DS210 Final Project Writeup

GitHub Repository: https://github.com/ShinThomas/ds210_project

Question of Interest:
"What patterns and relationships exist within the data science job market, and how might this be useful for future data scientists?"

Dataset: https://www.kaggle.com/datasets/ruchi798/data-science-job-salaries

Overview:
For my DS210 final project, I decided to analyze patterns and relationships within the data science job market using graph algorithms and variable comparisons to develop a better understanding of the fluctuating salaries and opportunities that I, as well as my data science peers, will have to navigate through in the near future. In particular, the primary objective of the project was to analyze the relationships between job titles, company locations, and salaries. I utilized graph structures to explore the connections between different job roles and locations, calculate degree distributions to understand the spread of job titles across regions and perform a similarity analysis between different job roles based on their geographical overlap. The analysis involved reading data from a CSV file, constructing a graph where nodes represent job titles and edges represent company locations, and analyzing the degree distribution of the graph to identify which roles are most widespread geographically. It also performed a similarity analysis between job titles, providing insights into how similar different roles are based on their company location distributions.

Project Implementation and Code:
For my project, I focused on a few key features, highlighting the main analyses: Graph Representation, Degree Distribution, Node Similarity (Jaccard), and Centrality. In order to carry out these analyses I started by reading my data from a CSV file (retrieved from Kaggle) and displaying the data to review the contents and check for any missing or incomplete/incorrect data. Fortunately, the dataset was well organized and cleaned and did not require any maintenance in terms of preparation for my analysis. I began by creating a struct called “Record” which houses detailed information about a specific employee's work history, salary, and company attributes in a module called models.rs. I also created a module called lib.rs which contains all of the modules I used, organizing it into one place.

The “main.rs” file serves as the entry point for the program and coordinates the various tasks of reading data, constructing a graph, and performing analyses on the graph structure. First, it reads a CSV file (ds_salaries.csv) containing job-related data using the “read_csv” function, which

deserializes each row into a “Record” struct. After loading the data, the program builds a graph where each node represents a job title and an edge is added between a job title and a company location based on the records. The graph is then printed, displaying each node (job title) and its connected neighbors (company locations). Next, the program performs a degree distribution analysis, which calculates the number of connections (edges) each node has and prints this distribution to the console. The degree distribution is also saved to a CSV file for later use. Additionally, the program calculates the Jaccard similarity between nodes, identifying the most similar and most dissimilar nodes in terms of their shared connections. The program prints out the results of these similarity calculations, showing the node pairs with the highest and lowest similarity scores.

The “main.rs” file is accompanied by various other modules such as “analysis.rs”, “stats.rs”, “graph.rs”, and “test.rs”. The “analysis.rs” file contains the bulk of the project’s analyses housing several functions related to the analysis of a graph's structure. First, for the degree distribution, the “calculate_degree_distribution” function computes the degree distribution of the graph, which is the count of nodes having a specific number of connections (edges). It returns a HashMap where the key is the degree (number of connections) and the value is the count of nodes with that degree. The “print_degree_distribution” function prints the degree distribution to the console, displaying each degree and its count. The “save_degree_distribution_to_file” is a simple function to save the degree distribution to a CSV file, which can be used for further analysis or visualization. Next, the file also contains centrality measures where the “calculate_centrality_measures” function calculates a simple centrality measure for each node. It counts the number of unique neighbors each node has, representing a basic form of centrality. This is returned as a HashMap, where the key is the node's name and the value is its centrality score. The “jaccard_similarity” function computes the Jaccard similarity between two nodes, which measures the similarity of their neighborhoods. It does this by calculating the ratio of the intersection of their neighbors to the union of their neighbors. A higher value indicates more similar nodes. Finally, the “find_most_similar_dissimilar_nodes” function iterates over all pairs of nodes in the graph and calculates their Jaccard similarity. It keeps track of the most similar and most dissimilar pairs based on their similarity scores, returning a tuple with the results.

Using the “stats.rs” file I also implemented two standard variable comparisons which were not as complex as graph algorithms just to broaden my analysis and information retrieval. The “stats.rs” file defines two functions for analyzing salary data from the collection of “Record” entries. These functions focus on calculating salary distributions based on job title and company location. The first is salary distribution by job title where the “calculate_salary_distribution” function calculates the distribution of salaries for each job title. It iterates over the provided records, collecting the “salary_in_usd” values for each job title into a vector. The function returns a HashMap where the key is the job title (as a String) and the value is a Vec<f64>, representing the salaries associated with that job title. The second function “calculate_salary_by_location”

calculates the salary distribution for each company location. Similar to the previous function, it iterates over the records and collects the “salary_in_usd” values for each company location into a vector. The function returns a HashMap where the key is the company location (as a String) and the value is a Vec<f64>, representing the salaries associated with that location.

I also defined a “Graph” struct in the module “graph.rs” which contains associated methods to represent and manipulate a graph. The graph is stored using an adjacency list, where each node (a String) is mapped to a list of its neighbors (a Vec<String>). It also implements the Graph by using “new()” and “add_edge(&mut self, node1: &str, node2: &str)” which adds an edge between two nodes, node1 and node2. If node1 doesn't already have a list of neighbors, a new Vec<String> is created. The method then adds node2 to the list of neighbors for node1. This represents an undirected graph, where edges are bidirectional.

Finally, to wrap it up I applied two tests to ensure the quality of the “Graph” struct and its methods. These tests ensure that the “Graph” behaves as expected when adding edges and handling various graph operations. The first test, “test_add_edge()” checks the functionality of adding edges to the graph while the second test, “test_empty_graph()”, ensures the behavior of the graph when it's empty. These tests validate the correct implementation of graph construction, edge addition, and behavior with empty graphs. They check both normal behavior and edge cases, such as duplicate edges and non-existent nodes.

Code Output:

Record { work_year: 2020, experience_level: "MI", employment_type: "FT", job_title: "Data Scientist", salary: 70000.0, salary_currency: "EUR", salary_in_usd: 79833.0, employee_residence: "DE", remote_ratio: 0, company_location: "DE", company_size: "L" }
Record { work_year: 2020, experience_level: "SE", employment_type: "FT", job_title: "Machine Learning Scientist", salary: 260000.0, salary_currency: "USD", salary_in_usd: 260000.0, employee_residence: "JP", remote_ratio: 0, company_location: "JP", company_size: "S" }
Record { work_year: 2020, experience_level: "SE", employment_type: "FT", job_title: "Big Data Engineer", salary: 85000.0, salary_currency: "GBP", salary_in_usd: 109024.0, employee_residence: "GB", remote_ratio: 50, company_location: "GB", company_size: "M" }
Record { work_year: 2020, experience_level: "MI", employment_type: "FT", job_title: "Product Data Analyst", salary: 20000.0, salary_currency: "USD", salary_in_usd: 20000.0, employee_residence: "HN", remote_ratio: 0, company_location: "HN", company_size: "S" }
Record { work_year: 2020, experience_level: "SE", employment_type: "FT", job_title: "Machine Learning Engineer", salary: 150000.0, salary_currency: "USD", salary_in_usd: 150000.0, employee_residence: "US", remote_ratio: 50, company_location: "US", company_size: "L" }
Graph Structure:
Node: Machine Learning Developer, Neighbors: ["IQ", "CA", "CA"]
Node: Applied Machine Learning Scientist, Neighbors: ["US", "US", "CZ", "US"]
Node: Head of Data Science, Neighbors: ["RU", "US", "US", "US"]
Node: Machine Learning Infrastructure Engineer, Neighbors: ["PT", "US", "PT"]
Node: Data Science Engineer, Neighbors: ["GR", "CA", "MX"]
Node: Lead Data Scientist, Neighbors: ["US", "AE", "IN"]
Node: Computer Vision Software Engineer, Neighbors: ["US", "US", "AU"]
Node: ETL Developer, Neighbors: ["GR", "GR"]
Node: Lead Data Engineer, Neighbors: ["NZ", "US", "US", "GB", "US", "CA"]
Node: Head of Machine Learning, Neighbors: ["IN"]

Node: NLP Engineer, Neighbors: ["US"]
Node: 3D Computer Vision Researcher, Neighbors: ["IN"]
Node: Applied Data Scientist, Neighbors: ["CA", "GB", "US", "US", "US"]
Node: Machine Learning Engineer, Neighbors: ["US", "CN", "US", "US", "HR", "ES", "DE", "IN", "BE", "US", "US", "JP", "CO", "IN", "US", "SI", "DE", "BE", "PL", "US", "IN", "DE", "IT", "US", "US", "GB", "GB", "GB", "DE", "US", "US", "US", "US", "AE", "AE", "US", "NL", "AU", "IE", "US", "US"]
Node: Data Specialist, Neighbors: ["US"]
Node: Data Analytics Engineer, Neighbors: ["DE", "US", "GB", "PK"]
Node: Data Analyst, Neighbors: ["US", "US", "PK", "FR", "NG", "IN", "US", "US", "FR", "US", "US", "US", "GB", "US", "US", "ES", "US", "US", "US", "CA", "US", "US", "US", "DE", "US", "CA", "CA", "CA", "CA", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "GB", "GB", "ES", "ES", "GR", "GR", "US", "US", "GR", "US", "US", "US", "CA", "CA", "US", "US", "US", "US", "US", "US", "GB", "US", "US", "US", "US", "US", "US", "US", "US", "GB", "US", "US", "US", "CA", "CA", "US", "US"]
Node: Lead Data Analyst, Neighbors: ["US", "US", "IN"]
Node: Product Data Analyst, Neighbors: ["HN", "IN"]
Node: Director of Data Science, Neighbors: ["US", "PL", "JP", "DE", "US", "DE", "CA"]
Node: Data Architect, Neighbors: ["US", "US", "US", "CA", "CA", "US", "US", "US", "US", "US", "US"]
Node: Big Data Architect, Neighbors: ["CA"]
Node: Staff Data Scientist, Neighbors: ["US"]
Node: Data Science Consultant, Neighbors: ["IN", "US", "DE", "DE", "ES", "US", "DE"]
Node: Principal Data Engineer, Neighbors: ["US", "US", "US"]
Node: BI Data Analyst, Neighbors: ["US", "US", "US", "US", "US", "KE"]
Node: Lead Machine Learning Engineer, Neighbors: ["DE"]
Node: Machine Learning Scientist, Neighbors: ["JP", "US", "US", "PK", "CA", "US", "US", "US"]
Node: Data Analytics Manager, Neighbors: ["US", "US", "US", "US", "US", "US", "US"]
Node: Principal Data Analyst, Neighbors: ["US", "CA"]
Node: Head of Data, Neighbors: ["US", "RU", "SI", "US", "EE"]
Node: Financial Data Analyst, Neighbors: ["US", "US"]
Node: Computer Vision Engineer, Neighbors: ["US", "BR", "DK", "BR", "US", "LU"]
Node: ML Engineer, Neighbors: ["DE", "US", "JP", "JP", "US", "PT"]
Node: Data Engineering Manager, Neighbors: ["DE", "ES", "US", "US", "US"]
Node: Data Science Manager, Neighbors: ["US", "US", "US", "FR", "US", "US", "IN", "US", "US", "US", "US", "US"]
Node: Business Data Analyst, Neighbors: ["US", "US", "LU", "IN", "CA"]
Node: Finance Data Analyst, Neighbors: ["GB"]
Node: Director of Data Engineering, Neighbors: ["GB", "US"]
Node: Machine Learning Manager, Neighbors: ["CA"]
Node: Marketing Data Analyst, Neighbors: ["DK"]
Node: Analytics Engineer, Neighbors: ["US", "US", "US", "US"]
Node: AI Scientist, Neighbors: ["DK", "US", "US", "AS", "ES", "US", "US"]
Node: Data Analytics Lead, Neighbors: ["US"]
Node: Principal Data Scientist, Neighbors: ["DE", "US", "US", "US", "US", "DE", "DE"]
Node: Data Engineer, Neighbors: ["JP", "GR", "MX", "AT", "US", "GB", "US", "FR", "DE", "US", "US", "US", "PL", "US", "US", "US", "IN", "NL", "US", "US", "US", "MT", "US", "GB", "US", "IR", "GB", "US", "US", "GB", "IN", "US", "GB", "TR", "GB", "US", "US", "US", "TR", "DE", "US", "US", "NL", "US", "US", "US", "GB", "US", "US", "GB", "GB", "GB", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "GB", "GB", "US", "US", "US", "US", "US", "US", "US", "GB", "US", "US", "US", "GB", "US", "GR", "GR", "GB", "GB", "US", "US", "US", "ES", "ES", "ES", "GB", "GB", "ES", "GR", "GB", "GB", "GR", "US", "US", "US", "US", "US", "DE", "US", "US", "US", "US", "US", "DE", "US", "US", "FR", "US"]
Node: Big Data Engineer, Neighbors: ["GB", "GB", "US", "RO", "IN", "MD", "CH", "IN"]
Node: Data Scientist, Neighbors: ["DE", "HU", "FR", "IN", "FR", "US", "US", "GB", "US", "ES", "US", "US", "US", "IT", "US", "DE", "DE", "US", "AT", "LU", "FR", "FR", "IN", "US", "NG", "IN", "FR", "US", "US", "US", "US", "CA", "UA", "IL", "MX", "CL", "US", "DE", "US", "AT", "ES", "US", "BR", "DE", "GB", "IN", "GB", "CA", "DE", "US", "ES", "VN", "CA", "US", "FR", "US", "US", "IN", "US", "IN", "DE", "FR", "US", "US", "TR", "US", "US", "US", "US", "US", "GB", "GB", "US", "US", "GB", "GB", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "US", "GB", "GB", "US", "US", "US", "US", "US", "IN", "IN", "US", "US", "US", "US", "GB", "GB", "US", "US", "US", "US", "DZ", "US", "PL", "US", "US", "CA", "MY", "AU", "US", "CH", "CA", "US"]

Node: Research Scientist, Neighbors: ["NL", "US", "GB", "CA", "FR", "PT", "CA", "US", "FR", "CA", "CN", "CZ", "US", "FR", "AT", "US"]
Node: Cloud Data Engineer, Neighbors: ["SG", "US"]
Degree Distribution:
Degree: 3, Count: 7
Degree: 4, Count: 4
Degree: 41, Count: 1
Degree: 2, Count: 6
Degree: 1, Count: 11
Degree: 132, Count: 1
Degree: 5, Count: 4
Degree: 7, Count: 5
Degree: 6, Count: 4
Degree: 8, Count: 2
Degree: 97, Count: 1
Degree: 16, Count: 1
Degree: 11, Count: 1
Degree: 143, Count: 1
Degree: 12, Count: 1
Degree distribution saved to degree_distribution.csv
Most Similar Nodes: Head of Machine Learning and 3D Computer Vision Researcher, Similarity: 1.0000
Most Dissimilar Nodes: Machine Learning Developer and Applied Machine Learning Scientist, Similarity: 0.0000

Here, the code outputted job records, showing detailed information for various roles such as Data Scientist, Machine Learning Scientist, and Big Data Engineer. Each record includes attributes like the work year, experience level, employment type, job title, salary, and currency, along with employee residence, company location, remote ratio, and company size. This highlights individual job data for analysis. The graph structure represents job titles as nodes, with connections to other nodes based on shared attributes, likely locations, or similar roles. For example, the “Machine Learning Engineer” node is highly connected, having neighbors spanning multiple locations, while other nodes, such as “Head of Machine Learning”, have fewer neighbors. This shows how interconnected or isolated different roles are within the dataset. The degree distribution output shows how many connections each node has. For instance, some nodes have very few connections, like 1 or 2, while others, such as the “Data Scientist” or “Machine Learning Engineer” nodes, have very high degrees, reflecting their importance or widespread presence. This distribution gives a sense of how connected job titles are within the graph. The similarity and dissimilarity results identify relationships between nodes. The “Head of Machine Learning” and “3D Computer Vision Researcher” nodes are the most similar, with a similarity score of 1.0, indicating they share identical neighbors or connections. Conversely, “Machine Learning Developer” and “Applied Machine Learning Scientist” are the most dissimilar, with no shared connections and a similarity score of 0.0. Finally, the degree distribution was saved to a CSV file, allowing for further use or visualization.

Results and Analysis:
After reviewing the results I was able to make inferences and notable conclusions to aid in answering my overall research question. Specifically when looking at the salary distributions and focusing on job titles, experience, and location, job roles such as "Machine Learning Scientist" and "Big Data Engineer" tended to have high salaries, with the "Machine Learning Scientist" earning 260,000 USD, and the "Big Data Engineer" earning 109,024 USD (converted). On the other hand, roles like "Product Data Analyst" report a significantly lower salary, at 20,000 USD. Furthermore, high salaries, particularly in the roles of "Machine Learning Scientist" and "Machine Learning Engineer", are generally seen in more developed regions such as the United States (US) and Japan (JP), where the data science industry is more mature and developed. In contrast, less developed regions, such as Honduras (HN), show much lower salaries, reflecting the disparity in data science compensation across different geographical regions. These factors come into play when searching and applying for jobs where titles and geographical locations can significantly change overall earnings. At a school like BU where there are students from around the world (even just in the data science department we have so much diversity), it is important to weigh the options of opportunities outside of just the US and the title “Data Scientist.”

The degree distribution analysis also displayed some important and interesting information. The graph structure represents nodes (job roles) and edges (connections between different job roles based on geographical regions). Through the analysis we can see that the most common degree in the graph is 3, with 7 nodes having this degree, meaning that these roles have connections to three different regions. We also observe high-degree nodes, such as the node with a degree of 132, which suggests this node is connected to many regions. Conversely, there are nodes with a degree of 1, indicating that some job roles are linked to only one region or have very few connections to other roles. The node with the highest degree (132) is likely a central role in the network, which could indicate a high level of international collaboration or importance. Utilizing the Jaccard similarity metric measures the overlap between the geographical regions associated with different job roles. The most similar nodes are "Head of Machine Learning" and "3D Computer Vision Researcher", with a Jaccard similarity of 1.0. This suggests that these two job roles share identical geographical connections, indicating that professionals in these roles are likely to be located in the same regions. The most dissimilar nodes are "Machine Learning Developer" and "Applied Machine Learning Scientist", with a Jaccard similarity of 0.0. This implies that these roles have no overlap in their geographical regions, suggesting that they may operate in entirely different global markets. The degree distribution overall shows a highly interconnected set of job roles, with some central nodes having numerous connections to various geographical regions, while others are more isolated. This will further aid in helping new students and employees entering the job market to assess all factors and variables when finding a job of fit.