CS181/DA210: Final Project Proposal

Team 3 - Uyen Nguyen, Yen Nguyen

I. Analysis Question

- Topic: Urban Services Analysis
- Central question: “How do the types, frequencies, and resolution times of service requests vary across different neighborhoods in San Francisco, and what does this tell us about the distribution of city services and community needs?”

II. Background Information

SF311 is a service provided by the City of San Francisco that offers residents, businesses, and visitors access to various non-emergency city services and information. It's designed as a one-stop solution for obtaining information and reporting issues within the city. Residents can report non-emergency issues like potholes, graffiti, street or sidewalk cleaning, and garbage collection issues. This helps the city quickly address and resolve such matters.

III. Primary Dataset

We will be examining the SF311 service request data as our main dataset. This dataset, which offers comprehensive information on different types of service requests from various neighborhoods, is available through the City of San Francisco's public data portal and can be accessed via this web API: https://data.sfgov.org/resource/vw6y-z8j6.json. Our analysis will adhere to all data usage regulations, and we will duly credit the dataset's source in our project.

IV. Dataset Description and Variables

This comprehensive dataset contains information about service requests from 2008 to 2023. It has approximately 6.59 million rows and 21 columns, as of December 2, 2023. We will mainly focus on data in November, 2023, which will be approximately 50,000 rows.

The key variables we will focus on include:

- ‘service_request_id’: The unique ID of the service request created (type: int)
- ‘requested_datetime’: The date and time when the service request was made (type: datetime)
- ‘updated_datetime’: The date and time when the service request was last modified. For requests with status = closed, this will be the date the request was closed (type: datetime)
- ‘status_description’: A single-word indicator (‘open’/‘closed’) of the current state of the service request (type: string)
- ‘status_notes’: Explanation of why status was changed to current state or more details on current status than conveyed with status alone (type: string)
- ‘agency_responsible’: The agency responsible for fulfilling or otherwise addressing the service request (type: string)
- ‘service_name’: The human readable name of the service request type (type: string)
- ‘service_subtype’: The human readable name of the service request subtype (type: string)
- ‘service_details’: The human readable name of the service request details (type: string)
- ‘address’: Location of the service request (type: string)
- ‘supervisor_district’: San Francisco Supervisor District as defined in “Supervisor Districts as of April 2012” (type: float)
- ‘neighborhoods_sffind_boundaries’: San Francisco Neighborhood as defined in “SF Find Neighborhoods” (type: string)
- ‘police_district’: San Francisco Police District as defined in “Current Police Districts” (type: string)
- ‘lat’: Latitude of the location, using the WGS84 projection (type: float)
- ‘long’: Longitude of the location, using the WGS84 projection (type: float)
- ‘source’: Mechanism or path by which the service request was received; typically ‘Phone’, ‘Text/SMS’, ‘Website’, ‘Mobile App’, ‘Twitter’ (type: string)

V. Methodology

1. Retrieve JSON Data from Web API:

- Use Python's *requests* library to fetch the JSON data from the web API.
- Given the data that we want to focus on (November, 2023) is approximately 50,000, we will use a while loop to scrape the data from the web till we hit the end of November since the default limit for *requests.get(url)* is 10,000 rows.

2. Convert JSON Data to Pandas DataFrame:

- Use the '*pd.DataFrame()*' function from the Pandas library to convert the JSON data into a DataFrame.
- After retrieving data for November 2023 from the web API, we convert it to a pandas dataframe to begin our analysis.

3. Data Description:

- Use '*.describe()*' for statistical summaries of numeric variables.
- Apply '*.shape*' to determine the dataset's dimensions.
- Implement '*.info()*' for a concise overview of variables, data types, and non-null counts.
- After this step, we should have a clear idea of how our dataset is structured and what are the independent variables as well as dependent variables.

4. Data Cleaning/Preprocessing:

- Reshape the data to adhere to tidy data principles.
- Handle missing values using appropriate methods like interpolation or filling with a placeholder.
- Standardize variable names to lowercase with underscores for consistency using '*.rename()*'.
- Convert data types where necessary using '*.astype()*'.

5. Comparative Analysis:

- Filter data by neighborhood using '*.loc[]*' to compare service request patterns.
- Utilize *'.groupby()'* to analyze differences in request types, frequencies, and resolution times across neighborhoods.

6. Visualization:

- Integrate Matplotlib and Seaborn for data visualization.
- Employ various chart types like bar, line, and heatmaps to illustrate findings.

7. Statistical Analysis:

- Conduct regression analysis to explore correlations between service request types, frequencies, resolution times, and neighborhoods.
- Use statistical tests to assess the significance of observed trends.

8. Expected Outcomes:

- Identification of neighborhoods with higher service request frequencies and longer resolution times.
- Insights into how different types of service requests are distributed across the city.
- Understanding of how city services and community needs are reflected in the data.

VI. Implications

Through our analysis, we can provide valuable information for city planners and policymakers to improve resource allocation. Also, with statistical analysis, we can offer insights into community needs and service distribution inequalities, which can contribute to discussions on urban development and public service management.