Data Mining Project on the Boston Crime Dataset

Hunter Boles

Montana Tech

Author Note

Completed for CSCI 347 – Data Mining at Montana Tech

Abstract

Throughout history, crime has shaped cultures. It could even be said that modern cultures would not have developed without the need for laws and governments to enforce them. For that reason, as data analytics and mining has evolved, data scientists have analyzed crime datasets to find patterns. In the last few years, some cities have opened their crime datasets to the public. With the help of Analyze Boston, Boston Massachusetts has published the Boston Incident Report Dataset that contains approximately 300,000 entries from June 14th, 2015 to September 3rd, 2018. The goal of the research was to find the parameters that predicted the type of crime. A variety of data mining and machine learning techniques were applied, and the best-fitting model was XGBoost with a max depth of 10 and a learning rate of 0.1. It provided around 18% accuracy across 59 unique classes.

Keywords: Data Mining, Boston Crime Dataset

Data Mining Project on the Boston Crime Dataset

Throughout history, crime has shaped cultures. It could even be said that modern cultures would not have developed without the need for laws and governments to enforce them. For that reason, as data analytics and mining has evolved, data scientists have analyzed crime datasets to find patterns. In the last few years, some cities have opened their crime datasets to the public. With the help of Analyze Boston, Boston Massachusetts has published the Boston Incident Report Dataset that contains approximately 300,000 entries from June 14th, 2015 to September 3rd, 2018.

Analysis of the Problem and Research Goals

The hope of the research is to answer whether it is possible to predict what crimes are likely to happen in certain areas given a set of characteristics. The benefits of answering this question is that it can help to determine incident hot spots and what incident is likely to happen in a given area. For example, certain areas may be notorious for car crashes, and another area may be a hub for violent crime. Both have different needs from the police force, and an answer will help to allocate police officers more efficiently. The question ties in many smaller questions and opens many more when the analysis is completed. Some examples are what effect the time of day has on crimes or how does the time of year affect the frequency of crimes.

Exploratory Data Analysis

Preliminary Variable Analysis

To start out the research, the variables were studied to note any potential red flags for machine learning algorithms, such as null values or inconsistent formatting. The Boston Police Department collected 11 variables listed in Table 1.

Variable	Description
Incident Number	A unique identifier for the incident
Offense Code	A numerical code of the incident
Offense Code Group	A short name for the incident
Offense Code Description	A longer description of the incident
District	The city district that the crime was in
Reporting Area	A location variable that starts with lower-numbered areas in the northeast and higher in the southwest. There are a few thousand
Shooting	Indicates whether there was a shooting. Is a “Y” if there was, and null otherwise.
Occurred on Date	Date and time of which the incident occurred
UCR Part	A larger group of crimes, defined by the Uniform Crime Reports (UCR)
Street	Street of the incident
Location	The latitude and longitude of the incident

Table 1: Variables found in the dataset

In addition to the 11 main variables, there are 7 derived variables, such as year, hour, and day of the week, that can be found through the other 11 variables. They were included to help in analysis.

Throughout the dataset, the values were mostly consistent. The offense code group column needed minor reformatting so that there was a consistent capitalization pattern, as there were some groups that had minor capitalization differences and therefore were not in the same class value.

After looking at values within each column, the next step was to make note of where there were null values. This is another red flag for certain variables, which may not have enough values to provide useful information to the model.

Variable	Number of Null Values
District	1,765
Shooting	318,054
Street	10,871
Lat/Long	19,999
UCR Part	90

Table 2: Null value analysis

Shooting has a disproportionate number of null values. However, this is not an issue as null denotes that no shooting happened in the incident. No other variable has a significant number of null values (<10%), so individual values can be omitted if needed.

Data Analysis

After performing an analysis of the variables, a data analysis was performed manually to see if any patterns were blatantly obvious as well as getting an understanding of the data given. The first visualization mapped out all incidents by district to get an idea of what the location looked like, and to find any inconsistencies in district reporting.

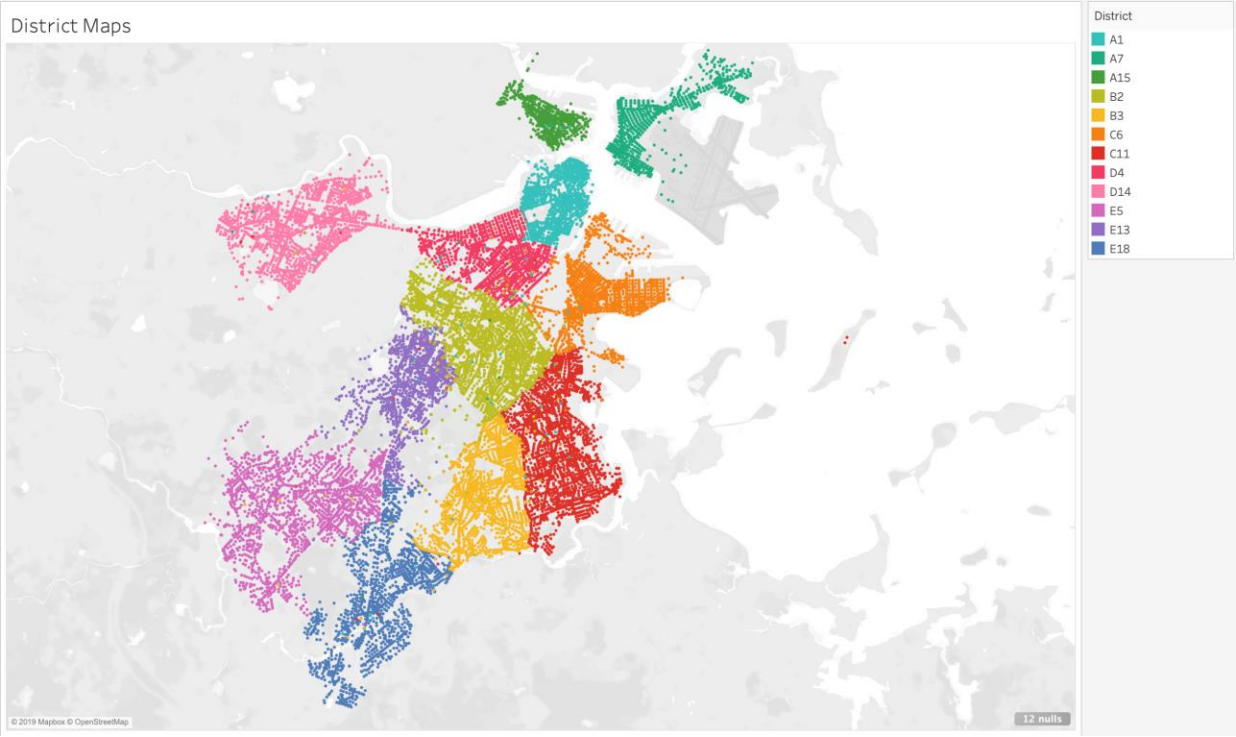

Figure 1: Districts in boston

For the most part, districts have been consistently reported. However, there are places where the district has been mis-reported, as noticed by certain dots being the wrong color in the zones. For this reason, it may be important to use latitude and longitude instead of districts when attempting to predict crimes.

It also became important to ask what crimes were being committed. There were 65 unique classes and determining top incidents as well as the severity of the crime using the *UCR_Part* variable could be useful information.

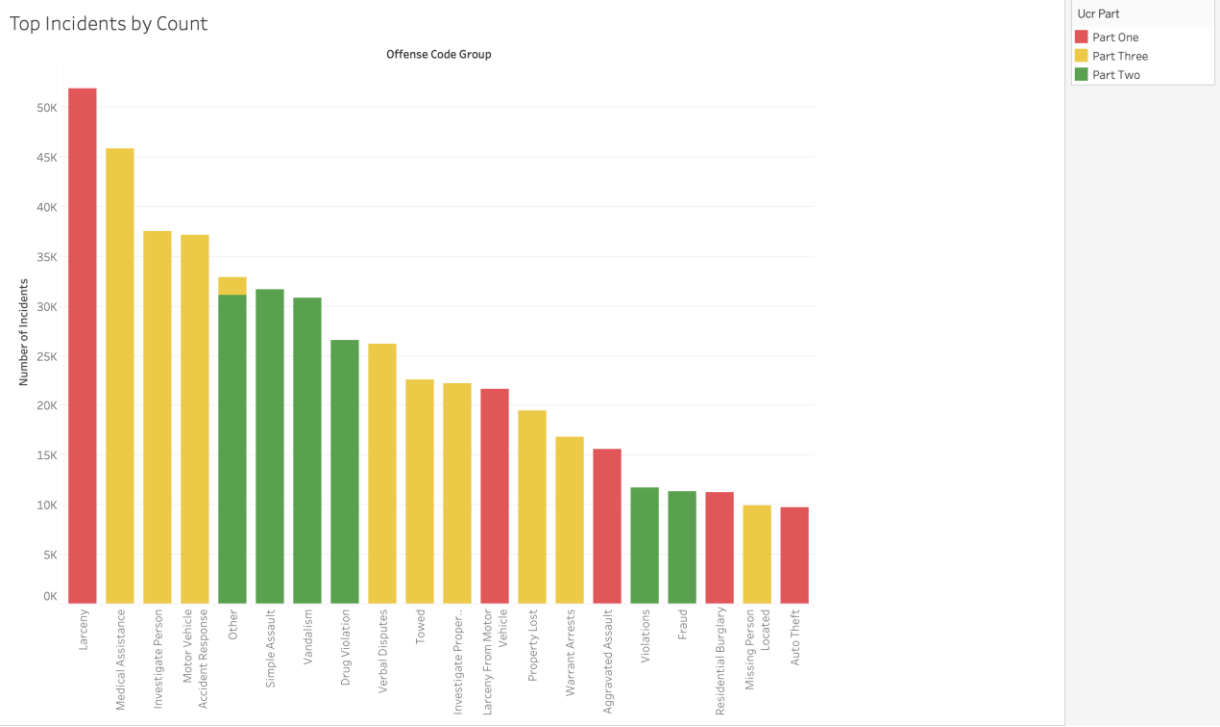

Figure 2: Number of incidents versus offense code group

As noticed, a lot of the incidents are *Part Two* and *Part Three*. These are lower severity crimes than *Part One* crimes, which include homicide, assault, and larceny (theft of personal property).

The *Shooting* variable was unlike many of the other variables since it had two values, and many were null. Thus, it became important to investigate it a little deeper since it could be useful to a machine learning model.

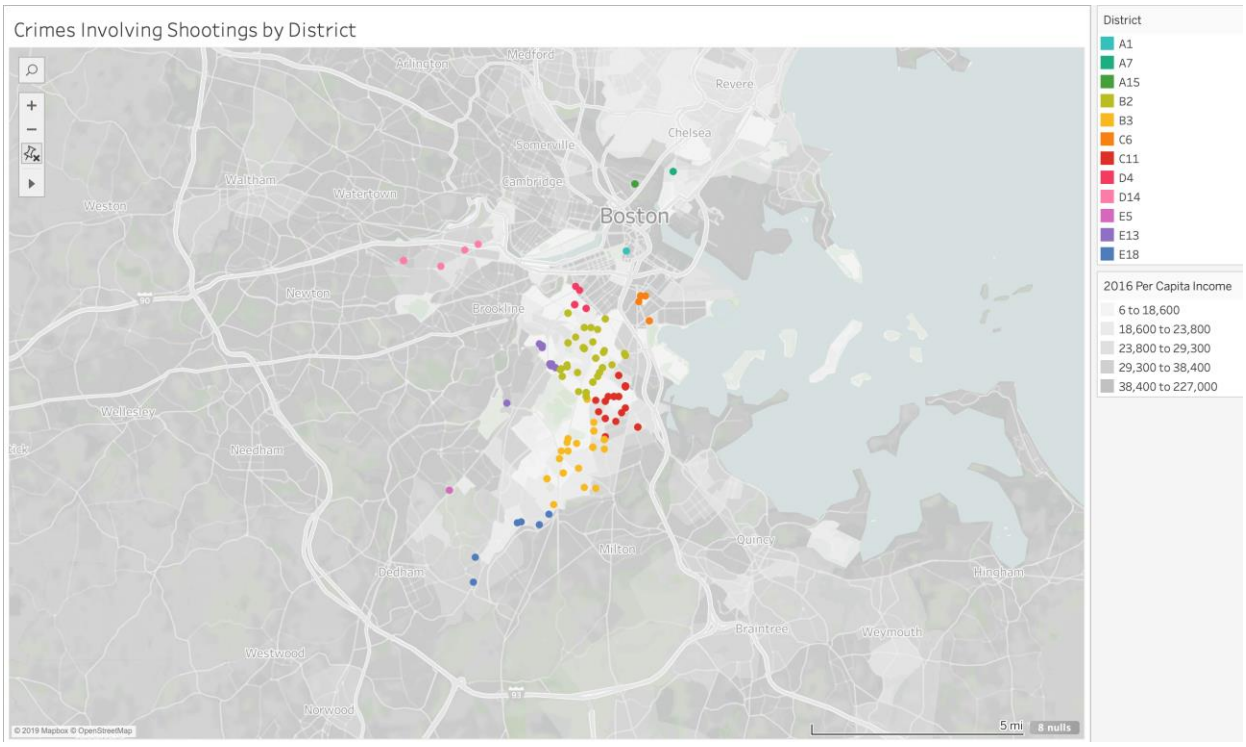

Figure 3: Shootings by district

From the chart, it becomes obvious that shootings nearly exclusively happened in low-income areas. Tableau provided the functionality to use an overlay, which meant that another dataset would have to provide the income information if it was to be used for a machine learning model. Due to time constraints and the complexity of mapping a latitude/longitude location to a census district, it was infeasible to add income to the dataset. From there, it was time to ask the question on what crimes were associated with a shooting.

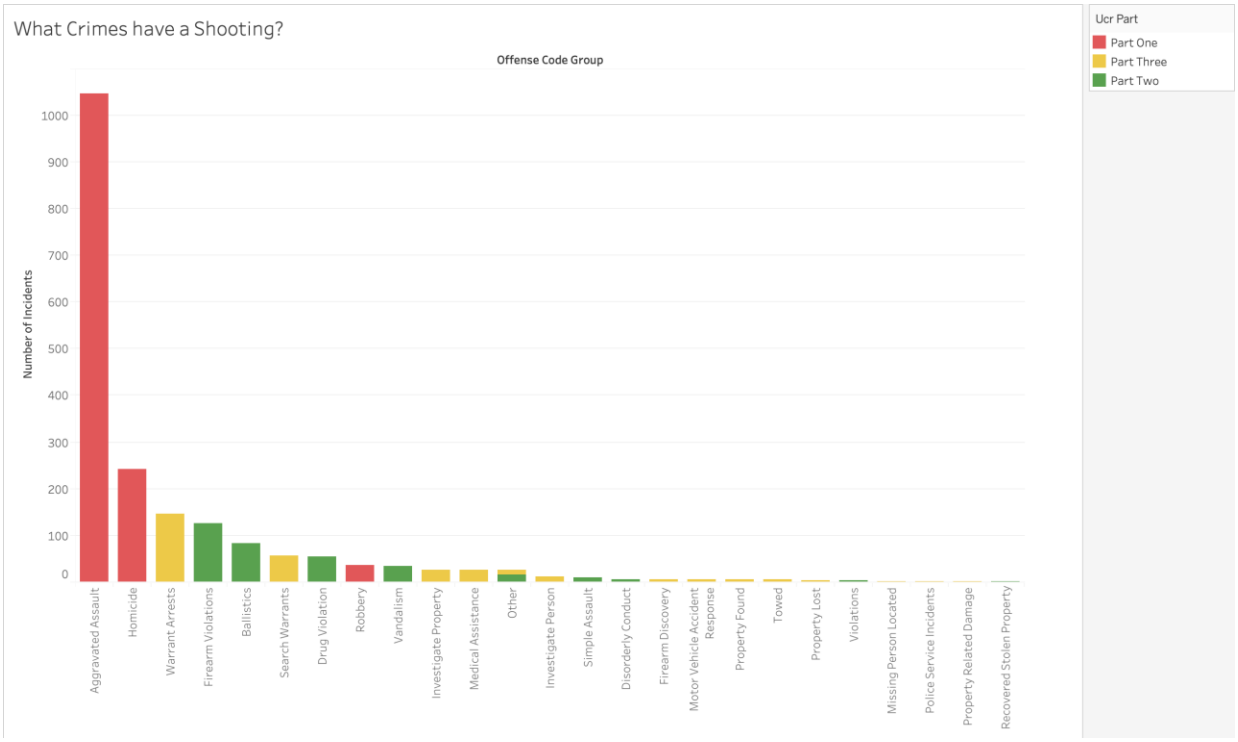

Figure 4: Number of incidents versus offense code group with a shooting value of 'Yes'

By far, aggravated assault was the most common in Figure 4. However, it was clearly not the only value, and thus necessary to keep around. Since shooting was such a black-and-white difference between high income and low income areas, Figure 5 was made to show the five most crime-ridden districts, and see how different crimes were committed in the different areas.

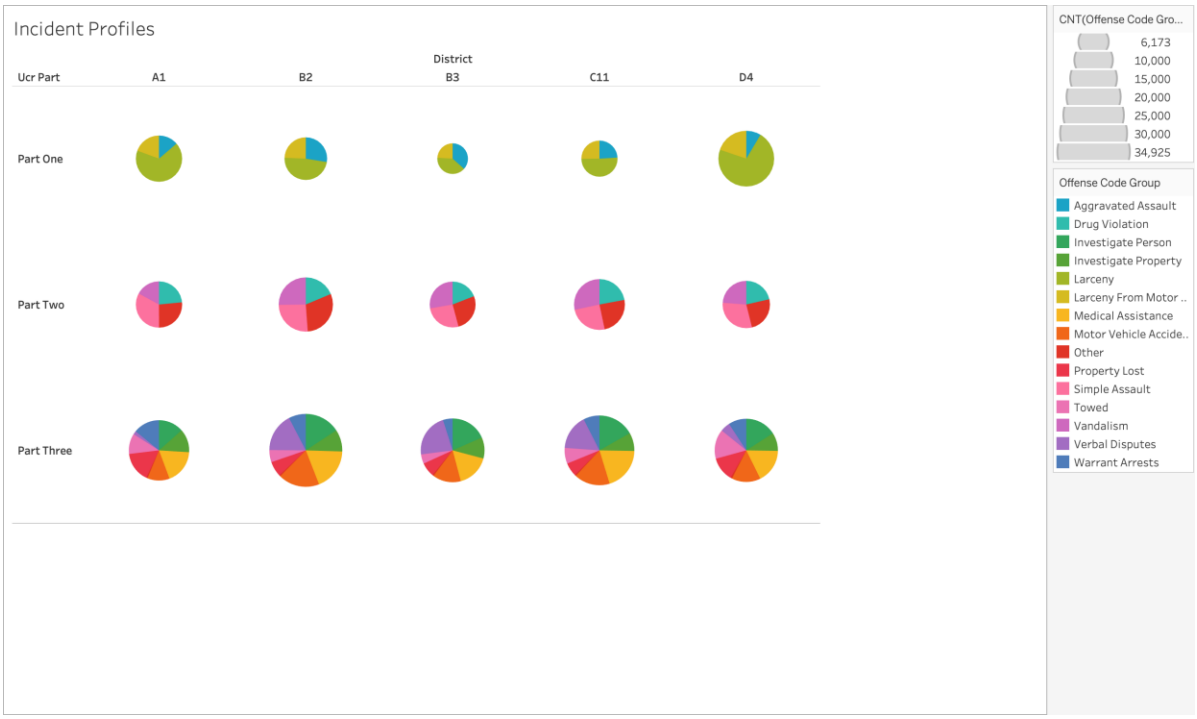

Figure 5: Incident profiles by district, count, and crime.

The districts displayed all have a different crime profile. For example, 75 percent of district D4's Part One crimes were larceny, but only 33 percent of district B3's Part One crimes were larceny. There is clearly a lot of room to categorize different districts by crime profile. This could easily be expanded into a K-Nearest Neighbors problem given the time. The final question is whether the time of day had any effect on crime. Figure 6 displays select crimes that have interesting variances in their crime profile.

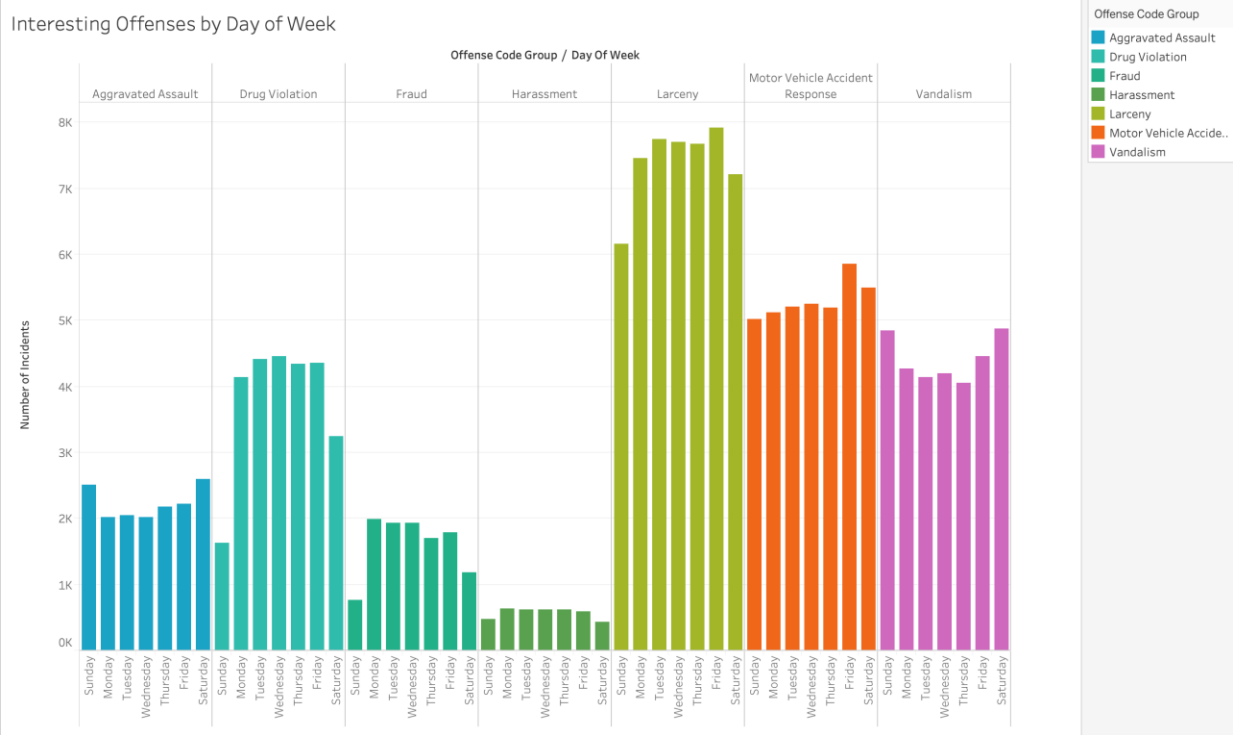

Figure 6: Select crimes with stark differences in counts across different days.

The results are self-explanatory and bring up a variety of interesting questions, such as why there is a startlingly low number of drug violations and larcenies on Sundays, or why vandalism spikes on weekends.

Overall, the dataset shows potential to be used in multiple different ways, and could possibly incorporate other factors given more time. While the analysis brings up interesting points, the purpose of the research is to predict, so the information gleaned from the exploratory analysis will be used to inform what should and should not be included in the model building process.

Phase 1 Model Building

As stated in the abstract and introduction, the goal is to predict any crime based on the characteristics such as location and time of day. After completing the exploratory data analysis, the author dove right in to creating machine learning models with Scikit-Learn.

Data Cleaning

The first important step was cleaning the data. While it was possible to make stunning graphs on Tableau, there were null values and some inconsistent values in columns that needed to be cleaned using the steps below:

- Reformatted the string in *Offense Code Group* to remove duplicates and provide consistent formatting
- Mapped *Day of Week* values from string values to integer values for use in decision trees.
- Mapped *Shooting* values from Null / Yes to 0 and 1, respectively for use in algorithms.
- Recalculated *Latitude* and *Longitude* values from *Location* variable and renamed columns to be consistent with the others.

Using the null value analysis done in the exploratory data analysis section, the following conclusions were made about those variables:

- *District:* As discussed in class, it was more important for me to have a *Latitude/Longitude* value than it was a *District*. Thus, I will ignore the variable in my analyses.
- *Shooting:* As null values are information, they were remapped to integers, which means nothing will need to be removed.
- *Street:* As the street is a derivative of a location, it will be ignored.

- *Latitude/Longitude:* The location variable was the most important, and the only case that eliminating entries was seriously considered. However, there was a *Location* variable that was not missing any entries. Therefore, *Latitude* and *Longitude* were recalculated using the *Location* variable, and then *Location* will be discarded.
- *UCR Part*: *UCR Part* describes the severity of the crime, but it shares a direct relationship with *Offense Code Group*, so it will be ignored when predicting the *Offense Code Group*. However, if predicting the *Offense Code Group* goes south, *UCR Part* could be predicted instead.

After cleaning the data, it was concluded that *Latitude*, *Longitude*, *Hour*, *Day of Week*, *Month*, and *Shooting* were the best variables to use to predict the type of crime. They were picked in part because they were not directly computable from any of the other variables, (such as *UCR Part* and *Offense Code Group)*. No entries were eliminated, as there were no null values across those columns after the data cleaning process.

0R Test

The model building process began by running a simple 0R/No-Skill test. It was a good first move, as it set a baseline to determine how well subsequent models perform. It could effectively guess that the incident was a Motor Vehicle Accident Response at 11.64% accuracy. This was not surprising, as accidents are a relatively common, low severity incident.

Decision Tree

For the decision tree, the test and training datasets were split 20/80 using stratified random sampling. As the Boston Police Incident dataset is complete, Boston's crime ratios are

expected to remain generally constant. A 20/80 split is generally common, and it gives a sizeable dataset for testing. The sampling strategy was implemented across all models to provide consistent baseline parameters.

Decision trees can be an effective way to classify different objects. As decision trees have been thoroughly implemented in Scikit-Learn, it was a good start for data mining (INRIA and others, 2019). When the model was built, the results of the decision tree were surprising. After the first run without a max depth limit, the accuracy was 73.962%, a significant improvement over the no-skill test. However, there were fears that the model was overfitted. The decision tree was retrained with a 40/60 test-train split and got comparable results. Again, overfitting was a serious issue. In order to find an optimal max depth, the model was programmatically tested at various maximum depths against accuracy, as shown in Figure 7.

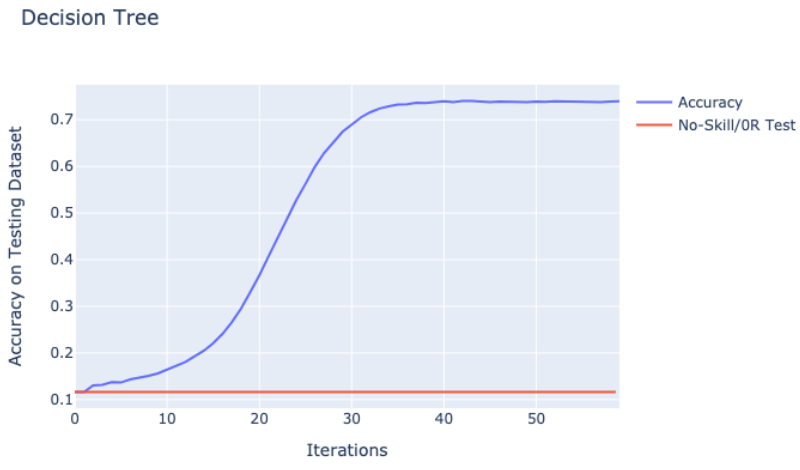

Figure 7: Accuracy on testing dataset versus max depth

As noticed, the best results happen at about 30 iterations, and it levels off. The performance of the decision tree was commendable, and it brings up a question on how well a

random forest, a collection of decision trees, will do. A random forest should outperform the decision tree significantly as well as reducing the number of branches.

Random Forest

Implementing a random forest should hopefully cut down on overfitting while preserving accuracy. On a run without any maximum depth, an accuracy of 73.9% was achieved, which is comparable to the decision tree. Again, overfitting became a concern, as the random forest should have been able to outperform the decision tree. In Figure 8, the accuracy versus the maximum depth is plotted again.

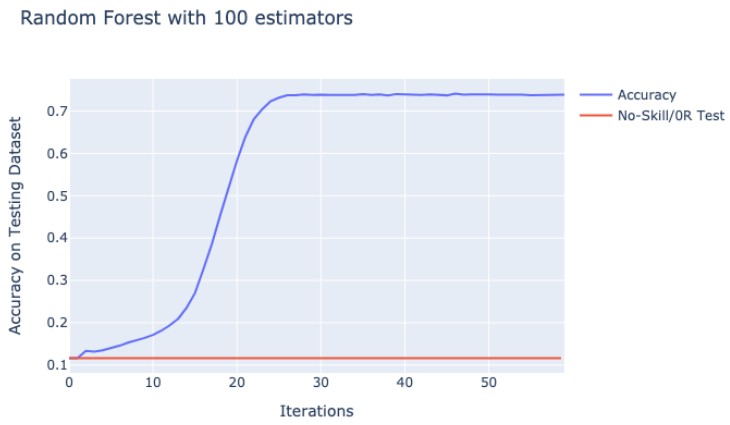

Figure 8: Accuracy versus iterations on a random forest.

The random forest, like the decision tree also achieved its best accuracy around a depth of 30. Decision trees and random forests were clearly not producing the expected results, so possibly moving to a neural network to solve the overfitting issues.

Multi-Layered Perceptron Neural Network

For the final phase 1 model, a multi-layered perceptron neural network (MLP-NN) was implemented, and it was expected to perform well. However, blind optimism got in the way again, and the out-of-the-box MLP-NN got an accuracy of 11.64%, which was as good as the no-skill test. The results were disappointing, and it called for a re-evaluation of the problem.

Phase 2 Model Building

There were a few fatal flaws made in the first phase of model building. The first of which was the fact that hyperparameters are what really determine the success and failure of a model. The hyperparameters were never tuned, as phase 1 models were haphazardly running based on whatever could be found online. Moreover, the phase 1 data cleaning was sloppy. There were 6 classes with less than 10 entries, such as a biological threat or manslaughter that should have been dropped. The days-of-the-week variable should have been one-hot encoded as it was more of a discrete, categorical variable than a continuous one like date. Moreover, the model building process was not consistent. The phase 2 model building process fixed the issues discussed, and used a new model evaluation technique:

1. Fit and optimize select hyperparameters
2. Fit model
3. Get results

As Scikit-Learn was well put together, multiple models used the same high-level code, minus the hyperparameters to tune and the actual model. After using the new process, the number of tested models went from 4 to 491 in a little under two days.

MLP-NN

The idea for the new evaluation technique stemmed from attempting to improve performance on the neural network. Dr. Michele Van Dyne was consulted, and she said it was a little odd that a neural network would perform so poorly on the data (Van Dyne, 2019). Some research was performed on improving the hyperparameters, and there was a way to run parameter searches autonomously. All permutations of the following parameters were run using a GridSearchCV class in Scikit-Learn. It returns the best model given a scoring metric, and all phase 2 models will use accuracy.

Parameter	Values (Bolded are best params)
Hidden Layer Sizes	**(50, 50, 50),** (50, 100, 50), (100,)
Activation	Tanh, **ReLU**
Solver	SGD, **Adam**
Alpha	**0.0001**, .05
Learning Rate	Constant, **Adaptive**

Table 3: Parameter searches on an MLP-NN.

The performance increased modestly from 11.64% to 12.31%. The accuracy rose, and thus it became standard across all phase 2 models. From there, all phase 1 models were reevaluated in hopes of squeezing out a few more percentage points of accuracy.

Decision Tree

Due to the findings with the MLP-NN, GridSearchCV was used to re-evaluate the accuracy of the decision tree to 80% or potentially more.

Parameter	Values (Bolded are best params)
Criterion	Entropy, **Gini**
Max depth	1, 2, …, **13**, …, 60
Splitter	**Best**, Random

Disappointment struck again with a 16.62% accuracy. GridSearchCV must have attempted to remediate overfitting, as the hyperparameters used to get a 74% accuracy were within the parameter search. This would explain why it discarded the high-accuracy and possibly overfitting models that could be found with a large max depth.

Random Forest

By this point it was obvious that a high accuracy score was unobtainable with GridSearchCV, but there was a peace of mind knowing that phase 2 models were not overfitting anymore.

Parameter	Values (Bolded are best params)
Criterion	Entropy, **Gini**
Max depth	1, 2, …, **15**, …, 30
N Estimators	10, 50, **100**

Like the first run-through of the random forest, more computations only gained a meager .3% gain in accuracy (for a total of 16.93%). GridSearchCV made an interesting call to allow max depth to be deeper than the normal decision tree, and it could be due to less risk of overfitting by averaging all the trees in the forest.

XGBoost

For the last model in phase 2, the author wanted to do something complex and state of the art. XGBoost is an ensemble learning algorithm that mixes a variety of features with gradient boosting to get better accuracy than many other generic models.

Parameter	Values (Bolded are best params)
Learning rate	.01, .02, **.1**, .2, .3
Max depth	4, 6, 8, **10**

The model did not get a phenomenal accuracy, but it did get 18.90%: a solid 2 percent better than any of the leading models. The performance was impressive, but the computational time was significantly more than any of the other models trained. The fitting process took a few hours, and the model took 30 minutes to train after finding the optimal hyperparameters.

Another XGBoost variant was built but did not have its hyperparameters tuned. XGBoost was used as a base for a one-versus-rest classifier, a model was trained on each class, and got similar accuracy with the default parameters.

Conclusion

In conclusion, 18.90% accuracy was the best across all models. However, it was significantly better than the 11.64% accuracy given that there were 60 unique classifications. For future projects, it will be essential to have more variables to get insights. A possible issue with the analysis is that there were not enough variables to easily discern 60 unique classes. This could be remediated with supplemental data, such as income-per-capita per census tract. It would also have been beneficial to use precision and recall over accuracy so there was a clearer picture of what my model was doing well on. One of the most beneficial concepts for the author's learning was setting up a framework for model-building. It was much more efficient when using a consistent set of functions, metrics, and visualizations across all models.

References

INRIA and others. (2019, November). *Scikit Learn*. Retrieved from Scikit Learn: https://scikit-learn.org/stable/

Van Dyne, M. (2019, November). Neural Network Advice. (H. Boles, Interviewer)