School of Computer Science and Engineering

Course Code: B22EF0505	**Big Data Analytics Lab** **Mini Project**	**Academic Year:2024-25** **Semester & sec:5th sem 'B' sec**
Project Details:		
Assignment Title:	**IT JOB ANALYSIS(2019-2023)**	
Place of Project:	**REVA UNIVERSITY, BENGALURU**	
Student Details:		
Name:	Vishwa .J	**Sign:**
Mobile No:	6361599801	
Email-ID:	Ugcet22100111@reva.edu.in	
SRN:	R22EH072	
Faculty Member Details		
Faculty Name	Prof. Thanuja K	**Sign:**
Grade by Faculty Member		

Contents **Page no.**

Abstract

The IT industry has experienced substantial growth over the years, generating a plethora of job opportunities worldwide. Understanding the clustering of IT job roles, locations, industries, and salary patterns is vital for recruiters, job seekers, and analysts. This project explores clustering techniques using PySpark to analyze a dataset of IT jobs spanning from 2019 to 2023. By employing K-Means clustering, we identify patterns and categorize job roles based on key features like salary range, job title, location, and industry. This report delves into the methodology, data cleaning processes, implementation steps, and visualization techniques, presenting insights and potential future applications of clustering in job analytics.

Introduction

IT sector becoming a cornerstone of global economies, job analytics has emerged as a crucial area of study. The vast pool of data generated by job postings, employee profiles, and industry reports can provide invaluable insights when analyzed effectively. Clustering algorithms, like K-Means, allow us to group similar entities, revealing hidden patterns and enabling data-driven decision-making.

This project focuses on the application of PySpark, a distributed computing framework, to handle a large dataset of IT job postings.

The primary objectives are:

1. To preprocess and clean the dataset for analysis.
2. To extract and encode relevant features, such as job titles, locations, and salary ranges.
3. To implement K-Means clustering and evaluate the results.
4. To visualize the clustering outcomes and derive meaningful insights.

The outcomes of this project can guide stakeholders in understanding job market-trends, salary distributions, and role-specific clustering patterns.

This project showcases the integration of big data analytics and machine learning for a critical real-world problem. It emphasizes the importance of preprocessing in ensuring the quality of input data, the power of distributed computing frameworks like PySpark in handling large datasets, and the capability of machine learning models.

Architecture

The architecture of this project follows the following steps:

1. **Data Ingestion:** The dataset was loaded into a PySpark DataFrame from a CSV file.
2. **Data Preprocessing:** Missing values were handled, and categorical columns were converted into numerical indices using StringIndexer.
3. **Feature Engineering:** Relevant features, including salary range and indexed categorical data, were assembled into a feature vector.
4. **Clustering Algorithm:** K-Means clustering was applied to group similar job profiles.
5. **Evaluation:** The clustering model's performance was evaluated using the Silhouette Score.
6. **Visualization:** Cluster distributions and salary trends were visualized using Matplotlib and Seaborn.
7. **Output:** Cluster predictions were saved to a new CSV file for further analysis.

Data Cleaning and Analysis

Data Cleaning

1. **Handling Missing Values:** The dataset contained missing values in key columns like "Job Title" and "Salary Range." These rows were removed to ensure data consistency.

```
# Handle missing values in the available columns
data = data.dropna(subset=available_cols)
```

2. **String Indexing:** Categorical variables such as "Job Title," "Location," and "Industry" were converted to numerical indices using PySpark's StringIndexer. This step ensured compatibility with the K-Means algorithm, which operates on numerical data.

```
indexers = []
for col_name in available_cols:
    indexer = StringIndexer(inputCol=col_name, outputCol=f"{col_
    indexers.append(indexer)
for indexer in indexers:
    data = indexer.transform(data)
```

3. **Feature Engineering:**
    - The "Salary Range" column was split into minimum and maximum salary values using regex.
    - An average salary column ("Salary_Avg") was calculated for clustering.

```
assembler = VectorAssembler(inputCols=["Salary_Avg", "Job Title_
data = assembler.transform(data)
```

Analysis

Key features for clustering:

- **Salary_Avg**: Represents the average salary of the job.
- **Job Title Index:** Encodes job titles as numerical values.
- **Location Index:** Encodes job locations as numerical values.
- **Industry Index:** Encodes industries as numerical values.

Code implementation

```
!pip install spark

!pip install pyspark
```

Show hidden output

```
# Import necessary libraries
from pyspark.sql import SparkSession
from pyspark.ml.feature import VectorAssembler, StringIndexer
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
import pandas as pd
import matplotlib.pyplot as plt

# Initialize Spark session
spark = SparkSession.builder.appName("JobClustering").getOrCreate()

# Load the dataset
data = spark.read.csv("/content/it job 2019-23.csv", header=True, inferSchema=True)

# Display schema and preview the data
data.printSchema()
data.show(5)

from pyspark.sql.functions import col

# Ensure columns exist in the DataFrame
expected_cols = ["Job Title", "Location", "Industry", "Job Type"]
available_cols = [col for col in expected_cols if col in data.columns]

# Handle missing values in the available columns
data = data.dropna(subset=available_cols)
```

```
# Apply StringIndexer only on available columns
indexers = []
for col_name in available_cols:
    try:
        indexer = StringIndexer(inputCol=col_name, outputCol=f"{col_name}_index").fit(data)
        indexers.append(indexer)
    except Exception as e:
        print(f"Error processing column '{col_name}': {e}")

# Transform the data with indexers
for indexer in indexers:
    data = indexer.transform(data)

# Confirm successful indexing
data.show(5)

# Clean column names by removing leading and trailing spaces
from pyspark.sql.functions import col

data = data.select([col(c).alias(c.strip()) for c in data.columns])

# Confirm columns are clean
print("Columns after cleaning:")
print(data.columns)

# Extract "Salary Range" into numerical values
from pyspark.sql.functions import regexp_extract

data = data.withColumn("Salary_Min", regexp_extract(col("Salary Range"), r"(\d+),?(\d+)?", 1).cast("float"))
data = data.withColumn("Salary_Max", regexp_extract(col("Salary Range"), r"(\d+),?(\d+)?", 2).cast("float"))
data = data.withColumn("Salary_Avg", (col("Salary_Min") + col("Salary_Max")) / 2)

# Show results
data.select("Salary Range", "Salary_Min", "Salary_Max", "Salary_Avg").show(5)
```

```
# Drop unnecessary columns to simplify the data
data = data.drop("Salary Range", "Salary_Min", "Salary_Max")

# Assemble features into a single vector column for clustering
feature_cols = ["Salary_Avg", "Job Title_index", "Location_index", "Industry_index"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
data = assembler.transform(data)

# Apply K-Means Clustering
kmeans = KMeans(featuresCol="features", k=4, seed=42)  # Choose k=4 clusters (can be adjusted)
model = kmeans.fit(data)

# Make predictions
clusters = model.transform(data)

# Evaluate clustering performance
evaluator = ClusteringEvaluator(featuresCol="features", predictionCol="prediction", metricName="silhouette")
silhouette_score = evaluator.evaluate(clusters)
print(f"Silhouette Score: {silhouette_score}")

# Show clustered data
clusters.select("Job Title", "Location", "Industry", "Salary_Avg", "prediction").show()

# Visualize cluster distributions
cluster_counts = clusters.groupBy("prediction").count().toPandas()

plt.figure(figsize=(8, 6))
plt.bar(cluster_counts["prediction"], cluster_counts["count"], color="skyblue", edgecolor="black")
plt.title("Cluster Distribution")
plt.xlabel("Cluster")
plt.ylabel("Number of Jobs")
plt.xticks(cluster_counts["prediction"])
plt.tight_layout()
plt.show()

# Cluster centroids
print("Cluster Centers:")
for center in model.clusterCenters():
    print(center)
```

```
# Save predictions back to a CSV (optional)
clusters.select("Job Title", "Location", "Industry", "Salary_Avg", "prediction") \
    .write.csv("/content/clustered_jobs.csv", header=True)
```

Visualization code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Visualizing the cluster distribution
cluster_counts = clusters.groupBy("prediction").count().toPandas()

plt.figure(figsize=(8, 6))
plt.bar(cluster_counts["prediction"], cluster_counts["count"], color="skyblue", edgecolor="black")
plt.title("Cluster Distribution", fontsize=16)
plt.xlabel("Cluster", fontsize=12)
plt.ylabel("Number of Jobs", fontsize=12)
plt.xticks(cluster_counts["prediction"], rotation=45)
plt.tight_layout()
plt.show()

# Visualizing Salary_Avg vs Cluster for all jobs
clusters_df = clusters.select("Salary_Avg", "prediction").toPandas()

plt.figure(figsize=(10, 6))
sns.scatterplot(x="Salary_Avg", y="prediction", data=clusters_df, palette="Set2", s=100, edgecolor="black")
plt.title("Average Salary vs Cluster", fontsize=16)
plt.xlabel("Average Salary", fontsize=12)
plt.ylabel("Cluster", fontsize=12)
plt.tight_layout()
plt.show()

# Visualizing Cluster Centers
centroids_df = pd.DataFrame(model.clusterCenters(), columns=["Salary_Avg", "Job Title_index", "Location_index", "Industry_index"])

plt.figure(figsize=(10, 6))
sns.heatmap(centroids_df, annot=True, cmap="viridis", fmt=".2f", linewidths=0.5)
plt.title("Cluster Centroids", fontsize=16)
plt.xlabel("Features", fontsize=12)
plt.ylabel("Clusters", fontsize=12)
plt.tight_layout()
plt.show()
```

Result and Analysis:

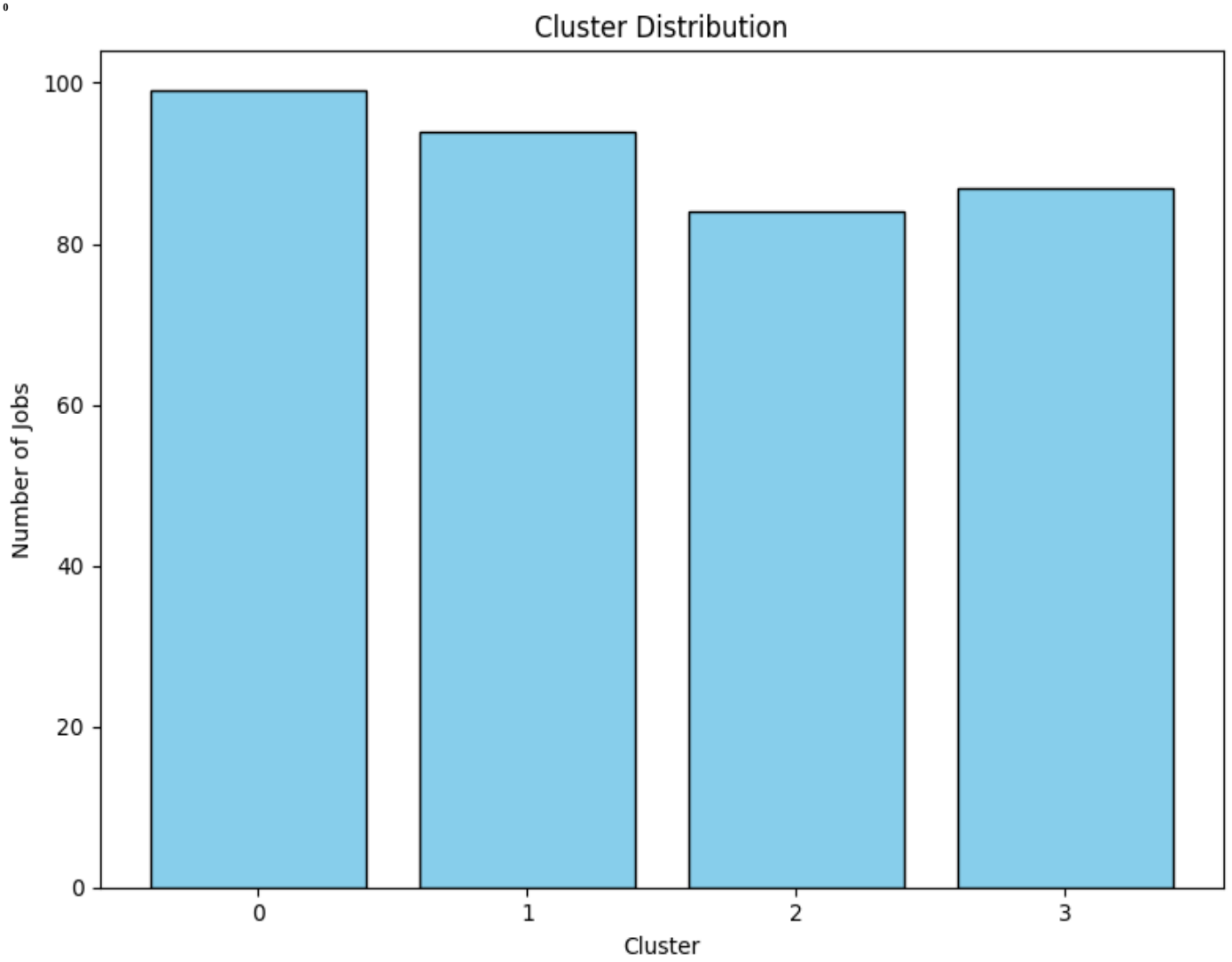
a)

```
+--------------------+----+--------------+------------------+---------+---------------+----+--------------+----+----+--------------+--------------+--------------+
|           Job Title| _c1|Required Skills|      Salary Range |  Location|        Company| _c6|      Industry| _c8| _c9|Job Title_index|Location_index|Industry_index|
+--------------------+----+--------------+------------------+---------+---------------+----+--------------+----+----+--------------+--------------+--------------+
|   Software Engineer|NULL|   Java, Python| ï¿½40,000 - ï¿½60,000 |    London|        ABC Tech|NULL|    Technology|NULL|NULL|          52.0|           7.0|          38.0|
|        Data Analyst|NULL|     SQL, Excel| ï¿½35,000 - ï¿½50,000 |Manchester|    XYZ Analytics|NULL|     Analytics|NULL|NULL|           8.0|           1.0|          11.0|
|    Network Engineer|NULL|     Cisco, WAN| ï¿½45,000 - ï¿½70,000 |Birmingham|Network Solutions|NULL|    Networking|NULL|NULL|          11.0|           0.0|          14.0|
|     Cloud Architect|NULL|     AWS, Azure| ï¿½60,000 - ï¿½90,000 | Edinburgh| Cloud Innovators|NULL|Cloud Computing|NULL|NULL|           7.0|           3.0|           2.0|
|Cybersecurity Ana...|NULL|  Cybersecurity| ï¿½50,000 - ï¿½80,000 |   Glasgow|      SecureGuard|NULL|      Security|NULL|NULL|           4.0|           2.0|           1.0|
+--------------------+----+--------------+------------------+---------+---------------+----+--------------+----+----+--------------+--------------+--------------+
only showing top 5 rows

Columns after cleaning:
['Job Title', '_c1', 'Required Skills', 'Salary Range', 'Location', 'Company', '_c6', 'Industry', '_c8', '_c9', 'Job Title_index', 'Location_index', 'Industry_index']
+------------------+----------+----------+----------+
|      Salary Range|Salary_Min|Salary_Max|Salary_Avg|
+------------------+----------+----------+----------+
| ï¿½40,000 - ï¿½60,000 |      40.0|       0.0|      20.0|
| ï¿½35,000 - ï¿½50,000 |      35.0|       0.0|      17.5|
| ï¿½45,000 - ï¿½70,000 |      45.0|       0.0|      22.5|
| ï¿½60,000 - ï¿½90,000 |      60.0|       0.0|      30.0|
| ï¿½50,000 - ï¿½80,000 |      50.0|       0.0|      25.0|
+------------------+----------+----------+----------+
only showing top 5 rows

Silhouette Score: 0.4371872697274797
+--------------------+----------+--------------------+----------+----------+
|           Job Title|  Location|            Industry|Salary_Avg|prediction|
+--------------------+----------+--------------------+----------+----------+
|   Software Engineer|    London|          Technology|      20.0|         0|
|        Data Analyst|Manchester|           Analytics|      17.5|         3|
|    Network Engineer|Birmingham|          Networking|      22.5|         1|
|     Cloud Architect| Edinburgh|     Cloud Computing|      30.0|         3|
|Cybersecurity Ana...|   Glasgow|            Security|      25.0|         3|
|  IT Project Manager|     Leeds|  Project Management|      35.0|         3|
|      Data Scientist|   Bristol|        Data Science|      27.5|         3|
|     DevOps Engineer| Newcastle|              DevOps|      25.0|         1|
|  IT Support Analyst| Sheffield|             Support|      15.0|         0|
|      UX/UI Designer|   Cardiff|              Design|      20.0|         0|
|    Database Analyst|    London|            Database|      20.0|         0|
|       UI Developer|Manchester|     Web Development|      17.5|         0|
|System Administrator|Birmingham|         IT Services|      22.5|         0|
|      AI/ML Engineer| Edinburgh|Artificial Intell...|      27.5|         2|
|          IT Auditor|   Glasgow|               Audit|      25.0|         2|
|Network Security ...|     Leeds|            Security|      27.5|         0|
|     Software Tester|   Bristol|   Quality Assurance|      15.0|         0|
|Cloud Solutions A...| Newcastle|     Cloud Computing|      32.5|         0|
|       IT Consultant| Sheffield|          Consulting|      30.0|         2|
| Front-end Developer|   Cardiff|     Web Development|      20.0|         1|
+--------------------+----------+--------------------+----------+----------+
only showing top 20 rows
```

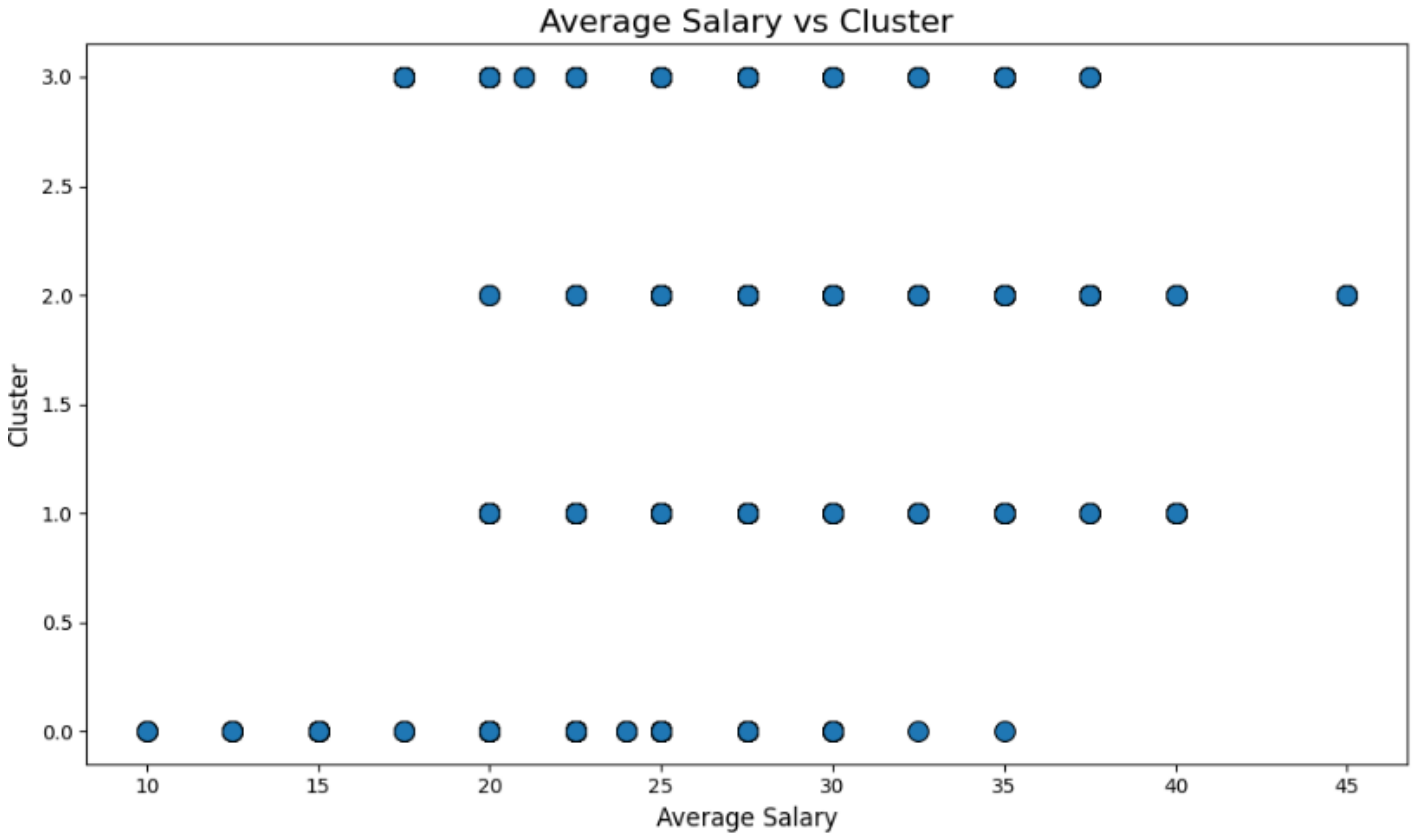
- The above output shows the dataframe before and after Preprocessing

- It shows the top 5 rows of the dataframe

b) visualization output




```
Cluster Centers:
[[22.15656566 45.32323232  4.37373737 10.78787879]
 [28.59042553 18.56382979  5.          7.91489362]
 [31.2202381  21.25        4.48809524 22.75      ]
 [26.79885057  3.08045977  3.59770115  3.85057471]]
```


Average Salary vs Cluster
Cluster
Average Salary
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10
15
20
25
30
35
40
45


Cluster Centroids
Clusters
Features
0
1
2
3
22.16
45.32
4.37
10.79
28.59
18.56
5.00
7.91
31.22
21.25
4.49
22.75
26.80
3.08
3.60
3.85
Salary_Avg
Job Title_index
Location_index
Industry_index
45
40
35
30
25
20
15
10
5

Applications and Impact

1. **For Job Seekers**:
    - Insights into salary trends, location preferences, and skill requirements can help individuals align their career goals with market demands.
2. **For Employers:**
    - Companies can optimize recruitment strategies by targeting clusters corresponding to their industry and salary benchmarks.
3. **For Policy Makers:**
    - Understanding job market trends enables policymakers to craft education and training initiatives, addressing skill gaps in the IT workforce.
4. **For Educators:**
    - Institutions can tailor curricula to meet industry needs, ensuring that graduates possess in-demand skills.

Future scopes , Enhancements & Job portals

1. **Feature Expansion**: Incorporating additional features such as company size, required experience, and job descriptions to improve clustering accuracy.
2. **Dynamic Clustering**: Automating the selection of the optimal number of clusters (k) using techniques like the elbow method.
3. **Real-Time Analysis**: Adapting the model for streaming data to analyze job postings in real-time.
4. **Integration with Recruitment Platforms**: Extending the model to provide personalized recommendations for job seekers and employers.
5. **Cross-Regional Analysis**: Comparing trends across different regions or countries to identify global patterns.

LinkedIn

- **Website**: linkedin.com
- Description: A professional networking platform that connects employers and job seekers. It has a vast collection of technical jobs from companies worldwide. Users can showcase skills, connect with industry professionals, and access job-specific insights.

Indeed

- **Website**: indeed.com
- Description: A leading job search engine that aggregates listings from various sources. It offers a user-friendly interface with filters for technical jobs, location, and salary ranges.

Glassdoor

- **Website**: glassdoor.com
- Description: In addition to job listings, Glassdoor provides company reviews, salaries, and interview experiences shared by employees, helping candidates make informed decisions about technical roles.

AngelList

- **Website**: angel.co
- Description: A platform focused on startups, AngelList connects job seekers with opportunities in emerging tech companies. It's ideal for roles in software development, data engineering, AI, and more.

Dice

- **Website**: dice.com
- Description: A specialized platform for technology professionals. It offers job listings, career advice, and market insights tailored to technical fields like software engineering, cloud computing, and data science.

Conclusion

This project demonstrates the effective application of PySpark and K-Means clustering to analyze IT job data. By preprocessing the dataset, extracting meaningful features, and leveraging distributed computing, we successfully grouped jobs into clusters based on salary, location, and industry. The insights derived from this analysis can inform decisions in recruitment, workforce planning, and market trend analysis. Future enhancements will further refine the model's capabilities, ensuring broader applicability and impact.