AlertNYC

Date – 12th May 2023

Prepared by – Shlok Goswami (sg6862)

Utsav Patel (up2023)

BIG DATA FINAL PROJECT REPORT

Introduction

This report presents a project that combines PySpark, MongoDB, and various python libraries for the analysis of the NYPD Complaints dataset. Leveraging PySpark's distributed computing capabilities, the dataset sourced from NYC Open Data, which includes felony, misdemeanour, and violation crimes reported to the NYPD from 2006 to 2020, was efficiently profiled, processed, cleaned, and merged. Furthermore, the project incorporated NYC neighbourhood data to provide comprehensive insights into crime patterns within specific neighbourhoods. The cleaned data was then loaded into MongoDB using PyMongo, facilitating data retrieval and manipulation for in-depth analysis of offense types, locations, and suspect/victim characteristics. To effectively present the findings, popular Python libraries such as Matplotlib, Folium, Geopy, Seaborn, and Plotly were employed for data visualization. The integration of these technologies enabled the project to gain valuable insights into crime trends, aiding decision-making processes related to crime prevention and law enforcement in New York City

Project Setup

The project utilized PySpark, a distributed data processing framework, for implementing various data processing tasks. The following libraries were installed using pip:

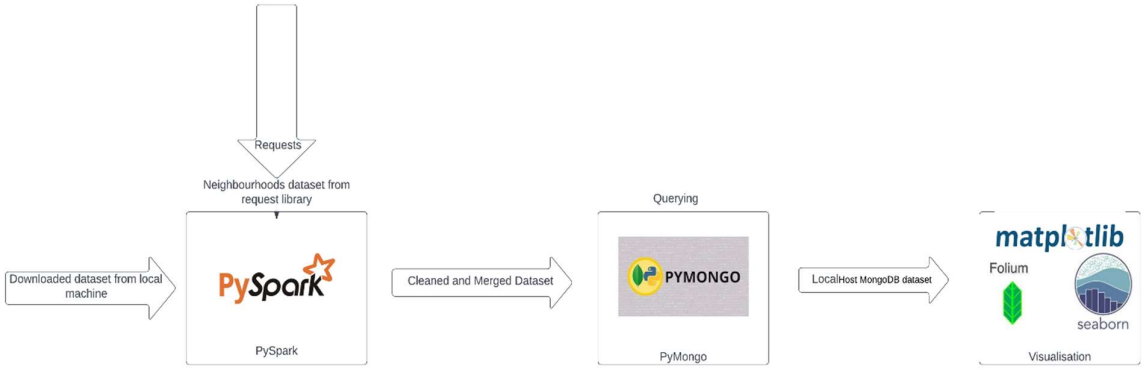
1. PySpark: PySpark provides a Python API for Apache Spark, allowing efficient distributed processing of large datasets.
2. PyMongo: PyMongo is a Python library for interacting with MongoDB, a popular NoSQL database. It enables integration between PySpark and MongoDB for seamless data operations.
3. Pandas: Pandas is a powerful data manipulation and analysis library in Python. It offers a wide range of functions for handling structured data efficiently.
4. Geopy: Geopy is a Python library that provides geocoding and reverse-geocoding capabilities. It enables the project to work with geographical data, such as latitude and longitude coordinates.
5. Folium: Folium is a Python library used for visualizing geospatial data. It integrates well with PySpark and allows the creation of interactive maps and visualizations.
6. Haversine: Haversine is a Python library used for calculating distances between two points on a sphere. It provides functions for computing distances based on latitude and longitude coordinates.
7. Matplotlib: Matplotlib is a popular data visualization library in Python. It offers a wide range of plotting functions and customization options for creating static plots.

8. Seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides high-level interfaces for creating attractive and informative statistical graphics.

The PySpark framework was used to load, process and merge the large NYPD Complaints dataset. The dataset was read into a Spark DataFrame using PySpark. Various data profiling and cleaning steps were performed using PySpark's powerful functions and transformations.

Project Architecture

Below is the architecture of the project we created. It starts with getting the dataset (NYPD Complains Data Historic) from the web by downloading the data then, loading it into PySpark dataframe using local Jupyter lab, then performing data profiling, validation, cleaning and pre-processing. After that we wanted to extract the neighbourhoods, dataset using some other way rather than downloading, so we decided to use request library of python and got the data onto another pyspark dataframe. Once we had both the datasets loaded we joined them on Borough name, and then performed haversine distance calculation using latitude and longitude columns of both the dataset later we used window function to only keep the distance which are the closest. After we had our cleaned and merged dataset we loaded that into MongoDB using pymongo for analysis, and finally we used various python libraries for visualization of the query results.



Data Profiling

The initial step in the project involves loading the NYPD Complaints dataset into a Spark DataFrame. Due to the large size of the dataset, a sampled version is used to ensure efficient processing. The dataset is loaded using PySpark, specifying the format of the dataset (e.g., CSV, JSON) and applying any necessary sampling techniques to obtain a representative subset.

Once the dataset is loaded, the schema of the DataFrame is printed to provide insights into the structure of the data, including information about the data types of each column. This helps in understanding the nature of the data and selecting appropriate data cleaning and transformation techniques. Additionally, summary statistics are generated using the **describe()** and **select()** methods to gain an overview of the data distribution. These statistics reveal important information such as the range, mean, standard deviation, and unique values with their frequencies. We also calculated the percentage value of null/empty values for each column.

By analysing summary statistics and column-level statistics, data anomalies can be identified, the distribution of the data can be understood, and informed decisions can be made regarding data cleaning and transformation techniques. These steps are crucial for ensuring data integrity and preparing the dataset for further analysis.

Data Preparation and Cleaning

The NYPD Complaints dataset was subjected to cleaning and pre-processing steps to ensure data quality and prepare it for further analysis. The following steps were performed:

1. Missing Value Handling:
    - Counting the number and percentage of missing values for each column allowed us to identify columns with missing data.
    - Rows with null values in selected columns, including 'Y_COORD_CD', 'X_COORD_CD', 'Latitude', 'Longitude', 'CRM_ATPT_CPTD_CD', 'CMPLNT_FR_TM', 'Lat_Lon', 'CMPLNT_FR_DT', 'BORO_NM', and 'OFNS_DESC', were dropped to remove incomplete or unusable data.
    - Columns that were deemed insignificant for future data exploration, such as 'PARKS_NM', 'STATION_NAME', 'TRANSIT_DISTRICT', 'HADEVELOPT', 'HOUSING_PSA', 'CMPLNT_TO_DT', 'CMPLNT_TO_TM', were dropped from the dataset.
    - Null values in 'LOC_OF_OCCUR_DESC', 'VIC_RACE', 'VIC_AGE_GROUP', and 'VIC_SEX' columns were replaced with the 'UNKNOWN' category to provide a meaningful value.
    - Null values in other columns were filled with 'UNKNOWN' to maintain consistency and completeness.
2. Data Type Conversion:
    - The 'CMPLNT_FR_DT' and 'RPT_DT' columns, initially in string format, were converted to date format. This conversion facilitates temporal analysis of the data.

- Selected columns, such as 'ADDR_PCT_CD', 'KY_CD', 'PD_CD', 'JURISDICTION_CODE', 'X_COORD_CD', and 'Y_COORD_CD', were converted to integer type for numerical analysis and efficient storage.

3. Standardizing Categorical Columns:

    - The abbreviations used in the 'VIC_SEX' and 'SUSP_SEX' columns were modified to their corresponding categories. Abbreviations like 'U', 'E', and 'D' were replaced with 'UNKNOWN', 'UNKNOWN', and 'BUSINESS/ORGANIZATION', respectively. 'F' and 'M' were replaced with 'FEMALE' and 'MALE' to represent gender categories accurately.

    - Incorrect age groups in the 'VIC_AGE_GROUP' and 'SUSP_AGE_GROUP' columns were removed to ensure consistency. Age groups that were not in the predefined categories (<18, 18-24, 25-44, 45-64, 65+, UNKNOWN) were filtered out.

    - Offenses in the 'OFNS_DESC' column were categorized into broader categories: 'PROPERTY', 'SEXUAL', 'DRUGS/ALCOHOL', 'PERSONAL', 'ADMINISTRATIVE', and 'OTHER'. This categorization helps in analysing crime patterns across different offense types.

4. Invalid Date Format:

    - Upon inspecting the sample dataset, we noticed that some rows had dates in improper formatting, different from the original "mm-dd-yyyy" format.

    - We utilized a combination of date and string manipulations to extract and format the values that deviated from the expected format.

    - Additionally, we accounted for variations in date formats encountered in some datasets, allowing for flexible handling of different timestamp formats.

5. Invalid Time Format:

    - Like the date format issue, we encountered rows with time values in improper formatting.

    - We employed time and string manipulation techniques to extract and format values that did not conform to the 24-hour format.

6. Validate Borough Names:

    - The validity of borough names associated with the incident location and patrol borough was checked.

    - Abbreviated borough names were replaced with their full forms to maintain consistency.

    - Missing values were imputed using reverse geocoding to assign the appropriate borough.

7. Validate Sex Column:

    - The sex column, representing the gender of suspects and victims, was validated and categorized into 'M' (Male), 'F' (Female), 'E' (Unknown), 'D' (Business/Organization), and 'Others'.

- Missing values in this column were replaced with "Unknown" to ensure completeness and consistency.

8. Validate Level of Offense:
    - The level of offense was validated and restricted to three categories: "Felony," "Misdemeanour," and "Violation".
    - Offenses that did not fall into these categories were removed from the dataset.

Data Merging

The purpose of this merging process is to add a neighbourhood column to the original crime dataset, which associates each crime with its corresponding neighbourhood in NYC. The following steps outline the data merging process:

1. Loading NYC Neighbourhoods Dataset:
    - Downloads the NYC neighbourhood's dataset, which is a collection of data related to boroughs, neighbourhoods, and their geographic coordinates (latitude and longitude). This dataset provides the necessary information to associate crimes with their corresponding neighbourhoods in NYC.
2. Creating a DataFrame for NYC Neighbourhoods:
    - Creates an empty list to store the neighbourhoods' data. A list is a data structure that can hold multiple values.
    - Then iterating over the downloaded neighbourhoods' data. In each iteration, extracting relevant information such as the borough name, neighbourhood name, latitude, and longitude.
    - The extracted information is stored as a dictionary, which is a collection of key-value pairs, in the list.
    - Finally, a Spark DataFrame is created from the list of dictionaries. A DataFrame is a distributed collection of data organized into named columns, like a table in a relational database. The appropriate schema, which defines the structure and data types of the columns, is also specified using the **pyspark.sql.types** module.
3. Joining Crime Dataset with Neighbourhoods Dataset:
    - The crime dataset (**neighborCrimeDF**) and the neighbourhood's dataset (**neighborhoodsDF**) are joined together using the **join** method.
    - The join condition in this case is based on matching the borough names in both datasets. Performing a case-insensitive comparison to ensure accurate matching.
    - The result of the join operation is a new DataFrame (**neighborCrimeDF2**) that contains all the columns from the crime dataset, along with additional columns from the neighbourhood's dataset, such as the borough, neighbourhood name, and geographic coordinates.

4. Calculating Distance between Crime Location and Neighbourhood:

    - Defines two user-defined functions (UDFs) to calculate the haversine distance between two sets of latitude and longitude coordinates. The haversine formula is used to compute the shortest distance between two points on a sphere, such as the Earth.
    - Two new columns, **LatLong** and **NLatLong**, are added to the DataFrame. These columns store the latitude and longitude coordinates of the crime location and the neighbourhood location, respectively.
    - Another column, **Distance**, is added to calculate the distance between the crime location and the associated neighbourhood. This calculation is performed using the UDFs and Spark SQL functions, leveraging the latitude and longitude values from the **LatLong** and **NLatLong** columns.

5. Selecting Nearest Neighbourhood for Each Crime:

    - A window function is used to assign a rank to each crime based on the distance from the associated neighbourhoods. Window functions enable the calculation of values over a specific window of rows in a DataFrame.
    - Filters the rows where the rank is equal to 1, indicating the nearest neighbourhood for each crime. This filtering operation results in a new DataFrame (**NCDF**) that contains the crime information along with the nearest neighbourhood for each crime.

Data Storing and Performing Query

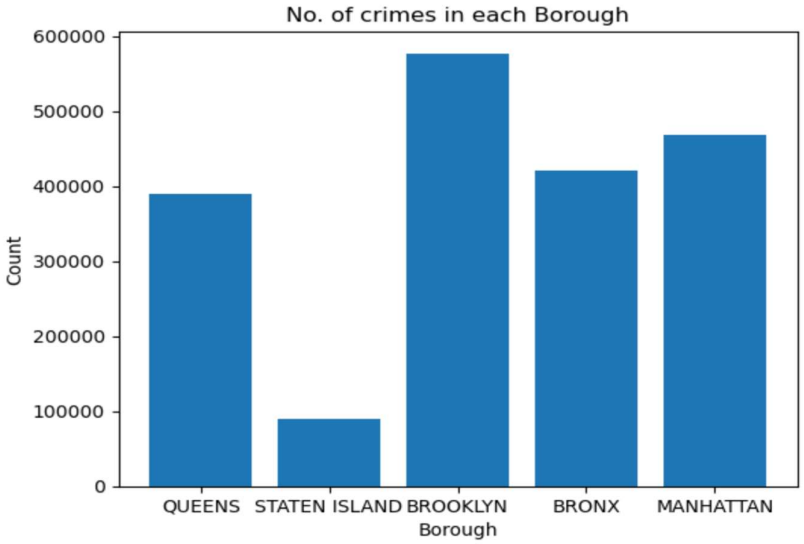
- The cleaned DataFrame, **NCDF**, is converted to a list of dictionaries using the **rdd.map(lambda row: row.asDict()).collect()** method.
- This step converts each row in the DataFrame to a dictionary, where the column names are the keys and the corresponding values are the dictionary values.
- The **asDict()** method is used to convert each row to a dictionary.
- The **collect()** method is called on the RDD (Resilient Distributed Dataset) to retrieve all the dictionaries as a list.
- Stored data into local mongoDB client using **pymongo, i**nserted the list of dictionaries into the MongoDB collection using the insert_many() method.

Data Analysis

The analysis phase of the project involves exploring the cleaned dataset and extracting insights from it. Various queries and aggregations are performed on the dataset using MongoDB's aggregation framework.

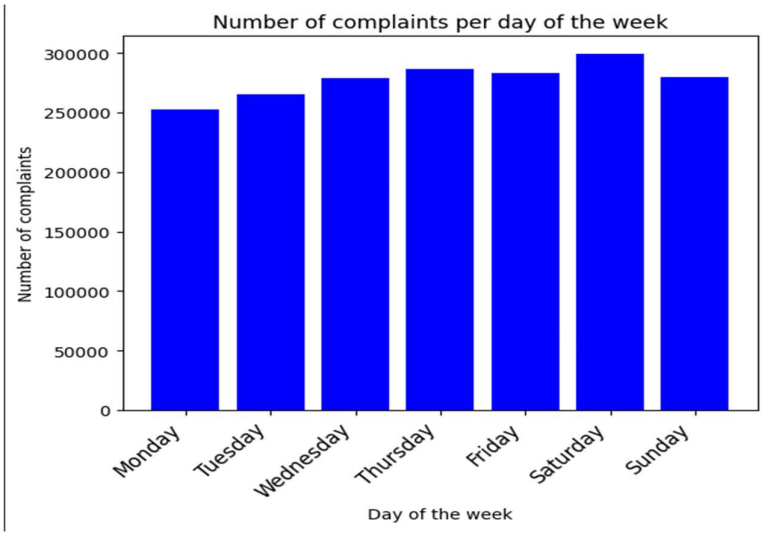
The following analyses are performed:

1. Counting the number of crimes reported in each borough: The dataset is aggregated to determine the count of crimes reported in each borough.



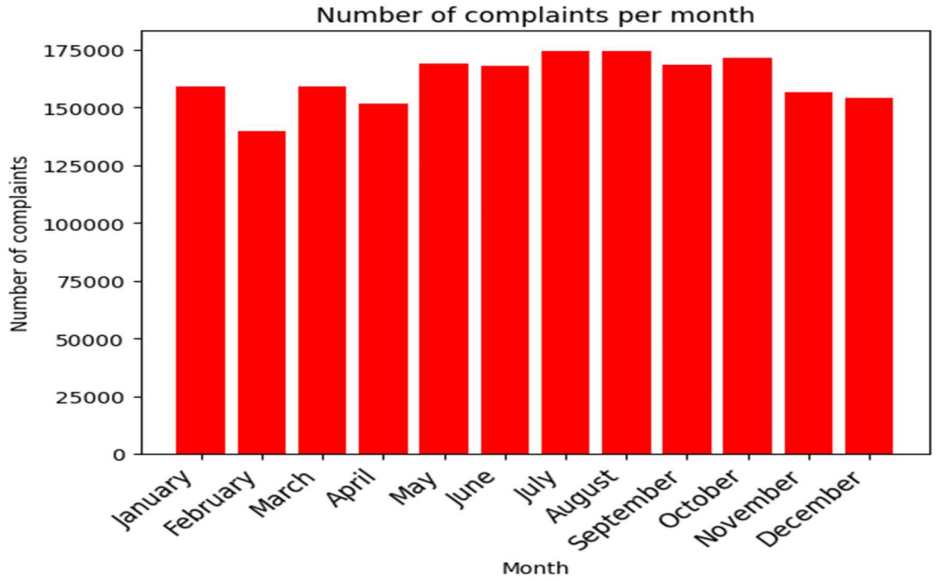
   - Based on the visualisation we can conclude that most crimes occurred in Brooklyn followed by Manhattan and the least crimes occurred in Staten Island.

2. Analysing the number of crimes reported per day of the week: The dataset is aggregated to analyse the number of crimes reported per day of week.



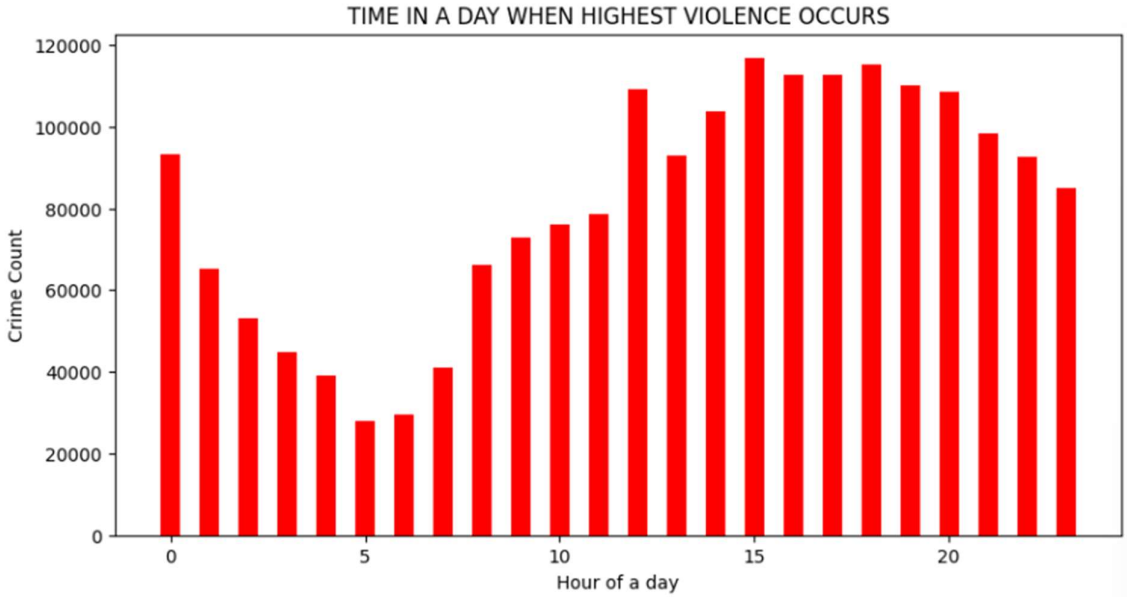
- Based on above diagram we can say that on average more crimes occur during weekends compared to weekdays.

3. Analysing the trends in number of crimes reported per month: The dataset is aggregated to analyse the number of crimes reported per month.



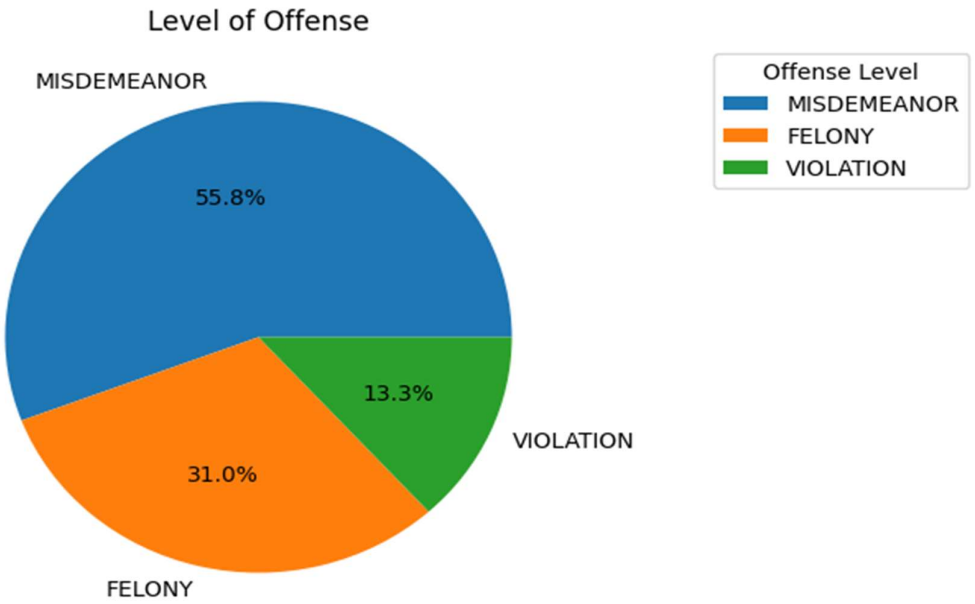
- Based on above chart we cannot conclude anything strong enough, but we can say that least crimes occur in the month of February.

4. Analysing the trends in Time in a day when highest violence occurs: The dataset is aggregated to analyse the Time in a day when highest violence occurs.



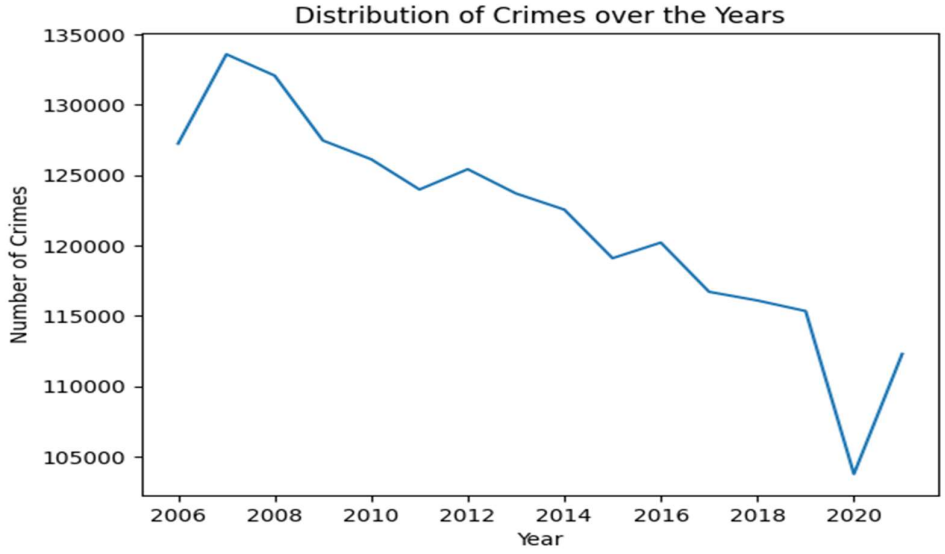
- Based on the graph we can see that majority of crimes happen during the hour 12-20, so we can say that there is less crime in the night, but there are more crimes happening in broad daylight.

5. Analysing the Level of Offence: The dataset is aggregated to analyse the Level of Offence.



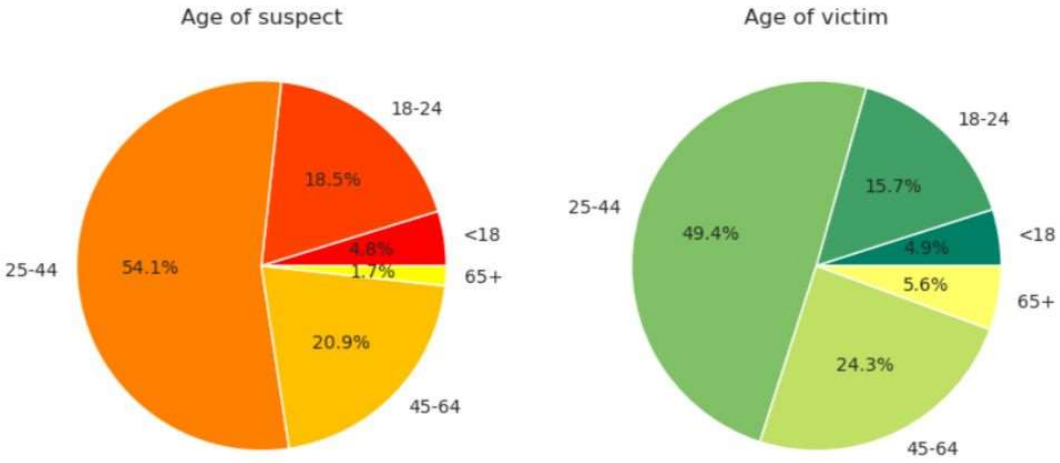
- From the graph above, we can tell that Misdemeanour, an offense of which a sentence in excess of 15 days but not greater than one year may be imposed, is the most popular level of crime. The second popular one is Felony, the most serious of offenses, and the third one is Violation, a lesser offense for which a sentence only be no more than 15 days.

6. Analysing the Distribution of Crimes over the Years: The dataset is aggregated to analyse the Distribution of Crimes over the Years.



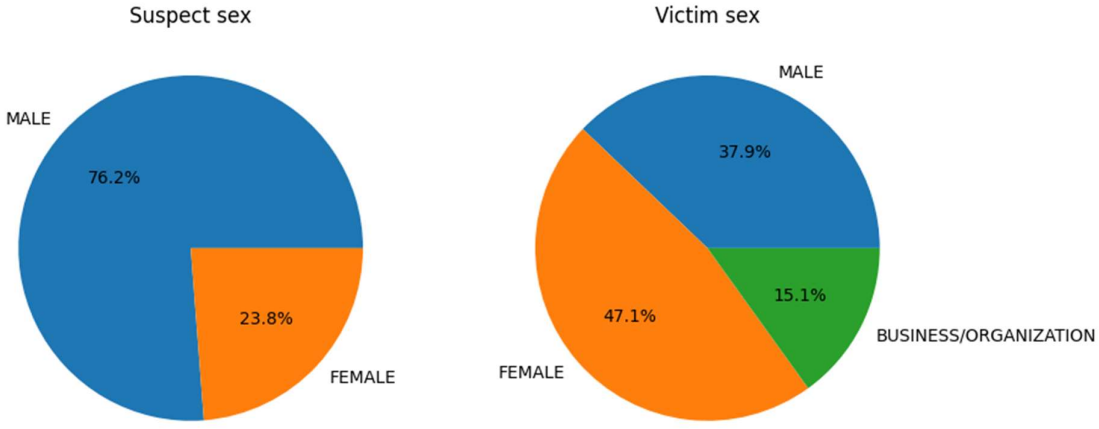
- Based on the above time graph, we can see a clear plummeting towards the end of the year 2019 and in 2020, which makes sense as COVID-19 caused a lockdown worldwide. We can also see that after covid there is a steep spike in the graph suggesting that crime rate increased rapidly. More data is needed after 2020 for more detailed and future analysis.

7. Identifying patterns in suspect and victim Age: The dataset is aggregated to analyse the Age of suspects and victims involved in reported crimes.



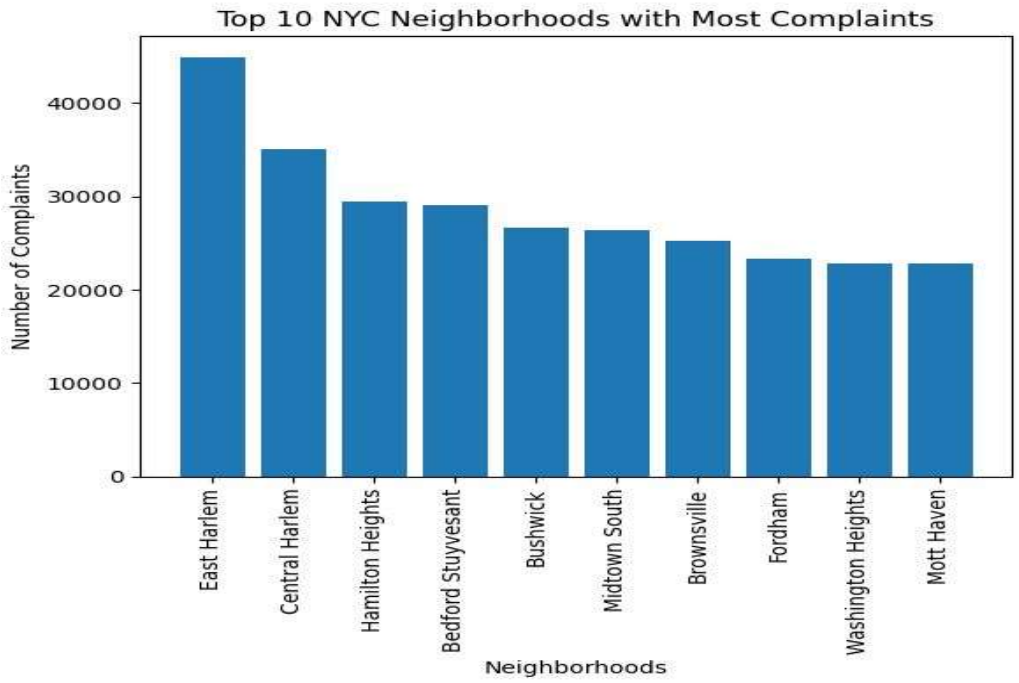
- Based on the above pie charts we can see that there seems to be a slight imbalance in age of suspects vs. victims, where suspects are younger than victims on average. We can also see that more than half of the suspects lie in the age group of 25-44. Whereas victims lie more in percentage compared to suspect ion other age groups.

8. Identifying patterns in suspect and victim Sex: The dataset is aggregated to analyse the Sex of suspects and victims involved in reported crimes.



- NYPD data only contains information on male and female as the biological sex of suspects and adds a 'Business/Organization' category for victims. While more than three quarters of all crimes are committed by males, it is females who are more likely to be the victim of a crime.

9. Identifying Top 10 NYC Neighbourhoods with most crimes: The dataset is aggregated to analyse the Top 10 NYC Neighbourhoods with most crimes.



- As we can see from the above graph the greatest number of crimes occur in East Harlem followed by Central Harlem, which are in Manhattan, but we saw before that most crimes occurred in Brooklyn. So, these neighbourhoods seem to be dangerous compared to other neighbourhoods of Brooklyn.

10. Created an interactive Map using the coordinates from the dataset and folium library making it easy for visualising. (Only took 100 rows for fast computation)

The results of each analysis are presented, and visualizations are used to provide a clear representation of the findings. Matplotlib and Folium libraries are utilized to create bar charts and maps for visualizing the results.

Limitations and Future Works

While this project has provided valuable insights into crime patterns in New York City, there are several limitations that should be acknowledged. Firstly, the analysis relies heavily on the quality and completeness of the NYPD Complaints dataset. Inaccuracies, inconsistencies, or missing data in the original dataset could affect the accuracy of the findings. Additionally, the analysis is limited to the timeframe of 2006 to 2020, potentially missing recent changes or emerging trends in crime patterns. Furthermore, the findings are specific to New York City and may not be applicable to other regions or cities with different crime dynamics. The merging process of the crime dataset with the NYC neighbourhood's dataset might have introduced errors or discrepancies, impacting the accuracy of crime-to-neighbourhood associations. Moreover, the dataset may lack comprehensive information on offender characteristics, which could limit the depth of understanding.

To address these limitations and further enhance the analysis, future work could focus on several areas. Incorporating real-time or near real-time data streams would allow for the capture of the most recent crime trends, enabling more timely interventions. Predictive analytics could be applied to develop models for forecasting crime hotspots, empowering law enforcement agencies to allocate resources effectively. Additionally, exploring the correlation between crime rates and social-economic factors like unemployment rates or poverty levels could provide a more comprehensive understanding of crime patterns. Comparative analysis with other cities or regions would offer valuable benchmarks and insights into unique characteristics or challenges specific to NYC. Integration of additional data sources, such as emergency service response times or demographic data, could provide a broader context for crime analysis. Furthermore, employing advanced spatial analysis techniques and visualization methods would facilitate a deeper understanding of crime dynamics and aid in targeted interventions. By addressing these limitations and pursuing future work, the analysis can continue to evolve and provide more accurate and actionable insights for crime prevention and law enforcement efforts in New York City.

Conclusion

In conclusion, this project successfully leveraged PySpark, MongoDB, and various Python libraries to analyse the NYPD Complaints dataset and gain valuable insights into crime patterns in New York City. Through data profiling, cleaning, and merging processes, the dataset was prepared for in-depth analysis. The integration of geospatial data allowed for the identification of crime patterns within specific neighbourhoods, enabling targeted interventions. The project also employed data visualization techniques to effectively present the findings, providing clear and informative visual representations of crime trends.

The analysis revealed several notable findings. Brooklyn emerged as the borough with the highest number of reported crimes, followed by Manhattan. Weekends showed a higher average crime rate compared to weekdays, and there was a slight decrease in crime during the month of February. Additionally, crimes were found to be more prevalent during daylight hours, specifically between 12

PM and 8 PM. Misdemeanour offenses were the most common, followed by felony and violation offenses.

The project also examined suspect and victim characteristics, highlighting a slight imbalance in age distribution between suspects and victims, with suspects generally being younger. The analysis of the top 10 neighbourhoods with the most crimes identified East Harlem and Central Harlem in Manhattan as areas with higher crime rates.

It is important to acknowledge the limitations of this analysis, including the reliance on the accuracy and completeness of the NYPD Complaints dataset, the limited timeframe of the data, and the specificity to New York City. Future work could address these limitations by incorporating real-time data, exploring the correlation between crime rates and socio-economic factors, conducting comparative analysis with other cities, and integrating additional data sources.

Overall, this project demonstrates the power of big data analytics in uncovering valuable insights from vast datasets. The findings can inform decision-making processes related to crime prevention and law enforcement in New York City, contributing to more effective strategies and targeted interventions. By continuing to address limitations and pursuing further research, this analysis can evolve and provide even more accurate and actionable insights for enhancing public safety in the city.