Data Collection & Processing

Identifying Open Datasets: We begin by finding open data on service requests in New Brunswick (the city encompassing Rutgers – New Brunswick). Many cities provide 311 or service request data via open portals. For example, New York City's public 311 dataset contains over 20 million geocoded service requests since 2010 (A "Rat" Every 14 Seconds: Analyzing NYC 311 Calls). While New Brunswick is smaller, we can seek similar data from local government or state portals. The City of Toronto's open data offers an example, with 4.7 million 311 service call records from 2010–2022 covering hundreds of request types (e.g. noise, potholes, garbage) (311 Call Centre Performance: Rating Service Levels - KDnuggets). Such datasets typically include request date/time, category (issue type), location, and resolution details. If no dedicated New Brunswick portal exists, data may be gathered via state open data platforms or even FOIA/OPRA requests. The Rutgers and New Brunswick Project Move Out initiative is one specific local data source on bulk waste pickups each May (Rutgers University and City of New Brunswick Team Up for Project Move Out | Rutgers University), illustrating a collaborative effort to log and address student-generated bulk waste.

Data Gathering: Once sources are identified, we download the data in formats like CSV or via API. Sometimes multiple files (by year or category) need to be combined. For instance, Washington D.C.'s 311 data is published by year (services - Big Ten Academic Alliance Geoportal Search Results) (services - Big Ten Academic Alliance Geoportal Search Results); we would concatenate such files for a multi-year analysis. We ensure the data covers the key areas of interest: waste management (trash, recycling, bulk pickup, illegal dumping), property maintenance (housing code, overgrown vegetation, snow removal), infrastructure/transportation (potholes, streetlights, parking), and public safety/social services (non-emergency police calls, welfare checks, etc.).

Data Cleaning: Raw service request data often requires substantial preprocessing. We handle missing or inconsistent values in fields like dates, locations, or descriptions. For example, the NYC 311 dataset requires parsing date strings into datetime objects and computing intervals (e.g. time to close a request) (NYC 311's Customer Service Requests Analysis - Kaggle). We will standardize address fields or map them to geospatial coordinates (latitude/longitude) if not already given. Categorical values (complaint types, wards/neighborhoods) are normalized to avoid duplicates due to spelling or formatting. We also remove obvious data errors or duplicates (e.g. identical requests logged multiple times). If data spans many years, we might create new features like year, month, or day-of-week for time-series analysis convenience. The cleaning process also involves filtering to our focus area – e.g. selecting records for New Brunswick city or even isolating those near campus if needed. By the end of this phase, we have a structured dataset with consistent schema, ready for analysis.

Data Integration: In some cases, we might merge datasets – for instance, linking service requests with weather data to see if snow or rain correlates with certain complaints. We might also bring in population or housing data by neighborhood to normalize service request rates. All external data would be joined on common keys like date or location. Finally, we document this entire collection and cleaning workflow in a Jupyter notebook and/or script, so that it's reproducible. The code would be in the repository (e.g. a data_processing.py script or a notebook) with comments and a README explaining data sources and cleaning steps.

Exploratory Data Analysis (EDA)

Volume and Trends: With clean data, we perform EDA to understand overall patterns. We start by calculating summary statistics: total number of service requests, breakdown by category, and requests per year. Large cities see millions of 311 requests (A "Rat" Every 14 Seconds: Analyzing NYC 311 Calls), but New Brunswick's volume will be smaller – perhaps on the order of thousands per year (for context, Toronto's 311 handled ~364,000 requests annually on average (311 Call Centre Performance: Rating Service Levels - KDnuggets)). We examine how the volume of requests has changed over time. A year-over-year trend could show whether New Brunswick residents are reporting more issues now than before (possibly due to increased population or better reporting tools). We also plot monthly or weekly time series of total requests to see seasonal patterns.

Seasonal Patterns: Many service requests have strong seasonal or cyclical trends (Time Series Forecasting with Facebook Prophet | by Ameya Damle). For example, *property maintenance* complaints often peak in summer (grass overgrowth, mosquitoes, etc.) and again after fall (leaf pickup), whereas snow removal complaints occur in winter. In fact, certain complaint types are only relevant in particular seasons (e.g. complaints about heating or lack of heat are mostly in winter, while complaints about *too much heat* in apartments occur in summer when landlords are mandated to keep temperatures below a threshold (Traffic Complaints in NYC Dropped 50% in the Fist Quarter of 2020) (Traffic Complaints in NYC Dropped 50% in the Fist Quarter of 2020)). In New Brunswick, we expect bulk waste and illegal dumping issues to spike in late spring – aligning with Rutgers students moving out. Indeed, each May, tons of furniture and appliances get discarded by departing students (Rutgers University and City of New Brunswick Team Up for Project Move Out | Rutgers

University), historically causing eyesores until the city adapted with Project Move Out. We would verify this by plotting bulk pickup requests by month, expecting a surge in May. Similarly, pothole reports likely rise in late winter and early spring; road freeze-thaw cycles create potholes, so complaints might peak after winter storms. *Emergency service requests* might show increases during certain events (e.g. around holidays or large campus events) or during extreme weather (heat waves, blizzards prompting welfare checks and shelter requests).

Category Breakdown: Using bar charts or pie charts, we visualize the proportion of requests by category. This reveals which issues are most common. In many cities, the top complaints are sanitation-related or housing-related (A "Rat" Every 14 Seconds: Analyzing NYC 311 Calls). For New Brunswick, we might find a large share of requests in *property maintenance* (e.g. housing code issues, pest complaints) or *sanitation* (missed trash pickup, recycling issues). For example, New York logged nearly 250,000 vermin (rat) complaints in its 311 data since 2010, and a significant fraction of all 311 calls were noise complaints (A "Rat" Every 14 Seconds: Analyzing NYC 311 Calls). While NB is smaller, we could see similar themes on a different scale (perhaps complaints about rodents or stray animals in certain neighborhoods, or noise complaints in student housing areas). We also look at *geographical distribution* of requests: using the provided coordinates or addresses, we can map the data to see which neighborhoods or campus areas generate the most service calls. Often, maps show clusters of certain issues – e.g. more noise complaints downtown or more infrastructure issues on certain roads. An interactive map (like NYC's 311 map (311 Services - Data Team - New York City Council)) or heatmap helps identify hotspots.

Anomalies and Outliers: Part of EDA is detecting anomalies – unusual spikes or drops in service requests. We scan the time series for outlier points or intervals. One obvious anomaly in recent data would be the COVID-19 pandemic period. Many cities saw abrupt changes in 311 patterns in 2020. For instance, New York City experienced a 37% year-over-year drop in traffic-related complaints in Q1 2020 (and a 49% drop compared to the previous quarter) due to pandemic lockdowns (Traffic Complaints in NYC Dropped 50% in the Fist Quarter of 2020). If our dataset spans 2020, we should check New Brunswick's data around March–June 2020 for similar effects (e.g. perhaps fewer parking complaints when people stayed home, or more complaints about gatherings/violations of health orders). We also look for spikes corresponding to specific events: e.g., a huge jump in public safety calls during a major storm or after a notable incident on campus. Each anomaly is an opportunity to annotate the dataset (adding a flag or note for that date/event) which can later inform forecasting models and prevent them from treating one-time events as recurring trends.

Visualization: Throughout the EDA, we use clear visualizations: time series line charts for trends, bar charts for category counts, box plots for seasonal variation, and maps for spatial distribution. For example, a box plot of monthly request counts could show the variability and highlight months that are consistently high or low each year. We ensure these visuals are included in the analysis notebooks and will ultimately be part of the reporting dashboard. Each visualization in EDA is accompanied by interpretation: we don't just plot, but also explain what the pattern means for the city (e.g. a steady rise in infrastructure complaints might indicate aging infrastructure or growing population, whereas a cyclical pattern in landscaping complaints confirms seasonality). By the end of EDA, we compile a list of key insights and questions for deeper analysis, setting the stage for the predictive modeling.

Machine Learning Models

With insights from EDA, we proceed to predictive modeling and anomaly detection using several machine learning approaches. The goal is twofold: forecast future service request volumes (especially time-series trends for planning) and classify/detect anomalies or significant categories in the data. We will implement and compare the following models:

1. Time-Series Forecasting with ARIMA and Prophet: For each major service category (and for total requests), we build forecasting models. We use classical ARIMA/SARIMA models as a baseline. ARIMA (Auto-Regressive Integrated Moving Average) is a well-established statistical model for time series data, effective at capturing autocorrelations and seasonality (via seasonal ARIMA) in many real-world cases (Handling Time Series Data with Seasonal ARIMA | by Subash Palvel). We would identify the best ARIMA parameters (p,d,q and seasonal P,D,Q) using techniques like ACF/PACF analysis and cross-validation on a training set. In parallel, we apply Facebook Prophet, a modern additive model for forecasting developed by Facebook's data science team. Prophet is designed to handle complex seasonality (daily/weekly/yearly effects) and holiday or event impacts with minimal manual tuning (Time Series Forecasting with Facebook Prophet | by Ameya Damle). It's well-suited for our problem where we expect weekly cycles (e.g., more reports on Mondays after weekend issues) and yearly cycles (seasonal spikes). Prophet also can accommodate missing data or outliers by treating them as anomalies rather than fitting them, which is useful given irregular events like storms.

We train Prophet models on the historical request counts (by day or week) for categories like waste management, infrastructure, etc. The models will produce forecasted trends and also provide confidence intervals. We compare ARIMA vs. Prophet in terms of accuracy (e.g. RMSE on a test set) and interpretability. Prophet might highlight a yearly seasonality component showing, say, a peak every summer and dip every winter, and it can explicitly model special days (we could add Rutgers academic calendar events as potential holidays in Prophet to see their effects). ARIMA, while sometimes slightly more accurate in strictly linear patterns, might need seasonal terms to match Prophet's flexibility (Multi-step Time Series Forecasting with ARIMA, LightGBM, and ...). We'll report which model fits best for each time series. For example, we might find Prophet captures long-term trend changes better (if the city is growing or policies changed), whereas ARIMA might excel on shorter-term forecasting if the pattern is very regular. Ultimately, these forecasts will help predict future service demand – e.g. forecasting next quarter's pothole reports or the expected number of trash complaints next month, enabling proactive resource allocation.

2. XGBoost for Classification: Aside from pure forecasting, we tackle classification tasks on the service request data. One such task could be predicting the type or priority of a service request based on its description or other attributes, or classifying which department should handle a request. For instance, given a text description of a problem, we could train a model to classify it into waste, infrastructure, property, or safety categories (this could assist triage for new requests). We choose XGBoost, an optimized gradient boosting decision tree algorithm, for this task due to its proven performance on structured data (XGBoost: A Scalable Tree Boosting System - Inspire HEP). XGBoost is highly efficient and has won many machine learning competitions for classification problems, thanks to its ability to handle nonlinear relationships and feature interactions. We will featurize the data for XGBoost: possible features include the text of the request (we can use text embeddings or key phrases), time features (month, day, hour of report), location features (which ward or neighborhood), etc. The model can be trained to classify, for example, whether a complaint is likely about public safety vs. public works, or to predict if a request will be escalated to an emergency response or not.

We also leverage XGBoost for anomaly detection in a supervised way. If we label past data points or days that were "anomalous" (far out of normal range) – for example, days with an unusually high number of requests – XGBoost can be trained to predict a binary label (anomaly or not) from features like day-of-week, weather, etc. This approach treats anomaly detection as a classification problem (Anomaly Detection in Network Data using GPU-Accelerated XGBoost). In practice, an unsupervised or semi-supervised approach might be needed for anomalies (since by definition anomalies are rare and not always labeled beforehand). For that, we could use XGBoost in a forecasting residual method: train XGBoost to predict next-day request count and then flag days where the actual count deviates significantly from the prediction (Anomaly detection on time series with Xgboost algorithm). This method can catch spikes that are not explained by usual patterns learned by the model.

3. Deep Learning Models: We also incorporate neural network models for both forecasting and anomaly detection/classification. For time series forecasting, a recurrent neural network (RNN) or LSTM model could capture temporal dependencies and potentially nonlinear trends in the data. We might set up an LSTM to take a sequence of past daily counts and predict the next value, using separate models per category. Deep learning can sometimes model complex patterns that ARIMA/Prophet miss, especially if there are multiple seasonalities or nonlinear effects. However, deep learning needs a lot of data – with several years of daily data we have enough to try it, though the performance must be weighed against simpler models. For classification, we can use deep learning in the form of deep neural networks or even NLP models. For example, a text classification model (like a TF-IDF + dense network, or fine-tuned BERT) could read the text of service requests and classify the issue type with high accuracy, as has been attempted in NYC 311 analyses (AvonleaFisher/Analyzing-NYC-311-Service-Requests - GitHub). One project, for instance, trained a neural network to predict which city agency would handle a 311 call based on the complaint description (AvonleaFisher/Analyzing-NYC-311-Service-Requests - GitHub). We could do similar for New Brunswick: e.g. predict if a request will require police involvement (public safety) or if it's purely a public works issue, which can help route the request faster.

For anomaly detection, deep learning offers approaches like autoencoders or generative models that learn the normal pattern of data and flag deviations ([PDF] Exploring Autoencoders and XGBoost for Predictive Maintenance in ...). We could train an autoencoder on historical daily patterns (perhaps using a sliding window of a week of data as input) so that it reconstructs typical sequences well but fails to reconstruct anomalous sequences, thereby signaling an anomaly. Another approach is using an LSTM-based prediction and checking the prediction error (similar to XGBoost method but with a neural net predictor). Given the complexity, we might focus on simpler anomaly detection first (like statistical thresholds or XGBoost) and use deep learning to see if it provides improvement.

4. Model Evaluation and Comparison: We will evaluate all models using appropriate metrics. Forecasting models (Prophet, ARIMA, LSTM) will be evaluated on mean absolute error (MAE) or root mean squared error (RMSE) of their predictions on a hold-out test set of time series (e.g., forecasting the last few months based on training on earlier data). We'll also check if they capture seasonality well by visual inspection of the forecast vs actual. Classification models (XGBoost, neural nets) will be evaluated with accuracy, precision/recall (especially if class distribution is imbalanced), and possibly AUC for binary classifications. For anomaly detection, since true anomalies are few, precision in catching those (and not raising false alarms) is key – we might simulate anomaly detection as a binary classification and use F1-score as a metric.

After evaluation, we will compare the results. We might find, for example, that Facebook Prophet provides the most interpretable and accurate forecasts for long-term trends (as it can easily incorporate yearly seasonality and special events) (Time Series Forecasting with Facebook Prophet | by Ameya Damle), whereas ARIMA might perform on par for short-term predictions if the data is very regular. XGBoost might outperform a deep neural network in classifying request types given the size of our data (XGBoost often shines on tabular data with smaller samples (XGBoost: A Scalable Tree Boosting System - Inspire HEP)), but the neural network might pick up subtle textual cues better. We will document these comparisons and provide insight into why a certain model worked best. The outcome might be that a hybrid approach is ideal: e.g. use Prophet for forecasting volumes, and use XGBoost for classifying and detecting outlier incidents. All model training code and results (error metrics, example forecast plots, feature importance from XGBoost, etc.) will be included in the repository, likely in a dedicated modeling notebook or script. We'll also include comments on how these models can be used by city planners – e.g. using the forecasts to budget for overtime or equipment, and using anomaly detection to trigger alerts for city officials.

Urban Impact Analysis

In this section, we translate the data insights and model results into urban planning and operational implications for New Brunswick, focusing on the key service areas:

Waste Management

Efficient waste management is vital for a clean urban environment, and our data analysis sheds light on how New Brunswick handles trash and recycling services. We examine trends in bulk waste pickups, illegal setouts, and overall waste collection efficiency. The data likely show a strong annual surge in bulk pickup requests each May, corresponding to Rutgers' end-of-semester move-out (Rutgers University and City of New Brunswick Team Up for Project Move Out | Rutgers University). This seasonal influx (dozens of tons of discarded furniture and appliances in the student neighborhoods) can strain Public Works schedules. With our forecasts, the city can anticipate these peaks and allocate extra crews or special programs (like "Project Move Out") in late spring. The analysis might reveal whether the Project Move Out (started in 2012) has reduced illegal curbside dumping in those months by comparing pre-2012 and post-2012 data (Rutgers University and City of New Brunswick Team Up for Project Move Out | Rutgers University).

Apart from student-driven bulk waste, we look at regular trash and recycling complaints. Are there patterns to missed pickups or overflowing trash bins? If we see recurring weekly patterns (e.g. many missed pickup complaints on the day after a holiday, or in certain blocks), that indicates where collection efficiency can improve. Perhaps one route is consistently delayed, or certain multi-family buildings generate frequent complaints – such findings can prompt route optimizations or targeted community outreach.

Illegal setouts – trash put out improperly or outside allowed times – are another focus. Our data might include 311 reports of "improper trash set-out" or code violations. Cities like Syracuse explicitly track and even fine illegal setouts (New Covered Trash Carts Begin Arriving at Phase One Homes), and New Brunswick likely does the same to keep streets tidy. By mapping illegal setout reports, we might find hotspots (maybe areas with a lot of transient renters or dense housing). If anomaly detection flags a spike in illegal dumping reports in a certain week, officials can investigate (was there a landlord eviction or a local event causing excess waste?). Over time, a decline in illegal setout complaints would signal improved compliance or enforcement – a positive trend for the city.

We also gauge waste collection efficiency through any available resolution time data. If the dataset logs when a waste complaint was closed, we can compute how quickly issues are resolved on average (NYC 311's Customer Service Requests Analysis - Kaggle). Suppose missed trash pickups are usually handled within 2 days – that's good to note, and outliers (cases that took a week+) can be examined for reasons. By forecasting waste-related requests, we help the city plan resources for sanitation. For instance, if our Prophet model predicts an increase in requests for recycling bins in coming months (maybe due to a new recycling program), the city can stock up on bins in advance.

In summary, the waste management analysis provides actionable insights: when and where to allocate extra trash collection resources, how to reduce illegal dumping, and how seasonal student cycles impact the city. These insights support urban sustainability efforts and a cleaner environment for both Rutgers and the broader community.

Property Maintenance

Property maintenance service requests cover issues like housing code violations, building repairs, overgrown yards, graffiti, and other upkeep concerns. Our analysis explores how these requests vary seasonally and what that means for residents and local businesses (e.g. landscaping and construction services). We likely observe clear seasonal variation in certain complaints. For example, during spring and summer, we expect more complaints about overgrown grass, weeds, or insects (like mosquito breeding sites). In fact, New Jersey researchers noted that complaints about mosquito infestations (e.g. the aggressive Asian tiger mosquito) rise significantly in warmer months, accounting for a majority of service requests to local mosquito control programs ([PDF] WILLINGNESS-TO-PAY FOR AN AREA-WIDE INTEGRATED PEST ...). This suggests that each summer New Brunswick's residents may lodge many requests for mosquito spraying or stagnant water removal, directly impacting pest control and landscaping efforts. Our data can confirm this: if we plot property maintenance requests by month, we might see spikes in summer for vegetation and pest issues. Local landscaping and extermination businesses could use this information to time their services or marketing – for instance, increasing capacity in late spring when demand for yard maintenance or mosquito treatment climbs.

Conversely, in winter, property maintenance calls might shift to snow and ice removal violations (e.g. uncleared sidewalks) and indoor issues like lack of heat. Many cities enforce sidewalk shoveling; a surge in "snow not removed" complaints after a snowstorm (likely reported via 311 in the days after a big snowfall) could inform city inspectors where to issue citations or send crews. By analyzing a few winters of data, we can identify the typical *time window* for snow complaints (often within 24-48 hours post-snowfall (Sidewalk Snow & Ice - City of Minneapolis)) and the areas often in non-compliance. This helps Public Works and Code Enforcement prioritize which neighborhoods need outreach about snow shoveling rules.

Our analysis also correlates these maintenance issues with potential impacts on the real estate market. For instance, if a particular area has chronic property maintenance problems (peeling paint, broken sidewalks, etc.), it might affect property values or rental desirability. While our project isn't a full economic analysis, we can note if certain blocks have recurring complaints – possibly flagging them as needing urban revitalization. On a positive note, a trend of decreasing property maintenance complaints over time could suggest that earlier interventions (grants for home repairs, stricter code enforcement) are working.

From the data, we derive insights for businesses too. For example, if *leaf collection* complaints (clogged storm drains or uncollected leaf piles) peak every November, landscaping companies could coordinate with the city for efficient pickups or offer services accordingly. If *graffiti reports* spike in summer (when school is out, perhaps more vandalism occurs), local mural artists or youth programs might be engaged to proactively address it. All these interpretations stem from the patterns we find in the service request data.

By forecasting property maintenance requests, the city can prepare seasonal crews: more inspectors in summer for yard issues and in winter for heating complaints or snow issues. The analysis might even predict emerging trends – e.g., if we see year-on-year growth in *abandoned property* complaints, that could indicate economic shifts that the city should address through policy (perhaps offering incentives to fill vacant properties). Our machine learning classification could also help here: by reading complaint text, we might automatically categorize whether a property issue is major (structural, needs urgent attention) or minor, which helps triage the city's response. Overall, the property maintenance analysis guides both municipal code enforcement and local businesses (landscaping, repair services) in planning for seasonal demand and addressing persistent issues that affect community quality of life.

Infrastructure & Transportation

Infrastructure and transportation-related service requests include problems like potholes, street light outages, broken traffic signals, sidewalk issues, and parking complaints. These are critical for urban mobility and safety. Our analysis pays special attention to pothole reports as a key indicator of road maintenance needs. Potholes tend to be seasonal: as winter ends, roads often develop potholes, so late winter and early spring bring a wave of complaints. By analyzing past data, we can quantify this – for example, we might find that 70% of annual pothole complaints occur between January and April, aligning with freeze-thaw cycles and heavy salting of roads. The City of Toronto's 311 data confirms that potholes are a common complaint type among millions of requests (311 Call Centre Performance: Rating Service Levels - KDnuggets), and New Brunswick likely shows the same (albeit with smaller absolute numbers). We will map pothole reports to see if certain streets or intersections are chronically problematic. This can guide the city's paving schedule: if one avenue consistently has many pothole fixes, it may need a more thorough resurfacing. Our forecasting models (ARIMA/Prophet) for potholes can project how many reports to expect next quarter or next winter, helping the Public Works department budget materials and labor. If, say, our model predicts an especially high count next year (maybe due to an unusually harsh winter forecast), the city can plan proactive road repairs.

Parking violations and complaints are another area of interest. These might include reports of illegal parking, requests for parking enforcement, or grievances about parking availability. Around a college campus, parking is often a hot topic. We might find spikes in parking complaints at the start of semesters (when new students or staff are unfamiliar with regulations) or during big events (football games, festivals causing parking scarcity). By examining temporal patterns, we can detect such surges. For instance, a surge in parking complaints every September could lead Rutgers or the city to improve communication of parking rules or adjust permit enforcement during that month. If location data is available, we can pinpoint parking trouble zones – perhaps near College Avenue or popular off-campus housing blocks. This insight might prompt the city to consider new parking lots or stricter enforcement in those zones. Our analysis might also overlap with public safety here; e.g. parking in fire lanes or near hydrants could be a subset of complaints that have direct safety implications. An anomaly detection model could flag if a particular day had an abnormally high number of parking complaints, possibly indicating an event that overwhelmed parking capacity – the city could then evaluate if temporary parking provisions are needed for such events in the future.

We also look at other infrastructure issues like streetlight outages or traffic signal malfunctions (these impact both transportation and safety after dark). By tracking how quickly these are fixed (if data provides open vs closed times), we can comment on the city's responsiveness. Perhaps most traffic light issues are resolved in a few hours, which is good, but some streetlight outages linger for weeks – indicating either resource constraints or communication gaps with the utility company. Highlighting these can lead to process improvements (maybe an automated alert from 311 to the electric utility for streetlight repairs).

Furthermore, our forecasts for infrastructure requests help capital planning. If we see a rising trend in a category like "street flooding complaints" or "sewer backups" (sometimes found in 311 data), that might suggest aging infrastructure needing investment. A steady increase in such complaints year over year could support proposals to upgrade drainage systems in the city's capital budget.

In summary, the infrastructure & transportation analysis uses data to keep New Brunswick moving safely. By predicting pothole occurrences, identifying parking hot spots, and monitoring maintenance response times, the city can prioritize repairs and policy changes. For example, if our model predicts a heavy pothole season, the city might allocate overtime for road crews in March. If parking complaints are trending down after a new policy, that's a success to note; if not, new solutions (like parking apps or adjusted pricing) might be explored. All these insights ensure that roads, sidewalks, and transit-related facilities are maintained proactively, benefiting both the Rutgers community and city residents in their daily commutes.

Public Safety

Public safety-related service requests encompass calls that may overlap with emergency services or community well-being concerns. In New Brunswick's context, this could include non-emergency police calls (noise complaints, suspicious activity reports), fire department calls for assistance, EMS calls, as well as social services requests like welfare checks, homeless outreach, or mental health assistance. While 911 calls are separate from 311 in many cities, sometimes less urgent safety requests or referrals come through the 311 system or city service lines. Our project analyzes any available data on these emergency and social service requests to glean patterns and help predict future demand on public safety agencies.

One key analysis is examining temporal patterns of public safety calls. There may be daily and weekly cycles – for example, noise complaints might spike on weekend nights. If data distinguishes different types (say, loud party complaints vs. other police matters), we can chart each. Seasonal trends are also worth noting: perhaps noise complaints dip in winter (when windows are closed and students are away during holidays) and peak in warmer months. Requests for social services (like shelter or heating assistance) likely peak in the coldest months, whereas calls about fireworks or outdoor nuisances peak around the 4th of July. Identifying these trends means police and social service departments can staff accordingly – e.g. more patrols during known problem times, more social workers on call during winter nights, etc.

We apply time-series forecasting here as well. Using our Prophet or ARIMA models, we forecast volumes of certain public safety related calls. For instance, we might forecast the number of noise complaints or "person in need of assistance" calls for the next few months. This is similar to forecasting 911 call volumes, a practice already used in some cities to optimize dispatch resources ([PDF] Time Series Analysis and Forecasting of Emergency Dispatch ...). In fact, a study of 911 dispatch data showed that modeling and forecasting call counts can help predict future emergency service demand ([PDF] Time Series Analysis and Forecasting of Emergency Dispatch ...). Likewise, our forecasts for New Brunswick can help Rutgers police and New Brunswick first responders prepare. If we anticipate an uptick (perhaps the model sees a trend that as student enrollment grows, certain call types increase proportionally), those agencies can request more support or initiate preventative measures.

Anomaly detection is particularly useful in public safety context. A sudden spike in certain types of calls could indicate an emerging crisis. For example, if in a single week there's a huge jump in "mental health assistance" requests, that might signal a community trauma or external event affecting many people. Our anomaly detection model would flag this, and we'd then cross-reference news or context (maybe a tragedy or a pandemic-related stress spike) and inform relevant departments to respond (such as deploying counseling services). Another example: a spike in emergency maintenance calls (like "tree down on power line") might coincide with a severe storm – combining 311 and 911 analyses could give a full picture of the storm's impact.

We also correlate public safety requests with geographic data to find hotspots of incidents. Perhaps certain blocks near campus bars generate repeated noise or fight complaints – that could influence police deployment or city ordinances (like adjusted bar closing times or more security). If certain intersections see many traffic incidents reported, maybe additional traffic calming is needed there.

Lastly, we consider resource implications. If our analysis shows, for instance, a steady year-over-year rise in social service requests (maybe more homeless individuals seeking shelter via 211/311 calls), the city can use that to argue for increased funding to social programs or shelters. If noise complaints are the majority of police-related 311 calls, perhaps Rutgers or NBPD can start a campaign or program for noise reduction in off-campus housing, which could preempt some calls.

By forecasting and understanding these public safety related trends, we aim to enhance community safety and wellness. The city and campus police can move from reactive to proactive – using data to anticipate when and where extra patrols or services will be needed. This not only helps in handling emergencies better but also in improving everyday quality of life (quiet neighborhoods, timely assistance for those in need, etc.). Our repository will detail these findings, with maps of public safety calls and charts of their trends, providing a clear picture of how demands on public safety are changing in the Rutgers–New Brunswick area.

Visualization & Reporting

To communicate our findings effectively, we will produce a set of interactive dashboards and static reports that present key insights in an accessible manner. These visualizations will help city officials, university administrators, and the public quickly grasp the patterns in service requests and the predictions from our models.

Interactive Dashboards: Using tools like Tableau, Power BI, or Python-based dashboards (Streamlit or Plotly Dash), we will create interactive visuals that allow users to explore the data. For example, a dashboard might have a map of New Brunswick with points or heatmaps showing service request density by location (311 Services - Data Team - New York City Council). Users could filter by category (waste, property, etc.) to see the hotspots for each type of issue. A time-series widget could allow selection of a date range and display the trend of requests over that period. We will include interactive charts such as:

- A time-series graph with actual vs. forecasted requests, with the ability to toggle different models (Prophet/ARIMA) or different categories. This could show, for instance, the predicted rise in potholes next spring and allow the user to hover and see exact values and confidence intervals.
- A seasonality decomposition view (from Prophet) that graphically shows weekly and yearly effects on the request volume, which helps illustrate

patterns (e.g. the weekly pattern might show lower reports on weekends, yearly pattern might highlight summer peaks).

- Bar charts or tree maps for breakdown of request types, where clicking on a segment (like "Illegal Dumping") updates other parts of the dashboard (map and time series) to focus on that subset.
- Possibly a calendar heatmap to show the volume of requests on each day of a year, which can highlight outlier days (anomalies) in a red-to-blue color scale.

The dashboard will also incorporate any geographical visualization we have, since location is crucial for urban planning. For instance, we can embed a map that shows wards or neighborhoods colored by number of service requests, or pins for each request (with pop-up details on clicking). Modern 311 data portals often provide such maps (311 Services - Data Team - New York City Council), and we will emulate that style, customized for our analysis of New Brunswick.

Static Reports and Visuals: In addition to interactive tools, we will prepare a formal report (likely in Jupyter Notebook form, exported to PDF/HTML) summarizing the project. This report will contain static versions of our key charts and graphs, with captions and interpretations. For example, we'll include:

- A line chart of total monthly service requests over several years, annotated with notable events (e.g., "COVID lockdown" on March 2020 where the line dips, "Project Move Out" each May where a spike occurs).
- A bar graph comparing model performance (Prophet vs ARIMA vs XGBoost vs LSTM) perhaps in terms of error rates, to visually communicate which model performed best for forecasting.
- Several subplot charts focusing on each domain: e.g. a small multiple of four plots, one each for waste, property, infrastructure, safety, showing their seasonal trend over a typical year. This makes it easy to compare timing (perhaps waste and property peak in summer, while safety has smaller peaks around specific events).
- Maps or density plots showing, for instance, pothole locations or distribution of property complaints by neighborhood. If possible, we can overlay service request data on a map of the Rutgers campus and city – highlighting, say, that the downtown area (near College Ave) had X% of all noise complaints.

We will also create some geographical visualizations explicitly. If the data has coordinates, we might generate a heatmap or cluster map of complaints. Alternatively, using neighborhood shapefiles, a choropleth map could depict the rate of complaints per 1,000 residents in each area, which normalizes by population. This can show which communities have disproportionately high service needs (which could correlate with socioeconomic factors – a potential point of further study).

All visualizations will be designed with clarity in mind – appropriate titles, labels, and legends. We'll adhere to best practices (avoiding misleading axes, using colorblind-friendly palettes, etc.). The interactive dashboard will be deployed (if possible) or at least included as a Tableau Public link or similar (311 NYC Open Data Visualization – Visualization and Design (S24)), so stakeholders can use it hands-on. The static report will be in the GitHub repository for quick viewing.

Narrative and Insights: Alongside the visuals, we provide concise commentary. For instance, under a chart showing seasonal variation in property maintenance, we might write a short paragraph explaining the likely reasons (weather, academic calendar) and the impact on businesses. Each section of the report ties back the visual evidence to actionable insights: e.g., "Map of illegal dumping incidents – shows clustering in the 5th Ward; suggests targeted enforcement in that area." By combining interactive exploration for power users and static summaries for decision-makers, our project's findings can reach a broad audience effectively.

GitHub Project Structure

Our entire project will be organized in a well-structured GitHub repository, following best practices for data science projects (Cookiecutter Data Science) (Cookiecutter Data Science). The repository will be publicly accessible and will contain all code, data references, and documentation needed to reproduce the analysis and models. Here's how we'll structure and document the repository:

- README.md: A detailed README at the root of the repo will provide an overview of the project. It will describe the project's objective, the data sources used (with links or instructions to obtain them), and a summary of the contents of the repository. The README will also have a quick start guide for how to set up the environment and run the analysis. For example, it will list required libraries and how to install them (possibly via a `requirements.txt` or environment file) and then outline the order in which to run the notebooks or scripts (e.g. "1. Run data_collection.ipynb to download and clean data, 2. Run eda.ipynb to generate exploratory analyses, …").

- Data Directory: We will have a `data/` folder (or a data download script). Given open data size, we might not store raw data in the repo, but we'll provide instructions to obtain it (like a script to download from an open API). Within `data/`, we can have subfolders for `raw/`, `processed/`, etc., following the cookiecutter data science structure (Cookiecutter Data Science). For example, `data/raw/` could be empty but with a note that running the data collection script will populate it. `data/processed/` might contain cleaned datasets ready for analysis (if size is manageable to include, or code to generate them). We will be cautious not to bloat the repository with huge files; if needed, we'll use Git LFS or links to external data.

- Notebooks: The analysis will be documented in Jupyter Notebooks for clarity. We may have notebooks such as:
  - 1_data_collection_and_cleaning.ipynb – showing how we load and clean the data (with outputs like summary stats of missing values, examples of cleaned records).
  - 2_eda_visualizations.ipynb – containing the exploratory analysis plots and narrative.
  - 3_forecasting_models.ipynb – where we develop and evaluate Prophet and ARIMA models (including visual comparisons of forecasts).
  - 4_classification_anomaly_models.ipynb – building XGBoost, neural nets, and performing anomaly detection, with evaluation results.
  - 5_dashboard_preparation.ipynb – if we prepare data specifically for visualization or create figures for the report.
- Each notebook will be clearly commented and organized with sections (using Markdown headings) for each step of the analysis. This makes it easy for another user to follow our process and reasoning.

- Scripts and Modules: For more complex tasks or to avoid duplicating code, we will include Python scripts (in a src/ or similar directory). For instance, we might have a src/data_processing.py that contains functions for cleaning data, which the notebook calls. Or a src/models.py with functions to train and evaluate a model. By modularizing, we adhere to software engineering principles and make the code reusable. If we have geospatial processing, those functions can live in a src/geoutils.py, for example.

- Models and Outputs: We will have a folder (e.g. models/) to save trained model artifacts if needed (like serialized Prophet models or saved XGBoost models), though often we can regenerate them. Also, a reports/ folder will hold the final compiled outputs – for example, the final PDF report or HTML summary, and maybe any slides if this was a presentation. A reports/figures/ subfolder can store important static images used in reporting (Cookiecutter Data Science) (ensuring the images displayed in our README or documentation are accessible from the repo).

- Environment & Reproducibility: We will include a requirements.txt listing all Python packages and versions used (pandas, fbprophet, scikit-learn, xgboost, etc.) (Cookiecutter Data Science). Additionally, if non-Python tools were used (maybe R for a part of analysis or Tableau for dashboard), we will mention those and provide the packaged workbook or code. The README will explain how to set up the Python environment (for example, using pip or conda to install the requirements). If the project is complex, we might also provide a Dockerfile for a fully reproducible environment, though that might be optional for this scope.

- Documentation: Apart from the README, we may use the GitHub wiki or a docs/ directory for additional documentation. However, given this is likely an academic or portfolio project, the notebooks themselves serve as documentation. We will ensure all explanations, interpretations, and instructions are either in the notebooks or the README. Key design decisions (like why we chose certain models or how we dealt with data limitations) will be recorded so that a reader understands the rationale without guesswork.

- Version Control and Collaboration: All changes will be tracked via Git commits with clear messages. If working with others, we'd use branches and pull requests, but in any case, the history will show the development process. The structure is inspired by standardized project templates (e.g. cookiecutter data science, which advocates for separate directories for data, notebooks, models, etc. (Cookiecutter Data Science) (Cookiecutter Data Science)). This logical organization makes it easy for anyone (including city officials who might not be coders) to navigate. For example, an interested stakeholder could open the EDA notebook and see the charts without digging through code, or open the forecasting_models notebook to see how we predicted future trends.

Finally, we will provide setup instructions in the README (and possibly as a separate INSTALL.md). This will include: how to clone the repo, how to get the data (with links or scripts), how to run the notebooks in order, and how to launch the dashboard (if it's a local Dash app, for instance, we'll explain the command to run it). We want someone to be able to replicate our entire analysis from scratch. For deployment, if we have a live dashboard or if the models could be deployed as an API, we'll include notes on that too – e.g. "to deploy the forecast model, run src/forecast_api.py which starts a Flask app on port 5000."

By maintaining a clear structure and thorough documentation, the project repository will not only demonstrate our analysis results but also serve as a reference toolkit for similar urban data analyses. Interested parties can adapt our code to other cities or update it with new data going forward, amplifying the impact of our work beyond just this project.

Overall, the final deliverable is a comprehensive, well-documented repository containing data, code, models, visualizations, and insights for service requests in Rutgers – New Brunswick, empowering data-driven decisions in urban planning and resource management (Cookiecutter Data Science) (Cookiecutter Data Science).