Kayson Shaikh
May 5th, 2024
CDS DS 210
Final Write-up

Written Output:

Length of time dataset covers: 5.214433733621893 years
(11, 2010) to (1, 2016)

Trust Ratings Over Time plot observations:
- The average trust rating dipped most significantly in August 2013 to about -2.3.
- It was also negative in December 2013 and December 2015.
- The highest the average trust rating has been was the first month on the dataset, November 2010.

Ratings distribution bar graph observations:
- A majority of the trust ratings given are 1.
- Less than 3500 ratings are negative.
- But there are around 2500 ratings of -10 which is the 4th highest total of any rating after 1, 2, and 3.

Number of trust ratings over 7: 1150 (3.23%)

Full data:
Connected Components: 4
Size of each component: [5875, 2, 2, 2]
Total amount of nodes: 5881

Strong ratings only data:
Connected Components: 156
Size of each component: [528, 3, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 2, 5, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 5, 2, 2, 2, 2, 5, 5, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 8, 15, 2, 2, 3, 4, 2, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 3, 2, 2, 3, 2, 2, 2, 2, 2, 6, 2, 4, 3, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 3, 2, 7, 5, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2]
Total amount of nodes: 915

In my final project, I analyze a graph dataset representing members of the Bitcoin OTC trading platform. In this directed graph, vertices correspond to platform users, and edges denote trust ratings given from one user to another. Each edge is weighted by an integer ranging from -10 to 10, representing the rating, and includes a timestamp in seconds since epoch indicating the time of the rating.

When executing this project, it generates two PNG files as part of the output. One PNG file displays a plot showing the average trust rating per month over the dataset's time range of approximately 5.2 years. The second PNG file is a bar graph illustrating the distribution of trust ratings in the dataset. The code also prints basic observations from these visuals in the terminal output. Additionally, the code displays the dataset's time range, the count and percentage of trust

ratings exceeding 7, and the outcomes of running connected components on both the full dataset and the filtered dataset containing edges with trust ratings higher than 7.

I chose to run a connected components algorithm rather than a strongly connected components algorithm, ignoring the fact that the edges are directed, as which way the users are connected didn't matter to me when attempting to discover communities within Bitcoin OTC. In the implementation of connected components, a custom struct ConnectedComponents is utilized to track and merge connected nodes efficiently. This struct maintains parent nodes, ranks to optimize tree structure, and sizes of components. The find_root function recursively finds the root node of a given node while shortening the height of the tree, and the merge function combines components based on connecting nodes. The components_and_sizes function leverages this struct to determine the number of connected components and their respective sizes by running the merge function for each edge in the dataset, providing insights into the network's structure and community formations. After applying this function to a filtered dataset containing only edges with trust ratings exceeding 7, derived from the initial dataset being processed by the strong_ratings_only function, I have discovered two distinct communities within the Bitcoin OTC platform. These communities, comprising 528 and 15 members respectively, exhibit a pattern of consistently providing each other with high trust ratings.

In order to get the range of time that the dataset covers, I used the chrono library to convert seconds since epoch to month and year. This is done in the epoch_to_date function.

When making the visuals, the col_to_vec function was very helpful. This function takes a column of the dataset and turns it into a vector. Since the columns have different data types, in order to have this function be able to return any column, the enum VecType was created. This enum has the possibility of either holding a vector of integers or a vector of floats. The col_to_vec function returns this enum then the enum is unwrapped using the functions defined in the impl block. Once I had the vectors to use for the visuals, I used ChatGPT to teach me the syntax in creating these visuals. I ended up using the plotters library, and through multiple questions asked to ChatGPT about how the methods of the plotters library works, I created the time_ratings_plot and ratings_distribution_bargraph functions which create the visuals in PNG files.

Other than asking ChatGPT about the plotters library, the only other outside source I used during this project was a quick google search of common ways to implement a connected components algorithm. This helped me decide on using the ConnectedComponents struct.

Finally, there is module called unused_functions which has one function that I thought would be useful in implementing the connected components algorithm. The function node_count uses a HashSet to count the number of nodes in the dataset. This was replaced in the components_and_sizes function by getting the length of the node_map HashMap. I kept this function in a private module as it could be useful if I were to come back to this project to analyze this dataset, or other datasets, further.