**Big Data Pipeline - Intelligent Job Matching with Real-Time ML Inference on AWS**

Chu-Chun Ku

Dataset: [LinkedIn Jobs and Skills 2024](https://www.kaggle.com/datasets/asaniczka/1-3m-linkedin-jobs-and-skills-2024)

**Documents:**

* Final Project \_Deliverables.pdf
* Final project\_output file.zip
* Final project\_script.zip
* Final project\_presentation recording.mp4 (with Demo videos)
* Final project\_presentation slide.pptx

**1. Business Use Case / ML Task**

**Scenario:**I work for a global recruitment company and am developing an intelligent talent-job matching platform. When a job seeker provides their skills, the system leverages a machine learning model to   
(1) predict the most suitable job title based on the skill set, and   
(2) optionally evaluate the compatibility between the candidate’s skills and their expected job title by generating a matching score.   
This dual prediction mechanism supports both recommendation and validation, streamlining the decision-making process for recruiters and job seekers.

**Goal:**To build a real-time, ML-powered job matching platform that performs two key functions: (1) recommends suitable job titles for each talent based on their provided skills  
(2) evaluates how well a candidate fits their expected job title by comparing their skills against typical skill requirements for that title.   
This dual approach enhances both talent recommendation and compatibility assessment, all automated through a scalable ETL pipeline.

**2. Data Pipeline Architecture Diagram**

Our architecture consists of two integrated data pipelines: one for ML model creation, and one for real-time job role inference.

**a. ML Model Creation Flow**

* Raw Job Data Ingestion (S3)
  + Upload original job datasets (linkedin\_job\_postings.csv, job\_skills.csv) to: s3://job-matching-group1/raw/
* ETL Preprocessing (Lambda: job\_data\_cleaner\_lambda)
  + This Lambda function cleans and merges the raw job data.
  + Output is stored at: s3://job-matching-group1/clean/cleaned\_jobs.csv
* Model Training (SageMaker Notebook)
  + A machine learning model is trained to predict job titles based on job skills using the cleaned data.
  + The trained model is exported to: model.tar6.gz to s3://job-matching-group1/model/
* Model Deployment (SageMaker Endpoint)
  + The trained model is deployed to a SageMaker endpoint named: job-title-endpoint-v6

**b. Real-Time Talent Inference Flow**

* Talent Input Simulation (CloudShell):
  + A Python script (simulate\_data1.py) generates mock job seeker input (skills and expected job title).
  + The data is streamed to: Kinesis Stream – incoming-job-stream
* Streaming Ingestion (Lambda – incoming-job-stream-lambda):
  + Triggered by Kinesis, this function decodes the incoming data and inserts it into: DynamoDB table – UnpredictedJobEvents
* ML Inference (Lambda – predict\_lambda):
  + This Lambda function fetches new records from UnpredictedJobEvents, sends the job skills to the SageMaker endpoint, and receives:
    - Predicted\_job\_title
    - matching\_score (if an expected\_title is provided)
    - Results are stored in: DynamoDB table – PredictedJobEvents
    - S3 – /predictions/{timestamp}.csv
* Full Sync to Unified CSV (Lambda – sync\_predictions\_lambda):
  + Triggered by new records in PredictedJobEvents, this Lambda aggregates predictions into a daily file at: s3://job-matching-group1/all\_predictions/all\_predictions-YYYY-MM-DD-HH-MM.csv
* Data Query & Analysis (Glue + Athena):
  + Glue Crawler indexes the prediction files in S3.
  + Data is cataloged in the Glue database: job\_predictions\_db
  + Athena is used to query and visualize the prediction results

**3. Implementation Plan and Documentation**

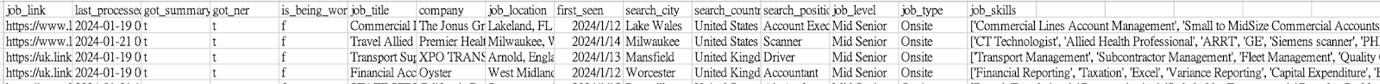
**a. AWS services used and why**

|  |  |  |
| --- | --- | --- |
| **Service** | **ML Model Creation** | **Real-Time Talent Inference** |
| **Amazon S3** | Stores raw job datasets (linkedin\_job\_postings.csv, job\_skills.csv), cleaned datasets, and trained ML models. | Store prediction results (/predictions/) and full synced results (/all\_predictions/). |
| **AWS Lambda** | job\_data\_cleaner\_lambda cleans and merges raw job data into a usable format. | incoming-job-stream-lambda is triggered by Kinesis and stores raw talent data in DynamoDB. predict\_lambda invokes SageMaker for predictions. sync\_predictions\_lambda consolidates predictions into a unified CSV. |
| **Amazon SageMaker** | Notebook instance used to preprocess and train job recommendation models. Model is exported and deployed as an endpoint. | The endpoint is invoked by predict\_lambda to generate job title prediction and optional match score. |
| **Amazon Kinesis** | - | Stream real-time talent input from CloudShell to Lambda for ingestion. |
| **Amazon DynamoDB** | - | Stores raw talent input UnpredictedJobEvents and prediction results PredictedJobEvents. |
| **AWS CloudShell** | - | Simulate real-time talent input with Python script (simulate\_data1.py). |
| **AWS Glue** | - | Glue Crawler scans S3 prediction files and builds a queryable schema. |
| **Amazon Athena** | - | Queries processed predictions and generate visual insights from job\_predictions\_db. |

**b. ETL design and schema decisions**

* **ETL design:**
  + **Extract:** Retrieve raw job datasets from S3 (linkedin\_job\_postings.csv and job\_skills.csv).
  + **Transform:** job\_data\_cleaner\_lambda merges the two files, removes null values, and standardizes fields like job\_level and job\_type.

The cleaned dataset is saved as cleaned\_jobs.csv in /clean/.



* **Load:** The cleaned dataset is used to train an ML model in SageMaker Notebook and exported to /model/ as a .tar.gz archive.
* **Schema Decisions:**
  + The “job\_skills” field is a multi-valued attribute and is encoded using MultiLabelBinarizer.

A close up of a list

AI-generated content may be incorrect.

**c. Data ingestion details**

* ML Model Training (Batch Processing): The model is trained on the preprocessed sampled\_jobs.csv in a one-time batch mode. The process can be manually triggered as needed.
* Talent Input and Automated Process (Real-Time Streaming): Simulated input is generated via CloudShell and pushed into a Kinesis Data Stream.
  + incoming-job-stream-lambda is triggered by Kinesis, decodes the data, and writes it into the DynamoDB table UnpredictedJobEvents.
  + predict\_lambda polls new items from UnpredictedJobEvents, sends job\_skills and expected\_title to the SageMaker Endpoint, and receives prediction results.
  + Results are written to:
    - DynamoDB table PredictedJobEvents
    - S3 folder /predictions/
  + A secondary Lambda function, sync\_predictions\_lambda, is triggered by updates in PredictedJobEvents, and consolidates the results into a unified CSV file in the /all\_predictions/ folder on S3.

**d. Security considerations**

All services follow the principle of least privilege to ensure end-to-end data protection.

* IAM Role Enforcement: All Lambda and SageMaker jobs use a shared IAM role (LabRole)
* S3 Bucket Protection: Buckets are private with access limited to the assigned IAM role. Structured directories: /raw/, /clean/, /model/, /predictions/, /all\_predictions/.
* DynamoDB Streams: PredictedJobEvents table streams only NewImage to sync\_predictions\_lambda.
* CloudShell Input: CloudShell only simulates Kinesis input and does not require elevated permissions.

**e. Challenges and how you addressed them**

* **Challenges and Solution**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| SageMaker model loading failed due to version incompatibility (pkl error) | Downgraded to scikit-learn==1.2.2, retrained and redeployed model |
| Lambda couldn't decode Kinesis base64 payloads | Added decoding logic with .decode("utf-8") |
| Multiple CSV prediction files made Athena queries complex | Used sync\_predictions\_lambda to consolidate into a single master CSV |

* **Limitation and Solution**

Model Training Constraints - During the process of training the job title prediction model, we encountered several key limitations:

* Data Volume Exceeded Learner Lab Limits

The cleaned dataset originally contained over 1,048,576 records, which exceeded the capacity of SageMaker instances available in the Learner Lab. Even after testing subsets of 50,000, 20,000, and 5,000 records, the training still failed. Ultimately, we reduced the data to a sample of 1,000 records to complete model training and prediction.

* Inconsistent and Sparse Titles and Skills:

The dataset was sourced from LinkedIn job postings across many different companies. As a result:

* Job titles were highly varied, even for similar roles (e.g., "Software Engineer" vs. "Engineer - Software")
* Some titles appeared only once, making it difficult to learn reliable class boundaries
* Skills were inconsistently formatted (e.g., "Problem Solving", "problem solving", and "Problem-solving" were treated as different inputs), increasing the complexity of preprocessing and reducing model effectiveness

To mitigate this, we:

* Filtered job titles to include only those with ≥ 20 occurrences before sampling
* Applied TF-IDF vectorization, retaining only the top 5,000 terms to manage memory usage
* Low Prediction Diversity and Accuracy

Due to the above constraints, model predictions were skewed:

* The most frequent output was "Office Manager", regardless of input
* Most matching scores were 0.0, indicating weak alignment between expected and predicted job titles

These results suggest that the model requires improvement through better data cleaning, label standardization, and potentially larger instance types for training.

**4. Visualizations and Insights**

**a. Figure 1: Top Expected Titles by Candidates**

A screenshot of a computer

AI-generated content may be incorrect.

SELECT expected\_title, COUNT(\*) AS total

FROM all\_predictions

WHERE expected\_title IS NOT NULL

  AND TRIM(expected\_title) <> ''

GROUP BY expected\_title

ORDER BY total DESC;

**Business Purpose:**

This query helps identify the most frequently entered expected job titles by users.  
It reveals which roles are in high demand and can be used to:

* Guide headhunters or recruiters in setting performance targets based on role popularity
* Understand trends in candidate job expectations

**b. Figure 2: Most Common Job Skills**

A screenshot of a computer

AI-generated content may be incorrect.

SELECT job\_skills, COUNT(\*) AS frequency

FROM all\_predictions

GROUP BY job\_skills

ORDER BY frequency DESC;

**Business Purpose:**

This query identifies the most frequently appearing job skills across all predictions. It helps:

* Detect current market demand trends based on skill popularity
* Inform compensation benchmarks by highlighting in-demand or scarce skills
* Guide recruiters and hiring managers in adjusting job requirements or salary expectations accordingly

**c. Figure 3: Most Predicted Job Titles**

A screenshot of a computer

AI-generated content may be incorrect.

SELECT predicted\_title, COUNT(\*) AS total

FROM all\_predictions

WHERE predicted\_title IS NOT NULL AND TRIM(predicted\_title) <> ''

GROUP BY predicted\_title

ORDER BY total DESC;

**Business Purpose:**

This query helps recruiters identify which predicted job titles appear most frequently. It enables them to:

* Prioritize talent pools based on predicted demand
* Streamline candidate sourcing for high-demand roles
* Align recruitment strategies with model-predicted trends in the job market

**d. Figure 4: Highest Matching Scores**

A screenshot of a computer

AI-generated content may be incorrect.

SELECT \*  
FROM all\_predictionsall\_predictions  
WHERE matching\_score >= 0  
ORDER BY matching\_score DESC  
LIMIT 20;

**Business Purpose:**

This query allows recruiters to focus on candidates with the highest alignment between their provided skills and their target job titles. It helps:

* Identify high-quality candidate-role matches
* Enable faster outreach for top candidates
* Support personalized and efficient recruitment strategies

**5. Demo Presentation**

* **Demo video → Please see the presentation recording (start from 13:50’)**
  + ML Model Creation Flow Demo.mp4
  + Real-Time Talent Inference Flow Demo.mp4
  + Live Inference Walkthrough\_Real-Time Talent Inference Flow Demo.mp4
* **Key business insights:**
  + **Skill Distribution Reflects Market Trends:**The most frequently appearing skills—such as Python, AWS, and Project Management—indicate strong demand for these competencies across industries. This insight allows HR teams to prioritize these skills in job descriptions and internal training programs. It also supports salary benchmarking efforts by highlighting in-demand capabilities.
  + **“Office Manager” Dominance Suggests Model Optimization Needed:**Although the model successfully returns predictions, a disproportionate number are labeled as “Office Manager.” This skew is likely caused by imbalanced training data or overly broad job title categories.  
    This observation highlights the need for future model improvements, including rebalancing training samples and standardizing job titles to improve prediction diversity and relevance.
  + **End-to-End Automation Enables Real-Time Feedback:**Once a job seeker submits their information, the system immediately predicts suitable job roles and evaluates skill compatibility. All results are automatically stored and synced in DynamoDB and S3, enabling real-time analysis and scalability. This ensures the system remains highly responsive and scalable, even as data volume and user load increase.

**6. Code and Scripts → Please see the Zip file (Final project\_Group1\_script.zip)**

**7. Final Output → Please see the Zip file (Final project\_Group1\_output file.zip)**

|  |  |  |  |
| --- | --- | --- | --- |
| **S3 File Path** | **Description** | **Source / Generation Process** | **Purpose** |
| s3://job-matching-group1/clean/cleaned\_jobs.csv | Cleaned and merged job dataset (job\_title, job\_skills, etc.) | Generated by job\_data\_cleaner\_lambda from raw LinkedIn datasets | Base dataset used for ML model training |
| s3://job-matching-group1/clean/sampled\_jobs.csv | Filtered and sampled job records (only frequent titles kept) | Processed in Google Colab, sampled from cleaned\_jobs.csv | Actual training dataset used for ML |
| s3://job-matching-group1/model/ job\_title\_model\_v4.pkl skill\_binarizer\_v4.pkl model\_v6.tar.gz  inference6.py | Trained ML model, feature encoder, and inference code | Created in SageMaker Notebook, then uploaded to S3 | Used for deployment to SageMaker Endpoint for predictions |
| s3://job-matching-output-group1/predictions/{timestamp}.csv | Per-candidate prediction results (one file per job\_id) | Written by predict\_lambda after inference | Individual prediction records for logging and traceability |
| s3://job-matching-output-group1/all\_predictions/all\_predictions-{timestamp}.csv | Aggregated prediction results (full dataset) | Generated by sync\_predictions\_lambda from all DynamoDB records | Used for Glue crawler and Athena queries analysis |