Analysis of Major Crimes in Toronto, Canada

Matthew Chen, Jasper Tsai, Mark Faynboym

June 11, 2023

1 Introduction

1.1 Subject of Interest

Criminology is an important field that utilizes statistics to help us develop strategies for maintaining public safety. The abundance of data available in larger metropolitan cities makes them valuable for studying this subject. In this report, we chose Toronto, Canada to be our subject of study as Toronto is the most populated city as well as the financial and business capital in Canada. The objective of this analysis is to utilize historical data to make observations and predictions about future crimes, which holds considerable significance in its potential application across numerous industries. For example, the ability to find patterns of crime in different parts of the city based on time and type can allow police departments to change their policing strategies based on the parts of the city officers are dispatched to. Also, in another example, overall crime rate of a neighborhood can influence the pricing of a house as crime rate and housing price often times have an inverse relationship. To conduct our analysis, we explore a variety of unsupervised (i.e association rule mining) and supervised learning techniques (i.e. random forest, gradient boosting, logistic regression, etc...), in addition to an interactive exploratory data analysis.

1.2 Data Source

We sourced our data from the Toronto Public Safety Data Portal created by the Toronto Police Department which maintains a public repository of the Major Crime Indicators among other databases of crimes and misdemeanors. The Major Crimes Database we sourced contains roughly 330,000 reported crimes from the 8 year span of 2014 to 2022. Other variables included 33 different location types such as apartment, commercial, etc..., 158 neighborhoods, nautical coordinates and much more.

The variables below are the main features we decided to focus on in carrying out our analysis.
Variable Introduction:

OCC_YEAR: Year Offence was reported
OCC_MONTH: Month Offence Occured
OCC_DAY: Day of the Month Offence Occured
OCC_DOY: Day of the Year Offence Occured
OCC_DOW: Day of the Week Offence Occurred
OCC_HOUR: Hour Offence Occurred
LOCATION_TYPE : Location Type of Offence
MCI_CATEGORY: MCI Category of Occurrence
NEIGHBOURHOOD_158: Name of Neighbourhood using City of Toronto's new 158 neighbourhood structure
LONG_WGS84: Longitude coordinate
LAT_WGS84: Latitude coordinate

For the full detailed description of all variables please refer to the original documentation: Documentation

1.2.1 Data Processing and Feature Engineering

We conduct feature engineering on the data, namely, we process location type so that all public transit locations are binned together. This significantly reduces the cardinality of the location type variable. Additionally, to reflect the cyclic nature of time of day and day of week, we conduct trigonometric encoding on these variables and include both the sine and cosine components. We also capture the interaction between the trigonometric encoded time variables and location by multiplying these variables with the longitude and latitude coordinates. Finally, to capture the potential effect of holidays, we add a variable that indicates whether or not a reported crime occurs on a Canadian holiday or not.

2 Methods

2.1 Association Rules

Association rule learning is an unsupervised machine learning method for discovering relationships between variables. This method originated from Market Basket Analysis which helped show associations of of items bought in a grocery store.
In the general sense, the association between some variables X and Y is written as follows:

$$[X \Rightarrow Y \quad \text{with} \quad \text{support}(X \cup Y) \quad \text{and} \quad \text{confidence}(X \Rightarrow Y)]$$

We call X the antecedent and Y the consequent. The support$(X \cup Y)$ represents the support of the combined itemset $X \cup Y$. This is estimated by the proportion of occurences in the dataset where both X and Y appear jointly. Intuitively, this measures the frequency of the occurrence of both $X$and Y together. confidence$(X \Rightarrow Y)$ represents the confidence of the association rule $X \Rightarrow Y$. This is estimated as the proportion of transactions containing $X \cup Y$ to the transactions containing X, estimating the conditional probability of Y occurring given the presence of X. Specifically, the confidence of $(X \Rightarrow Y)$ is written as

$$C(X \to Y) = \frac{T(X \to Y)}{T(X)}$$

where T is shorthand for the support. Finally, the lift measure in association rule mining provides an indication of the strength of an association rule.

$$L(X \to Y) = \frac{C(X \to Y)}{T(Y)}$$

For our dataset, we will use association rule mining to hopefully discover relationships between different crimes types and our predictor variables relating to the time and location of a reported crime. For example, we can search for conditions that are associated with an assault crime, which would be important information for informing individual safety.

2.2 Multi-Class Logistic Regression

In multi-class logistic regression (using a multinomial loss), we model the log ratios between different estimated probabilities as linear relationships with the predictor variables. Specifically, the we can write the multiclass probabilities with the following system of K equations, where $\mathbf{w}_k^T$ is the vector of weights for the k^{th} class including the intercept. Note that this assumes that the observation vectors $\mathbf{x}$ have a leading 1 that corresponds to the intercept.

$$\mathbb{P}(y = k \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^T \mathbf{x})} \; for \; k = 1, ..., K-1$$

$$\mathbb{P}(y = K \mid \mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^T \mathbf{x})} \; for \; k = K$$

We used multi-class logistic regression as a good initial model to try because it is easier to fit compared to many other models. However, the drawback is that the model assumes a linear relationship between input features, which may not be the case for our data. For example, we hypothesize our prediction based on Longitude and Latitude coordinates do not assume a linear relationship. This would mean that as longitude increases, the probability for a certain crime type also increases, which is not realistic.

2.3 Random Forest

Random Forest is an ensemble method that averages many different decision trees (based on bootstrap samples of the data and the features) to obtain a robust solution that reduces variance. For a classification task, the random forest averages the class probability estimates for each decision tree in the ensemble. Specifically, for a model with M decision trees, the estimated probability of predicting the class k can be written as

$$\hat{\mathbb{P}}(y = k \mid \mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \hat{\mathbb{P}}_m(y = k \mid \mathbf{x})$$

In general the predicted class is determined by majority voting or by selecting the class with the highest probability.

Random Forest has many benefits that make it a reasonable next model to try in our data after logistic regression. For example, Random Forest performs automatic feature selection, is robust to outliers, and discovers non-linear relationships. Since Random Forest averages many different decision trees, it is also not prone to overfitting and does not require much hyperparameter tuning, if at all.

Additionally, there are methods that deal with poor performance on imbalanced data that can be easily implemented on a forest. For example, in sample class weighting (whether on the full dataset or on each bootstrap sample), each constituent decision tree gives more priority to the minority class when calculating the impurity scores. Another strategy to deal with class imbalance is oversampling, that is, artificially balancing the dataset by randomly drawing additional observations from the minority classes. In this report, we end up using both strategies to try to deal with class imbalance in the data.

2.4 Gradient Boosted Trees(xgboost)

Gradient boosting is a useful method because every subsequent model trains on the errors of the previous model. In short, we fit our training data with an update of our m-th model iteration, $f_m(x)$, and calculate the gradient of the training data which is some constant c with $h_m(x)$ which is also representative of our error.

$$f_{m+1}(x_i) = f_m(x_i) + c * h_m(x_i)$$

On the subsequent page, we can find Algorithm 1, which outlines a generalized gradient boosting algorithm. This is useful for supervised learning. When one wants to focus on bettering the performance of their models, this method is useful because it updates the function by improving the model based on the mistakes of the previous model in the sequence of models.

Algorithm 1 General gradient boosting

$f_0(x) \leftarrow \underset{\gamma}{\text{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

for $m = 1, ..., M$ **do**

- Compute the pseudo-residuals:

$$r_{im} = -\left(\frac{\partial L(y_i, f_{m-1}(x_i)}{\partial f_{m-1}(x_i)} \right)$$

  for all $i = 1, ..., n$

- Fit a weak learner $h_m(x)$ to the pseudo-residuals using the training set $\{x_i, r_{im}\}_{i=1}^{n}$
- Compute γ_m using line search:

$$\gamma_m = \underset{\gamma}{\text{argmin}} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$$

- Perform the update step:

$$f_m(x) = f_{m-1}(x) + \gamma_m h_m(x)$$

end for
return $f_M(x)$

2.5 Decision Trees for Binary Classification

Decision Trees make predictions by partitioning the input data into several discrete rectangular regions, which can be represented in the form of a "tree" with a sequence of binary splits.

Decision Trees have several advantages, namely its high interpretability, as the resulting tree structure can be easily visualized and understood by humans. They can handle both numerical and categorical features and can capture nonlinear interactions between attributes.

In this project, note that we did not have the best performance on the minority crimes even after trying various methods to remedy the issue. That is, we found that our best model was mostly only skilled at predicting assault crimes. Thus, we turn to decision trees to focus on a binary classification case - assault or not. The choice of decision trees comes from the desire to get some interpretability from out data, and fundamentally learn more about assault crimes while still being able to capture nonlinear relationships.

3 Exploratory Data Analysis

We begin our EDA with looking at a plot of the count of each type of crime over a continuous time frame of the last 8 years. By doing so, we notice several interesting observations that are shown in Figure 1.
First assault incidences behave in a cyclical pattern with the lowest points of each year occurring at the very beginning of the year with the exception of the year 2020 when COVID-19 quarantines were put in place. Notably, after the pandemic, we observe a much higher fluctuation of assault compared to pre-pandemic patterns.

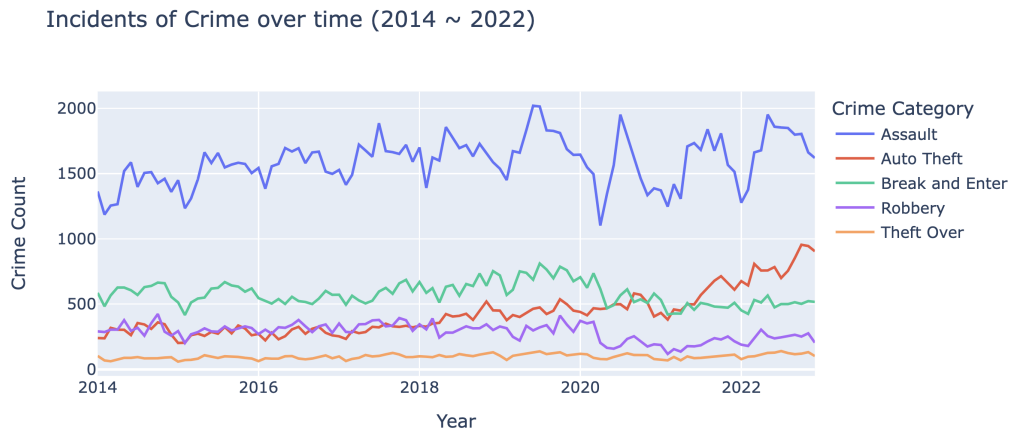


Figure 1: Crime Count over time

Furthermore if we remove the assault category which appears much more frequently than the other crime categories, we are able to see further patterns more clearly.

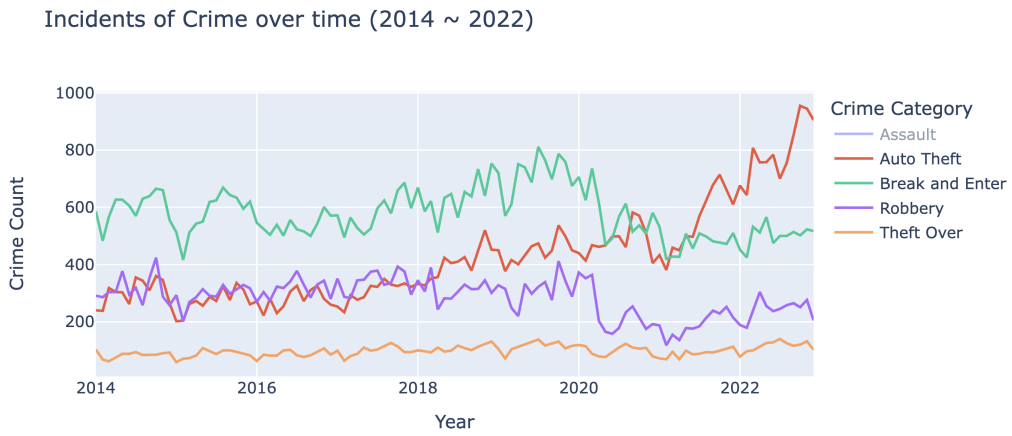


Figure 2: Crime Count over time w/o assault

Specifically, we notice in Figure 2 that the start of the pandemic marks a steep decrease in Break and Enter and Robbery crimes dramatically as well as a steady increase of auto thefts. However auto thefts may more than likely be attributed to another event that occurred at the same time. For example, the semiconductor chip shortage that spiked car prices may be the cause of this observation though this is a mere speculation.

We shift to focusing on the crime count based on hour of the day. In Figure 3, we look at our crime counts by hour of the day for 2019 and 2022 and notice that aside from the count of the day changing between crimes as mentioned already, we see a dramatic shift in auto theft distribution into the latter portion of the day.

Figure 3: Crime Count by hour of the day

Next, We generated Figure 4 to depict the frequency of crimes in specific location types with significantly higher crime counts compared to others. Notably, Apartments exhibit a high incidence of assault, followed by Streets, Roads, Highways, and Single Homes. Additionally, as expected, Parking Lots account for the majority of auto theft cases.

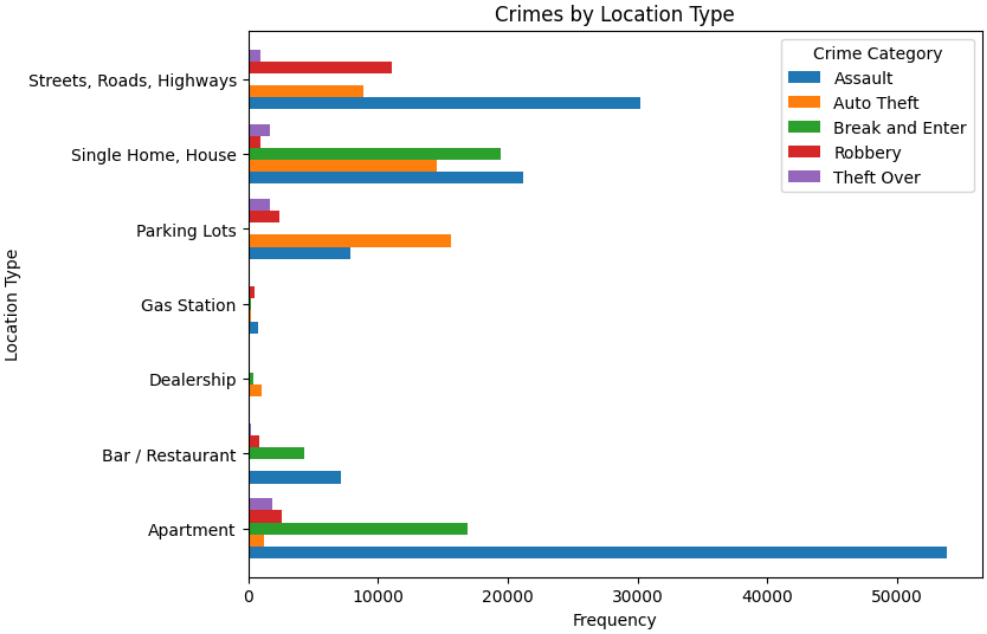
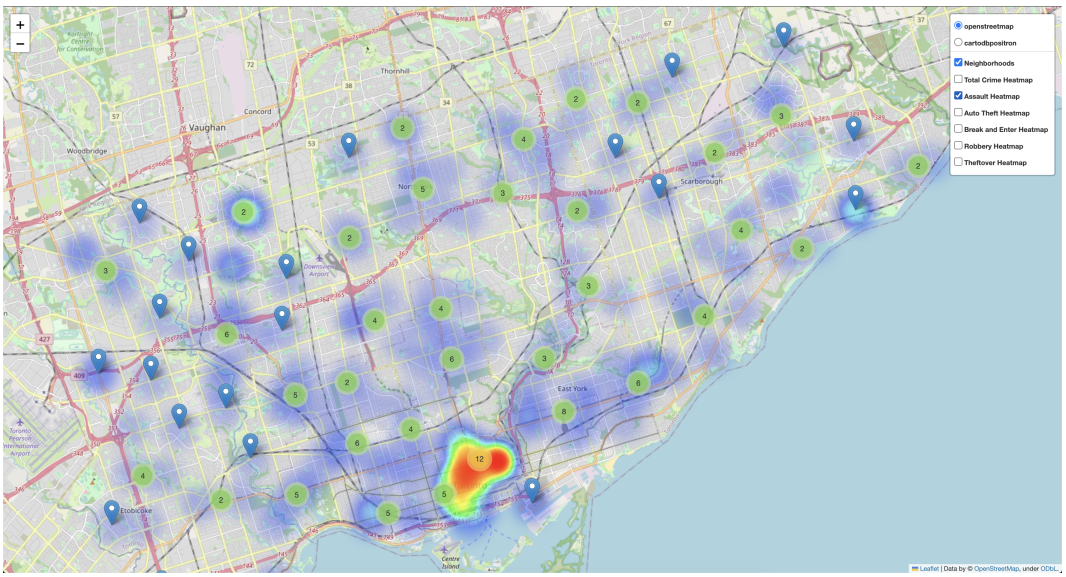
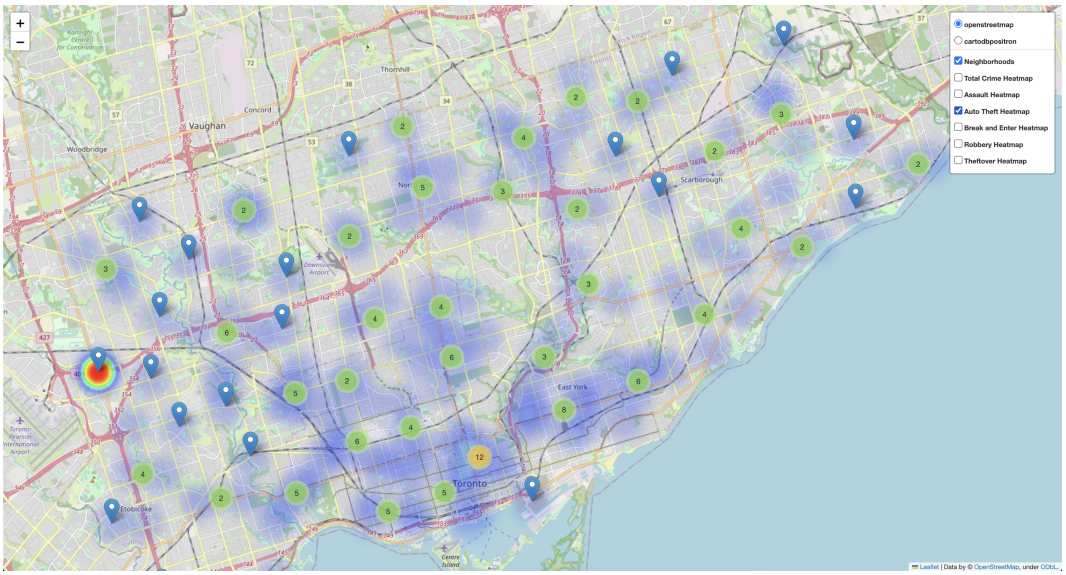


Figure 4: Crime by Location Type

Below we show a heat map (5) created to show both the incidence count as well as frequency of crime type by neighborhood. Note that can also select different crime types separately. Immediately, we notice that assault is concentrated around the downtown area of Moss Park whereas auto theft, for example is more concentrated around the West-Humber Clairville neighborhood. An issue that arises is that theft over $5000 seems to be concentrated around the same parts that assault and auto-theft are, so we suspect a prediction based on location alone may pose a challenge for these crime categories.



(a) Assault Crime Distribution



(b) Auto Theft Crime Distribution

Figure 5: Heatmaps of crime types by location

4 Results

In this section, we evaluate the performances for the models we discussed previously and walk through the decision making process to provide reasons for implementing each method. All of the models' performance evaluations were done using a holdout cross validation with 70% of the data used in our training set and 30% as our testing set, since this provides us a way to quickly evaluate performance even for the models that were more computationally expensive to train. However, hyperparameter tuning was done using 5-fold cross validation on the training set, using either a random search (xgboost) or full grid search (decision tree) on the parameter space depending on how computationally expensive the model in question was. Note that in the case of training xgboost, we choose the number of boosting rounds based on early-stopping by monitoring performance on a separate validation set.

Overall we model our performance using overall classification accuracy, precision, recall, and the f1 score. Precision is defined as the number of predicted positive values that are actually positive, and recall is the number of actually positive values that were predicted positive. Note that the f1 score is simply the harmonic mean between precision and recall, and summarizes the two scores with equal contributions.

4.1 Association Rules

In the association rule analysis, we find associations that have a support greater than 0.02 and confidence greater than 70%. Note that these parameters were chosen after manual tuning. The resulting association rules are shown in Table 1 where we find reasonably high confidence and lift values across the board. Interestingly, we find a strong association between apartments and assault crimes, particularly on Saturday and Sunday. These results were consistent with our findings in the EDA section.

Table 1: Association Rule Mining

Association Rule Mining				
antecedents	consequents	support	confidence	lift
Apartment, Sunday	Assault	0.028	0.768	1.436
Apartment, Saturday	Assault	0.028	0.749	1.400
Apartment	Assault	0.167	0.704	1.317

4.2 Multi-Class Logistic Regression

Now we move on to present results from our supervised learning methods. First, recall that we implemented Multi-Class Logistic model as the baseline in our analysis because it can be fit more quickly compared to other models but assumes a linear relationship. The model performance displayed in Table 2 showed an overall accuracy score of 61% but upon closer inspection, only assault crime had a good f1, precision, and recall. The other crime categories had poor predictive performance.

Table 2: Multi-Class Logistic

Multi-Class Logistic					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.733	0.533	0.442	0.184	0.001
precision	0.637	0.576	0.535	0.575	0.500
recall	0.862	0.497	0.377	0.109	0.000
accuracy	0.614				

4.3 Random Forest

Next, we chose to implement random forest, because random forest is capable of finding nonlinear relationships that were not possible to model with logistic regression. Additionally, it is a good next model to try

because it requires little to no hyperparameter tuning.

To process our data, we employed two encoding schemes before implementing random forest: One Hot Encoding, which converts each level of a categorical feature into binary indicator variables, and Ordinal Encoding, which assigns an integer to each level of the categorical feature.

Additionally, the feature importance plot generated from ordinal encoded data allows us to extract valuable insights for the modeling chain, unlike the one-hot encoded data which makes the plot difficult to interpret. This is shown in Figure 6 which suggests that the location type, the GPS coordinates, and the interaction between time of day and location are some important variables to consider.

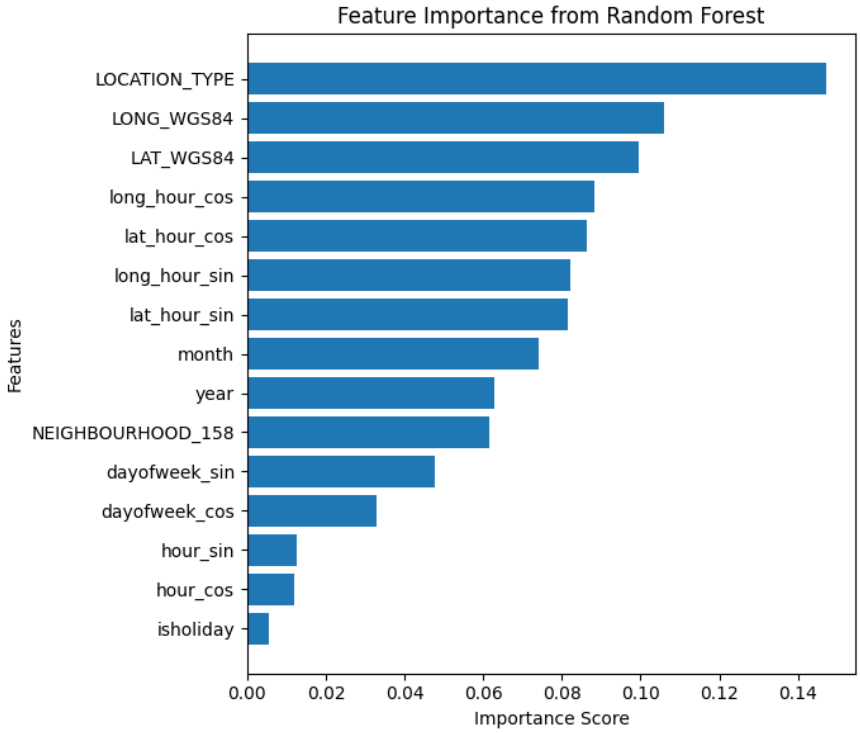


Figure 6: Feature Importance

From Tables 3 and 4, we compared the results and determined that the one hot encoding scheme produced a slightly better result. Thus, going forward in the subsequent sections we will exclusively use the one-hot encoding scheme when appropriate.

Table 3: Random Forest (One Hot Encoded) Performance

Random Forest Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.773	0.622	0.566	0.496	0.056
precision	0.706	0.644	0.633	0.654	0.201
recall	0.855	0.601	0.512	0.399	0.032
accuracy			0.680		

Table 4: Random Forest (Ordinal) Performance

Random Forest Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.766	0.602	0.538	0.456	0.056
precision	0.686	0.643	0.629	0.675	0.244
recall	0.866	0.566	0.470	0.345	0.032
accuracy			0.670		

Despite the fact that random forest's accuracy using one hot encoded data increased to 68% compared to the baseline logistic regression (61%), we notice that the same problem persists in which only the assault crimes had good prediction accuracy. This disparity is especially true for thefts over $5000. This is possibly due to the class imbalance in crime types, which are shown in the following pie chart (Figure 7).

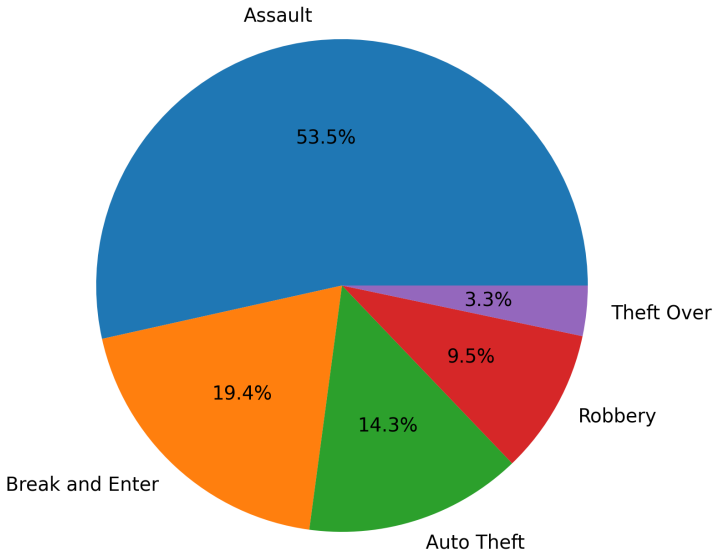


Figure 7: Class Balance

Note that the pie chart in shows that assault crimes occupy 53.5% of the data, and that thefts over $5000 represent only 3.3%. Going forward we will modify the random forest model using various methods such as class sample weighting and oversampling to account for the imbalance.

4.4 Weighted Random Forest

The first aforementioned strategy we employed to combat class imbalance is fitting a weighted random forest. Weighted random forests weight the impurity calculations for the constituent decision trees toward the minority classes. Intuitively, this method penalizes misclassifications on the minority classes more.

Unfortunately, even after using class sample weighting, the problem of poor performance on the minority class persists. The performance of our weighted random forest is summarized in Table 5.

Table 5: Weighted Random Forest Performance

Weighted Random Forest Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.772	0.619	0.561	0.500	0.057
precision	0.701	0.647	0.637	0.657	0.206
recall	0.857	0.593	0.502	0.404	0.033
accuracy	0.679				

We also tried fitting a random forest with bootstrap weighting. Recall that this is similar to the weighted random forest discussed above, but instead of calculating the sample class weights based on the entire dataset, we find unique weights for each constituent decision tree based on its respective bootstrap sample. Table 6 displayed similar results as before, and the issue of poor performance on the minority classes remain.

Table 6: Bootstrap Weighted RF Performance

Bootstrap Weighted RF Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.771	0.620	0.562	0.503	0.055
precision	0.702	0.645	0.636	0.658	0.196
recall	0.855	0.596	0.503	0.406	0.032
accuracy	0.679				

4.5 Minority Class Oversampling RF

Finally, another way that can deal with class imbalance is oversampling the minority class. To do so, we need to make two main assumptions. The first assumption is the oversampling minority classes are exchangeable with the existing observations. In other words, the synthetic minority cases should come from the same underlying distribution as the existing minority class observations. The other assumption is that synthetic minority cases should be independent of each other so that we do not learn spurious correlations between them. Further, note that by oversampling the existing minority cases, we also give more weight to the noise inherent in the existing cases. This may lead to the risk of overfitting, and reduce the generalization ability of the model.

Table 7 showed that the individual class prediction performance generally did not improve. From the table, note that only the out of sample performance for Theft Over $5000 improved, but not to acceptable levels.

Table 7: Minority Class Oversampling RF Performance

Minority Class Oversampling RF Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.758	0.627	0.585	0.502	0.096
precision	0.739	0.606	0.589	0.545	0.176
recall	0.777	0.649	0.580	0.465	0.066
accuracy	0.667				

Overall, class sampling weighting and oversampling did not fix the issue of poor performance for the minority classes. This suggests another method might be more suitable.

4.6 Gradient Boosted Trees(xgboost)

Here, we tried a different machine learning algorithm to deal with the low performance on the minority classes. Gradient boosting may be a good algorithm to consider because of its sequential nature, where each subsequent model focuses more on the previous model's missclassifications. This is usually a good way to

reduce bias, and in our case, it may help with the low performance on the minority classes.

The result of XGBoost is shown in Table 8 and unfortunately there is no noticeable improvement compared to the random forests.

Table 8: Gradient Boosted Trees Performance

XGBoost Performance					
	Assault	Auto Theft	Break and Enter	Robbery	Theft Over
f1	0.767	0.605	0.576	0.356	0.058
precision	0.691	0.631	0.638	0.610	0.296
recall	0.862	0.581	0.524	0.252	0.032
accuracy	0.670				

4.7 Comparison of Muticlass Models

In this section, we plotted each method's overall accuracy (Figure 8 and performance on the holdout test data for each crime category (Figure 9).

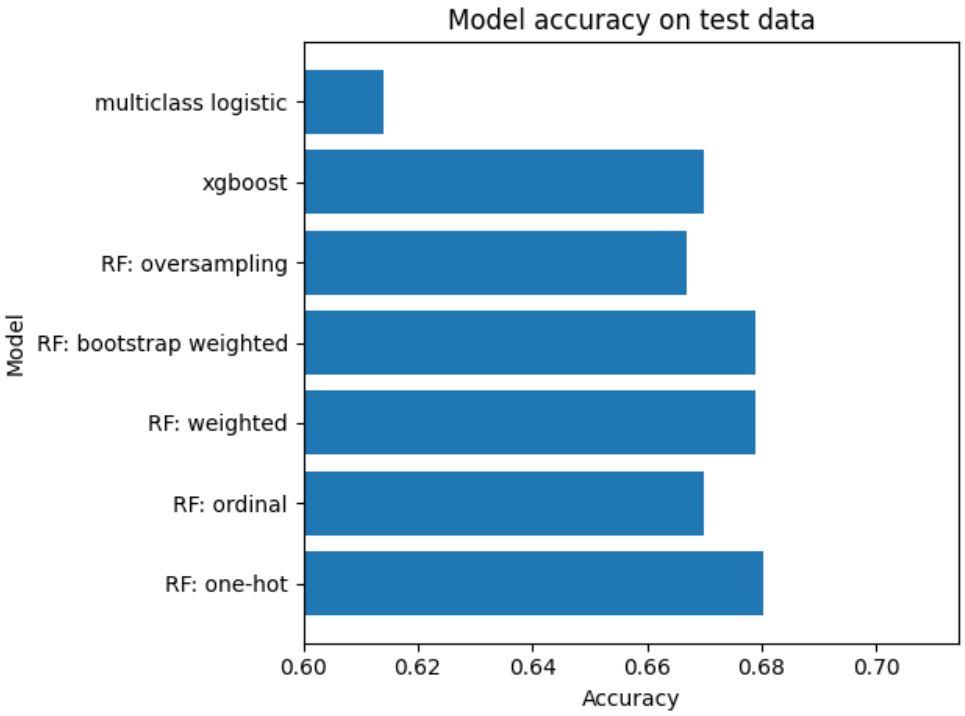


Figure 8: Overall Accuracy for Each Model

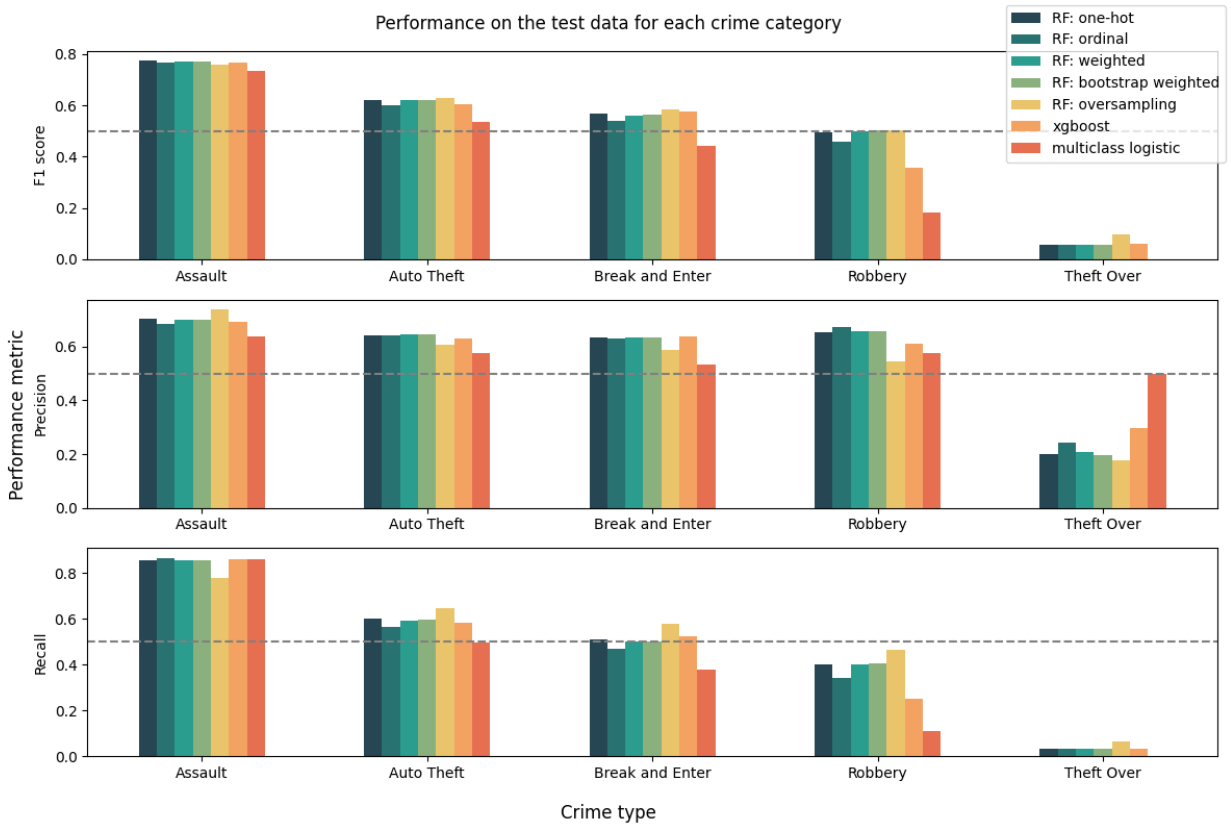


Figure 9: Performance Measures for Each Model by Crime Category

Overall, we see that F1 scores (which equally balances precision and recall) are only high for assault crime types, but there is not much predictive skill for the other crimes types because the F1 score is too close to 50% (which would be the score of a classifier with no skill). Performance for theft over $5000 (the smallest minority class) is especially low. Even if the overall accuracy of the model is reasonably high, but this is only because assaults (which are predicted with the highest accuracy) occur the most in the dataset.
The performance on the minority classes (everything except assualt) did not improve much after weighting the random forest, nor did it appreciably improve after oversampling. After this and trying xgboost (gradient boosted trees), which incrementally weights incorrectly classified observations and generally adds more nonlinear capacity compared to the random forest, we find the same issue persists. Thus, it is reasonable to conclude that the predictor variables do not have enough power to discriminate between the different classes.

4.8 Decision Trees for Binary Classification

Since the models above all have high performance on predicting assault crimes, we suspected that a model would able to accurately distinguish between assault and non-assault crimes. Thus, we focus on a binary classification based on this setting. Recall that the choice of decision trees comes from prioritizing interpretability as well as nonlinear modeling. After conducting a grid search to tune hyperpameters such as tree-depth, we obtain a final model with reasonably high performance. The accuracy, precision, recall, and f1 scores of the final tree are all about 70%. These are summarized in Table 9.

Table 9: Decision Tree Performance

Decision Tree Performance			
accuracy	f1	precision	recall
0.693	0.719	0.703	0.736

5 Conclusion

In this project we explored different patterns of crime in Toronto, Canada. After exploring the geographic densities of different crimes, we find that the neighborhood Moss Park (near downtown) has high occurrences of assault, and that West Humber-Clairville has high occurences of auto-theft. The association rule analysis reveals a strong association between assault crimes and apartment locations, especially on the weekend. In our predictive modeling section, we fit multi-class logistic regression, random forest, and gradient boosted trees (xgboost) to predict crime type based on time and location variables. However, even after applying class sample weighting and oversampling, we obtained low performance on the minority crime types, especially for theft over $5000. Performance was high for assault crimes, so we moved on to building a binary classification model to predict assault crimes. By doing so, we finally obtain an interpretable decision tree model with reasonably high (70%) accuracy, precision, recall, and f1.

6 References:

Hastie, T., Tibshirani, R., & Friedman, J. (n.d.). Elements of statistical learning: Data mining, inference, and prediction. 2nd edition

Major crime indicators open data. Toronto Police Service Public Safety Data Portal. (2023, March 31) https://data.torontopolice.on.ca/maps/major-crime-indicators-open-data

7 Repository

All python notebooks used for this project can be found in the following GitHub repository:
https://github.com/Matt2371/Toronto-Crimes