LinkedIn Job Market Insights

Siddharth Hiraou
UBID: shiraou

Aishvarya Samuel Salvi
UBID: aishvary

I. Introduction:

The job market is constantly evolving, with new roles emerging and skill requirements changing rapidly. Analyzing employment patterns can reveal important information about worker demands as companies adjust to changing economic conditions and technology progress. One such domain where data analysis is particularly impactful is the job market. The need for particular talents and employment responsibilities fluctuates quickly as industries change and new technologies are developed. In this industry, it is crucial for companies, policymakers, and job seekers to comprehend these developments. The 1.3M LinkedIn Jobs & Skills dataset, which includes over 1.3 million job advertisements extracted from LinkedIn, provides a comprehensive picture of the current labor market. For this project, we'll be building an end-to-end big data pipeline using Hadoop and Spark to analyze this dataset. The tasks performed are:

- Perform exploratory data analysis (EDA) using Pandas to understand key patterns and trends.
- Set up a local Hadoop cluster using Docker and verify its components.
- Develop a data ingestion script to store our dataset in HDFS for further processing.

II. Problem Statement:

The following ML problems are addressed using our dataset:

1. Classification of Job Title:
   The objective of this task is to classify job postings into predefined categories based on the job title, description, and required skills. By combining text data from job descriptions and skills, we can build a robust multi-class classification model to accurately categorize job postings. The expected outcome is a model that automatically classifies job postings into relevant categories, helping job seekers and employers streamline job searches and recruitment processes.
2. Skill Recommendation system: The objective of this task is to recommend a list of skills for a given job title or description using the skills data from the dataset. We can create a recommendation system that makes skill suggestions for particular positions by examining the relationship between job titles/descriptions and talents. A system that aids companies in crafting precise and thorough job descriptions and helps job seekers comprehend what skills are required for particular occupations is the expected outcome.
3. Predicting the Location of Jobs: The objective of this task is to predict the location of a job posting (city or country) based on the job title, company, and skills required. We can forecast the likely locations of particular employment roles by looking at trends in job names, employers, and skill sets. An approach that helps job searchers concentrate their employment searches geographically and gives employers insight into regional hiring trends is the expected result from this.

III. Data Analysis Objectives:

1. **Identify Top Recruiting Companies:** The objective of this analysis is to identify which companies are posting the most job listings and their geographic distribution. Job seekers can more effectively focus their applications and gain insights into the most active industries or regions by knowing which top recruiting organizations are situated in a given area. Employers looking to comprehend competitive hiring environments and job seekers seeking to narrow down their search can both benefit from these helpful insights.

2. **Evolution of Job Market Trends:**
   The objective of this analysis is to explore how job postings and required skills have evolved over time. Examining how job descriptions and necessary competencies have changed over time is the aim of this investigation. Finding these

trends enables stakeholders, including employers, schools, and job seekers, to comprehend how the labor market is evolving and predict demands in the future. We want to create visualizations (such as line charts and area charts) that show seasonal patterns, long-term trends, and new skill demands by monitoring the quantity of job posts and skill requirements over time. These insights can assist stakeholders make well-informed decisions and give a clearer picture of how the labor market has changed.

3. **Check for Skills Gaps in the Workforce**:

The objective of this analysis is to highlight skills that are in high demand but are not frequently listed in job postings, indicating potential skill gaps. Such insights will guide efforts to bridge skill gaps and align training programs with market needs. The goal of this analysis is to identify skills that are in high demand but are not frequently listed in job postings, indicating potential skill gaps. Finding these gaps is important for educators and training programs, as it helps them design curricula that address industry needs and prepare job seekers for future opportunities.

4. **Distribution of Job Opportunities Across various Locations:**

The objective of this analysis is to explore the geographic distribution of job postings to identify hotspots (regions with high job activity) and underserved regions (areas with fewer opportunities). Knowing these patterns is essential for employers as it gives them information about possible growth areas and for job seekers as it enables them to select places with more chances. Stakeholders will use these insights to inform data-driven choices on where to concentrate their efforts.

5. **Job Duration:**

Job seekers can identify positions that are expected to remain open for a longer period of time by knowing the posting duration, which also helps companies streamline their hiring procedures. Each job posting's duration is determined using the first_seen and last_processed_time fields, and the average duration per job title, business, or industry is examined. In order to give businesses and job searchers clear insights into how long job posts normally remain active, the findings are displayed using box plots or histograms, which aid in decision-making.

6. **Is Using hdfs dfs -put a Good Way to Write Files to an HDFS Cluster?**

The hdfs dfs -put command is a fairly simple, and straightforward way to write files out to an HDFS cluster and should only be used, for simple adhoc or manual uploads. However when dealing with the ingestion of significant amounts of data, this may not be the best option, due to network constraints, or NameNode performance. In our project where we are processing a dataset with a size of 5GB, we are more inclined to set up some automatic ingestion through the use of the Hadoop FileSystem API in Java. This allows us to get more elegant error handling, means of supporting integration into other workflows we want to do processing, and scalability. Automated ingestion is also able to be tuned for parallel writes, compression, and conversion to various formats, and can also be more suitable in production.

7. **Are You Satisfied with the Speed of Writing Using hdfs dfs -put? How Can You Improve It?**

Although the command `hdfs dfs -put` provides a simple way to put, it may not be fast enough for large files (like the 5GB dataset we will work with in this project). There are many factors creating the slowness, such as network latency (the time it takes to send data over the network), disk I/O (which can slow things down), and overloading the NameNode with many files in a single call. To improve performance, you work with `distcp`, a command designated to copy files in parallel; use a more efficient file format such as Parquet or ORC; use different HDFS block sizes to read/write simultaneously and gain better performance. At scale, integrating data ingestion with Apache Kafka or Flume may improve scalability and speed for data where the pipeline is continuously submitting updating records.

8. **What Format Should You Use to Store Data in HDFS for Easier Analysis?**

To facilitate storage and further analysis in HDFS, a structured, columnar format like Parquet or ORC is preferable than storing raw CSV. During this remainder of the project, as we need to handle a relatively small 5GB dataset, a Parquet format will have considerable benefits in terms of query performance and storage compression, while also lowering storage costs by bringing faster access to relevant data. A CSV file is row-based, meaning it is inefficient for large-scale analytics. By saving a

file in Parquet, it optimizes for columnar reads. With columnar formats, there is an additional performance boost when using Apache Spark or Hive, as many distributed processing frameworks are optimized for analytical workloads, which increases overall efficiency.

IV. Visualization:

We find out which job titles are in high demand, which firms are hiring, what talents employers are seeking, and how the labor market is evolving over time by using exploratory data analysis (EDA) and visualizations. We've created clear and intriguing visualizations using tools like Seaborn, Matplotlib, and Folium to make these insights simple to comprehend. These results give comprehensive view of the current employment market so that job seeker, employer, or educator, may make better decisions.

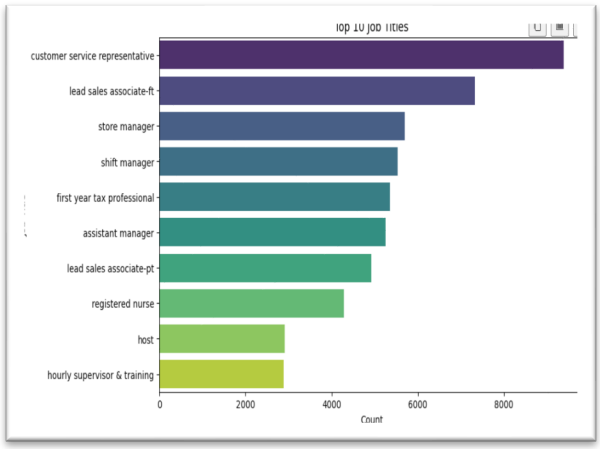


Figure: Top 10 Job Titles

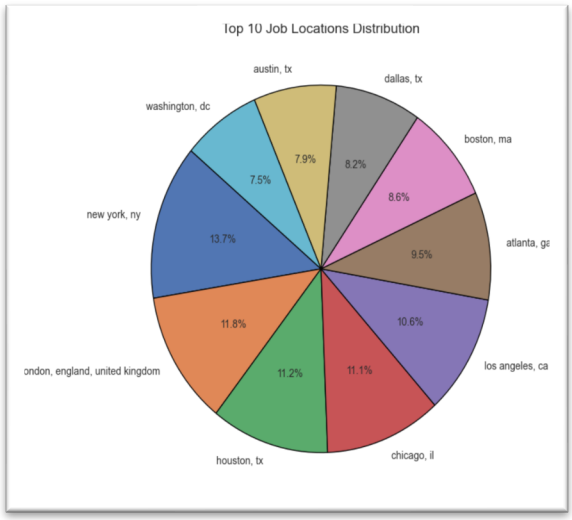


Figure: Top 10 Job locations

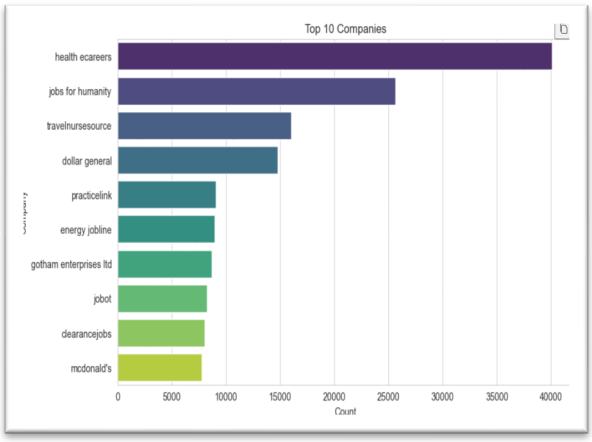
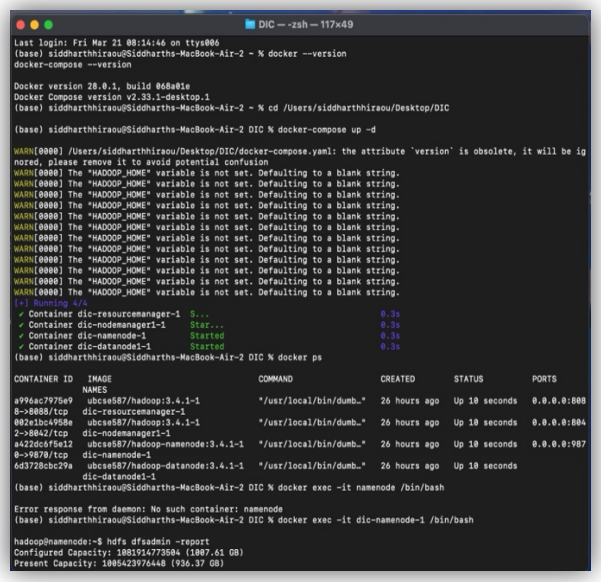
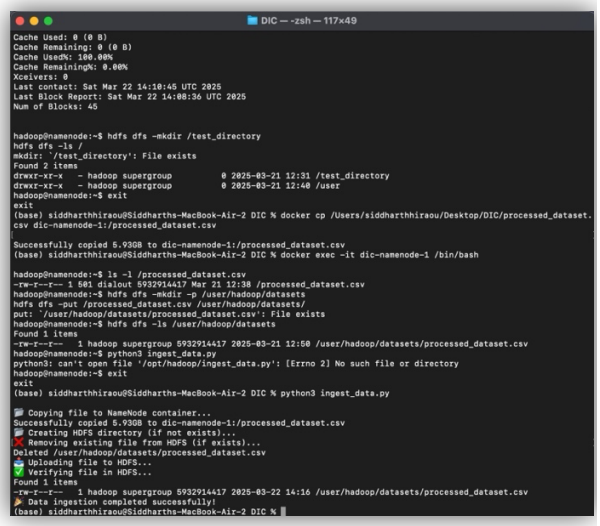


Figure: Top 10 Companies

Figure: Word cloud





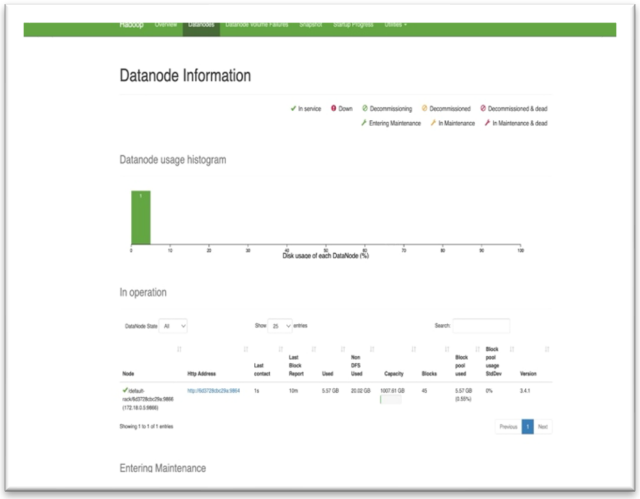


Figure: Datanode 1

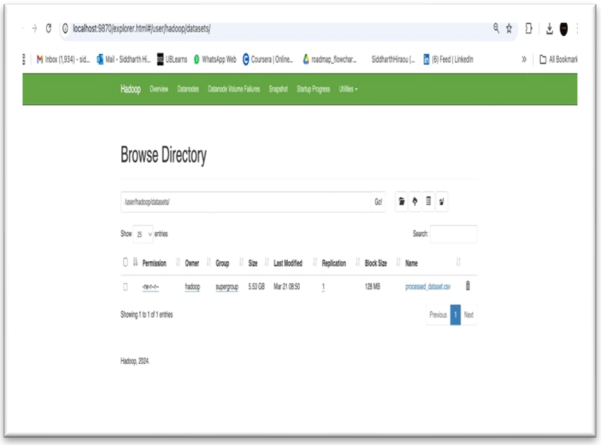


Figure: Browse Directory

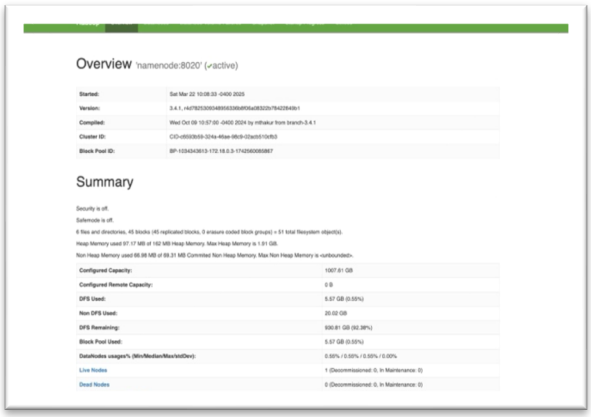
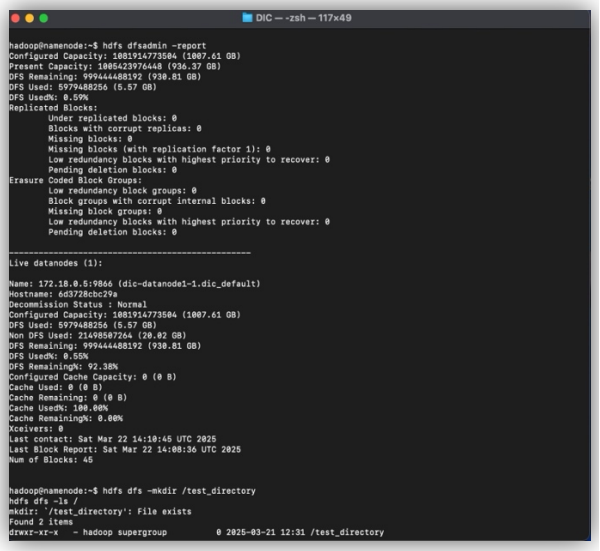


Figure: Datanode 2



V. Hadoop Cluster Running:

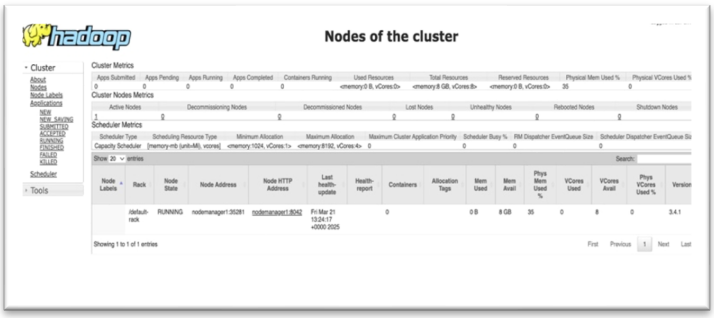


Figure: Cluster Running

VI. References:

1. https://github.com/UBCSE587/2025Spring-projectphase1
2. https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html
3. https://github.com/spinlud/py-linkedin-jobs-scraper
4. https://github.com/spinlud/linkedin-jobs-scraper

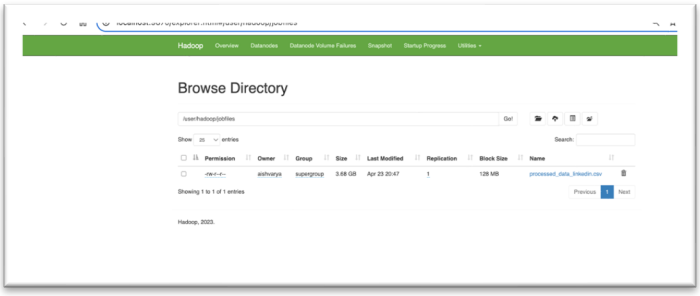
Milestone 2:

I. INTRODUCTION

The goal of this project is to build a comprehensive big data pipeline using Apache Hadoop and Apache Spark. We worked with LinkedIn job listings data to apply exploratory data analysis (EDA), data cleaning, feature engineering, machine learning model building, hyperparameter tuning, and model persistence into HDFS. This report outlines the detailed steps taken and the results achieved during Phase 2 of the project.

II. EDA

We performed the EDA on our dataset using Pyspark. We started by reading the processed data stored on HDFS at the path. Upon loading the dataset, we printed the schema to understand the structure of the data. The dataset contained fields such as job_link, last_processed_time, got_summary, got_ner, is_being_worked, job_title, company, job_location, first_seen, search_city, search_country, search_position, job_level, job_type, job_skills, and job_summary. Most of these fields were of StringType, while timestamps and dates were parsed appropriately.



We performed a null value analysis across all columns. The analysis revealed that there were no missing values, which indicated that our dataset was clean and ready for further processing.

```
✓ 13.6s
[Stage 865:>                                                (0 + 1) / 1]
+--------+-----------------+-----------+-------+---------------+---------+-------+
|job_link|last_processed_time|got_summary|got_ner|is_being_worked|job_title|company|
+--------+-----------------+-----------+-------+---------------+---------+-------+
|       0|                0|          0|      0|              0|        0|      0|
+--------+-----------------+-----------+-------+---------------+---------+-------+
```

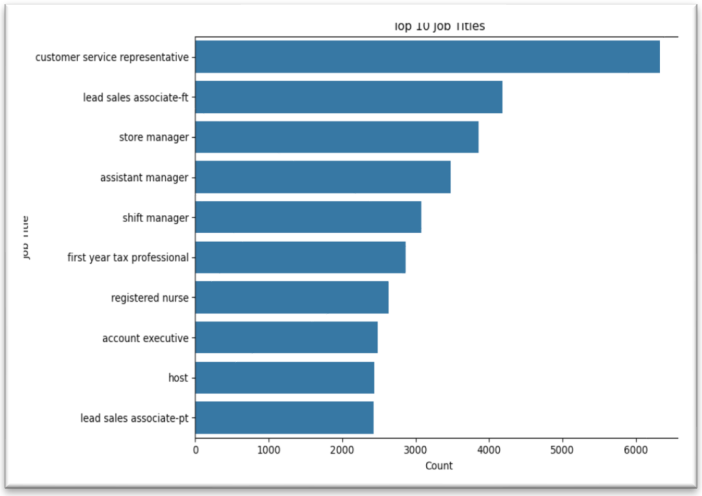
Outlier detection was performed using the IQR method on numeric columns, and the detected outliers were removed to ensure model stability.

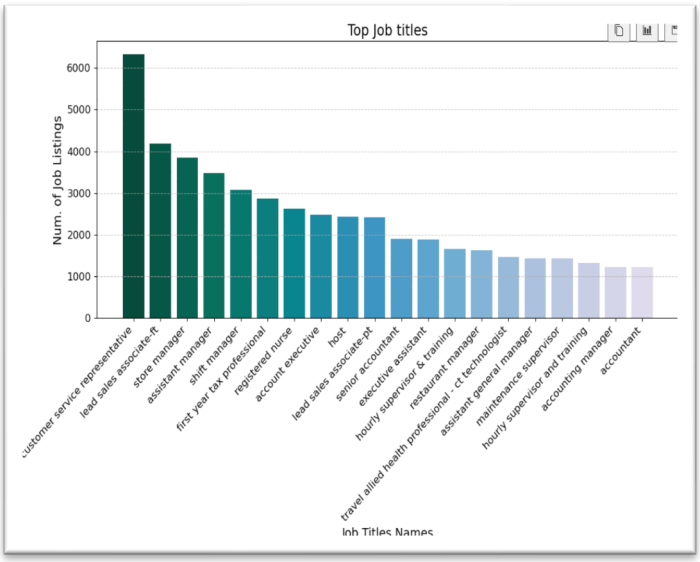
```
for col_name in numeric_cols:
    quantiles = df.approxQuantile(col_name, [0.25, 0.75], 0.05)
    Q1, Q3 = quantiles
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df = df.filter((col(col_name) >= lower_bound) & (col(col_name) <= upper_bound))

print("Outliers removed using IQR method.")

✓ 0.0s
Outliers removed using IQR method.
```

Data type conversion was also performed. String fields representing timestamps were converted into TimestampType fields using PySpark functions. Similarly, boolean indicators were cast to boolean types. Text fields such as job_title and company were trimmed to remove leading and trailing spaces.

Aggregation and grouping operations were carried out using Spark SQL and DataFrame functions to analyze job titles, companies, and locations. After cleaning and processing, all cleaned and processed datasets were written back to HDFS. This ensured that the data could be efficiently used for machine learning modeling without redundancy or the risk of inconsistency.

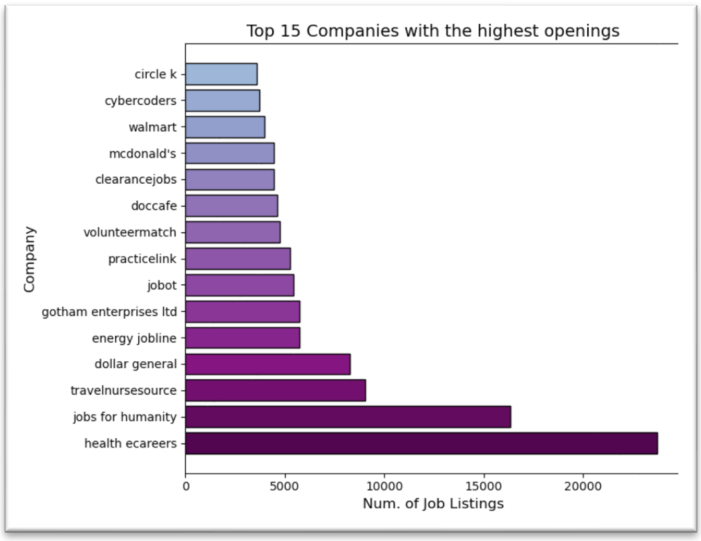
We generated multiple plots. The top 10 job titles indicated that "Customer Service Representative" was the most common role, followed by roles like "Sales Associate" and "Store Manager".



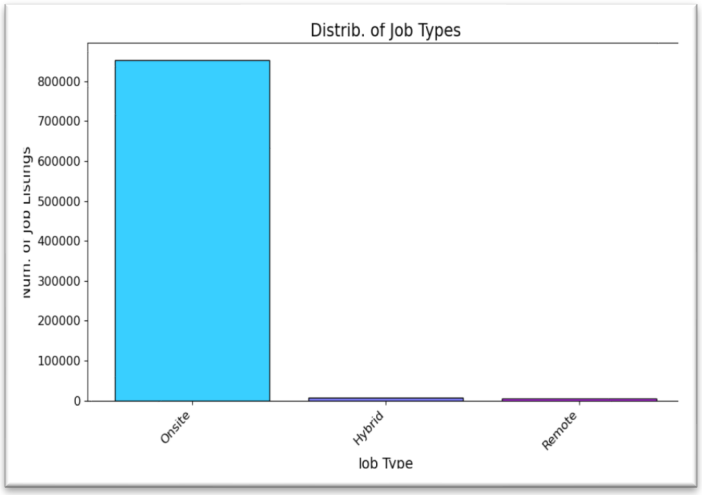


The analysis of jobs per company revealed that "health ecareers" had the most listings, followed by "jobs for humanity" and "travelnursesource".

```
jobs_per_company.show(20)
✓ 10.2s

[Stage 18:>
+--------------------+-----+
|             company|count|
+--------------------+-----+
|     health ecareers|23731|
|   jobs for humanity|16332|
|   travelnursesource| 9037|
|      dollar general| 8278|
|      energy jobline| 5735|
|gotham enterprise...| 5728|
|               jobot| 5451|
|        practicelink| 5284|
|      volunteermatch| 4731|
|             doccafe| 4625|
|       clearancejobs| 4454|
|          mcdonald's| 4430|
|             walmart| 3993|
|         cybercoders| 3722|
|            circle k| 3576|
|              target| 3284|
|     texas roadhouse| 3086|
|u.s. department o...| 3069|
|           h&r block| 3011|
|        michael page| 2903|
+--------------------+-----+
only showing top 20 rows
```



A distribution plot for job types showed that Onsite jobs dominate the market, while Hybrid and Remote roles were significantly fewer.



To understand the required skills across job postings, we extracted and visualized skills using a word cloud. Dominant terms such as Communication, Customer Service, Leadership, and Management appeared prominently, indicating the importance of these skills across jobs.

A pie chart was also generated to visualize the top 10 job locations by proportion. Job location analysis showed that New York, NY and London, UK were among the top cities for job postings and major cities dominate the job listing share.

III. SPARK ML

a) Data Preprocessing

We used StringIndexer to encode the categorical columns job_level and job_type, and employed VectorAssembler to combine numerical feature columns into a single feature vector for model training.

b) Model Training and Evaluation

Data was split into **80% train** and **20% test** datasets.

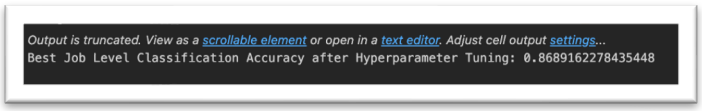
We developed three machine learning models.

One is Random Forest Classifier for Job Level which Classified job listings into different job levels.

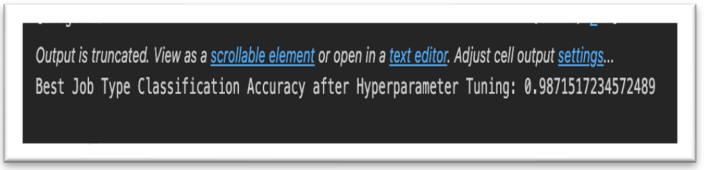
Second one is Random Forest Regression for Job Type which Classified job listings into different job types.

And third one is KMeans Clustering which discovered hidden clusters in the job data (k=3)

The Random Forest Classifier model for predicting job level achieved an accuracy of approximately 86.89% after hyperparameter tuning, while the model for Random Forest regression predicting job type achieved a high accuracy of approximately 98.71%.



Hyperparameter tuning was performed using grid search with a range of values for numTrees and maxDepth, along with three-fold cross-validation, to ensure the robustness of the models.



The clustering model was trained using KMeans with three clusters, and the model effectively grouped the job postings based on feature similarities.

The best-trained models for job level classification, job type classification, and clustering were saved into HDFS under the directory /user/hadoop/models/. Predictions generated by these models were also saved separately under /user/hadoop/predictions/. Evaluation metrics,

including the accuracy scores for classification models, were stored under /user/hadoop/metrics/.

The storage of both models and predictions into HDFS ensures that the results are reproducible and can be used for future analysis or deployment.

IV. CONCLUSION

Through this project, we demonstrated the successful construction of a complete big data pipeline utilizing Apache Spark and Hadoop. We efficiently handled the ingestion and processing of a large-scale dataset, performed insightful exploratory analysis, cleaned and transformed the data, built robust machine learning models, optimized their performance through hyperparameter tuning, and persisted the results to HDFS.

This project highlights the scalability, efficiency, and practical applicability of Spark and Hadoop for real-world big data and machine learning tasks. The methods used in this project serve as a strong foundation for handling even larger datasets and more complex machine learning problems in future endeavors

V. References:

1. https://github.com/adipolak/ml-with-apache-spark
2. http://github.com/NVIDIA/spark-rapids-ml
3. https://spark.apache.org/docs/latest/ml-guide.html
4. https://spark.apache.org/mllib/