Predictive Apprehension

Michael Feffer, Rui Li, Aaron Vontell

1 ABSTRACT

Crime analysis and prevention has always been an interesting topic. With the rise of machine learning models and methods in this data-rich age, one can now attempt to prevent and predict crime through an analysis of previous crimes and patterns. In this paper, we present our results in attempting to use neural networks, K-Nearest Neighbors, and generative models to predict and analyze crime, specifically in the City of Chicago. We conclude with a discussion of the next steps that one could take to further this work.

2 MOTIVATION AND BACKGROUND

We are working with a crime dataset from Kaggle, which includes crime data from the City of Chicago during the years ranging from 2012 to 2017 [1]. However, we contend that the same methods and techniques shared here are also applicable to Chicago crime datasets from previous years as well as crime datasets pertaining to other cities. Such datasets for the cities of Baltimore, Philadelphia, and New York City for years ranging from 2001 to 2017 are also easily accessible through the Kaggle database.

The motivation for this project stems from the fact that police staffing, spending, and planning can greatly benefit from predicting the type of crimes that are likely to occur in a given area at a certain time. For instance, if theft is found to be more likely in a certain situation, then more resources can be spent on patrolling. Given the amount of police data that has been collected throughout the years, it is no surprise that machine learning methods may be useful in aiding this planning. The goal of this project is therefore to determine which methods are most useful for predicting the type of crime most likely to occur given parameters such as time, location, and social factors.

To analyze the aforementioned Chicago data, we first developed a dedicated data sanitizer and feature extractor to obtain trainable information from raw data. After obtaining these features, we employed a variety of supervised learning techniques, ranging from K-Nearest Neighbors to neural networks implemented with Keras, in order to predict the crime type based on information given from our constructed feature vectors. We also attempted to construct space and time-based Gaussian mixture models and multilevel mixture models to perform unsupervised learning and data analysis. These additional unsupervised learning techniques allowed us to obtain different views and visualizations of the given data. All of our code can be found on GitHub.[1]

3 DATA COLLECTION

The data used consisted of crime reports and social factors for the City of Chicago. More specifically, a crime record database was downloaded from Kaggle, which included all reported crimes from 2012 to 2017 in CSV format. [1] A total of 1,456,714 crimes existed within the database, and after removing 37,083 crimes due to invalid formatting (issues with dates, missing data, etc), experiments were able to utilize up to 1,419,631 crimes. These crimes had attributes as shown in Table 1, which were used in constructing our feature vectors.

Table 1: Crime Data Properties

ID	CASE NUMBER	DATE
BLOCK	IUCR	PRIMARY TYPE
DESCRIPTION	LOCATION DESC.	ARREST
DOMESTIC	BEAT	DISTRICT
WARD	COMMUNITY AREA	FBI CODE
X COORDINATE	Y COORDINATE	YEAR
UPDATED ON	LATITUDE	LONGITUDE

Table 2: All Crime Categories

SEX OFFENSE	CRIMINAL TRESPASS	MOTOR VEHICLE THEFT
OTHER OFFENSE	INTIMIDATION	PUBLIC INDECENCY
PROSTITUTION	NON-CRIMINAL	PUBLIC PEACE VIOLATION
GAMBLING	CONC. CARRY VIOLATION	WEAPONS VIOLATION
NARCOTICS	INT. WITH PUBLIC OFFICER	THEFT
DECEPTIVE PRACTICE	STALKING	OFFENSE INV. CHILDREN
OTHER NARCOTICS	KIDNAPPING	CRIM SEXUAL ASSAULT
ARSON	BURGLARY	BATTERY
CRIMINAL DAMAGE	NON-CRIMINAL	ASSAULT
ROBBERY	HUMAN TRAFFICKING	HOMICIDE
OBSCENITY	LIQUOR LAW VIOLATION	

Further inspecting the data, Table 2 shows the list of all crimes types provided within the data set. Due to the sheer number of crime types, we further reduced this feature into a list of 9 crime categories, which are shown in Table 3.

Table 3: Condensed Crime Types

KIDNAPPING / CRIME	ROBBERY/BURGLARY/THEFT	ASSAULT/VIOLENCE
NARCOTICS	PUBLIC-RELATED CRIME	DAMAGE/ARSON
OTHER/NON-CRIMINAL	WEAPON-RELATED	PROHIBITIVE CRIME

Upon initial testing, it was clear that the City of Chicago crime dataset was not enough to complete the tasks as described earlier. Rather, it was also necessary to incorporate measures pertaining to social aspects such as poverty levels and population

[1]https://github.com/vontell/CrimePrediction

without a high school diploma into our data repository. This additional data, provided by the City of Chicago Data Portal, was used to further enhance the features that we included within our machine learning pipeline. [2] Table 4 shows the data properties that were included within this additional dataset for each of Chicago's 77 community areas while Figure 1 shows a map of these 77 community areas with their corresponding poverty rates. This additional dataset did not originally include location for each neighborhood, which was added manually to our data repository.

Table 4: Chicago Social Factors

COMMUNITY AREA	NAME	ASSAULT RATE
FIREARM RATE	BELOW POVERTY LEVEL	CROWDED HOUSING
HOUSING	DEPENDENCY	NO HIGH SCHOOL DIPLOMA
PER CAPITA INCOME	UNEMPLOYMENT	LATITUDE
LONGITUDE		

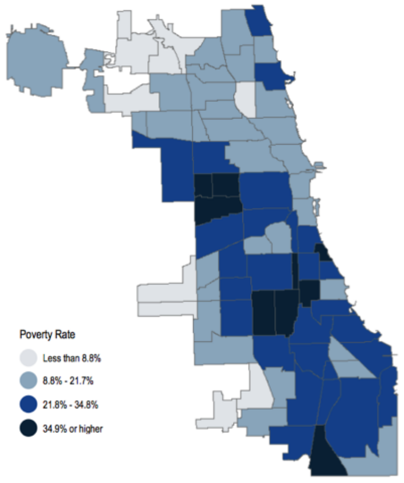

Figure 1: Poverty rates for each of the 77 community areas in 2013.

After using our sanitizer to load the crime and social data from the raw files, one can use our feature extractor to automatically load custom feature vectors to use with various machine learning algorithms. The list below displays the various features that can be chosen for a single data point. Note that not all data from these datasets were used, as the features below were thought to be most relevant in an analysis of crime in the Chicago area.

(1) **Day** - A one-hot encoded vector of length seven representing the day (Sunday through Saturday) that this crime occurred.
(2) **Time** - A one-hot encoded vector representing the time (Morning, Afternoon, Evening, and Night) that this crime occurred.
(3) **Time min** - An integer value representing the minute that this crime occurred, or the minutes elapsed since 12:00am.
(4) **Hour** - An integer value representing the hour that this crime occurred in the range [0, 23].
(5) **Location** - Two float values representing the latitude and longitude where this crime occurred.
(6) **Location normalized** - Two float values, each between 0 and 1, which represent the latitude and longitude in a scaled or normalized fashion.
(7) **Crime full** - A one-hot encoded vector of length 32 which represents the crime type of this crime, given by a value from Table 2.
(8) **Crime condensed** - A one-hot encoded vector of length 9 which represents the basic crime type of this crime, given by a value from Table 3.
(9) **Below poverty count** - A float value representing the rate of people who fall below the poverty level in terms of income for the community area corresponding to the given crime.
(10) **Crowded** - A float value representing the rate of crowding for the community area corresponding to the given crime.
(11) **No diploma** - A float value representing the rate of not having a diploma for the community area corresponding to the given crime.
(12) **Income** - A float value representing the average income per household for the community area corresponding to the given crime, normalized by subtracting by the average and dividing by the standard deviation.
(13) **Unemployment** - A float value representing the unemployment rate for the community area corresponding to the given crime.

4 DESIGN DECISIONS

We decided to try a variety of methods, both supervised and unsupervised, to solve the task of crime classification and analysis. For supervised learning techniques, we wished to produce a model that predicted the most likely crimes given a set of input features. Hence, we decided to try K-Nearest Neighbors to find the most likely crimes in terms of feature similarity, multiclass logisitic regression to find boundaries between different types of crimes, and a neural network to learn a mapping from features to crimes. For unsupervised learning techniques, we wished to find connections between features and crimes, so we employed both Gaussian mixture models and multilevel mixture models to cluster crimes and perform in-depth analysis given certain features.

Table 5: Baseline Methods Performance

	Test Set Top 1	Test Set Top 3
Majority Vote	36%	72%
Uniform Sampling	10%	N/A

5 BASELINES

Before any experimentation with various techniques, we obtained some baselines to compare our results against. Our first baseline was performed with a significant amount of training data points (60,000) and found the majority crime type (see Section 6.1 for Top k details). For all validation and test data, we simply predicted the majority crime category. We found that this baseline performed decently, with the results shown in Table 5. We also tried a baseline where we randomly assigned a crime type from the 9 possible condensed crime types, which gave us the expected results. These performance results for majority vote and uniform sampling prediction are reported in Figure 5.

6 SUPERVISED LEARNING: CLASSIFICATION TECHNIQUES

6.1 Neural Network

We decided to use a neural network as our first classification technique. Due to their prevalence as both prediction and classification models that work well with complex data, we believed that neural networks would be a "magic silver bullet" for our problem.

For our general network architecture, we chose a dense feedforward neural network to predict the type of crime given a set of features. We did not use convolutional neural network since we did not extract data in a way that would allow the use of locality with regard to input or intermediate features (although this was briefly considered). Our prediction model also assumes that crimes are independent of one another, and therefore we did not use a recurrent neural network since that would potentially violate this assumption.

To evaluate different network architectures, we created a pipeline using Keras [3] that trained a network with Adam optimization to predict crimes based on a feature vector. Our first feature vector included Day, Time, and Location, and our first neural network used 25 fully-connected (dense) hidden layers, each containing 100 units with ReLU activation that had dropout probability of 0.2, as well as a final output layer with softmax activation to predict one of the nine Crime condensed categories. We chose such a large architecture as a baseline because we figured that it would have a good chance of fitting complex data rather well (even if it ended up overfitting). After better processing of data and noticing that networks with smaller numbers of hidden layers classified data with the same accuracy, another network was produced that only used 5 hidden layers with the same number and type of hidden units per layer as before. However, the feature vector employed by this network involved Day, Location, and Time min (instead of Time) as input. Lastly, we ran one more experiment with the same architecture but a different feature vector. This feature vector included Income normalized, Below poverty count, No diploma, Unemployment, Time min, and Location normalized. Feature vector and network architecture combinations are summarized in Table 6. (Features corresponding to the numbers found in the feature vectors are given by the list in Section 3.)

Table 6: Neural Net Architectures and Feature Vectors

ID	Feature Vector	Hidden Layers	Units per Layer
1	[1, 2, 5]	25	100
2	[1, 3, 5]	5	100
3	[1, 3, 6, 12, 9, 13, 11]	5	100

Each neural net was trained and evaluated with a different set of data, but the data in each case was extracted from our overall data repository. Examples were shuffled and split according to a 60-20-20 partition into training, validation, and test sets. The first neural network saw the most data, but less data was used for the other networks (for easier training). The training set was used to train the network's weights, the validation set was used to determine when to stop training (when validation accuracy peaked, which happened around 5 epochs), and the test set was used to predict the network's performance on "real-world" data. Partition information and accuracy results are included in Table 7.

Table 7: Neural Net Best Accuracies

ID	Train Size	Val Size	Test Size	Train Acc.	Val Acc.
1	851778	283926	283927	36.61%	36.64%
2	58483	19494	19495	34.02%	36.62%
3	58469	19489	19491	36.63%	36.49%

ID	Test k = 1	Test k = 2	Test k = 3
1	36.69%	N/A	N/A
2	36.62%	62.20%	73.79%
3	36.85%	62.54%	73.41%

When working with the second network, upon observing one of the softmax vectors output by this model, we realized that our model actually assigned relatively high likelihood (around 20%) to multiple classes of crime for certain examples. Therefore, we reexamined the accuracy of our model on the test set by counting a prediction as “correct” if the correct crime label is within one of the top k most likely categories as predicted by the model. We tested both of the latter two networks with $k = 2$ and $k = 3$ on the test set and included their corresponding accuracy measures in Table 7. Since we believed that our first network overfit the data, we did not compute the additional accuracy measures for that first network.

We originally thought that the cause of the low accuracy was due to mapping time to only four discrete values, which we figured was causing different crime types to overlap in the feature space. However, our accuracy was almost the same with the next neural network, which made time a continuous feature and therefore should have helped reduce the overlapping issue (if it existed). We then thought the issue was due to having too few features to be able to learn the type of crime, so with our next network, we heavily augmented it with many location-based features. This also did not yield the results we were expecting.

Nevertheless, these results point to a few useful insights about the underlying data. First, the network's difficulty in classifying the the data properly suggests that the data is inherently difficult to classify. Feedforward networks like the one we employed with our dataset can perform well on the MNIST dataset (albeit not as well as convolutional neural nets).

Some key differences between our dataset and the MNIST one include the relative number of input features (MNIST digits have 784 when flattened while we only used at most 7) and how well the problem and data are structured. Different types of digits, even with variation between handwriting, can generally be discerned from one another based on the features included, and even if feedforward neural nets do not utilize the structure of each digit or the locality of features to the utmost degree, both still make classification a relatively easier problem. There is less structure in the crime landscape of Chicago.

Moreover, unsupervised learning techniques explored in a later section of this paper show that clustering crimes based on location or time alone does surprisingly little to separate out different types of crime. This partly explains why our networks seldom perform better than baseline performance where majority vote was used as the classifier.

At the same time, the additional features seemed to help our final network relative to our penultimate one, at least where our values for test accuracy with $k = 1$ and $k = 2$ are concerned. Those subtle increases in accuracy at the cost of a slight decrease in accuracy pertaining to $k = 3$ suggests that the additional features helped the classifier become more accurate when considering smaller numbers of likely crimes but also took likelihood away from other classes. Therefore, potentially with more features or features structured in a different way, future experiments may be able to bolster accuracy. At any rate, it suffices to say that this classification task was very challenging for our neural networks with the data and features extracted.

6.2 K-Nearest Neighbors

We also implemented K-Nearest Neighbors to predict the type of crime given the feature encodings. The motivation for using K-Nearest Neighbors was that we felt that with the limited number of features we had, a simple cosine-similarity to measure "closeness" between the feature vectors would give a good prediction of how similar the crimes would be, assuming the features had some correlation with the type of crime. Given a new data point, we would calculate the K nearest neighbors based on cosine-similarity as a distance metric. The data point is then assigned the type of crime that is the majority crime label of its K nearest neighbors. This was implemented with SciKit Learn's K-Nearest Neighbors Classifier. [4]

We tried a range of the number of datapoints, and found that generally as the number of datapoints increased, the accuracy of KNN increased significantly. The model was fit on the training data points with their corresponding crime types represented as the integers 0-9 for the condensed version of crime types. The parameter (value of K) was then tuned on the validation set for the values $K = 1...30$, for each setting of the number of training, validation, and test sets. Results are summarized in Figure 2, where the y-axis is the score on the test set using the optimal setting of K, and the x-axis is the number of points in the training set. The number of validation points and test points were both fixed as control variables at 500 points. As can be seen from Figure 2, the number of training points does not really make much of a difference once it is above approximately 5000. This trend occurs because once a sufficient number of training points are used, the general distribution is already captured, so adding more points is unnecessary.

We then tried to see whether the optimal value of K had a trend with respect to the number of points. Figure 3 shows the value of the optimal K value for each setting of the number of training points.

From this figure it could be seen that the optimal value of K did not really depend on the number of training points. This is surprising because we

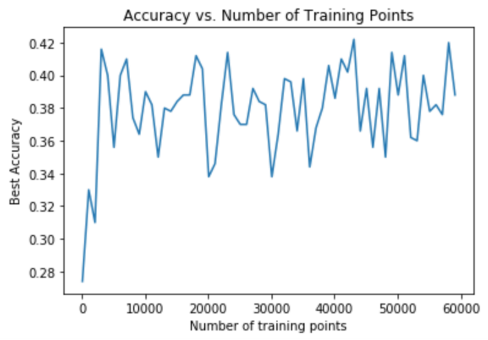

Figure 2: KNN Performance on Test Set vs. Number of Training Points

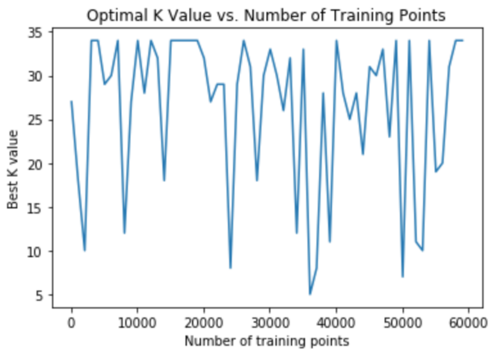

Figure 3: KNN Optimal K on Validation Set vs. Number of Training Points

expected that as the number of training points increased, we would see more relevant points closer to the given training example, meaning that smaller K values would be optimal as the number of training points grew. From this result, we had some suspicions that the features we chose are not truly expressive of the type of crime. Similar features did not necessarily correlate with the same type of crimes.

As a result of these suspicions, we proceeded to explore various feature spaces. The above plots were generated with the features "income", "below poverty count", "time min", and "unemployment" and "location normalized". We then tried using only 2 features, "time min" and "location normalized" to see if just these 2 features would be better or worse. The results are displayed in Figures 4 and 5.

Comparing Figures 4 and 5 to Figures 2 and 3, we could see that accuracy started off much higher with just 100 training points at 38%, while it only started off at 28% accuracy for the version with more features. In terms of overall performance, we did not see much of a decline. In fact, with only 2 features, KNN achieved a fairly high performance of nearly

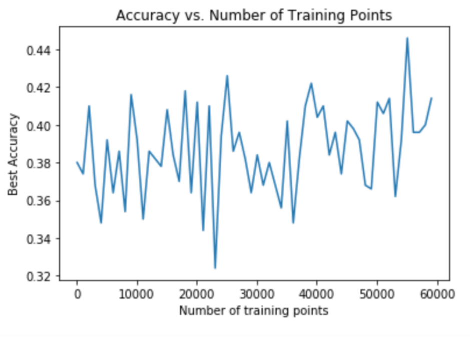

Figure 4: KNN Performance on Test Set vs. Number of Training Points

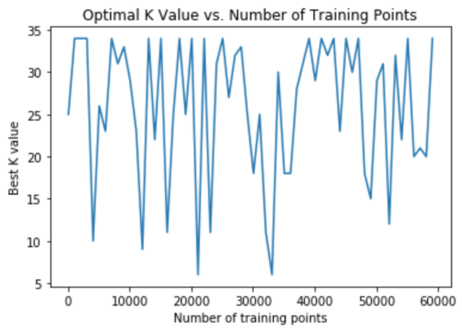

Figure 5: KNN Performance on Test Set vs. Number of Training Points

46% accuracy with approximately 50,000 training points. This lends to the suspicion that our features are not very strongly correlated with type of crime.

6.3 MultiClass Logistic Regression

Seeing the overall lackluster performance of KNN, we chose to experiment with multiclass logistic regression (Maximum Entropy Classifier). We figured that if there was roughly a linear separation between the class boundaries, this method would work well. For this method, we used SciKit Learn's Logistic Regression class.

Similar to KNN, we tried varying the number of training points used in an attempt to see if this played a factor in accuracy. For each setting of the number of training points, we tuned for the optimal regularization constant C by trying values in the range 0 to 50. We found as a general trend that C did not play a major role in determining the accuracy performance, except when the number of training points was low. With just 100 training points there is an optimal regularization constant value, $C = 2.0$. With many more training points, on the other hand, the value of C did not matter. The fact that the regularization constant does not matter for large number of data points is a reasonable result, since regularization should not be as needed when the number of data points is sufficiently large, since outliers would have less of an impact on the overall accuracy. The overall accuracy for the multiclass logistic regression hovered around 36% to 37%. The performance of logistic regression vs. the number of training points used is reflected in Figure 6.

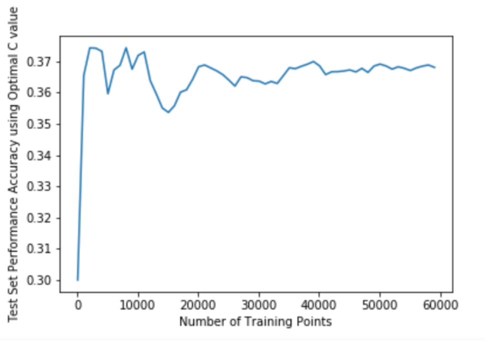

Figure 6: Multiclass Logistic Regression Performance on Test Set vs. Value of C(100 Training Points)

This illustration was generated with only the features "date time" and "location normalized". It seems that multiclass logistic regression does worse than KNN. This is reasonable because multiclass logistic regression assumes some linear separability along each dimension, so the fact that this does not perform as well means that the data points are scattered such that a simple regression model cannot perform as well.

7 UNSUPERVISED LEARNING: DATA ANALYTICS AND VISUALIZATION

7.1 Gaussian Mixture Models

After noticing that our supervised learning techniques were not working as well as we had predicted they would, we turned to unsupervised learning approaches to try and understand the dataset from other perspectives.

We first used Gaussian Mixture Models (GMMs) in two ways. In both ways, we clustered crimes based on only *one* feature to see how well different types of crime could be separated from each other. Therefore, each application of GMMs involved clustering with one feature and labeling each point with the corresponding crime after clustering.

The first application of GMMs used location as the only feature. Our crime generation model in this case assumed k "location-based crime clusters", where each cluster is assumed to be a multivariate Gaussian with means and variances for both latitude and longitude. A cluster is chosen based on a corresponding probability, and if chosen, a crime is sampled based on the cluster's parameters. Note that the cluster does *not* assign a crime label; it solely generates a crime of any type.

With this approach in mind, running EM with a dataset of 500 points and with $k = 2$ produced Figure 7. The ellipses correspond to the confidence ellipsoids of each cluster. The cluster weights, means, and variances are found in Table 8.

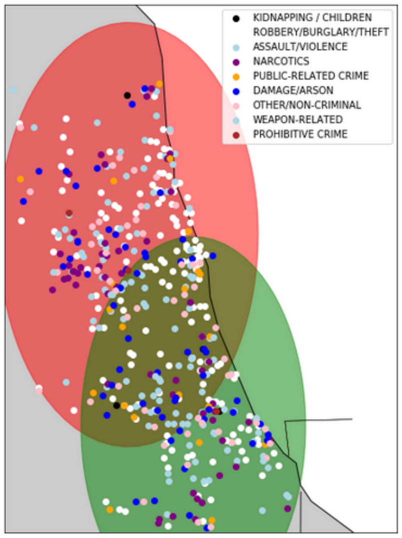

Figure 7: Mixture of Two Gaussians Corresponding to Location-Based Crime Generators.

Table 8: GMM Parameters for Location-Based Crime Generators

Cluster	Color	Weight	μ_{lon}	σ^2_{lon}	μ_{lat}	σ^2_{lat}
1	Red	0.56	-87.70	0.0029	41.91	0.0025
2	Green	0.44	-87.63	0.0022	41.76	0.0020

Note that both clusters are densely concentrated with a variety of different types of crime, so the notion that crime is limited to "South-Side Chicago" may actually be false based on this dataset. Additionally, based on the parameter weights, crime is more likely to be generated from the North cluster than from the South one. More analysis could be done to show whether more violent crime happens in the South cluster, but based on raw count of crimes alone, the North cluster actually claims more responsibility for crimes in Chicago.

The variety of different crimes in both clusters points to the difficulties of the aforementioned supervised learning techniques in classifying the data. If there was not as much of a variety in both clusters and one cluster saw more of one type of crime than the other, then any useful location-based feature would have helped those techniques. Since this was not the case, the variety could be one reason for poor performance.

Another difference between the clusters is in the variances between the two datasets. Based on both the figure and corresponding table, the South cluster is thinner than the North one due to having a smaller variance in both latitude and longitude. Therefore, the South cluster may be more densely concentrated, but the North cluster is still densely concentrated as well.

After clustering by space, we then used a GMM to cluster by time. Specifically, we used the Time min feature and divided by 60 to get the elapsed hours from midnight (as floats) and used EM to obtain a GMM with crime clusters corresponding to a crime generation model that generated crimes based on time (and again did not perform any labeling of the type of crime). This was a 1D clustering problem, so each cluster only had one mean and variance (corresponding to time in each case) as parameters, and a plot of the corresponding density functions with $k = 2$ is shown in Figure 8. Only 200 points were used in this case. The cluster weights, means, and variances are found in Table 9.

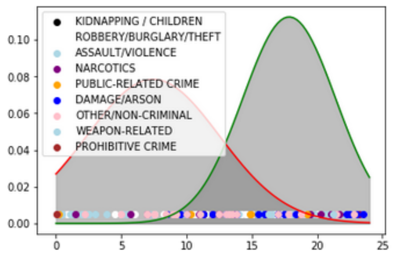

Figure 8: Mixture of Two Gaussians Corresponding to Time-Based Crime Generators.

Table 9: GMM Parameters for Time-Based Crime Generators

Cluster	Color	Weight	μ	σ^2
1	Red	0.36	7.44	25.88
2	Green	0.64	17.86	12.59

As one might expect, the crime cluster whose mean is centered over the evening hours has smaller variance, evidencing the fact that a lot of crime is concentrated in the evening hours. Additionally, there is a larger parameter weight associated with the evening cluster, further supporting this fact. The cluster whose mean is centered over the morning hours has a large variance to account for crimes that occur both in the early morning and early evening, but it also has a smaller parameter weight to account for the infrequency of crimes during that time period. However, note again the large variety of crimes that occur under each curve. This again points to the fact that time is not as useful of a feature as we had hoped, given that crimes can still happen regardless of the type at virtually any hour.

One might argue that these two methods are too simple to yield any potential analytical benefit. However, we argue that it is actually through these methods' simplicity that their true powers are realized. For instance, from a generalizability perspective, these methods are most easily applicable to other crime datasets out of the other methods we employed during analysis of this dataset. This is because while other crime datasets may have different labels for types of crime or names of neighborhoods, all crimes are associated with a time and place, both of which can be easily extracted from the dataset in question so that the two methods discussed here can be utilized. Additionally, from a comprehensibility perspective, these models are also the easiest to understand. Almost anyone can look at either of the two cluster graphs and be able to tell what the graphs mean, but neural net performance interpretation is more difficult to comprehend.

7.2 Multilevel Mixture Models

Given the complexity of the dataset, we wished to try to employ a more complex unsupervised learning technique. Inspired by LDA topic models, we devised a Multilevel Mixture Model (MMM) for this problem that combined the ideas of GMMs and LDA topic modeling to model crime clusters. The MMM employed for this problem assumed the following processing to "generate" a crime:

(1) Crime cluster i is chosen to generate a crime with probability π_i
(2) A crime type j is chosen given crime cluster i with probability θ_{ij}
(3) Time, latitude, and longitude are sampled from normal distributions with means μ_{ij} and variances σ_{ij}^2 given crime type and cluster

PyStan, a Python wrapper around the Stan probabilistic programming language, was used to train the model described from 100 data points and MCMC sampling. [5][6] Specifically, 4 Markov chains were used to obtain 1000 samples (half of which were used to warm-up the chains), to fit the model assuming 2 crime clusters and condensed crime categories. Results of the model we generated are stored in the `stan_output.txt` file in the root directory of our GitHub repository in terms of the parameters described as above.

Compared to the GMMs, which were easy to train and explain, this MMM took large amounts of time to train on only a small amount of data, and due to this fact, the hyperparameters were difficult to modify. However, with this additional complexity came additional power. For most crimes, one cluster was more likely the cause of a crime than the other, and through the Gaussians, each cluster had different "average locations" and times for the same type of crime.

8 COMMENTS AND NEXT STEPS

In the end, performing supervised learning with this dataset proved far more difficult than originally conjectured. We at first believed that our problems were due to our feature vector engineering and data representation, but even our models that utilized as many features as possible were no better than guessing the majority class.

As alluded to earlier in our paper, one hypothesized major problem with this dataset was the lack of features relative to the complexity of the problem. In future experiments, one might be able to improve prediction performance by reimagining the data representation. For instance, this could include classification or prediction based on "crime heatmaps" corresponding to crime activity at a certain time throughout all of Chicago. Also, we made an assumption with our neural network model that crimes are independent of each other, but if this assumption was actually too naive (e.g. a shooting in one place is highly correlated with a robbery in another), then a recurrent neural network or a convolutional neural network that performs temporal convolutions may be architectures better suited to prediction and classification tasks.

Additionally, if other neural networks have been effectively trained on crime datasets pertaining to other cities and the weights and architectures are open-source, then transfer learning and domain adaptation can be considered as viable techniques to fine-tune a well-trained classifier to our scenario.

Outside of neural network implementations, we could also tune parameters in both of our unsupervised learning approaches to perform analysis at a more granular level. Both GMMs and MMMs learning algorithms were only run with 2 crime clusters. However, when experimenting with the number of topics in the LDA topic model for the class assignment, it was observed that having greater numbers of topics allowed the topics that resulted to contain words that were more closely related compared to the topics that resulted from having smaller numbers of topics. Therefore, having more crime clusters for both GMMs and MMMs may improve analysis ability in that the crimes, times, and locations corresponding to each cluster may be more closely related to each other. The increased specificity could help obtain information about more general trends.

To further boost our performance we can also try ensemble methods, where the weaker KNN and multiclass logistic regression methods are combined to make a more powerful classifier. This could be done with boosting.

9 CONCLUSION

In this paper we used neural networks, K-Nearest Neighbors, and unsupervised learning methods to predict and analyze crime within the City of Chicago. Using the generated feature vectors, we found that using the supervised learning methods presented gave little to no improvement over the baseline methods for predicting crime, while our unsupervised methods gave a few insights into why this may be. The data used and features constructed were simply not well-suited to this sort of classification. However, further steps and experiments should be undertaken using the aforementioned methods, which may give rise to further crime prediction abilities.

10 FOOTNOTE: CONTRIBUTIONS

The contribution of each member on this project was quite uniform. Michael Feffer worked on the Keras pipeline, construction of neural nets, and the GMMs and MMMs. He wrote up the main summary portions detailing those experiments as well as comments and future work related to those models.

Rui Li worked on the K-nearest Neighbors implementation and experimentation, the multiclass logistic regression implementation and experimentation, and the implementation of the majority vote and uniform sampling baselines. She also wrote up those corresponding sections in the paper, in addition to the design decision section and some parts of the comments and future works.

Aaron Vontell worked on compiling the datasets, parsing and sanitizing the data, and creating a system that would automatically create feature vectors for use within the machine learning algorithms. He also worked on incorporating social factors and modified data into the algorithms to further increase performance. He contributed to writing the abstract, motivation, data collection, conclusion, and some next steps within the paper.

REFERENCES

[1] Currie32, *Crimes in Chicago*. Kaggle Dataset, https://www.kaggle.com/currie32/crimes-in-chicago.

[2] *Chicago poverty and crime*. Online Dataset, https://data.cityofchicago.org/Health-Human-Services/Chicago-poverty-and-crime/fwns-pcmk/data

[3] Francois Chollet, and others, *Keras*, 2015. https://github.com/fchollet/keras

[4] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.

[5] Stan developers. *Stan*. http://mc-stan.org/

[6] PyStan developers. *PyStan*. 2013. https://pystan.readthedocs.io/en/latest/