Chicago Crime Prediction and Analysis using Machine Learning

Kate Dums
kdums@wisc.edu

Adam Shedivy
ajshedivy@wisc.edu

Shane Spellman
sspellman3@wisc.edu

Abstract

The objective of this project is to predict whether a crime incident results in an arrest or not via several machine learning algorithms including KNN, decision tree, logistic regression, random forests, etc. We further analyze the most important features in our data set that contribute to a correct prediction. We combine two data sets, the first being "Crimes - 2001 to Present" published by the Chicago Data Portal. The second data set is census data from 2008-2012 that includes selected socioeconomic indicators in Chicago communities, also published by the Chicago Data Portal. We split this final data set into a training, validation, and test set in order to improve model parameters and test for accuracy.

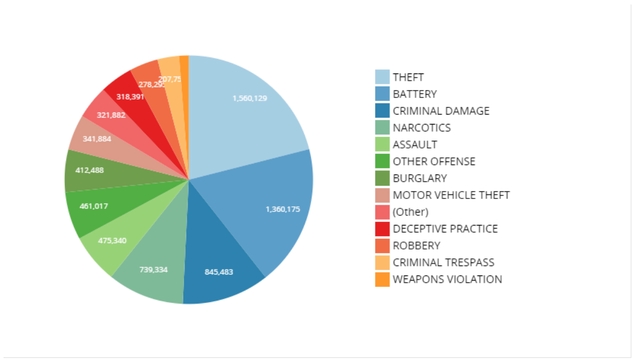

Figure 1. Pie Chart illustrating the proportion of crime types in Chicago from 2001-2021

1. Introduction

In recent decades, there has been an increasing expansion of technological advances in data collection and analysis that can lead to improved effectiveness of policing. A common reason to use predictive algorithms is to anticipate and gain the ability to preemptively act upon events likely to occur. For example, preventative measures such as surveillance can be targeted towards communities with a high likelihood of arrest. Or rather, in retrospect, an analysis can be completed regarding arrest rates in various communities in order to diagnose the criminal behavior that endangers each community.

Crime and violence, in general, have a direct effect on the safety, economy, quality of life, and well-being of community residents. Accurate prediction of whether an incident involving law enforcement results in an arrest can convey valuable insight into what factors ultimately lead to an arrest. Examples include which Chicago communities are arrested the most, which criminal activities lead to the most arrests, and much more. However, we are unable to draw causal inferences about why an arrest was made or not through our data. This could include a multitude of factors including whether the crime was solved, an officer's own discretion in making an arrest, or police resources dedicated to the area the arrest was made and the possible factors driving that decision.

Our original data set (Crimes - 2001 to Present) includes features including location, community number, type of crime, etc. However, to include a more holistic approach, we chose to add another data set including socioeconomic indicators about each community. Chicago has a total of 77 different communities with ranging socioeconomic statuses in each. This included features such as the percentage of the population without a high school diploma and the hardship index of each community. Including community-specific data allows for a more in-depth approach to determine the most important and relevant features that lead to an arrest. However, we are unable to determine whether these socioeconomic indicators imply anything about police practices or criminal activity stemming from each community as a result of their socioeconomic indicators.

2. Related Work

Recent literature regarding crime prediction is divided into different domains. There are several studies that highlight socio-economic factors like education, income level, and unemployment rates [5]. There has also been studies conducted using social networks such as Twitter, Facebook, and mobile phone data [3][1].

[1]For this project, we want to focus on predictive analyses primarily utilizing Daily crime report data. During our exploratory data analysis

Here are a couple former studies done using Chicago Crime data:

- Kang et al. used environmental context information to improve the prediction of models by proposing a feature-level data fusion method on deep neural networks. This study used four demographic datasets (City of Chicago Data Portal[2], American FactFinder, Weather Underground, and Google Street View) for the year 2014 and showed improved results in precision and recall [4].

- Christian et al. related socioeconomic and sustainable development indicators like poverty rate and unemployment toward crime by implementing Linear Regression Analysis from the year 2008 to 2012 for Chicago [2].

- There have been multiple studies that were performed by using geographical locations, meta-association rules and specific detection systems to examine the crime rate in Chicago [6].

We hope to use the recent work with crime analysis in Chicago as inspiration for developing our predictive models by focusing on using machine learning specific techniques within the scope of this course. Four major areas we plan to explore are: logistic regression, k-nearest neighbors (KNN), decision trees, and Bayes classifiers.

3. Proposed Method

3.1. Data Processing

SMOTE A common problem with many classification models is that the applied dataset suffers from class imbalance. This causes the model to over-predict towards the majority class, and possibly give the false impression of a well-performing model. Synthetic Minority Oversampling Technique, or SMOTE, is a remedy to this problem. The method over-samples from the minority class by synthesizing from existing examples, which avoids adding new information to the model. Specifically, SMOTE picks a random observation from the minority class, and uses K-nearest neighbors to draw a line between the original observation and a random selection of one of the K-closest neighbors. A new observation is produced somewhere randomly on this line. This process is repeated until a certain threshold of class balance is met. SMOTE will be used in our dataset to create a more even balance between the classes of arrest and non-arrest.

phase, it is our goal to attempt to incorporate other data sources such as income levels, and education data.

[2]this is the same data source we are using

3.2. Feature Selection

ExtraTreesClassifier An important aspect of all machine learning models to reduce as much ”noise” as possible in the dataset. Having so many features available to us, we felt it was important to only use ones that had a significant impact on our models. Our method of selecting features was ExtraTreesClassifier, an ensemble method that fits a certain amount of random decision trees to determine which features have the greatest cumulative impact across all the trees. The ”randomness” of each decision tree is ensured by inputting a random sample of the dataset into each tree, which helps us avoid over-fitting. This is especially important for our K-nearest neighbors models, which suffer from the curse of dimensionality with a large number of features.

3.3. Predictive Models

K-Nearest Neighbors The first classification model we used was K-Nearest Neighbors(KNN). KNN predicts the class of an unknown observation by considering observations that are closest in distance in the dataset. In our model, we use the euclidean distance equation to determine the distance between y, our unknown observation, and x, a possible reference observation. The euclidean distance equation is defined as:

$$d(\mathbf{y}, \mathbf{x}) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Where n is the number of dimensions in our dataset and i is the current dimension we are computing the distance in. The user of the model specifies the amount of observations to consider(k) when predicting the class of an observation. For classification KNN, the majority class in the k-nearest neighbors is what we predict our unknown observation to be.

Stacking Stacking is an ensemble method that consists of two layers of estimators. First, each baseline model outputs one prediction as an intermediate prediction. The second layer consists of a meta-classifier that takes each of the predictions from the baseline models as an input in order to generate a new prediction. In stacking, the models tend to be different algorithms and are fitted on the same dataset, rather than various samples of the training set, as is the case with bagging. Further, a single model is used to learn how to best combine the predictions of contributing models, rather than boosting, in which a sequence of models corrects the predictions of prior models.

Our stacking model included the base models: KNN (neighbors = 100), random forests, histogram-based gradient boosting, adaptive boosting, and decision trees (with no maximum depth).

Adaptive/Gradient Boosting Adaptive boosting is a method that takes weak learning machine models and combines them together to create a stronger learning model. It improves the combined model by adjusting the weight of certain observations and weak learners to increase accuracy of predictions. Gradient boosting is similar, except for instead of adjusting weights, it minimizes the loss of a weak learner over many rounds of boosting to produce a cumulative stronger model. In classification gradient boosting models, the loss function is logarithmic, and defined as:

$$l(yf(x)) = ln(1 + e^{-yf(x)})$$

Hypertuning Hypertuning is the process of testing a myriad of hyperparameters in a machine learning model to produce the best predictions. The result of hypertuning is the optimal values of hyperparamters for a model. We decided to use hypertuning with an XGBoost model, which is, in practice, a regularized version of the gradient boosting model we used in our stacking method. The hyperparamters we sought to optimize were the number of estimators, the learning rate, alpha(Lasso regression regularization term), and lambda(Ridge regression regularization term).

4. Experiments

4.1. Dataset

Our data was gathered from the the City of Chicago Data Portal[1] for years 2001-2021. This publicly available database lists all crimes within Chicago since 2001 and is updated daily. The data set comes out-of-the-box with many insightful features that we utilized during our analysis. In order to incorporate all of the data provided, we used feature engineering techniques in order to extract meta data from the location of the crime, time, and date.

In addition to the crime data collected from the Chicago Data Portal, we also Incorporated income data from 2010 for the 77 different communities in Chicago. Because of this, we filtered our data to only consider crimes in 2010. We selected the 17 columns described in tables 1 and 2 from the total feature engineered columns for the remainder of our project.

4.1.1 Additional preprocessing: One-Hot Encoding/dataset sampling

There were a couple challenges we had to address during pre-processing. Namely: dealing with many categorical features and the data set size. In order to prepare our data for our ML models, we needed to perform one-hot-encoding on our data set increasing the number of columns and sparsity of our data. This lead to computational challenges resulting in extremely slow training time for some ML algorithms. In order to combat this, we take a smaller sample of our data set before splitting into a train and test.

Column Name	Description
Arrest	(T/F): whether crime resulted in arrest
Holiday	(T/F): whether crime happened during a holiday
Weekend	(T/F): whether crime happened on a weekend
Business Hour	(T/F): whether crime happened during business hours
Cluster	Result of a kmeans clustering on X,Y location coordinates (k=40)
Ward	The ward (City Council district) where the incident occurred
District	Indicates the police district where the incident occurred.
Primary Type	Indicates the primary crime type (35 class types)
Location X and Y	coordinates of where the incident occurred
Community Income Data	See table 2

Table 1. Selected Columns of interest.

Column Name
% Housing Crowded
% households below poverty
% aged 16+ unemployed
% aged 25+ without high school diploma
% aged under 18 or over 64
Per Capita income
Hardship index

Table 2. Per community income statistics.

4.2. Baseline KNN

The first step in our experimentation was to create a baseline K-nearest neighbors model to evaluate performance before doing any more data processing. We used five nearest neighbors in this model, and the predictions of the model are shown in the confusion matrix:

4.3. SMOTE

After observing our model struggling to correctly predict when arrests occur, we hypothesized that class imbalance within the arrests variable could be hindering the performance of our model. We used **SMOTE** described in section 3.1 to fix this imbalance.

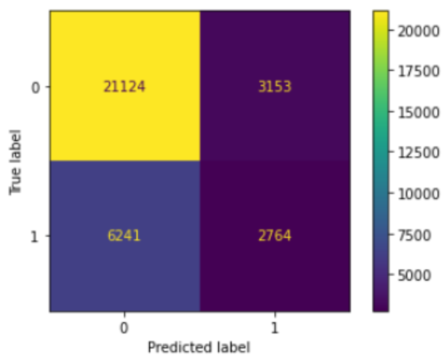

Figure 2. Confusion matrix of our baseline model 1: knn=5 with all features

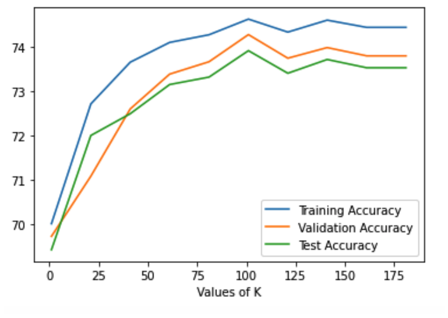

Figure 3. A look at accuracy measurements as k increases (KNN)

4.4. Feature Selection

Our next step was to run feature selection on our final data set using **ExtraTreesClassifier** described in section 3.2 in-order to mitigate over-fitting due to using irrelevant features.

4.5. Optimizing K-value

Not being confident in our first value for K, we decided to make a plot of accuracy rates in the splits of our dataset:

We decided upon analyzing the plot that an appropriate value for k would be about 99(odd number to avoid ties). Knowing KNN classification models perform better with less features, balanced classes, and an appropriate K-value, we expected our new model to perform much better than our first.

4.6. New KNN model

We then created a new KNN model with these changes to our dataset. The new confusion matrix is shown below:

Clearly, implementing these changes improved our model's ability to determine when arrests were occurring. Some of that improvement may also be contributed to the feature selection and optimized k-value.

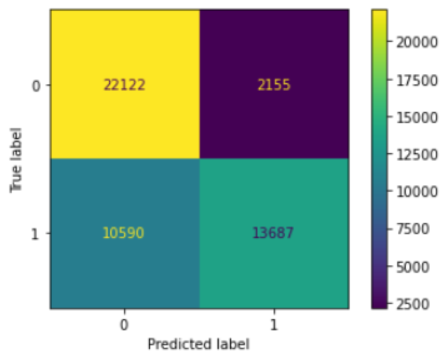

Figure 4. confusion matrix of model 4: knn=99, with feature selection and balanced class labels

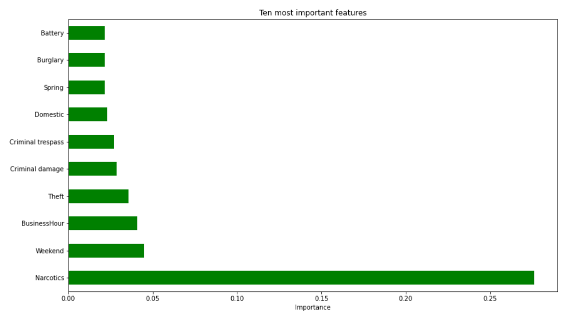

Figure 5. Top 10 features using ExtraTreesClassifier

Metric	Score
Train Acc.	79.646%
Valid Acc.	71.330%
Test Acc.	71.800%
Precision Score	0.467
Log loss	9.749

Table 3. MODEL 1: baseline, knn = 5 with all features

4.7. Stacking

In our next model we wanted to get a better feel for how would predict the probability of an arrest occurring. Thus, we chose to run a logistic regression meta classifier in our stacking model described in section 3.3. We used a cross-validation value of ten in our model. We then computed the accuracy across the training, validation, and test set, as well as computed the precision of the model, to be discussed in the results section.

4.8. Hypertuning

The method we chose for hypertuning was Bayesian optimization. We did this because it the fastest, most effective method according to Druce Verte's article on

Metric	Score
Train Acc.	79.013%
Valid Acc.	68.158%
Test Acc.	68.128%
Precision Score	0.667
Log loss	11.008

Table 4. MODEL 2: baseline, knn = 5 with balance class labels (Arrest)

Metric	Score
Train Acc.	71.859%
Valid Acc.	71.768%
Test Acc.	71.906%
Precision Score	0.730
Log loss	9.700

Table 5. MODEL 3: baseline, knn = 5 with balanced class labels (Arrest) and feature selection using features from figure 1 5

Metric	Score
Train Acc.	73.918%
Valid Acc.	73.882%
Test Acc.	73.751%
Precision Score	0.864
Log loss	9.066

Table 6. MODEL 4: baseline, knn = 99 with balance class labels (Arrest)

Metric	Score
Train Acc.	74.1%
Valid Acc.	74.2%
Test Acc.	74.1%
Precision Score	0.979
Log loss	8.911

Table 7. Hypertuning with XGBoost using feature selection and balanced class labels

Metric	Score
Train Acc.	74.738%
Valid Acc.	74.107%
Test Acc.	73.924%
Precision Score	0.857

Table 8. Stacking Ensemble with feature selection and balanced class labels

towardsdatascience.com[**?**]. This gave us optimal values of 300 for number of estimators, .01 for learning rate, $6.75e^{-}5$ for lambda, and $1.53e^{-}5$ for alpha.

4.9. Software

Here is a list of Software used for the project:

- Environment: Python, anaconda computing environment, jupyter lab/notebook
- GitHub for project management
- ML libraries: scikit-learn, imbalanced-learn, optuna

4.10. Hardware

None relevant

5. Results and Discussion

5.1. KNN models

Tables 3, 4, and 5 depict our accuracy scores from our baseline models (models 1-3). Our first baseline model including all of the features for knn showed promising results out-of-the-box, but upon further analysis, the model suffers from overfitting, and struggles with precision (0.46). Model 2 performs slightly better after we correct for class imbalance and model 2's precision score increases. Because we correct for class imbalance in this case, our test and validation accuracy's decrease from model 1. This is expected, as our model does not have an uneven representation of non-arrests, thus it cannot rely on only guessing non-arrest for the class label. After evaluating our baseline models, and optimal value of k, we construct our best preforming knn model with our balanced data set, top 10 features, and optimal k=99. This model has the best overall performance with highest accuracy and precision score.

A lot can still be done to improve the accuracy of these models. For example, one set back in our data is that many of the features we use are categorical, thus need to all be one-hot encoded in order to be useful in ML models. Adding more robust quantitative data about the crimes would increase the overall accuracy.

A challenge we ran into during our analysis was the computational payload of training these various models. We had to work with a subset of our data set in order to increase training turn around. This limited us to methods that could be feasibly completed within a few minutes, and decreases our search space for hyperparameter tuning.

Another area we could improve our models with is incorporating geospatial data from the crime incidents. Using location data can provide a more holistic picture of were incidents are happening that can also be used in post processing experiments.

5.2. Stacking

For the stacking model, we use KNN, decision tree, random forest, adaptive boosting, and histogram-based gradient boosting as our intermediate classifiers and logistic re-

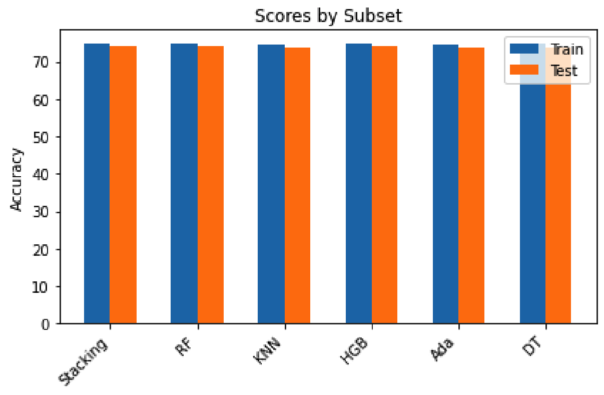

Figure 6. Differences in accuracy scores for individual models compared to stacking emsemble

gression as our meta-classifier. This is done using the balanced data set, having selected only the top 10 most important features. While we originally thought our stacking model had performed quite well with an accuracy of 74% on the test set, however upon further examination of the performance of each individual classifier we determined that the stacking model may not be worth the computational expense due to the accuracy of each individual model being roughly identical with that of the stacking model (Figure 6).

6. Conclusions

Our group analyzed the outlook on crime in Chicago in the year 2010. We wanted to provide valuable insights into how and why people get arrested. Thus, we created various models trained on the same dataset, measuring precision and accuracy. Models included KNN, random forest, gradient boosting, adaptive boosting, decision tree, logistic regression stacking, XGBoosting, and hyptertuning. The performance of these models were improved by correcting class imbalance with SMOTE, and selecting important features with ExtraTreeClassifier.

From our analysis, the hypertuned XGBoost model performed the best in terms of accuracy and precision, with scores of 74.2% and 97.88% respectively. The accuracy is only slightly better than the logistic regression stacking model, but the precision in significantly higher. Furthermore, if given superior computer hardware, we claim that the XGBoost hypertuning model could have performed even better, because our hypertuning search space was limited by our computers. We also believe that the stacking model could perform better if using different parameters. Perhaps hypertuning could help improve that model's performance as well. Our KNN model doesn't have much room for improvement, but it did prove that the SMOTE and feature selection methods were highly effective in improving models.

In conclusion, we believe our models were effective in giving useful insight into the way people are arrested in Chicago. The most interesting thing we learned was that criminal Narcotic use was by far the most impactful feature in determining whether someone was arrested. We hope our models can be built upon with higher-level computers to confirm this relationship, and perhaps use additional features to find more variables and help describe the way people in Chicago are arrested.

We anticipated community area and type of crime being highly predictive of being arrested.

7. Acknowledgements

We want to give our thanks to the Chicago data portal for provided the insightful, clean data. Also the students in Piazza for being helpful in answering questions and supporting us. And most of all, Sebastian Raschka has been extremely accomidating in pushing back deadlines, answering quesitons, and allowing us to present in our own preferred way. Thanks for a great semester.

8. Contributions

Adam found the dataset and did a lot of the cleaning to enable modeling. He also helped the group collaborate via Github. Kate assisted in creating and improving models throughout the project. Shane helped implement models and analyse their results. For the report, Kate worked on the abstract and introduction, Adam did the related work and added many plots. Shane worked on the proposed methods and experiments. All members contributed to the results and conclusion.

References

[1] Crimes - 2001 to present, city of chicago, data portal. *Chicago*.

[2] S. N. Christian, K. R. Majeed, and S. O. Etinosa. Application of data analytics techniques in analyzing crimes, 2018.

[3] M. S. Gerber. Predicting crime using twitter and kernel density estimation. *Decision Support Systems*, 61:115–125, 2014.

[4] H.-W. Kang and H.-B. Kang. Prediction of crime occurrence from multi-modal data using deep learning. *Plos One*, 12(4), 2017.

[5] L. Lochner. Education and crime. *The Economics of Education*, page 109–117, 2020.

[6] G. Rosser and T. Cheng. Improving the robustness and accuracy of crime prediction with the self-exciting point process through isotropic triggering. *Applied Spatial Analysis and Policy*, 12(1):5–25, 2016.