Predicting Overdue 311 Cases:

On a mission to improve NYC's Operational Efficiency

ISSS610 Applied Machine Learning G1

Project Report

Group 2

Lau Te Yang (teyang.lau.2021@mitb.smu.edu.sg)

Lim Hsien Yong (hy.lim.2021@mitb.smu.edu.sg)

Loh Si Jun Shauna (shauna.loh.2021@mitb.smu.edu.sg)

Yan Licheng (licheng.yan.2021@mitb.smu.edu.sg)

Vannarath Poeu (vannarathp.2021@mitb.smu.edu.sg)

2nd April 2022

Code Repository:

https://smu.sharepoint.com/:f:/t/Group2ISSS610-AML/EmUR6eivtO1Oi49xGam2lP0Bgt7DskzraEsZ6mNJo1X65Q?e=Ix6dlc

1. Introduction

The 311 hotline is a non-emergency hotline available in several cities of the United States of America. Instituted with the intention to divert non-emergency concerns away from the 911 emergency hotline, the 311 hotline was launched in New York City (NYC) in 2002 by then Mayor Michael Bloomberg[1]. The service is available 24 hours across all days in the year, and can be accessed by mobile apps, online and calls. Government agencies are required to respond to all complaints within a time limit specified in the Service Level Agreement (SLA). However, case resolution times often exceed the stipulated limits. As a result, there have been many studies by research groups and data science hobbyists alike – with many publicly available 311 datasets for different jurisdictions. Most prior studies predict the response times for individual cases. Instead, we study breach days[2] as a regression problem - quantifying how many days a given complaint is expected to breach the SLA. We envisage for insights of our study to assist agencies in prioritizing different service requests for follow-up.

2. Data Sources

The list of data sources used are as follows:

S/N	Dataset	Rows x Columns	Source
1.	NY 311 Service Requests	27.7mil x 46	311 Service Requests from 2010 to Present. *NYC Open Data* https://nycopendata.socrata.com/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9/data
2.	NY Service Level Agreement	3042 x 5	311 Service Level Agreements. *NYC Open Data.* https://data.cityofnewyork.us/City-Government/311-Service-Level-Agreements/cs9t-e3x8/data
3.	NYC Socioeconomic data	2440 x 7	*The Status of New York City Children*. https://data.cccnewyork.org/data

Table 1: List of data sources - for full list of features, refer to Annex A

As the dataset is voluminous and there is likely a shift in the intrinsic nature of the data from 2019 onwards due to COVID-19, the team decided to use (i) 2017's data for model training, validation and testing in an 80:10:10 ratio, and (ii) use 2018's data for robustness testing of models developed by ascertaining the extent of model drifts. We also incorporated socioeconomic data as x-features. The train-test-validation split is sufficient, as there are more than 1 million complaints per year, and we ascertain that a test and validation size of >100,000 rows each was sufficiently representative. Downloaded datasets are stored into and merged using the materialized view capability of a dedicated PostgreSQL database due to persistence and performance considerations[3]. An ER diagram describing how the data is stored in the database can be found in Annex A.

[1] Mayor Bloomberg Commemorates Ten Years Of Nyc311, The Nation's Largest And Most Comprehensive 311 Service. *NYC.* https://www1.nyc.gov/office-of-the-mayor/news/089-13/mayor-bloomberg-commemorates-ten-years-nyc311-nation-s-largest-most-comprehensive-311

[2] Defined as the difference between closure duration (date close – date open) and SLA duration for each complaint type.

[3] Persistence: we can make changes to the materialized view definition and they can be propagated to all team members without the need to export data and share across cloud storage. Performance: By using a dedicated database, we lessen the impact of data joining and large queries on team members' productivity.

3. Exploratory Data Analysis (EDA)

S/N	Observation	Insights
1.	Cases by the top 5 agencies[4] take up 99% of all complaints.	We can explore grouping complaints of other agencies under "Others".
2.	Large number of extreme outliers [5]. This suggests that the high variability is a characteristic of non-emergency complaints.	To explore techniques to handle outliers.
3.	High cardinality of categorical features.	We can explore binning infrequent complaint types into larger buckets to reduce feature explosion
4.	Strong correlation between some categorical variables[6].	This suggests that variables exhibiting collinearity can be removed to reduce dimensionality.
5.	A geo-plot of complaints shows frequent occurrences of high breach days in the upper side of New York City.	This suggests that latitude and longitude can prove to be a useful feature for our model. To reduce dimensionality, we explore binning instead of using latitude and longitude separately.
6.	We observed correlations between socioeconomic data (for e.g. income) and breach days.	This suggests that socioeconomic data (dataset #3 of table 1) could capture variance in breach days.

Table 2: Observations from EDA – all plots can be found in Annex B.

Our EDA suggests that socioeconomic data (dataset #3 of table 1) contains useful x-features to predict breach days, and also highlights the intrinsic characteristics of our dataset that require special attention prior to modelling.

4. Models explored

The team explored 4 different models:

a. **Linear Regression:** Including the study of regularization hyperparameters.
b. **Support Vector Regression:** Including the linear support vector regressor and radial basis function support vector regressor.
c. **Random Forest Regression:** A bagging tree-based ensemble technique.
d. **Extreme Gradient Boosting Regression:** A boosting tree-based ensemble technique.

5. Pre-processing techniques explored

In total, we attempted 14 different feature engineering techniques.

S/N	Technique	Category	Insight
1.	Dropping all outliers above the upper quartile.	Outlier removal	Improves model fit but model suffers from lack of generalizability to future data that might also contain these outliers.
2.	Manual setting of breach day threshold and thereby dropping all outliers beyond the breach day threshold.		Not a good approach as the threshold is too arbitrary.

[4] They are the New York City Police Department (NYPD), Department of Housing, Department of Mental Health and Hygiene, Department of Transportation and Department of Parks and Recreation
[5] Defined to be located beyond the upper whisker of categorical boxplots.
[6] New York City comprises of 5 boroughs, with each borough comprising of a few 'cities' and community boards.

3.	Retaining all data points.		Better approach because the large volume of outliers is representative of the dataset, which allows for model generalization.
4.	Binning of longitude and latitude (Feature crossing)	Feature engineering	The crossed feature provides better predictive ability beyond using latitude and longitude individually.
5.	Extraction of week, month, time of day from datetime features.		Model can consider seasonality effects in prediction.
6.	Standard scaler, Min Max scaler	Scaling	The approaches are heavily affected by the presence of outliers
7.	Robust scaler		The scaler is based on percentiles and is hence not influenced by large marginal outliers.
8.	Principal Component Analysis	Dimensionality Reduction	Resultant components were difficult to interpret, and the approach did not improve model performance.
9.	Singular Value Decomposition		
10.	Manual re-group	Grouping of categorical features	Too many arbitrary ways to regroup because of large number of levels.
11.	Pareto principle[7] re-group		Many disparate complaint and location types were grouped into “Others" group, thus losing information.
12.	Quartile binning		The approach did not improve model performance.
13.	One-hot encoding	Encoding	Results in an explosion in the number of features due to the high cardinality of categorical data.
14.	Target encoding		Prevents feature explosion while yielding comparable performance to one-hot encoding for Linear Regression, Support Vector Machine and Random Forest models.

Table 3: Feature engineering techniques attempted

The techniques of feature engineering attempted, as summarized in Table 3, can be grouped into the following categories:

a. **Outlier removal:** The large proportion of datapoints above the upper quartile were of concern, as these would greatly impact modelling. The techniques explored aimed at sufficiently addressing this.

b. **Dimensionality reduction:** The high cardinality of categorical features resulted in a large increase in the number of x-features (>200 columns) from one-hot encoding. Techniques explored in this category aimed to mitigate the curse of dimensionality.

[7] The Pareto Principle postulates that 80% of the consequences come from 20% of the causes. In this approach, we grouped the bottom 80% of the complaint types into ‘others’ and leave 20% of the most common complaint types as is. Reference: Pareto Principle. *Investopedia.* https://www.investopedia.com/terms/p/paretoprinciple.asp

c. **Grouping of categorical features:** The techniques attempted aimed to address issues arising from high cardinality of categorical features differently – by grouping categories where possible. This would in turn reduce the increase in one-hot encoded columns.
d. **Encoding:** The techniques attempted aimed to represent categorical data in numerical format for machine learning.

Differences in feature engineering during individual model explorations made it difficult to compare between models directly. This necessitated a streamlined approach, which allowed for all models to leverage on the team's combined discoveries.

6. Results and Discussion

Hyperparameter tuning was done using GridSearch for each model type based on the streamlined approach. A summary of our hyperparameters tuned and their best values are tabulated as follows:

Linear Regression	**Support Vector Regression**	**Random Forest Regression**	**Extreme Gradient Boosting Regression**
Regularization = 'L1' (Lasso)	C = 0.50	Min_samples_leaf = 4	Min_child_weight = 5
Penalty term (alpha) = 0.01	Epsilon = 0	Max_features = 'auto'	Learning rate (eta) = 0.3
	Kernel = Linear SVR	Max_samples = 'None'	Number of estimators = 300
		N_estimator = 200	Max_depth = 9
		Max_depth = 'None'	Reg_alpha = 0.3
		Ccp_alpha = 0.00	Reg_lambda =0.5
			Gamma = 0.01

Table 4: Summary of hyperparameters tuned

A summary of our models and their respective best encoding methods are as follows:

	Linear (Lasso) Regression	**Linear Support Vector Regression**	**Random Forest Regression**	**Extreme Gradient Boosting Regression**
Type of encoding	Target	Target	Target	One-Hot
Training R^2 (2017)	0.32	0.24	0.76	0.77
Validation R^2 (2017)	0.30	0.23	0.52	0.50
Testing R^2 (2017)	0.31	0.22	0.50	0.47
MAE (2017)	9.55 days	8.20 days	7.06 days	8.20 days
R^2 (2018)	0.25	0.18	0.17	0.07
MAE (2018)	9.77 days	8.41 days	10.49 days	11 days

Table 5: Summary scores of models tuned

Our best performing model is the random forest regression model with target encoding.

Observation 1: Our models were able to predict breach days within an error range of 7-9.6 days, which was encouraging due to the high variability of breach day data - some complaints within our dataset are closed within hours, while others are closed in more than 100 days. However, we opined that the testing R^2 value is a more suitable metric, as the low scores suggest our models do not fit the data well.

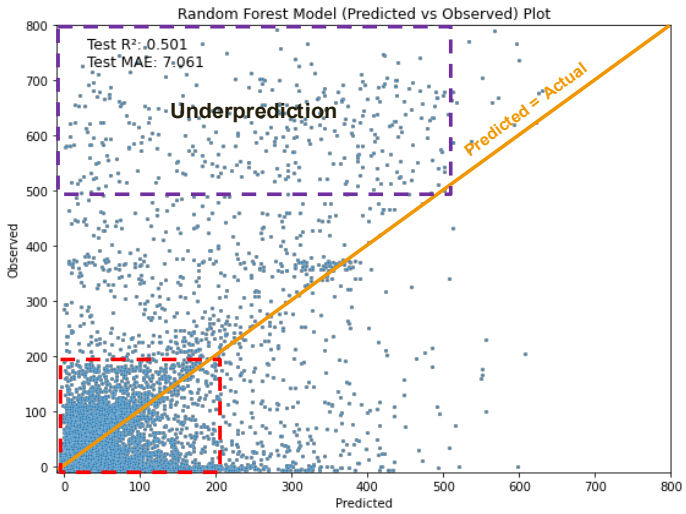


Figure 1: Plot of model predictions against test data

We plotted our best model's predictions against test values (figure 1) to investigate further and observed that most of the data points on the plot lie within the (0-200, 0-200) region as marked by the red box. When compared against the diagonal line, it is evident that our model is inconsistent, predicting heavily on both sides of the diagonal (i.e. overpredicting and underpredicting). When the actual breach days exceed 200, our model tends to underpredict as indicated by the purple box.

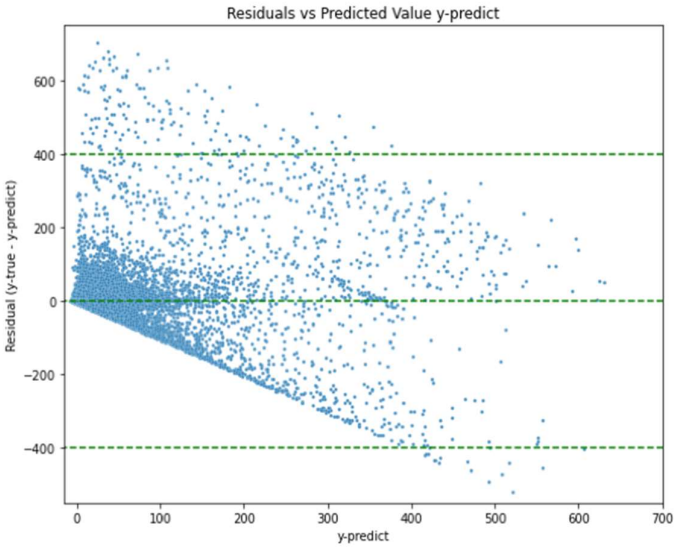


Figure 2: Residual plot of the model

A plot of model residuals against prediction (figure 2) showed that residuals are not randomly dispersed around the residual=0 horizontal line. This indicates possible auto-correlation and/or uncaptured relationships not accounted for by the model.

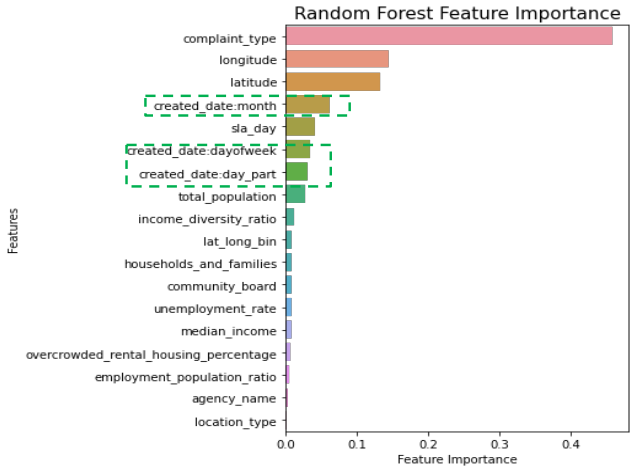


Figure 3: Feature importance plot

A feature importance plot reveals that the high cardinality column 'complaint_type' was identified to be the most important feature by the model. This is expected, as a water leakage complaint is resolved within a different time window than a complaint about a porthole in the road. The model's identification of engineered time series features (boxed in figure 3) also suggests that time series features have a bearing on breach days.

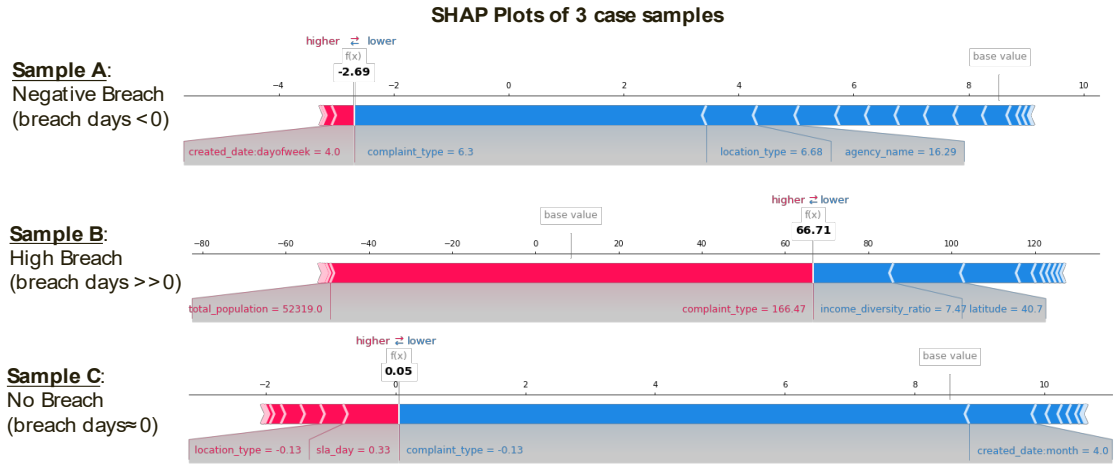


Figure 4: SHAP force plot of breach day archetypes

However, we suspected that the ranking of feature importance in figure 3 is inconsistent across different breach day types as the scatterplot in figure 1 suggests inconsistent predictive patterns across different breach day zones. To better illustrate the inconsistency, we used a SHAP force plot to identify the top features and their influence on the score across 3 different breach day archetypes – negative breach days (sample A), high breach days (sample B) and no breach (sample C, i.e. breach days = 0) for local interpretability. 'complaint_type' remained the most important feature, but pushed breach day predictions downward as actual breach days increase. Under negative or no breach circumstances, 'complaint_type' pushed the final predictions upward instead. With the exception of 'complaint_type', the mix of the top features and their importance differ across all samples.

We observed that residuals appear to exhibit heteroscedasticity, as the variance of residuals is uneven across the range of values in figure 2. The presence of heteroscedasticity verifies our prior understanding that the model is unable to accurately capture some of the relationships between the x-variables, resulting in poor predictive accuracy. A common method to address heteroscedasticity is scaling of the dependent variable. We checked for heteroscedasticity using the Breusch-Pagan test with scaling and report the results as follows:

Scaling	**Lagrange Multiplier Statistic**	**Lagrange p-value**	**f-statistic of the hypothesis that the error variance does not depend on x**	**p-value of the f-statistic**
No scaling	14,030	0.00	877.69	0.00
Standard Scaling	13,940	0.00	871.32	0.00
Min Max Scaling	14,009	0.00	876.19	0.00
Robust Scaling	14,022	0.00	877.01	0.00
Log2 scaling	5557	0.00	350.19	0.00

Table 6: Summary of Breusch-Pagan Test

p-values of <0.05 suggest heteroscedasticity which is the case for our model, as both p-values breach this threshold.

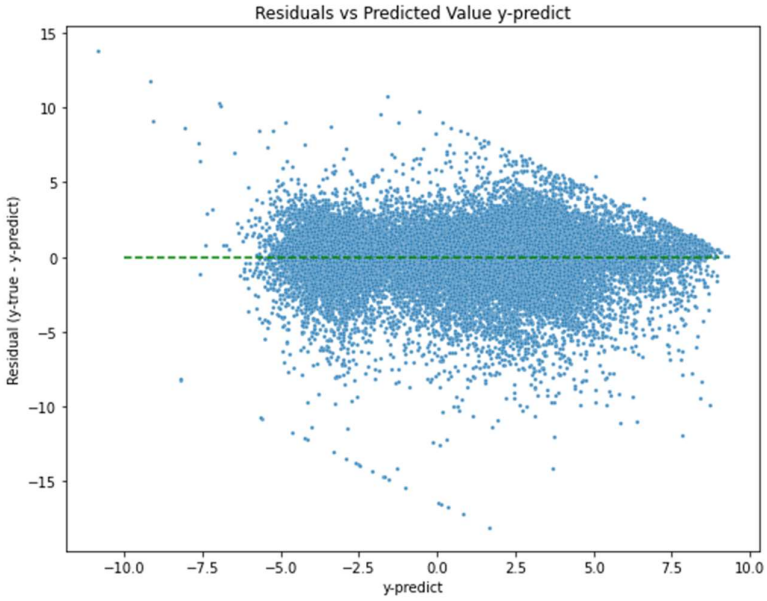


Figure 5: Residual Plot of dependent variable scaled with log2

We continued to observe heteroscedasticity despite scaling. We observe an overall downward trend in the residuals of figure 5. Heteroscedasticity suggests that additional feature engineering work is required. A potential workaround could be to develop new explanatory features by scrutinizing cases with similar feature values and different predicted breach days.

Observation 2: Both linear models suffered from **high bias** (i.e., very low train and validation r^2), which was expected given the complexity and non-linearity of our objective problem. Feature engineering efforts through feature crossing like lat-long binning and polynomial feature transformations did not improve performance by much. Hence, more complex models, collecting more data and advanced feature engineering techniques might be necessary – such as computing the cumulative unsolved cases of each agency.

Tree-based models were observed to have **high variance** (i.e., higher train r^2 than validation r^2, which suggests overfitting to the train set. This could be due to several reasons.

a. Having too many features/dimensions. Nonetheless this persists despite target encoding.
b. Overfitting to the noise inherent in the train set which was hard to isolate.
c. Different data distributions between our train and validation set despite train_test_split with random shuffling.

Observation 3: Of concern was the rapid degradation of performance of ensemble models developed from 2017 data when tested against 2018 data. This is due to data distribution differences across both years. A quick investigation revealed that some of the complaint types and location types that existed in 2017 did not exist in 2018 at all and vice-versa[8].

Complaint types in 2017 but not in 2018	'day care'

[8] See Annex C(a) for code snippet.

Complaint types in 2018 but not in 2017	'bridge condition', 'highway condition', 'calorie labeling'
Location types in 2017 but not in 2018	'Health Club', "Veterinarian's Office", 'Parking Lot', 'Private School', 'Apartment', 'Public School', 'Private House', 'Public Plaza'
Location types in 2018 but not in 2017	'Dentist's Office', 'Highway', 'Roadway Tunnel', 'Cemetery', 'Bridge'

Table 7: Summary differences in categorical data.

In a live system, we could use advanced outlier detection methods (for e.g. Mahalanobis distance) and anomaly methods (for e.g. Density-based clustering) during pre-processing to quickly flag out data points that do not cohere with the training set for model re-training.

7. Conclusion

In conclusion, all models did not perform as expected despite extensive feature engineering and data pre-processing. While ensemble models performed better during the model development, validation and testing stage, model performance degraded rapidly when tested against 2018 data to compute model drift.

Nonetheless, from our error analysis we propose the following suggestions for future modelling:

a. Training on multiple years of data: This ensures that models capture as many complaint types and location types as possible. However, each year has >1.5m complaints so large computational resources are needed.
b. Adopting deep learning approaches: Neural network architectures such as FNN and RNN might be able to better handle high variability data with regularization techniques such as inserting dropout layers to prevent overfitting.
c. Advanced feature engineering: We could develop other salient features such as
    i. Number of open cases each agency has at any point in time
    ii. Distance of case location from closest service center using geo data of service centers
d. Widening the scope of data collection: We feel that the performance of our models could improve with the following data points, though some datasets might not be available, and different agencies might handle cases differently.
    i. Detailed manpower data of agencies at different service centers
    ii. Text data giving details of each complaint
    iii. Average number of rolling cases
    iv. Granular traffic data
    v. Weather data

- END OF REPORT -

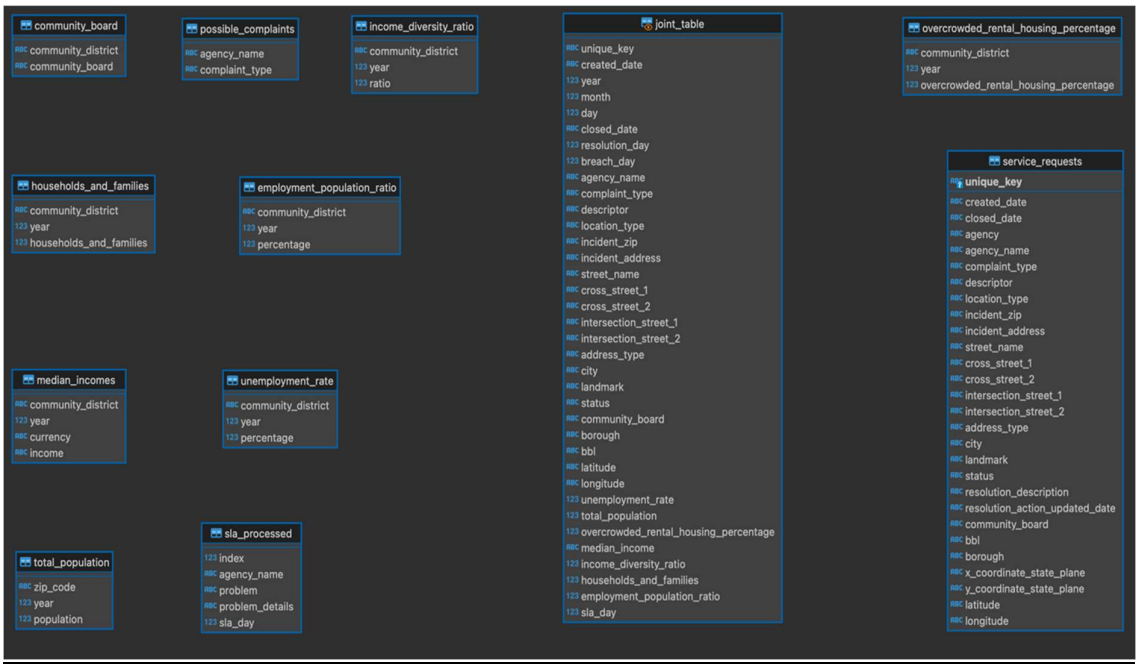
Annex A: Full list of raw features from datasets

1. Datasets and raw features

S/N	Dataset	Features
1.	NY 311 Service Requests	- unique_key - created_date - year - month - day - hour - closed_date - agency_name - complaint_type - descriptor - location_type - incident_zip - incident_address - street_name - cross_street_1 - cross_street_2 - intersection_street_1 - intersection_street_2 - address_type - city - landmark - status - community_board - borough - borough block lot - latitude - longitude
2.	NY Service Level Agreement	- Complaint type - SLA duration
3.	NYC Socioeconomic data	- Total population - Employment population - Median income - Income diversity ratio - Overcrowding rental housing percentage - Unemployment rate

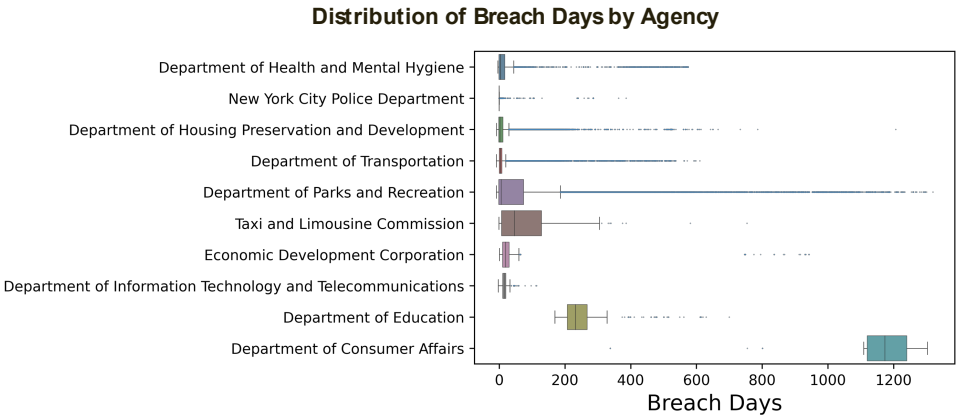
2. ER Diagram

Although no explicit connections are drawn, the diagram below illustrates how the materialized view joint_table is created by joining all other tables.

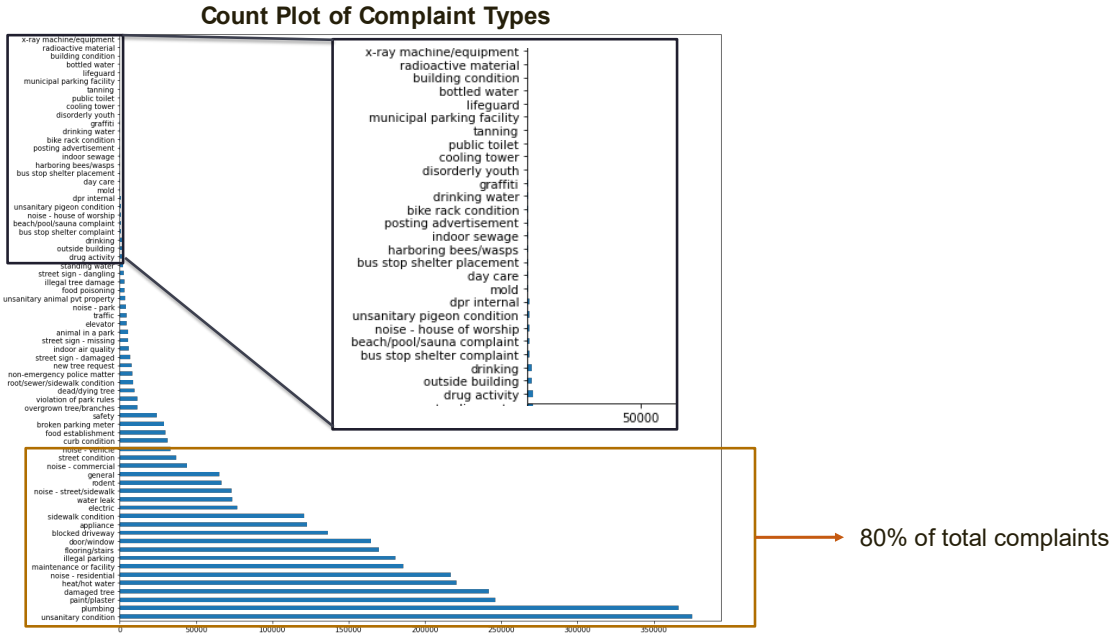


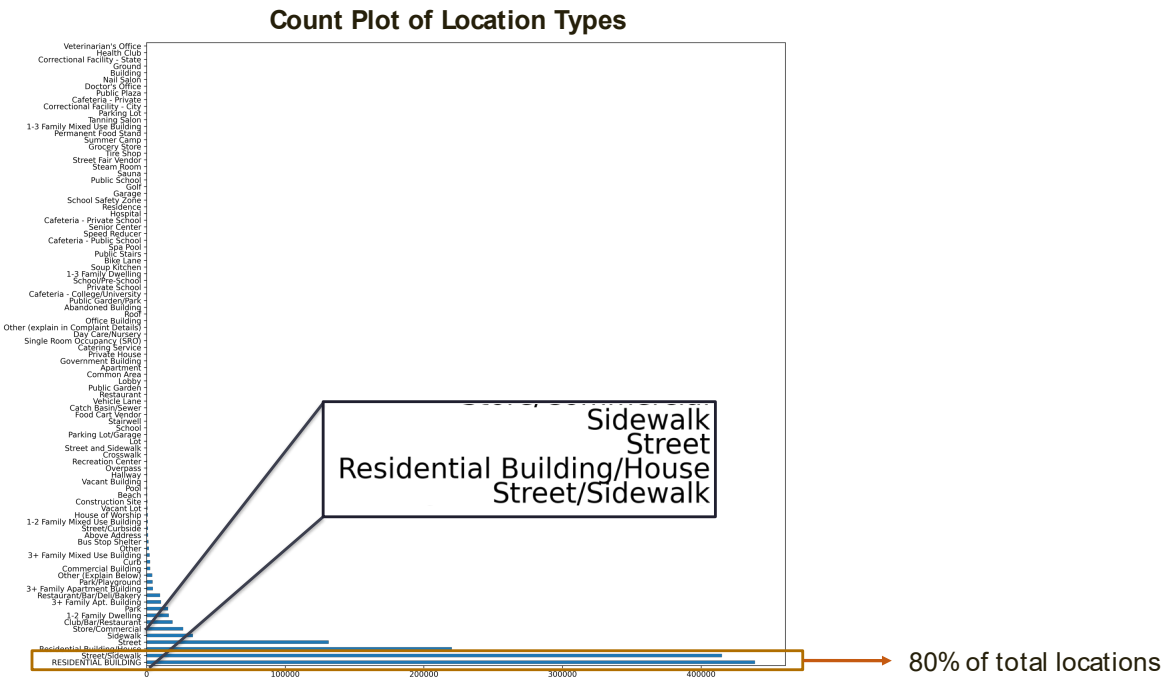
Annex B: EDA Plots, our data in pictures

1. **Large number of outliers.** This suggests that the high variability is a characteristic of non-emergency complaints.



2. **High cardinality of categorical features.** The count plot does suggest that we can explore binning infrequent complaint types into a large bucket.

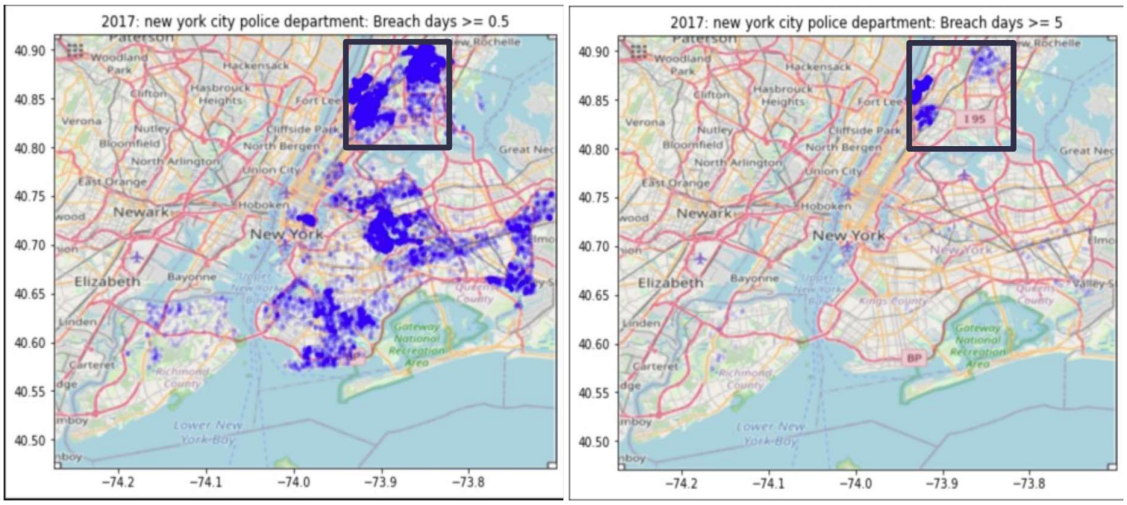




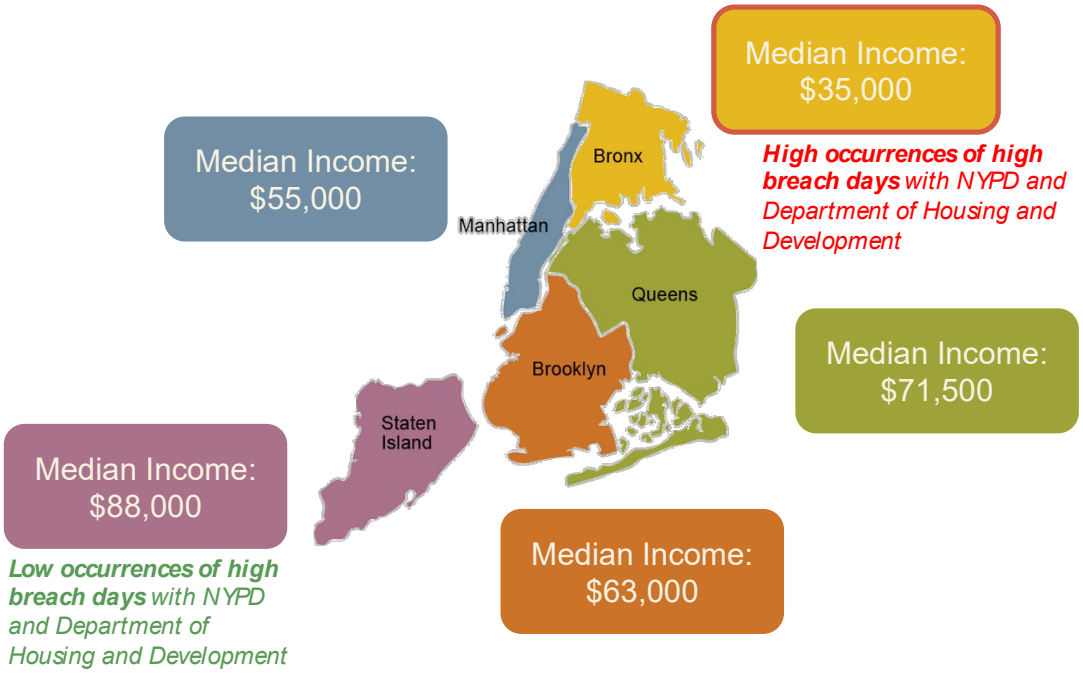
3. Strong association between some categorical variables[9]. This suggests that some variables can be removed to reduce dimensionality.

4. A geo plot of incidents shows **higher breach days at the upper side** of New York. This suggests that binned latitude and longitude can prove to be a useful feature for our model.

[9] New York City comprises of 5 boroughs, with each borough comprising of a few 'cities' and community boards.



5. We noticed possible correlations between socioeconomic data (for e.g. income) and breach days.



Annex C: Code snippets

a. Identification of differences in categorical data between 2017 and 2018

```
[12] import pandas as pd
     df_2017 = pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/311_2017.csv')
```

```
[6] df_2018 = pd.read_csv('/content/gdrive/My Drive/Colab Notebooks/311_2018.csv')
```

```
[8] complaint_not_in2018 = list(set(df_2018['complaint_type'])-set(df_2017['complaint_type']))
    print(complaint_not_in2018)

    ['day care']
```

```
[10] complaint_not_in2017 = list(set(df_2017['complaint_type'])-set(df_2018['complaint_type']))
     print(complaint_not_in2017)

     ['bridge condition', 'highway condition', 'calorie labeling']
```

```
[9] location_not_in2018 = list(set(df_2018['location_type'])-set(df_2017['location_type']))
    print(location_not_in2018)

    ['Health Club', "Veterinarian's Office", 'Parking Lot', 'Private School', 'Apartment', 'Public School', 'Private House', 'Public Plaza']
```

```
[11] location_not_in2017 = list(set(df_2017['location_type'])-set(df_2018['location_type']))
     print(location_not_in2017)

     ["Dentist's Office", 'Highway', 'Roadway Tunnel', 'Cemetery', 'Bridge']
```

b. Heteroscedasticity Test

```
from statsmodels.stats.diagnostic import het_breuschpagan
```

```
labels=['lm_statistic','lm_p-value','F-Statistic','F-Test p-value']
```

```
[63] bp_test_overall = het_breuschpagan(residual,X_test)
     print(dict(zip(labels,bp_test_overall)))

     {'lm_statistic': 14030.824548229864, 'lm_p-value': 0.0, 'F-Statistic': 877.6819366056109, 'F-Test p-value': 0.0}
```