Police Incident Patterns and Response Time Modeling in Montgomery County, MD

The Washington Post [1]

Introduction and Motivation

As a long-time resident of Montgomery County since my sophomore year of high school, I have often wondered about police activity in my neighborhood. Every day, police dispatch centers in Montgomery County, MD, receive hundreds of varying calls, from petty disturbances to life-threatening emergencies. The time it takes to dispatch officers and their arrival on the scene can make a significant difference between resolving the incident and a critical situation. This project aims to understand police response times and the final disposition of calls. With the various dispatch data available, including location, call type, priority, time, etc., I hope to identify patterns that will help determine optimal staffing and resource allocation for a public safety initiative.

Project Significance and Goals

In this project, I will evaluate the **Police Dispatched Incidents**[2] dataset from Montgomery County, Maryland. The police dispatch incidents dataset contains a lot of time based information about each incident call in Montgomery County, including timestamp, geocoded address, call type, priority, response time, and description. With response time as a continuous variable, low and high priority as binary variables, and disposition code as a multi-class variable, I would be able to analyze police incident patterns and build a model of response time to support future emergency response planning.

Project Goals:

1. **Data Collection** – Fetch October 2025 incident records from the Montgomery County open data API.
2. **Data Cleaning** – Parse numeric fields, handle missing values, handle duplicate values, etc.
3. **Exploratory Data Analysis** – Create visualizations to understand the distribution of different call type, priority, timing, and geography.
4. **Unsupervised Learning** – Apply feature selection to the data and apply clustering algorithms to group police incidents.
5. **Supervised Learning** – Define targets for regression, binary classification, and multi-class classification.
6. **Reporting and Storytelling** – Summarize exploratory data analysis findings and modeling to create a report for non-technical audiences.

Research Questions

1. **What are some factors that drive dispatch-to-arrival time?**
   What are the combinations of original call type, priority, location, and time of day that are correlated the strongest with response times?
2. **How do response times vary by time during the day or days in a week?**
   Are there specific displays of dispatch and arrival times for specific call types, and do we observe a pattern on a day of the week?
3. **Are there distinct spatial clusters of incident types and response times?**
   Are there specific areas within Montgomery County that consistently have similar comments, priorities, and response times?
4. **Can we predict whether a call will be classified as high priority?**
   Is it possible to predict a call's priority level based on other information, such as the original call type, location, and timing metrics?
5. **How accurately can we forecast the final disposition of an incident?**
   Is it possible to predict the final call disposition based on the original call type, priority, location, and response times?

Literature Review

1. The literature review section identifies what previous researchers have done and examines the data, methods, and findings of their work. To speed up the literature review, LLM tools are used in this section to better understand the following papers[1].

Predictors of Police Response Time — A Scoping Review[2]

Objectives:
The research document uses empirical studies about police response duration to study response time variables through spatial analysis of police response duration. The authors combine response time elements to establish relevant variables that impact response time while identifying the unclear factors that make response time evaluation challenging. The authors combine new research with outdated and insufficient data to establish specific variables for response-time assessment by organizing existing knowledge.

Data and Methods:
The authors examine 39 empirical studies written in English that detail how response time varies with certain predictors. Given the large volume and diversity, the authors decided to use a narrative synthesis approach tailored to areas with large, complex bodies of literature where straightforward qualitative reviews are inappropriate, and meta-analyses cannot be applied. The authors report 630 statistical relationships involving 122 different predictors, which they systematically classified into different levels of a hierarchy. The authors detail their process and have uploaded all predictor datasets to GitHub to facilitate further validation of their results.

Key Findings:
This review particularly underscores factors relating to police response time and emphasizes them as among the oldest, most limited in scope, and least well documented; therefore, no causal conclusions can be drawn. Instead of estimating causal impacts, the authors take this as an opportunity to draw a lesson. They encourage more precise work in the future on the underlying mechanisms of the delays, on differentiating the precise elements of reporting, response, and travel time to construct a meaningfully formulated response, and on the use of systemic terminology for the variables and outcomes of interest. They note the reporting phase as a weakness in the literature as well and have called for greater attention to this gap. The review attempts to provide, in the literature, a first-pass account of what could be understood causally, while being clear that this is merely observational; therefore, they have approached the predictor variables in a somewhat mythical sense.

The Effect of Police Response Time on Crime Clearance Rates[3]

Objectives:
This paper challenges the popular belief that police response time relies heavily on crime clearance outcomes. The authors of this paper aim to quantify how much a response time delay shifts an offender's probability of apprehension or of a crime being solved, to inform understanding of the operational significance of response time in policing.

Data and Methods:
The study uses a detailed dataset obtained from the Greater Manchester Police that records responses to crime incidents. To identify causal effects, the authors employ a boundary-discontinuity approach that leverages abrupt changes in distance to police-division boundaries—adjacent neighborhoods served by different divisions with systematically different response times. This experimental design provides variation in response times, allowing the authors to estimate how a 10% increase in response time affects the likelihood that a crime is cleared.

Key Findings:
The outcomes show that the longer the response time, the less likely it is to clear the crime. The authors show that with

each additional 10% of response time, the likelihood of clearing the crime lessens by 4.7%. This negative effect is especially pronounced for theft offenses; however, it is still noteworthy for all other offenses. More rapid response times also increase the chances of on-the-spot arrests, and the odds are raised that the victim and/or witnesses will be able to name the perpetrator.

What Conditions Affect Police Response Time? Examining Situational and Neighborhood Factors[4]

Objectives:
The aim of this analysis is to examine individual call attributes and the more integrated features of the adjoining areas. The goal of the authors is to assess whether the quicker or slower police officer response is due to specific characteristics of the call (i.e., the presence of a weapon, day and time of calls, or complainants' characteristics) and/or is due to a community-level correlate, such as socioeconomic disadvantage or residential instability. This inquiry goes beyond the response-time norm and seeks to determine whether response time is due to call or community attributes.

Data and Methods:
The authors examine neighborhood effects at census-tract levels by analyzing domestic dispute calls to a large police department in Texas. To appropriately handle the nested structure of calls by neighborhood, the authors apply multilevel (or hierarchical) modeling to adjust for call prioritization, neighborhood, and situation variables simultaneously. This focuses on the complainant's race and weaponization, response time, and other situational characteristics, as well as the variables of concentrated disadvantage, immigration density, and residential turnover.

Key Findings:
This research shows that several factors affect police response time, including whether a weapon is present, the time and day, and the ethnicity of the complainants. Response times are fastest for Hispanic complainants. Geographic factors like clustered disadvantage, immigration saturation, and residential turnover suggest that social structure is important for understanding response times. However, call volume and population diversity do not seem to predict response time. The data also shows a lot of variation between census tracts, so it is important to consider both incident-specific and contextual factors when measuring police effectiveness.

After conducting several literature reviews, I did not find any academic studies that focus on police incident patterns and response time modeling. Accordingly, I have decided to proceed with my own research on police incident patterns and response-time modeling in Montgomery County, MD.

Audience

This site is my DSAN 5000 final project for the Data Science and Analytics Program at Georgetown University, and it also targets data professionals, academics, and public policy professionals interested in the public-safety analytics sphere in Montgomery County, MD, and the surrounding areas. It combines the concepts of exploratory data analysis, predictive modeling, and data storytelling to demonstrate the value of data-science methodologies and how they can substantially improve our insights into the workings of the police, including the decisions made and the processes employed.

About Me

I am currently pursuing a Master's in Data Science and Analytics at Georgetown University with a concentration in Artificial Intelligence, building on my experience in HR analytics, business intelligence, and AI solutions at

Mercedes-Benz, Farfetch, and Forbes. I am passionate about using data, analytics, and AI to drive innovation, efficiency, and strategic impact in industries such as automotive, enterprise transformation, and advanced analytics. If you'd like to learn more about my background and experiences, please visit my full bio HERE.

References

1. ChatGPT, version-5.1, OpenAI, NOV-2025, chat.openai.com.
2. Verlaan, T. & Ruiter, S. Predictors of police response time: A scoping review. *Crime Science* **12**, 1–23 (2023).
3. Blanes i Vidal, J. & Kirchmaier, T. The effect of police response time on crime clearance rates. *Review of Economic Studies* (2017).
4. Lee, J.-S., Lee, J. & Hoover, L. T. What conditions affect police response time? Examining situational and neighborhood factors. *Police Quarterly* **20**, 1–26 (2016).

Footnotes

1. The Washington Post: Montgomery Co. police data collection is 'woefully inadequate,' audit says ↩
2. Montgomery County Open Data Portal – Police Dispatched Incidents ↩

DSAN 5000 - Incident Response Modeling (MCMD)

Police Incident Patterns and Response Time Modeling in Montgomery County, MD

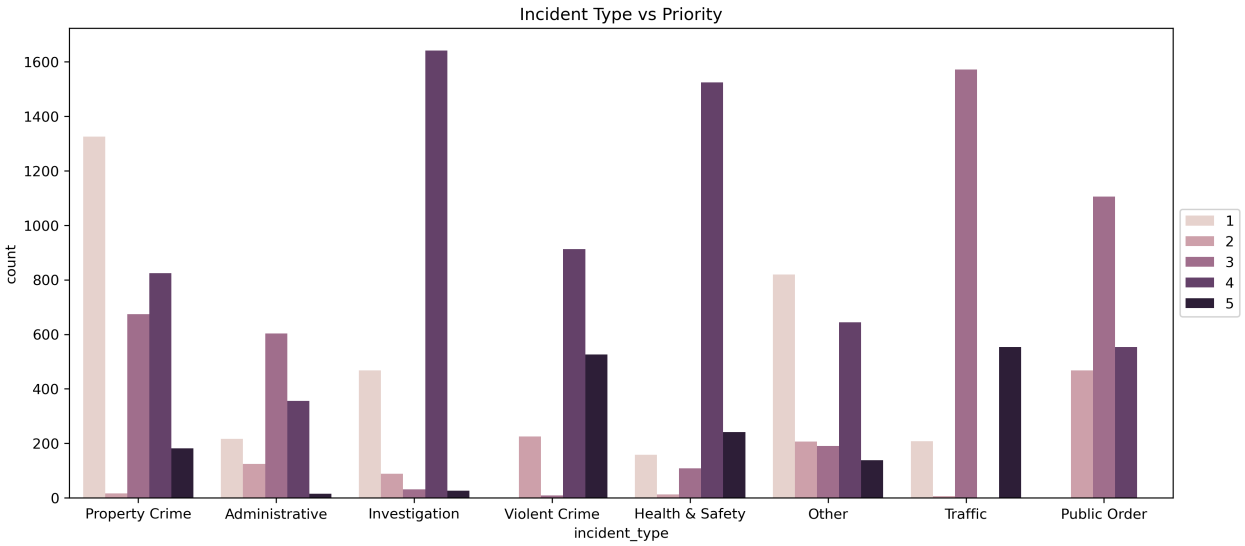
Introduction

Every day, in Montgomery County, the police dispatch center receives various incident calls from minor disturbances, such as traffic-related incidents, to life-threatening emergencies such as violent crimes. As a Montgomery County resident, it is natural to wonder how quickly police officers arrive on the scene and how responses differ by neighborhood, time of week, and incident type. The time between a call and officers' arrival at the scene can change the outcome of an event, especially in serious cases.

WUSA9 [1]

In this project, I will study police-dispatched incident records from Montgomery County, Maryland. The focus of this research is on patterns in response time and on the final outcome of incident calls. The police incident dataset contains detailed information, including location, call type, priority level, various timestamps, and police district, that describes the flow of incident calls from initial contact with dispatch centers to incident closure. By examining these variables, this project provide a clearer picture of how local police respond across different locations in Montgomery County and at different times. More importantly, how those responses can inform public safety planning.

Prior research on police response time has looked at factors such as call characteristics, neighborhood conditions, and distance to patrol units. Recent reviews show that many studies rely on limited data and often cannot draw strong causal conclusions. At the same time, evidence from specific cities suggests that longer response times are linked with lower chances of solving crimes and making arrests. There is still a need for detailed case studies that connect incident-level data with day-to-day operations. In this project, I will evaluate this gap for Montgomery County, Maryland.

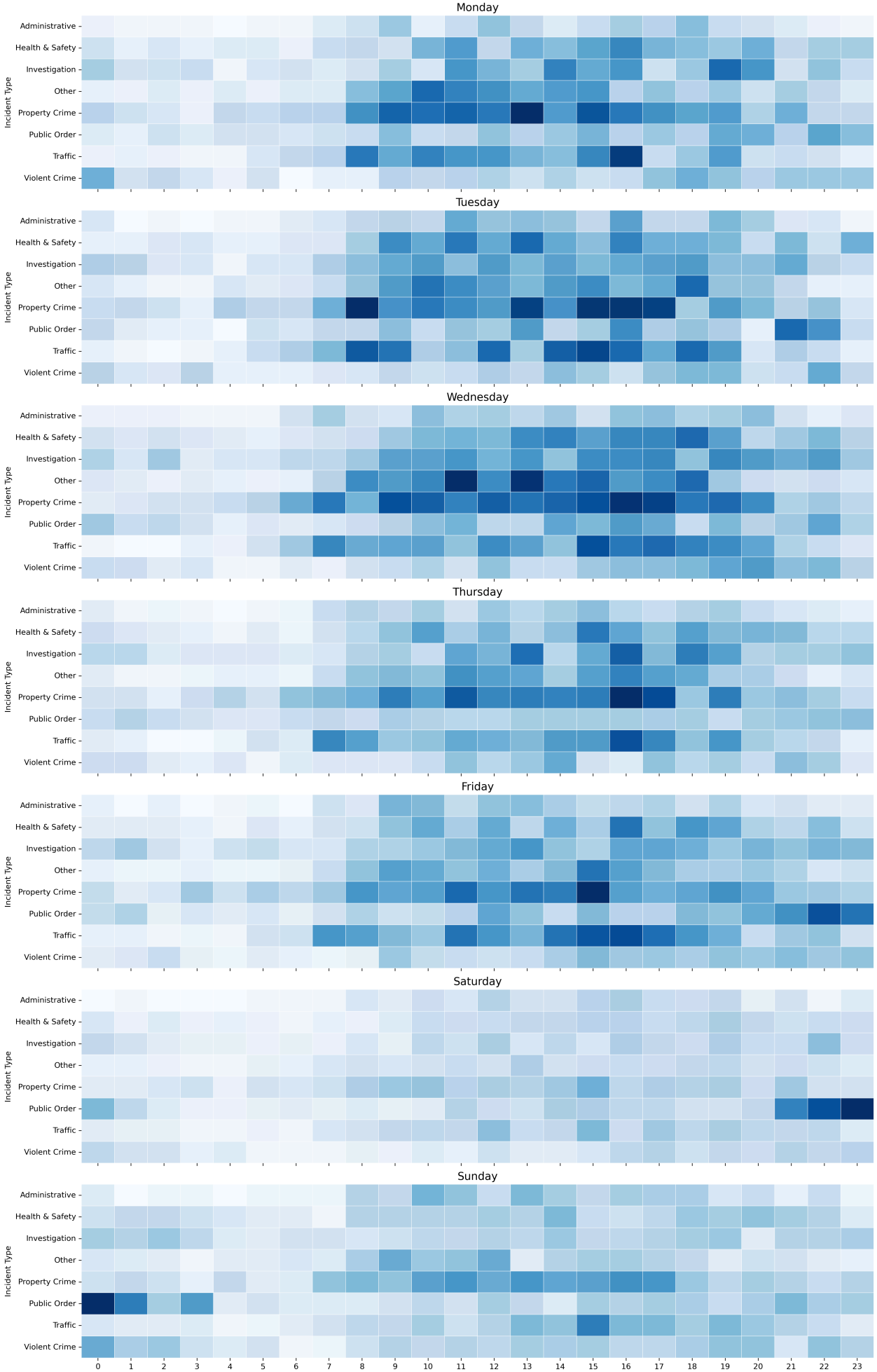


Incident Type by Priority Level

Objective

The main goal of this project is to understand how police incident characteristics relate to response time and call outcomes in Montgomery County. Using the Police Dispatched Incidents dataset, the project examines both patterns in the data and the potential of basic predictive models to support future emergency response planning.

More specifically, we will first examine which combinations of incident call type, priority, location, and time of day are most strongly associated with dispatch-to-arrival time. We will then investigate how response times vary by hour of day and day of week. With this topic, we will check if there are any distinct temporal patterns. Next, we will explore whether the incidents form natural spatial or temporal clusters based on their features. Finally, we will test whether simple models can predict high-priority calls, forecast final dispositions, and estimate response time based on the available variables.

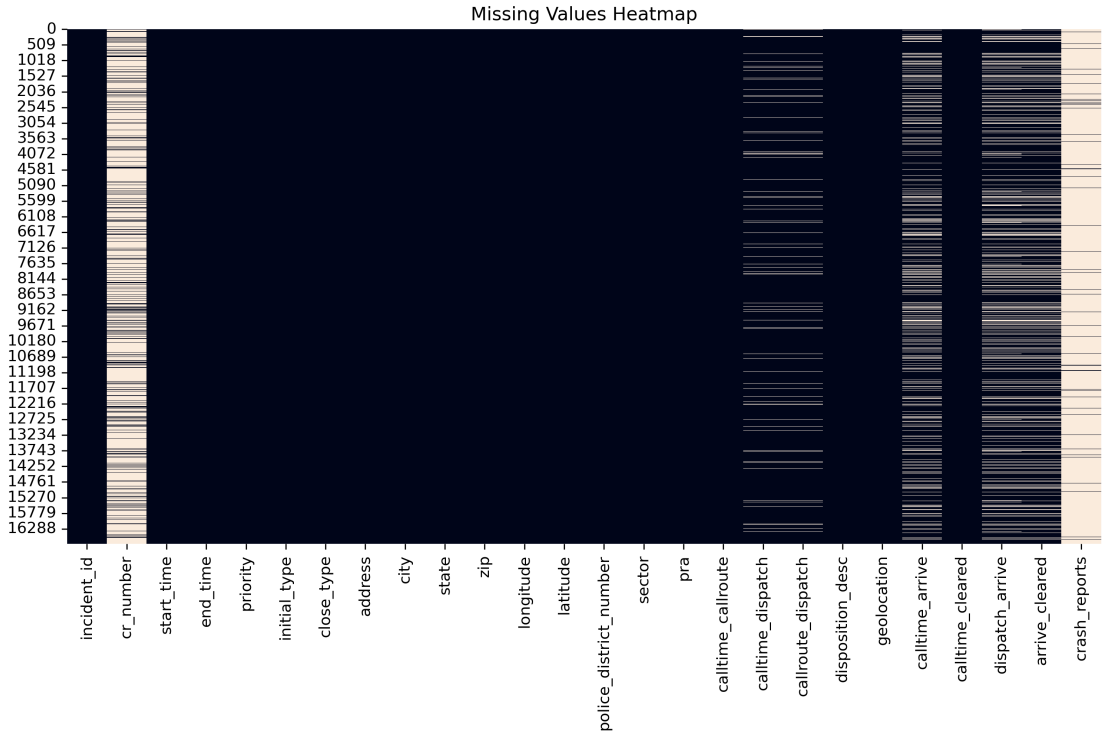

Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Incident Type
Administrative
Health & Safety
Investigation
Other
Property Crime
Public Order
Traffic
Violent Crime
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Incident Type by Hour of Day and Day of Week

Methodology Overview

Data Collection

In this analysis, I will use Montgomery County's Police Dispatched Incidents dataset from the county's open data portal. The extracted police incident data for October 2025 contains 16,782 records, each with 26 variables. These variables contain unique incident identifiers, multiple timestamps, initial and final classifications of the call, priority level, address, police district, and derived timing intervals, including call-to-dispatch time and dispatch-to-arrival time. In the police incident calls dataset, columns including incident ID, start time, and initial type are fully populated, while other columns, such as crime report numbers and several time interval fields, contain substantial missing values.



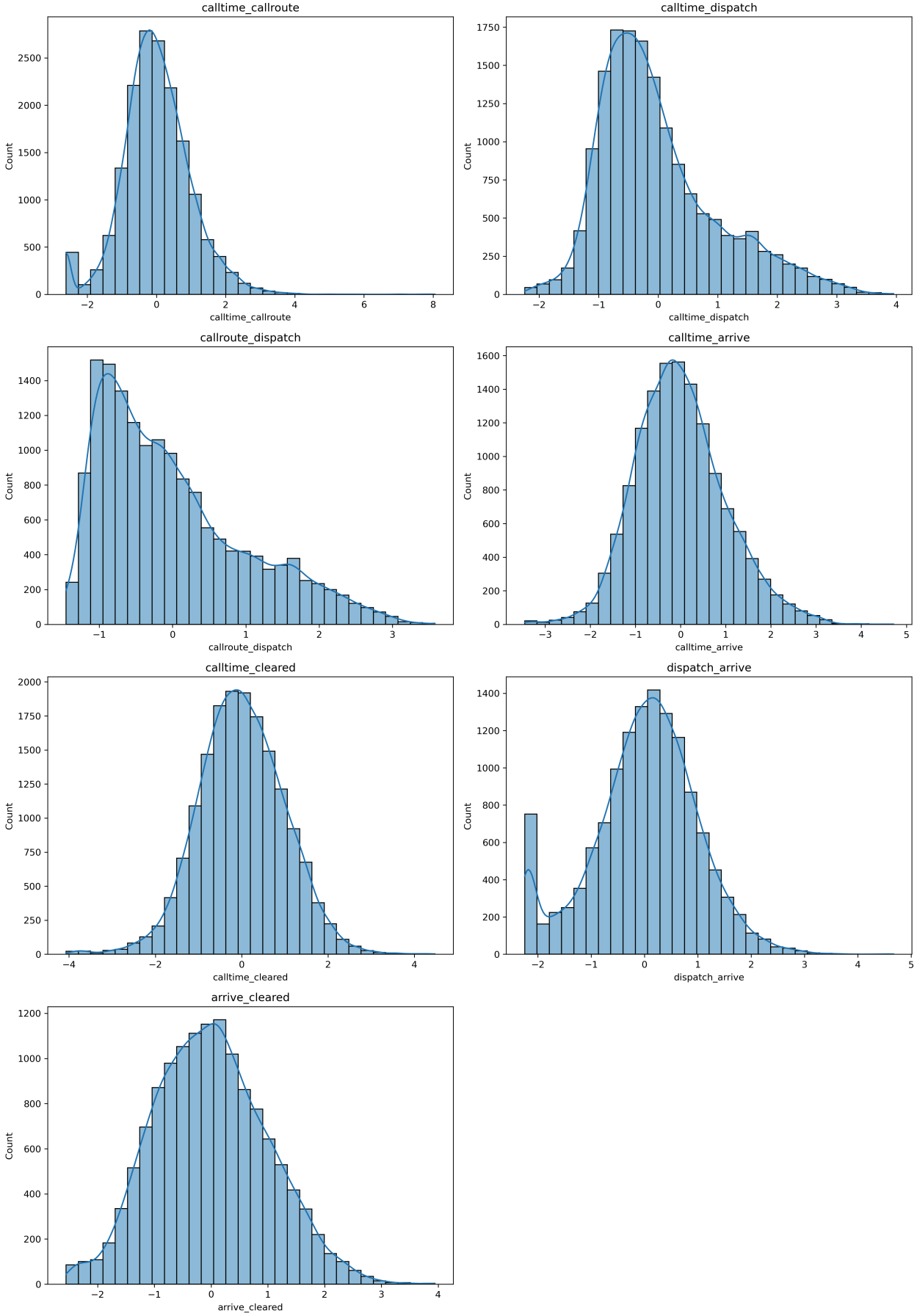
Police Incident Calls Dataset Missing Value Heatmap

Data Cleaning

The raw police incident call data are being cleaned into two main datasets during the data cleaning stage. The first is a cleaned dataset specifically for exploratory data analysis. In this step, timestamps are cleaned into proper dates, numeric fields are corrected, missing values are handled, and irrelevant columns are removed. Categorical variables, such as initial and closing types, are rephrased to improve readability. In the first step, I still kept the original structure and ranges, specifically for exploratory data analysis.

The second dataset is cleaned specifically for machine learning, including supervised and unsupervised learning. This step begins with the cleaned exploratory dataset and applies additional steps. In this process, numeric columns are z-score normalized, and categorical variables are encoded into numbers. Any remaining missing

values are removed so that models can train without errors. However, a major challenge in the data-cleaning stage is the large number of missing coordinates. To support spatial analysis, missing longitude and latitude values have to be filled. Luckily, the specific address of each incident is recorded in the data, so I was able to use the Google Geocoding API to obtain accurate longitudes and latitudes.



Distribution of Normalized Data

Exploratory Data Analysis

The exploratory data analysis examines both single variables and relationships between variables. For numeric features, summary statistics and histograms indicate that time-related measures, including call-to-dispatch or dispatch-to-arrival, are highly skewed to the right and exhibit long right tails. Most of the incidents are handled within a short period of time, while a small number take much longer, resulting in extreme values in the distribution.

The categorical variables, including incident call type, priority, and police district, are examined through frequency tables and bar charts. Crosstab tables and grouped bar plots are then used to study how incident types vary by time of day, weekday, and district. Correlation analysis of the standardized time intervals helps to reveal how different stages of the response process move together.

Unsupervised Learning

In the unsupervised learning stage, I focus on feature selection, dimensionality reduction, and clustering. Dimensionality reduction methods, including principal component analysis and t-distributed stochastic neighbor embedding, project the data into two dimensions, which makes it easier to visualize relationships among different types of incidents. Clustering algorithms such as the K-means, density-based clustering, and hierarchical clustering are then applied to these representations with different parameter settings and distance functions. The goal of this is to determine whether the incidents form natural groups based on features such as timing and priority.

Supervised Learning

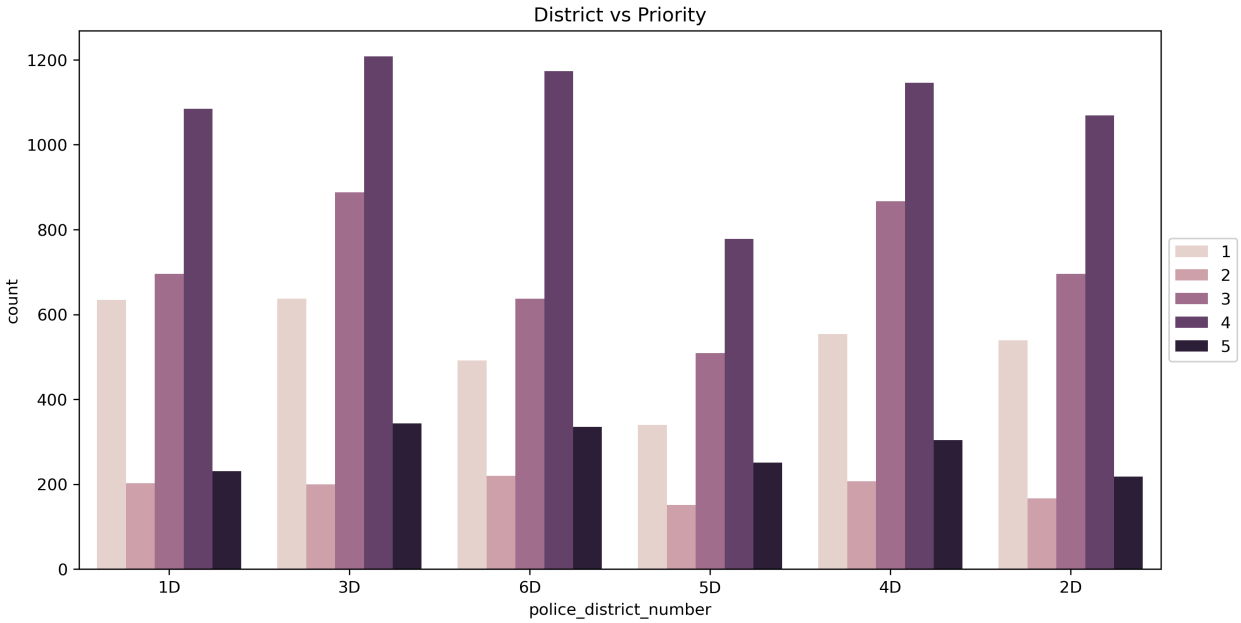
The supervised learning stage builds models with three different targets. The first is a binary classification method that predicts whether an incident is high or low priority. The second is a multi-class classification model that predicts the final disposition code of an incident. The last one is a linear regression that predicts the dispatch-to-arrival time as a continuous outcome. For the regression analysis, it starts by predicting the mean standardized response time. It is then compared to more complex models such as random forests and gradient boosting regressors. The input features used in this process include timing intervals, incident type, priority, and district, and the model performance is evaluated using classification accuracy value, recall value, and error measures.

Key Findings

Patterns in Incidents and Response Times

The exploratory analysis shows that the time related variables are all positively correlated. Incidents with long times in one stage of the process, such as call to dispatch, also tend to have long times in later stages, such as dispatch to arrival and arrival to clearance. This suggests that delays accumulate across the workflow rather than occurring in isolation.

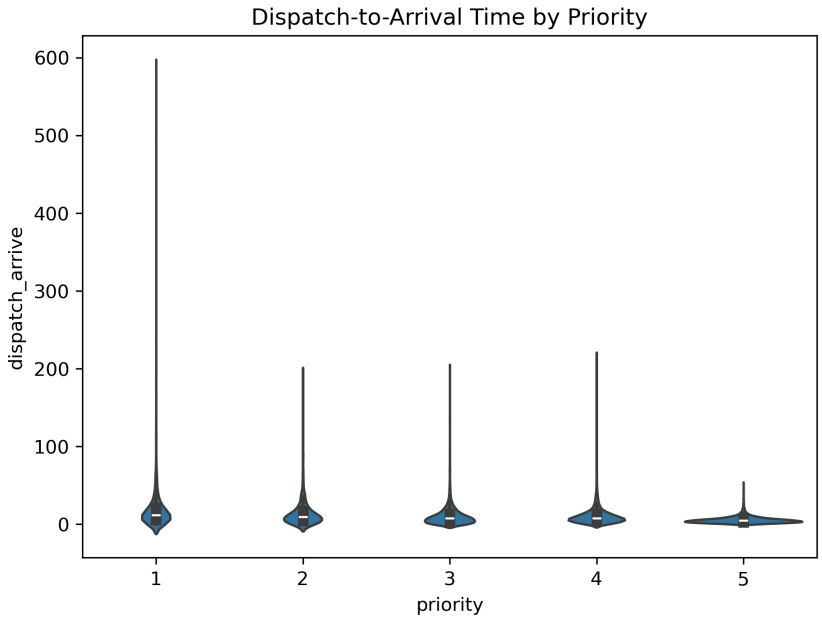
Looking at the different incident types, property crime is the incident type that occurred the most. Property crime is also prominent among high-priority calls, along with violent crime, health and safety, and traffic incidents. On the other hand, some low-priority calls tend to involve less urgent issues. The grouped bar chart of incident types by police district below shows that certain districts, such as the Silver Spring (3D) area, have particularly high counts of property incidents, which points to local variation in workload.



District Police Incidents by Priority

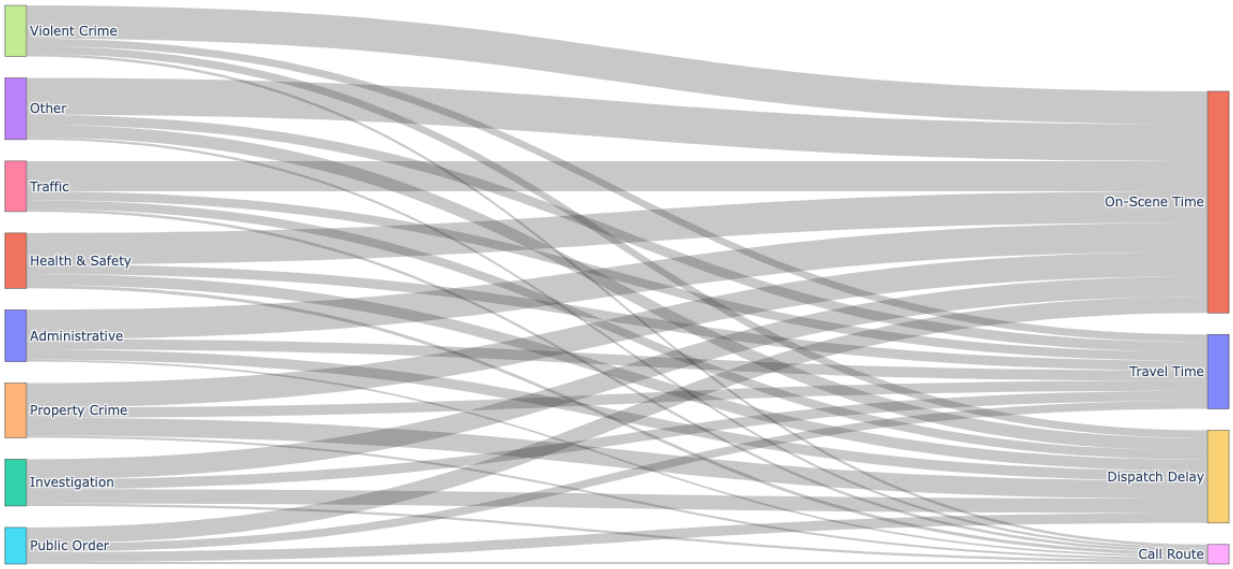
Temporal patterns are also pretty obvious. When incidents are grouped by hour of the day and incident type, most types peak around four o'clock in the afternoon. Public order incidents are an exception, with peaks late at night and especially on weekend evenings. Traffic incidents are more common on workdays. When grouped by weekday, total incident counts tend to be higher in the middle of the week, from Wednesday through Friday.

However, response time differs across these dimensions. Incidents that occur at night often have shorter dispatch-to-arrival times on average than daytime incidents, but the total number of incidents that occur is lower at night. Also, high-priority calls tend to have shorter response times and fewer extreme delays than the low-priority category. Intermediate priority levels show similar distributions and share some extreme outliers. Across districts, median response times are broadly similar, though a few have more very long dispatch-to-arrival times than others.



Dispatch-to-Arrival Time by Priority

The Sankey diagram below highlights the time it spends in the call process, dispatch delay, and on-scene as key components of the total duration from call to clearance for police incidents. From the results, we can tell that the dispatch delay contributes a lot to long total handling times, given that police dispatch delay should normally be pretty short. Also, some incidents remain open for a very long period after officers arrive on the scene. These visualizations support the view that operational delays can occur at multiple stages, not just during travel to the scene.

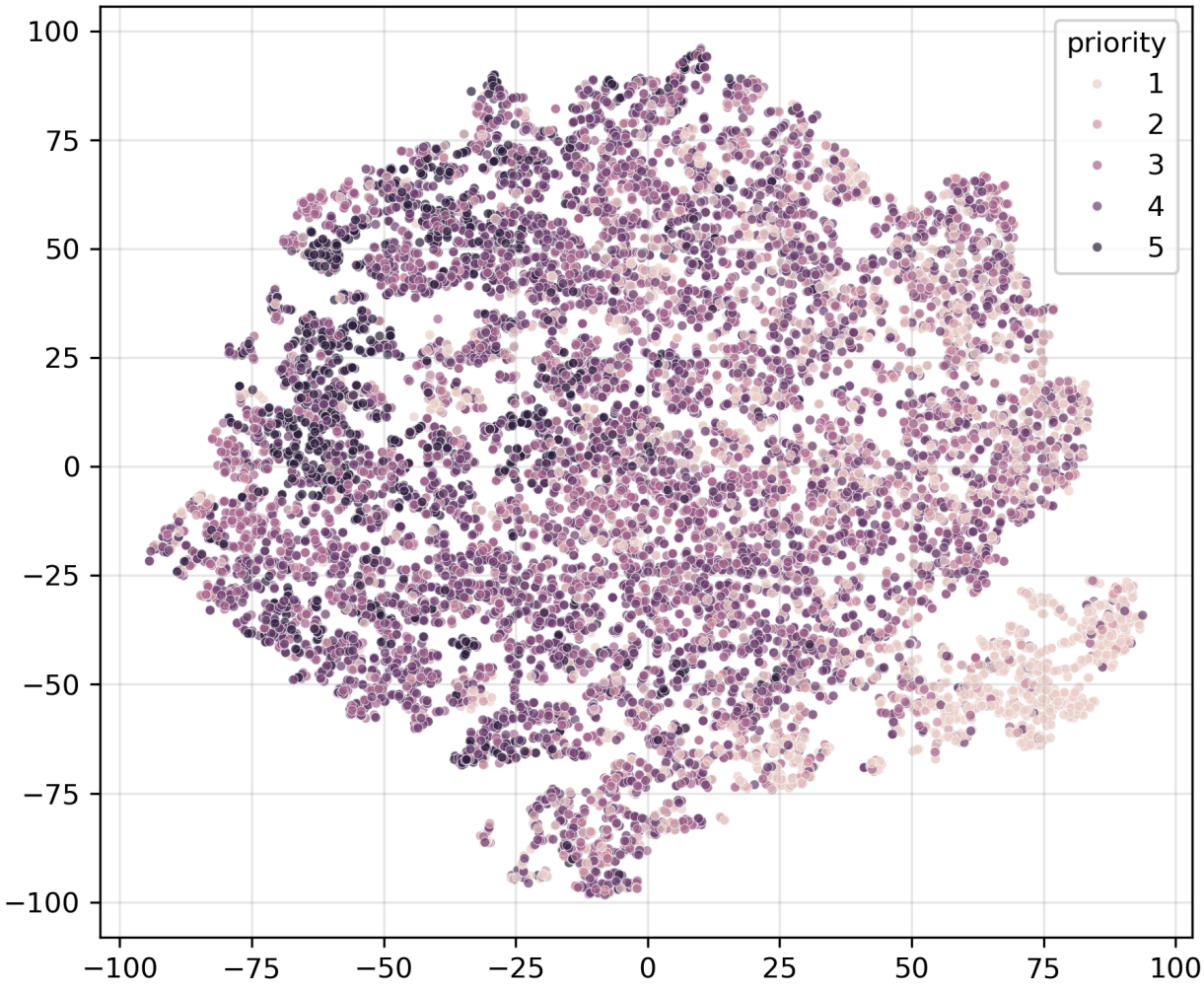


Sankey Diagram of Call Route, Dispatch, Arrival, and Clearance Stages

Unsupervised Learning Results

The dimensionality reduction plots show that all police incidents form a single and dense cloud with smooth transitions between types and priorities. The different clusters are not well separated groups when looking at the two main components. Clustering algorithms struggle to find distinct, interpretable, and stable clusters. Instead, the data form a continuous spectrum. Incidents that are close in the reduced space do not always share the same type, priority, or outcome.

The attempts to cluster on different subsets of features and with different distance measures leads to a similar conclusion. While algorithms can force a partition of the data, the boundaries between clusters do not reflect sharp differences in police operations. Instead, they often cut across incident types or split regions of the cloud without a clear operational meaning.



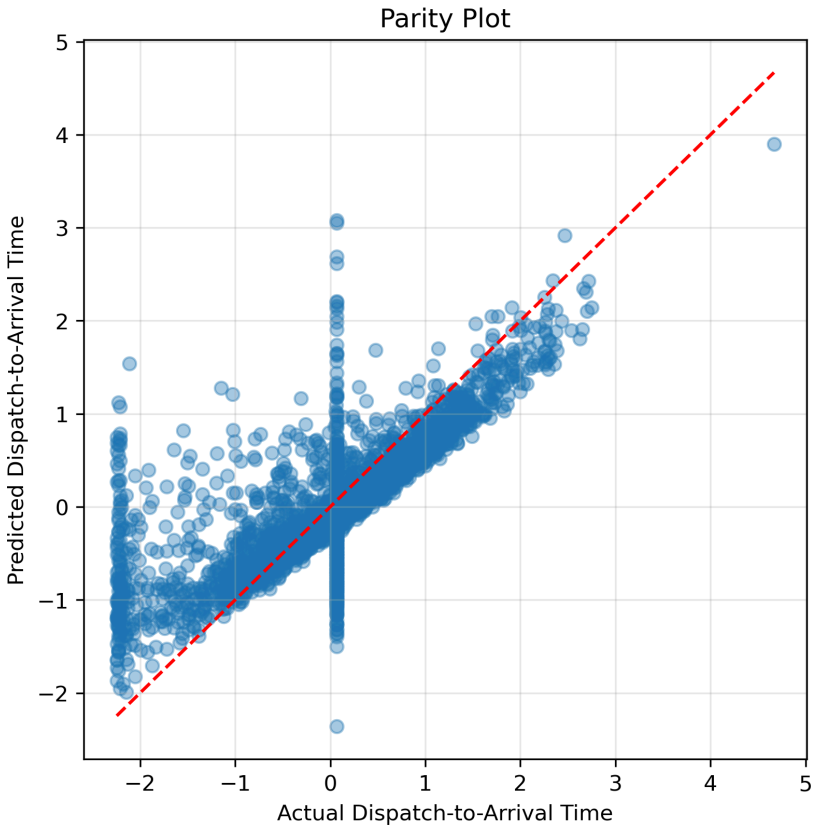
t-Distributed Stochastic Neighbor Embedding Result

Supervised Learning Results

For the binary priority classification method, models such as logistic regression and random forest reach reasonable accuracy and recall for high-priority incidents. These models show that priority can be predicted using incident details and call time features. However, many calls lie near the decision boundary, and many high-priority calls look similar to low-priority ones based on the features used.

For the multi-class disposition prediction, the models did better on common disposition codes. Random forest and gradient boosting capture part of the variation in final outcomes but leave a great number of misclassifications, especially for infrequent codes. This result reflects the unbalanced nature of the data and suggests that a richer context, such as narrative fields or neighborhood-level variables, would be needed to improve performance.

The regression models for dispatch-to-arrival time provide a similar story. A simple baseline that predicts the average standardized response time already achieves moderate accuracy. More complicated models, including random forest and gradient boosting, reduce errors but still leave a large amount of unexplained variation. They are better at capturing broad patterns than predicting the exact response time for each individual incident.



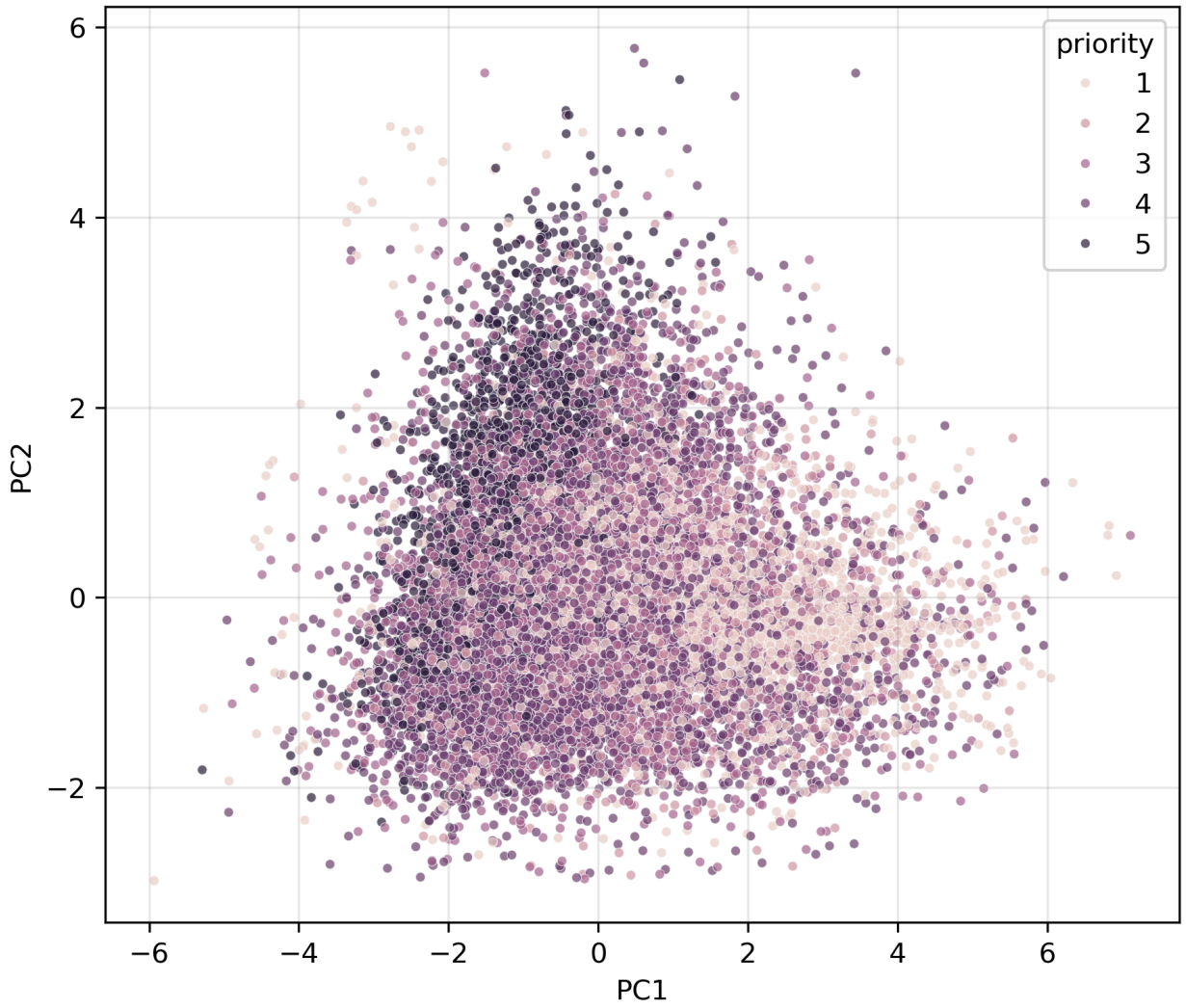
Predicted versus Actual Standardized Dispatch-to-Arrival Time

Societal Implications

In this research, we've looked at the distribution of police incidents by incident type, call time, and different areas of the county. We've identified areas with high counts of property crime, peaks in incident volume during late afternoons, and high volumes of public order incidents on weekends. The findings above could help adjust the police staffing levels and patrol areas to better align with the observed patterns.

In most incident calls, police officers arrive in a very short time, which is also why the data visualizations of all different call times are skewed to the right. However, on the other hand, we observed data with very long delays. We've concluded that this might be driven by dispatch delays. Understanding when and where these delays occur in the process can help to reduce them, improving safety outcomes.

In the next part, several clustering results all concluded that the police incident call times don't have clearly separate groups. It is very likely that the response times are driven by specific factors unique to each incident, which previous research has also concluded.



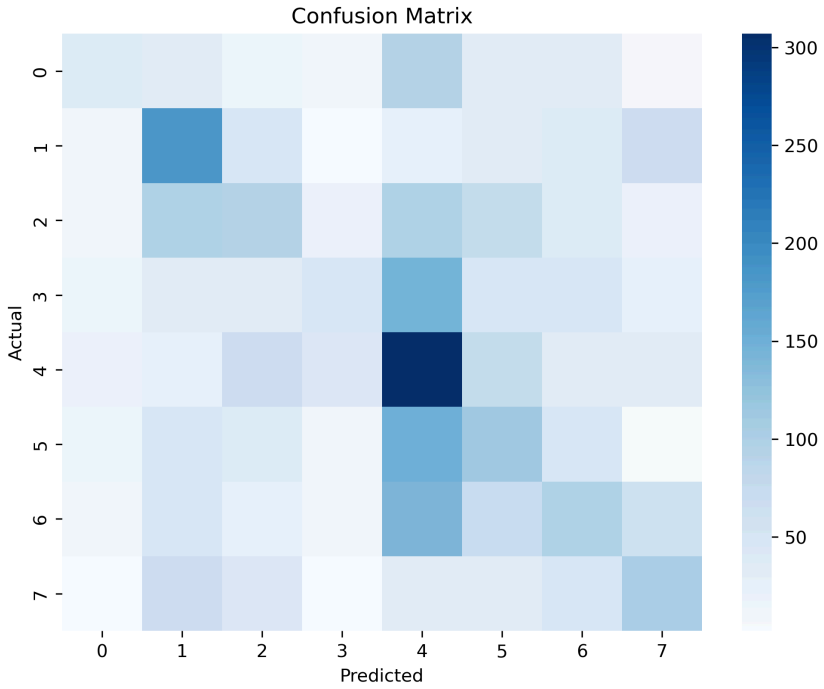
Principal Component Analysis

At the same time, the supervised models demonstrate that some aspects of police work can be supported by predictive tools, but also that these tools should not be treated as decision makers on their own. They are most useful for summarizing trends and highlighting potential high risk calls rather than for replacing human judgment.

Call to Action

Police managers and data analysts in Montgomery County can take several steps based on this project. Regular checks of incident patterns by hour of day, day of week type, and district can help match staffing and deployment to real demand. Areas with steady levels of property crime or traffic incidents may need to be focused on for patrols. Time periods with many incidents and slow response times may show that shift schedules or dispatch procedures should be adjusted.

Focusing on stages where delays build up can guide more targeted actions to reduce dispatch-to-clearance time. These stages include dispatch queues and long on-scene times. Looking closely at call routing rules and the handling of complicated incidents may help uncover the sources of slowdowns. Small changes in these parts of the process can create clear improvements in the overall response time.



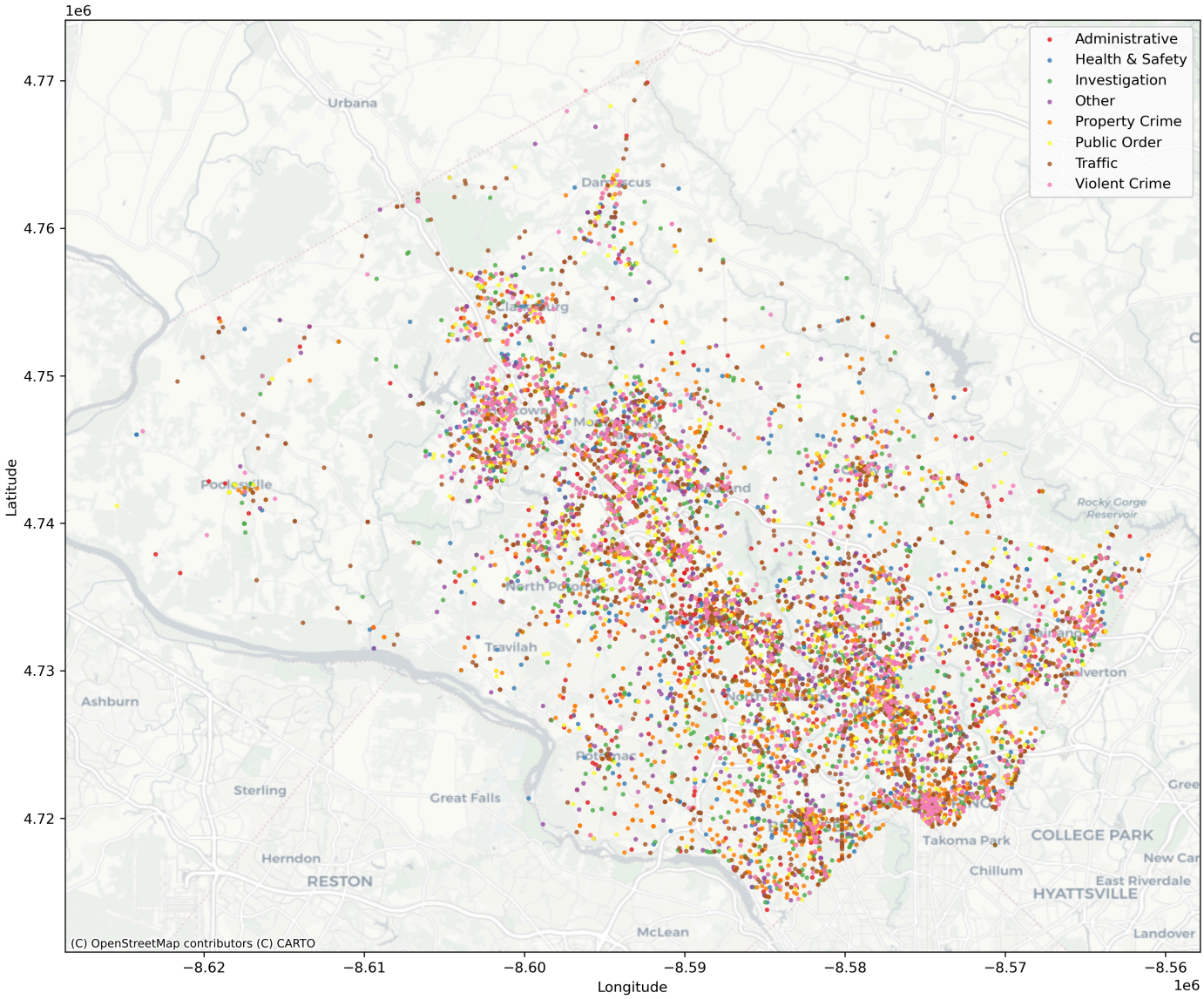
Multiclass Logistic Regression Confusion Matrix

For data teams and policymakers, this research demonstrates the importance of detailed police incident data. The models' limitations emphasize the need for richer features, such as several text fields and neighborhood markers. Future work can link the current incident data to census records to build a deeper understanding of how police response times behave.

Conclusion

This project brings together the full data science process from data collection to data cleaning to exploratory data analysis to unsupervised learning to supervised learning. The goal is to study police incident call patterns in Montgomery County Maryland. The work starts with a large dataset and turns it into forms that are ready for analysis. It is then prepared for machine learning and used in statistical and machine learning methods to answer questions about police incident calls.

Incident patterns and response times change by type of incident time of day and location. Property crime and traffic incidents make up a large part of the workload and there are strong peaks by hour and weekday. At the same time the county does not break into simple groups based on incident features or response times. Predictive models can offer helpful signals for priority and response time yet they cannot capture the full complexity of real world policing and should be viewed as tools that support decisions rather than final decision makers.



Overall, this research shows that detailed data work combined with exploration and modeling can improve our understanding of public safety systems. By refining the data, adding missing context, and trying new models, future work can build on these results and support the design of more effective and fair emergency response strategies in Montgomery County.

Footnotes

1. Montgomery County sheriff's deputy will not be charged in shooting death of man who struck him with a large piece of wood, authorities say ↩

DSAN 5000 - Incident Response Modeling (MCMD)

Data Collection

Overview

The following section explains how I will collect incident-level records from the Police Dispatched Incidents dataset using Montgomery County's open data API. In a data scientist's workflow, reliable raw data is the foundation of everything, as all following steps, including data wrangling, analysis, and modeling, depend on the quality of the raw data. In the details below, I will outline how I will set up and run the API query, organize and save raw files, check the schema, and record counts.

Methods

Data Source Information

Information utilized in this project is from the Montgomery County, Maryland Open Data Portal, specifically the dataset "Police Dispatched Incidents[1]" (dataset ID 98cc-bc7d). This dataset consists of about 1.76 million records of police dispatch calls, containing their timestamps, locations, nature of the calls, and duration of responses. This dataset is publicly available and can be downloaded from the Open Data Portal and from the Socrata API in the JSON format. Since I won't be needing all 1.76 million records, I will be only using data from October 2025.

Data Collection Methods

To collect data, I used the Socrata Open Data API to query the dataset, from the API endpoint: `https://data.montgomerycountymd.gov/resource/98cc-bc7d.json`. To get only records of October 2025, I added a SoQL `where` clause to filter the `start_time` field for that month. I also added a `limit` clause and set it to a large limit to ensure I receive all data of October 2025. By using the `requests` library, I sent an HTTP GET request to the endpoint mentioned above and parsed the JSON response into a Pandas DataFrame.

> **What is the Socrata Open Data API (SODA)?**[1]
>
> The **Socrata Open Data API (SODA)** is a standardized web API used by governments to publish and share open data in machine-readable formats such as JSON, CSV, and OData. It enables analysts, developers, and researchers to query public datasets directly—filtering, aggregating, and retrieving only the records they need.
>
> SODA is widely adopted across U.S. federal, state, and local governments, including major cities (New York City, Chicago, Los Angeles), states (Maryland, Utah, Washington), and several federal agencies. Its broad adoption makes it one of the most common and reliable ways to access public-sector data programmatically.

Data Structure and Format

The dataset comprises a sequential table wherein every unit is an individual incident reported and dispatched by police. Each unit tray records a variety of data, including but not limited to, identifiers, timestamps, dispositions, and location data along with different types of response time data. The data is updated 4 times per day. A complete list of all columns, including their field names and data types is provided in the table below.

Column Name	Description	API Field Name	Data Type
Incident_ID	CAD Incident Number	incident_id	Text
Crime Reports	Crime reports written for the event.	cr_number	Text
Crash Reports	Crash reports written for the event.	crash_reports	Text
Start Time	Incident call pickup date/time	start_time	Floating Timestamp
End Time	Last unit cleared date/time	end_time	Floating Timestamp
Priority	0 thru 4, with 0 as the highest priority	priority	Text
Initial Type	159 different codes for initial call type.	initial_type	Text
Close Type	Incident call type could change by the end of the call.	close_type	Text
Address	GIS process to geocode the actual address from police to the 100-block level address.	address	Text
City	City (mailing address value, not jurisdiction/municipality).	city	Text
State	State	state	Text
Zip	Zip Code	zip	Text
Longitude	Longitude for actual address, geocoded to 100-block level.	longitude	Number
Latitude	Latitude for actual address, geocoded to 100-block level.	latitude	Number
Police District	Incident location.	police_district_number	Text

Column Name	Description	API Field Name	Data Type
Number			
Beat	Incident location.	sector	Text
PRA	Incident location.	pra	Text
CallTime CallRoute	Seconds from call pickup to data entry start.	calltime_callroute	Number
Calltime Dispatch	Seconds from call pickup to first unit dispatched.	calltime_dispatch	Number
Calltime Arrive	Seconds from call pickup to first unit arriving on scene.	calltime_arrive	Number
Calltime Cleared	Seconds from call pickup to last unit cleared.	calltime_cleared	Number
CallRoute Dispatch	Seconds from data entry start to first unit dispatched.	callroute_dispatch	Number
Dispatch Arrive	Seconds from first unit dispatched to first unit arriving on scene.	dispatch_arrive	Number
Arrive Cleared	Seconds from first unit arriving on scene to last unit cleared.	arrive_cleared	Number
Disposition Desc	Final call disposition from last unit cleared.	disposition_desc	Text
Location	Geospatial point (latitude/longitude) of the incident.	geolocation	Point

The table above is directly based on the table from Montgomery County Open Data.

Linking to Data

Dataset Portal Page:

Police Dispatched Incidents – Montgomery County Open Data

API Endpoint (JSON):

```
https://data.montgomerycountymd.gov/resource/98cc-bc7d.json
```

Filtered Endpoint for October 2025:

https://data.montgomerycountymd.gov/resource/98cc-bc7d.json?$where=start_time%20between%20'2025-10-01T00:00:00'%20and%20'2025-10-31T23:59:59'

Code

The code block below fetches police dispatch data from Montgomery County's open-data API using API requests, converts the returned JSON into a pandas DataFrame, and saves it as a CSV file for future use.

```
import requests
import pandas as pd

# Step 1: Define the API endpoint for October 2025 data with a limit of 50000 records
url = ("https://data.montgomerycountymd.gov/resource/98cc-bc7d.json"
       "?$where=start_time between '2025-10-01T00:00:00' and '2025-10-31T23:59:59'"
       "&$limit=50000")

# Step 2: Send a GET request to the API endpoint above and parse the JSON response
data = requests.get(url).json()

# Step 3: Convert the JSON data into a pandas DataFrame
police_incident = pd.DataFrame(data)

# Step 4: Save the DataFrame to a CSV file
police_incident.to_csv("../../data/raw-data/police_incident.csv", index=False)

# Step 5: Display the DataFrame info
print(police_incident.describe())
```

```
        incident_id             cr_number               start_time  \
count         16782                  3056                    16782
unique        16782                  3056                    16714
top     P2500322473  250048786, 250050052  2025-10-09T16:14:56.000
freq              1                     1                        2

                      end_time priority                    initial_type  \
count                    16782    16782                           16782
unique                   16720        5                             161
top     2025-10-08T10:53:27.000        1  TRAFFIC/TRANSPORTATION INCIDENT
freq                         2     6459                            1394

                           close_type                  address  \
count                           16782                    16767
unique                            160                     6976
top     TRAFFIC/TRANSPORTATION INCIDENT  11100 BLK  VEIRS MILL RD
freq                              1392                      141

                city  state  ... calltime_callroute calltime_dispatch  \
count          16781  16781  ...              16782             15811
unique            32      1  ...                752              3104
```

```
top     SILVER SPRING     MD  ...                  0                204
freq             5528  16781  ...                442                 47

       callroute_dispatch    disposition_desc  \
count               15811               16782
unique               2937                 272
top                     0  OTHERMISCELLANEOUS
freq                  183                2960

                                    geolocation calltime_arrive  \
count                                     16782           13532
unique                                     5412            3628
top     {'type': 'Point', 'coordinates': [0, 0]}             613
freq                                       2406              20

       calltime_cleared dispatch_arrive arrive_cleared crash_reports
count             16779           13200          13531           664
unique             6780            2292           4592           664
top                1294               2            289     250048974
freq                 14             193             20             1

[4 rows x 26 columns]
```

```
print(police_incident.head())
```

```
   incident_id             cr_number               start_time  \
0  P2500322473  250048786, 250050052  2025-10-30T16:33:10.000
1  P2500324177             250048984  2025-10-31T23:30:25.000
2  P2500323992             250048964  2025-10-31T20:33:57.000
3  P2500323724                   NaN  2025-10-31T16:07:23.000
4  P2500324147             250048985  2025-10-31T22:50:14.000

                 end_time priority  \
0  2025-11-07T14:57:06.000        4
1  2025-11-01T02:45:49.000        2
2  2025-11-01T02:25:52.000        2
3  2025-11-01T01:37:13.000        4
4  2025-11-01T01:36:36.000        4

                                      initial_type  \
0  VANDALISM, DAMAGE, MISCHIEF-TRS - TELEPHONE RE...
1                              ASSIST OTHER AGENCY
2                              ASSIST OTHER AGENCY
3  FRAUDT-TRS FRAUD / DECEPTION - TELEPHONE REPOR...
4  STLVEHT - TRS STOLEN VEHICLE - TELEPHONE REPOR...

                                        close_type  \
0  VANDALISM, DAMAGE, MISCHIEF-TRS - TELEPHONE RE...
1                              ASSIST OTHER AGENCY
2                              ASSIST OTHER AGENCY
```

```
3  FRAUDT–TRS FRAUD / DECEPTION – TELEPHONE REPOR...
4  STLVEHT – TRS STOLEN VEHICLE – TELEPHONE REPOR...

                    address           city state  ... calltime_callroute  \
0     19900 BLK  WOOTTON AVE    POOLESVILLE    MD  ...                123
1      8200 BLK  GEORGIA AVE  SILVER SPRING    MD  ...                  0
2     1  BLK  TURTLE DOVE CT   GAITHERSBURG    MD  ...                  0
3  15200 BLK  SIESTA KEY WAY      ROCKVILLE    MD  ...                183
4      8000 BLK  EASTERN AVE  SILVER SPRING    MD  ...                412

  calltime_dispatch callroute_dispatch       disposition_desc  \
0              6804               6681     VANDALISM–DWELLING
1               137                137        MENTALTRANSPORT
2               NaN                NaN          CDS–USEDRUGOD
3              2329               2145   FORGERY/CNTRFT–ALLOTH
4              3191               2779  AUTOTHEFT–PASSENGERVEH

                                      geolocation calltime_arrive  \
0          {'type': 'Point', 'coordinates': [0, 0]}            NaN
1  {'type': 'Point', 'coordinates': [–77.0266, 38...            190
2  {'type': 'Point', 'coordinates': [–77.1873, 39...            217
3  {'type': 'Point', 'coordinates': [–77.1946, 39...            NaN
4  {'type': 'Point', 'coordinates': [–77.031, 38....            NaN

  calltime_cleared dispatch_arrive arrive_cleared crash_reports
0              NaN             NaN            NaN           NaN
1            11723              53          11532           NaN
2            21115             NaN          20898           NaN
3            34189             NaN            NaN           NaN
4             9982             NaN            NaN           NaN

[5 rows x 26 columns]
```

Conclusion

Among all the future steps in the data workflow, the data collection stage is the simplest, with only a single task, which is obtaining Montgomery County's police dispatch incident records from the county's open-data portal. The data extracted included a total of 16,782 records about police dispatch incidents. Each of the records contains 26 attributes, including unique identifiers, timestamps, initial and closing call classifications, priority, address, police district, and derived timing intervals. The `incident_id`, `start_time` and `initial_type` columns are fully populated, but some other columns such as `cr_number`, `calltime_arrive` and `crash_reports` contain null values. As a result, data cleaning steps like imputing or dropping missing data will be necessary.

Challenges

The only technical difficulty occured during the initial data acquisition stage is that the data.gov[2] CKAN API delivers only metadata, so the workflow had to pivot to the Montgomery County Socrata Open Data API to fetch the actual records.

References

1. ChatGPT, version-5.1, OpenAI, NOV-2025, chat.openai.com.

Footnotes

1. Montgomery County Police Dispatched Incident Dataset ↩
2. DATA.GOV, Police Dispatched Incidents, Montgomery County of Maryland ↩

DSAN 5000 - Incident Response Modeling (MCMD)

Data Cleaning

Overview

During data cleaning, I will transform the raw Police Dispatched Incidents data into a usable dataset. If this step is not performed well, misleading values or categories can lead to misleading results later on. I will convert timestamps to date and time, ensure that numeric fields are correct, fix missing values, and add features such as priority flags or indicators. More specifically, I will first parse fields like `start_time`, `end_time`, `latitude`, `longitude`, and various call-time durations, standardize columns like `initial_type`, `close_type`, and `disposition_desc`, and handle missing and duplicate values. For each of the code blocks below, I will document each step with notes and explanations. I will include a visualization of the data before and after cleaning. Also, the process of data cleaning might need to be revisited as the project evolves and analysis goals change.

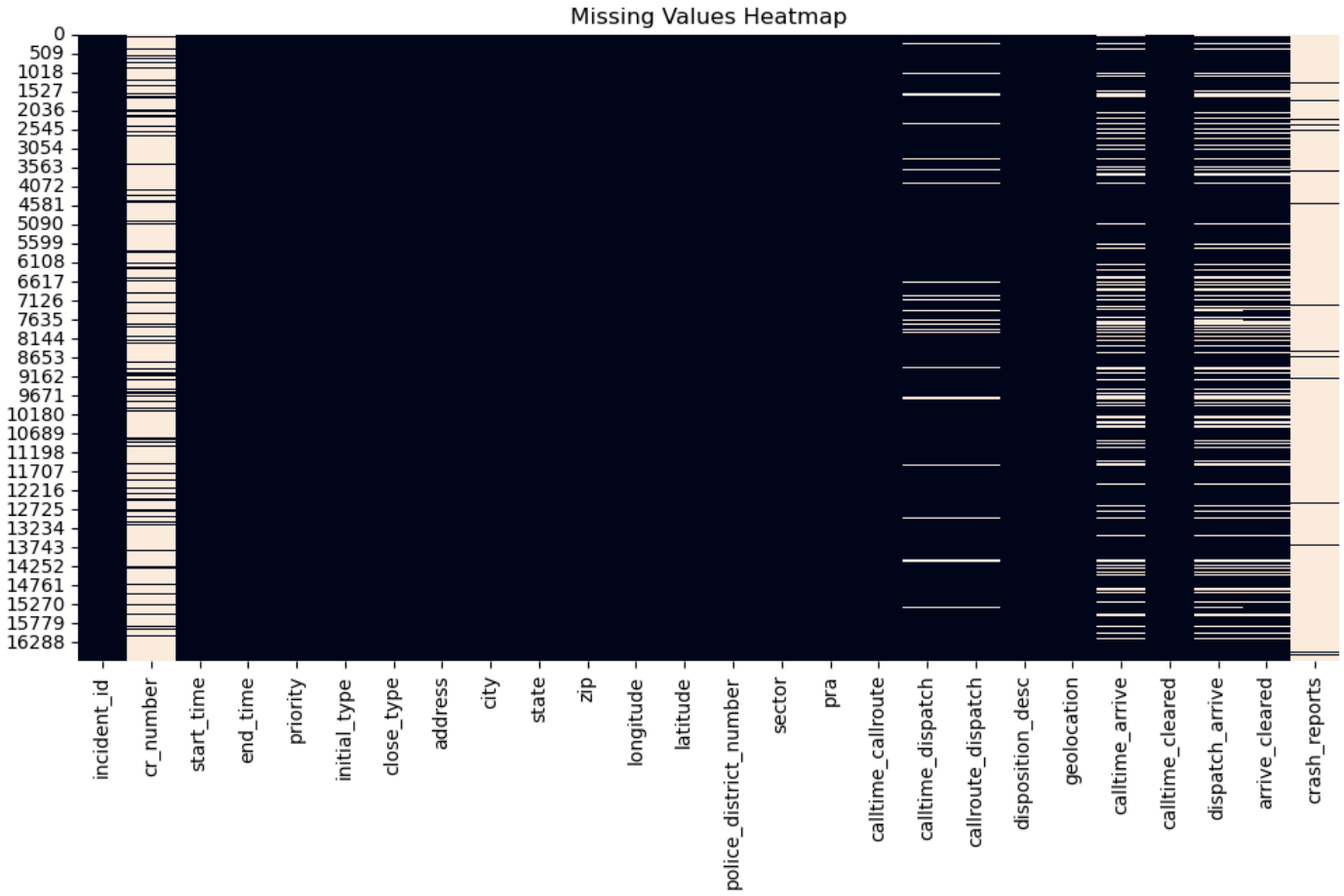
Methods and Codes (EDA)

In this section, I will mainly focuse on preparing the dataset for exploratory data analysis (EDA). This includes handling missing values, visualizing missing data patterns, converting units, generate missing longitude and latitudes based on addresses, generate new columns, and ensuring the dataset is clean and ready for analysis.

Visualize Missing Data

In this step, I will create a heatmap to show the distribution of missing values in the dataset. This helps to identify patterns of missingness and informs decisions on how to handle them. Also, I can remove columns with excessive missing values based on the visualization.

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

# Load the police incident dataset
police_incident = pd.read_csv("../../data/raw-data/police_incident.csv")

# Visualize missing data using a heatmap
plt.figure(figsize=(12,6))
sns.heatmap(police_incident.isna(), cbar=False)
plt.title("Missing Values Heatmap")
plt.savefig("../../report/visual3.png", dpi=300, bbox_inches="tight")
plt.show()
```



Handling Columns with Large Amounts of Missing Data

From the visualization above, we can identify two columns with significant missing data: `cr_number` and `crash_reports`. Since these columns have substantial missing data, they are not helpful to our analysis, so I will remove these columns to improve the dataset's quality and ensure that subsequent analyses are not biased by this information. I will also provide a summary of data tyoes, nulls and zeros of the remaining columns.

```
# Remove columns with large amounts of missing data
police_incident = police_incident.drop("cr_number", axis=1)
police_incident = police_incident.drop("crash_reports", axis=1)

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
incident_id	0	0	object

	null_count	zero_count	dtype
start_time	0	0	object
end_time	0	0	object
priority	0	1682	int64
initial_type	0	0	object
close_type	0	0	object
address	15	0	object
city	1	0	object
state	1	0	object
zip	1	0	float64
longitude	0	2406	float64
latitude	0	2406	float64
police_district_number	0	0	object
sector	1	0	object
pra	1	0	float64
calltime_callroute	0	442	int64
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
disposition_desc	0	0	object
geolocation	0	0	object
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64

Data Type Correction

By reviewing the table above, I will review the data types of each column to ensure they are appropriate for analysis. I converted `start_time` and `end_time` into datetime objects. The `zip` column was cast to a string. Several descriptive fields, including `priority`, `initial_type`, `close_type`, `city`, `state`, `sector`, and `disposition_desc`, were converted to categorical types. Finally, all response-time metrics, including `calltime_callroute`, `calltime_dispatch`, `callroute_dispatch`, `calltime_arrive`, `calltime_cleared`, `dispatch_arrive`, and `arrive_cleared`, were converted to numeric values.

```
# Data Type Correction
police_incident['start_time'] = pd.to_datetime(police_incident['start_time'])
police_incident['end_time'] = pd.to_datetime(police_incident['end_time'])
```

```
police_incident['zip'] = police_incident['zip'].astype('Int64').astype('string')

police_incident['priority'] = police_incident['priority'].astype('category')
police_incident['initial_type'] = police_incident['initial_type'].astype('category')
police_incident['close_type'] = police_incident['close_type'].astype('category')
police_incident['city'] = police_incident['city'].astype('category')
police_incident['state'] = police_incident['state'].astype('category')
police_incident['sector'] = police_incident['sector'].astype('category')
police_incident['disposition_desc'] = police_incident['disposition_desc'].astype('categor

police_incident['calltime_callroute'] = pd.to_numeric(police_incident['calltime_callroute
police_incident['calltime_dispatch'] = pd.to_numeric(police_incident['calltime_dispatch']
police_incident['callroute_dispatch'] = pd.to_numeric(police_incident['callroute_dispatch
police_incident['calltime_arrive'] = pd.to_numeric(police_incident['calltime_arrive'], er
police_incident['calltime_cleared'] = pd.to_numeric(police_incident['calltime_cleared'],
police_incident['dispatch_arrive'] = pd.to_numeric(police_incident['dispatch_arrive'], er
police_incident['arrive_cleared'] = pd.to_numeric(police_incident['arrive_cleared'], erro

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
incident_id	0	0	object
start_time	0	0	datetime64[ns]
end_time	0	0	datetime64[ns]
priority	0	1682	category
initial_type	0	0	category
close_type	0	0	category
address	15	0	object
city	1	0	category
state	1	0	category
zip	1	0	string[python]
longitude	0	2406	float64
latitude	0	2406	float64
police_district_number	0	0	object
sector	1	0	category
pra	1	0	float64
calltime_callroute	0	442	int64

	null_count	zero_count	dtype
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
disposition_desc	0	0	category
geolocation	0	0	object
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64

Handle Missing Longitude and Latitude

Since the longitude and latitude columns are essential for generating maps and spatial analysis, I will fill in the missing values using the Google Geocoding API, since I have the address information for each incident. This approach ensures that we have complete geographical data for all incidents, which is crucial for accurate spatial analysis and visualization.

```
import requests
import numpy as np
import time
from dotenv import load_dotenv
import os

# Load the API key from environment variables
load_dotenv("../../apikey.env")
API_KEY = os.getenv("GOOGLE_API_KEY")
GEOCODE_URL = "https://maps.googleapis.com/maps/api/geocode/json"

# Define the geocoding function
def geocode(address):
    req = requests.get(GEOCODE_URL, params={"address": address, "key": API_KEY}).json()
    if req.get("status") == "OK":
        loc = req["results"][0]["geometry"]["location"]
        return loc["lng"], loc["lat"]
    return np.nan, np.nan

# Identify rows with missing latitude or longitude
missing = police_incident[(police_incident["longitude"] == 0) | (police_incident["latitud

# Iterate over missing rows and update longitude and latitude
count = 0

for idx, row in missing.iterrows():
    address = f"{row['address']}, {row['city']}, {row['state']} {row['zip']}"
    lon, lat = geocode(address)
```

```
    police_incident.loc[idx, "longitude"] = lon
    police_incident.loc[idx, "latitude"] = lat
    count += 1
    print(f"[{count}/{len(missing)}] Index {idx}: {lon}, {lat}")

# To avoid too many api requests for my google api key
police_incident.to_csv("../../data/processed-data/police_incident_longitude&latitude.csv"
print("Saved")
```

```
[1/2406] Index 0: -77.41542679999999, 39.1434676
[2/2406] Index 13: -77.2742373, 39.1467753
[3/2406] Index 21: -77.2160456, 39.0097548
[4/2406] Index 25: -77.1564864, 38.9909745
[5/2406] Index 33: -77.13367579999999, 38.9778763
[6/2406] Index 37: -76.9414044, 39.0842899
[7/2406] Index 50: -77.2747315, 39.2144536
[8/2406] Index 56: -77.1102449, 38.9782397
[9/2406] Index 58: -77.2707555, 39.0555524
[10/2406] Index 60: -77.20849450000001, 39.2872181
[11/2406] Index 67: -77.261729, 39.187493
[12/2406] Index 74: -77.0507011, 39.0369934
[13/2406] Index 84: -77.1928255, 39.1442494
[14/2406] Index 115: -77.1972425, 39.1836612
[15/2406] Index 120: -77.19296560000001, 39.1474639
[16/2406] Index 128: -77.0923952, 39.0501657
[17/2406] Index 133: -76.9879646, 39.1473132
[18/2406] Index 137: -76.9844652, 39.0184131
[19/2406] Index 146: -77.11292259999999, 39.0213603
[20/2406] Index 150: -77.0218648, 38.9971837
[21/2406] Index 152: -77.4090434, 39.14368750000001
[22/2406] Index 165: -77.05541989999999, 39.0556524
[23/2406] Index 167: -77.0503559, 39.058254
[24/2406] Index 188: -77.17634819999999, 38.9962098
[25/2406] Index 189: -76.9472612, 39.0827914
[26/2406] Index 194: -77.0046161, 39.0696777
[27/2406] Index 197: -77.2039928, 39.2508087
[28/2406] Index 202: -76.9898226, 38.9991051
[29/2406] Index 213: -77.0482408, 39.0413702
[30/2406] Index 217: -77.087991, 39.1411867
[31/2406] Index 219: -77.0804689, 39.0901267
[32/2406] Index 233: -76.96803369999999, 39.0565845
[33/2406] Index 234: -77.1394384, 39.0792326
[34/2406] Index 235: -77.056175, 38.9941413
[35/2406] Index 238: -77.0218648, 38.9971837
[36/2406] Index 240: -77.0169196, 38.9879087
[37/2406] Index 259: -77.1004896, 39.0907042
[38/2406] Index 276: -77.07751449999999, 39.0760398
[39/2406] Index 278: -77.2095056, 39.0186509
[40/2406] Index 316: -77.2056459, 39.01591
[41/2406] Index 320: -77.0503559, 39.058254
```

```
[42/2406] Index 328: -77.2586498, 39.2317476
[43/2406] Index 336: -76.9788651, 39.09785110000001
[44/2406] Index 338: -76.97203809999999, 39.1008533
[45/2406] Index 339: -77.096732, 38.9911745
[46/2406] Index 351: -77.0503559, 39.058254
[47/2406] Index 354: -77.0304429, 38.9947772
[48/2406] Index 356: -77.0531, 39.061364
[49/2406] Index 376: -77.22719649999999, 39.0392587
[50/2406] Index 387: -77.10145729999999, 38.9632952
[51/2406] Index 394: -77.0513692, 39.0603083
[52/2406] Index 402: -77.10197699999999, 39.0931164
[53/2406] Index 413: -77.0477767, 39.045273
[54/2406] Index 418: -77.137371, 39.0917302
[55/2406] Index 420: -77.18687419999999, 39.1739128
[56/2406] Index 423: -77.0521418, 39.0418728
[57/2406] Index 425: -77.04706279999999, 38.9956155
[58/2406] Index 434: -77.1332048, 39.0014769
[59/2406] Index 440: -77.05341059999999, 39.0478294
[60/2406] Index 448: -77.2118407, 39.2813446
[61/2406] Index 454: -77.096732, 38.9911745
[62/2406] Index 470: -77.17634819999999, 38.9962098
[63/2406] Index 476: -77.2325831, 39.1153651
[64/2406] Index 504: -77.4206647, 39.1425355
[65/2406] Index 507: -77.071968, 39.1516646
[66/2406] Index 508: -77.20279239999999, 39.0760616
[67/2406] Index 510: -77.0321359, 38.9974075
[68/2406] Index 514: -76.9472612, 39.0827914
[69/2406] Index 517: -77.01672649999999, 39.0412122
[70/2406] Index 519: -77.0454558, 39.0365977
[71/2406] Index 524: -77.096732, 38.9911745
[72/2406] Index 525: -77.2855295, 39.2218699
[73/2406] Index 536: -77.0754709, 39.030936
[74/2406] Index 541: -77.00316140000001, 39.0330374
[75/2406] Index 546: -77.1364919, 39.0535941
[76/2406] Index 553: -77.1100027, 39.0509191
[77/2406] Index 565: -77.0284697, 38.9942096
[78/2406] Index 566: -77.2614478, 39.1846433
[79/2406] Index 575: -77.0284697, 38.9942096
[80/2406] Index 577: -77.2055748, 39.2906426
[81/2406] Index 579: -77.0321359, 38.9974075
[82/2406] Index 584: -77.0531484, 39.1455341
[83/2406] Index 590: -77.0304429, 38.9947772
[84/2406] Index 591: -77.1895389, 39.0185405
[85/2406] Index 603: -76.9403895, 39.0870183
[86/2406] Index 616: -76.9472612, 39.0827914
[87/2406] Index 624: -77.1191525, 39.0729866
[88/2406] Index 628: -77.0704485, 39.0770626
[89/2406] Index 640: -77.0704485, 39.0770626
[90/2406] Index 651: -77.19418329999999, 39.1450142
[91/2406] Index 657: -77.0320065, 38.9960781
[92/2406] Index 660: -77.1944214, 39.1887816
```

```
[93/2406] Index 674: -77.17495989999999, 39.1529136
[94/2406] Index 678: -77.0521418, 39.0418728
[95/2406] Index 680: -77.17634819999999, 38.9962098
[96/2406] Index 681: -77.1972425, 39.1836612
[97/2406] Index 695: -76.9677966, 39.0540986
[98/2406] Index 700: -77.0704485, 39.0770626
[99/2406] Index 704: -77.096732, 38.9911745
[100/2406] Index 713: -77.1394384, 39.0792326
[101/2406] Index 720: -77.153899, 39.1798986
[102/2406] Index 725: -77.0218648, 38.9971837
[103/2406] Index 728: -77.1218317, 39.0848457
[104/2406] Index 742: -77.0218648, 38.9971837
[105/2406] Index 746: -77.1218317, 39.0848457
[106/2406] Index 750: -77.071489, 39.07236109999999
[107/2406] Index 767: -77.0357279, 39.0260298
[108/2406] Index 776: -76.97423549999999, 39.0921702
[109/2406] Index 779: -77.0503559, 39.058254
[110/2406] Index 789: -77.0304429, 38.9947772
[111/2406] Index 790: -77.0570594, 39.0808601
[112/2406] Index 802: -77.10026839999999, 38.9907397
[113/2406] Index 810: -77.0503559, 39.058254
[114/2406] Index 818: -77.037466, 38.9958963
[115/2406] Index 829: -77.0521418, 39.0418728
[116/2406] Index 845: -77.2601712, 39.2452183
[117/2406] Index 846: -77.01437419999999, 39.0034711
[118/2406] Index 852: -77.10273579999999, 39.00523039999999
[119/2406] Index 854: -77.0513359, 39.0387866
[120/2406] Index 859: -77.1009826, 39.00844319999999
[121/2406] Index 862: -77.2637907, 39.182903
[122/2406] Index 881: -77.0516921, 39.0506864
[123/2406] Index 884: -77.28758409999999, 39.2198997
[124/2406] Index 888: -77.096732, 38.9911745
[125/2406] Index 896: -77.0579836, 39.0471735
[126/2406] Index 900: -77.1056768, 39.0466561
[127/2406] Index 908: -77.1770173, 39.155269
[128/2406] Index 911: -77.11194189999999, 38.9696285
[129/2406] Index 919: -77.096732, 38.9911745
[130/2406] Index 926: -77.096732, 38.9911745
[131/2406] Index 942: -77.1077272, 39.05715989999999
[132/2406] Index 948: -77.037466, 38.9958963
[133/2406] Index 969: -77.1197521, 38.9516143
[134/2406] Index 980: -77.05233, 39.047348
[135/2406] Index 981: -77.0304429, 38.9947772
[136/2406] Index 998: -76.9378869, 39.0833988
[137/2406] Index 1001: -77.17634819999999, 38.9962098
[138/2406] Index 1022: -77.092275, 38.977352
[139/2406] Index 1029: -77.037466, 38.9958963
[140/2406] Index 1032: -76.9435713, 39.0836213
[141/2406] Index 1033: -77.15125859999999, 39.1060666
[142/2406] Index 1038: -77.0900886, 38.9818753
[143/2406] Index 1045: -77.0221782, 39.0435465
```

```
[144/2406] Index 1049: -77.092275, 38.977352
[145/2406] Index 1052: -77.1471339, 38.9804177
[146/2406] Index 1056: -77.0524647, 39.0436051
[147/2406] Index 1065: -77.037466, 38.9958963
[148/2406] Index 1073: -77.0221782, 39.0435465
[149/2406] Index 1092: -77.0513692, 39.0603083
[150/2406] Index 1096: -77.037466, 38.9958963
[151/2406] Index 1104: -77.0507011, 39.0369934
[152/2406] Index 1115: -77.037466, 38.9958963
[153/2406] Index 1118: -76.9898226, 38.9991051
[154/2406] Index 1119: -76.9859108, 39.0417993
[155/2406] Index 1126: -76.95035179999999, 39.0821822
[156/2406] Index 1139: -77.0503559, 39.058254
[157/2406] Index 1141: -77.0513359, 39.0387866
[158/2406] Index 1143: -77.0490394, 39.032466
[159/2406] Index 1160: -77.0304429, 38.9947772
[160/2406] Index 1169: -77.02966769999999, 38.9869576
[161/2406] Index 1170: -77.1762407, 39.06334409999999
[162/2406] Index 1177: -77.096732, 38.9911745
[163/2406] Index 1180: -77.2080678, 39.0167227
[164/2406] Index 1181: -77.2297057, 39.0854362
[165/2406] Index 1189: -77.1165606, 39.0424394
[166/2406] Index 1192: -76.96803369999999, 39.0565845
[167/2406] Index 1199: -77.1056768, 39.0466561
[168/2406] Index 1203: -77.096732, 38.9911745
[169/2406] Index 1207: -76.9403895, 39.0870183
[170/2406] Index 1222: -77.4169311, 39.1459392
[171/2406] Index 1226: -77.19296560000001, 39.1474639
[172/2406] Index 1228: -77.27337949999999, 39.1774144
[173/2406] Index 1232: -77.2095763, 39.1500846
[174/2406] Index 1237: -77.0040186, 38.9984446
[175/2406] Index 1238: -77.10145729999999, 38.9632952
[176/2406] Index 1239: -77.20211619999999, 39.2873064
[177/2406] Index 1242: -77.096732, 38.9911745
[178/2406] Index 1245: -77.31250039999999, 39.18453239999999
[179/2406] Index 1251: -77.1013438, 39.1929288
[180/2406] Index 1252: -77.1150734, 39.0564729
[181/2406] Index 1287: -77.0521418, 39.0418728
[182/2406] Index 1292: -76.9904531, 39.0854584
[183/2406] Index 1296: -77.0628988, 39.0452792
[184/2406] Index 1310: -77.13412640000001, 38.9756446
[185/2406] Index 1311: -76.982007, 39.0030968
[186/2406] Index 1340: -77.19514079999999, 39.1449196
[187/2406] Index 1342: -77.0963288, 39.1554368
[188/2406] Index 1351: -77.0880416, 39.0740484
[189/2406] Index 1359: -77.1056768, 39.0466561
[190/2406] Index 1360: -77.0745917, 39.0796341
[191/2406] Index 1367: -77.2421268, 39.1865723
[192/2406] Index 1370: -77.0218648, 38.9971837
[193/2406] Index 1379: -77.0304429, 38.9947772
[194/2406] Index 1383: -77.17296850000001, 39.183898
```

```
[195/2406] Index 1386: -77.096732, 38.9911745
[196/2406] Index 1399: -77.0373762, 39.0337511
[197/2406] Index 1413: -77.1016389, 39.0003388
[198/2406] Index 1416: -77.0418055, 39.0047868
[199/2406] Index 1425: -77.0633007, 39.0796727
[200/2406] Index 1431: -77.204864, 39.0149954
[201/2406] Index 1444: -77.092275, 38.977352
[202/2406] Index 1450: -76.9403895, 39.0870183
[203/2406] Index 1456: -77.1310137, 39.0581032
[204/2406] Index 1461: -77.0915969, 38.9857681
[205/2406] Index 1464: -77.0368367, 39.0467477
[206/2406] Index 1466: -77.182754, 39.1862695
[207/2406] Index 1469: -77.05233, 39.047348
[208/2406] Index 1475: -77.2637907, 39.182903
[209/2406] Index 1483: -77.0962672, 38.9821416
[210/2406] Index 1500: -77.110891, 39.048689
[211/2406] Index 1501: -77.091251, 38.9632933
[212/2406] Index 1503: -77.0810892, 39.0830252
[213/2406] Index 1505: -77.10273579999999, 39.00523039999999
[214/2406] Index 1506: -77.096732, 38.9911745
[215/2406] Index 1508: -77.1527813, 39.0839994
[216/2406] Index 1517: -77.22087479999999, 39.109296
[217/2406] Index 1522: -77.20111200000001, 39.2659119
[218/2406] Index 1523: -76.9898226, 38.9991051
[219/2406] Index 1531: -77.019441, 39.0019344
[220/2406] Index 1532: -77.0868922, 39.0505251
[221/2406] Index 1537: -77.0165935, 38.9904667
[222/2406] Index 1543: -77.02828980000001, 38.9917849
[223/2406] Index 1546: -76.9320314, 39.0795989
[224/2406] Index 1568: -77.13618, 39.0783682
[225/2406] Index 1579: -77.2657333, 39.161475
[226/2406] Index 1581: -77.1193251, 39.0748421
[227/2406] Index 1584: -77.18905579999999, 39.1857936
[228/2406] Index 1613: -77.20111200000001, 39.2659119
[229/2406] Index 1614: -77.1480007, 39.1069963
[230/2406] Index 1621: -77.13618, 39.0783682
[231/2406] Index 1642: -77.04869280000001, 39.0390935
[232/2406] Index 1650: -77.1086715, 39.055793
[233/2406] Index 1656: -77.1374668, 39.0781619
[234/2406] Index 1660: -77.0549836, 39.0501145
[235/2406] Index 1675: -77.05910810000002, 39.0644698
[236/2406] Index 1691: -77.0366887, 39.0266216
[237/2406] Index 1698: -77.0490394, 39.032466
[238/2406] Index 1699: -77.0304429, 38.9947772
[239/2406] Index 1715: -77.0325277, 39.0680401
[240/2406] Index 1717: -77.190445, 39.1499694
[241/2406] Index 1724: -77.0785044, 39.0455767
[242/2406] Index 1730: -77.13091659999999, 39.0814459
[243/2406] Index 1734: -77.13685579999999, 38.9651816
[244/2406] Index 1750: -77.092275, 38.977352
[245/2406] Index 1761: -77.19701719999999, 39.1730914
```

```
[246/2406] Index 1766: -77.0321359, 38.9974075
[247/2406] Index 1775: -77.0304429, 38.9947772
[248/2406] Index 1779: -77.0490394, 39.032466
[249/2406] Index 1788: -77.2904185, 39.0954511
[250/2406] Index 1792: -77.265412, 39.159749
[251/2406] Index 1796: -77.2118407, 39.2813446
[252/2406] Index 1800: -77.135762, 39.0977372
[253/2406] Index 1805: -77.219678, 39.2745326
[254/2406] Index 1808: -77.096732, 38.9911745
[255/2406] Index 1837: -77.1395149, 39.1044579
[256/2406] Index 1843: -77.1360301, 39.0870662
[257/2406] Index 1844: -77.28286899999999, 39.1668802
[258/2406] Index 1853: -77.088227, 39.0571154
[259/2406] Index 1854: -77.05196910000001, 39.064882
[260/2406] Index 1856: -77.0304429, 38.9947772
[261/2406] Index 1865: -77.0586546, 39.1762383
[262/2406] Index 1869: -77.05280719999999, 39.0517045
[263/2406] Index 1880: -76.9420805, 39.0858282
[264/2406] Index 1887: -77.2885931, 39.2152704
[265/2406] Index 1897: -77.163574, 39.164211
[266/2406] Index 1899: -77.02966769999999, 38.9869576
[267/2406] Index 1900: -77.0304429, 38.9947772
[268/2406] Index 1921: -77.1677473, 39.127297
[269/2406] Index 1924: -77.0804689, 39.0901267
[270/2406] Index 1934: -77.096732, 38.9911745
[271/2406] Index 1963: -77.049961, 39.0536409
[272/2406] Index 1965: -77.1723102, 39.009687
[273/2406] Index 1967: -77.1165606, 39.0424394
[274/2406] Index 1968: -77.05387379999999, 39.0504851
[275/2406] Index 1969: -76.96803369999999, 39.0565845
[276/2406] Index 1970: -77.151068, 38.98308249999999
[277/2406] Index 1972: -77.15518800000001, 39.08695420000001
[278/2406] Index 1975: -77.0304429, 38.9947772
[279/2406] Index 1976: -77.0991291, 38.9774035
[280/2406] Index 1978: -77.11535789999999, 39.0816808
[281/2406] Index 1982: -77.0668043, 39.0366406
[282/2406] Index 2028: -77.0330571, 39.0255961
[283/2406] Index 2050: -77.4117346, 39.1426842
[284/2406] Index 2055: -76.996917, 39.0671395
[285/2406] Index 2079: -77.0669763, 39.1483588
[286/2406] Index 2085: -77.1403486, 39.1014718
[287/2406] Index 2089: -77.0284697, 38.9942096
[288/2406] Index 2104: -77.0275423, 38.9942077
[289/2406] Index 2108: -77.2657869, 39.19148819999999
[290/2406] Index 2109: -77.0936086, 39.0492134
[291/2406] Index 2119: -77.0934925, 38.98267200000001
[292/2406] Index 2125: -77.4179784, 39.1462327
[293/2406] Index 2126: -77.259652, 39.2217251
[294/2406] Index 2140: -77.0745917, 39.0796341
[295/2406] Index 2142: -77.18191780000001, 38.9771617
[296/2406] Index 2155: -76.9873487, 39.0934061
```

```
[297/2406] Index 2161: -77.20644589999999, 39.2873125
[298/2406] Index 2165: -76.9403895, 39.0870183
[299/2406] Index 2166: -77.2629418, 39.1887446
[300/2406] Index 2167: -76.996917, 39.0671395
[301/2406] Index 2168: -77.0704485, 39.0770626
[302/2406] Index 2190: -77.092275, 38.977352
[303/2406] Index 2195: -77.0362699, 39.0366465
[304/2406] Index 2196: -77.02966769999999, 38.9869576
[305/2406] Index 2199: -77.0368367, 39.0467477
[306/2406] Index 2207: -77.10458229999999, 38.9894803
[307/2406] Index 2211: -76.9320314, 39.0795989
[308/2406] Index 2216: -77.0804689, 39.0901267
[309/2406] Index 2223: -77.0880087, 39.06434429999999
[310/2406] Index 2225: -77.0963288, 39.1554368
[311/2406] Index 2226: -77.0702678, 39.0423743
[312/2406] Index 2230: -77.19418329999999, 39.1450142
[313/2406] Index 2232: -77.0404126, 39.0822851
[314/2406] Index 2239: -77.0745917, 39.0796341
[315/2406] Index 2240: -76.9403895, 39.0870183
[316/2406] Index 2245: -76.9472612, 39.0827914
[317/2406] Index 2252: -76.9648662, 39.1431112
[318/2406] Index 2268: -77.26832379999999, 39.1805994
[319/2406] Index 2272: -77.2614478, 39.1846433
[320/2406] Index 2275: -77.2586498, 39.2317476
[321/2406] Index 2285: -77.1544077, 39.1517734
[322/2406] Index 2289: -77.0633007, 39.0796727
[323/2406] Index 2298: -77.0507011, 39.0369934
[324/2406] Index 2311: -77.031183, 39.0006298
[325/2406] Index 2316: -77.0579836, 39.0471735
[326/2406] Index 2337: -77.0513692, 39.0603083
[327/2406] Index 2338: -76.9403895, 39.0870183
[328/2406] Index 2345: -77.0682222, 39.078769
[329/2406] Index 2347: -77.110891, 39.048689
[330/2406] Index 2354: -77.110891, 39.048689
[331/2406] Index 2368: -77.4206647, 39.1425355
[332/2406] Index 2403: -76.9339092, 39.0814418
[333/2406] Index 2421: -77.1150734, 39.0564729
[334/2406] Index 2424: -77.0915969, 38.9857681
[335/2406] Index 2431: -77.21004549999999, 39.1412831
[336/2406] Index 2436: -77.096732, 38.9911745
[337/2406] Index 2442: -76.996917, 39.0671395
[338/2406] Index 2451: -76.9648662, 39.1431112
[339/2406] Index 2454: -77.0919363, 38.9896885
[340/2406] Index 2462: -76.99392519999999, 39.0575886
[341/2406] Index 2463: -77.0763229, 39.0808091
[342/2406] Index 2466: -77.2614478, 39.1846433
[343/2406] Index 2477: -76.9403895, 39.0870183
[344/2406] Index 2488: -76.9790304, 39.0101886
[345/2406] Index 2495: -77.0127272, 38.9911529
[346/2406] Index 2515: -77.0369025, 38.99313370000001
[347/2406] Index 2516: -77.0459346, 39.0238691
```

```
[348/2406] Index 2519: -77.096732, 38.9911745
[349/2406] Index 2521: -77.02456269999999, 38.9949258
[350/2406] Index 2533: -77.219678, 39.2745326
[351/2406] Index 2536: -76.95688659999999, 39.0493731
[352/2406] Index 2537: -77.1395149, 39.1044579
[353/2406] Index 2543: -77.2361209, 39.1170588
[354/2406] Index 2545: -76.9886635, 38.9988623
[355/2406] Index 2546: -77.2276104, 39.0370991
[356/2406] Index 2569: -76.9648662, 39.1431112
[357/2406] Index 2570: -77.21004549999999, 39.1412831
[358/2406] Index 2573: -77.1056768, 39.0466561
[359/2406] Index 2575: -77.0919363, 38.9896885
[360/2406] Index 2579: -77.0635149, 39.0799114
[361/2406] Index 2580: -77.11075799999999, 39.051765
[362/2406] Index 2603: -77.1048279, 38.9622969
[363/2406] Index 2614: -77.2875245, 39.2301977
[364/2406] Index 2625: -77.1145153, 39.0677783
[365/2406] Index 2628: -77.0716671, 39.1318185
[366/2406] Index 2632: -77.102543, 39.0772032
[367/2406] Index 2639: -77.11610639999999, 39.0493736
[368/2406] Index 2641: -77.037466, 38.9958963
[369/2406] Index 2653: -77.190445, 39.1499694
[370/2406] Index 2655: -77.2614478, 39.1846433
[371/2406] Index 2680: -77.0320065, 38.9960781
[372/2406] Index 2696: -77.0503365, 39.0402975
[373/2406] Index 2709: -77.06873130000001, 39.149436
[374/2406] Index 2711: -77.0804689, 39.0901267
[375/2406] Index 2712: -77.0797542, 39.0876618
[376/2406] Index 2722: -77.1056768, 39.0466561
[377/2406] Index 2725: -77.0671495, 39.15114010000001
[378/2406] Index 2727: -77.0275423, 38.9942077
[379/2406] Index 2730: -77.4179784, 39.1462327
[380/2406] Index 2738: -77.096732, 38.9911745
[381/2406] Index 2741: -77.0502849, 39.0399696
[382/2406] Index 2746: -77.0669763, 39.1483588
[383/2406] Index 2764: -77.2637907, 39.182903
[384/2406] Index 2777: -77.02966769999999, 38.9869576
[385/2406] Index 2779: -77.08727189999999, 38.9637719
[386/2406] Index 2780: -77.037466, 38.9958963
[387/2406] Index 2781: -76.9420805, 39.0858282
[388/2406] Index 2787: -76.95688659999999, 39.0493731
[389/2406] Index 2788: -77.2614478, 39.1846433
[390/2406] Index 2793: -77.06584, 39.07928
[391/2406] Index 2803: -77.0676743, 39.19433
[392/2406] Index 2809: -76.9472612, 39.0827914
[393/2406] Index 2812: -77.2907196, 39.2189773
[394/2406] Index 2816: -77.0521418, 39.0418728
[395/2406] Index 2819: -77.0880087, 39.06434429999999
[396/2406] Index 2820: -77.0669593, 39.1560082
[397/2406] Index 2822: -77.0218648, 38.9971837
[398/2406] Index 2826: -77.0052675, 39.0315443
```

```
[399/2406] Index 2845: -77.0067976, 39.0047359
[400/2406] Index 2846: -77.2875245, 39.2301977
[401/2406] Index 2855: -77.07648710000001, 39.1088267
[402/2406] Index 2856: -77.2008352, 39.1173966
[403/2406] Index 2872: -77.2637907, 39.182903
[404/2406] Index 2891: -77.037466, 38.9958963
[405/2406] Index 2894: -77.2637907, 39.182903
[406/2406] Index 2899: -77.2888242, 39.0692884
[407/2406] Index 2902: -77.2421268, 39.1865723
[408/2406] Index 2905: -77.056175, 38.9941413
[409/2406] Index 2952: -77.16273079999999, 39.0886146
[410/2406] Index 2955: -77.224117, 38.985018
[411/2406] Index 2957: -77.0503628, 39.054544
[412/2406] Index 2965: -76.9873294, 39.1007119
[413/2406] Index 2966: -77.19296560000001, 39.1474639
[414/2406] Index 2983: -77.17640759999999, 39.0630713
[415/2406] Index 2989: -77.0495275, 39.0350568
[416/2406] Index 2995: -77.1056768, 39.0466561
[417/2406] Index 3010: -77.1145153, 39.0677783
[418/2406] Index 3019: -77.0503559, 39.058254
[419/2406] Index 3023: -77.0503559, 39.058254
[420/2406] Index 3039: -77.02456269999999, 38.9949258
[421/2406] Index 3046: -77.096732, 38.9911745
[422/2406] Index 3062: -77.2630061, 39.1611321
[423/2406] Index 3069: -77.0745917, 39.0796341
[424/2406] Index 3070: -77.110891, 39.048689
[425/2406] Index 3072: -77.254024, 39.0640898
[426/2406] Index 3084: -77.0104648, 39.0098985
[427/2406] Index 3091: -77.2875245, 39.2301977
[428/2406] Index 3093: -77.2181457, 39.2700913
[429/2406] Index 3098: -77.0787016, 39.0772915
[430/2406] Index 3099: -77.2858795, 39.2185127
[431/2406] Index 3100: -77.1765885, 39.1393941
[432/2406] Index 3105: -77.05648939999999, 38.9961702
[433/2406] Index 3113: -77.0503559, 39.058254
[434/2406] Index 3117: -76.9898226, 38.9991051
[435/2406] Index 3124: -77.2056459, 39.01591
[436/2406] Index 3125: -77.1928255, 39.1442494
[437/2406] Index 3129: -77.19514079999999, 39.1449196
[438/2406] Index 3130: -77.0633007, 39.0796727
[439/2406] Index 3132: -77.1354726, 39.0903168
[440/2406] Index 3138: -77.114448, 38.9814131
[441/2406] Index 3141: -77.0521418, 39.0418728
[442/2406] Index 3142: -77.1185491, 39.0329419
[443/2406] Index 3145: -77.1770173, 39.155269
[444/2406] Index 3153: -77.0704485, 39.0770626
[445/2406] Index 3161: -77.2875245, 39.2301977
[446/2406] Index 3164: -77.0730049, 38.97073750000001
[447/2406] Index 3165: -77.0704485, 39.0770626
[448/2406] Index 3168: -77.26572689999999, 39.1628577
[449/2406] Index 3173: -77.0704485, 39.0770626
```

```
[450/2406] Index 3174: -77.05184179999999, 39.0406395
[451/2406] Index 3175: -77.0513692, 39.0603083
[452/2406] Index 3179: -77.06584, 39.07928
[453/2406] Index 3183: -77.037466, 38.9958963
[454/2406] Index 3186: -77.06584, 39.07928
[455/2406] Index 3194: -77.0507011, 39.0369934
[456/2406] Index 3198: -77.1056768, 39.0466561
[457/2406] Index 3203: -76.9731272, 39.0503614
[458/2406] Index 3206: -77.2604454, 39.1868928
[459/2406] Index 3215: -76.9414044, 39.0842899
[460/2406] Index 3220: -77.0720712, 39.0777976
[461/2406] Index 3239: -77.0531, 39.061364
[462/2406] Index 3249: -77.190445, 39.1499694
[463/2406] Index 3261: -77.0804689, 39.0901267
[464/2406] Index 3265: -77.133561, 39.03353380000001
[465/2406] Index 3272: -77.0991291, 38.9774035
[466/2406] Index 3279: -76.996917, 39.0671395
[467/2406] Index 3280: -77.2123694, 39.0884661
[468/2406] Index 3289: -77.1666272, 39.1233901
[469/2406] Index 3305: -77.0727481, 39.2015834
[470/2406] Index 3309: -77.0627216, 39.0526522
[471/2406] Index 3316: -77.06590159999999, 39.1897598
[472/2406] Index 3323: -77.2395202, 39.1885819
[473/2406] Index 3338: -77.0507011, 39.0369934
[474/2406] Index 3346: -77.2133834, 39.2773498
[475/2406] Index 3352: -77.0887791, 38.966956
[476/2406] Index 3355: -77.09997489999999, 39.0784146
[477/2406] Index 3359: -77.1564864, 38.9909745
[478/2406] Index 3360: -77.1356295, 39.0764352
[479/2406] Index 3363: -77.0172908, 39.0948725
[480/2406] Index 3369: -77.096732, 38.9911745
[481/2406] Index 3380: -76.9859108, 39.0417993
[482/2406] Index 3387: -76.9877189, 39.1390333
[483/2406] Index 3390: -76.9472612, 39.0827914
[484/2406] Index 3401: -77.096732, 38.9911745
[485/2406] Index 3403: -77.0635149, 39.0799114
[486/2406] Index 3406: -77.05648939999999, 38.9961702
[487/2406] Index 3415: -77.2178036, 39.02324249999999
[488/2406] Index 3419: -76.94513239999999, 39.0794562
[489/2406] Index 3421: -77.02456269999999, 38.9949258
[490/2406] Index 3441: -77.0726375, 39.0782147
[491/2406] Index 3442: -76.93803299999999, 39.090077
[492/2406] Index 3464: -76.9869282, 39.0157488
[493/2406] Index 3467: -77.0580645, 39.0650044
[494/2406] Index 3471: -76.9453497, 39.0629967
[495/2406] Index 3472: -77.3380668, 39.2432644
[496/2406] Index 3475: -77.163574, 39.164211
[497/2406] Index 3479: -77.2726184, 39.1528746
[498/2406] Index 3482: -77.05648939999999, 38.9961702
[499/2406] Index 3502: -77.0962672, 38.9821416
[500/2406] Index 3506: -77.1185491, 39.0329419
```

```
[501/2406] Index 3510: -77.0516921, 39.0506864
[502/2406] Index 3521: -76.9472612, 39.0827914
[503/2406] Index 3527: -77.0919804, 39.0443376
[504/2406] Index 3530: -77.2165903, 39.268511
[505/2406] Index 3553: -77.1150734, 39.0564729
[506/2406] Index 3583: -77.0291769, 38.99432669999999
[507/2406] Index 3591: -77.1788461, 39.1281689
[508/2406] Index 3593: -77.1218317, 39.0848457
[509/2406] Index 3594: -77.2875245, 39.2301977
[510/2406] Index 3599: -77.07733859999999, 39.082604
[511/2406] Index 3611: -77.0810892, 39.0830252
[512/2406] Index 3613: -77.2091526, 39.2857514
[513/2406] Index 3614: -77.1564864, 38.9909745
[514/2406] Index 3625: -77.21004549999999, 39.1412831
[515/2406] Index 3629: -76.9472612, 39.0827914
[516/2406] Index 3632: -77.1577628, 39.1451481
[517/2406] Index 3640: -77.20211619999999, 39.2873064
[518/2406] Index 3647: -77.2327505, 39.2513626
[519/2406] Index 3653: -77.28863969999999, 39.289498
[520/2406] Index 3666: -76.9898226, 38.9991051
[521/2406] Index 3668: -77.0708149, 39.1333911
[522/2406] Index 3707: -77.1100651, 39.0517496
[523/2406] Index 3708: -77.2048251, 39.11228819999999
[524/2406] Index 3716: -77.0215992, 39.0390249
[525/2406] Index 3717: -77.0531, 39.061364
[526/2406] Index 3719: -77.156399, 39.1166735
[527/2406] Index 3720: -77.08653699999999, 38.96556140000001
[528/2406] Index 3727: -77.071968, 39.1516646
[529/2406] Index 3734: -77.0503559, 39.058254
[530/2406] Index 3740: -77.08727189999999, 38.9637719
[531/2406] Index 3752: -77.0503382, 39.0402974
[532/2406] Index 3755: -77.12170189999999, 39.0066596
[533/2406] Index 3760: -77.1630295, 38.9933386
[534/2406] Index 3769: -77.134969, 39.09467799999999
[535/2406] Index 3771: -77.08727189999999, 38.9637719
[536/2406] Index 3775: -76.96803369999999, 39.0565845
[537/2406] Index 3799: -77.29391629999999, 39.10040890000001
[538/2406] Index 3810: -77.1354221, 39.0959603
[539/2406] Index 3813: -77.2008352, 39.1173966
[540/2406] Index 3820: -77.2724365, 39.1755688
[541/2406] Index 3822: -76.99026599999999, 39.0445302
[542/2406] Index 3829: -77.0040186, 38.9984446
[543/2406] Index 3838: -77.26832379999999, 39.1805994
[544/2406] Index 3846: -77.0513692, 39.0603083
[545/2406] Index 3862: -77.0503559, 39.058254
[546/2406] Index 3864: -77.28286899999999, 39.1668802
[547/2406] Index 3866: -77.09408669999999, 39.1568247
[548/2406] Index 3875: -77.0266401, 39.0374797
[549/2406] Index 3876: -77.2319673, 39.117667
[550/2406] Index 3887: -77.2742373, 39.1467753
[551/2406] Index 3896: -76.9420805, 39.0858282
```

```
[552/2406] Index 3902: -77.0247497, 38.99575660000001
[553/2406] Index 3914: -77.2056459, 39.01591
[554/2406] Index 3921: -76.9677966, 39.0540986
[555/2406] Index 3925: -77.09365389999999, 39.0547944
[556/2406] Index 3931: -77.0797542, 39.0876618
[557/2406] Index 3943: -77.0746602, 39.1543922
[558/2406] Index 3950: -77.0511162, 38.9975909
[559/2406] Index 3957: -77.1218317, 39.0848457
[560/2406] Index 3967: -77.0988333, 38.9898621
[561/2406] Index 3968: -76.9420805, 39.0858282
[562/2406] Index 3979: -76.99274129999999, 39.0196541
[563/2406] Index 3980: -77.0511162, 38.9975909
[564/2406] Index 3993: -77.21862899999999, 39.26458299999999
[565/2406] Index 3997: -77.1328238, 39.0739878
[566/2406] Index 4001: -77.2502871, 39.2284677
[567/2406] Index 4004: -77.2528367, 39.1113677
[568/2406] Index 4005: -77.0790462, 38.99497059999999
[569/2406] Index 4011: -77.1016389, 39.0003388
[570/2406] Index 4020: -77.0515738, 39.05205430000001
[571/2406] Index 4021: -77.2692987, 39.1629982
[572/2406] Index 4022: -77.0511162, 38.9975909
[573/2406] Index 4026: -77.0795042, 39.0634033
[574/2406] Index 4027: -77.2327505, 39.2513626
[575/2406] Index 4028: -77.10145729999999, 38.9632952
[576/2406] Index 4029: -77.1862852, 39.188433
[577/2406] Index 4038: -77.0810892, 39.0830252
[578/2406] Index 4050: -76.94452369999999, 39.0565466
[579/2406] Index 4058: -77.0925096, 38.9806173
[580/2406] Index 4070: -77.1837431, 39.0760868
[581/2406] Index 4092: -77.1395149, 39.1044579
[582/2406] Index 4095: -77.15100699999999, 39.02315979999999
[583/2406] Index 4102: -77.096732, 38.9911745
[584/2406] Index 4109: -77.05184179999999, 39.0406395
[585/2406] Index 4119: -77.0763229, 39.0808091
[586/2406] Index 4120: -77.08217719999999, 39.0542509
[587/2406] Index 4135: -77.10839589999999, 39.0909214
[588/2406] Index 4137: -77.20211619999999, 39.2873064
[589/2406] Index 4138: -77.19909539999999, 39.1832489
[590/2406] Index 4142: -77.1666384, 39.1008338
[591/2406] Index 4148: -77.096732, 38.9911745
[592/2406] Index 4149: -76.9320314, 39.0795989
[593/2406] Index 4175: -77.1776258, 39.13513040000001
[594/2406] Index 4184: -77.4117346, 39.1426842
[595/2406] Index 4192: -77.1185491, 39.0329419
[596/2406] Index 4216: -77.096732, 38.9911745
[597/2406] Index 4218: -77.0284697, 38.9942096
[598/2406] Index 4221: -77.0104648, 39.0098985
[599/2406] Index 4223: -77.1075535, 39.0493426
[600/2406] Index 4224: -77.0040186, 38.9984446
[601/2406] Index 4239: -77.02966769999999, 38.9869576
[602/2406] Index 4242: -77.0023949, 39.0044805
```

```
[603/2406] Index 4259: -77.2657333, 39.161475
[604/2406] Index 4264: -77.0165935, 38.9904667
[605/2406] Index 4266: -77.16115909999999, 39.10657820000001
[606/2406] Index 4267: -77.13159259999999, 39.0942094
[607/2406] Index 4274: -77.13016069999999, 39.092175
[608/2406] Index 4287: -77.1403907, 39.1469528
[609/2406] Index 4295: -77.01160920000001, 38.9910696
[610/2406] Index 4309: -77.0513359, 39.0387866
[611/2406] Index 4318: -77.0889296, 39.0630047
[612/2406] Index 4320: -77.1564169, 39.0867698
[613/2406] Index 4324: -77.16600319999999, 39.2087033
[614/2406] Index 4332: -77.00716419999999, 39.0022426
[615/2406] Index 4333: -77.2515057, 39.1671724
[616/2406] Index 4338: -77.092275, 38.977352
[617/2406] Index 4339: -77.0165935, 38.9904667
[618/2406] Index 4341: -77.2604454, 39.1868928
[619/2406] Index 4346: -77.037466, 38.9958963
[620/2406] Index 4352: -77.114448, 38.9814131
[621/2406] Index 4353: -77.0511162, 38.9975909
[622/2406] Index 4356: -77.0778693, 39.0575276
[623/2406] Index 4363: -77.0482914, 39.0304179
[624/2406] Index 4365: -77.02966769999999, 38.9869576
[625/2406] Index 4378: -77.2724365, 39.1755688
[626/2406] Index 4382: -77.2553343, 39.2231551
[627/2406] Index 4384: -77.1948953, 39.14448
[628/2406] Index 4390: -77.0720712, 39.0777976
[629/2406] Index 4391: -77.27272669999999, 39.1549213
[630/2406] Index 4402: -77.2002711, 39.2870758
[631/2406] Index 4407: -77.0513692, 39.0603083
[632/2406] Index 4411: -77.1678427, 39.1440239
[633/2406] Index 4417: -77.0669593, 39.1560082
[634/2406] Index 4422: -77.21529910000001, 39.2684772
[635/2406] Index 4438: -77.2604454, 39.1868928
[636/2406] Index 4441: -77.05184179999999, 39.0406395
[637/2406] Index 4442: -77.102688, 38.9554212
[638/2406] Index 4473: -77.05184179999999, 39.0406395
[639/2406] Index 4479: -77.02456269999999, 38.9949258
[640/2406] Index 4481: -77.26629849999999, 39.1678662
[641/2406] Index 4484: -77.0194084, 38.9899254
[642/2406] Index 4499: -77.1772737, 39.1733025
[643/2406] Index 4500: -77.0503559, 39.058254
[644/2406] Index 4507: -77.1376535, 39.0811578
[645/2406] Index 4537: -77.0780384, 39.0618899
[646/2406] Index 4542: -77.1150734, 39.0564729
[647/2406] Index 4547: -77.05233, 39.047348
[648/2406] Index 4554: -77.10839589999999, 39.0909214
[649/2406] Index 4555: -77.0520682, 39.0693465
[650/2406] Index 4560: -77.03679, 38.9922193
[651/2406] Index 4572: -77.0503559, 39.058254
[652/2406] Index 4582: -77.05466729999999, 39.0484421
[653/2406] Index 4606: -77.096732, 38.9911745
```

```
[654/2406] Index 4612: -77.1056768, 39.0466561
[655/2406] Index 4619: -77.1759146, 39.1545295
[656/2406] Index 4624: -77.20849450000001, 39.2872181
[657/2406] Index 4626: -76.9403895, 39.0870183
[658/2406] Index 4629: -77.1771164, 39.0198132
[659/2406] Index 4631: -77.10881289999999, 39.1005688
[660/2406] Index 4632: -77.0808156, 39.0876561
[661/2406] Index 4633: -77.10881289999999, 39.1005688
[662/2406] Index 4634: -77.096732, 38.9911745
[663/2406] Index 4641: -77.1277433, 39.0727685
[664/2406] Index 4644: -77.26832379999999, 39.1805994
[665/2406] Index 4652: -77.1368481, 39.0772753
[666/2406] Index 4654: -77.1218317, 39.0848457
[667/2406] Index 4672: -77.0811566, 39.0243127
[668/2406] Index 4673: -77.1056768, 39.0466561
[669/2406] Index 4681: -77.0880087, 39.06434429999999
[670/2406] Index 4689: -77.2742373, 39.1467753
[671/2406] Index 4704: -77.10881289999999, 39.1005688
[672/2406] Index 4716: -76.982154, 39.0940539
[673/2406] Index 4719: -76.933341, 39.094852
[674/2406] Index 4730: -76.95688659999999, 39.0493731
[675/2406] Index 4740: -77.18598940000001, 39.1180423
[676/2406] Index 4747: -77.0499111, 39.0439884
[677/2406] Index 4761: -77.0524647, 39.0436051
[678/2406] Index 4771: -76.97502659999999, 39.0599924
[679/2406] Index 4809: -77.0672783, 39.1538273
[680/2406] Index 4810: -77.0804689, 39.0901267
[681/2406] Index 4814: -77.0482914, 39.0304179
[682/2406] Index 4815: -77.06737439999999, 39.1549633
[683/2406] Index 4825: -77.0507011, 39.0369934
[684/2406] Index 4837: -77.05184179999999, 39.0406395
[685/2406] Index 4842: -76.9403895, 39.0870183
[686/2406] Index 4851: -77.1359319, 39.1326001
[687/2406] Index 4857: -77.19296560000001, 39.1474639
[688/2406] Index 4862: -77.0330571, 39.0255961
[689/2406] Index 4869: -77.071968, 39.1516646
[690/2406] Index 4880: -76.9403895, 39.0870183
[691/2406] Index 4890: -77.0993246, 39.051896
[692/2406] Index 4892: -77.0320065, 38.9960781
[693/2406] Index 4898: -77.27785589999999, 39.1830215
[694/2406] Index 4920: -77.19296560000001, 39.1474639
[695/2406] Index 4921: -77.1654477, 39.1043408
[696/2406] Index 4930: -77.2787518, 39.1116095
[697/2406] Index 4933: -77.19296560000001, 39.1474639
[698/2406] Index 4938: -77.2724365, 39.1755688
[699/2406] Index 4964: -77.02966769999999, 38.9869576
[700/2406] Index 4970: -76.9753799, 39.0204953
[701/2406] Index 4995: -77.07860099999999, 39.0498689
[702/2406] Index 5008: -77.04122199999999, 39.11809179999999
[703/2406] Index 5010: -77.096732, 38.9911745
[704/2406] Index 5018: -77.0296393, 39.0413693
```

```
[705/2406] Index 5032: -76.9403895, 39.0870183
[706/2406] Index 5039: -77.27175299999999, 39.156692
[707/2406] Index 5058: -77.1360301, 39.0870662
[708/2406] Index 5068: -77.0991291, 38.9774035
[709/2406] Index 5073: -77.26498169999999, 39.2134459
[710/2406] Index 5077: -77.099536, 38.9900928
[711/2406] Index 5086: -77.26832379999999, 39.1805994
[712/2406] Index 5088: -76.9396507, 39.0890355
[713/2406] Index 5093: -77.2600427, 39.1134002
[714/2406] Index 5094: -77.10273579999999, 39.00523039999999
[715/2406] Index 5096: -77.1564864, 38.9909745
[716/2406] Index 5099: -77.2637907, 39.182903
[717/2406] Index 5114: -77.2742373, 39.1467753
[718/2406] Index 5116: -77.18905579999999, 39.1857936
[719/2406] Index 5124: -77.0320065, 38.9960781
[720/2406] Index 5130: -77.24991779999999, 39.228839
[721/2406] Index 5150: -77.1043462, 38.9908525
[722/2406] Index 5166: -77.0787021, 39.0295424
[723/2406] Index 5174: -77.19296560000001, 39.1474639
[724/2406] Index 5180: -76.9396507, 39.0890355
[725/2406] Index 5189: -77.2522485, 39.1931793
[726/2406] Index 5193: -77.055351, 39.0286801
[727/2406] Index 5194: -77.2050368, 39.267095
[728/2406] Index 5202: -77.0804689, 39.0901267
[729/2406] Index 5210: -77.0750647, 39.0150467
[730/2406] Index 5211: -77.2649107, 39.1014875
[731/2406] Index 5227: -77.20211619999999, 39.2873064
[732/2406] Index 5230: -77.1193251, 39.0748421
[733/2406] Index 5233: -77.05387379999999, 39.0504851
[734/2406] Index 5241: -77.0932661, 39.088362
[735/2406] Index 5243: -77.0669763, 39.1483588
[736/2406] Index 5245: -77.0872452, 39.0666198
[737/2406] Index 5252: -77.0366963, 39.0015611
[738/2406] Index 5257: -77.1809099, 39.0020277
[739/2406] Index 5260: -77.0669763, 39.1483588
[740/2406] Index 5269: -77.1837431, 39.0760868
[741/2406] Index 5270: -77.2604454, 39.1868928
[742/2406] Index 5274: -77.24991779999999, 39.228839
[743/2406] Index 5281: -77.0804689, 39.0901267
[744/2406] Index 5292: -77.1982414, 39.3043045
[745/2406] Index 5294: -76.98073830000001, 39.09080549999999
[746/2406] Index 5297: -76.9414044, 39.0842899
[747/2406] Index 5304: -77.2142329, 39.0854569
[748/2406] Index 5320: -76.9748036, 39.0201898
[749/2406] Index 5330: -77.03456840000001, 39.0261093
[750/2406] Index 5333: -77.2181457, 39.2700913
[751/2406] Index 5345: -77.1218317, 39.0848457
[752/2406] Index 5354: -77.1056768, 39.0466561
[753/2406] Index 5357: -77.1972425, 39.1836612
[754/2406] Index 5366: -77.1788461, 39.1281689
[755/2406] Index 5367: -77.05341059999999, 39.0478294
```

```
[756/2406] Index 5368: -77.037466, 38.9958963
[757/2406] Index 5377: -77.190445, 39.1499694
[758/2406] Index 5381: -77.27262379999999, 39.2237091
[759/2406] Index 5384: -77.037466, 38.9958963
[760/2406] Index 5405: -77.04057139999999, 38.9991745
[761/2406] Index 5414: -77.0957566, 39.0479517
[762/2406] Index 5416: -77.1625364, 39.1022238
[763/2406] Index 5418: -76.99971590000001, 39.0369836
[764/2406] Index 5426: -77.12657639999999, 39.0728166
[765/2406] Index 5428: -77.21130169999999, 39.2835196
[766/2406] Index 5429: -77.0704485, 39.0770626
[767/2406] Index 5445: -77.0040186, 38.9984446
[768/2406] Index 5446: -77.03456840000001, 39.0261093
[769/2406] Index 5455: -77.0296393, 39.0413693
[770/2406] Index 5457: -77.2459984, 39.0859873
[771/2406] Index 5466: -77.096732, 38.9911745
[772/2406] Index 5467: -77.1360301, 39.0870662
[773/2406] Index 5486: -77.2008352, 39.1173966
[774/2406] Index 5494: -76.9399071, 39.0900694
[775/2406] Index 5545: -77.07860099999999, 39.0498689
[776/2406] Index 5548: -77.2181672, 38.980492
[777/2406] Index 5562: -77.11666989999999, 38.9694155
[778/2406] Index 5571: -77.19701719999999, 39.1730914
[779/2406] Index 5574: -77.1668188, 39.0891173
[780/2406] Index 5576: -77.24991779999999, 39.228839
[781/2406] Index 5582: -77.20027449999999, 39.186097
[782/2406] Index 5583: -77.0633007, 39.0796727
[783/2406] Index 5586: -77.0503559, 39.058254
[784/2406] Index 5602: -77.10839589999999, 39.0909214
[785/2406] Index 5603: -77.1862852, 39.188433
[786/2406] Index 5608: -77.02456269999999, 38.9949258
[787/2406] Index 5609: -77.00038409999999, 39.0441495
[788/2406] Index 5626: -77.0448852, 39.0431205
[789/2406] Index 5629: -77.24991779999999, 39.228839
[790/2406] Index 5631: -77.02456269999999, 38.9949258
[791/2406] Index 5634: -77.0797542, 39.0876618
[792/2406] Index 5638: -77.05280719999999, 39.0517045
[793/2406] Index 5643: -76.9403895, 39.0870183
[794/2406] Index 5678: -77.0218648, 38.9971837
[795/2406] Index 5683: -77.0052675, 39.0315443
[796/2406] Index 5686: -77.05280719999999, 39.0517045
[797/2406] Index 5688: -77.17414819999999, 39.1278928
[798/2406] Index 5692: -77.0640532, 39.0655092
[799/2406] Index 5695: -77.0341713, 39.1593696
[800/2406] Index 5697: -77.1948953, 39.14448
[801/2406] Index 5701: -77.2742373, 39.1467753
[802/2406] Index 5704: -77.1415731, 39.0510628
[803/2406] Index 5706: -77.0320065, 38.9960781
[804/2406] Index 5708: -77.0804689, 39.0901267
[805/2406] Index 5735: -77.198882, 39.28654720000001
[806/2406] Index 5745: -77.0880087, 39.06434429999999
```

```
[807/2406] Index 5757: -77.27175299999999, 39.156692
[808/2406] Index 5761: -77.03456840000001, 39.0261093
[809/2406] Index 5766: -77.06584, 39.07928
[810/2406] Index 5778: -77.05280719999999, 39.0517045
[811/2406] Index 5785: -76.9472612, 39.0827914
[812/2406] Index 5806: -77.4206647, 39.1425355
[813/2406] Index 5810: -77.0495275, 39.0350568
[814/2406] Index 5817: -77.11343409999999, 39.057984
[815/2406] Index 5832: -77.18905579999999, 39.1857936
[816/2406] Index 5841: -76.9372195, 39.08154469999999
[817/2406] Index 5844: -76.9414044, 39.0842899
[818/2406] Index 5845: -77.0661733, 39.2054295
[819/2406] Index 5853: -77.11292259999999, 39.0213603
[820/2406] Index 5890: -77.027671, 39.01932619999999
[821/2406] Index 5900: -77.28758409999999, 39.2198997
[822/2406] Index 5907: -77.1167706, 39.0504708
[823/2406] Index 5909: -76.99137499999999, 39.0512846
[824/2406] Index 5915: -77.0745917, 39.0796341
[825/2406] Index 5918: -77.1056768, 39.0466561
[826/2406] Index 5921: -77.1862852, 39.188433
[827/2406] Index 5922: -77.0543926, 39.1639931
[828/2406] Index 5928: -77.092275, 38.977352
[829/2406] Index 5931: -77.05184179999999, 39.0406395
[830/2406] Index 5945: -76.9904531, 39.0854584
[831/2406] Index 5966: -77.10839589999999, 39.0909214
[832/2406] Index 5967: -77.02307569999999, 39.0689676
[833/2406] Index 5971: -77.110891, 39.048689
[834/2406] Index 5997: -77.09997489999999, 39.0784146
[835/2406] Index 5998: -77.26832379999999, 39.1805994
[836/2406] Index 6001: -77.0659909, 39.0520973
[837/2406] Index 6015: -77.0513359, 39.0387866
[838/2406] Index 6017: -77.2614478, 39.1846433
[839/2406] Index 6028: -77.190445, 39.1499694
[840/2406] Index 6033: -77.1948953, 39.14448
[841/2406] Index 6034: -77.2777092, 39.1845652
[842/2406] Index 6038: -77.0503559, 39.058254
[843/2406] Index 6050: -76.96803369999999, 39.0565845
[844/2406] Index 6052: -77.1477026, 39.0090732
[845/2406] Index 6056: -77.26832379999999, 39.1805994
[846/2406] Index 6073: -77.02456269999999, 38.9949258
[847/2406] Index 6077: -77.0691592, 39.1507428
[848/2406] Index 6085: -77.0320065, 38.9960781
[849/2406] Index 6094: -77.0513359, 39.0387866
[850/2406] Index 6095: -77.02456269999999, 38.9949258
[851/2406] Index 6104: -77.0633007, 39.0796727
[852/2406] Index 6114: -77.096732, 38.9911745
[853/2406] Index 6126: -77.05233, 39.047348
[854/2406] Index 6127: -77.2892317, 39.1717282
[855/2406] Index 6128: -77.05233, 39.047348
[856/2406] Index 6135: -77.15100699999999, 39.02315979999999
[857/2406] Index 6139: -77.2787518, 39.1116095
```

```
[858/2406] Index 6148: -77.20805159999999, 39.0211909
[859/2406] Index 6153: -77.1374668, 39.0781619
[860/2406] Index 6160: -77.0726375, 39.0782147
[861/2406] Index 6165: -77.09940499999999, 38.9615662
[862/2406] Index 6166: -76.9396507, 39.0890355
[863/2406] Index 6174: -77.1268809, 39.0963928
[864/2406] Index 6176: -77.0548489, 39.0624821
[865/2406] Index 6191: -77.04364129999999, 39.0377771
[866/2406] Index 6200: -76.9348042, 39.1116179
[867/2406] Index 6204: -77.2731171, 39.1786765
[868/2406] Index 6215: -77.0991291, 38.9774035
[869/2406] Index 6218: -77.099536, 38.9900928
[870/2406] Index 6221: -77.1374668, 39.0781619
[871/2406] Index 6225: -77.0880087, 39.06434429999999
[872/2406] Index 6245: -77.1374668, 39.0781619
[873/2406] Index 6252: -77.0133321, 38.9998929
[874/2406] Index 6258: -77.2637907, 39.182903
[875/2406] Index 6259: -76.9403895, 39.0870183
[876/2406] Index 6260: -76.9453497, 39.0629967
[877/2406] Index 6261: -77.1145153, 39.0677783
[878/2406] Index 6262: -77.02456269999999, 38.9949258
[879/2406] Index 6285: -77.0654603, 39.0719491
[880/2406] Index 6288: -77.1747619, 39.1305079
[881/2406] Index 6293: -77.096732, 38.9911745
[882/2406] Index 6317: -76.9403895, 39.0870183
[883/2406] Index 6321: -77.1374668, 39.0781619
[884/2406] Index 6322: -77.0889296, 39.0630047
[885/2406] Index 6326: -77.059703, 39.0510695
[886/2406] Index 6335: -77.07860099999999, 39.0498689
[887/2406] Index 6336: -77.1424286, 39.0887906
[888/2406] Index 6351: -77.2879931, 39.1730156
[889/2406] Index 6354: -77.0962672, 38.9821416
[890/2406] Index 6362: -77.0911273, 39.0419381
[891/2406] Index 6363: -76.95688659999999, 39.0493731
[892/2406] Index 6364: -77.17498189999999, 39.1529138
[893/2406] Index 6367: -77.1145153, 39.0677783
[894/2406] Index 6368: -77.04514759999999, 39.0256368
[895/2406] Index 6378: -77.0448852, 39.0431205
[896/2406] Index 6380: -77.254024, 39.0640898
[897/2406] Index 6381: -76.9327084, 39.1172429
[898/2406] Index 6382: -77.0534906, 39.0462108
[899/2406] Index 6392: -77.096732, 38.9911745
[900/2406] Index 6393: -77.16619080000001, 39.1713074
[901/2406] Index 6417: -77.44809900000001, 39.1044704
[902/2406] Index 6418: -77.26541399999999, 39.185223
[903/2406] Index 6419: -76.9472612, 39.0827914
[904/2406] Index 6445: -77.1681907, 38.996311
[905/2406] Index 6447: -77.4090434, 39.14368750000001
[906/2406] Index 6470: -77.02828980000001, 38.9917849
[907/2406] Index 6473: -77.044902, 39.0205653
[908/2406] Index 6477: -77.0165935, 38.9904667
```

```
[909/2406] Index 6480: -77.0409949, 39.0410919
[910/2406] Index 6493: -77.0704485, 39.0770626
[911/2406] Index 6510: -77.1933889, 39.1759417
[912/2406] Index 6513: -77.037466, 38.9958963
[913/2406] Index 6526: -77.0218648, 38.9971837
[914/2406] Index 6546: -77.044902, 39.0205653
[915/2406] Index 6553: -77.0509596, 39.0527092
[916/2406] Index 6555: -77.0991291, 38.9774035
[917/2406] Index 6557: -77.0127272, 38.9911529
[918/2406] Index 6567: -77.26832379999999, 39.1805994
[919/2406] Index 6578: -77.0335862, 38.9967597
[920/2406] Index 6586: -77.0704485, 39.0770626
[921/2406] Index 6590: -76.9378869, 39.0833988
[922/2406] Index 6593: -77.0252639, 39.0434849
[923/2406] Index 6597: -77.0491605, 39.0393225
[924/2406] Index 6599: -77.05184179999999, 39.0406395
[925/2406] Index 6608: -77.0669593, 39.1560082
[926/2406] Index 6610: -76.9320314, 39.0795989
[927/2406] Index 6617: -77.08551469999999, 39.013921
[928/2406] Index 6625: -77.0308238, 39.0294929
[929/2406] Index 6627: -77.2371534, 39.1211752
[930/2406] Index 6629: -77.1694049, 39.14664399999999
[931/2406] Index 6630: -77.17495989999999, 39.1529136
[932/2406] Index 6631: -76.92784859999999, 39.101697
[933/2406] Index 6632: -77.15100699999999, 39.02315979999999
[934/2406] Index 6639: -77.049961, 39.0536409
[935/2406] Index 6645: -77.4117776, 39.1444488
[936/2406] Index 6650: -77.0320065, 38.9960781
[937/2406] Index 6652: -77.1268809, 39.0963928
[938/2406] Index 6653: -76.9898226, 38.9991051
[939/2406] Index 6658: -77.2637907, 39.182903
[940/2406] Index 6665: -77.092275, 38.977352
[941/2406] Index 6679: -76.9859108, 39.0417993
[942/2406] Index 6696: -77.037466, 38.9958963
[943/2406] Index 6701: -77.071968, 39.1516646
[944/2406] Index 6703: -77.19296560000001, 39.1474639
[945/2406] Index 6706: -77.2371534, 39.1211752
[946/2406] Index 6711: -77.0513359, 39.0387866
[947/2406] Index 6712: -77.1564864, 38.9909745
[948/2406] Index 6718: -77.05184179999999, 39.0406395
[949/2406] Index 6720: -77.2677282, 39.16984739999999
[950/2406] Index 6728: -77.10145729999999, 38.9632952
[951/2406] Index 6737: -77.0787016, 39.0772915
[952/2406] Index 6743: -77.0552367, 39.0475383
[953/2406] Index 6748: -77.1150734, 39.0564729
[954/2406] Index 6749: -77.0691592, 39.1507428
[955/2406] Index 6773: -77.26832379999999, 39.1805994
[956/2406] Index 6794: -76.9558742, 39.0631143
[957/2406] Index 6797: -77.0491605, 39.0393225
[958/2406] Index 6798: -77.0495382, 39.0389167
[959/2406] Index 6801: -77.071968, 39.1516646
```

```
[960/2406] Index 6802: -77.0552367, 39.0475383
[961/2406] Index 6803: -77.0320065, 38.9960781
[962/2406] Index 6805: -76.9558742, 39.0631143
[963/2406] Index 6814: -77.091251, 38.9632933
[964/2406] Index 6820: -77.21571449999999, 39.2702698
[965/2406] Index 6828: -77.092275, 38.977352
[966/2406] Index 6836: -77.0797542, 39.0876618
[967/2406] Index 6839: -77.110891, 39.048689
[968/2406] Index 6841: -77.14303799999999, 39.1440208
[969/2406] Index 6863: -76.9378869, 39.0833988
[970/2406] Index 6864: -77.049961, 39.0536409
[971/2406] Index 6866: -77.19282009999999, 39.0439327
[972/2406] Index 6879: -77.0039874, 38.9997693
[973/2406] Index 6889: -76.9378869, 39.0833988
[974/2406] Index 6893: -77.0693411, 39.0659336
[975/2406] Index 6897: -77.0503559, 39.058254
[976/2406] Index 6900: -77.10458229999999, 38.9894803
[977/2406] Index 6905: -77.12206590000001, 39.1094468
[978/2406] Index 6909: -77.0773275, 39.1006362
[979/2406] Index 6911: -77.0754682, 38.9890593
[980/2406] Index 6912: -77.0955652, 39.0489839
[981/2406] Index 6915: -76.97910619999999, 39.0519185
[982/2406] Index 6930: -76.97910619999999, 39.0519185
[983/2406] Index 6932: -77.19296560000001, 39.1474639
[984/2406] Index 6946: -77.0640532, 39.0655092
[985/2406] Index 6947: -77.3432456, 39.2453683
[986/2406] Index 6954: -77.04057139999999, 38.9991745
[987/2406] Index 6956: -77.2057379, 39.2812858
[988/2406] Index 6958: -77.044902, 39.0205653
[989/2406] Index 6964: -77.0973258, 38.9889647
[990/2406] Index 6975: -77.0804689, 39.0901267
[991/2406] Index 6976: -77.092275, 38.977352
[992/2406] Index 6981: -77.07261919999999, 39.0308686
[993/2406] Index 6983: -77.04975730000001, 39.0404778
[994/2406] Index 6987: -77.0516921, 39.0506864
[995/2406] Index 6989: -77.04975730000001, 39.0404778
[996/2406] Index 6995: -77.044902, 39.0205653
[997/2406] Index 7002: -77.0165935, 38.9904667
[998/2406] Index 7003: -77.2614478, 39.1846433
[999/2406] Index 7007: -77.0490394, 39.032466
[1000/2406] Index 7010: -76.9403895, 39.0870183
[1001/2406] Index 7018: -76.9898226, 38.9991051
[1002/2406] Index 7021: -77.0509596, 39.0527092
[1003/2406] Index 7030: -76.95688659999999, 39.0493731
[1004/2406] Index 7032: -77.2057379, 39.2812858
[1005/2406] Index 7036: -77.0747311, 39.0628784
[1006/2406] Index 7052: -77.0936086, 39.0492134
[1007/2406] Index 7053: -77.05910810000002, 39.0644698
[1008/2406] Index 7055: -77.14303799999999, 39.1440208
[1009/2406] Index 7058: -77.19514079999999, 39.1449196
[1010/2406] Index 7061: -77.26866389999999, 39.1601131
```

```
[1011/2406] Index 7065: -77.27389699999999, 39.0480081
[1012/2406] Index 7073: -77.209713, 39.0231759
[1013/2406] Index 7075: -77.19909539999999, 39.1832489
[1014/2406] Index 7077: -77.0492534, 39.0262666
[1015/2406] Index 7081: -77.0693411, 39.0659336
[1016/2406] Index 7090: -77.049961, 39.0536409
[1017/2406] Index 7092: -77.092275, 38.977352
[1018/2406] Index 7100: -77.1701239, 39.2561568
[1019/2406] Index 7103: -77.27262379999999, 39.2237091
[1020/2406] Index 7116: -77.0511162, 38.9975909
[1021/2406] Index 7141: -76.9859108, 39.0417993
[1022/2406] Index 7145: -77.1086715, 39.055793
[1023/2406] Index 7152: -76.9898226, 38.9991051
[1024/2406] Index 7153: -77.0503559, 39.058254
[1025/2406] Index 7157: -77.037466, 38.9958963
[1026/2406] Index 7162: -77.0515738, 39.05205430000001
[1027/2406] Index 7163: -77.0218648, 38.9971837
[1028/2406] Index 7167: -77.05233, 39.047348
[1029/2406] Index 7175: -77.0503303, 39.0492751
[1030/2406] Index 7178: -77.2492405, 39.12297950000001
[1031/2406] Index 7187: -77.092275, 38.977352
[1032/2406] Index 7196: -77.071418, 39.0421745
[1033/2406] Index 7198: -77.0218648, 38.9971837
[1034/2406] Index 7202: -77.0513359, 39.0387866
[1035/2406] Index 7206: -77.07860099999999, 39.0498689
[1036/2406] Index 7211: -77.0521418, 39.0418728
[1037/2406] Index 7215: -77.0172008, 39.0970628
[1038/2406] Index 7216: -76.9420805, 39.0858282
[1039/2406] Index 7222: -77.20211619999999, 39.2873064
[1040/2406] Index 7230: -77.0531, 39.061364
[1041/2406] Index 7247: -77.2131675, 39.0312783
[1042/2406] Index 7249: -77.0348771, 39.0444036
[1043/2406] Index 7250: -77.2045867, 38.9905816
[1044/2406] Index 7259: -77.1162913, 39.0267214
[1045/2406] Index 7260: -76.99026599999999, 39.0445302
[1046/2406] Index 7270: -77.0818536, 39.0579318
[1047/2406] Index 7272: -77.2819517, 39.1664048
[1048/2406] Index 7282: -77.0513359, 39.0387866
[1049/2406] Index 7287: -77.02966769999999, 38.9869576
[1050/2406] Index 7294: -77.1564864, 38.9909745
[1051/2406] Index 7296: -76.9851717, 39.1137519
[1052/2406] Index 7297: -77.0781993, 39.0668224
[1053/2406] Index 7298: -77.096732, 38.9911745
[1054/2406] Index 7300: -76.9348042, 39.1116179
[1055/2406] Index 7304: -77.2355735, 39.1209347
[1056/2406] Index 7306: -77.0696332, 39.1132509
[1057/2406] Index 7307: -77.1354221, 39.0959603
[1058/2406] Index 7315: -76.97558839999999, 39.0669581
[1059/2406] Index 7326: -77.05233, 39.047348
[1060/2406] Index 7331: -77.0502849, 39.0399696
[1061/2406] Index 7334: -77.0503559, 39.058254
```

```
[1062/2406] Index 7335: -77.096732, 38.9911745
[1063/2406] Index 7336: -77.02456269999999, 38.9949258
[1064/2406] Index 7339: -77.03205, 39.0124489
[1065/2406] Index 7350: -77.1100651, 39.0517496
[1066/2406] Index 7357: -77.0218648, 38.9971837
[1067/2406] Index 7399: -77.0991291, 38.9774035
[1068/2406] Index 7404: -77.0515738, 39.05205430000001
[1069/2406] Index 7405: -77.1395149, 39.1044579
[1070/2406] Index 7406: -77.15176079999999, 39.2109536
[1071/2406] Index 7407: -77.10145729999999, 38.9632952
[1072/2406] Index 7413: -76.96803369999999, 39.0565845
[1073/2406] Index 7417: -77.43331859999999, 39.1222742
[1074/2406] Index 7424: -77.2829471, 39.2200531
[1075/2406] Index 7427: -77.0218648, 38.9971837
[1076/2406] Index 7435: -77.0811566, 39.0905365
[1077/2406] Index 7446: -77.135762, 39.0977372
[1078/2406] Index 7448: -77.08996929999999, 39.0557793
[1079/2406] Index 7453: -77.2660798, 39.2257001
[1080/2406] Index 7458: -77.135762, 39.0977372
[1081/2406] Index 7472: -77.0218648, 38.9971837
[1082/2406] Index 7476: -77.2875245, 39.2301977
[1083/2406] Index 7480: -77.1145153, 39.0677783
[1084/2406] Index 7482: -77.096732, 38.9911745
[1085/2406] Index 7503: -77.19514079999999, 39.1449196
[1086/2406] Index 7527: -77.21004549999999, 39.1412831
[1087/2406] Index 7529: -77.0052675, 39.0315443
[1088/2406] Index 7536: -77.0218648, 38.9971837
[1089/2406] Index 7547: -77.1167706, 39.0504708
[1090/2406] Index 7548: -76.9860298, 39.0157474
[1091/2406] Index 7552: -77.0756122, 39.0585367
[1092/2406] Index 7558: -77.0513359, 39.0387866
[1093/2406] Index 7567: -77.11610639999999, 39.0493736
[1094/2406] Index 7570: -77.0695104, 39.0728593
[1095/2406] Index 7576: -77.16775679999999, 39.1834375
[1096/2406] Index 7579: -77.0973258, 38.9889647
[1097/2406] Index 7582: -76.95688659999999, 39.0493731
[1098/2406] Index 7583: -77.037466, 38.9958963
[1099/2406] Index 7585: -77.19418329999999, 39.1450142
[1100/2406] Index 7587: -77.1276338, 38.9948247
[1101/2406] Index 7598: -77.1056768, 39.0466561
[1102/2406] Index 7599: -76.9403895, 39.0870183
[1103/2406] Index 7600: -77.17498189999999, 39.1529138
[1104/2406] Index 7603: -77.13159259999999, 39.0942094
[1105/2406] Index 7607: -77.05412729999999, 38.9950056
[1106/2406] Index 7615: -77.0495275, 39.0350568
[1107/2406] Index 7616: -77.2414349, 39.20007440000001
[1108/2406] Index 7617: -77.0320065, 38.9960781
[1109/2406] Index 7618: -77.02456269999999, 38.9949258
[1110/2406] Index 7627: -77.1056768, 39.0466561
[1111/2406] Index 7630: -77.02456269999999, 38.9949258
[1112/2406] Index 7641: -77.2875245, 39.2301977
```

```
[1113/2406] Index 7644: -77.0671495, 39.15114010000001
[1114/2406] Index 7645: -77.0516932, 39.0556323
[1115/2406] Index 7650: -77.19418329999999, 39.1450142
[1116/2406] Index 7651: -77.08435589999999, 38.963271
[1117/2406] Index 7662: -77.0218648, 38.9971837
[1118/2406] Index 7673: -77.2080678, 39.0167227
[1119/2406] Index 7678: -77.29393879999999, 39.2217882
[1120/2406] Index 7690: -77.1729754, 39.0497709
[1121/2406] Index 7712: -77.1150734, 39.0564729
[1122/2406] Index 7717: -77.0465547, 39.02237909999999
[1123/2406] Index 7718: -77.0612835, 39.0799494
[1124/2406] Index 7719: -77.1670266, 38.979011
[1125/2406] Index 7729: -76.9558742, 39.0631143
[1126/2406] Index 7731: -77.0630236, 39.0332479
[1127/2406] Index 7740: -76.9338125, 39.0900147
[1128/2406] Index 7752: -77.096732, 38.9911745
[1129/2406] Index 7756: -76.9472612, 39.0827914
[1130/2406] Index 7770: -76.9859108, 39.0417993
[1131/2406] Index 7772: -77.02456269999999, 38.9949258
[1132/2406] Index 7774: -77.0485448, 39.0492929
[1133/2406] Index 7777: -77.0194084, 38.9899254
[1134/2406] Index 7779: -77.26832379999999, 39.1805994
[1135/2406] Index 7786: -77.02966769999999, 38.9869576
[1136/2406] Index 7787: -77.0513359, 39.0387866
[1137/2406] Index 7792: -77.0499111, 39.0439884
[1138/2406] Index 7793: -77.02456269999999, 38.9949258
[1139/2406] Index 7795: -77.05940749999999, 39.17071079999999
[1140/2406] Index 7807: -77.2008352, 39.1173966
[1141/2406] Index 7814: -77.0754709, 39.030936
[1142/2406] Index 7817: -77.049961, 39.0536409
[1143/2406] Index 7820: -77.2008352, 39.1173966
[1144/2406] Index 7821: -77.1016389, 39.0003388
[1145/2406] Index 7837: -77.2756288, 39.186923
[1146/2406] Index 7848: -76.95688659999999, 39.0493731
[1147/2406] Index 7852: -77.21895669999999, 39.1756251
[1148/2406] Index 7857: -77.04975730000001, 39.0404778
[1149/2406] Index 7875: -77.0320065, 38.9960781
[1150/2406] Index 7878: -77.1033929, 38.9645541
[1151/2406] Index 7883: -77.2742373, 39.1467753
[1152/2406] Index 7885: -77.17498189999999, 39.1529138
[1153/2406] Index 7886: -77.0726375, 39.0782147
[1154/2406] Index 7889: -76.9403895, 39.0870183
[1155/2406] Index 7898: -77.0335862, 38.9967597
[1156/2406] Index 7899: -77.19296560000001, 39.1474639
[1157/2406] Index 7900: -77.2849834, 39.24470609999999
[1158/2406] Index 7914: -77.1862852, 39.188433
[1159/2406] Index 7917: -77.1197521, 38.9516143
[1160/2406] Index 7921: -77.1167706, 39.0504708
[1161/2406] Index 7923: -77.096732, 38.9911745
[1162/2406] Index 7925: -77.0726375, 39.0782147
[1163/2406] Index 7930: -77.17651149999999, 39.0628612
```

```
[1164/2406] Index 7934: -77.2875245, 39.2301977
[1165/2406] Index 7935: -77.0991291, 38.9774035
[1166/2406] Index 7937: -77.13539349999999, 39.0939105
[1167/2406] Index 7949: -77.18905579999999, 39.1857936
[1168/2406] Index 7958: -77.1986617, 39.07997840000001
[1169/2406] Index 7968: -77.05184179999999, 39.0406395
[1170/2406] Index 7979: -77.0503559, 39.058254
[1171/2406] Index 7988: -77.2742373, 39.1467753
[1172/2406] Index 7989: -77.1193251, 39.0748421
[1173/2406] Index 7991: -77.0503559, 39.058254
[1174/2406] Index 7993: -76.94513239999999, 39.0794562
[1175/2406] Index 7994: -76.9715432, 39.0284321
[1176/2406] Index 7998: -77.0671495, 39.15114010000001
[1177/2406] Index 7999: -76.9320314, 39.0795989
[1178/2406] Index 8000: -77.28039369999999, 39.2172918
[1179/2406] Index 8003: -76.9435713, 39.0836213
[1180/2406] Index 8015: -77.0291769, 38.99432669999999
[1181/2406] Index 8020: -77.2051482, 39.2896591
[1182/2406] Index 8024: -76.9435713, 39.0836213
[1183/2406] Index 8032: -77.096732, 38.9911745
[1184/2406] Index 8037: -77.1150734, 39.0564729
[1185/2406] Index 8053: -77.037466, 38.9958963
[1186/2406] Index 8054: -77.2095763, 39.1500846
[1187/2406] Index 8057: -76.9748036, 39.0201898
[1188/2406] Index 8086: -77.0542474, 39.0246929
[1189/2406] Index 8088: -77.05233, 39.047348
[1190/2406] Index 8099: -76.9568121, 39.0785899
[1191/2406] Index 8110: -77.2679237, 39.207809
[1192/2406] Index 8111: -77.1573672, 39.1840189
[1193/2406] Index 8113: -77.1100027, 39.0509191
[1194/2406] Index 8115: -77.1374668, 39.0781619
[1195/2406] Index 8122: -77.2614478, 39.1846433
[1196/2406] Index 8124: -77.0070363, 39.00257939999999
[1197/2406] Index 8136: -77.224117, 38.985018
[1198/2406] Index 8146: -77.26541399999999, 39.185223
[1199/2406] Index 8151: -77.2604454, 39.1868928
[1200/2406] Index 8165: -77.2614478, 39.1846433
[1201/2406] Index 8175: -77.0513359, 39.0387866
[1202/2406] Index 8179: -76.9403895, 39.0870183
[1203/2406] Index 8181: -77.0513359, 39.0387866
[1204/2406] Index 8185: -77.05184179999999, 39.0406395
[1205/2406] Index 8192: -77.09365389999999, 39.0547944
[1206/2406] Index 8199: -76.9860298, 39.0157474
[1207/2406] Index 8206: -77.2637907, 39.182903
[1208/2406] Index 8213: -77.1625364, 39.1022238
[1209/2406] Index 8214: -77.06873130000001, 39.149436
[1210/2406] Index 8215: -77.0671098, 39.1986646
[1211/2406] Index 8220: -77.0373762, 39.0337511
[1212/2406] Index 8221: -77.1493697, 38.9882729
[1213/2406] Index 8236: -77.21297520000002, 39.2747713
[1214/2406] Index 8244: -77.05233, 39.047348
```

```
[1215/2406] Index 8250: -77.0513692, 39.0603083
[1216/2406] Index 8263: -77.03456840000001, 39.0261093
[1217/2406] Index 8274: -77.1710883, 39.2588837
[1218/2406] Index 8277: -77.0482349, 39.0437129
[1219/2406] Index 8282: -77.1009794, 39.1831184
[1220/2406] Index 8302: -77.0682222, 39.078769
[1221/2406] Index 8329: -77.1163752, 38.9806085
[1222/2406] Index 8335: -77.0983151, 39.0693602
[1223/2406] Index 8337: -76.9420805, 39.0858282
[1224/2406] Index 8344: -76.9751974, 39.0460319
[1225/2406] Index 8351: -77.0507011, 39.0369934
[1226/2406] Index 8354: -77.0691592, 39.1507428
[1227/2406] Index 8356: -77.0127272, 38.9911529
[1228/2406] Index 8365: -77.0482349, 39.0437129
[1229/2406] Index 8366: -77.04761260000001, 39.208609
[1230/2406] Index 8369: -77.0973258, 38.9889647
[1231/2406] Index 8373: -77.0259615, 38.9905203
[1232/2406] Index 8374: -77.0503628, 39.054544
[1233/2406] Index 8375: -77.1210048, 38.992934
[1234/2406] Index 8377: -77.2115898, 39.0165556
[1235/2406] Index 8383: -76.9645737, 39.0291019
[1236/2406] Index 8408: -77.0218648, 38.9971837
[1237/2406] Index 8410: -77.096732, 38.9911745
[1238/2406] Index 8417: -77.096732, 38.9911745
[1239/2406] Index 8421: -77.1656074, 39.1371709
[1240/2406] Index 8434: -77.0507011, 39.0369934
[1241/2406] Index 8435: -77.02456269999999, 38.9949258
[1242/2406] Index 8436: -77.0637563, 39.0327809
[1243/2406] Index 8446: -77.09313999999999, 39.0495848
[1244/2406] Index 8452: -77.0635149, 39.0799114
[1245/2406] Index 8456: -77.096732, 38.9911745
[1246/2406] Index 8457: -77.0691592, 39.1507428
[1247/2406] Index 8459: -77.02456269999999, 38.9949258
[1248/2406] Index 8461: -77.2091526, 39.2857514
[1249/2406] Index 8468: -77.0716671, 39.1318185
[1250/2406] Index 8470: -77.2115898, 39.0165556
[1251/2406] Index 8473: -77.071418, 39.0421745
[1252/2406] Index 8485: -77.0165935, 38.9904667
[1253/2406] Index 8487: -77.2115898, 39.0165556
[1254/2406] Index 8500: -77.0691592, 39.1507428
[1255/2406] Index 8503: -77.0320065, 38.9960781
[1256/2406] Index 8504: -77.1837431, 39.0760868
[1257/2406] Index 8540: -77.0513692, 39.0603083
[1258/2406] Index 8541: -77.0507011, 39.0369934
[1259/2406] Index 8547: -77.08279279999999, 38.9873893
[1260/2406] Index 8548: -77.07477089999999, 39.0616151
[1261/2406] Index 8562: -77.071968, 39.1516646
[1262/2406] Index 8581: -77.0627216, 39.0526522
[1263/2406] Index 8582: -77.04875, 39.0714088
[1264/2406] Index 8589: -77.0704485, 39.0770626
[1265/2406] Index 8600: -77.08899819999999, 39.205462
```

```
[1266/2406] Index 8604: -77.2637907, 39.182903
[1267/2406] Index 8611: -77.05233, 39.047348
[1268/2406] Index 8612: -77.2355735, 39.1209347
[1269/2406] Index 8621: -77.23406419999999, 39.173114
[1270/2406] Index 8626: -77.10314900000002, 38.95671
[1271/2406] Index 8638: -77.037466, 38.9958963
[1272/2406] Index 8670: -77.06737439999999, 39.1549633
[1273/2406] Index 8688: -77.0670894, 39.0739288
[1274/2406] Index 8709: -77.04975730000001, 39.0404778
[1275/2406] Index 8711: -77.2742373, 39.1467753
[1276/2406] Index 8737: -77.1048279, 38.9622969
[1277/2406] Index 8738: -76.9320314, 39.0795989
[1278/2406] Index 8741: -77.209577, 39.1770962
[1279/2406] Index 8747: -77.16569779999999, 38.99494929999999
[1280/2406] Index 8752: -76.9414044, 39.0842899
[1281/2406] Index 8769: -76.9420805, 39.0858282
[1282/2406] Index 8773: -77.0214459, 39.0184461
[1283/2406] Index 8782: -77.05671869999999, 39.0462634
[1284/2406] Index 8799: -77.0133321, 38.9998929
[1285/2406] Index 8801: -77.0779125, 39.0836874
[1286/2406] Index 8811: -77.110891, 39.048689
[1287/2406] Index 8825: -77.06608609999999, 39.07303479999999
[1288/2406] Index 8827: -77.03336089999999, 39.1382043
[1289/2406] Index 8828: -77.18905579999999, 39.1857936
[1290/2406] Index 8839: -77.0785044, 39.0455767
[1291/2406] Index 8844: -77.06098209999999, 39.0355053
[1292/2406] Index 8847: -77.0513692, 39.0603083
[1293/2406] Index 8855: -77.0133321, 38.9998929
[1294/2406] Index 8865: -77.096732, 38.9911745
[1295/2406] Index 8867: -77.0218648, 38.9971837
[1296/2406] Index 8870: -77.1374668, 39.0781619
[1297/2406] Index 8876: -77.0341713, 39.1593696
[1298/2406] Index 8882: -77.0570594, 39.0808601
[1299/2406] Index 8895: -77.07648710000001, 39.1088267
[1300/2406] Index 8899: -77.1948953, 39.14448
[1301/2406] Index 8903: -77.0880087, 39.06434429999999
[1302/2406] Index 8906: -77.0507011, 39.0369934
[1303/2406] Index 8920: -77.19269779999999, 39.2505344
[1304/2406] Index 8922: -76.9365875, 39.0857799
[1305/2406] Index 8925: -77.2629418, 39.1887446
[1306/2406] Index 8933: -77.0507011, 39.0369934
[1307/2406] Index 8935: -77.0797542, 39.0876618
[1308/2406] Index 8941: -77.0320065, 38.9960781
[1309/2406] Index 8951: -77.0973258, 38.9889647
[1310/2406] Index 8973: -77.096732, 38.9911745
[1311/2406] Index 8976: -76.933341, 39.094852
[1312/2406] Index 8978: -77.0320065, 38.9960781
[1313/2406] Index 8980: -77.0931825, 38.9706984
[1314/2406] Index 8981: -77.0931825, 38.9706984
[1315/2406] Index 8983: -77.0792458, 39.0265508
[1316/2406] Index 8987: -77.1185491, 39.0329419
```

```
[1317/2406] Index 9005: -77.0021703, 39.0707695
[1318/2406] Index 9015: -77.21895669999999, 39.1756251
[1319/2406] Index 9036: -77.096732, 38.9911745
[1320/2406] Index 9040: -77.19183400000001, 39.3452828
[1321/2406] Index 9048: -77.11282829999999, 39.0032694
[1322/2406] Index 9050: -77.05412729999999, 38.9950056
[1323/2406] Index 9051: -77.1100651, 39.0517496
[1324/2406] Index 9064: -77.10839589999999, 39.0909214
[1325/2406] Index 9084: -77.0708149, 39.1333911
[1326/2406] Index 9088: -77.096732, 38.9911745
[1327/2406] Index 9103: -77.0252728, 39.01178540000001
[1328/2406] Index 9109: -77.13539349999999, 39.0939105
[1329/2406] Index 9123: -77.2742373, 39.1467753
[1330/2406] Index 9124: -77.2742373, 39.1467753
[1331/2406] Index 9147: -77.135762, 39.0977372
[1332/2406] Index 9155: -77.19514079999999, 39.1449196
[1333/2406] Index 9156: -77.1891179, 39.074223
[1334/2406] Index 9160: -77.20211619999999, 39.2873064
[1335/2406] Index 9171: -77.1056768, 39.0466561
[1336/2406] Index 9173: -77.2350039, 39.1907612
[1337/2406] Index 9182: -77.0363665, 38.99447019999999
[1338/2406] Index 9189: -77.096732, 38.9911745
[1339/2406] Index 9201: -77.0750647, 39.0150467
[1340/2406] Index 9214: -77.0704485, 39.0770626
[1341/2406] Index 9234: -77.110891, 39.048689
[1342/2406] Index 9238: -77.1831567, 39.0762953
[1343/2406] Index 9244: -76.97880479999999, 39.1064668
[1344/2406] Index 9252: -77.0932661, 39.088362
[1345/2406] Index 9256: -77.0745917, 39.0796341
[1346/2406] Index 9261: -76.95688659999999, 39.0493731
[1347/2406] Index 9264: -77.20849450000001, 39.2872181
[1348/2406] Index 9266: -77.0809486, 39.06691610000001
[1349/2406] Index 9267: -77.02966769999999, 38.9869576
[1350/2406] Index 9271: -77.0745917, 39.0796341
[1351/2406] Index 9273: -77.19418329999999, 39.1450142
[1352/2406] Index 9279: -77.1374668, 39.0781619
[1353/2406] Index 9285: -77.0571956, 39.0489609
[1354/2406] Index 9287: -77.04869280000001, 39.0390935
[1355/2406] Index 9298: -77.0571956, 39.0489609
[1356/2406] Index 9305: -77.1855401, 39.1246338
[1357/2406] Index 9316: -77.17498189999999, 39.1529138
[1358/2406] Index 9321: -77.21237789999999, 39.0203536
[1359/2406] Index 9324: -77.0507011, 39.0369934
[1360/2406] Index 9326: -77.0531, 39.061364
[1361/2406] Index 9333: -77.03679, 38.9922193
[1362/2406] Index 9348: -76.9435713, 39.0836213
[1363/2406] Index 9351: -77.2724365, 39.1755688
[1364/2406] Index 9356: -77.1075638, 39.2078525
[1365/2406] Index 9366: -77.0797542, 39.0876618
[1366/2406] Index 9371: -77.0631247, 39.0549676
[1367/2406] Index 9372: -77.1080269, 39.01055849999999
```

```
[1368/2406] Index 9374: -77.2601712, 39.2452183
[1369/2406] Index 9384: -77.17498189999999, 39.1529138
[1370/2406] Index 9387: -77.16191189999999, 39.1026153
[1371/2406] Index 9398: -76.9472612, 39.0827914
[1372/2406] Index 9400: -77.0513359, 39.0387866
[1373/2406] Index 9404: -76.9320314, 39.0795989
[1374/2406] Index 9418: -77.19269779999999, 39.2505344
[1375/2406] Index 9419: -77.4117776, 39.1444488
[1376/2406] Index 9426: -77.2637907, 39.182903
[1377/2406] Index 9431: -76.9751974, 39.0460319
[1378/2406] Index 9436: -77.20206809999999, 39.0081887
[1379/2406] Index 9452: -77.05412729999999, 38.9950056
[1380/2406] Index 9458: -77.1218317, 39.0848457
[1381/2406] Index 9460: -77.0991291, 38.9774035
[1382/2406] Index 9462: -77.1268809, 39.0963928
[1383/2406] Index 9469: -77.4033426, 39.1349349
[1384/2406] Index 9470: -77.0886215, 39.0687885
[1385/2406] Index 9471: -77.14267149999999, 39.08706230000001
[1386/2406] Index 9473: -77.26832379999999, 39.1805994
[1387/2406] Index 9478: -77.1268809, 39.0963928
[1388/2406] Index 9481: -76.9403895, 39.0870183
[1389/2406] Index 9484: -77.1197521, 38.9516143
[1390/2406] Index 9491: -77.1360301, 39.0870662
[1391/2406] Index 9505: -77.0594225, 39.0805058
[1392/2406] Index 9524: -77.2601712, 39.2452183
[1393/2406] Index 9533: -77.2350039, 39.1907612
[1394/2406] Index 9545: -77.19296560000001, 39.1474639
[1395/2406] Index 9561: -77.13539349999999, 39.0939105
[1396/2406] Index 9563: -77.0503559, 39.058254
[1397/2406] Index 9567: -77.10145729999999, 38.9632952
[1398/2406] Index 9573: -77.2637907, 39.182903
[1399/2406] Index 9580: -77.0804689, 39.0901267
[1400/2406] Index 9586: -77.09651819999999, 39.0798055
[1401/2406] Index 9590: -77.0671495, 39.15114010000001
[1402/2406] Index 9591: -77.210089, 39.1696591
[1403/2406] Index 9592: -77.2144649, 39.0321742
[1404/2406] Index 9615: -76.9403895, 39.0870183
[1405/2406] Index 9620: -77.2115898, 39.0165556
[1406/2406] Index 9622: -77.0991291, 38.9774035
[1407/2406] Index 9645: -77.2637907, 39.182903
[1408/2406] Index 9652: -77.0516921, 39.0506864
[1409/2406] Index 9655: -77.049961, 39.0536409
[1410/2406] Index 9662: -77.21130169999999, 39.2835196
[1411/2406] Index 9663: -77.037466, 38.9958963
[1412/2406] Index 9675: -77.0991291, 38.9774035
[1413/2406] Index 9681: -77.2657333, 39.161475
[1414/2406] Index 9689: -77.0482914, 39.0304179
[1415/2406] Index 9693: -77.10314900000002, 38.95671
[1416/2406] Index 9699: -77.2181672, 38.980492
[1417/2406] Index 9713: -77.1056768, 39.0466561
[1418/2406] Index 9723: -77.2601712, 39.2452183
```

```
[1419/2406] Index 9740: -77.14065090000001, 38.9837551
[1420/2406] Index 9750: -77.1056768, 39.0466561
[1421/2406] Index 9755: -77.26629849999999, 39.1678662
[1422/2406] Index 9756: -77.082461, 39.0305039
[1423/2406] Index 9759: -76.97558839999999, 39.0669581
[1424/2406] Index 9760: -77.02420339999999, 39.015439
[1425/2406] Index 9765: -77.06737439999999, 39.1549633
[1426/2406] Index 9781: -77.2875245, 39.2301977
[1427/2406] Index 9824: -77.2586498, 39.2317476
[1428/2406] Index 9829: -77.0513692, 39.0603083
[1429/2406] Index 9835: -77.110891, 39.048689
[1430/2406] Index 9846: -77.0373762, 39.0337511
[1431/2406] Index 9847: -77.0369025, 38.99313370000001
[1432/2406] Index 9860: -77.1395149, 39.1044579
[1433/2406] Index 9871: -77.0320065, 38.9960781
[1434/2406] Index 9874: -77.2724365, 39.1755688
[1435/2406] Index 9909: -77.1625364, 39.1022238
[1436/2406] Index 9939: -77.0622796, 39.0470141
[1437/2406] Index 9943: -77.0555068, 39.0458022
[1438/2406] Index 9956: -77.20211619999999, 39.2873064
[1439/2406] Index 9960: -77.07725959999999, 38.9948778
[1440/2406] Index 9961: -77.0194084, 38.9899254
[1441/2406] Index 9964: -77.190445, 39.1499694
[1442/2406] Index 9972: -77.0507011, 39.0369934
[1443/2406] Index 9978: -77.11343409999999, 39.057984
[1444/2406] Index 9981: -77.0277916, 39.0948125
[1445/2406] Index 9986: -77.04975730000001, 39.0404778
[1446/2406] Index 9997: -77.02456269999999, 38.9949258
[1447/2406] Index 10014: -77.0362699, 39.0366465
[1448/2406] Index 10030: -77.096732, 38.9911745
[1449/2406] Index 10034: -77.0635149, 39.0799114
[1450/2406] Index 10040: -77.2115898, 39.0165556
[1451/2406] Index 10041: -77.1790419, 39.0012991
[1452/2406] Index 10052: -77.1690282, 38.9733257
[1453/2406] Index 10062: -77.20324769999999, 39.2920412
[1454/2406] Index 10069: -77.19364879999999, 39.3273539
[1455/2406] Index 10075: -77.04967769999999, 39.011402
[1456/2406] Index 10076: -77.0218648, 38.9971837
[1457/2406] Index 10077: -77.20693210000002, 39.0164789
[1458/2406] Index 10082: -77.0524647, 39.0436051
[1459/2406] Index 10084: -77.0075093, 38.9967272
[1460/2406] Index 10112: -77.190445, 39.1499694
[1461/2406] Index 10115: -77.2115898, 39.0165556
[1462/2406] Index 10129: -77.04967769999999, 39.011402
[1463/2406] Index 10143: -77.26572689999999, 39.1628577
[1464/2406] Index 10144: -77.2373503, 39.2323981
[1465/2406] Index 10147: -77.04295359999999, 39.0252471
[1466/2406] Index 10150: -76.9472612, 39.0827914
[1467/2406] Index 10155: -77.1442376, 39.1493474
[1468/2406] Index 10157: -77.1056768, 39.0466561
[1469/2406] Index 10160: -77.0810892, 39.0830252
```

```
[1470/2406] Index 10165: -77.0633687, 39.0315347
[1471/2406] Index 10167: -77.2276104, 39.0370991
[1472/2406] Index 10169: -77.219678, 39.2745326
[1473/2406] Index 10170: -77.1150734, 39.0564729
[1474/2406] Index 10176: -77.02456269999999, 38.9949258
[1475/2406] Index 10181: -77.03679, 38.9922193
[1476/2406] Index 10187: -77.0516921, 39.0506864
[1477/2406] Index 10191: -77.0779125, 39.0836874
[1478/2406] Index 10200: -77.091251, 38.9632933
[1479/2406] Index 10204: -77.13367579999999, 38.9778763
[1480/2406] Index 10205: -77.0691592, 39.1507428
[1481/2406] Index 10209: -77.2118407, 39.2813446
[1482/2406] Index 10211: -77.0579836, 39.0471735
[1483/2406] Index 10212: -77.0910308, 38.9834634
[1484/2406] Index 10215: -77.02456269999999, 38.9949258
[1485/2406] Index 10227: -77.26572689999999, 39.1628577
[1486/2406] Index 10238: -77.0704485, 39.0770626
[1487/2406] Index 10252: -77.2829471, 39.2200531
[1488/2406] Index 10258: -77.17634819999999, 38.9962098
[1489/2406] Index 10273: -77.2008352, 39.1173966
[1490/2406] Index 10278: -77.02966769999999, 38.9869576
[1491/2406] Index 10286: -77.02828980000001, 38.9917849
[1492/2406] Index 10290: -77.02828980000001, 38.9917849
[1493/2406] Index 10291: -77.0507011, 39.0369934
[1494/2406] Index 10293: -77.1831567, 39.0762953
[1495/2406] Index 10294: -77.0513359, 39.0387866
[1496/2406] Index 10295: -77.2586498, 39.2317476
[1497/2406] Index 10300: -77.2097137, 39.1691788
[1498/2406] Index 10302: -77.0704485, 39.0770626
[1499/2406] Index 10304: -77.19296560000001, 39.1474639
[1500/2406] Index 10313: -77.26572689999999, 39.1628577
[1501/2406] Index 10318: -77.04975730000001, 39.0404778
[1502/2406] Index 10319: -77.08727189999999, 38.9637719
[1503/2406] Index 10320: -77.2008352, 39.1173966
[1504/2406] Index 10323: -77.203924, 39.2013141
[1505/2406] Index 10326: -77.1837431, 39.0760868
[1506/2406] Index 10327: -77.21004549999999, 39.1412831
[1507/2406] Index 10346: -77.26572689999999, 39.1628577
[1508/2406] Index 10347: -77.2742373, 39.1467753
[1509/2406] Index 10349: -76.9702623, 39.0523975
[1510/2406] Index 10351: -77.21004549999999, 39.1412831
[1511/2406] Index 10352: -77.0405946, 39.0022358
[1512/2406] Index 10355: -77.092275, 38.977352
[1513/2406] Index 10361: -77.0215992, 39.0390249
[1514/2406] Index 10367: -77.2742373, 39.1467753
[1515/2406] Index 10376: -77.21004549999999, 39.1412831
[1516/2406] Index 10377: -77.0320065, 38.9960781
[1517/2406] Index 10378: -77.198062, 39.3106608
[1518/2406] Index 10383: -77.2008352, 39.1173966
[1519/2406] Index 10406: -77.0218648, 38.9971837
[1520/2406] Index 10426: -77.0218648, 38.9971837
```

```
[1521/2406] Index 10431: -77.2586498, 39.2317476
[1522/2406] Index 10437: -77.04834840000001, 39.0246079
[1523/2406] Index 10440: -77.134969, 39.09467799999999
[1524/2406] Index 10451: -77.0804689, 39.0901267
[1525/2406] Index 10453: -77.096732, 38.9911745
[1526/2406] Index 10474: -77.26572689999999, 39.1628577
[1527/2406] Index 10475: -77.18905579999999, 39.1857936
[1528/2406] Index 10478: -76.9403895, 39.0870183
[1529/2406] Index 10479: -77.0520682, 39.0693465
[1530/2406] Index 10490: -77.0763229, 39.0808091
[1531/2406] Index 10492: -76.9748036, 39.0201898
[1532/2406] Index 10494: -77.064649, 39.0376891
[1533/2406] Index 10504: -77.02966769999999, 38.9869576
[1534/2406] Index 10507: -77.17498189999999, 39.1529138
[1535/2406] Index 10523: -77.0833823, 39.1376502
[1536/2406] Index 10524: -76.9873487, 39.0934061
[1537/2406] Index 10525: -77.1654477, 39.1043408
[1538/2406] Index 10534: -77.2376563, 39.1244818
[1539/2406] Index 10536: -77.18905579999999, 39.1857936
[1540/2406] Index 10548: -77.0052675, 39.0315443
[1541/2406] Index 10558: -77.0493637, 39.0116156
[1542/2406] Index 10559: -77.0304429, 38.9947772
[1543/2406] Index 10563: -77.0704485, 39.0770626
[1544/2406] Index 10569: -77.2055646, 39.11562199999999
[1545/2406] Index 10571: -77.0218648, 38.9971837
[1546/2406] Index 10582: -77.05697049999999, 39.0504564
[1547/2406] Index 10587: -77.12206590000001, 39.1094468
[1548/2406] Index 10589: -77.1860343, 39.00363249999999
[1549/2406] Index 10590: -77.210886, 39.0958119
[1550/2406] Index 10604: -77.1376629, 39.0274787
[1551/2406] Index 10606: -77.1252174, 39.0988934
[1552/2406] Index 10607: -77.0066985, 38.9971624
[1553/2406] Index 10609: -77.26832379999999, 39.1805994
[1554/2406] Index 10637: -77.12689499999999, 38.9759874
[1555/2406] Index 10646: -77.0550536, 39.0271085
[1556/2406] Index 10648: -77.2875245, 39.2301977
[1557/2406] Index 10659: -77.1013476, 39.0687271
[1558/2406] Index 10660: -77.2777092, 39.1845652
[1559/2406] Index 10681: -77.0177862, 38.9990192
[1560/2406] Index 10684: -77.0594225, 39.0805058
[1561/2406] Index 10695: -77.10839589999999, 39.0909214
[1562/2406] Index 10723: -77.08065169999999, 39.0656113
[1563/2406] Index 10747: -77.1762407, 39.06334409999999
[1564/2406] Index 10748: -77.2218728, 39.2624817
[1565/2406] Index 10750: -77.091319, 39.0110117
[1566/2406] Index 10756: -77.25201009999999, 39.10480320000001
[1567/2406] Index 10763: -77.0513359, 39.0387866
[1568/2406] Index 10765: -77.0354366, 39.0262049
[1569/2406] Index 10768: -77.037466, 38.9958963
[1570/2406] Index 10778: -77.0304429, 38.9947772
[1571/2406] Index 10789: -77.2492405, 39.12297950000001
```

```
[1572/2406] Index 10793: -77.0513359, 39.0387866
[1573/2406] Index 10804: -77.0531, 39.061364
[1574/2406] Index 10807: -77.0304429, 38.9947772
[1575/2406] Index 10822: -76.9898226, 38.9991051
[1576/2406] Index 10831: -77.0304429, 38.9947772
[1577/2406] Index 10837: -77.07952329999999, 39.0422433
[1578/2406] Index 10866: -77.0194084, 38.9899254
[1579/2406] Index 10870: -77.0962672, 38.9821416
[1580/2406] Index 10871: -77.2049452, 39.2920299
[1581/2406] Index 10881: -77.0304429, 38.9947772
[1582/2406] Index 10890: -77.0321359, 38.9974075
[1583/2406] Index 10902: -77.0499111, 39.0439884
[1584/2406] Index 10903: -77.0304429, 38.9947772
[1585/2406] Index 10910: -77.0509596, 39.0527092
[1586/2406] Index 10929: -77.0513359, 39.0387866
[1587/2406] Index 10936: -77.090615, 39.0130005
[1588/2406] Index 10950: -77.1056768, 39.0466561
[1589/2406] Index 10955: -77.0503628, 39.054544
[1590/2406] Index 10960: -77.037868, 39.083287
[1591/2406] Index 10962: -77.0454405, 39.0219839
[1592/2406] Index 10969: -77.0872364, 39.0349022
[1593/2406] Index 10981: -77.1150734, 39.0564729
[1594/2406] Index 10982: -77.0603656, 39.0616197
[1595/2406] Index 10988: -77.0635149, 39.0799114
[1596/2406] Index 10991: -77.1481175, 39.0068536
[1597/2406] Index 11007: -77.0284801, 39.0047968
[1598/2406] Index 11008: -77.0284697, 38.9942096
[1599/2406] Index 11010: -77.1374668, 39.0781619
[1600/2406] Index 11020: -77.0186854, 39.0791445
[1601/2406] Index 11021: -77.05233, 39.047348
[1602/2406] Index 11023: -77.096732, 38.9911745
[1603/2406] Index 11025: -77.190445, 39.1499694
[1604/2406] Index 11035: -77.2742373, 39.1467753
[1605/2406] Index 11039: -76.95688659999999, 39.0493731
[1606/2406] Index 11052: -77.1887747, 39.1438746
[1607/2406] Index 11073: -77.1356295, 39.0764352
[1608/2406] Index 11083: -77.0039874, 38.9997693
[1609/2406] Index 11088: -77.0507011, 39.0369934
[1610/2406] Index 11091: -77.0962672, 38.9821416
[1611/2406] Index 11098: -77.0507011, 39.0369934
[1612/2406] Index 11118: -77.0284697, 38.9942096
[1613/2406] Index 11119: -77.0507011, 39.0369934
[1614/2406] Index 11143: -77.02966769999999, 38.9869576
[1615/2406] Index 11146: -77.2657869, 39.19148819999999
[1616/2406] Index 11147: -76.9665761, 39.1165185
[1617/2406] Index 11150: -77.210886, 39.0958119
[1618/2406] Index 11161: -77.0872364, 39.0349022
[1619/2406] Index 11165: -76.95688659999999, 39.0493731
[1620/2406] Index 11173: -77.2642098, 39.1593777
[1621/2406] Index 11185: -77.0507011, 39.0369934
[1622/2406] Index 11193: -77.2642098, 39.1593777
```

```
[1623/2406] Index 11199: -77.229027, 39.0364362
[1624/2406] Index 11221: -77.096732, 38.9911745
[1625/2406] Index 11225: -76.99026599999999, 39.0445302
[1626/2406] Index 11243: -77.19514079999999, 39.1449196
[1627/2406] Index 11252: -77.0726375, 39.0782147
[1628/2406] Index 11263: -77.0872364, 39.0349022
[1629/2406] Index 11272: -77.17651149999999, 39.0628612
[1630/2406] Index 11285: -77.1056768, 39.0466561
[1631/2406] Index 11326: -77.1013754, 39.1022184
[1632/2406] Index 11328: -77.04967769999999, 39.011402
[1633/2406] Index 11330: -77.0693411, 39.0659336
[1634/2406] Index 11340: -77.0320065, 38.9960781
[1635/2406] Index 11342: -77.0704485, 39.0770626
[1636/2406] Index 11346: -77.1411955, 39.0939841
[1637/2406] Index 11347: -77.1374668, 39.0781619
[1638/2406] Index 11360: -77.0745917, 39.0796341
[1639/2406] Index 11368: -77.134186, 39.0751514
[1640/2406] Index 11371: -77.0513692, 39.0603083
[1641/2406] Index 11378: -77.2604454, 39.1868928
[1642/2406] Index 11392: -77.2614478, 39.1846433
[1643/2406] Index 11393: -77.27189760000002, 39.2230247
[1644/2406] Index 11402: -76.9898226, 38.9991051
[1645/2406] Index 11415: -77.06021400000002, 39.10024
[1646/2406] Index 11420: -77.0991409, 39.03750489999999
[1647/2406] Index 11425: -77.07860099999999, 39.0498689
[1648/2406] Index 11426: -76.9320314, 39.0795989
[1649/2406] Index 11431: -76.9396189, 39.0894465
[1650/2406] Index 11434: -77.0509059, 39.1708039
[1651/2406] Index 11439: -77.1374668, 39.0781619
[1652/2406] Index 11452: -77.2724365, 39.1755688
[1653/2406] Index 11453: -77.0513692, 39.0603083
[1654/2406] Index 11474: -77.11954990000001, 39.0259383
[1655/2406] Index 11476: -77.0720712, 39.0777976
[1656/2406] Index 11497: -77.0693411, 39.0659336
[1657/2406] Index 11498: -77.0670894, 39.0739288
[1658/2406] Index 11499: -77.190445, 39.1499694
[1659/2406] Index 11501: -77.02456269999999, 38.9949258
[1660/2406] Index 11503: -77.060367, 39.15128
[1661/2406] Index 11504: -77.4136152, 39.1423448
[1662/2406] Index 11508: -77.0956185, 38.9827421
[1663/2406] Index 11510: -77.0218648, 38.9971837
[1664/2406] Index 11519: -77.02828980000001, 38.9917849
[1665/2406] Index 11526: -77.02456269999999, 38.9949258
[1666/2406] Index 11531: -77.05997669999999, 39.22203409999999
[1667/2406] Index 11545: -77.0633687, 39.0315347
[1668/2406] Index 11552: -77.0320065, 38.9960781
[1669/2406] Index 11563: -77.0522975, 39.0596655
[1670/2406] Index 11565: -77.0612835, 39.0799494
[1671/2406] Index 11574: -77.096732, 38.9911745
[1672/2406] Index 11577: -77.0896937, 39.051896
[1673/2406] Index 11593: -77.19514079999999, 39.1449196
```

```
[1674/2406] Index 11595: -77.060367, 39.15128
[1675/2406] Index 11599: -76.9472612, 39.0827914
[1676/2406] Index 11600: -77.0633687, 39.0315347
[1677/2406] Index 11617: -77.096732, 38.9911745
[1678/2406] Index 11622: -77.0218648, 38.9971837
[1679/2406] Index 11625: -76.99026599999999, 39.0445302
[1680/2406] Index 11628: -77.1593321, 39.1706615
[1681/2406] Index 11636: -77.0720814, 39.1035596
[1682/2406] Index 11646: -77.15818229999999, 39.1474412
[1683/2406] Index 11654: -77.1396781, 39.1000257
[1684/2406] Index 11666: -76.9472612, 39.0827914
[1685/2406] Index 11673: -77.2371534, 39.1211752
[1686/2406] Index 11681: -77.207628, 39.26965200000001
[1687/2406] Index 11690: -76.9898226, 38.9991051
[1688/2406] Index 11693: -77.0669763, 39.1483588
[1689/2406] Index 11695: -77.2522485, 39.1931793
[1690/2406] Index 11701: -77.0915969, 38.9857681
[1691/2406] Index 11702: -77.096732, 38.9911745
[1692/2406] Index 11716: -77.135762, 39.0977372
[1693/2406] Index 11717: -77.0612835, 39.0799494
[1694/2406] Index 11719: -77.2604454, 39.1868928
[1695/2406] Index 11726: -77.16569779999999, 38.99494929999999
[1696/2406] Index 11731: -76.9738354, 39.0727449
[1697/2406] Index 11735: -77.02966769999999, 38.9869576
[1698/2406] Index 11739: -77.0507294, 39.0535097
[1699/2406] Index 11743: -77.01844439999999, 38.9861003
[1700/2406] Index 11749: -77.10197699999999, 39.0931164
[1701/2406] Index 11750: -77.21004549999999, 39.1412831
[1702/2406] Index 11762: -77.091251, 38.9632933
[1703/2406] Index 11772: -77.2406211, 39.2125543
[1704/2406] Index 11777: -77.1056768, 39.0466561
[1705/2406] Index 11779: -77.096732, 38.9911745
[1706/2406] Index 11799: -77.21529910000001, 39.2684772
[1707/2406] Index 11800: -77.096732, 38.9911745
[1708/2406] Index 11801: -77.0459346, 39.0238691
[1709/2406] Index 11809: -77.03456840000001, 39.0261093
[1710/2406] Index 11812: -76.9403895, 39.0870183
[1711/2406] Index 11832: -77.0503559, 39.058254
[1712/2406] Index 11838: -77.1177968, 39.1615606
[1713/2406] Index 11854: -76.96803369999999, 39.0565845
[1714/2406] Index 11866: -77.0722129, 39.0263513
[1715/2406] Index 11871: -77.1481175, 39.0068536
[1716/2406] Index 11873: -76.9403895, 39.0870183
[1717/2406] Index 11891: -77.2875245, 39.2301977
[1718/2406] Index 11897: -77.0745917, 39.0796341
[1719/2406] Index 11928: -77.19701719999999, 39.1730914
[1720/2406] Index 11931: -77.19296560000001, 39.1474639
[1721/2406] Index 11933: -77.07952329999999, 39.0422433
[1722/2406] Index 11952: -76.9885506, 39.01621
[1723/2406] Index 11956: -77.2614478, 39.1846433
[1724/2406] Index 11961: -76.9403895, 39.0870183
```

```
[1725/2406] Index 11979: -77.2875245, 39.2301977
[1726/2406] Index 11980: -77.2660798, 39.2257001
[1727/2406] Index 11986: -77.0693411, 39.0659336
[1728/2406] Index 11998: -76.9898226, 38.9991051
[1729/2406] Index 12003: -77.0880416, 39.0740484
[1730/2406] Index 12004: -77.3338783, 39.0741178
[1731/2406] Index 12006: -76.9158114, 39.1217561
[1732/2406] Index 12007: -77.4179784, 39.1462327
[1733/2406] Index 12017: -77.1133466, 39.0960664
[1734/2406] Index 12022: -77.274261, 39.1079578
[1735/2406] Index 12031: -77.0507011, 39.0369934
[1736/2406] Index 12037: -77.1481175, 39.0068536
[1737/2406] Index 12047: -77.0333504, 39.0027934
[1738/2406] Index 12052: -77.0218648, 38.9971837
[1739/2406] Index 12057: -77.02456269999999, 38.9949258
[1740/2406] Index 12062: -77.2118407, 39.2813446
[1741/2406] Index 12064: -77.05412729999999, 38.9950056
[1742/2406] Index 12067: -77.0168374, 39.0308045
[1743/2406] Index 12071: -77.05233, 39.047348
[1744/2406] Index 12083: -77.05184179999999, 39.0406395
[1745/2406] Index 12088: -77.096732, 38.9911745
[1746/2406] Index 12090: -77.0627216, 39.0526522
[1747/2406] Index 12092: -77.0610567, 39.17468360000001
[1748/2406] Index 12095: -77.0321359, 38.9974075
[1749/2406] Index 12104: -77.1218317, 39.0848457
[1750/2406] Index 12106: -77.18905579999999, 39.1857936
[1751/2406] Index 12114: -77.0304429, 38.9947772
[1752/2406] Index 12118: -77.0320065, 38.9960781
[1753/2406] Index 12130: -77.07860099999999, 39.0498689
[1754/2406] Index 12134: -77.02456269999999, 38.9949258
[1755/2406] Index 12152: -77.2208679, 39.00942670000001
[1756/2406] Index 12153: nan, nan
[1757/2406] Index 12209: -77.1185384, 38.9660363
[1758/2406] Index 12211: -77.07902849999999, 39.0553051
[1759/2406] Index 12212: -77.1811065, 39.144375
[1760/2406] Index 12220: -77.10197699999999, 39.0931164
[1761/2406] Index 12225: -77.05233, 39.047348
[1762/2406] Index 12228: -77.0846413, 39.1417104
[1763/2406] Index 12232: -77.2614478, 39.1846433
[1764/2406] Index 12238: -77.17498189999999, 39.1529138
[1765/2406] Index 12240: -77.0664547, 39.0426171
[1766/2406] Index 12248: -77.190445, 39.1499694
[1767/2406] Index 12249: -77.07860099999999, 39.0498689
[1768/2406] Index 12261: -77.0703125, 39.0525474
[1769/2406] Index 12262: -77.1670266, 38.979011
[1770/2406] Index 12268: -77.0218648, 38.9971837
[1771/2406] Index 12273: -76.9414044, 39.0842899
[1772/2406] Index 12275: -77.03205, 39.0124489
[1773/2406] Index 12276: -77.04834840000001, 39.0246079
[1774/2406] Index 12289: -77.1252174, 39.0988934
[1775/2406] Index 12291: -77.0499111, 39.0439884
```

```
[1776/2406] Index 12298: -77.2742373, 39.1467753
[1777/2406] Index 12300: -77.2724365, 39.1755688
[1778/2406] Index 12302: -77.2151743, 39.1715296
[1779/2406] Index 12307: -77.0477226, 39.0341297
[1780/2406] Index 12320: -76.9414044, 39.0842899
[1781/2406] Index 12322: -77.096732, 38.9911745
[1782/2406] Index 12325: -77.09334439999999, 39.1187769
[1783/2406] Index 12335: -77.1722834, 39.0080326
[1784/2406] Index 12340: -77.092275, 38.977352
[1785/2406] Index 12341: -77.0052675, 39.0315443
[1786/2406] Index 12343: -77.0304429, 38.9947772
[1787/2406] Index 12352: -77.23910649999999, 39.2409536
[1788/2406] Index 12361: -77.1694049, 39.14664399999999
[1789/2406] Index 12363: -77.0937627, 39.0752669
[1790/2406] Index 12365: -77.282617, 39.2219393
[1791/2406] Index 12367: -77.1396781, 39.1000257
[1792/2406] Index 12371: -77.2055646, 39.11562199999999
[1793/2406] Index 12375: -77.0507011, 39.0369934
[1794/2406] Index 12379: -76.97201869999999, 39.06709559999999
[1795/2406] Index 12381: -76.9241509, 39.1013841
[1796/2406] Index 12387: -76.9414044, 39.0842899
[1797/2406] Index 12398: -77.2614478, 39.1846433
[1798/2406] Index 12399: -77.0503559, 39.058254
[1799/2406] Index 12404: -77.1981265, 39.3092387
[1800/2406] Index 12405: -77.1363021, 39.0853828
[1801/2406] Index 12412: -77.2181457, 39.2700913
[1802/2406] Index 12413: -77.0772048, 39.0743893
[1803/2406] Index 12429: -77.0546806, 39.01534669999999
[1804/2406] Index 12453: -76.96803369999999, 39.0565845
[1805/2406] Index 12455: -77.4001969, 39.1389863
[1806/2406] Index 12459: -77.198882, 39.28654720000001
[1807/2406] Index 12466: -76.9435713, 39.0836213
[1808/2406] Index 12468: -77.2742373, 39.1467753
[1809/2406] Index 12472: -77.0304429, 38.9947772
[1810/2406] Index 12477: -77.0763229, 39.0808091
[1811/2406] Index 12482: -77.1056768, 39.0466561
[1812/2406] Index 12510: -77.2601712, 39.2452183
[1813/2406] Index 12511: -77.0516921, 39.0506864
[1814/2406] Index 12513: -77.0503559, 39.058254
[1815/2406] Index 12538: -76.9472612, 39.0827914
[1816/2406] Index 12541: -77.0218648, 38.9971837
[1817/2406] Index 12546: -77.17296850000001, 39.183898
[1818/2406] Index 12548: -77.1075535, 39.0493426
[1819/2406] Index 12551: -77.2875245, 39.2301977
[1820/2406] Index 12557: -77.2447321, 39.1129354
[1821/2406] Index 12562: -77.0811566, 39.0905365
[1822/2406] Index 12563: -77.0218648, 38.9971837
[1823/2406] Index 12569: -77.0751274, 39.0788977
[1824/2406] Index 12571: -76.94513239999999, 39.0794562
[1825/2406] Index 12588: -77.0168374, 39.0308045
[1826/2406] Index 12590: -77.26832379999999, 39.1805994
```

```
[1827/2406] Index 12594: -77.22093939999999, 39.3319037
[1828/2406] Index 12596: -77.20849450000001, 39.2872181
[1829/2406] Index 12598: -77.0964807, 39.0515476
[1830/2406] Index 12606: -77.0507294, 39.0535097
[1831/2406] Index 12610: -77.2875245, 39.2301977
[1832/2406] Index 12613: -77.049961, 39.0536409
[1833/2406] Index 12621: -77.2802401, 39.1475708
[1834/2406] Index 12628: -77.07733859999999, 39.082604
[1835/2406] Index 12638: -77.0503365, 39.0402975
[1836/2406] Index 12639: -77.26832379999999, 39.1805994
[1837/2406] Index 12649: -77.2469022, 39.23232890000001
[1838/2406] Index 12651: -77.19296560000001, 39.1474639
[1839/2406] Index 12656: -77.0659129, 39.1393463
[1840/2406] Index 12664: -77.0513359, 39.0387866
[1841/2406] Index 12665: -77.1150734, 39.0564729
[1842/2406] Index 12673: -77.1268809, 39.0963928
[1843/2406] Index 12689: -77.0284697, 38.9942096
[1844/2406] Index 12697: -77.110891, 39.048689
[1845/2406] Index 12704: -77.0218648, 38.9971837
[1846/2406] Index 12706: -77.20693210000002, 39.0164789
[1847/2406] Index 12729: -77.096732, 38.9911745
[1848/2406] Index 12730: -77.110891, 39.048689
[1849/2406] Index 12743: -77.0304429, 38.9947772
[1850/2406] Index 12750: -77.20693210000002, 39.0164789
[1851/2406] Index 12771: -77.02721559999999, 39.0231312
[1852/2406] Index 12779: -76.96803369999999, 39.0565845
[1853/2406] Index 12783: -77.1252174, 39.0988934
[1854/2406] Index 12794: -77.04057139999999, 38.9991745
[1855/2406] Index 12797: -76.9898226, 38.9991051
[1856/2406] Index 12842: -77.0373762, 39.0337511
[1857/2406] Index 12851: -77.2190443, 39.0265846
[1858/2406] Index 12854: -77.2802401, 39.1475708
[1859/2406] Index 12855: -77.2055646, 39.11562199999999
[1860/2406] Index 12856: -77.0656774, 39.0673327
[1861/2406] Index 12858: -77.10839589999999, 39.0909214
[1862/2406] Index 12868: -76.99026599999999, 39.0445302
[1863/2406] Index 12882: -77.0874457, 39.0456479
[1864/2406] Index 12893: -76.9420805, 39.0858282
[1865/2406] Index 12895: -77.04410000000001, 39.0875657
[1866/2406] Index 12911: -77.203483, 39.1182696
[1867/2406] Index 12912: -77.052525, 38.996674
[1868/2406] Index 12913: -77.1225809, 39.0655468
[1869/2406] Index 12927: -77.08035629999999, 39.0745872
[1870/2406] Index 12930: -77.0868455, 39.0919572
[1871/2406] Index 12938: -77.0507011, 39.0369934
[1872/2406] Index 12945: -76.97470659999999, 39.09662369999999
[1873/2406] Index 12953: -77.2311941, 39.0981956
[1874/2406] Index 12963: -77.2055748, 39.2906426
[1875/2406] Index 12965: -77.19296560000001, 39.1474639
[1876/2406] Index 12971: -77.0165935, 38.9904667
[1877/2406] Index 12983: -77.4206647, 39.1425355
```

```
[1878/2406] Index 12997: -77.1972425, 39.1836612
[1879/2406] Index 13030: -77.1196397, 39.0703535
[1880/2406] Index 13031: -77.2892317, 39.1717282
[1881/2406] Index 13044: -76.9378869, 39.0833988
[1882/2406] Index 13068: -77.0704485, 39.0770626
[1883/2406] Index 13091: -77.0691592, 39.1507428
[1884/2406] Index 13095: -77.2210782, 39.2072958
[1885/2406] Index 13098: -77.0731534, 39.1265915
[1886/2406] Index 13104: -76.9859108, 39.0417993
[1887/2406] Index 13107: -77.2875245, 39.2301977
[1888/2406] Index 13113: -77.2226171, 39.1643683
[1889/2406] Index 13134: -77.1465355, 39.0858928
[1890/2406] Index 13140: -77.0726375, 39.0782147
[1891/2406] Index 13155: -77.0804689, 39.0901267
[1892/2406] Index 13160: -77.0991291, 38.9774035
[1893/2406] Index 13166: -77.08727189999999, 38.9637719
[1894/2406] Index 13168: -77.0633007, 39.0796727
[1895/2406] Index 13171: -77.13367579999999, 38.9778763
[1896/2406] Index 13175: -77.0922022, 39.0747489
[1897/2406] Index 13181: -76.9339092, 39.0814418
[1898/2406] Index 13184: -77.2677282, 39.16984739999999
[1899/2406] Index 13195: -76.9399071, 39.0900694
[1900/2406] Index 13203: -77.11587279999999, 39.0188542
[1901/2406] Index 13205: -77.074524, 39.2071722
[1902/2406] Index 13208: -77.0804689, 39.0901267
[1903/2406] Index 13211: -77.2095763, 39.1500846
[1904/2406] Index 13214: -77.2660798, 39.2257001
[1905/2406] Index 13262: -77.1360301, 39.0870662
[1906/2406] Index 13264: -77.0886215, 39.0687885
[1907/2406] Index 13274: -77.0664547, 39.0426171
[1908/2406] Index 13275: -77.02966769999999, 38.9869576
[1909/2406] Index 13278: -77.096732, 38.9911745
[1910/2406] Index 13287: -77.340037, 39.07878710000001
[1911/2406] Index 13295: -77.2118407, 39.2813446
[1912/2406] Index 13296: -77.0482408, 39.0413702
[1913/2406] Index 13303: -77.0515738, 39.05205430000001
[1914/2406] Index 13324: -77.0373762, 39.0337511
[1915/2406] Index 13329: -77.1223728, 39.0814241
[1916/2406] Index 13335: -77.0304429, 38.9947772
[1917/2406] Index 13338: -77.26832379999999, 39.1805994
[1918/2406] Index 13341: -77.2731171, 39.1786765
[1919/2406] Index 13357: -77.0487782, 39.0703681
[1920/2406] Index 13363: -77.1716642, 39.0986657
[1921/2406] Index 13366: -77.2115898, 39.0165556
[1922/2406] Index 13371: -77.0503365, 39.0402975
[1923/2406] Index 13373: -77.1707797, 39.1668502
[1924/2406] Index 13374: -77.1762407, 39.06334409999999
[1925/2406] Index 13379: -77.1268809, 39.0963928
[1926/2406] Index 13384: -77.10197699999999, 39.0931164
[1927/2406] Index 13405: -77.0745917, 39.0796341
[1928/2406] Index 13407: -77.1925009, 39.1517393
```

```
[1929/2406] Index 13413: -76.9378869, 39.0833988
[1930/2406] Index 13418: -77.0800719, 39.0858275
[1931/2406] Index 13422: -77.20825599999999, 39.139288
[1932/2406] Index 13428: -77.0459346, 39.0238691
[1933/2406] Index 13432: -77.210756, 39.2914955
[1934/2406] Index 13433: -77.2726184, 39.1528746
[1935/2406] Index 13441: -77.1416349, 39.0868494
[1936/2406] Index 13452: -77.0165935, 38.9904667
[1937/2406] Index 13458: -77.1056768, 39.0466561
[1938/2406] Index 13467: -77.23614239999999, 39.1941032
[1939/2406] Index 13471: -77.09365389999999, 39.0547944
[1940/2406] Index 13475: -77.0492251, 39.05346979999999
[1941/2406] Index 13484: -77.19296560000001, 39.1474639
[1942/2406] Index 13485: -77.1218317, 39.0848457
[1943/2406] Index 13488: -77.1710023, 39.1811565
[1944/2406] Index 13490: -77.1056768, 39.0466561
[1945/2406] Index 13506: -77.2072579, 39.25869280000001
[1946/2406] Index 13510: -77.0601944, 39.0498962
[1947/2406] Index 13516: -77.19296560000001, 39.1474639
[1948/2406] Index 13534: -77.072043, 39.0279637
[1949/2406] Index 13543: -77.0513359, 39.0387866
[1950/2406] Index 13546: -77.0887305, 39.1342584
[1951/2406] Index 13555: -77.27175299999999, 39.156692
[1952/2406] Index 13562: -77.1150734, 39.0564729
[1953/2406] Index 13572: -77.0275423, 38.9942077
[1954/2406] Index 13586: -77.0490394, 39.032466
[1955/2406] Index 13608: -77.037466, 38.9958963
[1956/2406] Index 13613: -77.096732, 38.9911745
[1957/2406] Index 13618: -77.190445, 39.1499694
[1958/2406] Index 13623: -77.05184179999999, 39.0406395
[1959/2406] Index 13624: -77.02828980000001, 38.9917849
[1960/2406] Index 13633: -77.0321359, 38.9974075
[1961/2406] Index 13642: -77.19296560000001, 39.1474639
[1962/2406] Index 13649: -77.4206647, 39.1425355
[1963/2406] Index 13657: -77.1564864, 38.9909745
[1964/2406] Index 13658: -77.0052675, 39.0315443
[1965/2406] Index 13662: -77.1030262, 39.0524126
[1966/2406] Index 13676: -77.096732, 38.9911745
[1967/2406] Index 13677: -77.2060573, 39.288471
[1968/2406] Index 13679: -77.190445, 39.1499694
[1969/2406] Index 13681: -77.17495989999999, 39.1529136
[1970/2406] Index 13693: -77.0325277, 39.0680401
[1971/2406] Index 13704: -76.9852236, 39.0128424
[1972/2406] Index 13706: -77.2649107, 39.1014875
[1973/2406] Index 13717: -76.9403895, 39.0870183
[1974/2406] Index 13728: -77.0548489, 39.0624821
[1975/2406] Index 13731: -77.2875245, 39.2301977
[1976/2406] Index 13739: -77.04057139999999, 38.9991745
[1977/2406] Index 13750: -77.21004549999999, 39.1412831
[1978/2406] Index 13751: -77.1765885, 39.1393941
[1979/2406] Index 13752: -77.0516921, 39.0506864
```

```
[1980/2406] Index 13757: -77.20211619999999, 39.2873064
[1981/2406] Index 13760: -77.0460288, 39.0286293
[1982/2406] Index 13771: -77.1898377, 39.1513944
[1983/2406] Index 13772: -77.1939438, 39.1879979
[1984/2406] Index 13773: -77.0177862, 38.9990192
[1985/2406] Index 13781: -77.2115898, 39.0165556
[1986/2406] Index 13788: -76.96247509999999, 39.0686413
[1987/2406] Index 13789: -77.0800719, 39.0858275
[1988/2406] Index 13792: -77.2138061, 39.0119435
[1989/2406] Index 13797: -77.2050368, 39.267095
[1990/2406] Index 13799: -77.06539169999999, 39.2005123
[1991/2406] Index 13803: -77.0552367, 39.0475383
[1992/2406] Index 13811: -76.9414044, 39.0842899
[1993/2406] Index 13813: -77.2637907, 39.182903
[1994/2406] Index 13819: -77.1885064, 39.1515574
[1995/2406] Index 13835: -77.04057139999999, 38.9991745
[1996/2406] Index 13836: -77.0330571, 39.0255961
[1997/2406] Index 13846: -77.20211619999999, 39.2873064
[1998/2406] Index 13848: -77.1480007, 39.1069963
[1999/2406] Index 13850: -77.0779125, 39.0836874
[2000/2406] Index 13855: -77.4179784, 39.1462327
[2001/2406] Index 13864: -77.110891, 39.048689
[2002/2406] Index 13866: -76.95688659999999, 39.0493731
[2003/2406] Index 13871: -77.0503559, 39.058254
[2004/2406] Index 13877: -77.21130169999999, 39.2835196
[2005/2406] Index 13898: -77.1480007, 39.1069963
[2006/2406] Index 13902: -77.1716642, 39.0986657
[2007/2406] Index 13904: -77.1710023, 39.1811565
[2008/2406] Index 13905: -77.1428425, 39.0048685
[2009/2406] Index 13909: -77.0669763, 39.1483588
[2010/2406] Index 13920: -77.0966854, 39.0816321
[2011/2406] Index 13937: -77.2614478, 39.1846433
[2012/2406] Index 13951: -77.1268809, 39.0963928
[2013/2406] Index 13967: -77.02456269999999, 38.9949258
[2014/2406] Index 13972: -77.1725276, 39.0056539
[2015/2406] Index 13979: -77.18905579999999, 39.1857936
[2016/2406] Index 13980: -77.0291769, 38.99432669999999
[2017/2406] Index 13982: -77.0513359, 39.0387866
[2018/2406] Index 13984: -77.096732, 38.9911745
[2019/2406] Index 13986: -77.1056768, 39.0466561
[2020/2406] Index 13988: -77.190445, 39.1499694
[2021/2406] Index 13991: -77.2637907, 39.182903
[2022/2406] Index 13996: -77.190445, 39.1499694
[2023/2406] Index 13999: -77.190445, 39.1499694
[2024/2406] Index 14002: -77.0708149, 39.1333911
[2025/2406] Index 14008: -77.190445, 39.1499694
[2026/2406] Index 14015: -77.0321359, 38.9974075
[2027/2406] Index 14017: -77.24355419999999, 39.1742477
[2028/2406] Index 14025: -77.1891179, 39.074223
[2029/2406] Index 14038: -77.052525, 38.996674
[2030/2406] Index 14042: -76.9859108, 39.0417993
```

```
[2031/2406] Index 14044: -77.1056768, 39.0466561
[2032/2406] Index 14056: -77.182754, 39.1862695
[2033/2406] Index 14066: -77.1056768, 39.0466561
[2034/2406] Index 14072: -76.9327084, 39.1172429
[2035/2406] Index 14073: -77.27337949999999, 39.1774144
[2036/2406] Index 14074: -77.1356295, 39.0764352
[2037/2406] Index 14084: -77.0726375, 39.0782147
[2038/2406] Index 14085: -77.19296560000001, 39.1474639
[2039/2406] Index 14095: -77.1891179, 39.074223
[2040/2406] Index 14112: -77.182754, 39.1862695
[2041/2406] Index 14115: -77.2724365, 39.1755688
[2042/2406] Index 14116: -77.0467737, 39.1410438
[2043/2406] Index 14120: -77.0582585, 39.0249844
[2044/2406] Index 14121: -77.2660798, 39.2257001
[2045/2406] Index 14130: -77.0359143, 39.0311234
[2046/2406] Index 14132: -77.2093673, 39.1803973
[2047/2406] Index 14136: -77.0627216, 39.0526522
[2048/2406] Index 14140: -77.1197521, 39.0542102
[2049/2406] Index 14141: -77.20211619999999, 39.2873064
[2050/2406] Index 14168: -77.2055646, 39.11562199999999
[2051/2406] Index 14169: -77.1675164, 39.1248176
[2052/2406] Index 14171: -77.0991291, 38.9774035
[2053/2406] Index 14173: -77.1075535, 39.0493426
[2054/2406] Index 14176: -77.0513359, 39.0387866
[2055/2406] Index 14188: -77.0194084, 38.9899254
[2056/2406] Index 14197: -77.0218648, 38.9971837
[2057/2406] Index 14199: -77.189471, 39.202259
[2058/2406] Index 14207: -76.99026599999999, 39.0445302
[2059/2406] Index 14223: -77.0726375, 39.0782147
[2060/2406] Index 14231: -77.0503559, 39.058254
[2061/2406] Index 14232: -77.11343409999999, 39.057984
[2062/2406] Index 14233: -77.04834840000001, 39.0246079
[2063/2406] Index 14239: -77.07393789999999, 39.122004
[2064/2406] Index 14244: -76.99026599999999, 39.0445302
[2065/2406] Index 14253: -77.2154374, 39.3148805
[2066/2406] Index 14258: -76.99026599999999, 39.0445302
[2067/2406] Index 14263: -77.01160920000001, 38.9910696
[2068/2406] Index 14272: -77.0621682, 39.0491378
[2069/2406] Index 14275: -77.1364919, 39.0535941
[2070/2406] Index 14276: -77.0726375, 39.0782147
[2071/2406] Index 14280: -77.0621682, 39.0491378
[2072/2406] Index 14281: -77.037466, 38.9958963
[2073/2406] Index 14282: -77.135762, 39.0977372
[2074/2406] Index 14284: -77.0490394, 39.032466
[2075/2406] Index 14294: -77.0363665, 38.99447019999999
[2076/2406] Index 14310: -77.2614478, 39.1846433
[2077/2406] Index 14318: -77.190445, 39.1499694
[2078/2406] Index 14325: -77.070274, 39.135743
[2079/2406] Index 14334: -77.0284697, 38.9942096
[2080/2406] Index 14357: -77.2839518, 39.1702882
[2081/2406] Index 14360: -77.0934925, 38.98267200000001
```

```
[2082/2406] Index 14363: -77.0509596, 39.0527092
[2083/2406] Index 14379: -77.2614478, 39.1846433
[2084/2406] Index 14382: -77.04869280000001, 39.0390935
[2085/2406] Index 14389: -77.190445, 39.1499694
[2086/2406] Index 14392: -77.0656249, 39.0676211
[2087/2406] Index 14393: -77.0639874, 39.0682538
[2088/2406] Index 14394: -77.0363665, 38.99447019999999
[2089/2406] Index 14397: -77.248353, 39.23031599999999
[2090/2406] Index 14401: -76.9865189, 39.0640232
[2091/2406] Index 14421: -77.2048251, 39.11228819999999
[2092/2406] Index 14423: -77.19296560000001, 39.1474639
[2093/2406] Index 14428: -77.0516921, 39.0506864
[2094/2406] Index 14432: -77.0503559, 39.058254
[2095/2406] Index 14447: -76.95688659999999, 39.0493731
[2096/2406] Index 14450: -77.16583899999999, 39.1415388
[2097/2406] Index 14457: -77.21004549999999, 39.1412831
[2098/2406] Index 14466: -77.0175395, 38.99796500000001
[2099/2406] Index 14467: -77.074879, 39.04566
[2100/2406] Index 14474: -77.0507011, 39.0369934
[2101/2406] Index 14479: -77.1145153, 39.0677783
[2102/2406] Index 14483: -77.0174128, 39.00700399999999
[2103/2406] Index 14484: -76.9420805, 39.0858282
[2104/2406] Index 14486: -77.0530059, 39.0570328
[2105/2406] Index 14500: -77.2637907, 39.182903
[2106/2406] Index 14522: -77.1016807, 39.1875516
[2107/2406] Index 14545: -77.06873130000001, 39.149436
[2108/2406] Index 14558: -77.2355735, 39.1209347
[2109/2406] Index 14561: -77.0320065, 38.9960781
[2110/2406] Index 14562: -77.1056768, 39.0466561
[2111/2406] Index 14573: -77.0715122, 39.0672479
[2112/2406] Index 14576: -77.2604454, 39.1868928
[2113/2406] Index 14578: -77.2094602, 39.0285387
[2114/2406] Index 14580: -77.10881289999999, 39.1005688
[2115/2406] Index 14588: -77.1394384, 39.0792326
[2116/2406] Index 14602: -77.0503559, 39.058254
[2117/2406] Index 14607: -77.037868, 39.083287
[2118/2406] Index 14617: -77.11292259999999, 39.0213603
[2119/2406] Index 14627: -77.0594225, 39.0805058
[2120/2406] Index 14628: -77.1240349, 38.9881728
[2121/2406] Index 14630: -77.0448852, 39.0431205
[2122/2406] Index 14637: -77.2763086, 39.1554598
[2123/2406] Index 14646: -77.0359143, 39.0311234
[2124/2406] Index 14654: -77.2657333, 39.161475
[2125/2406] Index 14655: -77.0494415, 38.9964805
[2126/2406] Index 14662: -77.02456269999999, 38.9949258
[2127/2406] Index 14676: -77.1860636, 39.126073
[2128/2406] Index 14692: -77.1808684, 39.1281947
[2129/2406] Index 14701: -77.1681907, 38.996311
[2130/2406] Index 14706: -77.2604454, 39.1868928
[2131/2406] Index 14723: -77.0991291, 38.9774035
[2132/2406] Index 14725: -77.02966769999999, 38.9869576
```

```
[2133/2406] Index 14734: -77.110891, 39.048689
[2134/2406] Index 14735: -77.0886215, 39.0687885
[2135/2406] Index 14740: -77.2614478, 39.1846433
[2136/2406] Index 14743: -77.1564864, 38.9909745
[2137/2406] Index 14751: -77.0476676, 39.0283969
[2138/2406] Index 14753: -77.0811566, 39.0905365
[2139/2406] Index 14765: -77.037466, 38.9958963
[2140/2406] Index 14782: -77.096732, 38.9911745
[2141/2406] Index 14799: -77.1573672, 39.1840189
[2142/2406] Index 14816: -76.96803369999999, 39.0565845
[2143/2406] Index 14817: -77.0513359, 39.0387866
[2144/2406] Index 14818: -77.0503559, 39.058254
[2145/2406] Index 14824: -77.0503559, 39.058254
[2146/2406] Index 14828: -77.2115898, 39.0165556
[2147/2406] Index 14834: -77.0218648, 38.9971837
[2148/2406] Index 14842: -77.096732, 38.9911745
[2149/2406] Index 14845: -77.2138061, 39.0119435
[2150/2406] Index 14850: -76.9414044, 39.0842899
[2151/2406] Index 14855: -77.17011649999999, 39.2036338
[2152/2406] Index 14861: -77.0745917, 39.0796341
[2153/2406] Index 14862: -77.0522975, 39.0596655
[2154/2406] Index 14865: -77.0325277, 39.0680401
[2155/2406] Index 14872: -77.0810892, 39.0830252
[2156/2406] Index 14874: -77.06873130000001, 39.149436
[2157/2406] Index 14876: -77.02966769999999, 38.9869576
[2158/2406] Index 14892: -77.1897451, 39.1897444
[2159/2406] Index 14893: -77.0513359, 39.0387866
[2160/2406] Index 14907: -77.1722834, 39.0080326
[2161/2406] Index 14909: -77.0092489, 38.99471740000001
[2162/2406] Index 14922: -77.0513359, 39.0387866
[2163/2406] Index 14930: -77.10273579999999, 39.00523039999999
[2164/2406] Index 14932: -77.0503559, 39.058254
[2165/2406] Index 14934: -77.0409949, 39.0410919
[2166/2406] Index 14941: -77.096732, 38.9911745
[2167/2406] Index 14942: -77.0218648, 38.9971837
[2168/2406] Index 14943: -76.94513239999999, 39.0794562
[2169/2406] Index 14949: -77.21004549999999, 39.1412831
[2170/2406] Index 14960: -77.0218648, 38.9971837
[2171/2406] Index 14965: -77.0671495, 39.15114010000001
[2172/2406] Index 14967: -77.2657333, 39.161475
[2173/2406] Index 14974: -77.10145729999999, 38.9632952
[2174/2406] Index 14975: -77.13685579999999, 38.9651816
[2175/2406] Index 14986: -77.21297520000002, 39.2747713
[2176/2406] Index 14988: -77.2724365, 39.1755688
[2177/2406] Index 14993: -77.0786991, 39.0637162
[2178/2406] Index 15000: -77.1443789, 39.0866386
[2179/2406] Index 15010: -77.1056768, 39.0466561
[2180/2406] Index 15021: -77.0847987, 39.05231630000001
[2181/2406] Index 15038: -76.9403895, 39.0870183
[2182/2406] Index 15039: -77.03945949999999, 39.1513822
[2183/2406] Index 15041: -77.0797542, 39.0876618
```

```
[2184/2406] Index 15056: -77.0548489, 39.0624821
[2185/2406] Index 15069: -77.1396781, 39.1000257
[2186/2406] Index 15071: -77.0127272, 38.9911529
[2187/2406] Index 15083: -77.0304429, 38.9947772
[2188/2406] Index 15085: -77.1197521, 38.9516143
[2189/2406] Index 15089: -77.0509596, 39.0527092
[2190/2406] Index 15112: -77.0612835, 39.0799494
[2191/2406] Index 15143: -77.092275, 38.977352
[2192/2406] Index 15149: -77.1912847, 39.1516673
[2193/2406] Index 15151: -77.092275, 38.977352
[2194/2406] Index 15159: -77.0127272, 38.9911529
[2195/2406] Index 15163: -76.99026599999999, 39.0445302
[2196/2406] Index 15165: -77.196947, 39.11168199999999
[2197/2406] Index 15168: -77.0991291, 38.9774035
[2198/2406] Index 15170: -77.26832379999999, 39.1805994
[2199/2406] Index 15182: -77.190445, 39.1499694
[2200/2406] Index 15183: -77.0218648, 38.9971837
[2201/2406] Index 15187: -77.0531, 39.061364
[2202/2406] Index 15189: -77.1948953, 39.14448
[2203/2406] Index 15196: -77.142997, 39.1233359
[2204/2406] Index 15201: -77.40725599999999, 39.1385474
[2205/2406] Index 15206: -77.196947, 39.11168199999999
[2206/2406] Index 15216: -77.090615, 39.0130005
[2207/2406] Index 15217: -77.0603656, 39.0616197
[2208/2406] Index 15218: -76.99151, 39.0437458
[2209/2406] Index 15219: -77.2297057, 39.0854362
[2210/2406] Index 15220: -77.1442376, 39.1493474
[2211/2406] Index 15222: -76.9748036, 39.0201898
[2212/2406] Index 15232: -77.0218648, 38.9971837
[2213/2406] Index 15233: -77.2604454, 39.1868928
[2214/2406] Index 15244: -77.1862852, 39.188433
[2215/2406] Index 15245: -77.2742373, 39.1467753
[2216/2406] Index 15255: -76.9420805, 39.0858282
[2217/2406] Index 15258: -77.088124, 39.1277601
[2218/2406] Index 15260: -77.21297520000002, 39.2747713
[2219/2406] Index 15263: -77.4090434, 39.14368750000001
[2220/2406] Index 15278: -77.071418, 39.0421745
[2221/2406] Index 15301: -77.0218648, 38.9971837
[2222/2406] Index 15303: -77.1135887, 39.0982621
[2223/2406] Index 15306: -76.9420805, 39.0858282
[2224/2406] Index 15310: -76.9414044, 39.0842899
[2225/2406] Index 15321: -77.08217719999999, 39.0542509
[2226/2406] Index 15322: -77.2875245, 39.2301977
[2227/2406] Index 15348: -77.0503559, 39.058254
[2228/2406] Index 15366: -77.0500902, 38.9953312
[2229/2406] Index 15368: -77.0454405, 39.0219839
[2230/2406] Index 15378: -77.0052675, 39.0315443
[2231/2406] Index 15390: -77.0513935, 39.0923639
[2232/2406] Index 15395: -77.1252174, 39.0988934
[2233/2406] Index 15396: -77.110891, 39.048689
[2234/2406] Index 15399: -77.2604454, 39.1868928
```

```
[2235/2406] Index 15401: -77.1394384, 39.0792326
[2236/2406] Index 15412: -77.0012218, 39.0723902
[2237/2406] Index 15414: -76.9327084, 39.1172429
[2238/2406] Index 15433: -77.2724365, 39.1755688
[2239/2406] Index 15440: -77.1218317, 39.0848457
[2240/2406] Index 15452: -76.9374441, 39.0985715
[2241/2406] Index 15459: -77.20211619999999, 39.2873064
[2242/2406] Index 15484: -77.198882, 39.28654720000001
[2243/2406] Index 15503: -76.99026599999999, 39.0445302
[2244/2406] Index 15504: -77.2521841, 39.1270814
[2245/2406] Index 15507: -76.96803369999999, 39.0565845
[2246/2406] Index 15516: -77.2355735, 39.1209347
[2247/2406] Index 15521: -77.0081921, 39.0123939
[2248/2406] Index 15523: -77.0321359, 38.9974075
[2249/2406] Index 15538: -77.0509596, 39.0527092
[2250/2406] Index 15559: -77.0726375, 39.0782147
[2251/2406] Index 15588: -77.096732, 38.9911745
[2252/2406] Index 15590: -77.1072085, 39.0134915
[2253/2406] Index 15594: -77.1254044, 39.0785579
[2254/2406] Index 15603: -77.0920184, 39.0515042
[2255/2406] Index 15614: -77.2144649, 39.0321742
[2256/2406] Index 15616: -76.9472612, 39.0827914
[2257/2406] Index 15632: -77.11343409999999, 39.057984
[2258/2406] Index 15663: -77.12170189999999, 39.0066596
[2259/2406] Index 15664: -77.26498169999999, 39.2134459
[2260/2406] Index 15674: -77.0635149, 39.0799114
[2261/2406] Index 15682: -76.98073830000001, 39.09080549999999
[2262/2406] Index 15687: -77.0601944, 39.0498962
[2263/2406] Index 15701: -76.96803369999999, 39.0565845
[2264/2406] Index 15718: -77.11343409999999, 39.057984
[2265/2406] Index 15733: -76.9414044, 39.0842899
[2266/2406] Index 15745: -77.26498169999999, 39.2134459
[2267/2406] Index 15762: -77.0409949, 39.0410919
[2268/2406] Index 15763: -76.9962711, 39.01320339999999
[2269/2406] Index 15764: -77.0745917, 39.0796341
[2270/2406] Index 15767: -77.04869280000001, 39.0390935
[2271/2406] Index 15775: -76.9472612, 39.0827914
[2272/2406] Index 15787: -77.0687238, 39.1439038
[2273/2406] Index 15795: -77.26832379999999, 39.1805994
[2274/2406] Index 15801: -77.20211619999999, 39.2873064
[2275/2406] Index 15804: -77.0218648, 38.9971837
[2276/2406] Index 15805: -77.0991291, 38.9774035
[2277/2406] Index 15807: -76.9898226, 38.9991051
[2278/2406] Index 15821: -77.27403129999999, 39.1537142
[2279/2406] Index 15822: -77.1150734, 39.0564729
[2280/2406] Index 15824: -77.096732, 38.9911745
[2281/2406] Index 15828: -77.18733999999999, 39.335184
[2282/2406] Index 15845: -77.0320065, 38.9960781
[2283/2406] Index 15846: -77.27403129999999, 39.1537142
[2284/2406] Index 15853: -77.0493637, 39.0116156
[2285/2406] Index 15861: -77.0991291, 38.9774035
```

```
[2286/2406] Index 15864: -77.0530059, 39.0570328
[2287/2406] Index 15866: -76.9241509, 39.1013841
[2288/2406] Index 15869: -77.1884153, 39.1831781
[2289/2406] Index 15874: -77.0503256, 39.0218572
[2290/2406] Index 15877: -77.0619011, 39.0507752
[2291/2406] Index 15886: -76.98854089999999, 39.0726288
[2292/2406] Index 15889: -77.0772272, 38.9862812
[2293/2406] Index 15896: -77.1360301, 39.0870662
[2294/2406] Index 15912: -76.9898226, 38.9991051
[2295/2406] Index 15920: -77.3338783, 39.0741178
[2296/2406] Index 15927: -77.0503559, 39.058254
[2297/2406] Index 15928: -76.9677966, 39.0540986
[2298/2406] Index 15936: -77.02966769999999, 38.9869576
[2299/2406] Index 15942: -77.21125669999999, 39.0926856
[2300/2406] Index 15943: -77.03456840000001, 39.0261093
[2301/2406] Index 15950: -77.27174339999999, 39.201727
[2302/2406] Index 15956: -76.92784859999999, 39.101697
[2303/2406] Index 15969: -77.0672783, 39.1538273
[2304/2406] Index 15974: -77.15143510000001, 39.2109965
[2305/2406] Index 15982: -77.096732, 38.9911745
[2306/2406] Index 15986: -77.0530059, 39.0570328
[2307/2406] Index 15998: -77.1056768, 39.0466561
[2308/2406] Index 16007: -77.1588119, 39.08094
[2309/2406] Index 16023: -77.0467737, 39.1410438
[2310/2406] Index 16024: -76.9403895, 39.0870183
[2311/2406] Index 16026: -77.22093939999999, 39.3319037
[2312/2406] Index 16045: -77.2956311, 39.1603112
[2313/2406] Index 16048: -77.06584, 39.07928
[2314/2406] Index 16050: -77.096732, 38.9911745
[2315/2406] Index 16051: -77.19269779999999, 39.2505344
[2316/2406] Index 16054: -77.0751274, 39.0788977
[2317/2406] Index 16057: -77.08727189999999, 38.9637719
[2318/2406] Index 16060: -77.1389348, 39.0897316
[2319/2406] Index 16071: -77.0530059, 39.0570328
[2320/2406] Index 16074: -77.0374032, 39.00063069999999
[2321/2406] Index 16083: -77.0516921, 39.0506864
[2322/2406] Index 16084: -77.1016389, 39.0003388
[2323/2406] Index 16091: -77.037466, 38.9958963
[2324/2406] Index 16097: -77.0142071, 38.9859888
[2325/2406] Index 16104: -77.17495989999999, 39.1529136
[2326/2406] Index 16125: -76.9677966, 39.0540986
[2327/2406] Index 16133: -77.0920184, 39.0515042
[2328/2406] Index 16135: -77.0503559, 39.058254
[2329/2406] Index 16136: -77.0495275, 39.0350568
[2330/2406] Index 16146: -76.9320314, 39.0795989
[2331/2406] Index 16151: -77.1252737, 39.0006757
[2332/2406] Index 16154: -77.0503559, 39.058254
[2333/2406] Index 16163: -77.1681199, 39.046466
[2334/2406] Index 16173: -77.2138061, 39.0119435
[2335/2406] Index 16179: -76.9206124, 39.09312790000001
[2336/2406] Index 16190: -77.02456269999999, 38.9949258
```

```
[2337/2406] Index 16195: -77.02966769999999, 38.9869576
[2338/2406] Index 16196: -77.0513359, 39.0387866
[2339/2406] Index 16208: -77.1084276, 39.0118564
[2340/2406] Index 16213: -77.05184179999999, 39.0406395
[2341/2406] Index 16218: -77.13091659999999, 39.0814459
[2342/2406] Index 16232: -77.0919238, 38.9817251
[2343/2406] Index 16236: -76.9320314, 39.0795989
[2344/2406] Index 16239: -77.0594225, 39.0805058
[2345/2406] Index 16246: -77.0779125, 39.0836874
[2346/2406] Index 16247: -77.1862852, 39.188433
[2347/2406] Index 16252: -77.1056768, 39.0466561
[2348/2406] Index 16255: -77.1900252, 39.0932995
[2349/2406] Index 16267: -77.096732, 38.9911745
[2350/2406] Index 16271: -77.1403486, 39.1014718
[2351/2406] Index 16272: -77.0818536, 39.0579318
[2352/2406] Index 16285: -77.0531, 39.061364
[2353/2406] Index 16290: -77.0631247, 39.0549676
[2354/2406] Index 16295: -76.9494957, 39.1111622
[2355/2406] Index 16296: -77.0335862, 38.9967597
[2356/2406] Index 16304: -77.0482349, 39.0437129
[2357/2406] Index 16318: -77.0482349, 39.0437129
[2358/2406] Index 16327: -77.06021400000002, 39.10024
[2359/2406] Index 16337: -77.0218648, 38.9971837
[2360/2406] Index 16340: -77.0991291, 38.9774035
[2361/2406] Index 16342: -76.94912599999999, 39.0714207
[2362/2406] Index 16348: -77.2604454, 39.1868928
[2363/2406] Index 16367: -77.4090434, 39.14368750000001
[2364/2406] Index 16387: -77.091251, 38.9632933
[2365/2406] Index 16400: -77.0499111, 39.0439884
[2366/2406] Index 16406: -77.1424876, 39.1408809
[2367/2406] Index 16408: -77.0556758, 39.0234368
[2368/2406] Index 16412: -77.4001969, 39.1389863
[2369/2406] Index 16417: -77.0320065, 38.9960781
[2370/2406] Index 16419: -77.2707539, 39.1545466
[2371/2406] Index 16438: -77.28206899999999, 39.146942
[2372/2406] Index 16446: -77.0580815, 39.0717931
[2373/2406] Index 16462: -77.0513359, 39.0387866
[2374/2406] Index 16464: -77.096732, 38.9911745
[2375/2406] Index 16466: -77.1389348, 39.0897316
[2376/2406] Index 16467: -77.096732, 38.9911745
[2377/2406] Index 16483: -77.0499111, 39.0439884
[2378/2406] Index 16498: -77.0398813, 39.1712015
[2379/2406] Index 16515: -77.096732, 38.9911745
[2380/2406] Index 16525: -77.0493637, 39.0116156
[2381/2406] Index 16531: -76.9403895, 39.0870183
[2382/2406] Index 16537: -77.1240349, 38.9881728
[2383/2406] Index 16546: -77.091251, 38.9632933
[2384/2406] Index 16556: -77.0503628, 39.054544
[2385/2406] Index 16563: -77.0470292, 39.0323997
[2386/2406] Index 16567: -77.1150734, 39.0564729
[2387/2406] Index 16583: -77.08551, 39.0260975
```

```
[2388/2406] Index 16584: -77.2310849, 39.134141
[2389/2406] Index 16586: -77.0158683, 39.0098863
[2390/2406] Index 16590: -77.0513359, 39.0387866
[2391/2406] Index 16591: -77.1167706, 39.0504708
[2392/2406] Index 16612: -77.1234697, 39.0125086
[2393/2406] Index 16613: -77.1048279, 38.9622969
[2394/2406] Index 16627: -76.95688659999999, 39.0493731
[2395/2406] Index 16646: -77.0503559, 39.058254
[2396/2406] Index 16669: -76.9399071, 39.0900694
[2397/2406] Index 16670: -77.1301427, 39.0059686
[2398/2406] Index 16674: -77.1772737, 39.1733025
[2399/2406] Index 16677: -77.16569779999999, 38.99494929999999
[2400/2406] Index 16683: -77.0693411, 39.0659336
[2401/2406] Index 16686: -77.2049452, 39.2920299
[2402/2406] Index 16689: -77.137371, 39.0917302
[2403/2406] Index 16701: -77.2414349, 39.20007440000001
[2404/2406] Index 16707: -77.20432819999999, 39.0466638
[2405/2406] Index 16742: -77.0513935, 39.0923639
[2406/2406] Index 16781: -77.22087479999999, 39.109296
Saved
```

Removing Unwanted Columns and Rows

After using Google API to fill in missing longitude and latitude values, I will remove the address-related columns, including `address`, `city`, `state`, `zip`, `pra` and `geolocation`, since the `longitude` and `latitude` columns are enough for spatial analysis and visualization. Also, I notice there are null values in the `longitude` and `latitude` columns after geocoding. I will take a look at the value first. After identifying below that this data point has an invalid address and is outside the county, I will remove this row from the dataset to ensure data integrity. Also, there are several columns that are not useful for my EDA analysis or modeling, such as `incident_id`. I will remove these unique identifier columns to focus on relevant information.

```
import pandas as pd

# Load the updated police incident dataset with geocoded longitude and latitude
police_incident = pd.read_csv("../../data/processed-data/police_incident_longitude&latitu

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
incident_id	0	0	object
start_time	0	0	object

	null_count	zero_count	dtype
end_time	0	0	object
priority	0	1682	int64
initial_type	0	0	object
close_type	0	0	object
address	15	0	object
city	1	0	object
state	1	0	object
zip	1	0	float64
longitude	1	0	float64
latitude	1	0	float64
police_district_number	0	0	object
sector	1	0	object
pra	1	0	float64
calltime_callroute	0	442	int64
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
disposition_desc	0	0	object
geolocation	0	0	object
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64

```
# Check the row with missing longitude and latitude
police_incident[police_incident['longitude'].isna()]
```

	incident_id	start_time	end_time	priority	initial_type	close_type	address	city	state	zip	...	pra	calltime_c
12153	P2500299963	2025-10-09 16:21:04	2025-10-09 16:53:28	4	OC - OUT OF COUNTY	OC - OUT OF COUNTY	NaN	NaN	NaN	NaN	...	NaN	25

1 rows × 24 columns

```
# Remove rows with missing longitude and latitude and unwanted columns related to address
police_incident = police_incident.dropna(subset=['longitude', 'latitude'])
police_incident = police_incident.drop(columns=['address', 'city', 'state', 'zip', 'geolo
```

```
# Remove unique identifier columns & unwanted columns
police_incident = police_incident.drop(columns=['disposition_desc', 'incident_id'])

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
start_time	0	0	object
end_time	0	0	object
priority	0	1682	int64
initial_type	0	0	object
close_type	0	0	object
longitude	0	0	float64
latitude	0	0	float64
police_district_number	0	0	object
sector	0	0	object
calltime_callroute	0	442	int64
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64

To reduce data size and improve processing efficiency, I will review the initial_type and close_type columns. It turns out that only 447 cases have different initial and close types out of all the records. I've decided to drop the `initial_type` column since `close_type` is the "ground truth" after investigation. This column will be enough for the EDA and modeling. I will rename `close_type` to `incident_type` for clarity.

```
# Compare initial vs close types
different = police_incident[police_incident['initial_type'] != police_incident['close_typ
print(f"Total incidents: {len(police_incident)}")
print(f"Difference: {len(different)} ({len(different)/len(police_incident)*100:.2f}%)")

# Remove rename close_type to incident_type
police_incident = police_incident.rename(columns={'close_type': 'incident_type'})
police_incident = police_incident.drop(columns=['initial_type'])
```

```
Total incidents: 16781
Difference: 447 (2.66%)
```

Modify Data for Better Intepretation

To enhance the interpretability of my dataset, I will modify several columns, including priority and incident types. Originally, the `priority` column ranges from 0 to 4, with 0 being the highest priority. To make it more intuitive, I will modify this scale so that 1 is the lowest priority and 5 is the highest priority. Also, for the incident types columns, I will group around 180 columns into several broader categories based on incident type, which will be better for exploratory data analysis and modeling. The new categories include 'Violent Crime', 'Property Crime', 'Public Order', 'Health & Safety', 'Investigation', 'Assistance & Administrative', and 'Other'. I will also convert all numeric columns from a unit of seconds to unit of minutes.

Modify Scale of Priority

```
police_incident['priority'] = 5 - police_incident['priority']

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
start_time	0	0	object
end_time	0	0	object
priority	0	0	int64
incident_type	0	0	object
longitude	0	0	float64
latitude	0	0	float64
police_district_number	0	0	object
sector	0	0	object
calltime_callroute	0	442	int64
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64

Group Incident Types into Broad Categories

```
# Check catagories in initial_type and close_type
pd.set_option('display.max_rows', None)
print(sorted(police_incident['incident_type'].unique()))
```

```
['ABDUCTION (KIDNAPPING) – CUSTODIAL ABDUCTION, HOSTAGE SITUAT', 'ABDUCTION (KIDNAPPING)
– CUSTODIAL ABDUCTION, HOSTAGE SITUAT – OCCURRED EARLIER', 'ABUSE, ABANDONMENT, NEGLECT',
'ABUSE, ABANDONMENT, NEGLECT – OCCURRED EARLIER', 'ABUSEOS', 'ADMINISTRATIVE (DOCUMENT,
LOST OR FOUND PROP, MESSAGES – OCCURRED EARLIER', 'ADMINISTRATIVE (DOCUMENT, LOST OR
FOUND PROPERTY, MESSAGES,', 'ADMINISTRATIVE (DOCUMENT, LOST OR FOUND PROPERTY, MESSAGES,
– OCCURRED EARLIER', 'ALARMBB – BANK BURGLARY/INTRUSION', 'ALARMBH – BANK
HOLDUP/DURESS/PANIC', 'ALARMRB – ALARM COMMERCIAL BURGLARY/INTRUSION', 'ALARMRB –
COMMERCIAL HOLDUP/DURESS/PANIC', 'ALARMRB – RESIDENTIAL BURGLARY/INTRUSION', 'ALARMRB –
RESIDENTIAL HOLDUP/DURESS/PANIC', 'ALARMU – ALARM OTHER/UNKNOWN', 'ALARMV – ALARM
VEHICLE', 'ANI – ANIMAL COMPL ON PATROL', 'ANIMAL ABUSE', 'ANIMAL ABUSE – OCCURRED
EARLIER', 'ANIMAL COMPL', 'ANIMAL MISC', 'ANIMAL RESCUE', 'ANIMAL VICIOUS', 'ANIMAL
VICIOUS – OCCURRED EARLIER', 'ANIMALVJ – ANIMAL VICIOUS JUST OCCURRED', 'ASLTA',
'ASLTJA', 'ASLTOS', 'ASSAULT', 'ASSAULT – OCCURRED EARLIER', 'ASSAULT JUST OCCURRED –
ROUTINE', 'ASSAULT–TRS – TELEPHONE REPORTING UNIT', 'ASSIST OTHER AGENCY',
'ASSIST/STANDBY', 'BOMB DEVICE FOUND, SUSP PACKAGE, CONTAMINATION', 'BOMB THREAT, CBRN,
CONTAMINATION', 'BOX ALARM – VIA FRS', 'BURGLARY', 'BURGLARY – OCCURRED EARLIER',
'BURGLARY JUST OCCURRED', 'BURGOS', 'CAR JACKING', 'CDS', 'CDS – OCCURRED EARLIER',
'CHECK WELFARE', 'CRYWOLF INTERFACE INCIDENT TYPE', 'DECEASED PERSON', 'DISPJ', 'DISPOS',
'DISTURBANCE/NUISANCE', 'DISTURBANCE/NUISANCE – OCCURRED EARLIER', 'DOMDOS', 'DOMESTIC
DISPUTE', 'DOMESTIC DISTURBANCE/VIOLENCE', 'DOMESTIC DISTURBANCE/VIOLENCE – OCCURRED
EARLIER', 'DOMESTIC DISTURBANCE/VIOLENCE OCCURRED EARLIER', 'DOMESTIC VIOLENCE', 'DOMVA',
'DOMVJA', 'DOMVOS', 'DRIVING UNDER THE INFLUENCE', 'DT – DETAIL', 'E911 DISCONNECT',
'EVALUATION BY MCOT', 'FOLLOW UP/SUPPLEMENTAL INFORMATION', 'FOLLOW UP/SUPPLEMENTAL
INFORMATION – OCCURRED EARLIER', 'FOLLOWT–TRS / SUPPLEMENTAL INFORMATION – TELEPHONE
REPORTING UNIT', 'FRAUD/DECEPTION', 'FRAUD/DECEPTION – OCCURRED EARLIER', 'FRAUDJ',
'FRAUDT–TRS FRAUD / DECEPTION – TELEPHONE REPORTING UNIT', 'HARASSMENT, STALKING,
THREATS', 'HARASSMENT, STALKING, THREATS – OCCURRED EARLIER', 'HARASSMENT, STLAKING,
THREATS', 'HARASSOS', 'HARASST–TRS HARASSMENT, STALKING, THREATS – TELEPHONE REPORTING
UNIT', 'HAZARDOUS MATERIAL – VIA FRS', 'HUNT – HUNTING – ILLEGAL', 'INDECENCY/LEWDNESS',
'INDECENCY/LEWDNESS – OCCURRED EARLIER', 'INV – POLICE INVESTIGATION', 'LOCK OUT/IN',
'LOSTT–TRS ADMIN (DOCUMENT, LOST OR FOUND PROP, MESSAGES – TELEPHONE REPORTING UNIT',
'MENTAL DISORDER', 'MENTAL DISORDER – VIA FRS', 'MIS – MISC ON PATROL', 'MISC–ADMIN
(DOCUMENT, LOST OR FOUND PROPERTY, MESSAGES,', 'MISSARDD', 'MISSING, RUNAWAY, FOUND
PERSON', 'NEGLECTOS', 'NOISE – NOISE – OTHER COMPLAINTS', 'NON–PRIORITY RESPONSE
TRANSPORT', 'ORDNANCE – FOUND UNEXPLODED', 'OVERDOSE – VIA FRS', 'PARKING OFFENSE',
'PEDESTRIAN STRUCK', 'PEDESTRIAN STRUCK – OCCURRED EARLIER', 'PRIORITY RESPONSE
TRANSPORT', 'PROSTITUTION', 'RAPE', 'RAPEO – OCCURRED EARLIER', 'RAPEOS', 'RESCUE WITH
FRS', 'ROBBERY', 'ROBBERY – OCCURRED EARLIER', 'ROBBERY JUST OCCURRED', 'ROBOS', 'SEX
ASSAULT', 'SEXASLTJA', 'SEXASLTOS', 'SEXUAL ASSAULT', 'SEXUAL ASSAULT – OCCURRED
EARLIER', 'SHOOT – SHOOTING', 'SHOOTING', 'SHOOTO – SHOOTING – OCCURRED EARLIER',
'SHOTSOS', 'STAB – STABBING', 'STABBING', 'STABO – STABBING – OCCURRED EARLIER',
'STALKOS', 'STATION RESPONSE', 'STLVEHT – TRS STOLEN VEHICLE – TELEPHONE REPORTING UNIT',
'SUICIDAL PERSON/ATTEMPTED SUICIDE', 'SUICIDAL PERSON/ATTEMPTED SUICIDE – OCCURRED
EARLIER', 'SUSICIOUS CIRCUMSTANCE, PERSON, VEHICLE – OCCURRED EARLIER', 'SUSPICIOUS CIRC,
```

```
PERSONS, VEHICLE', 'TD – TRAFFIC DETAIL', 'THEFT/LARCENY', 'THEFT/LARCENY – HOLDING
SUSPECT', 'THEFT/LARCENY – OCCURRED EARLIER', 'THEFT/LARCENY FROM AUTO', 'THEFT/LARCENY
FROM AUTO – OCCURRED EARLIER', 'THEFTT – TRS THEFT/LARCENY – TELEPHONE REPORTING UNIT',
'THREATOS', 'THREATT – TRS HARASSMENT, STALKING, THREATS – TELEPHONE REPORTING UNIT',
'TRAFFIC ASSIST FOR FRS', 'TRAFFIC VIOLATION', 'TRAFFIC VIOLATION – OCCURRED EARLIER',
'TRAFFIC/TRANSPORTATION  INCIDENT – OCCURRED EARLIER', 'TRAFFIC/TRANSPORTATION INCIDENT',
'TRAFFIC/TRANSPORTATION INCIDENT – OCCURRED EARLIER', 'TRAIN COLLISION – VIA FRS', 'TRE –
TRESP ON PATROL', 'TRESPASSING/UNWANTED', 'TRESPASSING/UNWANTED – OCCURRED EARLIER',
'TRESPJ', 'TRESPOS', 'URGENT ASSIST', 'VANDALISM, DAMAGE, MISCHIEF', 'VANDALISM, DAMAGE,
MISCHIEF – OCCURRED EARLIER', 'VANDALISM, DAMAGE, MISCHIEF–TRS – TELEPHONE REPORTING
UNIT', 'WANTED PERSON, VEHICLE', 'WEAPJ', 'WEAPONS/FIREARMS', 'WEAPONS/FIREARMS –
OCCURRED EARLIER', 'WEAPOS', 'WEAPSUR', 'WORKING CODE', 'WS – WARRANT SERVICE']
```

To improve efficiency, the following code block us generated by Claude to group 180+ incident types into broader categories.[1]

```
# Group incident types into broader categories (Claude Sonnet4.5 Generated)
def group_type(incident_type):
    incident_type = str(incident_type).upper()

    if any(x in incident_type for x in ['ASSAULT', 'ASLT', 'SHOOT', 'STAB', 'DOMESTIC', '
        return 'Violent Crime'
    elif any(x in incident_type for x in ['THEFT', 'LARCENY', 'BURGLARY', 'BURG', 'ROBBER
        return 'Property Crime'
    elif 'TRAFFIC' in incident_type or 'PEDESTRIAN STRUCK' in incident_type or 'DUI' in i
        return 'Traffic'
    elif any(x in incident_type for x in ['DISTURBANCE', 'NUISANCE', 'NOISE', 'TRESPASS',
        return 'Public Order'
    elif any(x in incident_type for x in ['WELFARE', 'MENTAL', 'SUICID', 'OVERDOSE', 'MIS
        return 'Health & Safety'
    elif any(x in incident_type for x in ['SUSPICIOUS', 'ALARM', 'FRAUD', 'HARASSMENT', '
        return 'Investigation'
    elif any(x in incident_type for x in ['ASSIST', 'ADMINISTRATIVE', 'FOLLOW UP', 'STATI
        return 'Administrative'
    else:
        return 'Other'

police_incident['incident_type'] = police_incident['incident_type'].apply(group_type)
```

```
# Verify the new incident types
print(sorted(police_incident['incident_type'].unique()))
```

```
['Administrative', 'Health & Safety', 'Investigation', 'Other', 'Property Crime', 'Public
Order', 'Traffic', 'Violent Crime']
```

Convert Time Columns from Seconds to Minutes

```
# Convert time-related columns from seconds to minutes
calltime_cols = ['calltime_callroute', 'calltime_dispatch', 'callroute_dispatch', 'callti
```

```
for col in calltime_cols:
    police_incident[col] = police_incident[col] / 60
```

Generate Day of Week and Hour Column

```
# Correct start_time to data type
police_incident['start_time'] = pd.to_datetime(police_incident['start_time'], errors='coe

# Extract hour and weekday from start_time
police_incident['hour'] = police_incident['start_time'].dt.hour

# Extract weekday number
police_incident['weekday'] = police_incident['start_time'].dt.dayofweek + 1
```

```
# Verify the data after all previous modifications
print(police_incident.describe())

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

```
                      start_time      priority     longitude  \
count                      16781  16781.000000  16781.000000
mean   2025-10-16 14:41:22.890054144      3.135749    -77.119698
min              2025-10-01 00:01:37      1.000000    -77.483699
25%              2025-10-08 18:16:50      2.000000    -77.196409
50%              2025-10-16 16:04:51      3.000000    -77.116200
75%              2025-10-24 10:34:18      4.000000    -77.048500
max              2025-10-31 23:59:04      5.000000    -76.905300
std                              NaN      1.264269      0.093557

           latitude  calltime_callroute  calltime_dispatch  \
count  16781.000000        16781.000000       15810.000000
mean      39.083999            2.896116          14.518516
min       38.945130            0.000000           0.133333
25%       39.031930            1.600000           3.633333
50%       39.077568            2.366667           6.183333
75%       39.141840            3.533333          12.933333
max       39.345283          154.366667        2854.433333
std        0.069480            3.043298          32.897465

       callroute_dispatch  calltime_arrive  calltime_cleared  dispatch_arrive  \
count        15810.000000     13531.000000      16778.000000     13199.000000
mean            11.608343        23.708330         56.591461        11.553745
min              0.000000         0.300000          0.333333         0.016667
```

```
25%              1.133333         9.583333        22.037500         3.783333
50%              3.083333        15.766667        38.041667         7.533333
75%              9.300000        26.891667        68.312500        13.566667
max           2852.316667      4433.150000      4549.766667      4410.633333
std             32.706549        50.818737        85.391884        46.686458
```

```
       arrive_cleared          hour       weekday
count    13530.000000  16781.000000  16781.000000
mean        32.711503     13.387343      4.026578
min          0.016667      0.000000      1.000000
25%          6.866667      9.000000      3.000000
50%         15.750000     14.000000      4.000000
75%         35.437500     18.000000      6.000000
max       4541.316667     23.000000      7.000000
std         68.250035      5.929021      1.904611
```

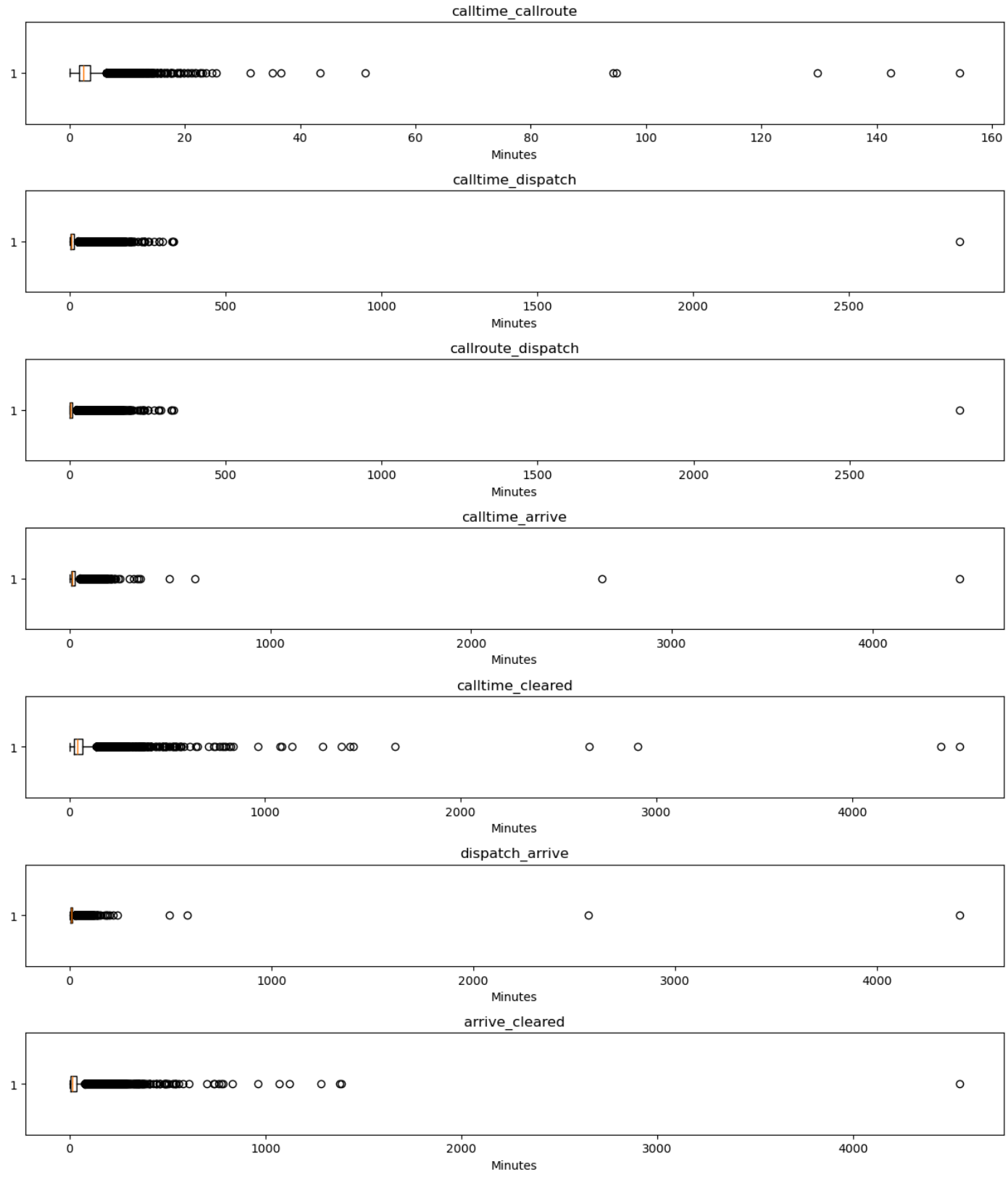
	null_count	zero_count	dtype
start_time	0	0	datetime64[ns]
end_time	0	0	object
priority	0	0	int64
incident_type	0	0	object
longitude	0	0	float64
latitude	0	0	float64
police_district_number	0	0	object
sector	0	0	object
calltime_callroute	0	442	float64
calltime_dispatch	971	0	float64
callroute_dispatch	971	183	float64
calltime_arrive	3250	0	float64
calltime_cleared	3	0	float64
dispatch_arrive	3582	0	float64
arrive_cleared	3251	0	float64
hour	0	465	int32
weekday	0	0	int32

Outlier Detection and Treatment

In this step, I will visualize and check for any extreme outliers that might seems like its an error using IQR fencing. After this step, the data is ready for exploratory data analysis. I will export a cleaned version of the dataset for EDA and proceed with normalization for future data modeling. I will not remove outliers that is not an error, since these

outliers might be important for the analysis. Some police incidents might take longer time to respond due to various reasons.

```
import matplotlib.pyplot as plt

# Visualize outliers in time-related columns using boxplots
fig, axes = plt.subplots(len(calltime_cols), 1, figsize=(12, len(calltime_cols)*2))
for i, col in enumerate(calltime_cols):
    axes[i].boxplot(police_incident[col].dropna(), vert=False)
    axes[i].set_title(col)
    axes[i].set_xlabel('Minutes')
plt.tight_layout()
plt.show()
```




```
# Check for any rows with values of 1000 or more minutes in any time column
pd.set_option('display.max_columns', None)
print(police_incident[(police_incident[calltime_cols] > 1000).any(axis=1)])
```

```
             start_time             end_time  priority    incident_type  \
853   2025-10-28 20:49:25  2025-10-30 17:06:22         1    Investigation
```

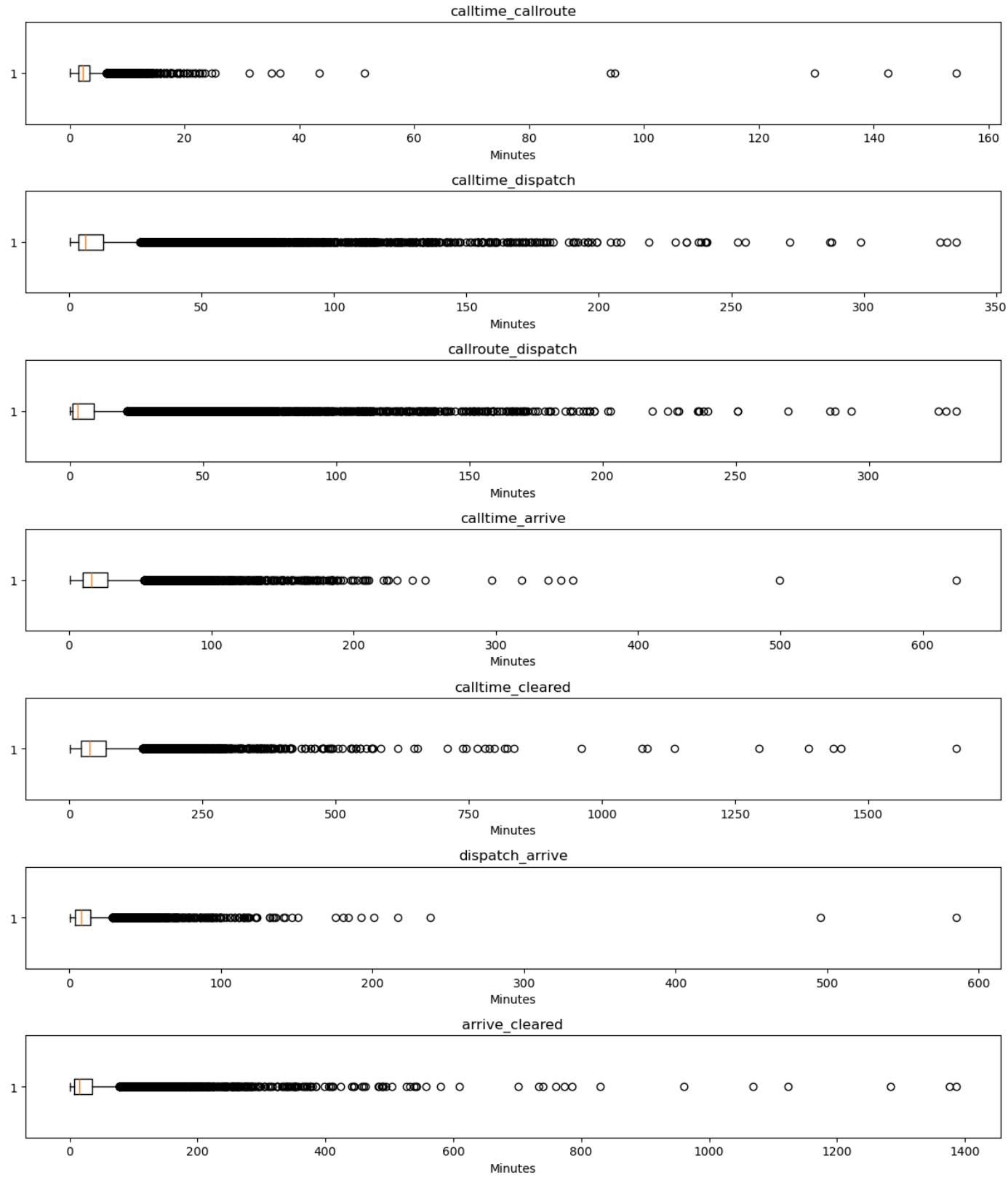
```
1508  2025-10-26 09:31:00  2025-10-29 13:20:46         1   Property Crime
2614  2025-10-26 15:59:53  2025-10-27 10:04:58         5            Other
3067  2025-10-25 12:35:26  2025-10-26 11:44:03         4            Other
4470  2025-10-23 00:02:29  2025-10-23 18:59:20         4    Violent Crime
7331  2025-10-17 22:01:01  2025-10-18 15:57:17         1   Property Crime
9191  2025-10-14 06:04:59  2025-10-15 09:50:49         1   Property Crime
9197  2025-10-14 09:17:10  2025-10-15 09:26:55         4    Violent Crime
10994 2025-10-08 18:20:13  2025-10-11 20:33:34         1   Property Crime
11150 2025-10-10 15:49:03  2025-10-11 15:43:35         1    Investigation
12284 2025-10-07 12:07:11  2025-10-09 12:34:03         3          Traffic
16104 2025-10-01 13:47:35  2025-10-02 11:22:23         4  Health & Safety

       longitude   latitude police_district_number sector  calltime_callroute  \
853   -77.073700  39.046920                     4D     L1            2.000000
1508  -77.152781  39.083999                     1D     A3            1.716667
2614  -77.287525  39.230198                     5D     N3            2.466667
3067  -77.091700  39.040480                     2D     D1            2.300000
4470  -77.068900  39.098150                     4D     K1            1.850000
7331  -77.050285  39.039970                     4D     L2            4.766667
9191  -77.248900  39.229450                     5D     M3            2.733333
9197  -77.247300  39.164670                     5D     N1            9.016667
10994 -77.170900  39.032870                     1D     B1            3.800000
11150 -77.210886  39.095812                     1D     B2            1.316667
12284 -77.094200  38.984660                     2D     D2            2.100000
16104 -77.174960  39.152914                     6D     P3            1.150000

       calltime_dispatch  callroute_dispatch  calltime_arrive  \
853            78.950000           76.933333      2651.500000
1508            2.983333            1.266667         8.433333
2614            3.383333            0.900000        16.650000
3067            8.366667            6.066667        12.500000
4470            2.433333            0.583333        12.933333
7331           14.216667            9.433333              NaN
9191           60.800000           58.066667              NaN
9197           10.283333            1.250000        63.316667
10994          22.500000           18.700000      4433.150000
11150         132.400000          131.083333              NaN
12284        2854.433333         2852.316667              NaN
16104           2.066667            0.916667        11.633333

       calltime_cleared  dispatch_arrive  arrive_cleared  hour  weekday
853         2656.933333      2572.550000        5.416667    20        2
1508        4549.766667         5.433333     4541.316667     9        7
2614        1085.066667        13.250000     1068.416667    15        7
3067        1388.600000         4.116667     1376.100000    12        6
4470        1136.833333        10.483333     1123.900000     0        4
7331        1076.266667              NaN             NaN    22        5
9191        1665.833333              NaN             NaN     6        2
9197        1449.750000        53.033333     1386.416667     9        2
10994       4453.333333      4410.633333       20.183333    18        3
11150       1434.516667              NaN             NaN    15        5
```

```
12284        2906.866667              NaN              NaN     12         2
16104        1294.783333         9.550000      1283.133333     13         3
```

For the indices of `853`, `1508`, `10994`, and `12284`, I will remove them from the dataset since these extreme outliers are likely due to data entry errors. It is unusual for `dispatch_arrive`, `calltime_dispatch`, and `calltime_arrive` to have such high values.

```
# Remove extreme outliers by their indices
police_incident = police_incident.drop([853, 1508, 10994, 12284])
```

```
# Visualize updated box plots
fig, axes = plt.subplots(len(calltime_cols), 1, figsize=(12, len(calltime_cols)*2))
for i, col in enumerate(calltime_cols):
    axes[i].boxplot(police_incident[col].dropna(), vert=False)
    axes[i].set_title(col)
    axes[i].set_xlabel('Minutes')
plt.tight_layout()
plt.show()
```




```
police_incident.describe(include='all')
```

	start_time	end_time	priority	incident_type	longitude	latitude	police_district_number	se
count	16777	16777	16777.000000	16777	16777.000000	16777.000000	16777	16

	start_time	end_time	priority	incident_type	longitude	latitude	police_district_number	se
unique	NaN	16715	NaN	8	NaN	NaN	6	3
top	NaN	2025-10-19 13:06:03	NaN	Property Crime	NaN	NaN	3D	G
freq	NaN	2	NaN	3022	NaN	NaN	3276	1(
mean	2025-10-16 14:40:56.703165184	NaN	3.136139	NaN	-77.119697	39.084010	NaN	N
min	2025-10-01 00:01:37	NaN	1.000000	NaN	-77.483699	38.945130	NaN	N
25%	2025-10-08 18:16:50	NaN	2.000000	NaN	-77.196409	39.031930	NaN	N
50%	2025-10-16 16:04:51	NaN	3.000000	NaN	-77.116200	39.077570	NaN	N
75%	2025-10-24 10:34:00	NaN	4.000000	NaN	-77.048500	39.141840	NaN	N
max	2025-10-31 23:59:04	NaN	5.000000	NaN	-76.905300	39.345283	NaN	N
std	NaN	NaN	1.264097	NaN	0.093566	0.069482	NaN	N

```
police_incident.to_csv("../../data/processed-data/police_incident_eda.csv",index=False)
print("Saved")
```

```
Saved
```

Methods and Codes (ML)

In this section, I will focus on preparing the dataset for machine learning modeling. I will continue from the cleaned dataset used for EDA, and perform steps such as normalization, make sure catagorical data are encoded, perform log transformation and z-score transformation to ensure the data is suitable for later supervised or unsupervised modeling.

Check Skewness and Kurtosis

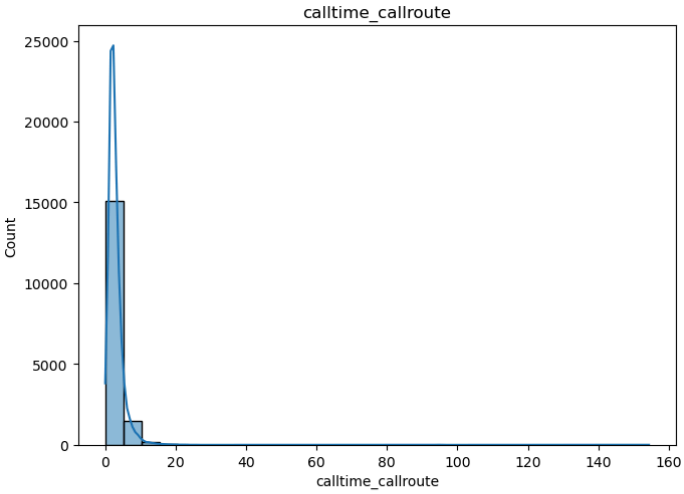
Before normalization, I will first check the skewness and kurtosis of each numeric column to understand their distribution. This helps to identify any columns that may require transformation to meet the assumptions of certain machine learning algorithms.

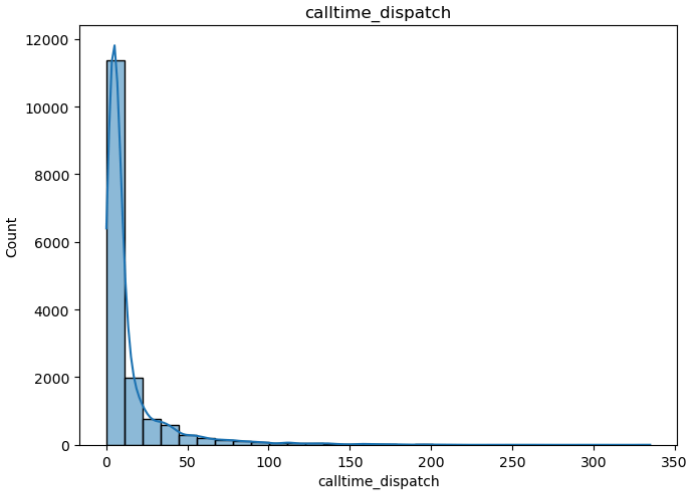
```
import pandas as pd
import numpy as np

# Import the cleaned police incident dataset
police_incident = pd.read_csv("../../data/processed-data/police_incident_eda.csv")
```

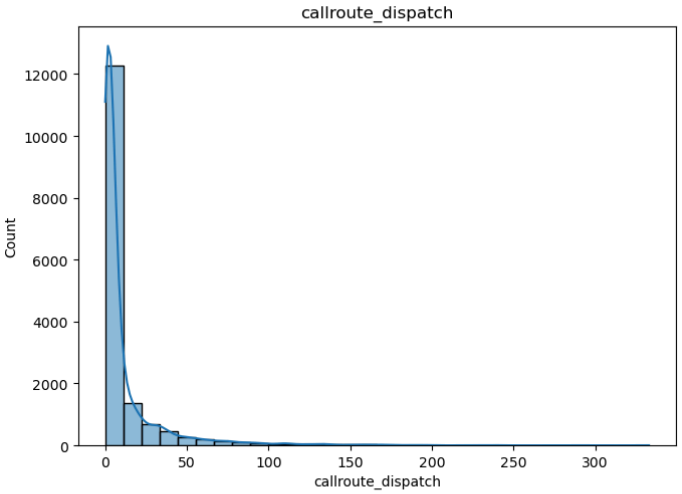
```
# Define time-related columns
cols = ['calltime_callroute','calltime_dispatch','callroute_dispatch','calltime_arrive','

# Check skewness and kurtosis of time-related columns
police_incident[cols].agg(['skew', 'kurt'])
```

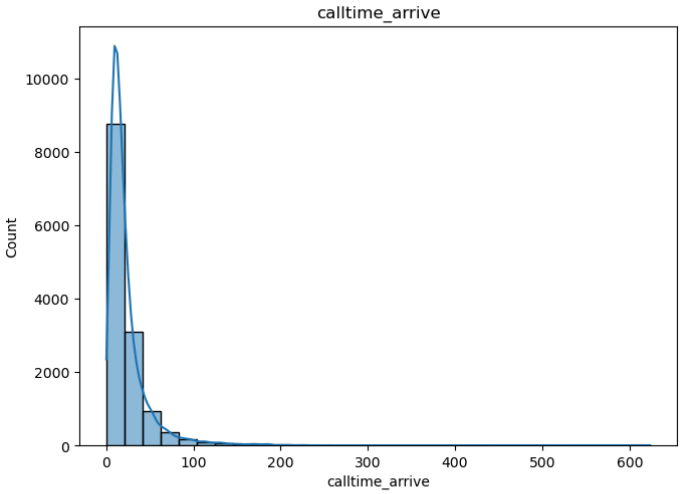
	calltime_callroute	calltime_dispatch	callroute_dispatch	calltime_arrive	calltime_cleared	dispatch_arrive	arrive_
skew	22.104397	4.483037	4.568264	4.865430	6.963982	10.595679	7.8982
kurt	915.915062	28.585118	29.562967	52.672626	104.300570	290.717513	119.03

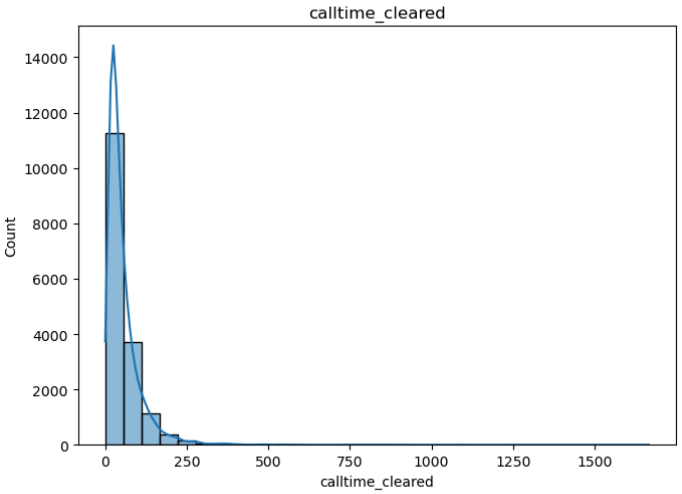
```
import matplotlib.pyplot as plt
import seaborn as sns

# Define figure size for plots
plt.figure(figsize=(14,20))

# Iterate over time-related columns to plot histograms
for i, c in enumerate(cols, 1):
    plt.subplot(4,2,i)
    sns.histplot(police_incident[c], kde=True, bins=30)
    plt.title(f"{c}")
    plt.tight_layout()

plt.show()
```

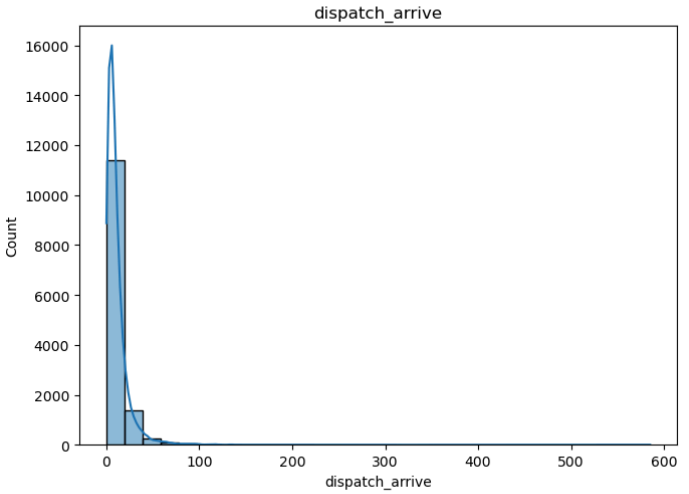

calltime_callroute
Count
calltime_callroute

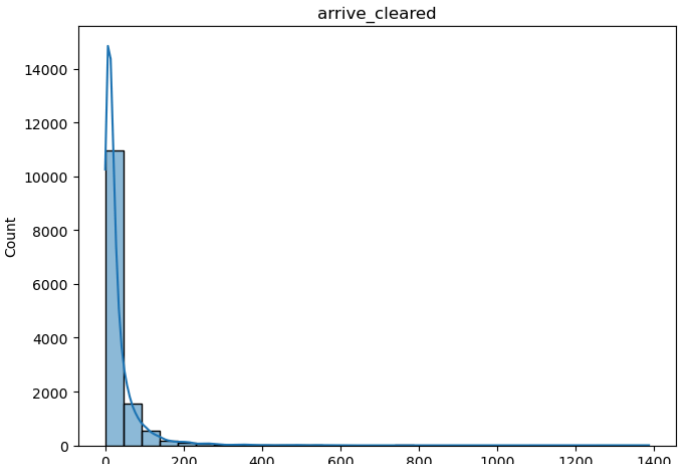

calltime_dispatch
Count
calltime_dispatch


callroute_dispatch
Count
callroute_dispatch


calltime_arrive
Count
calltime_arrive


calltime_cleared
Count
calltime_cleared


dispatch_arrive
Count
dispatch_arrive


arrive_cleared
Count

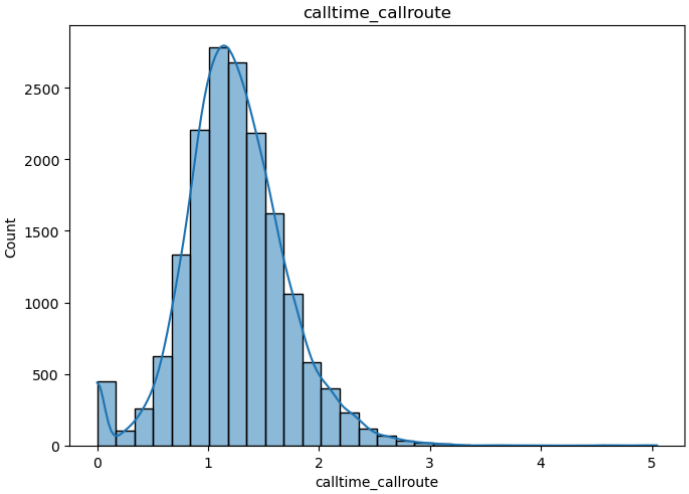
arrive_cleared

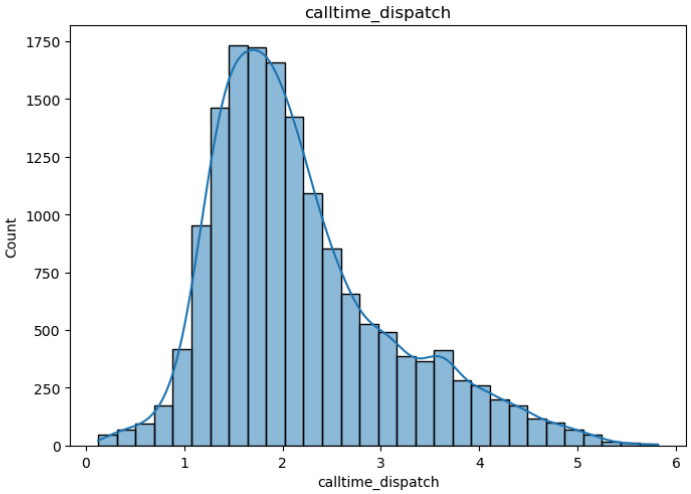
Log Transformation

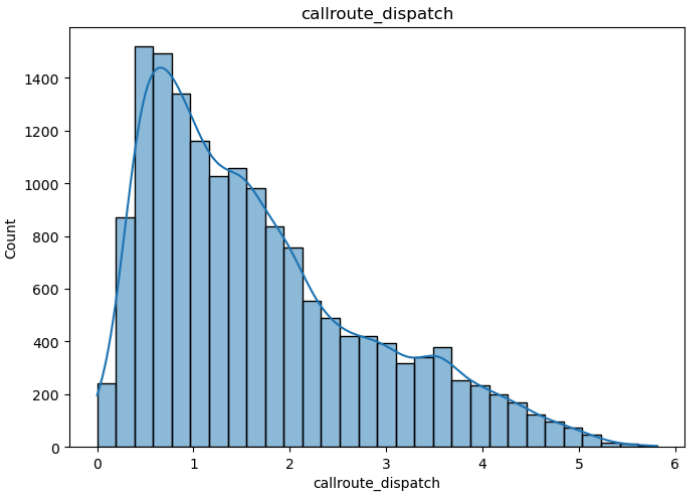
After reviewing the skewness, kurtosis values, and the distributions of each numeric column, I will apply a log transformation to all the numeric columns, because each of the columns shows high skewness. By doing a log transformation, it helps to reduce skewness, stabilize variance, and make the data more normally distributed.

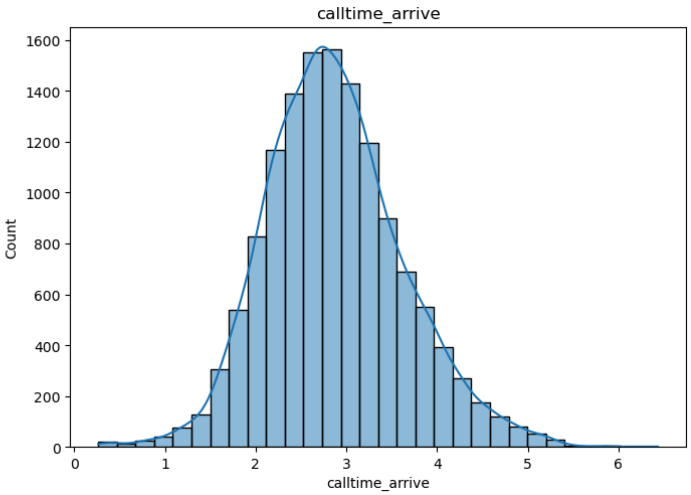
```
# Apply log transformation to time-related columns to reduce skewness
police_incident['calltime_callroute'] = np.log1p(police_incident['calltime_callroute'])
police_incident['calltime_dispatch'] = np.log1p(police_incident['calltime_dispatch'])
police_incident['callroute_dispatch'] = np.log1p(police_incident['callroute_dispatch'])
police_incident['calltime_arrive'] = np.log1p(police_incident['calltime_arrive'])
police_incident['calltime_cleared'] = np.log1p(police_incident['calltime_cleared'])
police_incident['dispatch_arrive'] = np.log1p(police_incident['dispatch_arrive'])
police_incident['arrive_cleared'] = np.log1p(police_incident['arrive_cleared'])
```

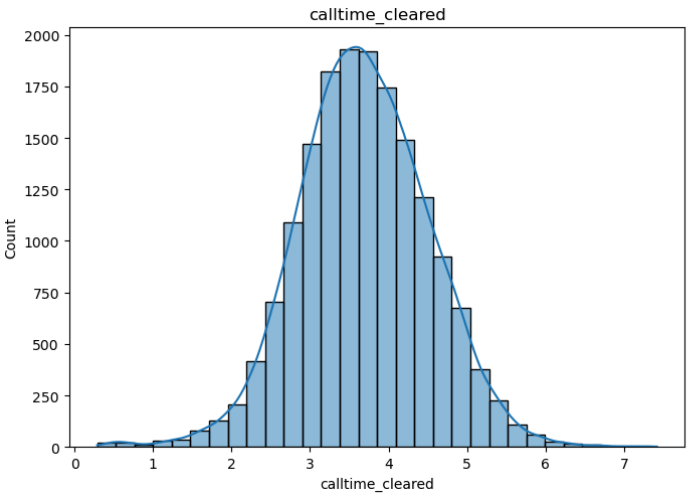
```
import matplotlib.pyplot as plt
import seaborn as sns

# Define figure size for plots
plt.figure(figsize=(14,20))

# Iterate over time-related columns to plot histograms
for i, c in enumerate(cols, 1):
    plt.subplot(4,2,i)
    sns.histplot(police_incident[c], kde=True, bins=30)
    plt.title(f"{c}")
    plt.tight_layout()

plt.show()
```

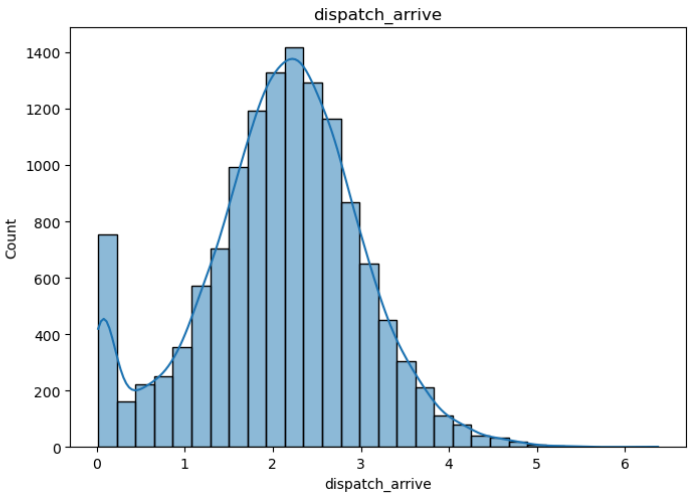

calltime_callroute
Count
calltime_callroute

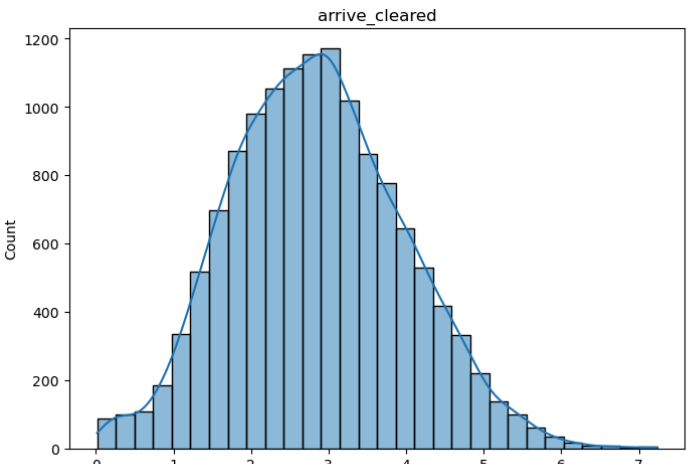

calltime_dispatch
Count
calltime_dispatch


callroute_dispatch
Count
callroute_dispatch


calltime_arrive
Count
calltime_arrive


calltime_cleared
Count
calltime_cleared


dispatch_arrive
Count
dispatch_arrive


arrive_cleared
Count

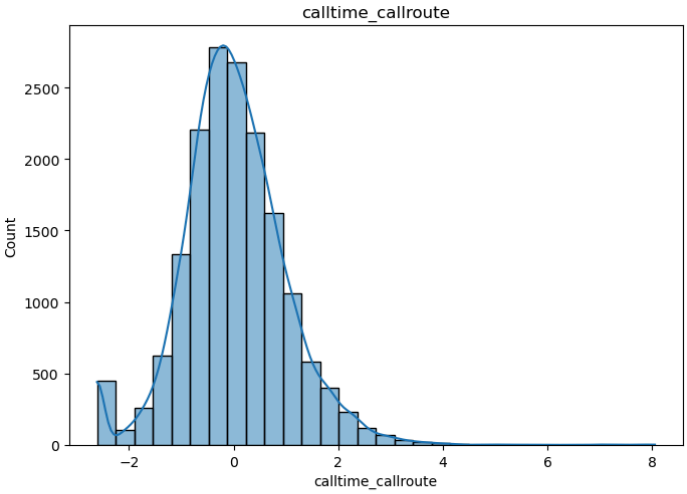
arrive_cleared

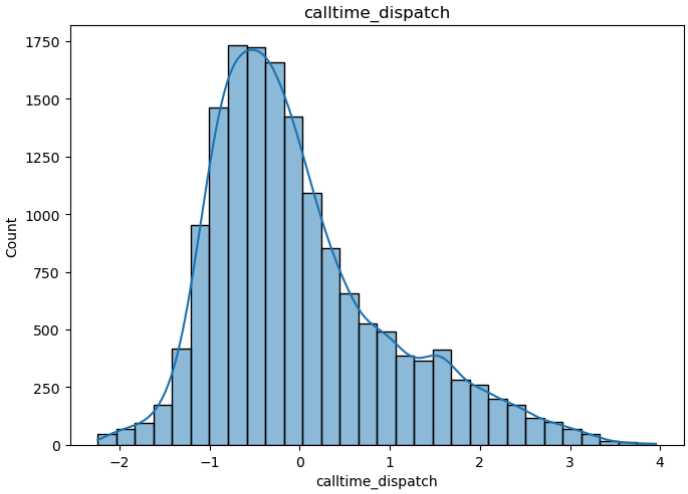
Z-Score Normalization

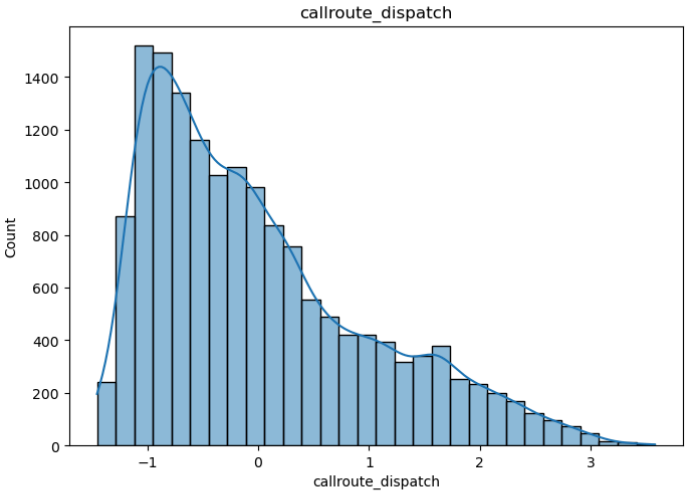
In this step, I will apply z-score normalization to all numeric columns in the dataset. Z-score normalization standardizes the data by subtracting the mean and dividing by the standard deviation, resulting in a distribution with a mean around 0. Z-Score normalization is essential for machine learning algorithms.

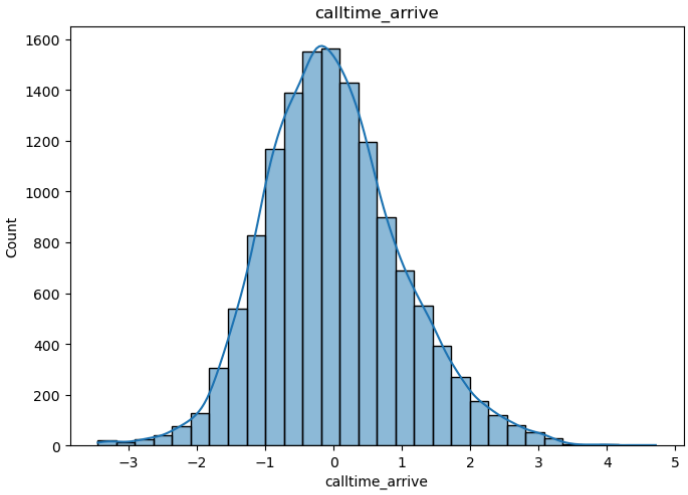
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
police_incident[cols] = pd.DataFrame(scaler.fit_transform(police_incident[cols]), columns
```

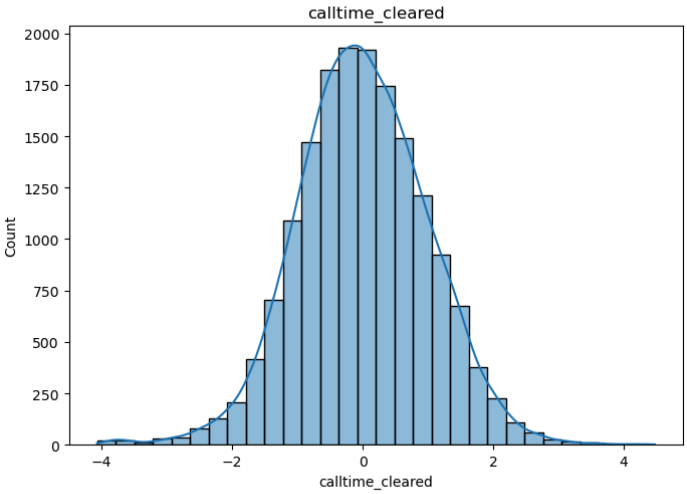
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14,20))

# Iterate over time-related columns to plot histograms
for i, c in enumerate(cols, 1):
    plt.subplot(4,2,i)
    sns.histplot(police_incident[c], kde=True, bins=30)
    plt.title(f"{c}")
    plt.tight_layout()
plt.savefig("../../report/visual4.png", dpi=300, bbox_inches="tight")
plt.show()
```

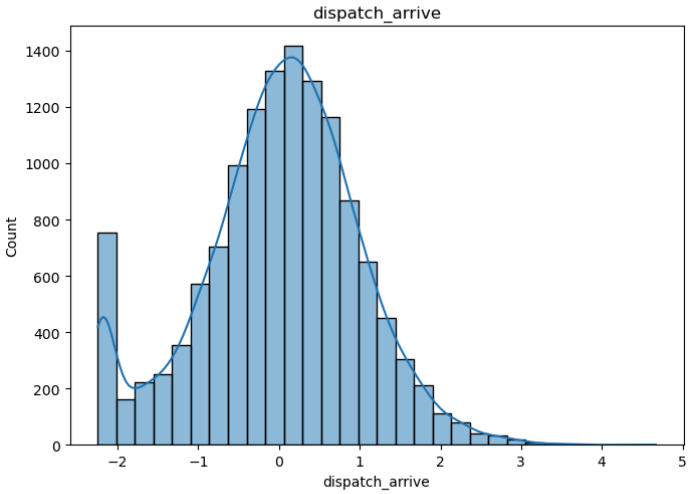

calltime_callroute
Count
calltime_callroute

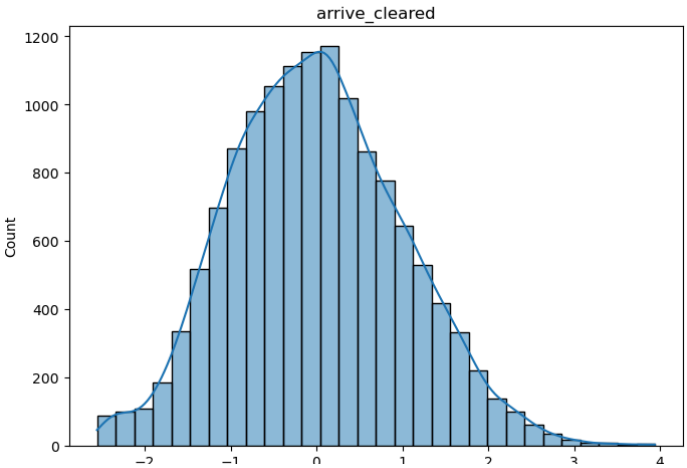

calltime_dispatch
Count
calltime_dispatch


callroute_dispatch
Count
callroute_dispatch


calltime_arrive
Count
calltime_arrive


calltime_cleared
Count
calltime_cleared


dispatch_arrive
Count
dispatch_arrive


arrive_cleared
Count

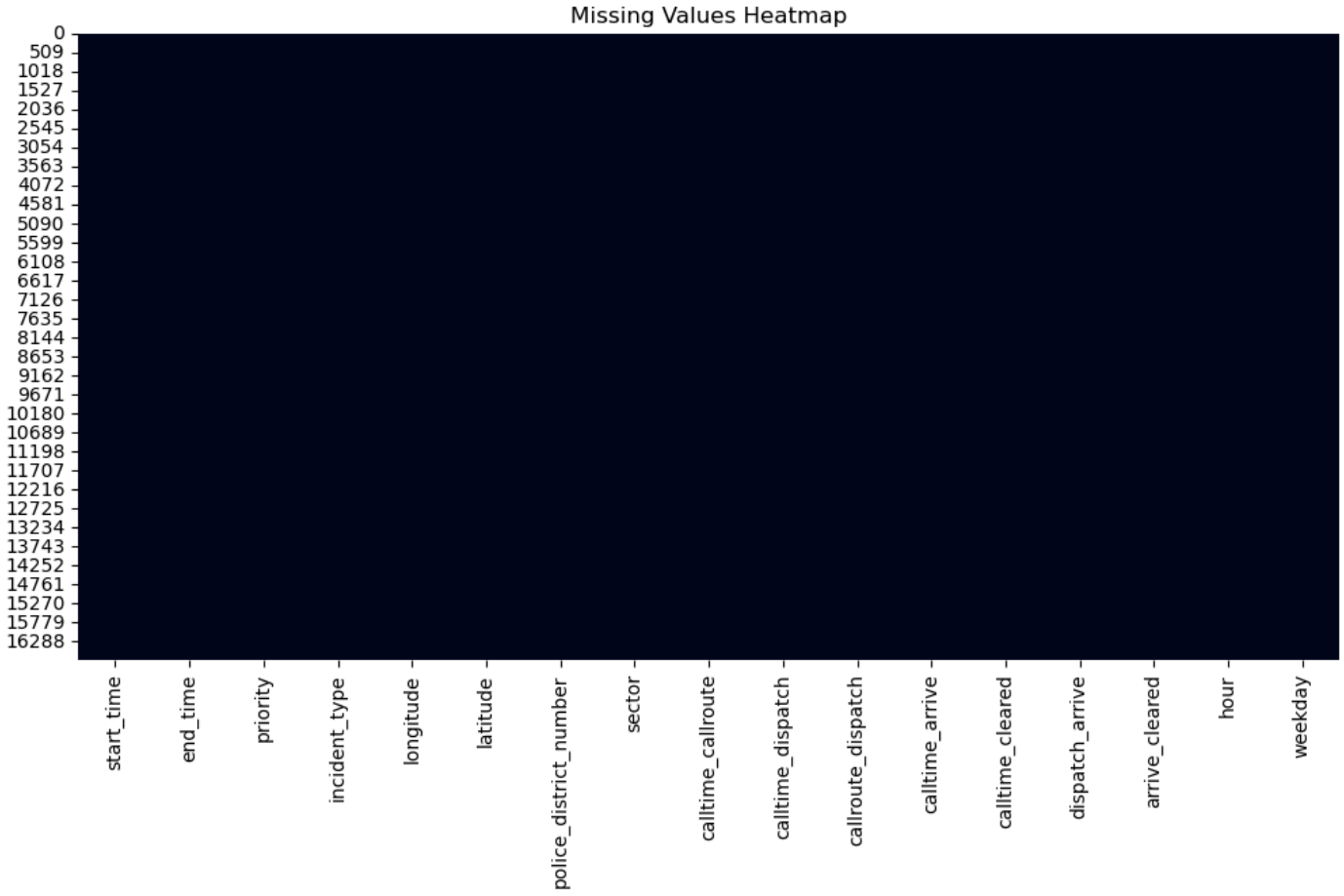
arrive_cleared

Handling Missing Value for Machine Learning

Since some machine learning algorithms cannot handle missing values, I will fill in any remaining missing values using the median of each column, because my columns and skewed even after log transformation. Using the median helps to reduce the impact of outliers and skewness.

```
# Impute missing numeric values with median
numeric_cols = police_incident.select_dtypes(include=['float64','int64']).columns
police_incident[numeric_cols] = police_incident[numeric_cols].fillna(police_incident[nume
```

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


# Visualize missing data using a heatmap
plt.figure(figsize=(12,6))
sns.heatmap(police_incident.isna(), cbar=False)
plt.title("Missing Values Heatmap")
plt.show()
```



```
# Verify no missing values remain
police_incident.isna().sum()
```

```
start_time                0
end_time                  0
priority                  0
incident_type             0
longitude                 0
latitude                  0
police_district_number    0
sector                    0
calltime_callroute        0
calltime_dispatch         0
callroute_dispatch        0
calltime_arrive           0
calltime_cleared          0
dispatch_arrive           0
arrive_cleared            0
hour                      0
weekday                   0
dtype: int64
```

Remove Unnecessary Columns for Machine Learning

Since the following data will be purely used for machine learning modeling and I already generated new columns s `day_of_week` and `hour`, I will remove the `start_time` and `end_time` columns to avoid redundancy. Also, I will remove the sector column since it has too many catagories, which is not suitable for machine learning.

```
police_incident = police_incident.drop(['start_time', 'end_time', 'sector'], axis=1)

# Summary of nulls, zeros, and data types for each column
pd.DataFrame({
    'null_count': police_incident.isnull().sum(),
    'zero_count': (police_incident == 0).sum(),
    'dtype': police_incident.dtypes
})
```

	null_count	zero_count	dtype
priority	0	0	int64
incident_type	0	0	object
longitude	0	0	float64
latitude	0	0	float64
police_district_number	0	0	object
calltime_callroute	0	0	float64

	null_count	zero_count	dtype
calltime_dispatch	0	0	float64
callroute_dispatch	0	0	float64
calltime_arrive	0	0	float64
calltime_cleared	0	0	float64
dispatch_arrive	0	0	float64
arrive_cleared	0	0	float64
hour	0	465	int64
weekday	0	0	int64

Label Encoding for Categorical Columns

In this step, I will apply label encoding to categorical columns `incident_type` and `police_district_number`. Label encoding converts categorical variables into a number that can be provided to machine learning algorithms to improve predictions.

```
from sklearn.preprocessing import LabelEncoder

# Initialize Label Encoder
le = LabelEncoder()

# Define columns to be label encoded
encode = ['incident_type', 'police_district_number']

# Identify categorical columns to encode
for col in encode:
    police_incident[col] = le.fit_transform(police_incident[col])
    print(dict(zip(le.classes_, le.transform(le.classes_))))
```

```
{'Administrative': 0, 'Health & Safety': 1, 'Investigation': 2, 'Other': 3, 'Property
Crime': 4, 'Public Order': 5, 'Traffic': 6, 'Violent Crime': 7}
{'1D': 0, '2D': 1, '3D': 2, '4D': 3, '5D': 4, '6D': 5}
```

```
# Verify the final dataset
print(police_incident.describe(include='all'))

# Save dataset for machine learning
police_incident.to_csv("../../data/processed-data/police_incident_ml.csv",index=False)
print("Saved")
```

```
           priority  incident_type     longitude      latitude  \
count  16777.000000   16777.000000  16777.000000  16777.000000
mean       3.136139       3.638314    -77.119697     39.084010
std        1.264097       2.111452      0.093566      0.069482
```

```
min        1.000000       0.000000    -77.483699     38.945130
25%        2.000000       2.000000    -77.196409     39.031930
50%        3.000000       4.000000    -77.116200     39.077570
75%        4.000000       5.000000    -77.048500     39.141840
max        5.000000       7.000000    -76.905300     39.345283

       police_district_number  calltime_callroute  calltime_dispatch  \
count            16777.000000        1.677700e+04       16777.000000
mean                 2.436431       -6.717058e-16          -0.013467
std                  1.687856        1.000030e+00           0.972179
min                  0.000000       -2.612978e+00          -2.242074
25%                  1.000000       -5.937044e-01          -0.682764
50%                  2.000000       -4.760530e-02          -0.232681
75%                  4.000000        5.811714e-01           0.421137
max                  5.000000        8.050240e+00           3.951448

       callroute_dispatch  calltime_arrive  calltime_cleared  dispatch_arrive  \
count        16777.000000     16777.000000      16777.000000     16777.000000
mean            -0.013588        -0.013745         -0.000004         0.014537
std              0.972206         0.898430          0.999940         0.887344
min             -1.452536        -3.451517         -4.068133        -2.245511
25%             -0.770082        -0.523680         -0.654916        -0.361219
50%             -0.234774        -0.070976         -0.023295         0.068107
75%              0.485387         0.412183          0.664607         0.473064
max              3.576855         4.713377          4.474310         4.667700

       arrive_cleared          hour       weekday
count    16777.000000  16777.000000  16777.000000
mean        -0.006527     13.387018      4.026703
std          0.898058      5.929295      1.904554
min         -2.555425      0.000000      1.000000
25%         -0.539374      9.000000      3.000000
50%         -0.033692     14.000000      4.000000
75%          0.464969     18.000000      6.000000
max          3.941508     23.000000      7.000000
Saved
```

Conclusion

During the data cleaning phase, I transformed the raw Police Dispatched Incidents data into two dataset. I frist created a cleaned dataset suitable for exploratory data analysis by handling missing values, correcting data types, removing irrelevant columns, and modifying data for better interpretation. Then, I further processed the cleaned eda dataset to prepare it for machine learning modeling by normalizing numeric columns, encoding categorical variables, and ensuring no missing values remain.

Challenges

One of the main challenges I faced during the data cleaning process was handling missing longitude and latitude values. At first, almost 2500 of the longitude and latitude values were missing, but luckily, the columns associated with the actual address were filled. Since these columns are crucial for spatial analysis, I used the Google Geocoding API to fill in missing longitude and latitude values based on address information. Another challenge I faced was having to revisit the data cleaning section after completing half of the exploratory data analysis. I had to make several modifications to different columns, which was pretty time-consuming and repetitive.

References

1. Claude, version-4.5, anthropic, NOV-2025, claude.ai.

DSAN 5000 - Incident Response Modeling (MCMD)

Exploratory Data Analysis

Overview

In this section, I will demonstrate the methods and codes used for exploratory data analysis on the dataset. I will first create univariate analysis for numerical variables, displaying their mean, median, and standard deviation. I will also include histograms to visualize this information. I will then conduct categorical variables analysis by using bar charts and highlight any notable trends observed. Then, I will perform multivariate analysis including correlation analysis, crosstabulations, and feature pairings by including heatmaps, scatterplots, and boxplots. Lastly, I will analyze the skewness and Kurtosis of the data. After defining what steps are needed for normalization, I will head back to the machine learing trasformation portion of the previous stage, data cleaning, to apply transformations, and introduce normalization techniques.

Methods and Codes

Univariate Analysis

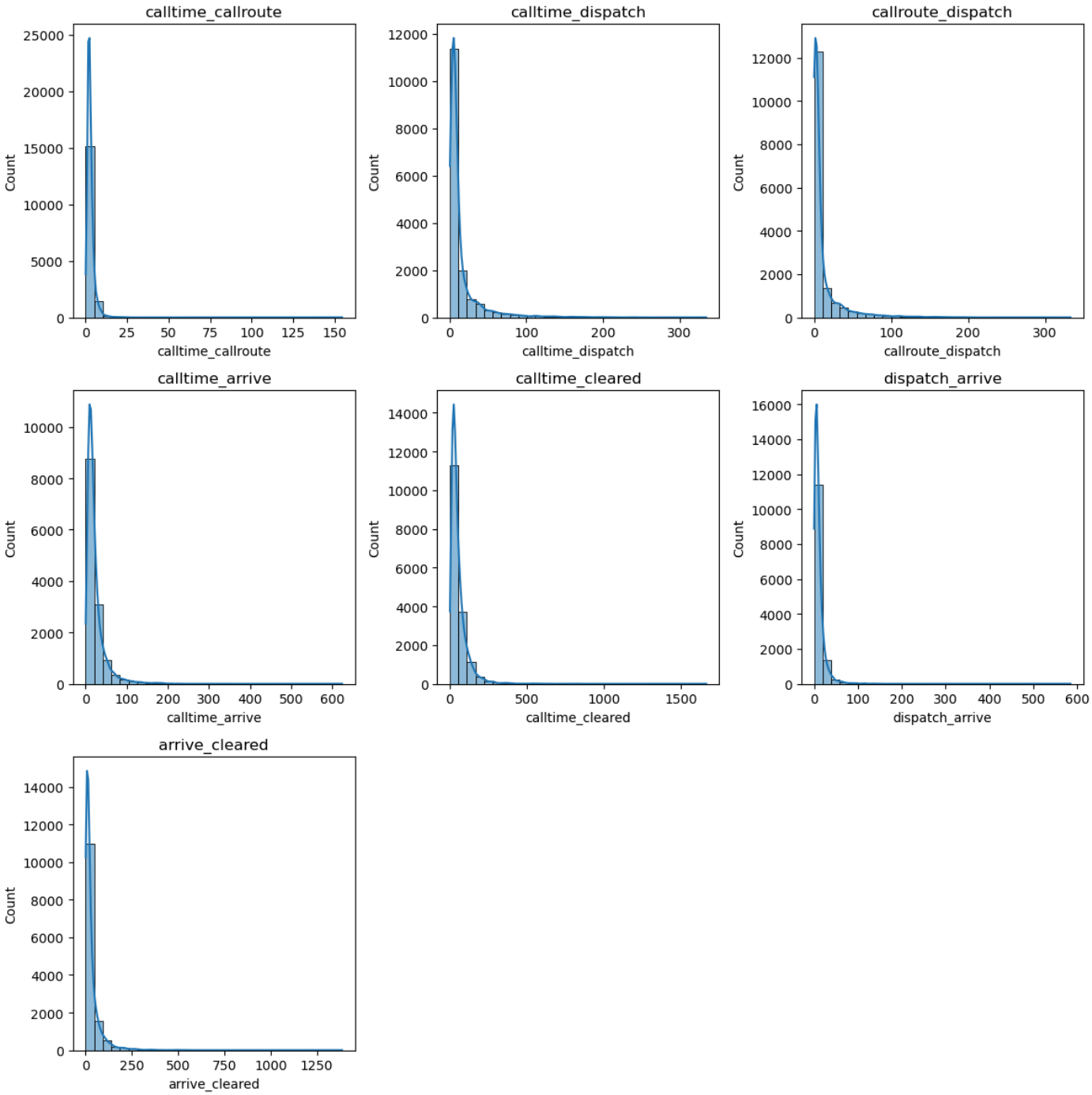
In the following sections, I will explore the distribution of individual numerical variables in the dataset, exploring its mean, median, standard deviation, and visualizing the distribution using histograms. Then, I will analyze categorical variables by examining their frequency counts and visualizing them using bar plots.

Numerical Variables

By looking at the summary statistics of numerical variables below, we can tell that all of the numerical data are skewed to the right, and all of them have an extremely long tail on the right side of the distribution. This is possibly because most police incidents are dispatched, arrived, and cleared within a short period of time, while only a small portion of them take a very long time.

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

# Load the processed police incident data
police_incident = pd.read_csv("../../data/processed-data/police_incident_eda.csv")

# Define all time or numeric related columns
cols = ['calltime_callroute','calltime_dispatch','callroute_dispatch','calltime_arrive','
```

```
# Set up the matplotlib figure
plt.figure(figsize=(12, 12))

# Create histograms for each time-related column
for i, c in enumerate(cols, start=1):
    plt.subplot(3, 3, i)
    sns.histplot(police_incident[c], bins=30, kde=True)
    plt.title(f"{c}")
plt.tight_layout()
plt.show()
```



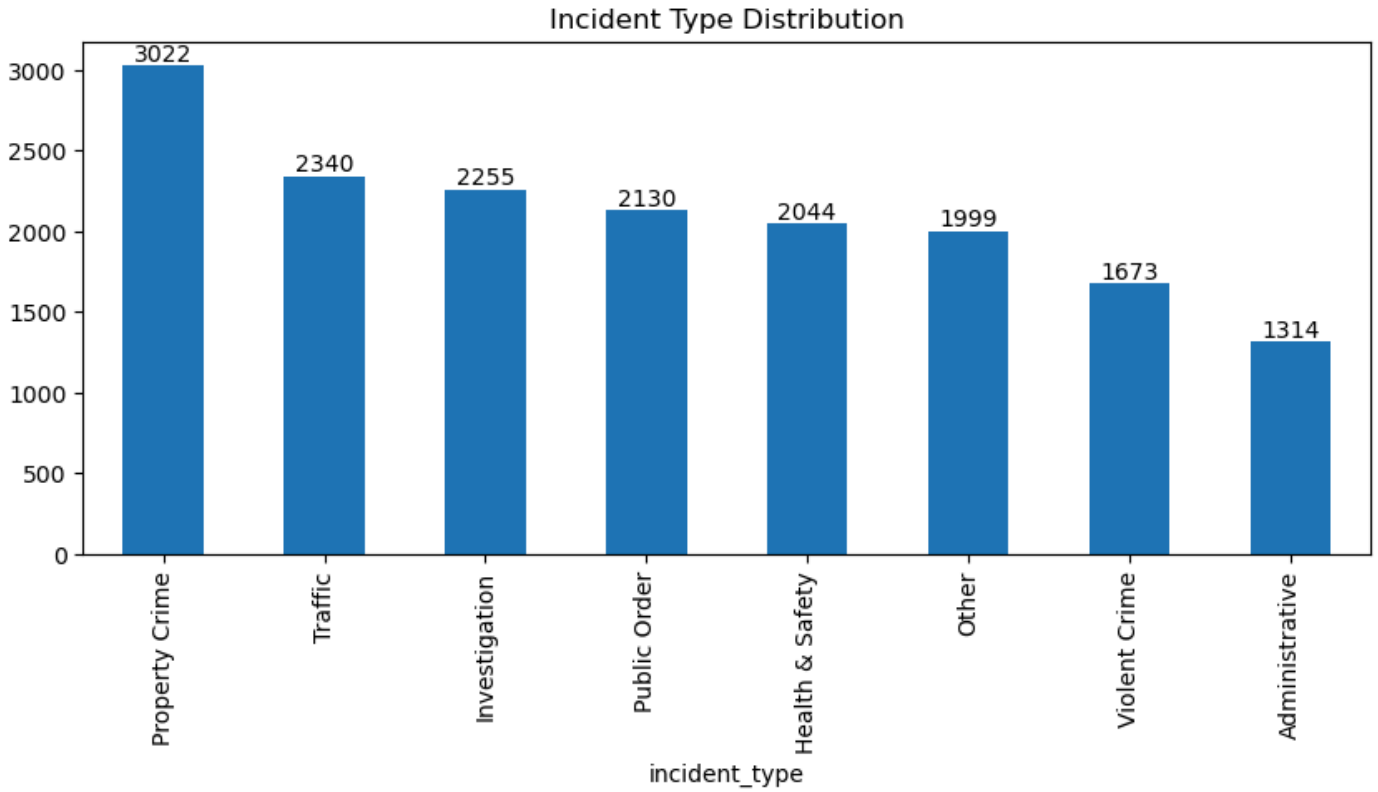
```
# Generate descriptive statistics for the numeric columns
police_incident[cols].describe()
```

	calltime_callroute	calltime_dispatch	callroute_dispatch	calltime_arrive	calltime_cleared	dispatch_arrive	arrive
count	16777.000000	15806.000000	15806.000000	13528.000000	16774.000000	13196.000000	13527.
mean	2.896233	14.334991	11.424692	23.189262	55.736535	11.026771	32.381
std	3.043625	23.914837	23.645323	25.198960	63.632757	14.702792	56.178
min	0.000000	0.133333	0.000000	0.300000	0.333333	0.016667	0.0166
25%	1.600000	3.633333	1.133333	9.583333	22.033333	3.783333	6.8666
50%	2.366667	6.183333	3.083333	15.758333	38.025000	7.533333	15.750
75%	3.533333	12.929167	9.283333	26.870833	68.300000	13.566667	35.425
max	154.366667	334.933333	332.816667	623.566667	1665.833333	585.083333	1386.4

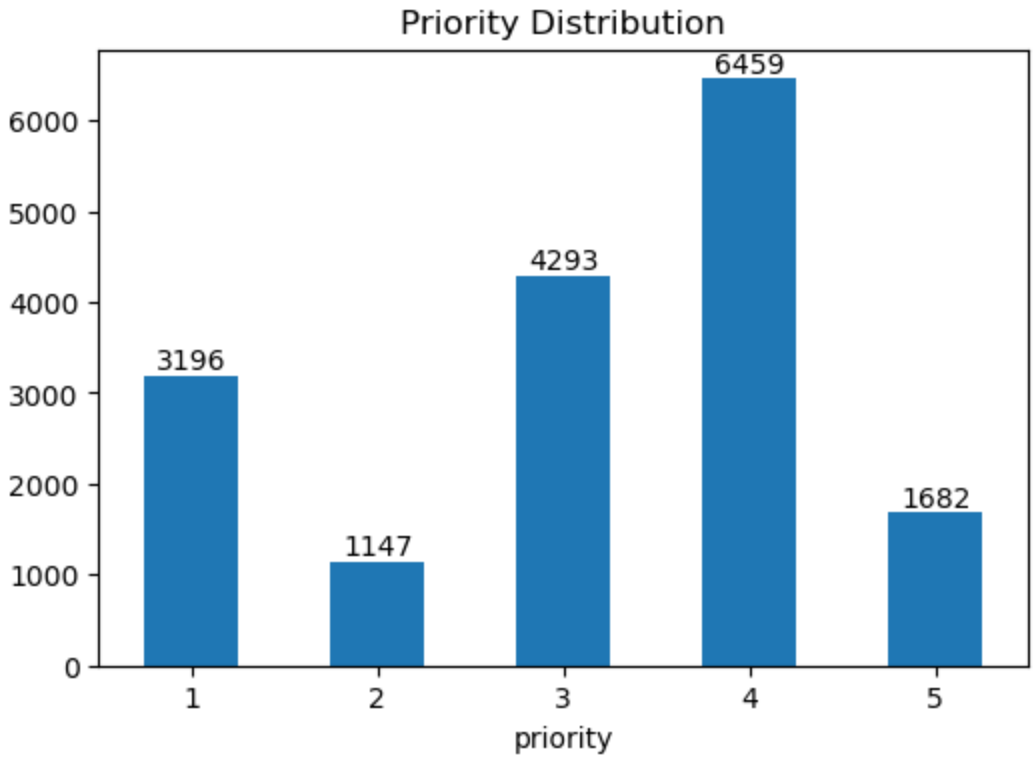
Catagorical Variables

In the following graphs, I will create bar charts for each of the categorical variables, including `incident_type`, `priority`, `police_district_number`, `sector`, and `weekday`. Based on the frequency counts below, the sector contains too many unique values. In the future, I will only be using `police_district_number` for my analysis.

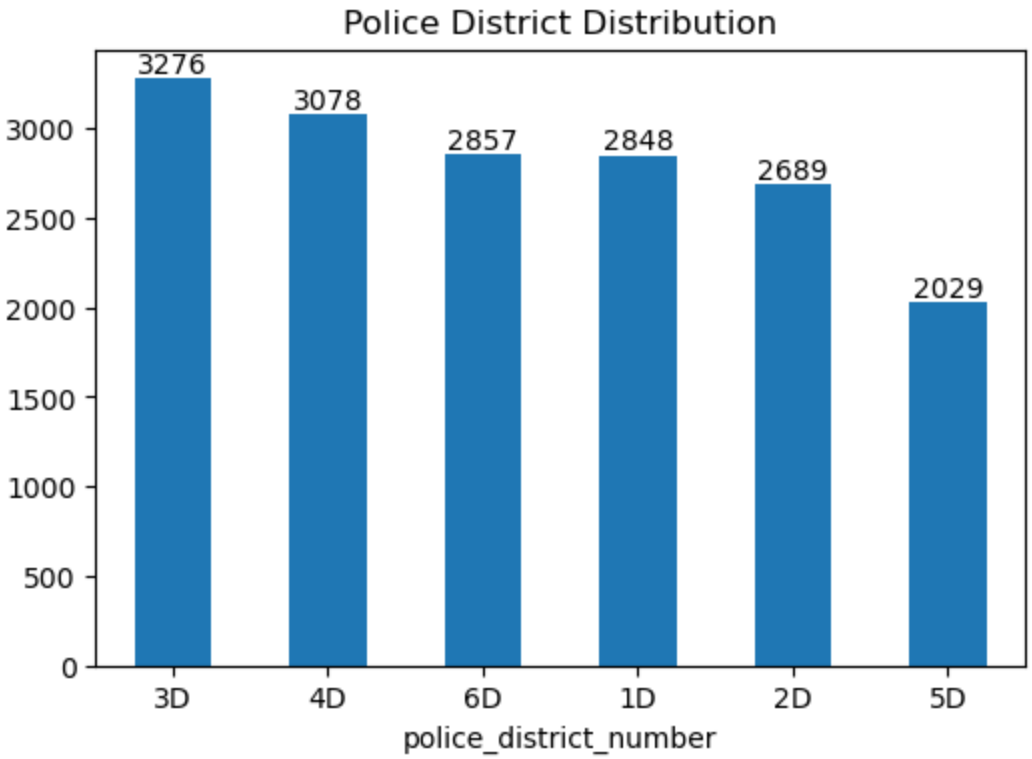
```
# Plot the distribution of incident types
plt.figure(figsize=(10,4))

ax = police_incident['incident_type'].value_counts().plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.title("Incident Type Distribution")
plt.show()
```



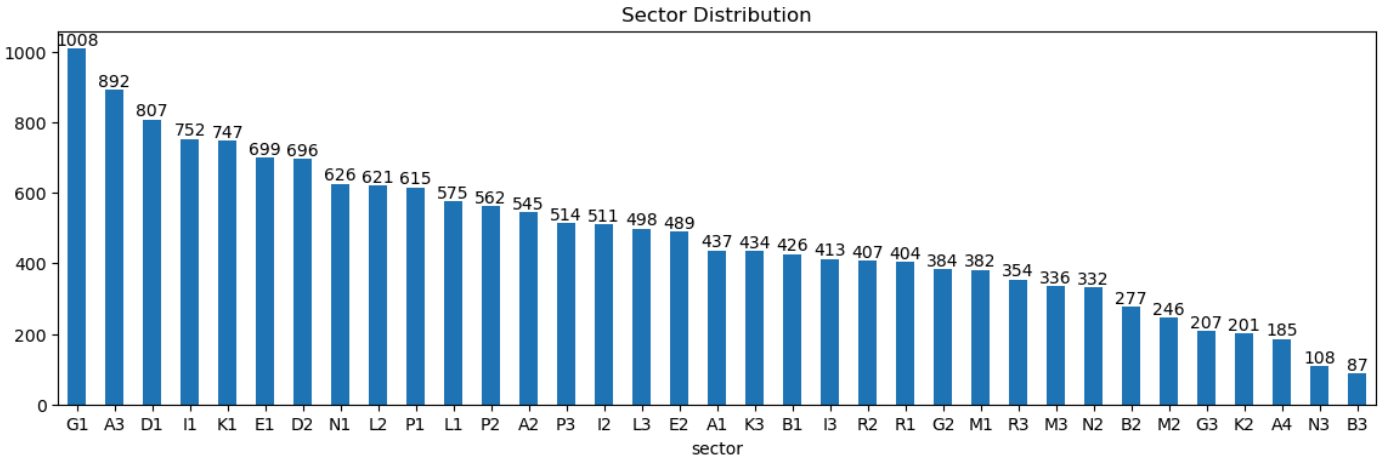

```
# # Plot the distribution of priorities
plt.figure(figsize=(6,4))

ax = police_incident['priority'].value_counts().sort_index().plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.title("Priority Distribution")
plt.xticks(rotation=0)
plt.show()
```




```
# Plot the distribution of police districts
plt.figure(figsize=(6,4))

ax = police_incident['police_district_number'].value_counts().plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.title("Police District Distribution")
plt.xticks(rotation=0)
plt.show()
```

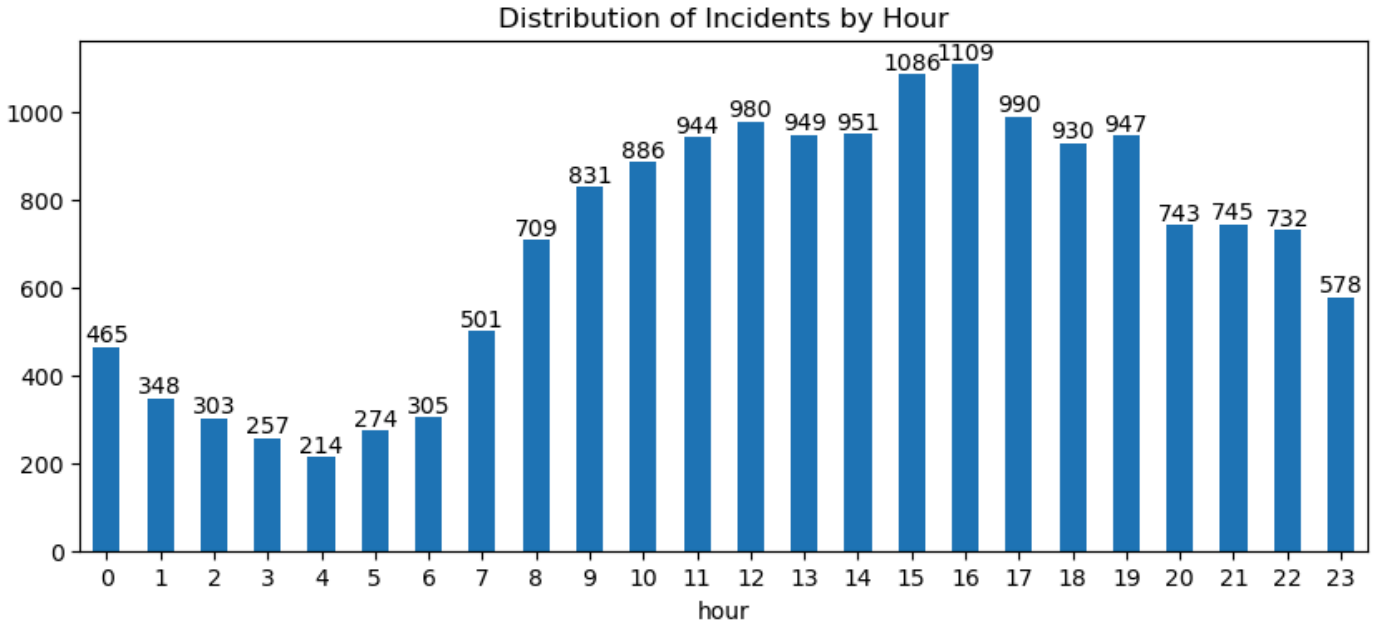



```
# Plot the distribution of sectors
plt.figure(figsize=(14,4))

ax = police_incident['sector'].value_counts().plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.title("Sector Distribution")
plt.xticks(rotation=0)
plt.show()
```

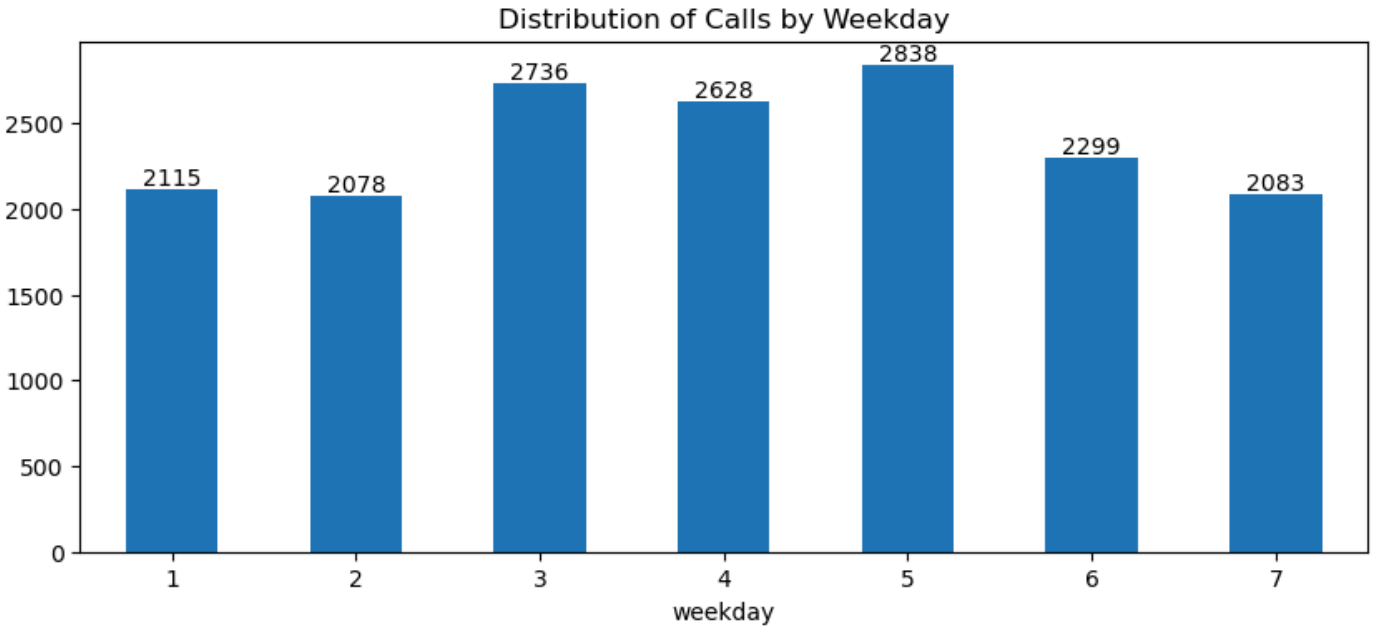



```
# Plot the distribution of incidents by hour
plt.figure(figsize=(10,4))

ax = police_incident['hour'].value_counts().sort_index().plot(kind='bar')
```

```
ax.bar_label(ax.containers[0])
plt.title("Distribution of Incidents by Hour")
plt.xticks(rotation=0)
plt.show()
```




```
# Plot the distribution of incidents by weekday
plt.figure(figsize=(10,4))

ax = police_incident['weekday'].value_counts().sort_index().plot(kind='bar')
ax.bar_label(ax.containers[0])
plt.title("Distribution of Calls by Weekday")
plt.xticks(rotation=0)
plt.show()
```



Key Insights

For all numerical variables, just as I mentioned earlier, we can tell that all of the numerical data including `calltime_callroute`, `calltime_dispatch`, `callroute_dispatch`, `calltime_arrive`, `calltime_cleared`, `dispatch_arrive`, `arrive_cleared`, are skewed to the right, and all of these numeric columns have an extremely long tail on the right side of the distribution, possibly because most police incidents are dispatched, arrived, and cleared within a short period of time, while only a small portion of them take a very long time. For the categorical variable bar charts, we can tell that the order of the incident type occurring is in property crime, traffic, investigation, public order, health & safety, other, violent crime, and administrative. police district 3D has the most incidents occurring, followed by 4D, 6D, 1D, 2D, and 5D, which are in the order of Silver Spring, Wheaton, Gaithersburg, Rockville, Bethesda, and Germantown. Incidents are more likely to occur between noon and midnight, peaking around 4 PM. Incidents are more likely to occur on Wednesdays, Thursdays, and Fridays.

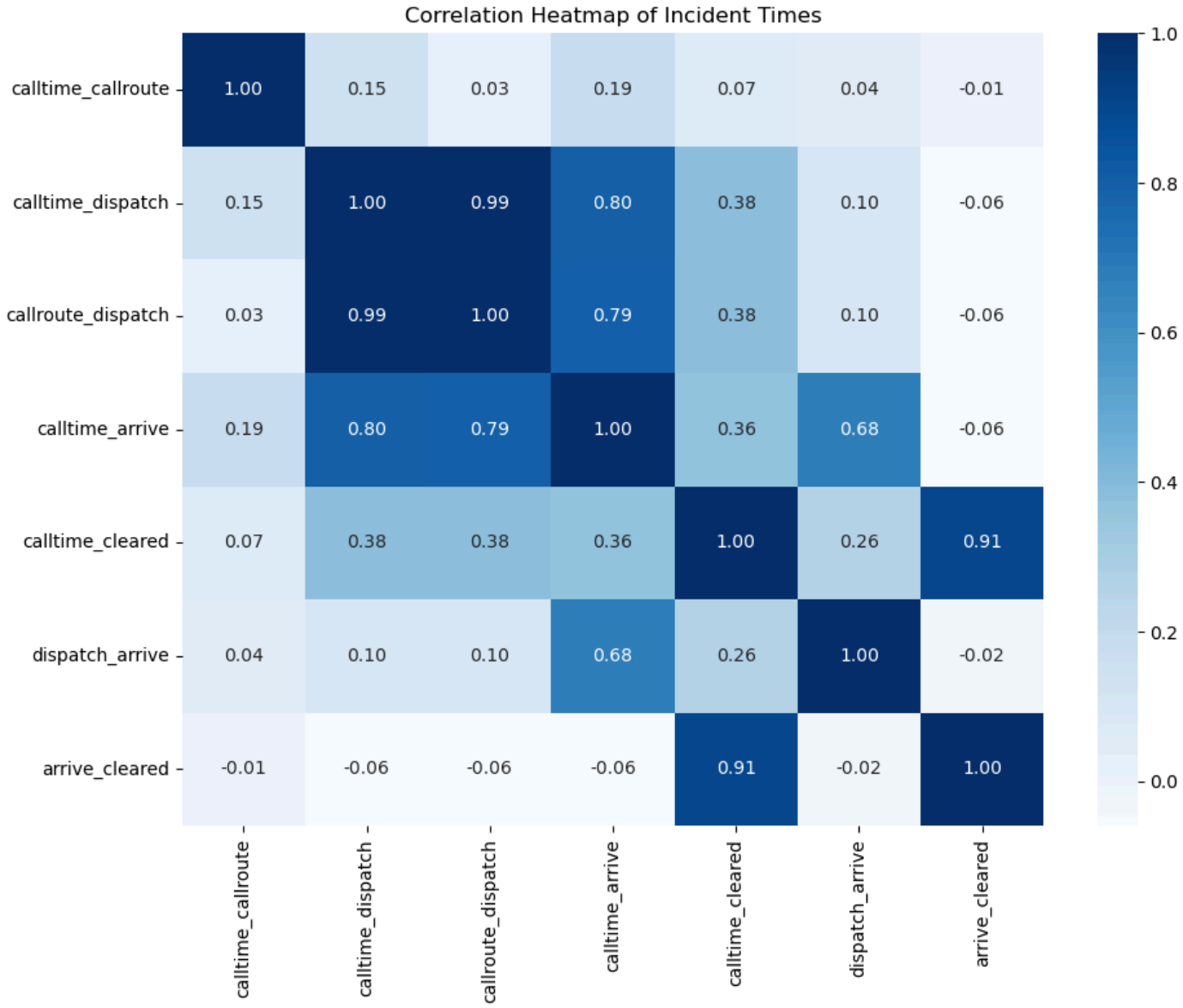
Bivariate and Multivariate Analysis

In the following sections, I will explore relationships between pairs of variables in the dataset, conducting bivariate and multivariate analysis. I will use heatmaps to visualize relationships between numerical variables and violin plots to examine interactions between categorical and numerical variables.

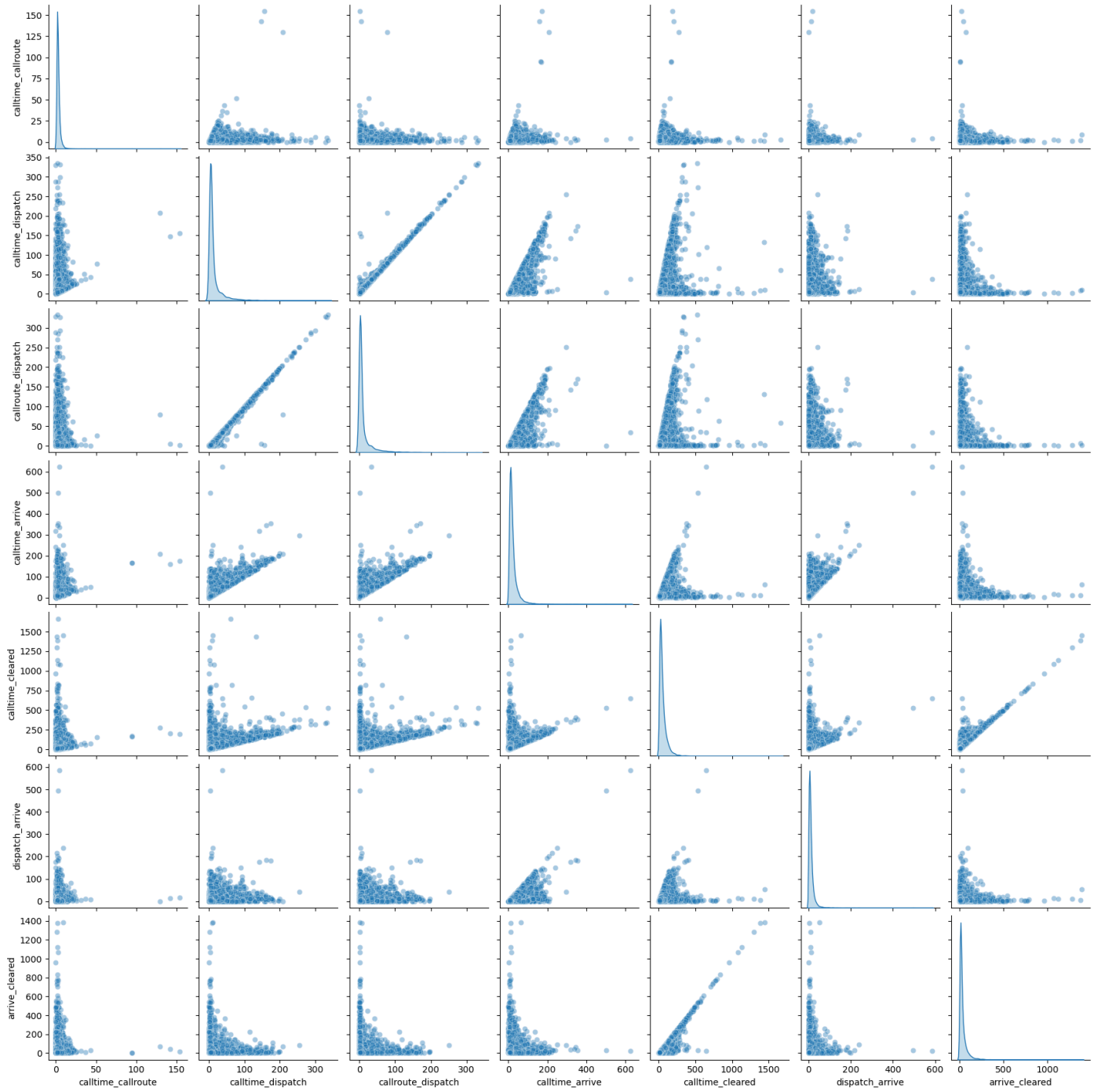
Correlation Analysis

In this part, I will analyze relationships between numerical variables using a correlation matrix, and a heat map. I will explore if any strong correlation exists between numerical variables of different call time variables.

```
# Correlation heatmap of time-related columns
plt.figure(figsize=(10,8))

sns.heatmap(police_incident[cols].corr(), annot=True, cmap='Blues', fmt=".2f")
plt.title("Correlation Heatmap of Incident Times")
plt.tight_layout()
plt.show()
```




```
# Pairplot of time-related columns
sns.pairplot(police_incident[cols], diag_kind="kde", plot_kws={'alpha':0.4})
plt.show()
```

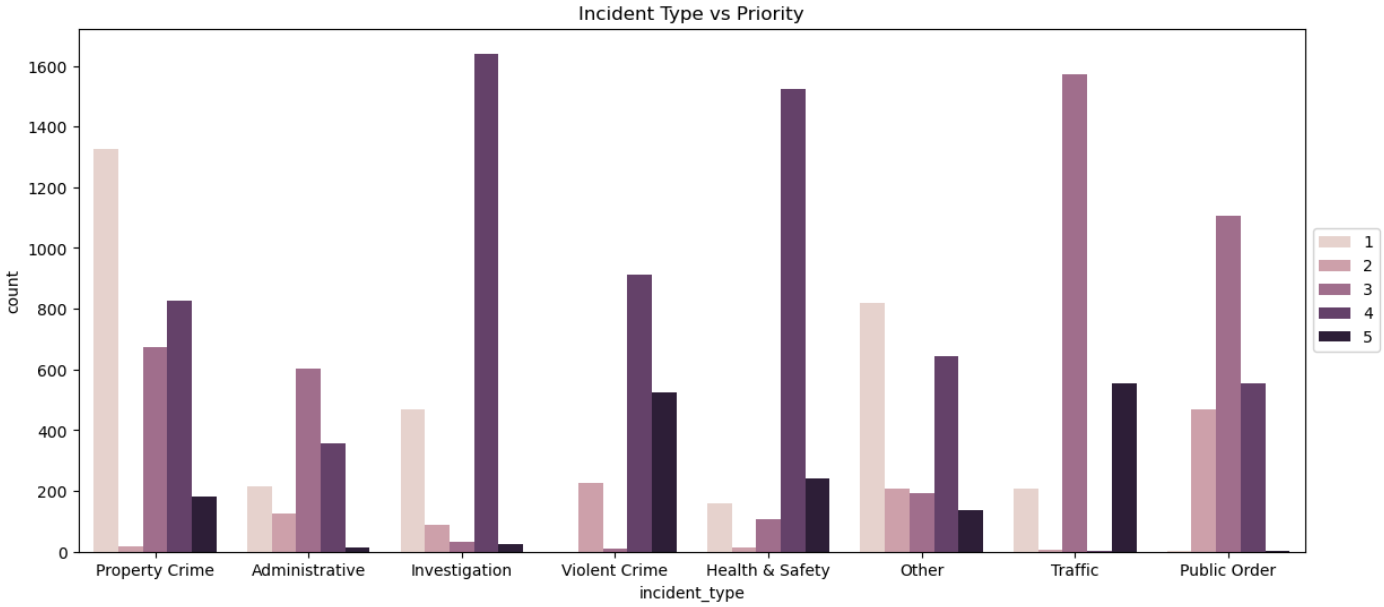


Crosstabulations

For this following section about categorical variables, I will use crosstabulations to explore the relationships and visualize them with grouped bar plots.

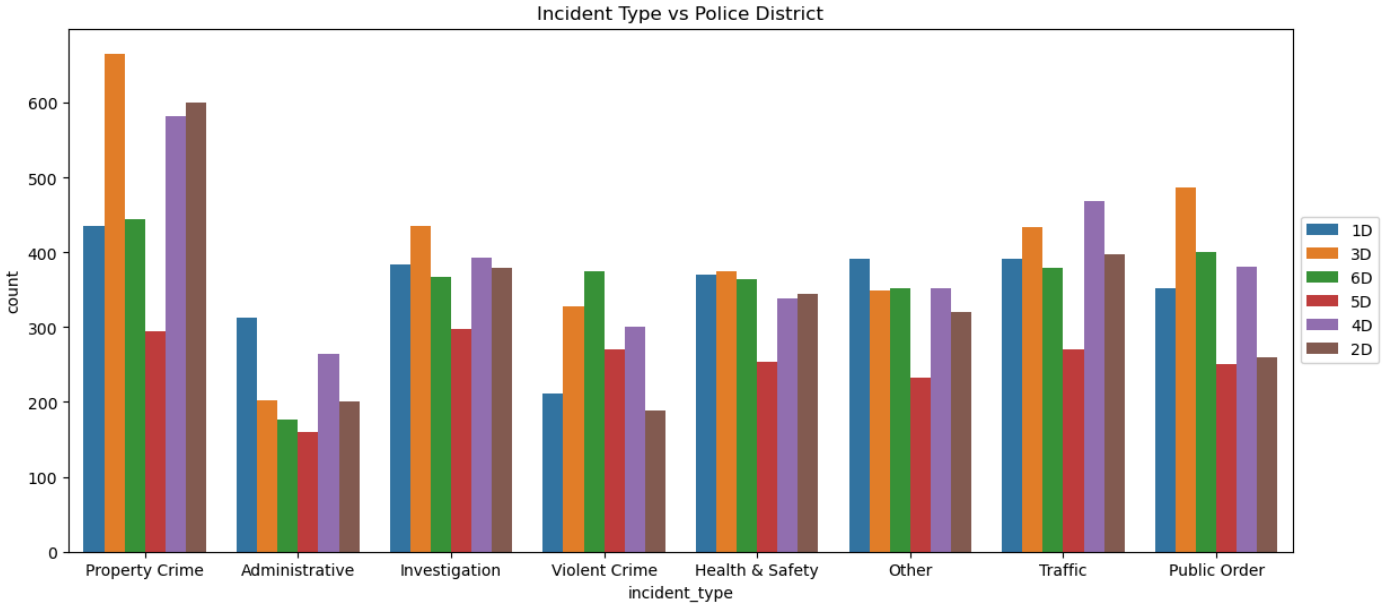
```
# Grouped bar plot of incident type vs priority
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='incident_type', hue='priority')
plt.title("Incident Type vs Priority")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig("../../report/visual1.png", dpi=300, bbox_inches="tight")
plt.show()
```

```
# Crosstab of incident type and priority
print(pd.crosstab(police_incident['incident_type'], police_incident['priority']))
```



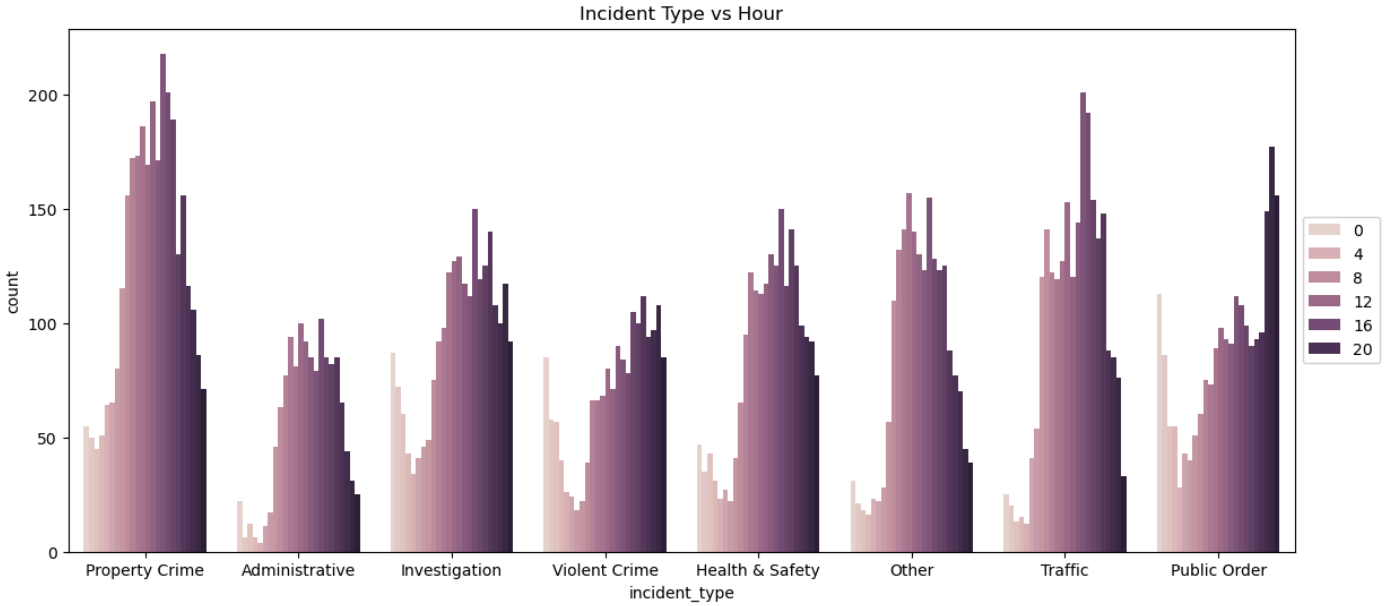
priority	1	2	3	4	5
incident_type					
Administrative	216	124	603	356	15
Health & Safety	158	13	108	1524	241
Investigation	468	89	31	1641	26
Other	820	206	191	644	138
Property Crime	1325	16	674	825	182
Public Order	1	468	1106	554	1
Traffic	208	6	1571	2	553
Violent Crime	0	225	9	913	526

```
# Grouped bar plot of incident type vs police district
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='incident_type', hue='police_district_number')
plt.title("Incident Type vs Police District")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()

# Crosstab of incident type and police district
print(pd.crosstab(police_incident['incident_type'], police_incident['police_district_numb
```



police_district_number	1D	2D	3D	4D	5D	6D
incident_type						
Administrative	312	200	202	264	160	176
Health & Safety	370	344	375	338	253	364
Investigation	384	379	435	393	297	367
Other	392	321	349	352	233	352
Property Crime	435	600	666	582	295	444
Public Order	352	260	487	381	250	400
Traffic	391	397	434	468	270	380
Violent Crime	212	188	328	300	271	374

```
# Grouped bar plot of incident type vs hour
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='incident_type', hue='hour')
plt.title("Incident Type vs Hour")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()

# Crosstab of incident type and hour
print(pd.crosstab(police_incident['incident_type'], police_incident['hour']))
```

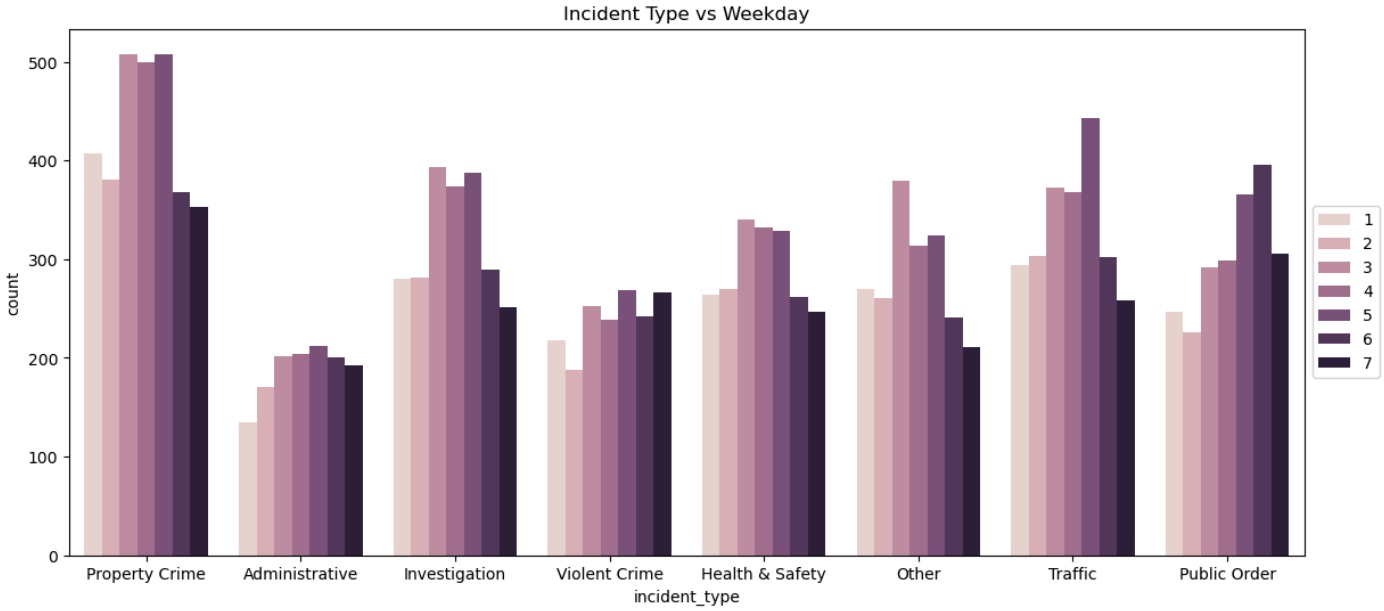


hour	0	1	2	3	4	5	6	7	8	9	...	14	15	\
incident_type											...			
Administrative	22	6	12	6	4	11	17	46	63	77	...	85	79	
Health & Safety	47	35	43	31	23	27	22	41	65	95	...	130	125	
Investigation	87	72	60	43	34	41	46	49	75	92	...	117	112	
Other	31	21	18	16	23	22	28	57	110	132	...	123	155	
Property Crime	55	50	45	51	64	65	80	115	156	172	...	171	218	
Public Order	113	86	55	55	28	43	40	51	60	75	...	91	112	
Traffic	25	20	13	15	12	41	54	120	141	122	...	144	201	
Violent Crime	85	58	57	40	26	24	18	22	39	66	...	90	84	

hour	16	17	18	19	20	21	22	23
incident_type								
Administrative	102	85	82	85	65	44	31	25
Health & Safety	150	116	141	125	99	94	92	77
Investigation	150	119	125	140	108	100	117	92
Other	128	123	125	88	77	70	45	39
Property Crime	201	189	130	156	116	106	86	71
Public Order	108	99	90	93	96	149	177	156
Traffic	192	154	137	148	88	85	76	33
Violent Crime	78	105	100	112	94	97	108	85

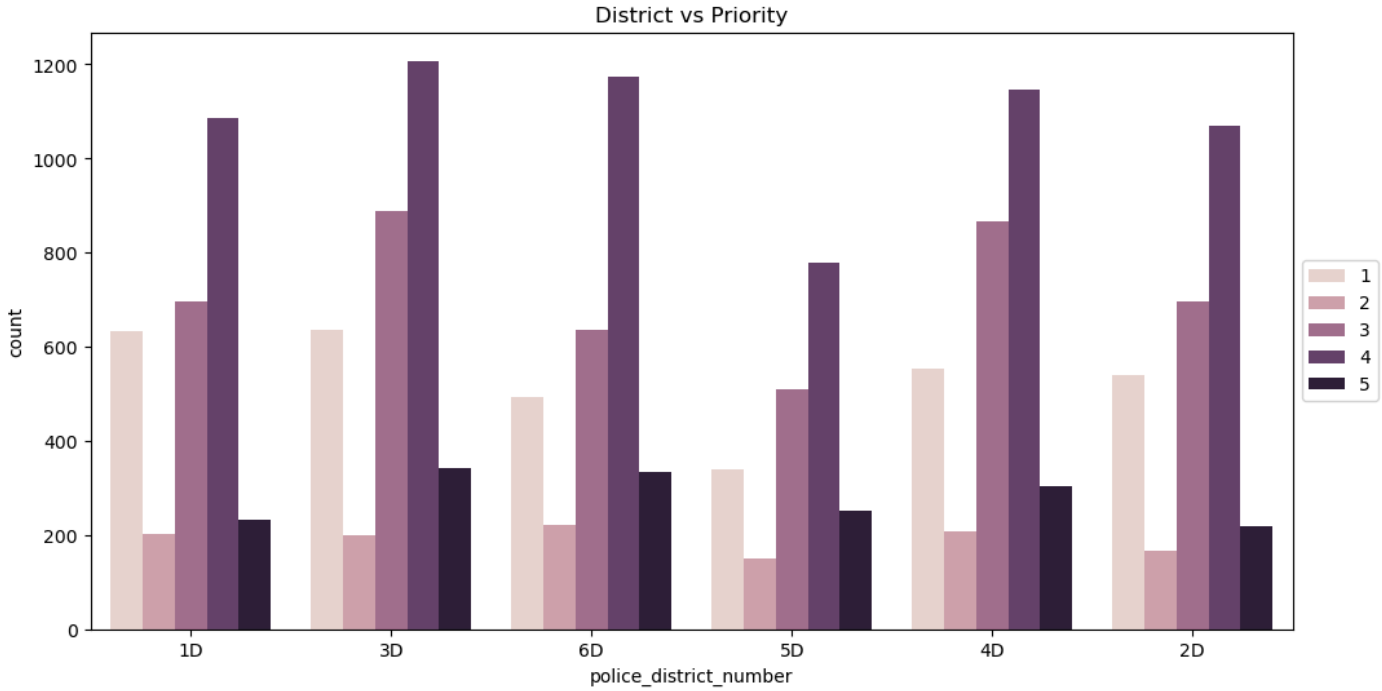
[8 rows x 24 columns]

```
# Grouped bar plot of incident type vs weekday
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='incident_type', hue='weekday')
plt.title("Incident Type vs Weekday")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()

# Crosstab of incident type and weekday
print(pd.crosstab(police_incident['incident_type'], police_incident['weekday']))
```



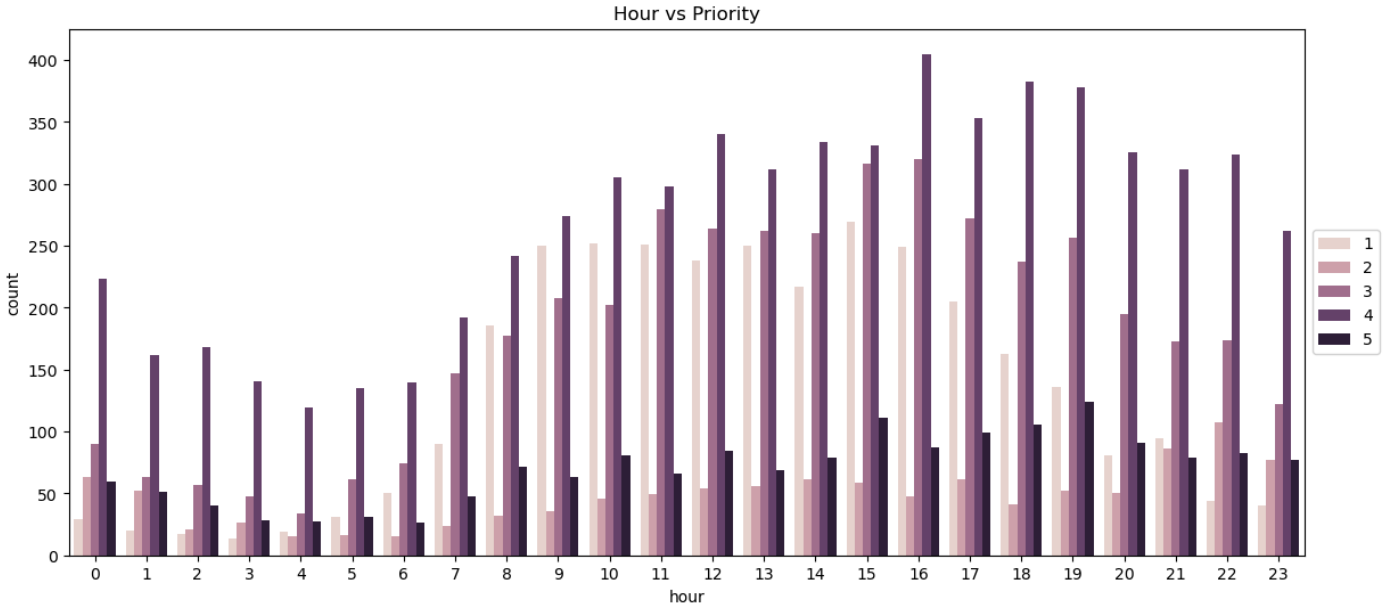
weekday	1	2	3	4	5	6	7
incident_type							
Administrative	135	170	201	204	212	200	192
Health & Safety	264	270	340	332	329	262	247
Investigation	280	281	393	374	387	289	251
Other	270	260	379	314	324	241	211
Property Crime	407	380	507	499	508	368	353
Public Order	247	226	292	299	366	395	305
Traffic	294	303	372	368	443	302	258
Violent Crime	218	188	252	238	269	242	266

```
# Grouped bar plot of district vs priority
plt.figure(figsize=(12,6))
sns.countplot(data=police_incident, x='police_district_number', hue='priority')
plt.title("District vs Priority")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.savefig("../../report/visual5.png", dpi=300, bbox_inches="tight")
plt.show()

# Crosstab of police district and priority
print(pd.crosstab(police_incident['police_district_number'], police_incident['priority']))
```



priority	1	2	3	4	5
police_district_number					
1D	634	202	696	1085	231
2D	539	167	696	1069	218
3D	637	200	888	1208	343
4D	554	207	867	1146	304
5D	340	151	509	778	251
6D	492	220	637	1173	335

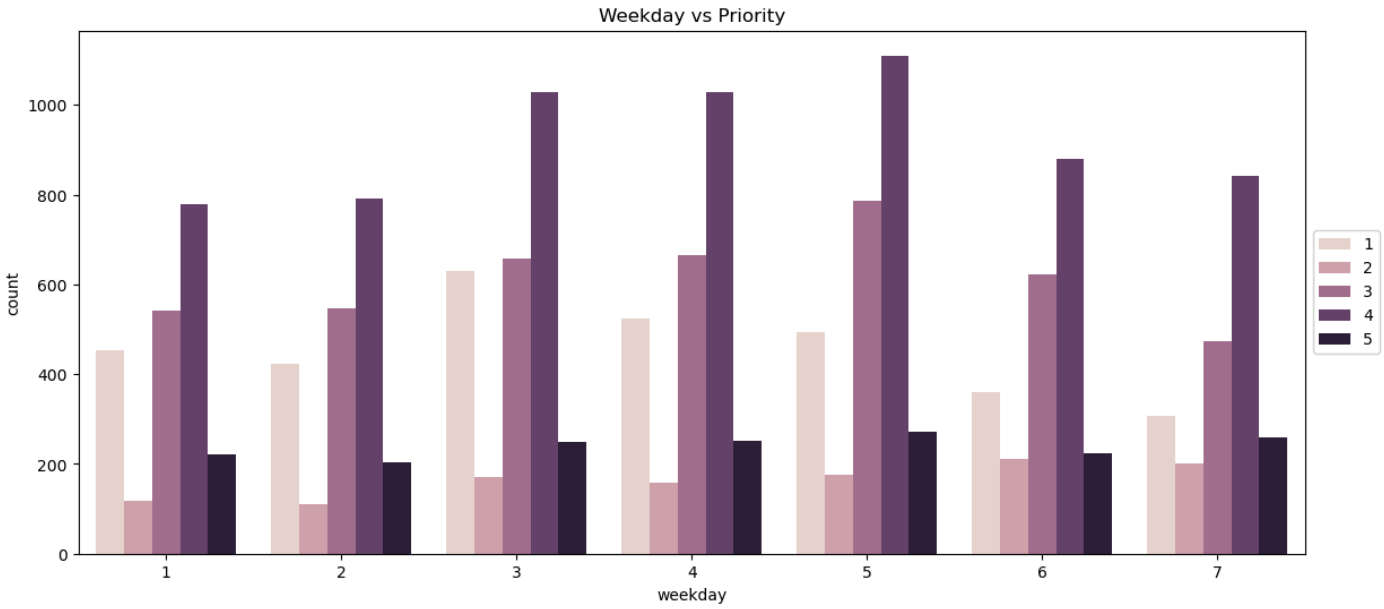
```
# Grouped bar plot of hour vs priority
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='hour', hue='priority')
plt.title("Hour vs Priority")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()

# Crosstab of hour and priority
print(pd.crosstab(police_incident['hour'], police_incident['priority']))
```



priority	1	2	3	4	5
hour					
0	29	63	90	223	60
1	20	52	63	162	51
2	17	21	57	168	40
3	14	26	48	141	28
4	19	15	34	119	27
5	31	16	61	135	31
6	50	15	74	140	26
7	90	24	147	192	48
8	186	32	177	242	72
9	250	36	208	274	63
10	252	46	202	305	81
11	251	49	280	298	66
12	238	54	264	340	84
13	250	56	262	312	69
14	217	61	260	334	79
15	269	59	316	331	111
16	249	48	320	405	87
17	205	61	272	353	99
18	163	41	237	383	106
19	136	52	257	378	124
20	81	50	195	326	91
21	95	86	173	312	79
22	44	107	174	324	83
23	40	77	122	262	77

```
# Grouped bar plot of weekday vs priority
plt.figure(figsize=(14,6))
sns.countplot(data=police_incident, x='weekday', hue='priority')
plt.title("Weekday vs Priority")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```

```
# Crosstab of weekday and priority
print(pd.crosstab(police_incident['weekday'], police_incident['priority']))
```

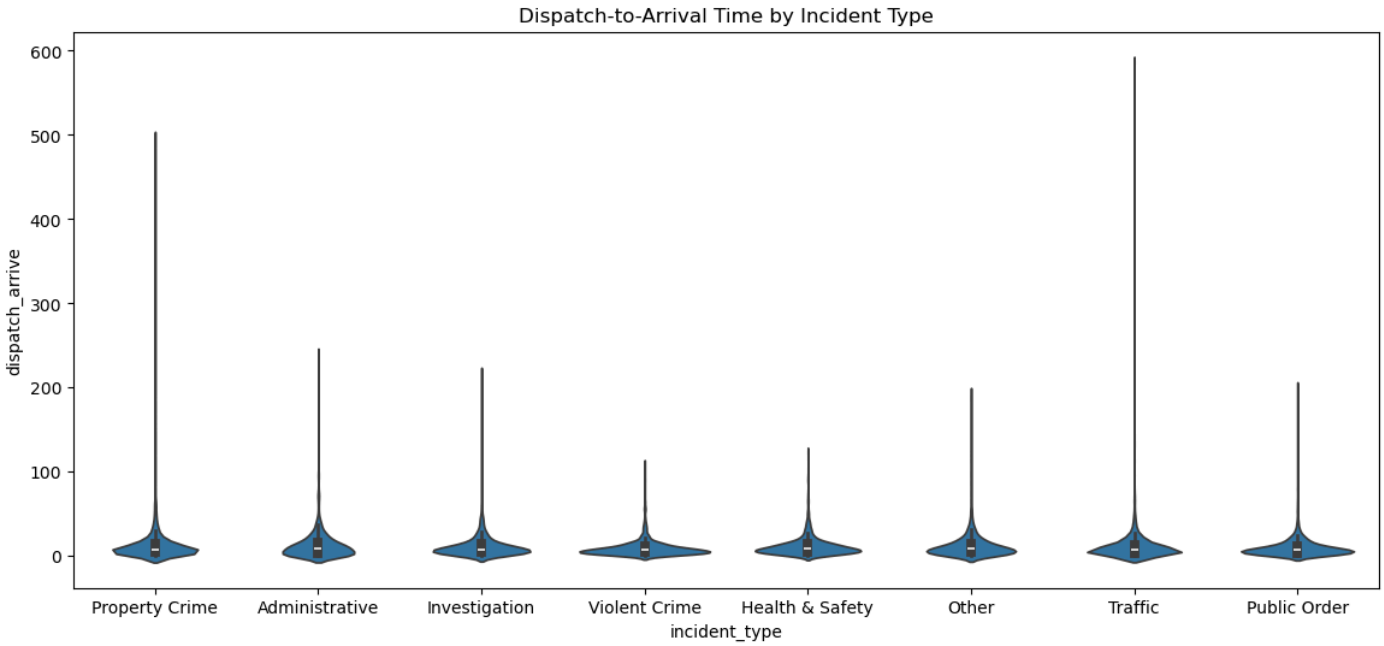


priority	1	2	3	4	5
weekday					
1	453	119	542	779	222
2	424	111	547	792	204
3	631	170	657	1029	249
4	524	159	665	1028	252
5	495	175	786	1110	272
6	361	212	623	879	224
7	308	201	473	842	259

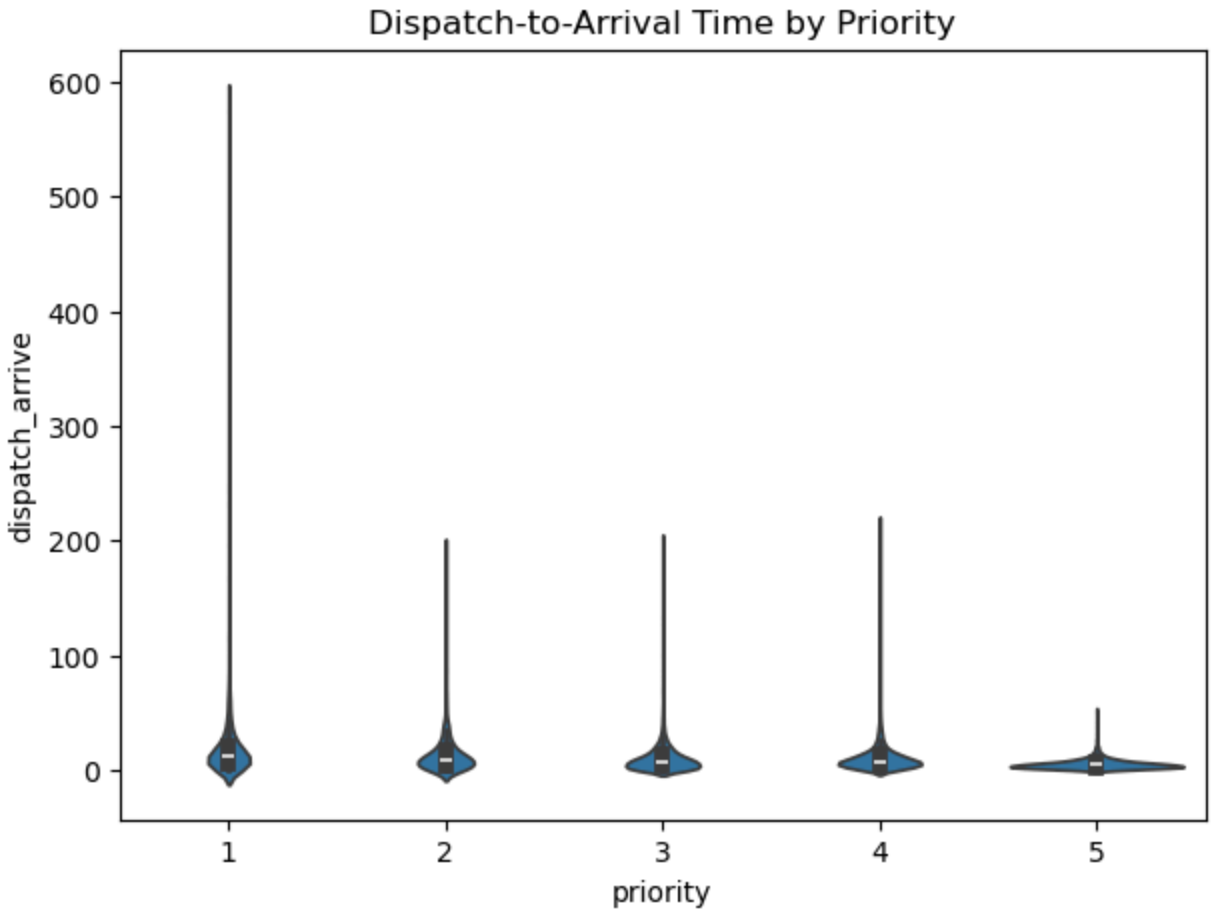
Feature Pairings

In this section, I will analyze relationships between key variables using violin plots. I will explore how different variables interact with each other, such as how incident types vary across dispatch-to-arrival time and how dispatch-to-arrival times differ by priority.

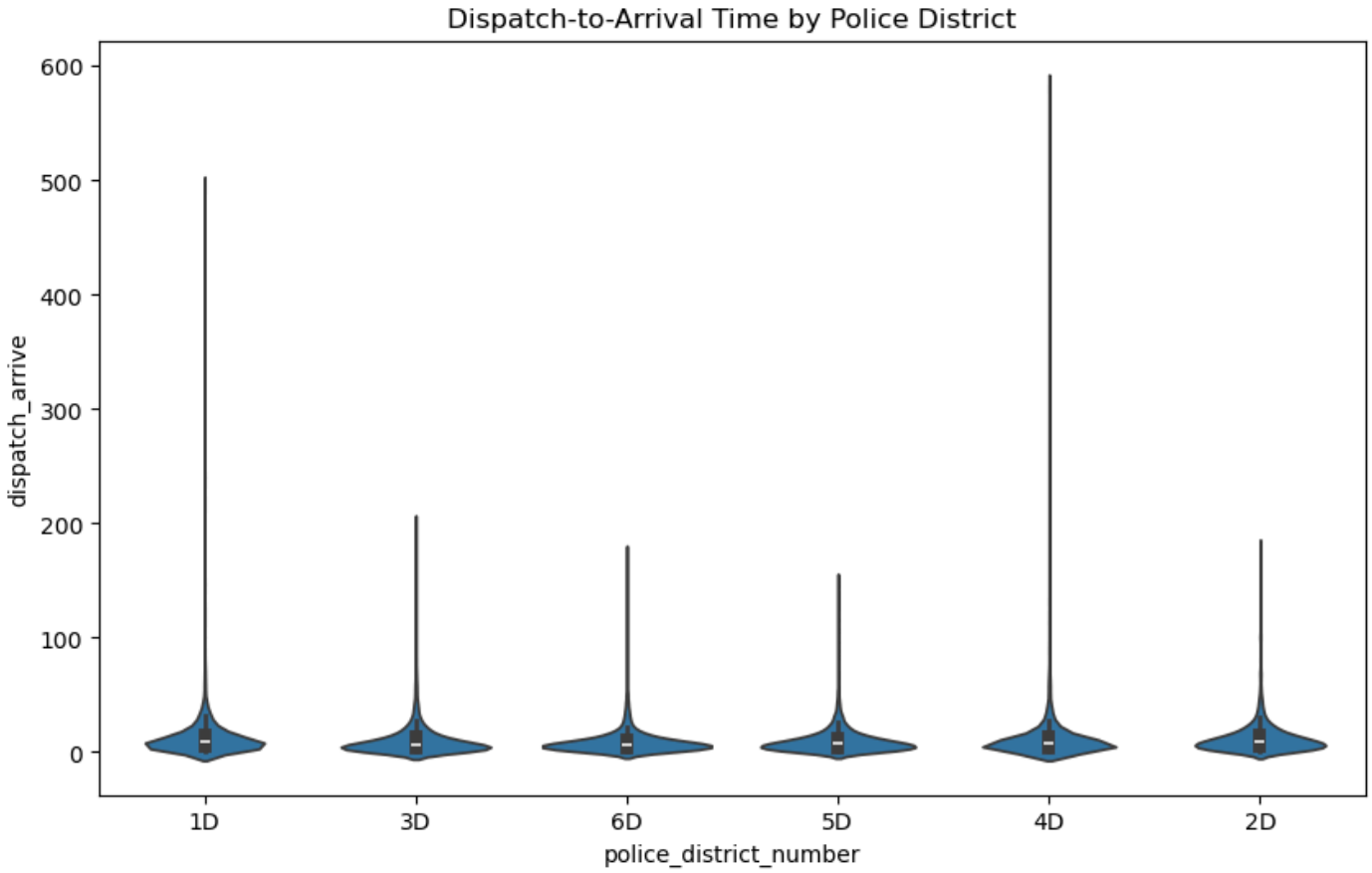
```
# Violin plot of dispatch to arrival time by incident type
plt.figure(figsize=(14,6))
sns.violinplot(data=police_incident, x='incident_type', y='dispatch_arrive')
plt.title("Dispatch-to-Arrival Time by Incident Type")
plt.show()
```



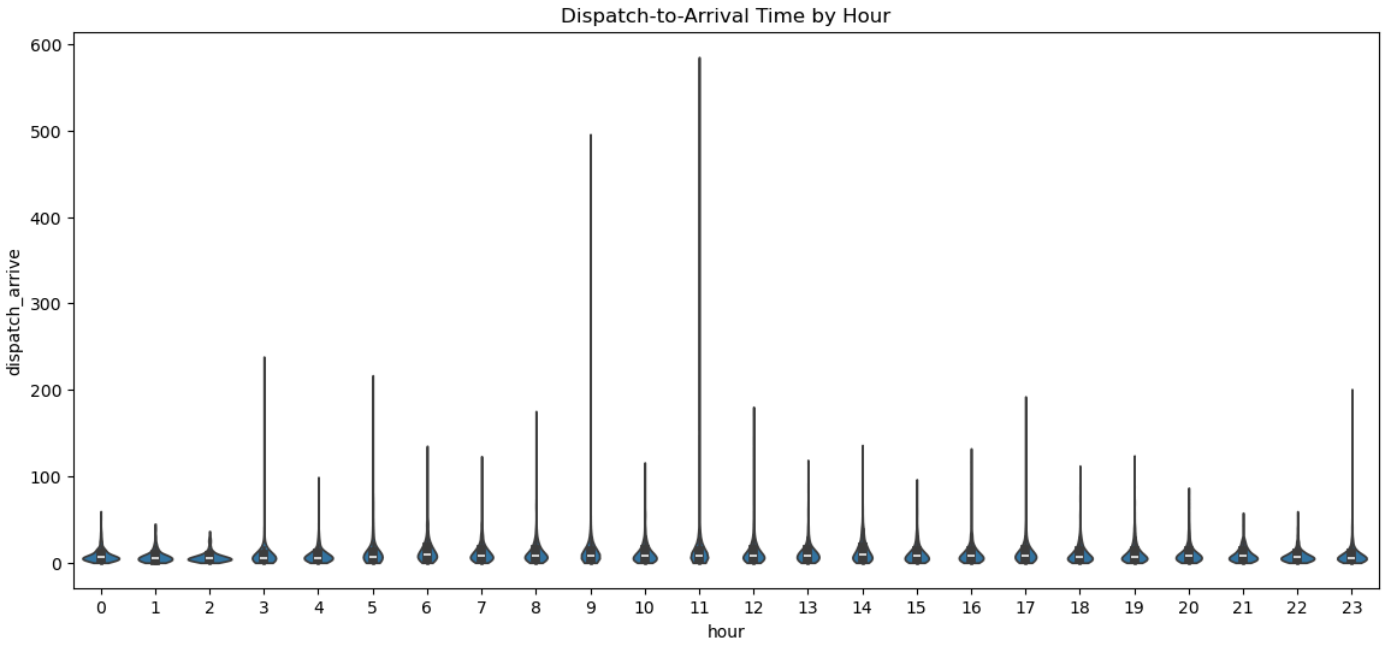

```
# Violin plot of dispatch to arrival time by priority
plt.figure(figsize=(7,5))
sns.violinplot(data=police_incident, x='priority', y='dispatch_arrive')
plt.title("Dispatch-to-Arrival Time by Priority")
plt.savefig("../../report/visual6.png", dpi=300, bbox_inches="tight")
plt.show()
```



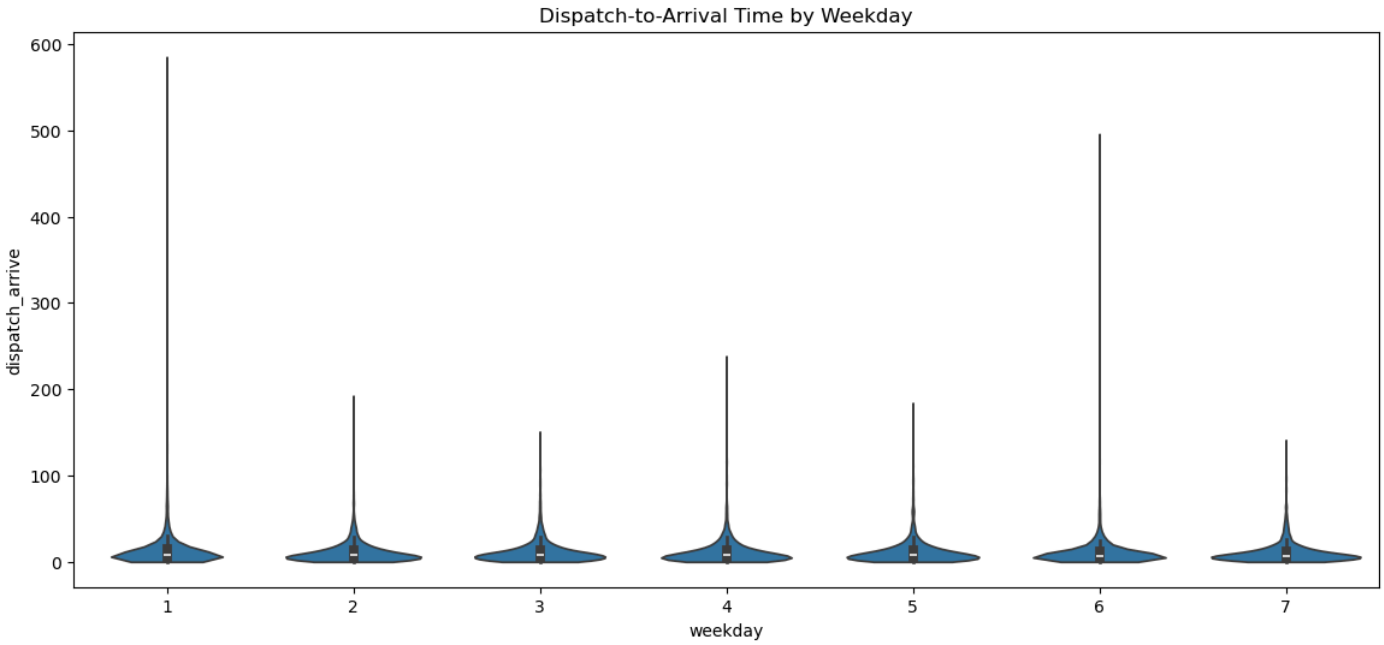

```
# Violin plot of dispatch to arrival time by police district
plt.figure(figsize=(10,6))
sns.violinplot(data=police_incident, x='police_district_number', y='dispatch_arrive')
plt.title("Dispatch-to-Arrival Time by Police District")
plt.show()
```



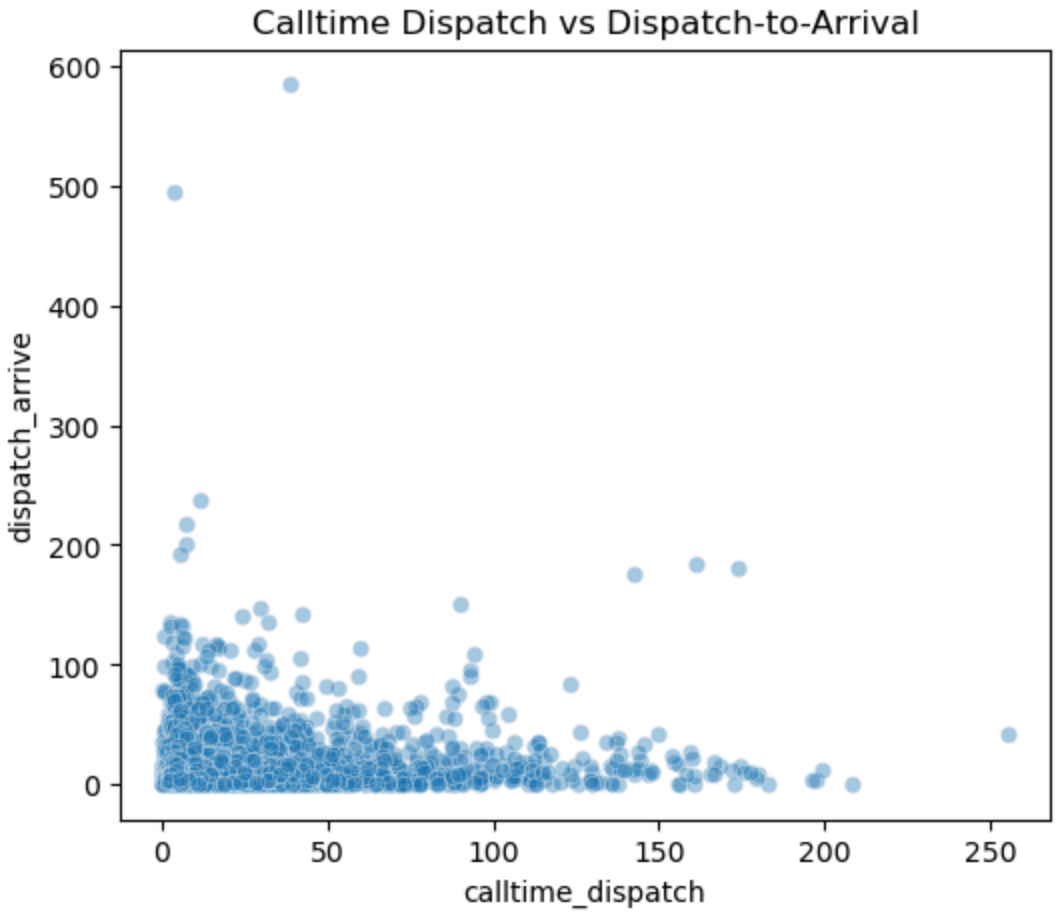

```
# Violin plot of dispatch to arrival time by hour
plt.figure(figsize=(14,6))
sns.violinplot(data=police_incident, x='hour', y='dispatch_arrive', cut=0)
plt.title("Dispatch-to-Arrival Time by Hour")
plt.show()
```



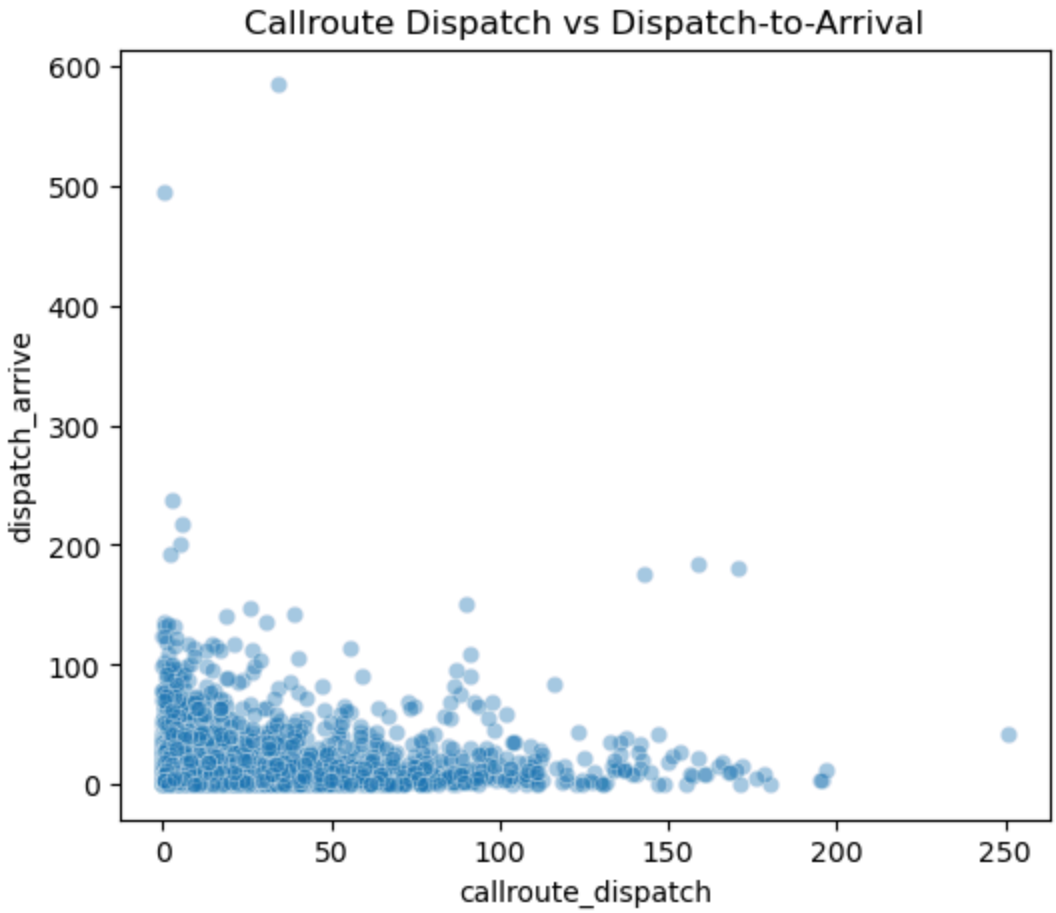
```
# Violin plot of dispatch to arrival time by weekday
plt.figure(figsize=(14,6))
sns.violinplot(data=police_incident, x='weekday', y='dispatch_arrive', cut=0)
plt.title("Dispatch-to-Arrival Time by Weekday")
plt.show()
```




```
# Scatter plot of calltime_dispatch vs dispatch_arrive
plt.figure(figsize=(6,5))
sns.scatterplot(data=police_incident, x='calltime_dispatch', y='dispatch_arrive', alpha=0
plt.title("Calltime Dispatch vs Dispatch-to-Arrival")
plt.show()
```




```
# Scatter plot of callroute_dispatch vs dispatch_arrive
plt.figure(figsize=(6,5))
sns.scatterplot(data=police_incident, x='callroute_dispatch', y='dispatch_arrive', alpha=
plt.title("Callroute Dispatch vs Dispatch-to-Arrival")
plt.show()
```



Key Insights

In this bivariate and multivariate analysis section, I found that there is no strong correlation between most of the numeric variables, except for `callroute_dispatch` and `calltime_dispatch`, `calltime_arrive` and `calltime_dispatch`, `calltime_arrive` and `callroute_dispatch`, `arrive_cleared` and `calltime_cleared`, which all have a moderate to strong positive correlation, which makes sense because these variables are related to each other in terms of time intervals. In the crosstabulation section, several interesting patterns were discovered. For the highest incident priority level (5), the most common incident types were property crime, violent crime, health & safety, and traffic-related incidents. What is interesting is that property crime is also the type with the most incidents, and it is associated with the lowest priority level. In the `incident_type` vs `police_district_number` grouped bar chart, all crime types were distributed relatively evenly, except for property crime in district 3D (Silver Spring), which had a significantly higher count than in other districts. In the `incident_type` vs. `hour` graph, all incident types peaked around 4pm except public order. Public order included the incident type of noise complaints, which peaked late at night. Except for public order incidents, all other incident types showed a similar distribution throughout the day. In the `incident_type` vs. `weekday` graph, nearly all incident types peaked on Wednesday, Thursday, and Friday, which is pretty consistent with the univariate analysis. From the `police_district_number` vs. `priority` graph, nothing stood out; all districts had a similar distribution of incident priority levels. For the `hour` vs. `priority` graph, all priority levels peaked between 3pm and 7pm. Same for the `weekday` vs `priority` graph: all priority levels peaked on Wednesday, Thursday, and Friday.

Now let's look at the difference feature pairing graphs. In the `dispatch_arrive` time by `incident_type` graph, the dispatch to arrival times are distributed similarly across all incident types. However, for property crime and traffic-related incidents, there are some outliers with extremely high dispatch to arrival times. In the `dispatch_arrive` by `priority` graph, we can see that priority level 1 has the highest dispatch to arrival time, while priority level 5 has the lowest, with no outliers or extreme cases. Priority levels 2, 3, and 4 are all relatively similar. `dispatch_arrive` distribution by `police_district_number`: all districts show a similar distribution, with some extreme outliers in 1D and 4D. In the `dispatch_arrive` by `hour` graph, we can see that incidents occurring at night arrive significantly faster than those during the day. `dispatch_arrive` by `weekday` graph: all weekdays show a similar distribution, with no significant difference, but Monday and Thursday have extreme outliers with very high dispatch to arrival times. For the two numerical variable pairings, `calltime_dispatch` vs `dispatch_arrive` and `callroute_dispatch` vs `dispatch_arrive`, no clear patterns were observed.

Data Distribution and Normalization

For Skewness and Kurtosis, and the Normalization section of the EDA, please see the data cleaning section Codes and Method (Machine Learning).

Statistical Insights

In this section, I will conduct several basic statistical tests, including T-tests, ANOVA, and chi-square tests, to explore relationships among variables. I will also summarize the statistical results and their implications for my analysis.

T-Test

Research Question: Do high-priority calls have significantly different dispatch to arrival times than low-priority calls?

Null Hypothesis: The mean dispatch to arrival time for high-priority calls is equal to that of low-priority calls.

Alternative Hypothesis: The mean dispatch to arrival time for high-priority calls is different to that of low-priority calls.

```
import scipy.stats as stats
import pandas as pd

# Load the processed police incident data
police_incident = pd.read_csv("../../data/processed-data/police_incident_eda.csv")

# Define high and low priority groups
low = police_incident[police_incident['priority'] <= 3]['dispatch_arrive']
high  = police_incident[police_incident['priority'] >= 4]['dispatch_arrive']

# Drop NaN values
high = high.dropna()
low = low.dropna()
```

```
# Perform t-test
t_stat, p_val = stats.ttest_ind(high, low, equal_var=False)
print("T-statistic:", t_stat)
print("P-value:", p_val)
```

```
T-statistic: -11.911813620480993
P-value: 1.7549529290249306e-32
```

Since the p-value of the t-test above is less than 0.05, we will reject the null hypothesis. The mean dispatch to arrival time for high-priority calls is significantly different from that of low-priority calls. Also, since the t-stat is negative, we can tall that the dispatch to arrival time for high-priority calls is significantly lower than that of low-priority calls.

ANOVA

Research Question: Do different incident types have significantly different arrival to cleared times?

Null Hypothesis: The mean arrival to cleared time is the same across all incident types.

Alternative Hypothesis: At least on incident type has a mean arrival to cleared time different than other incident types.

```
import scipy.stats as stats

# Perform ANOVA for arrive_cleared time across different incident types
groups = police_incident.groupby('incident_type')['arrive_cleared'].apply(lambda x: x.dro
f_stat, p_val = stats.f_oneway(*groups)
print("F-statistic:", f_stat)
print("P-value:", p_val)
```

```
F-statistic: 35.70717979864037
P-value: 8.496045230736917e-50
```

Since the p-value is less than 0.05, we reject the null hypothesis. There is a very strong statistical evidence prooving that at least one incident type has a mean arrival to cleared time different than other incident types.

Chi-Square Test of Independence

Research Question: Is incident type associated with priority level?

Null Hypothesis: Incident type and priority level are independent. There is no association between incident type and priority level.

Alternative Hypothesis: Incident type and priority level are not independent. There is an association between incident type and priority level.

```
import scipy.stats as stats
import pandas as pd
```

```
# Crosstab of incident type and priority
ct = pd.crosstab(police_incident['incident_type'], police_incident['priority'])

# Chi-square test
chi2, p_val, dof, expected = stats.chi2_contingency(ct)
print("Chi-square statistic:", chi2)
print("Degrees of freedom:", dof)
print("P-value:", p_val)
```

```
Chi-square statistic: 12050.397407578494
Degrees of freedom: 28
P-value: 0.0
```

Since the p-value of the Chi-Square test above is less than 0.05, we will reject the null hypothesis. The is strong evidence proving that incident type and priority level are not independent. There is an association between incident type and priority level.

Data Visualization and Storytelling

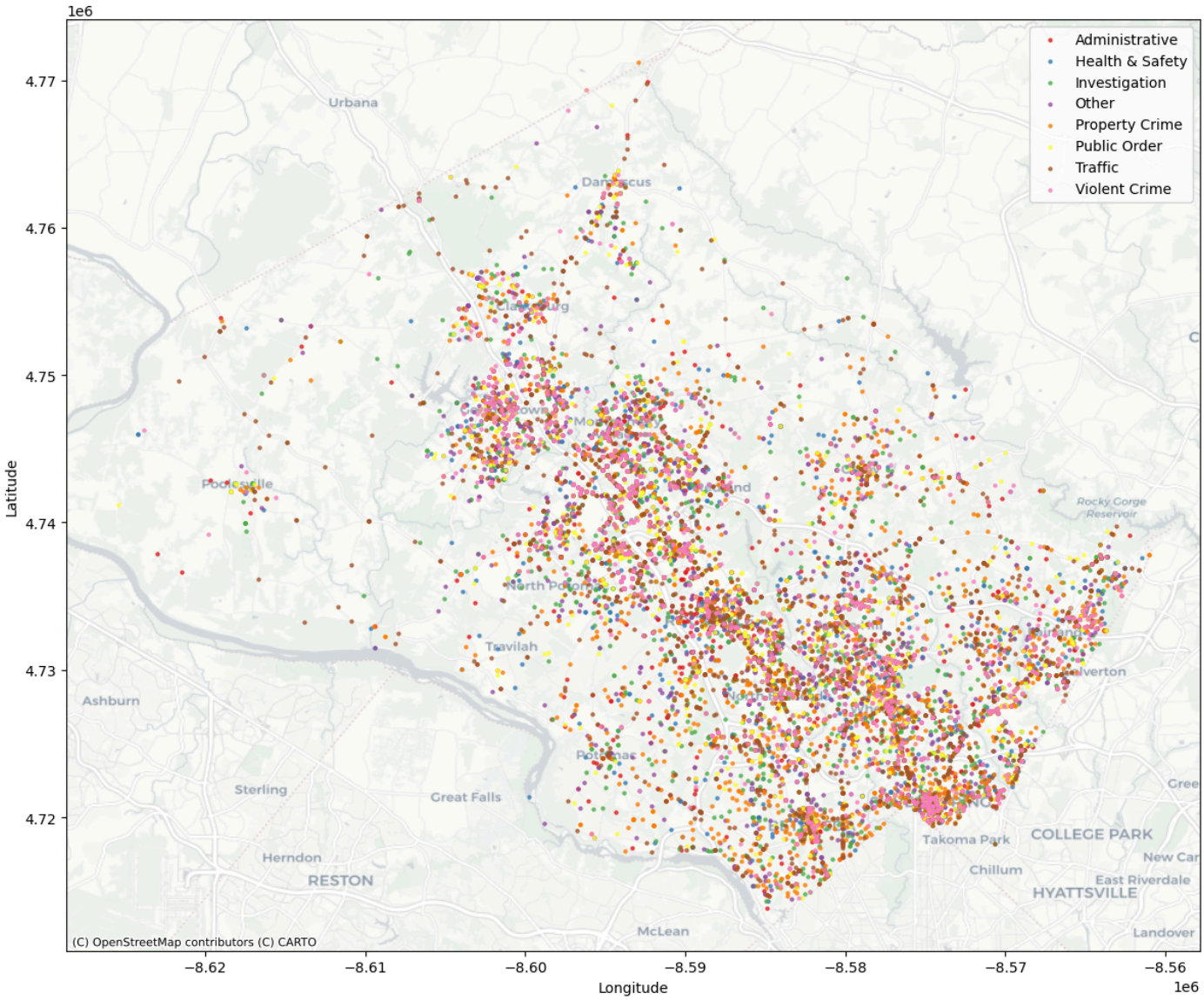
In the following section, I will present several key insights using charts and visualizations, primarily with the Matplotlib and Geopandas packages. I will include interesting graphs that are not in the above sections, and unrelated to machine learning, just for a fun interpretation of the data.

Incident Location by Incident Type

The following visual is a map of Montgomery County that plots each incident occurrence location on top of its true location and distinguishes them by incident type. The following Python codes are partially generated by ChatGPT 5.1[1].

```
import geopandas as gpd
import matplotlib.pyplot as plt
import contextily as ctx
import pandas as pd

# Load processed data
police_incident = pd.read_csv("../../data/processed-data/police_incident_eda.csv")

# Define incident types (sorted for consistent color ordering)
incident_types = sorted(police_incident['incident_type'].unique())

colors = [
    "#e41a1c", "#377eb8", "#4daf4a", "#984ea3",
    "#ff7f00", "#ffff33", "#a65628", "#f781bf"
]

type_to_color = {itype: colors[i] for i, itype in enumerate(incident_types)}

# Create GeoDataFrame
gdf = gpd.GeoDataFrame(
```

```
    police_incident,
    geometry=gpd.points_from_xy(police_incident['longitude'], police_incident['latitude']
    crs="EPSG:4326"
).to_crs(epsg=3857)

fig, ax = plt.subplots(figsize=(12,10))
for incident_type in incident_types:
    subset = gdf[gdf['incident_type'] == incident_type]
    ax.scatter(
        subset.geometry.x,
        subset.geometry.y,
        s=5,
        color=type_to_color[incident_type],
        label=incident_type,
        alpha=0.7
    )

ctx.add_basemap(ax, source=ctx.providers.CartoDB.Positron)
plt.legend(loc='upper right')
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.tight_layout()
plt.savefig("../../report/visual12.png", dpi=300, bbox_inches="tight")
plt.show()
```



From the graph above, we can see that more populated areas have a much higher chance of incidents, which makes sense. Of the above areas, Silver Spring has the most incidents. Along with North Bethesda, around Bethesda Row. We can also see that many incidents occurred along Rockville Pike. As a Montgomery County resident, Rockville Pike is like the main vein of Montgomery County. Nearly all commercial areas, including restaurants, movie theaters, and shopping centers, are built this way. So it is not surprising that there are many incidents.

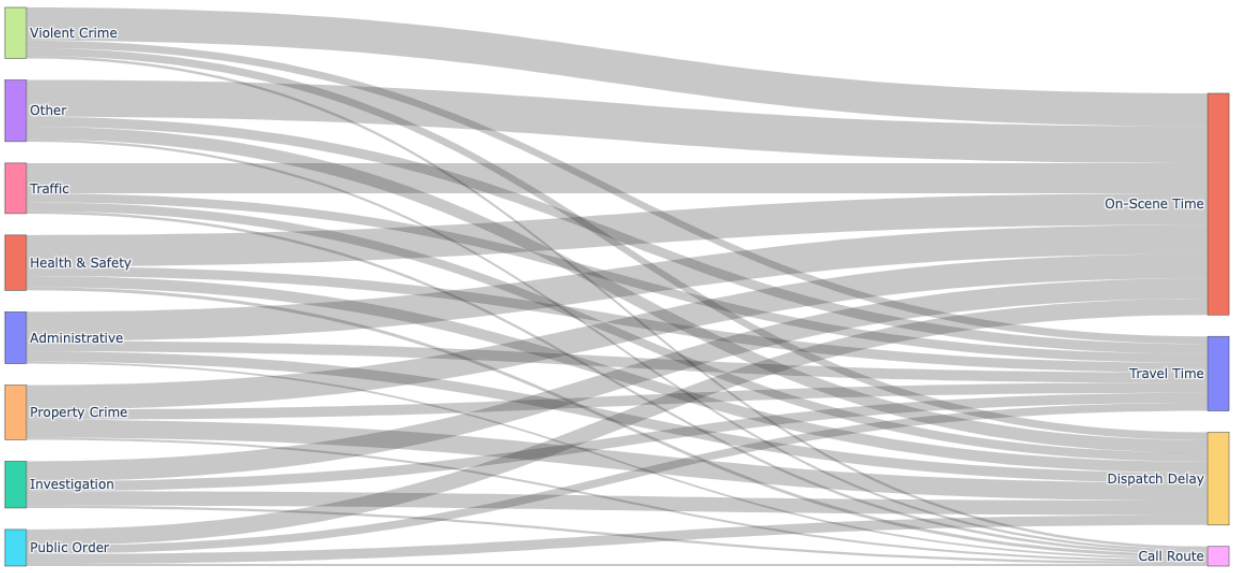
Operational Delay Bottlenecks Sankey Diagram

The following visual shows how time is wasted through the 911 response pipeline, from the initial call to final clearance. It highlights where key operational delays occur and shows which incident types contribute most to bottlenecks. Due to the complexity of the codes, the following codes are partially generated by ChatGPT 5.1[1].

```
import plotly.graph_objects as go
import pandas as pd

incident_types = police_incident['incident_type'].unique()
```

```
avg_times = police_incident.groupby('incident_type')[['calltime_callroute', 'calltime_dis

left_labels = list(avg_times.index)
right_labels = ["Call Route", "Dispatch Delay", "Travel Time", "On-Scene Time"]
labels = left_labels + right_labels

label_to_index = {label: i for i, label in enumerate(labels)}

sources = []
targets = []
values  = []

for inc in avg_times.index:
    for i, stage in enumerate(right_labels):
        sources.append(label_to_index[inc])
        targets.append(label_to_index[stage])
        values.append(avg_times.loc[inc].iloc[i])

fig = go.Figure(data=[go.Sankey(
    node=dict(pad=20, thickness=20, label=labels),
    link=dict(source=sources, target=targets, value=values)
)])

fig.update_layout(font_size=12, height=700, width=1300)
fig.show()
```

```
Unable to display output for mime type(s): application/vnd.plotly.v1+json
```



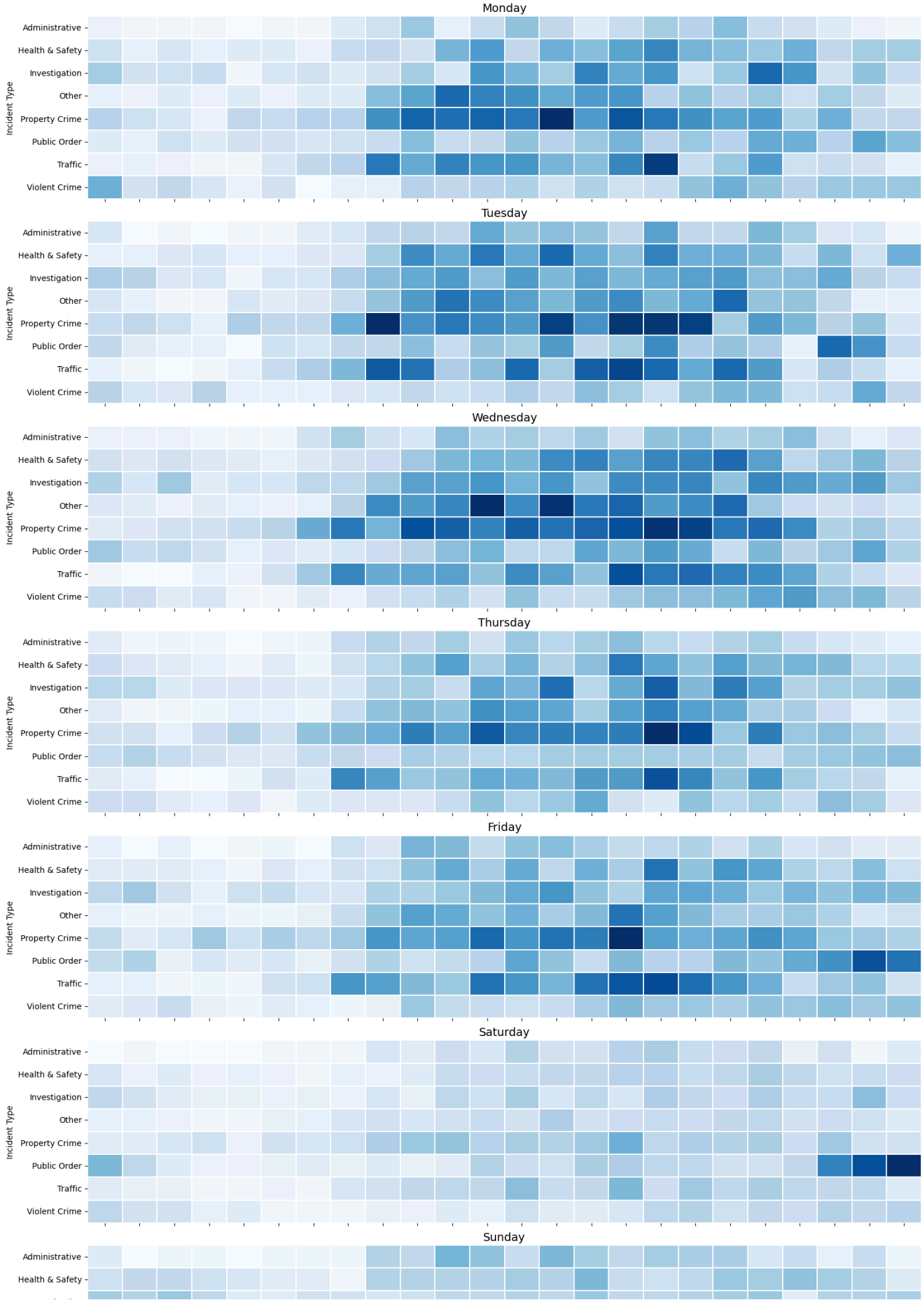
Above plot is shown as image because Plotly version could not be rendered by Quarto due to renderer compatibility issues.

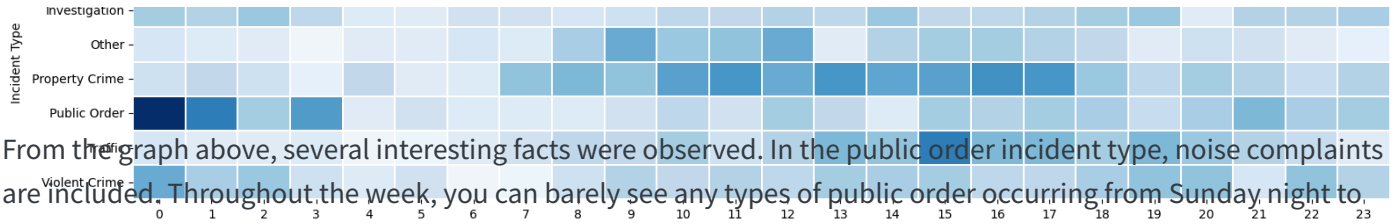
To better understand the graph above, on-scene time is the time it takes the officer to clear the incident, starting from arrival. Travel time is the time it takes from the officer's dispatch to their arrival on scene. Dispatch delay is when the call details are ready, and officers are officially dispatched. The call route is the time it takes 911 to redirect the call to the correct department. From the graph, we can clearly see that the true bottleneck in this whole system is the dispatch-to-delay time. Besides, on-scene time is the longest, which makes sense. Dispatch delay is taking the longest, even longer than travel time.

Incident Type Frequency by Hour and Weekday

The following graph is relatively complex, containing three data fields: `weekday`, `hour`, and `incident_type`. The following graphs display the frequency of incidents occurring by incident type and hours, and are split into weekdays. Due to the complexity of the code, part of the following code is generated by ChatGPT 5.1[1].

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

weekday_map = {1:"Monday", 2:"Tuesday", 3:"Wednesday", 4:"Thursday", 5:"Friday", 6:"Satur
police_incident['weekday_name'] = police_incident['weekday'].map(weekday_map)

week_order = ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"]

fig, axes = plt.subplots(7, 1, figsize=(16, 25), sharex=True)

for ax, day in zip(axes, week_order):
    df_day = police_incident[police_incident['weekday_name'] == day]

    ct = pd.crosstab(df_day['incident_type'], df_day['hour'])

    sns.heatmap(
        ct,
        cmap="Blues",
        linewidths=0.1,
        ax=ax,
        cbar=False
    )

    ax.set_title(f"{day}", fontsize=14)
    ax.set_xlabel("")
    ax.set_ylabel("Incident Type")

plt.tight_layout()
plt.savefig("../../report/visual2.png", dpi=300, bbox_inches="tight")
plt.show()
```


Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Incident Type
Administrative
Health & Safety
Investigation
Other
Property Crime
Public Order
Traffic
Violent Crime



From the graph above, several interesting facts were observed. In the public order incident type, noise complaints are included. Throughout the week, you can barely see any types of public order occurring from Sunday night to Thursday night. However, Friday nights and Saturday nights have a whole different story. You can clearly see that the frequency of public order incidents is extremely high, extending into the early AM of the next day. In general, the frequency table shows that all incidents are more likely to occur during the daytime, especially in the afternoon, except for public order incidents. Another interesting point is that traffic incidents are more likely to occur during workdays; they rarely occur on weekends.

Conclusion

In this EDA section, I first conducted univariate analyses of numerical and categorical variables. Then I've conducted bivariate and multivariate analysis, including correlation analysis, crosstabulation tables, and feature pairings. I then visualized the skewness and kurtosis of all variables, and returned to the data cleaning section to resolve skewness by applying log transformations. I then normalized the data using z-score normalization. I also conducted a t-test, a chi-square test of independence, and an ANOVA test. Lastly, I've created three complex graphs to see how this police incident data tells a story. After doing all those visualizations and tests, I now have a better sense of the data, and I will continue to work on my data in the following supervised and unsupervised learning sections.

Challenges

One of the major challenges was that I originally had only the `start_time` and `end_time` fields, which were time-related. Although these specific columns contain the original exact time. But as I move forward in the EDA section, I realized it's kinda hard to process this data as categorical; for example, if I wanted to analyze by day of week or time of day. I had to revisit the data cleaning section halfway through. It is very true that about 70% of the time in a data workflow is spent on data cleaning.

References

1. ChatGPT, version-5.1, OpenAI, NOV-2025, chat.openai.com.

DSAN 5000 - Incident Response Modeling (MCMD)

Unsupervised Learning

Overview

In this section, I will perform several unsupervised learning techniques to explore whether distinct clusters of incident types and response times exist in the police-incident data. The unsupervised learning techniques include dimensionality reduction methods such as PCA and t-SNE, as well as clustering methods such as K-Means, DBSCAN, and hierarchical clustering. After completing the PCA and t-SNE, I will evaluate the effectiveness of these methods and compare their visualization capabilities. In the clustering methods section, I will also discuss the results of each clustering analysis and compare them.

Methods and Codes

Dimensionality Reduction

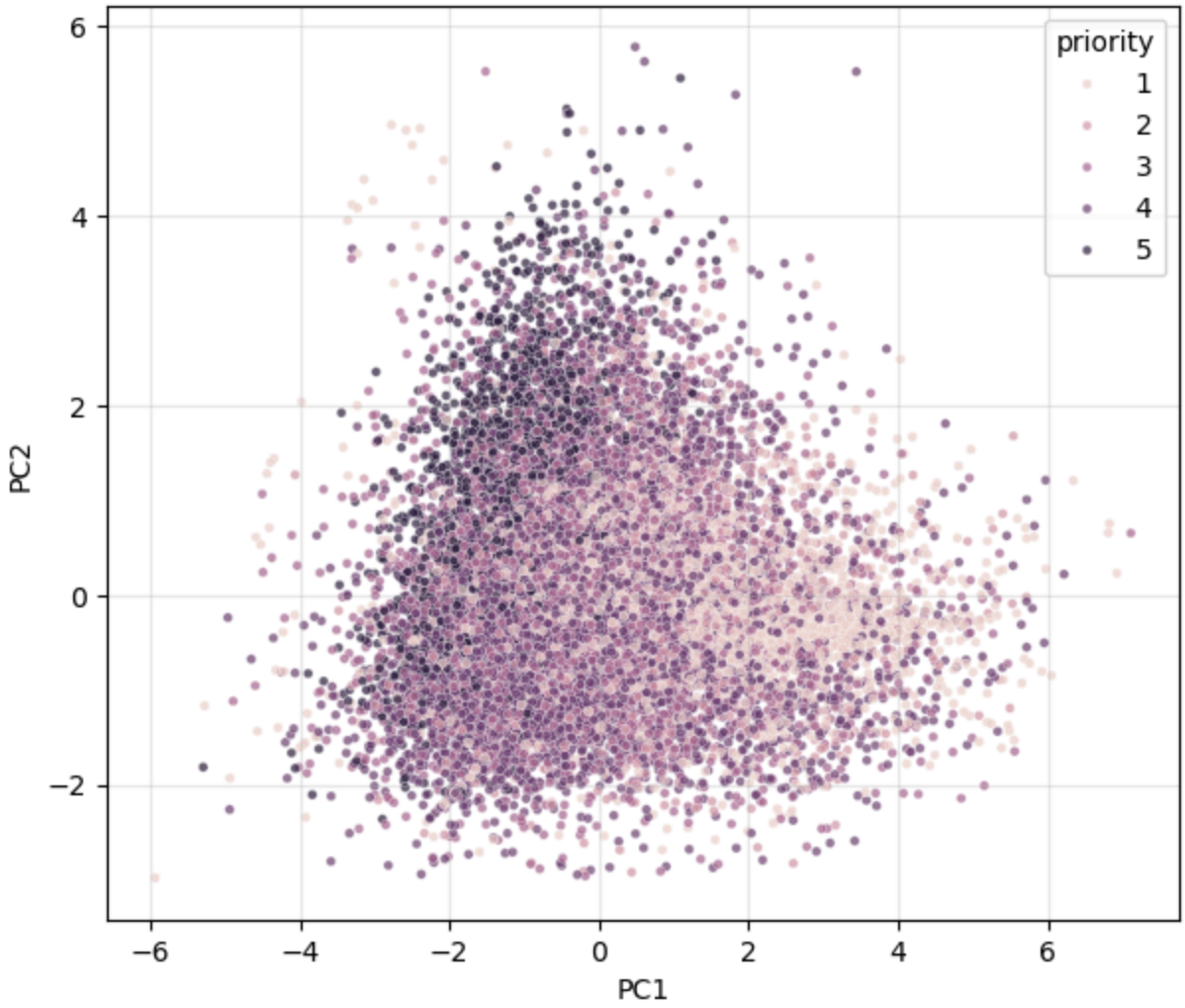
The objective of this section is to explore and demonstrate the effectiveness of PCA and t-SNE in reducing the dimensionality of complex data while preserving essential information and improving visualization.

PCA (Principal Component Analysis)

> **Principal Component Analysis (PCA)**[1]
>
> **Principal Component Analysis (PCA)** is a dimensionality-reduction method that transforms high-dimensional data into a smaller set of uncorrelated variables called principal components. PC1 captures the most variance, PC2 captures the next most while remaining orthogonal, and so on. By focusing on directions of greatest variation, PCA simplifies complex data and makes patterns easier to visualize.

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Load the processed police incident data
police_incident = pd.read_csv("../../data/processed-data/police_incident_ml.csv")

# Define features for PCA
cols = ['longitude', 'latitude', 'calltime_callroute', 'calltime_dispatch', 'callroute_di
X = police_incident[cols].values
```

```
# Perform PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Create a DataFrame for PCA results
pca_df = pd.DataFrame({
    'PC1': X_pca[:, 0],
    'PC2': X_pca[:, 1],
    'priority': police_incident['priority']
})

# Plot the PCA results
plt.figure(figsize=(7,6))
sns.scatterplot(
    data=pca_df,
    x='PC1', y='PC2',
    hue='priority',
    s=12,
    alpha=0.7
)
plt.grid(alpha=0.3)
plt.savefig("../../report/visual10.png", dpi=300, bbox_inches="tight")
plt.show()
```



The PCA results above show that the features form a smooth cloud of points, indicating that police dispatch behavior varies gradually rather than forming distinct linear clusters. To better understand the features, I embedded the colors by priority. I discovered that higher-priority calls tend to shift towards one side of PC1, while lower-priority calls shift to the opposite side. This might suggest that priority level is associated with systematic differences in dispatch timing, even though the categories overlap. As a result, we can conclude that dispatch operations exhibit a continuous, correlated structure without sharp boundaries, making them an effective tool for summarizing global variance but not for identifying discrete groups.

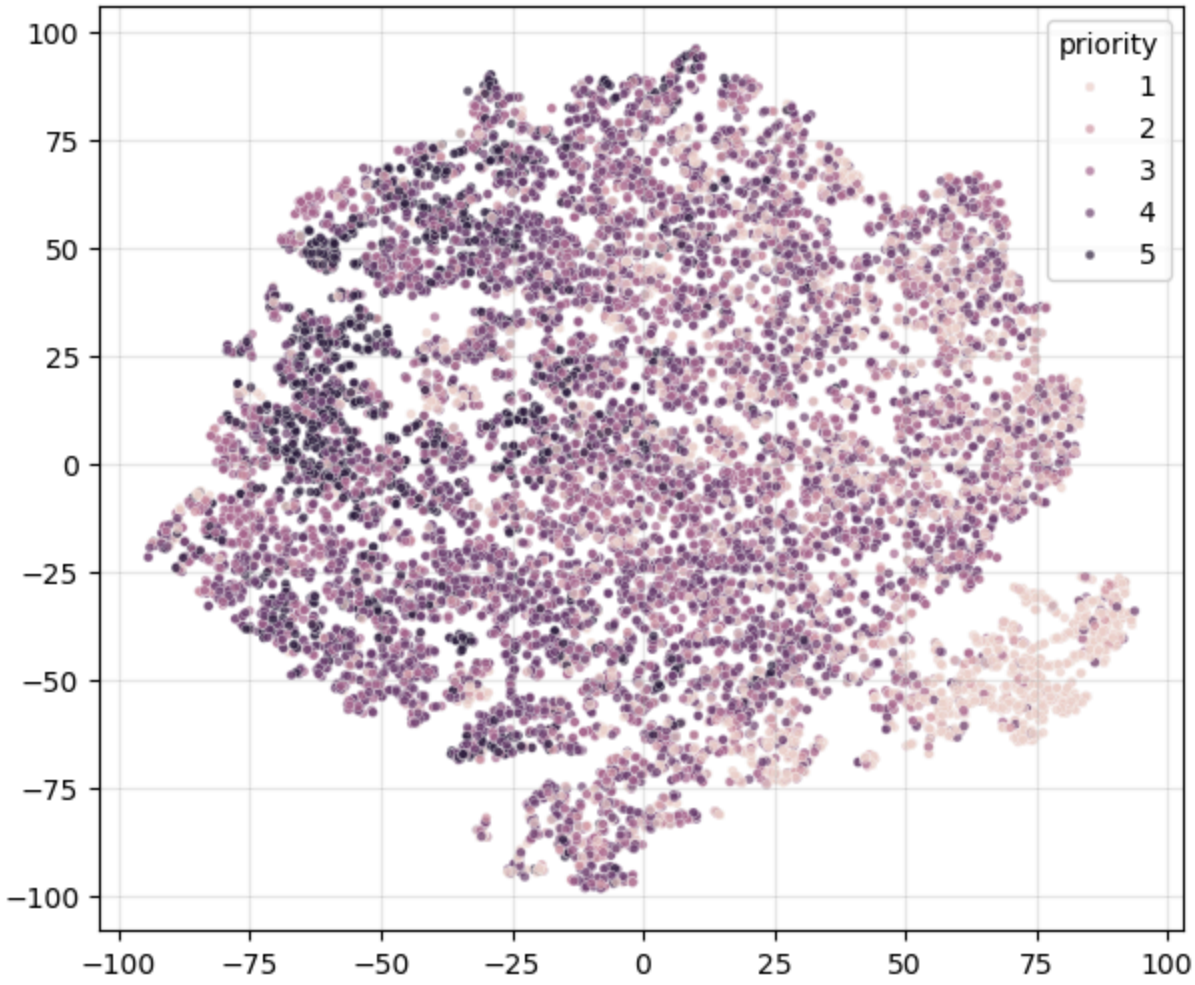
t-SNE (t-distributed Stochastic Neighbor Embedding)

t-Distributed Stochastic Neighbor Embedding (t-SNE)[1]

t-Distributed Stochastic Neighbor Embedding (t-SNE) t-SNE is a nonlinear dimensionality-reduction method that maps high-dimensional data into a low-dimensional space by preserving local neighborhood structure. It converts distances between points into probabilities and finds an embedding that keeps similar points close together while pushing dissimilar points apart, making it especially useful for visualizing patterns or clusters.

```
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Standardize the features
X_scaled = StandardScaler().fit_transform(X)

# Perform t-SNE
tsne_res = TSNE(
    n_components=2,
    perplexity=30,
    learning_rate='auto',
    init='pca',
    random_state=42
).fit_transform(X_scaled)

# Plot the t-SNE results
plt.figure(figsize=(7,6))
sns.scatterplot(
    x=tsne_res[:, 0],
    y=tsne_res[:, 1],
    hue=police_incident['priority'],
    s=12,
    alpha=0.7
)
plt.grid(alpha=0.3)
plt.savefig("../../report/visual8.png", dpi=300, bbox_inches="tight")
plt.show()
```



Evaluation and Comparison

As I've mentioned, the PCA provides a basic overview of how the data is distributed. It shows a general pattern and how the main features relate to each other. In the case above, the PCA shows one large cloud with slight differences based on priority, but nothing that forms clear groups. The t-SNE takes a different approach, aiming to place similar points closer together. After testing perplexity values from 5 to 50, the results are similar to those of PCA. The t-SNE formed a large, diffuse cloud with no strong clusters. However, the t-SNE shows a small shift, with different priority levels appearing in different areas, but the overlap is still very large. In comparison, PCA is easier to understand and faster to run, while t-SNE is more detailed but does not reveal anything very different for this dataset. Both methods suggest that the data changes gradually and does not break into clear groups.

Clustering Methods

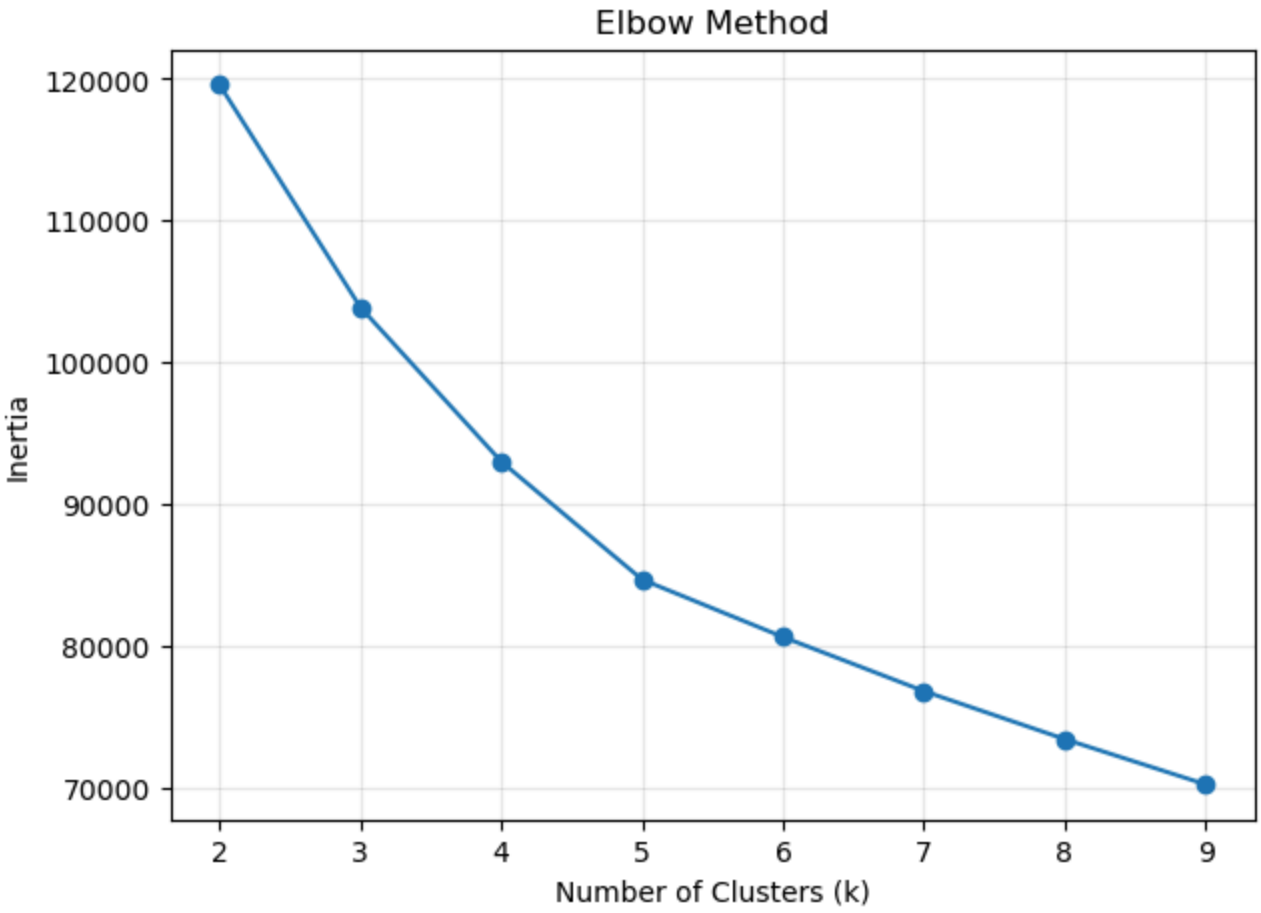
The objective of this section is to apply clustering techniques, including K-Means, DBSCAN, and Hierarchical clustering, to the police incident calls dataset. I will explain how each method works and compare their performance.

K-Means

K-Means[1]

K-Means is a clustering method that groups data into *k* clusters based on similarity. It works by placing *k* centers in the data space, assigning each point to the nearest center, and then adjusting the centers until the assignments stop changing. K-Means is simple, fast, and works best when clusters are roughly round and similar in size.

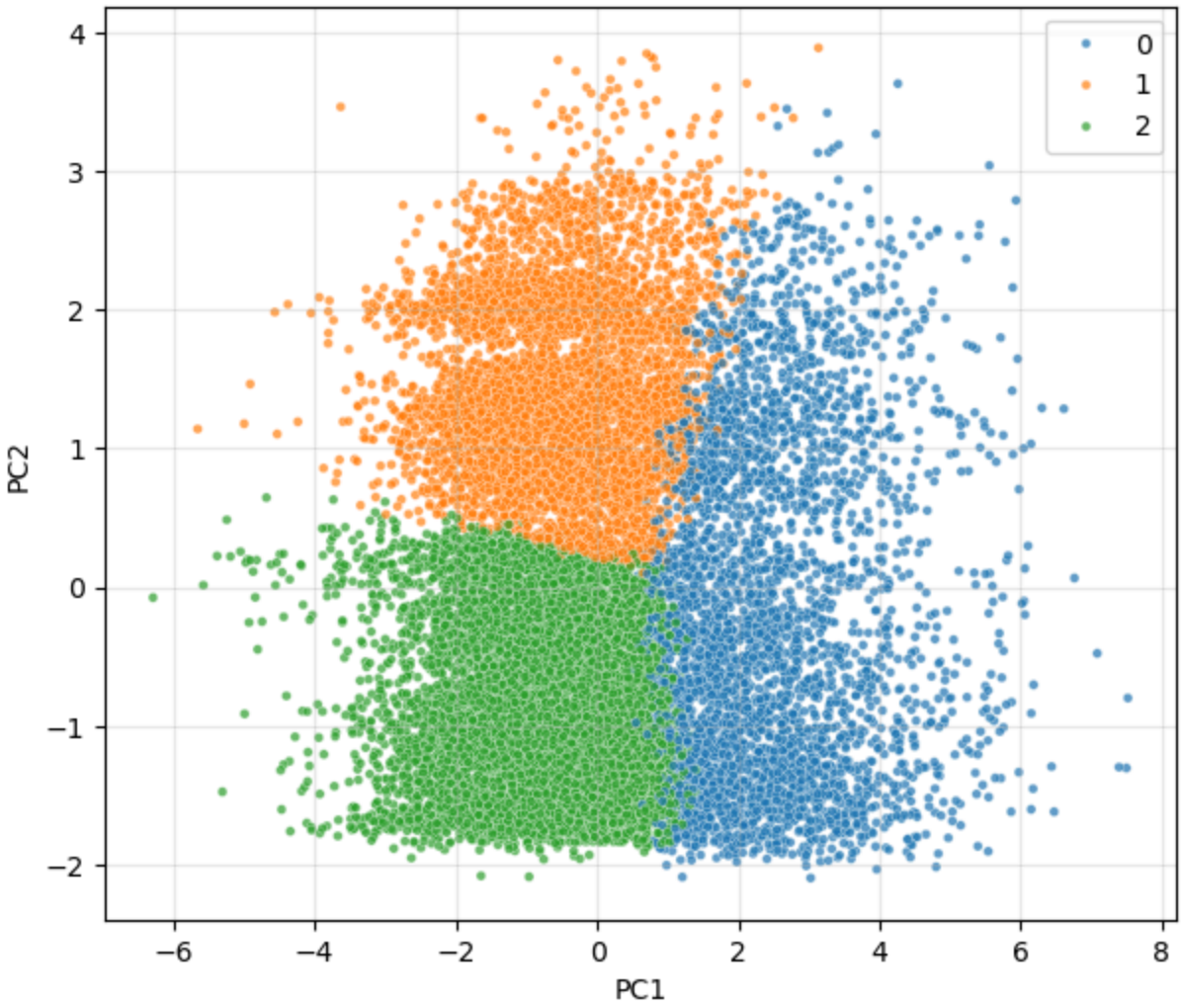
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Determine the optimal number of clusters using the Elbow Method
inertia_values = []
k_values = range(2, 10)

for k in k_values:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    inertia_values.append(km.inertia_)

# Plot the Elbow Method results
plt.figure(figsize=(7,5))
plt.plot(k_values, inertia_values, marker='o')
plt.title("Elbow Method")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.grid(alpha=0.3)
plt.show()
```



The Elbow plot above shows how inertia decreases as more clusters are added. The inertia dropped by over 50% from k = 2 to k = 4. After k=4, the decrease slows slightly. In this case, no elbow turning point is observed, which indicates that the data does not split into clear clusters. In this case, adding more clusters will only reduce inertia slightly because the data forms a single and smooth shape. Since there is no clear best value of k, I will be using k = 3 in the following k-means clustering to show the overall pattern.

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans_labels = kmeans.fit_predict(X_scaled)

# Plot K-Means clustering results
plt.figure(figsize=(7,6))
sns.scatterplot(
    x=X_pca[:,0], y=X_pca[:,1],
    hue=kmeans_labels, palette="tab10",
    s=12, alpha=0.7
)
plt.xlabel("PC1")
plt.ylabel("PC2")
```

```
plt.grid(alpha=0.3)
plt.show()
```



In this clustering method, I used k-means to look for patterns in the police calltime features of the police incident data. Since this data contains spatial information, I decided to use the two PCA dimensions to create my k-means analysis, PC1 and PC2. I first scaled all features using StandardScaler, which, in theory, makes the patterns easier to see.

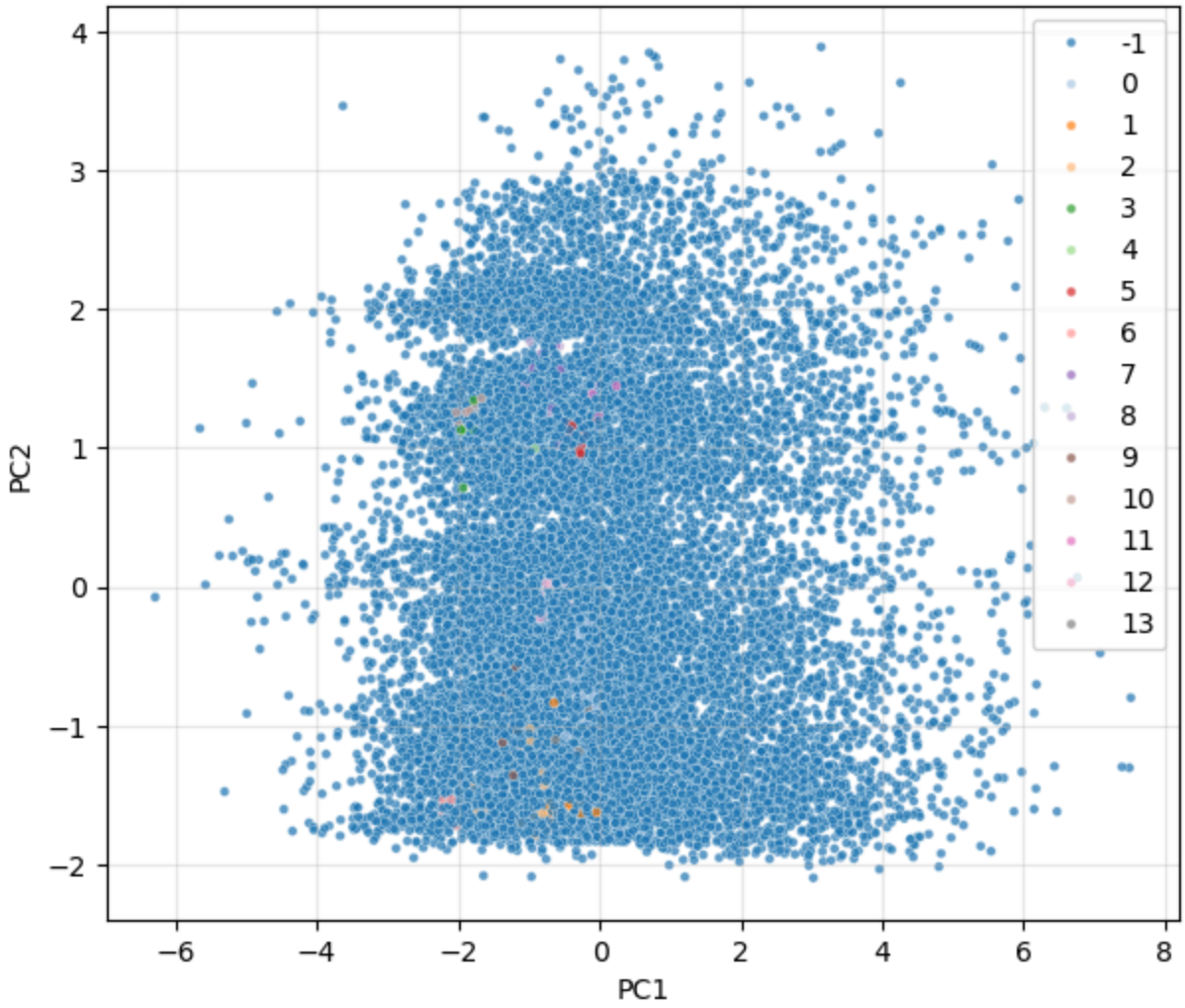
Once the data were scaled and reduced, I ran K-Means with k = 3, based on the previous elbow plot. The above k-means plot shows how the data cloud splits into three main groups, labeled 0, 1, and 2. Since the three groups overlap significantly and there are no clear boundaries, we were unable to define any clusters. Although no clusters were found, the k-means analysis still helps us summarize the general behavior of the timing features, suggesting that police response times change smoothly from one incident to another.

DBSCAN

DBSCAN[1]

> **DBSCAN** is a clustering method that groups points based on how close they are to many neighbors. It forms clusters from dense areas in the data and marks isolated points as noise. Because it looks for dense regions, DBSCAN can find clusters of many different shapes without needing to choose the number of clusters ahead of time.

```
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import seaborn as sns

# Apply DBSCAN clustering
db = DBSCAN(eps=0.5, min_samples=10)
db_labels = db.fit_predict(X_scaled)

# Plot DBSCAN clustering results
plt.figure(figsize=(7,6))
sns.scatterplot(
    x=X_pca[:,0], y=X_pca[:,1],
    hue=db_labels,
    palette="tab20",
    s=12, alpha=0.7
)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(alpha=0.3)
plt.show()
```



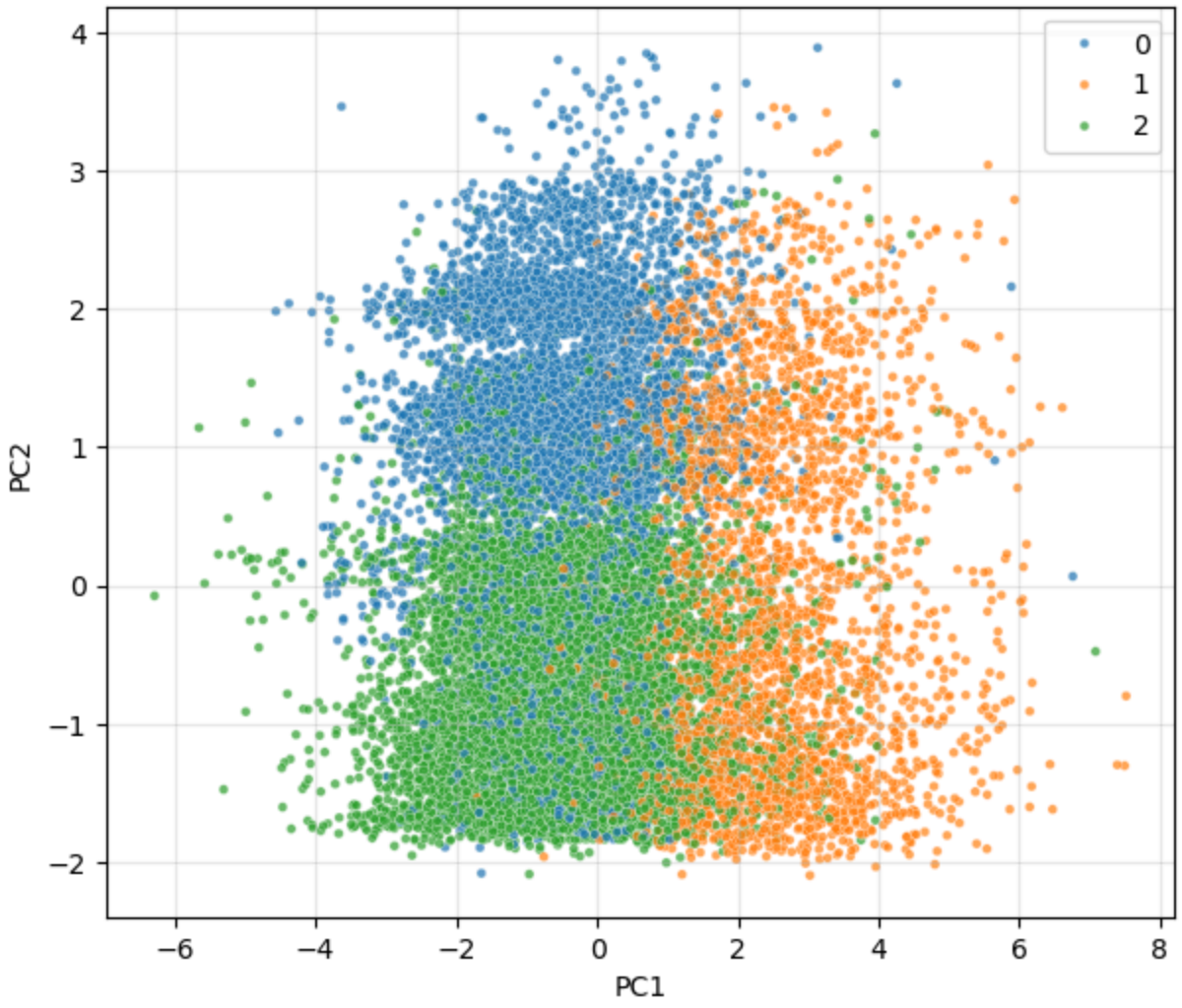
As I mentioned in the definition of DBSCAN above, it is a clustering method that identifies areas where points are densely packed. Instead of using distance to a center, it checks how many neighbors a point has within a set radius. When I ran DBSCAN on the scaled police incident call features, no strong clusters were found. Almost every single point was labeled as having a noise of -1, and the other few clusters that did appear were extremely small. Although no clear clusters were found, this result matches those of PCA and t-SNE, which indicated that the data form a single cloud without clear clusters. Since DBSCAN relies on detecting density differences, it cannot separate the data when the density is nearly uniform everywhere.

Hierarchical Clustering

> **Hierarchical Clustering**[1]
>
> **Hierarchical Clustering** builds clusters step by step by joining the closest points or clusters at each stage. The process creates a tree-like diagram called a dendrogram, which shows how groups form at different levels. This method is useful for exploring structure in the data because you can choose the number of clusters by “cutting” the tree at different heights.

```
from sklearn.cluster import AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Apply Hierarchical Clustering
hclust = AgglomerativeClustering(n_clusters=3)
h_labels = hclust.fit_predict(X_scaled)

# Plot Hierarchical Clustering results
plt.figure(figsize=(7,6))
sns.scatterplot(
    x=X_pca[:,0], y=X_pca[:,1],
    hue=h_labels,
    palette="tab10",
    s=12, alpha=0.7
)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(alpha=0.3)
plt.show()
```



In the above definition, Hierarchical clustering works by joining the closest points or clusters repeatedly until everything forms a single structure. In this part, I used the agglomerative clustering method, meaning that each incident starts as its own cluster. It then merges the most similar clusters in steps. When I plotted the results using the two PCA dimensions, the clusters looked very similar to the K-Means output. The data formed a single large

cloud, with overlap among all three groups. This suggests that the data lacks a strong hierarchical structure that would yield clear and separate branches. Instead, the method divides the data based on small, gradual changes along the PCA. In the end, hierarchical clustering reaches the same conclusion as the methods above. The police incident call features shift smoothly across incidents, so the dataset lacks well-defined clusters.

Evaluation and Comparison

Across all three methods above, the results point to the same conclusion. The data form a single, continuous cloud with no breaks or natural groupings between. K-Means produced broad clusters that overlap heavily. DBSCAN finds almost no dense regions, marking most points as noise. Hierarchical clustering also produced groups with no clear boundaries. Together, these results indicate that the police incident call features change gradually across incidents and do not form clusters.

Conclusion

Both the graphs of dimensionality reduction methods and the clustering methods point to the same result. The police incident call data forms a single large cloud with no cluster boundaries. Dimensionality reduction methods, PCA and t-SNE, show gradual changes rather than sharp groups. Clustering methods such as k-means produce heavily overlapping clusters, DBSCAN finds no dense regions, and hierarchical clustering forms groups that blend together.

Challenges

One of the challenges I faced in this step was that my data showed basically no clusters. All methods, including PCA, t-SNE, k-means, DBSCAN, and hierarchical clustering, failed to produce clear clusters. Throughout the semester, when completing homework or lab assignments, we've always used data that show clear clusters. So, during my project, I thought I had done something wrong. Is it because of data cleaning, a problem with my data transformation, or a problem with z-score normalization? As a result, it was really time-consuming for me to verify my data over and over again, and by conducting research, I am confident in my data.

References

1. ChatGPT, version-5.1, OpenAI, NOV-2025, chat.openai.com.

DSAN 5000 - Incident Response Modeling (MCMD)

Supervised Learning

Overview

In the supervised learning stage, we will first preprocess the raw data to prepare it for training. As this step has already been completed in the data cleaning stage, please refer to that stage for details. Techniques used include z-score normalization, log transformation, and categorical variable encoding, among others. I will use a binary classification method to predict whether an incident is high or low priority, a multiclass classification method to predict the final disposition of each call, and regression models to forecast dispatch-to-arrival times based on the prepared police-incident dataset.

Methods and Codes

Data Preprocessing

In the data processing step, I will perform normalization and standardization, encode categorical variables, apply log and z-score transformations, and select relevant features to ensure the data is suitable for supervised learning. For details, please see the Codes and Method (Machine Learning) section.

Binary Classification

Model Rationale

For binary classification, I decided to predict whether a police incident is high-priority, with priority levels of 4 or 5, or low-priority, with priority levels of 1, 2, or 3. This would allow us to look at how emergency dispatch systems often distinguish between routine calls and those that require an immediate response. This analysis can reveal whether the underlying call characteristics and call times align with the urgency level assigned by the control centers. It can also help identify potential mismatches between actual call times and assigned priority levels.

Overview of Algorithms

For this task, I've decided to use logistic regression to model the binary priority classification. Logistic regression works by combining the input features into a simple linear combination. This combination then passes through a sigmoid function, which turns it into a probability between 0 and 1. The model learns by minimizing a loss that compares its predicted probabilities to the true labels. This causes the model to adjust until the predictions stabilize. With this method, I can see which features push an incident towards high priority. This also scales well to my current police incident call dataset and gives stable probability estimates that I can evaluate using ROC curves.

Binary Logistic Regression

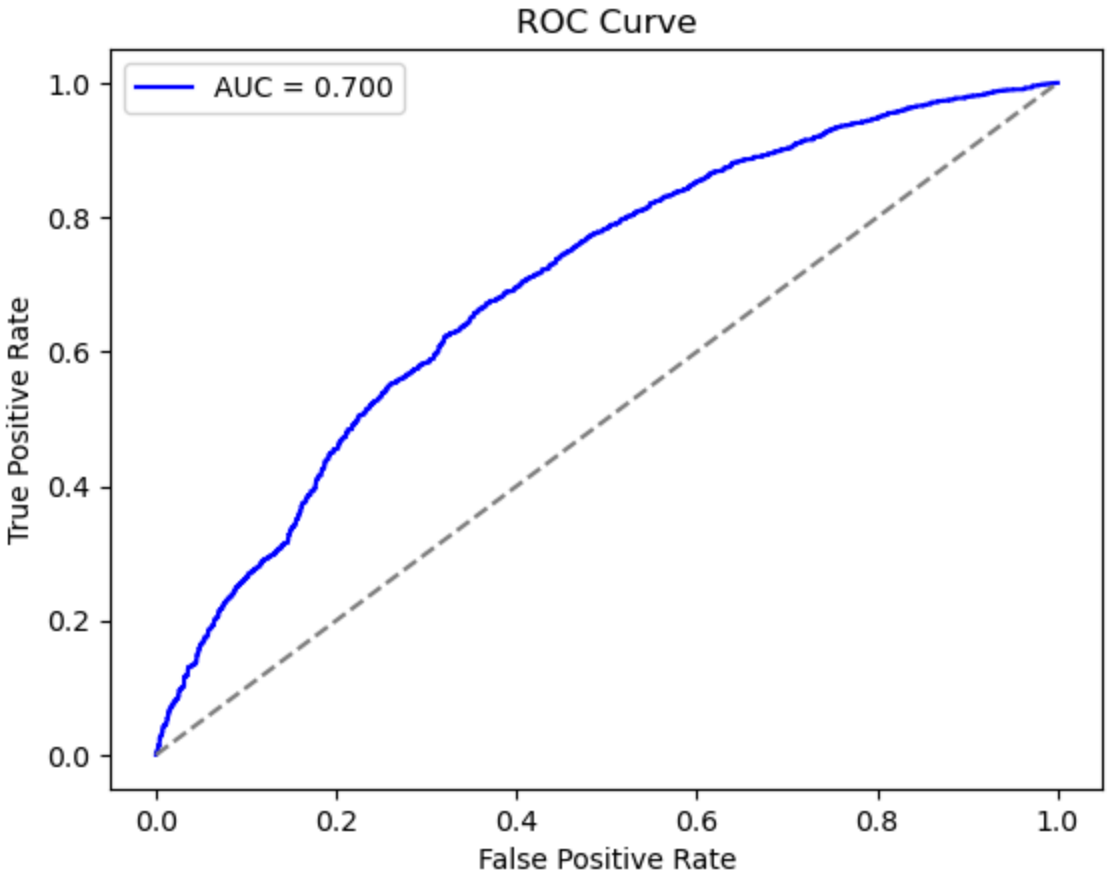
Binary Logistic Regression[1]

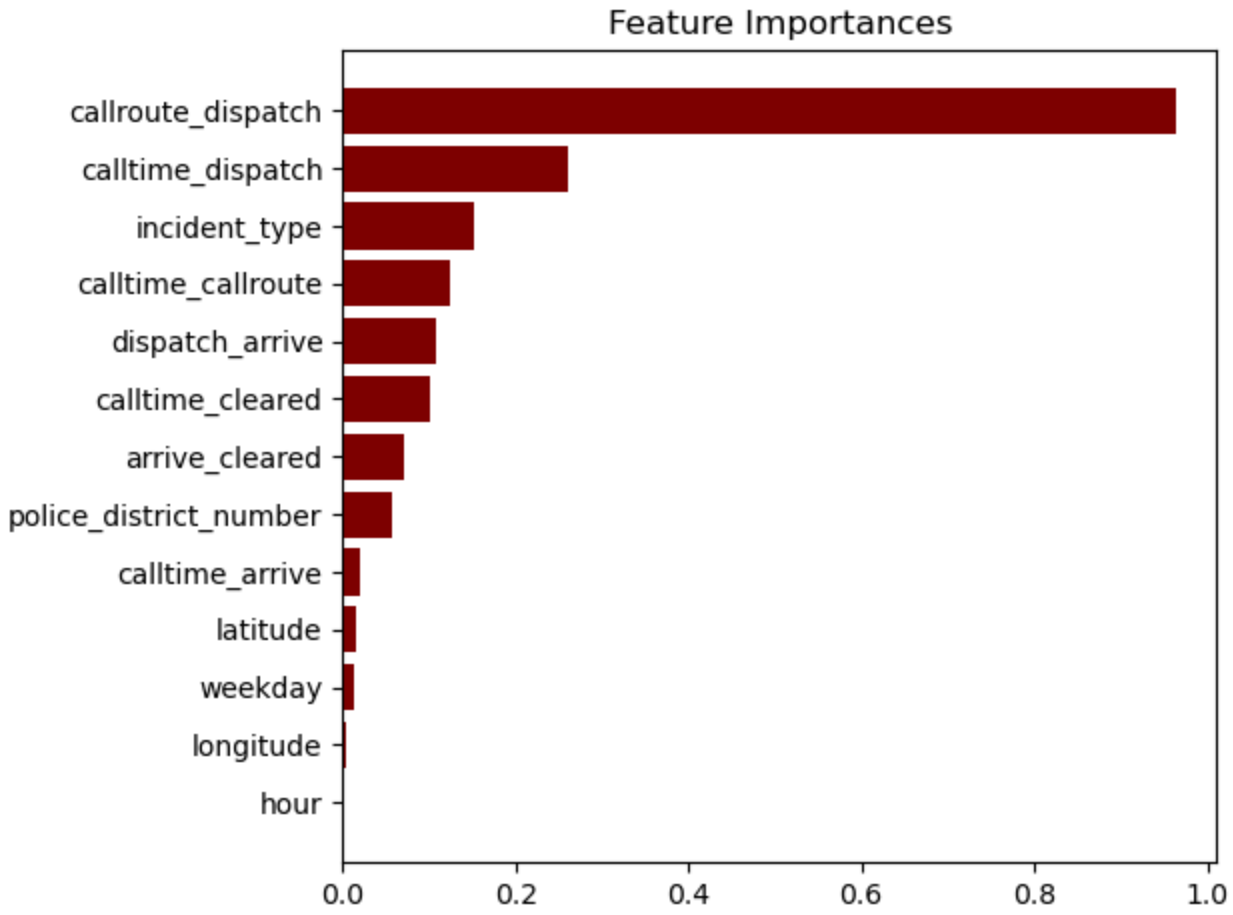
Binary Logistic Regression is a generalized linear classification method used when the target variable has two possible outcomes. The model learns a weighted combination of the input features and converts that result into a probability between 0 and 1 using the logistic transformation. These probabilities describe how likely an observation belongs to the positive class. The model parameters are estimated by finding the values that best match the observed labels, typically by maximizing the likelihood of the data. Once trained, the model assigns a class by applying a probability threshold, making logistic regression both interpretable and effective for many real-world binary prediction tasks, including your priority classification problem.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score, roc
import numpy as np

# Import police incident data
df = pd.read_csv("../../data/processed-data/police_incident_ml.csv")

# Modify priority to binary classification
df["priority_binary"] = (df["priority"] >= 4).astype(int)

# Remove unnecessary columns
X = df.drop(columns=["priority_binary", "priority"])
y = df["priority_binary"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42

# Train logistic regression model
model = LogisticRegression(max_iter=5000)
model.fit(X_train, y_train)

# Evaluate model
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))

# Plot ROC curve
fpr, tpr, _ = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}", color="blue")
```

```
plt.plot([0,1], [0,1], linestyle="--", color="gray")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.show()

# Plot feature importances
coef = model.coef_[0]
feature_names = X.columns

top_idx = np.argsort(np.abs(coef))[-15:]

plt.barh(feature_names[top_idx], np.abs(coef[top_idx]), color="maroon")
plt.title("Feature Importances")
plt.tight_layout()
plt.show()
```

```
Accuracy: 0.6477949940405244
Precision: 0.6251402918069585
Recall: 0.6842751842751843
F1 Score: 0.6533724340175953
```





Model Performance & Insights

From the results, we are able to tell that the binary logistic regression performed pretty well, since the accuracy, precision, recall, and F1 score all falls in the same range. This indicates that the model is producing balanced decisions, meaning that it is not favoring one class. From the results, the recall score is slightly higher than the precision score, indicating that the model is better at identifying high-priority calls. From the ROC curve below, we can see that the AUC is about 0.70, meaning that the model can separate high and low priority calls better than chance and captures real patterns in the data. Although the separation is not perfect, it is sufficient for early screening. Also, the predicted probabilities show a steady improvement over random guessing. In the graph below, the feature importance plot highlights which inputs influence the model the most, with the strongest being `callroute_distach`, followed by `calltime_dispatch`, `incident_type`, `calltime_callroute`, `dispatch_arrive`, `calltime_cleared`, `arrive_cleared`, `police_district_number`, `calltime_arrive`, `latitude`, `longitude`, `weekday`, and `hour`.

Multiclass Classification

Model Rationale

For the multiclass classification problem, I will predict the original police call incident type with the other variables. The result of this would allow us to check how all the other variables in the dataset could separate the eight different categories of police incident call types. I will use a multiclass logistic regression model and a KNN

classifier as a comparison model. With the result of these modes, we will be able to see how much structure actually exists in the incident call type labels and whether the engineered features capture enough signal to distinguish them.

Overview of Algorithms

The multiclass logistic regression uses the softmax function, which transforms a set of raw model scores into probabilities for each class by exponentiating the scores and normalizing so they sum to one. The model picks the class with the highest probability. This gives a simple linear decision surface across all classes. The second model I will use, KNN, looks at the nearest neighbors in the feature space and assigns the class that is most common among them. This makes KNN very sensitive to distances in the data. These two algorithms behave very differently, so using both helped reveal whether linear patterns or local neighborhood patterns existed in the data.

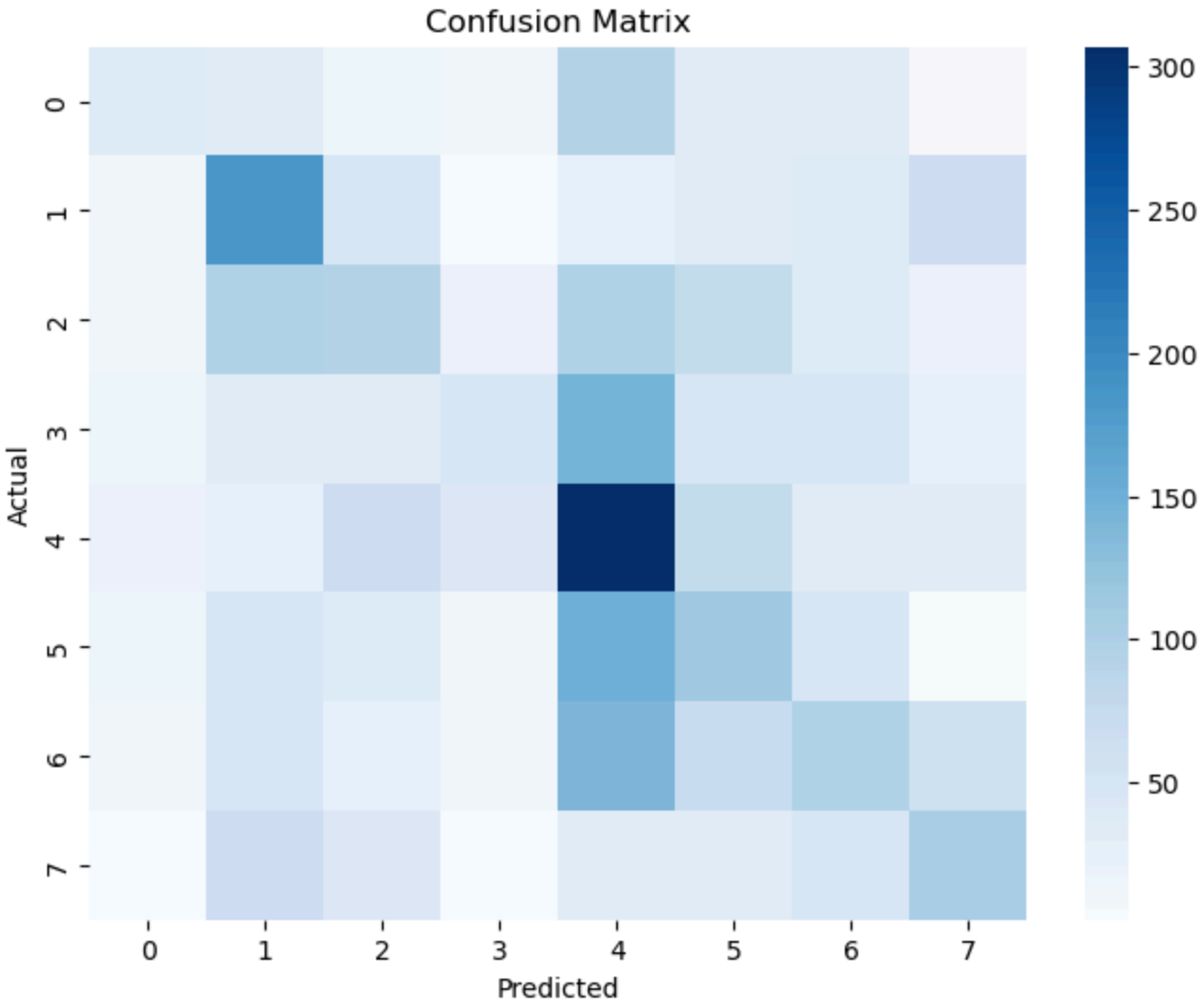
Multiclass Logistic Regression

> **Multiclass Logistic Regression**[1]
>
> **Multiclass Logistic Regression** is a generalized linear classification model used when the response variable contains more than two categories. The model assigns each class a linear score and converts these scores into probabilities using the softmax function, ensuring they sum to one. Parameter estimation is performed by maximizing the multinomial likelihood or minimizing categorical cross-entropy through gradient-based optimization. The predicted class is the one with the highest estimated probability, making this method the direct extension of binary logistic regression to multi-class settings.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load machine learning police incident data
df = pd.read_csv("../../data/processed-data/police_incident_ml.csv")

# Drop unnecessary columns
X = df.drop(columns=["incident_type"])
y = df["incident_type"]

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42

# Multi-class logistic regression
log_model = LogisticRegression(max_iter=5000, multi_class="multinomial")
log_model.fit(X_train, y_train)

y_pred_log = log_model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred_log))
print(classification_report(y_test, y_pred_log))
```

```
# Confusion Matrix
cm_log = confusion_matrix(y_test, y_pred_log)
plt.figure(figsize=(8,6))
sns.heatmap(cm_log, annot=False, cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.savefig("../../report/visual11.png", dpi=300, bbox_inches="tight")
plt.show()
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/linear_model/_logistic.py:1247:
FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7.
From then on, it will always use 'multinomial'. Leave it to its default value to avoid
this warning.
  warnings.warn(

Accuracy: 0.2938021454112038
              precision    recall  f1-score   support

           0       0.34      0.16      0.21       263
           1       0.35      0.45      0.39       409
           2       0.25      0.21      0.23       451
           3       0.33      0.12      0.18       400
           4       0.31      0.51      0.38       604
           5       0.23      0.26      0.24       426
           6       0.26      0.21      0.23       468
           7       0.32      0.31      0.31       335

    accuracy                           0.29      3356
   macro avg       0.30      0.28      0.27      3356
weighted avg       0.29      0.29      0.28      3356
```



K-Nearest Neighbors (KNN)

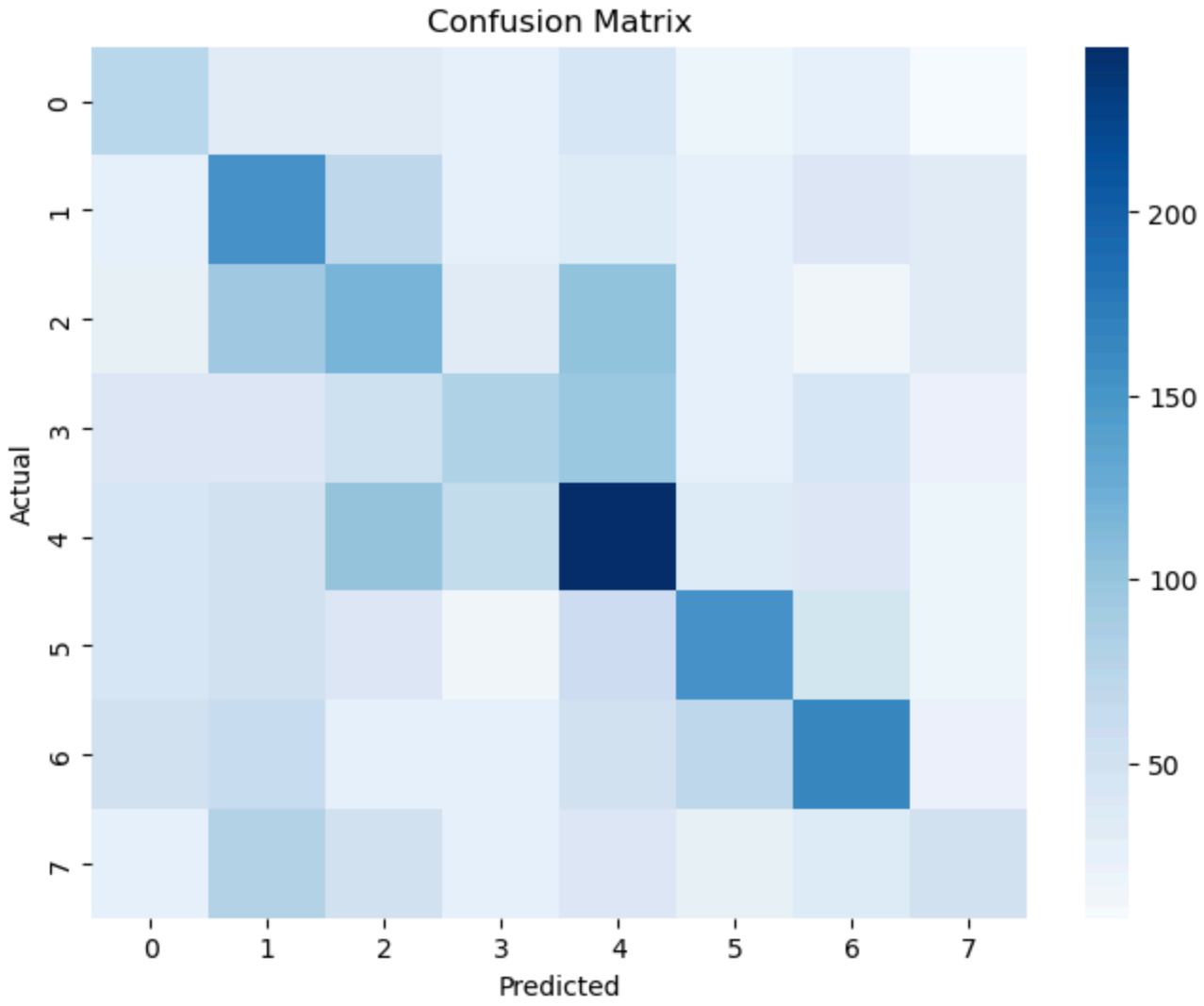
K-Nearest Neighbors (KNN)[1]

K-Nearest Neighbors is a non-parametric classification method that predicts a label based on the classes of the closest observations in the feature space. Instead of learning model parameters, KNN stores the training data and makes decisions by measuring the distance between a new point and existing points. The model identifies the k most similar neighbors and assigns the class most common among them. Because it relies directly on distances, KNN is sensitive to feature scaling and works best when similar cases naturally cluster together in the data.

```
# K-Nearest Neighbors Classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
```

```
# Confusion Matrix for KNN
cm_knn = confusion_matrix(y_test, y_pred_knn)
plt.figure(figsize=(8,6))
sns.heatmap(cm_knn, annot=False, cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Accuracy: 0.31078665077473183
              precision    recall  f1-score   support

           0       0.23      0.29      0.25       263
           1       0.28      0.37      0.32       409
           2       0.24      0.26      0.25       451
           3       0.28      0.20      0.23       400
           4       0.36      0.41      0.38       604
           5       0.39      0.36      0.38       426
           6       0.40      0.35      0.38       468
           7       0.25      0.15      0.19       335

    accuracy                           0.31      3356
   macro avg       0.30      0.30      0.30      3356
weighted avg       0.31      0.31      0.31      3356
```



Model Performance & Insights

In the multiclass classification models, both the multiclass logistic regression and the KNN model struggle with predicting police incident types, which is consistent with previous unsupervised learning results from PCA and clustering. The result was that the incident types overlap heavily, meaning that the numerical features do not have a clean boundary between different classes or police incident types. Logistic regression reached an accuracy of about 29%, and the KNN achieved an accuracy rate of 31%. Although the accuracy is low, these values still exceeded the accuracy of random guessing, but it is still low for eight classes. Both the macro and weighted F1 scores is around the 0.18–0.4 range for both models, indicating that both models did not capture strong class-specific patterns, which can also be visualized in the confusion matrices.

Regression

Model Rationale

For this part, I will focus on predicting the dispatch-to-arrival time for each incident using a linear regression. This variable, dispatch_arrive, reflects how long it takes for police officers to reach the scene after they are dispatched, which is the most important operational measure in police emergency response. Modeling this outcome could

help reveal whether features such as location or incident type contain enough signal to explain variations in how quickly units arrive.

Overview of Algorithms

As I mentioned above, I will be using a standard linear regression model for the following analysis. Linear regression attempts to uncover straight-line relationships between the predictors and the continuous target. The model estimates a set of coefficients that minimize the overall prediction error, making it one of the simplest and most interpretable methods for low-dimensional regression and optimization.

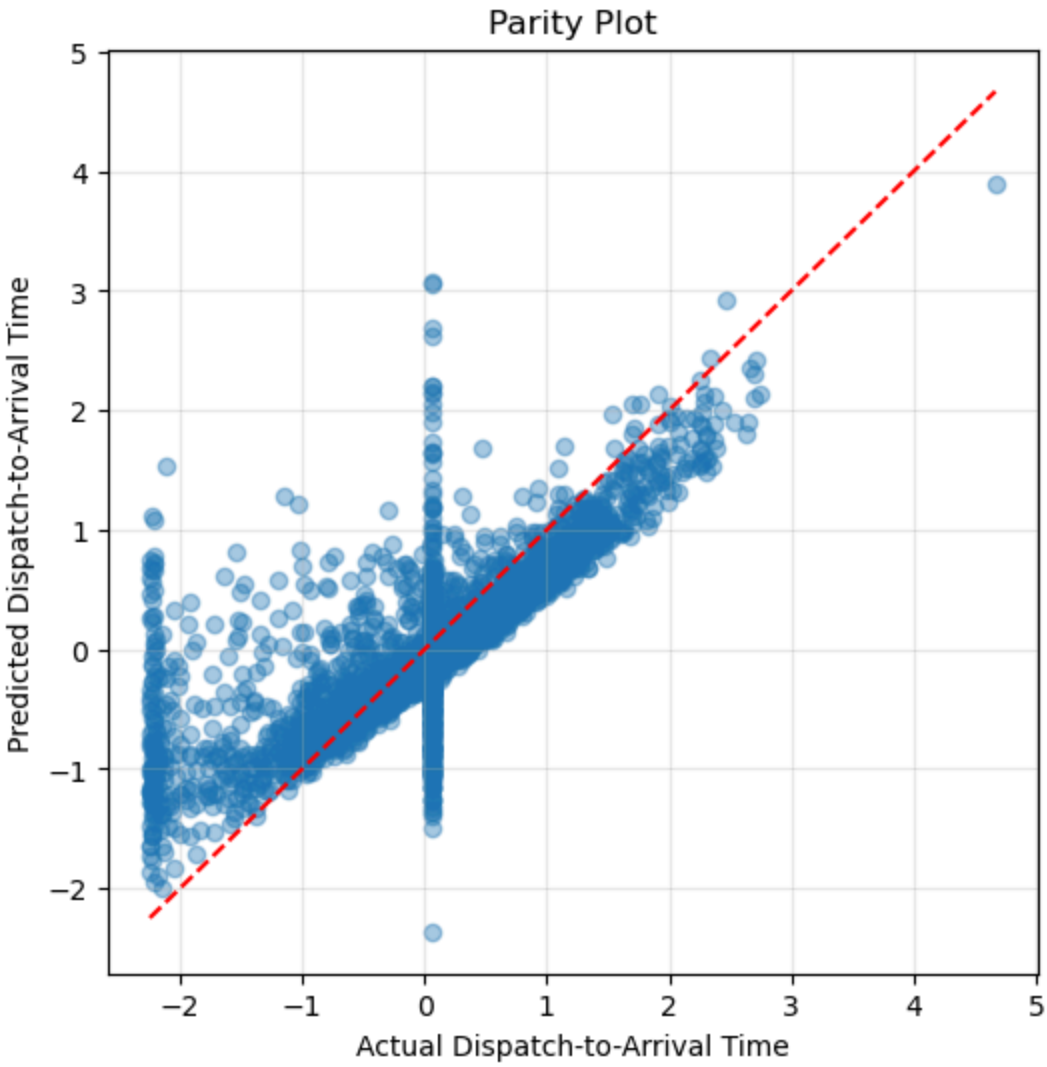
Linear Regression

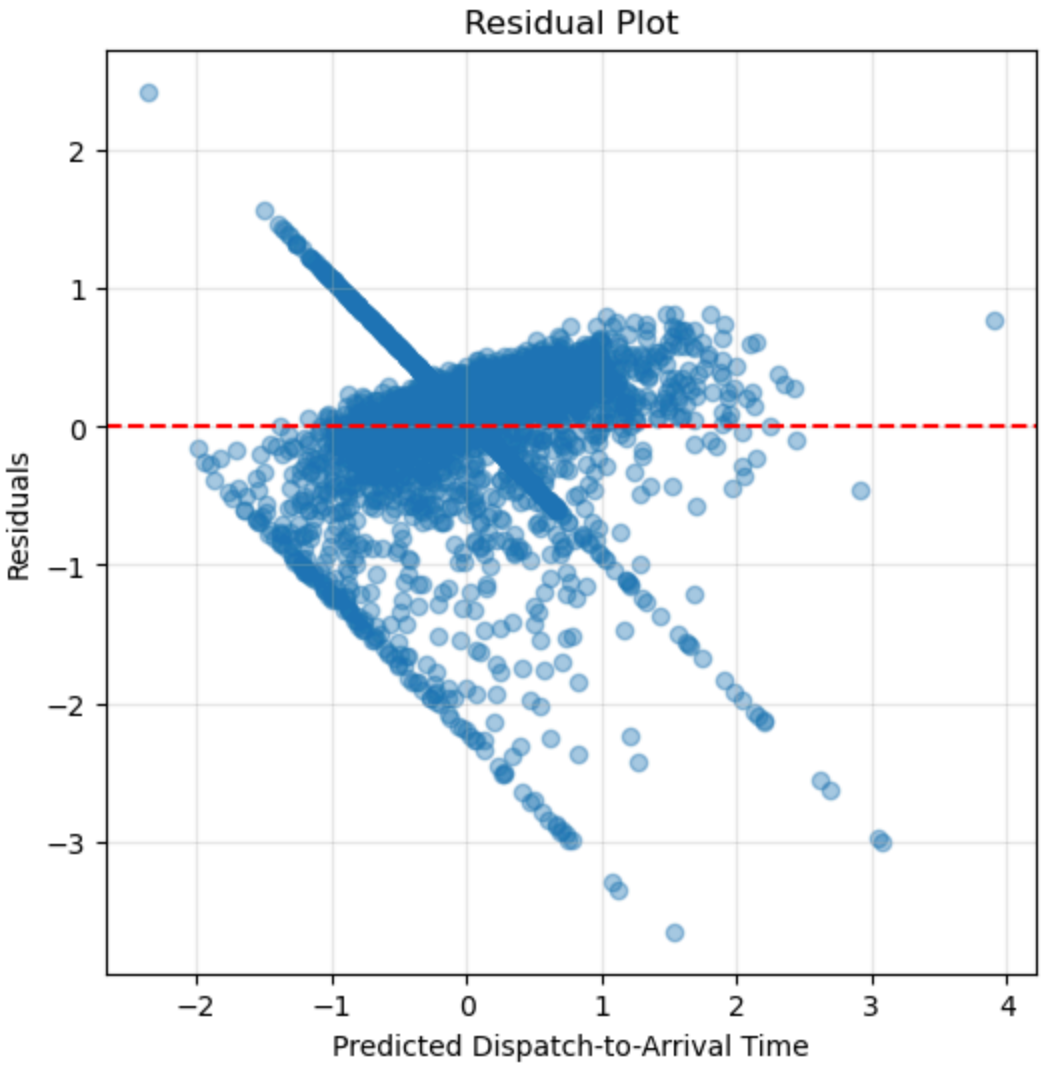
> **Linear Regression**[1]
>
> **Linear Regression** is a supervised learning method used to predict a continuous outcome by estimating a straight-line relationship between the input features and the target variable. The model finds a set of weights that best explain the variation in the data by minimizing prediction error. Because it is simple, fast, and easy to interpret, linear regression is often used as a baseline approach for understanding how different factors influence a numerical response.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Import police incident data
df = pd.read_csv("../../data/processed-data/police_incident_ml.csv")
y = df["dispatch_arrive"]

# Drop unnecessary columns
X = df.drop(columns=["dispatch_arrive"])

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate model
y_pred = model.predict(X_test)

mae  = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2   = r2_score(y_test, y_pred)

print("MAE:", mae)
print("RMSE:", rmse)
```

```
print("R²:", r2)

# Plot Parity Plot
plt.figure(figsize=(6,6))
plt.scatter(y_test, y_pred, alpha=0.4)
plt.plot([y_test.min(), y_test.max()],
         [y_test.min(), y_test.max()],
         "--", color="red")
plt.xlabel("Actual Dispatch-to-Arrival Time")
plt.ylabel("Predicted Dispatch-to-Arrival Time")
plt.title("Parity Plot")
plt.grid(alpha=0.3)
plt.savefig("../../report/visual9.png", dpi=300, bbox_inches="tight")
plt.show()

# Plot Residuals
residuals = y_test - y_pred
plt.figure(figsize=(6,6))
plt.scatter(y_pred, residuals, alpha=0.4)
plt.axhline(0, linestyle="--", color="red")
plt.xlabel("Predicted Dispatch-to-Arrival Time")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.grid(alpha=0.3)
plt.show()
```

```
MAE: 0.38191615721899286
RMSE: 0.572601099206643
R²: 0.5876219624467904
```


Parity Plot
Predicted Dispatch-to-Arrival Time
Actual Dispatch-to-Arrival Time
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5



Model Performance & Insights

From the results above, we conclude that the linear regression model achieved moderate predictive performance. The MAE and RMSE values indicate that the average errors are below 1 standardized unit. With an R^2 of about 0.59, the model explains a meaningful portion of the variation in dispatch-to-arrival time. The parity plot above shows a general diagonal trend, indicating that the model is capturing the broad structure of the data. Also, in the plot, there exists an obvious straight line. The straight vertical lines appear due to a huge number of incidents that have the exact same recorded dispatch to arrival time. However, this pattern is normal for this kind of data due to the limitations of the linear model when working with partially discretized features. The residual plot is also reflecting the huge amount of the same dispatch to arrival times. In the residual plot, most residuals fall close to zero, indicating that the model captures the broad pattern. However, there are clear pockets where the spread widens. These results together highlight where the data carries a meaningful signal.

Conclusion

The different models in the supervised learning section show that different predictions reveal different kinds of structure in the police incident call dataset. The binary classification model performed best, indicating that the

features of the data have enough signal to separate high-priority calls from low-priority calls. On the other hand, the multiclass classification model struggled due to heavy overlap among the incident types. The regression model fell somewhere in between the previous two, capturing broad trends in dispatch-to-arrival time but leaving much of the variation unexplained due to factors not represented in the data. In conclusion, the above findings show that some of the models made good predictions, while others remain challenging.

References

1. ChatGPT, version-5.1, OpenAI, NOV-2025, chat.openai.com.