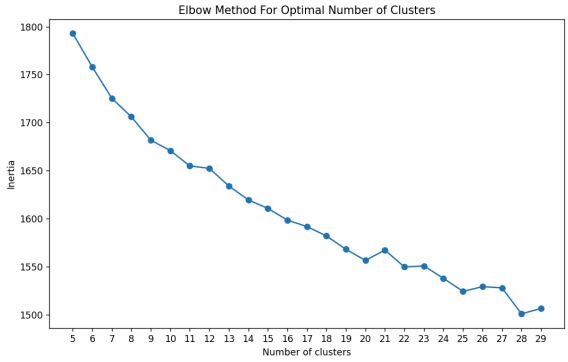


Clusters 2 to 40, new data, no desc

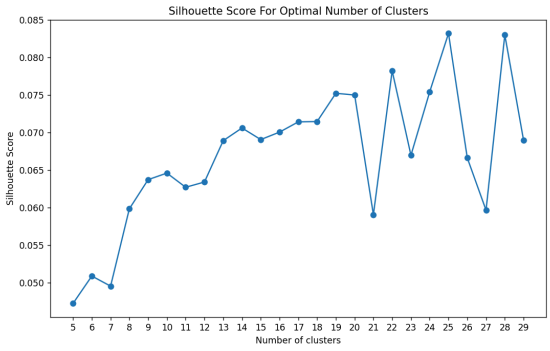


After changing vectorizer

No description in matrix



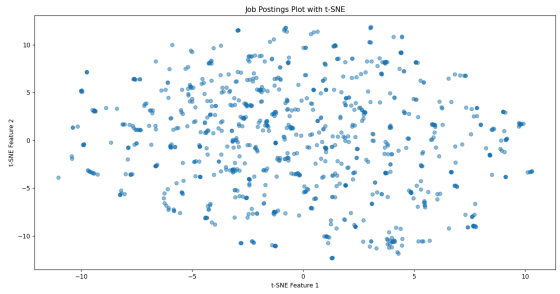
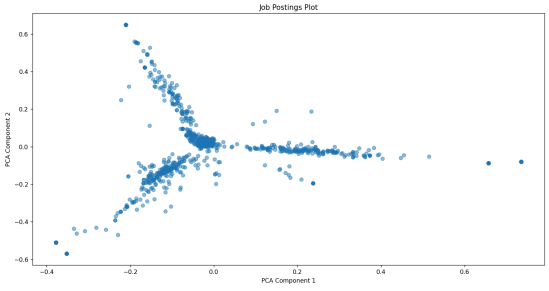
# of clusters: 5 to 30

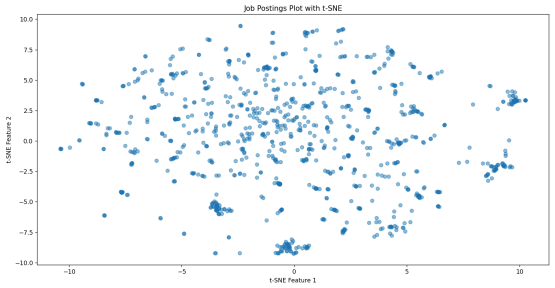


# of clusters: 5 to 30

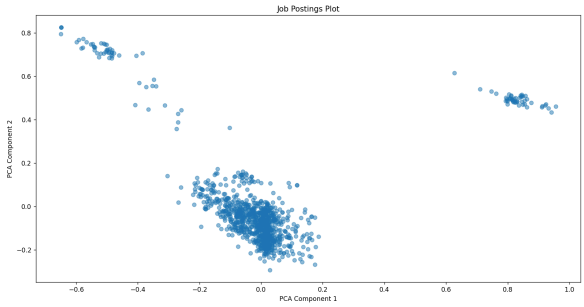
Based on the silhouette plot above the max score is at cluster 25 with a score of ~0.083

Plotting the jobs

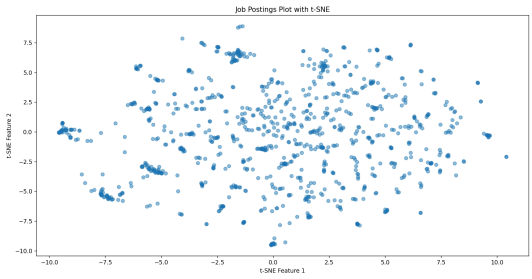
TSNE	PCA
	
Just job titles	Only job title, using PCA



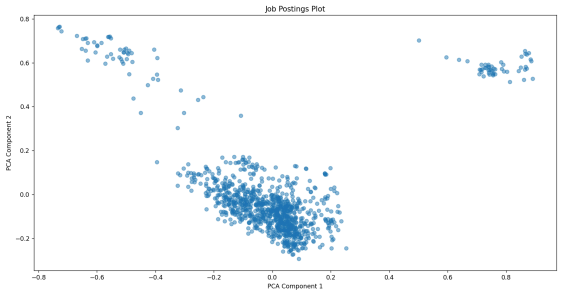
Job titles + skills



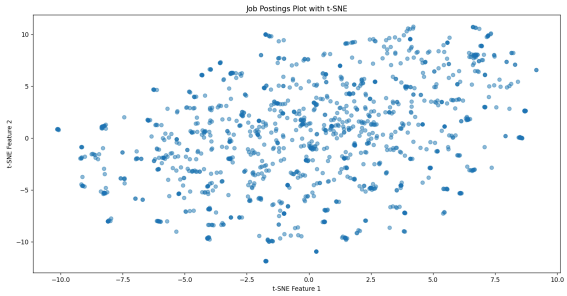
Job titles + skills, using PCA



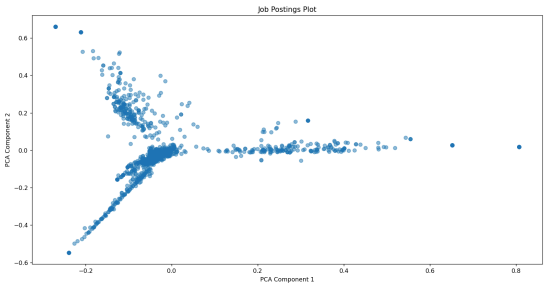
Job title + skills + desc



Job title + skills + desc, using PCA

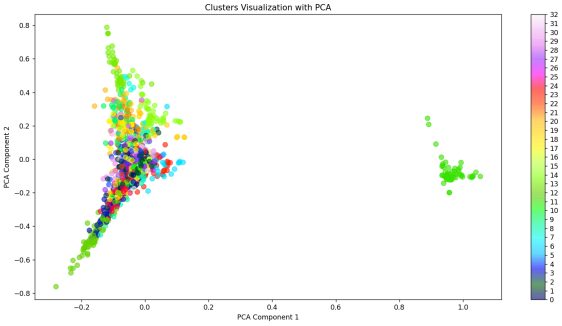


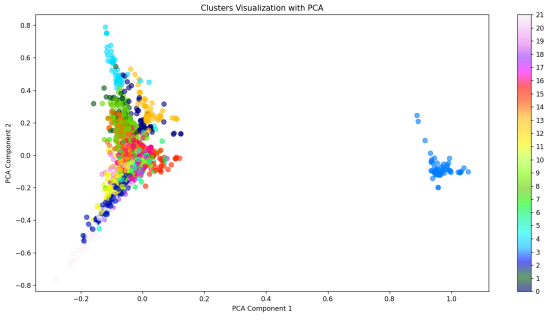
Added data, only title

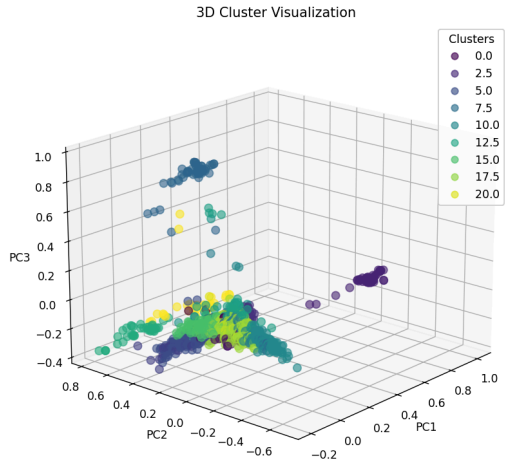
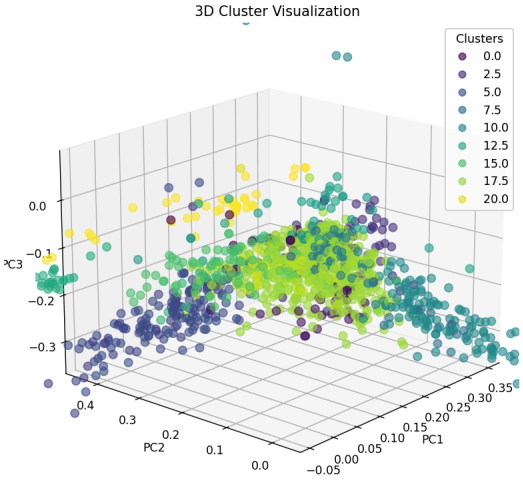


Added data, only title

Clustering

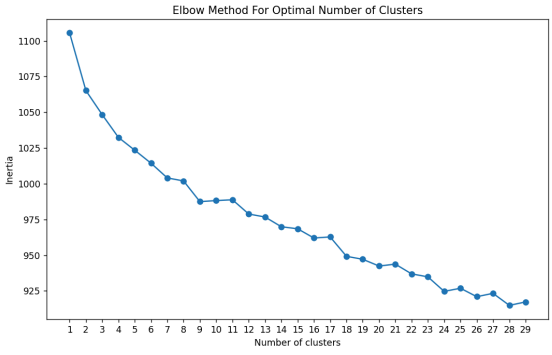




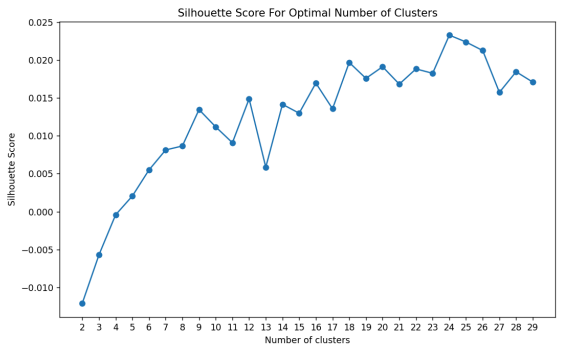
33 clusters	22 clusters
 22 clusters, no desc	 Same plot but zoomed in

Recruiter

“Elbow Method”

 # of clusters: 1 to 30	

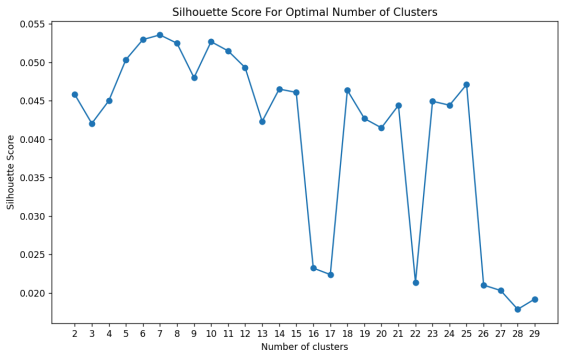
Silhouette Score



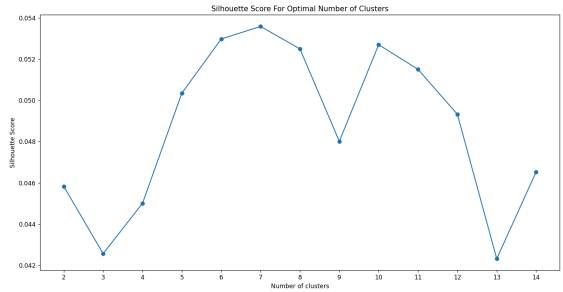
# of clusters: 1 to 30

Based on the plot above the max score is at cluster 24 but the score is around 0.025

No description in matrix



# of clusters: 1 to 30

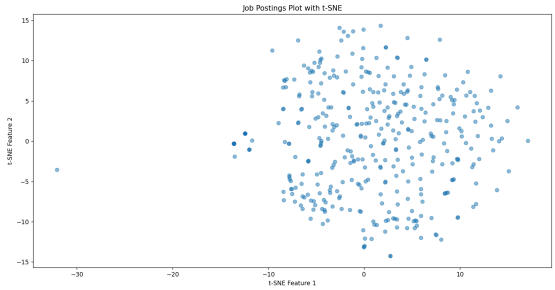


Same thing just a smaller range

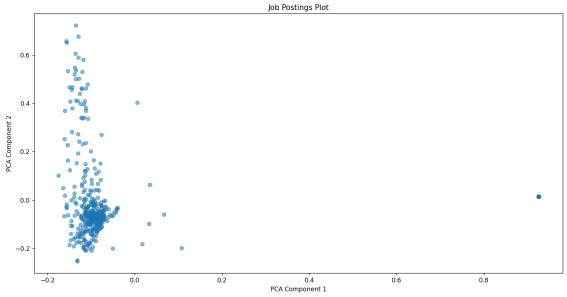
Based on the plot above the max score is at cluster 7 with the score being about 0.05

Plotting applicants

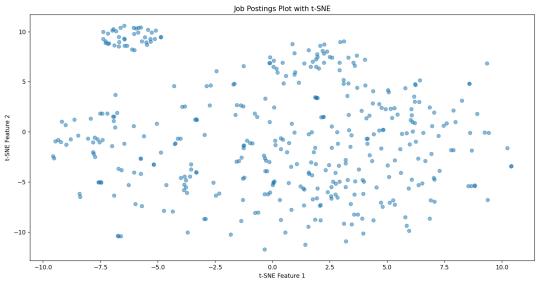
TSNE	PCA



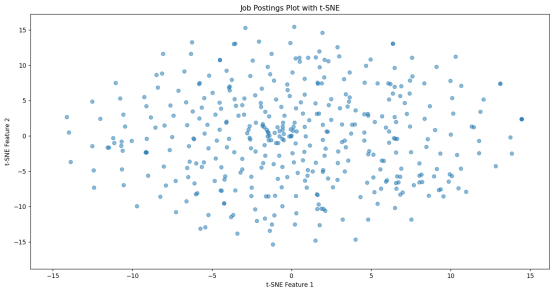
*ignore the title, also this is just job titles of applicants



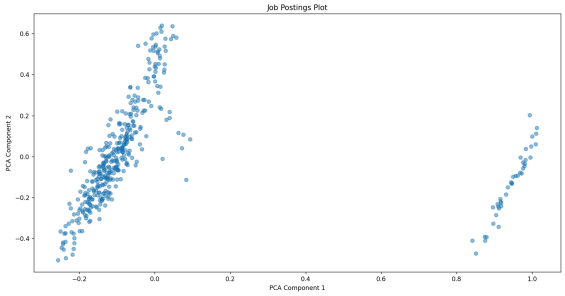
*ignore the title, also this is just job titles of applicants



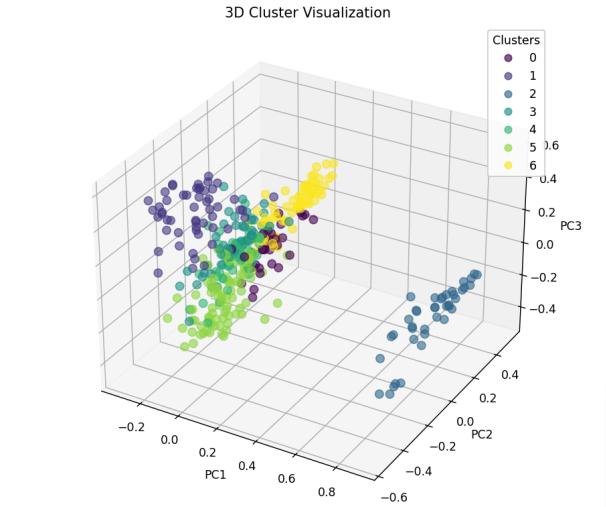
Job titles + skills



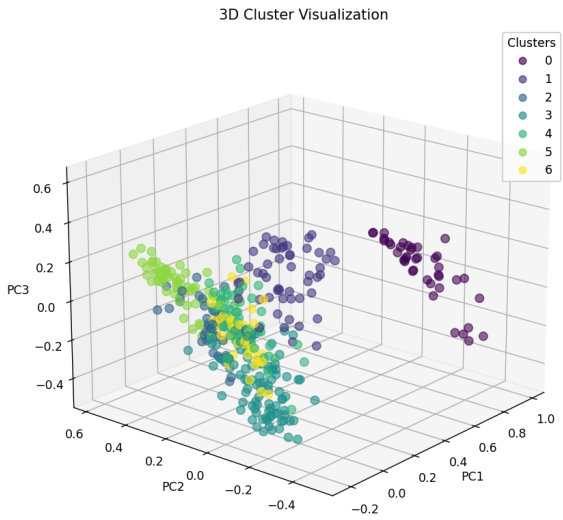
Job titles + skills + desc



Job titles + skills + desc



No desc, 7 clusters



Different angle

Additional Info

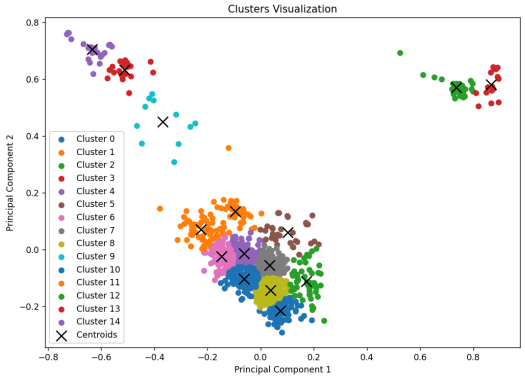
Elbow Method is an empirical method to find the optimal number of clusters for a dataset. In this method, we pick a range of candidate values of k, then apply K-Means clustering using each of the values of k. Find the average distance of each point in a cluster to its centroid, and represent it in a plot. Pick the value of k, where the average distance falls suddenly.
https://towardsdatascience.com/silhouette-method-better-than-elbow-method-to-find-optimal-clusters-378d62ff6891

Silhouette Analysis: It measures how similar an object is to its own cluster compared to other clusters. The Silhouette Score ranges from -1 to +1, where a high value indicates that the object is well-matched to its own cluster and poorly matched to neighboring clusters. If the Elbow Method is unclear, this might provide more insight.

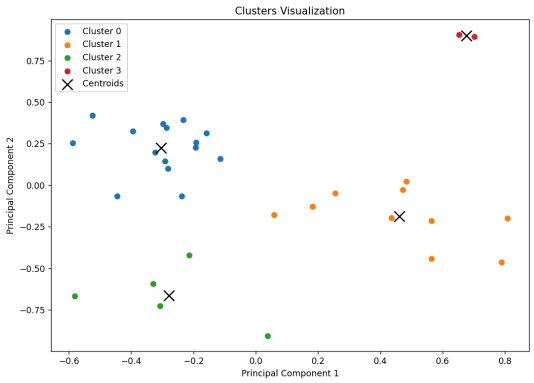
https://www.sciencedirect.com/science/article/pii/S1877050920318020
User similarity:
https://github.com/jalajthanaki/Job_recommendation_engine/blob/master/Job_recommendation_engine.ipynb

Train Logistic Regression with Feedback: Train the logistic regression model on the cluster assignments and the feedback. The model will learn the relationship between the clusters and the likelihood of positive feedback.

Prediction and Evaluation: Use the trained logistic regression model to predict the feedback for new, unseen job postings. Evaluate the model's performance using the actual feedback from users.



Job rec clusters from some time ago



Resume clusters from sometime ago

Ideas:
Cluster, give results, user gives feedback, new user gets recs+find similar user and rec jobs they liked
Still should retrain model

- Use clustering to group similar jobs and recommend jobs from the same cluster to users with similar preferences.
- Use Logistic Regression (or another classification model) to predict user interactions with jobs based on job features and possibly user features.
- Integrate user feedback to either refine the clustering (e.g., by adjusting features used for clustering) or as labels in a supervised learning task to predict user preferences.

ChatGPT

To create a confidence rating for these results, we should consider both the `score` (cosine similarity) and the `like_probability` (from logistic regression), as these metrics reflect different aspects of how well a job matches the user's profile and preferences. The `combined_score` already aggregates these two, but for a confidence rating, we might want to emphasize the importance of a match being both similar (high `score`) and likable (high `like_probability`).

One approach is to use a weighted average of the `score` and `like_probability`, giving more weight to the metric you believe is more important for the confidence rating. Alternatively, you could consider scaling these values or applying a function that accentuates higher values more than lower ones.

Here's a basic formula for a confidence rating:

$$\text{Confidence Rating} = \alpha \times \text{Normalized Score} + \beta \times \text{Normalized Like Probability}$$

Where:

- α and β are weights that sum to 1, reflecting the importance of the score and like probability, respectively.
- Normalized Score and $\text{Normalized Like Probability}$ are the score and like probability normalized to the same scale, typically [0, 1].

First, let's normalize the `score` and `like_probability` to ensure they are directly comparable: