# Improving Machine Learning based Crime Prediction for the city of Philadelphia

**Alex Zavalny**[1] az548@drexel.edu
**Mustafa Eren**[2] me624@drexel.edu
**Naman Bajpai** [1] nb3283@drexel.edu
**Justin Getty**[1,2] jrg349@drexel.edu
**Ishan Patel**[1] ip335@drexel.edu
**Rithvik Sukumaran**[1] rs3673@drexel.edu
**Arya Gopikrishnan**[1] ag3974@drexel.edu
**Shams Abrar**[1] sa3868@drexel.edu
**Andrew Rogers**[2] ar3933@drexel.edu

Drexel University, [1]College of Computing and Informatics, [2]LeBow College of Business

## Abstract

This study delves into predicting crime types in Philadelphia using cutting-edge machine learning techniques. By harnessing the power of deep neural networks, we uncover that location and time are the most pivotal features for accurate crime prediction in urban areas. Our models, on average, have a 17% increase in accuracy than state of the art, providing a robust framework for understanding crime patterns. This study aims to empower policymakers with data-driven insights, enabling them to craft legislation that better protects their communities. Our Code and Models used in this study are available for open source use.

## 1 Introduction

Crime prediction in urban areas is influenced by socio-economic and demographic factors like poverty and population density, environmental and spatial variables such as land use and urban infrastructure, and temporal patterns including time of day, season, and weather conditions [4, 6]. Historical crime data and community social dynamics also play critical roles in developing accurate predictive models. Machine learning based models can be used to identify patterns that are difficult for a human to understand and forecast future occurrences [6, 8]. A better understanding of crime statistics can determine the best route for policy makers to analyze crime, especially in areas where it's most prevalent, and use predictive models to implement legislation to mitigate criminal activities. Further uses of machine learning include extrapolating more information using the known factors such as time and location of crime events in one area with those in areas across the United States or across the globe or monitoring and analyzing the relationships among causes of crime events (such as firearms) and the location of crime events. Philadelphia was chosen to conduct an analysis since its crime data is publicly accessible online, and is in the top 10 most populated cities in the United States.

## 2 Related Works

Former work by Alparslan et al. similarly applied several machine learning models to the prediction of crime in Philadelphia using multiclass classification models [5]. Their work focused on considering various approaches and factors into their predictive model including working with 30 types of crime, clustering crime hotspots, and working with supervised learning models to increase classification accuracy. Their work addresses the issue of dividing the city into different grid cells. Through creating clusters using crime points for each year instead of using a grid, then clusters are stacked on top of each other to remove one of the constraints, the time element. Their models have a mean accuracy of 20.8% while ours have a 24.5% as highlighted in Table 2. In addition, our models have a smaller range of accuracies of 14% attributed to our improved model robustness while Alparslan et al. have a larger range of 27%. Another important insight is that they highlighted how location and time based features proved to be the most important when predicting crime which is consistent with what we found as well.

Crime prediction and analysis using machine learning techniques have also been explored in other cities, highlighting diverse methodologies and outcomes. A study by Kim et al. analyzed Vancouver's crime data and developed crime prediction models using K-Nearest Neighbors and Boosted Decision Trees. The models achieved accuracies of 39% and 44%, respectively, providing insights into crime analysis methodologies and the potential of machine learning in this domain. The study also reviewed other related research, emphasizing the use of different techniques for crime prediction [3]. We used K Nearest Neighbors and Extreme Gradient Boosting (XGBoost) models within our own study.

Khan et al. developed a crime prediction model for San Francisco using Naive Bayes, Random Forest, and Gradient Boosting Decision Tree algorithms. The model achieved notable accuracies of 65.82%, 63.43%, and 98.5%, respectively, for predicting crime types [7]. This research utilized the SF Crime Classification dataset and highlighted the effectiveness of Gradient Boosting Decision Trees in crime prediction tasks which we ended up using in our own study.

Another relevant study is the time series analysis conducted by Schinasi et al., which investigated the relationship between daily temperature fluctuations and crime rates in Philadelphia. The findings revealed a positive, linear association between deviations from the seasonal mean heat index and rates of violent crime and disorderly conduct, particularly in colder months [2]. This study underscores the importance of considering environmental factors in crime analysis.

These studies collectively demonstrate the application of machine learning and statistical analysis in crime prediction, showcasing varying approaches and outcomes across different geographic and environmental contexts. The integration of spatial and temporal data, as well as the consideration of environmental variables, provides a comprehensive understanding of crime dynamics and supports the development of more accurate prediction models.

## 3 Problem Statement

In this study, we aim out if we can develop a machine learning model to accurately classify the type of crime given its details. That way, given information of a new crime, police could use this model to help them classify the type of crime that occurred to help more accurately assess the severity of the case.

In machine learning, we often deal with the task of learning a function $f$ that maps input data $x$ to an output $y$. The model's objective is to learn the function $f$, which describes the relationship between the input $x$ and the output $y$. Mathematically, this relationship can be written as $y = f(x)$. To train the model, we use a set of training data $\{(x_i, y_i)\}$ for $i = 1, 2, \ldots, N$, where $N$ is the number of data points. Each $x_i$ is an input vector, and $y_i$ is the corresponding output. The model aims to approximate the true function $f$ by providing a hypothesis function $\hat{f}$ parameterized by a set of parameters $\theta$, such that $y \approx \hat{f}(x; \theta)$.

To measure how well the model's predictions $\hat{f}(x; \theta)$ match the actual outputs $y$, we use a loss function $L$. For classification tasks, a common choice is the log loss or cross-entropy, defined as

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{f}(x_i;\theta)) + (1-y_i)\log(1-\hat{f}(x_i;\theta))\right]$$

The goal is to find the parameters $\theta$ that minimize the loss function $L$. This optimization process is typically done using algorithms like Gradient Descent. Mathematically, we solve for $\theta^* = \arg\min_\theta L(\theta)$. In Gradient Descent, we iteratively update the parameters $\theta$ in the direction that reduces the loss. The update rule for $\theta$ is $\theta \leftarrow \theta - \eta \nabla_\theta L(\theta)$, where $\eta$ is the learning rate, and $\nabla_\theta L(\theta)$ is the gradient of the loss function with respect to $\theta$.

After training, the learned model $\hat{f}(x; \theta^*)$ can be used to make predictions on new, unseen data. Thus, for a new input $x_{\text{new}}$, the predicted output is $\hat{y} = \hat{f}(x_{\text{new}}; \theta^*)$.

For example, in crime prediction, the input $x$ could represent various features such as time of day, location, and historical crime data, while the output $y$ represents the likelihood of a crime occurring. The model learns the function $f$ that best maps these features to the crime likelihood, enabling law enforcement to predict and potentially prevent future crimes by deploying resources more effectively.

## 4 Dataset Analysis

### 4.1 Feature Overview

Raw crime data isn't the most interpretable and is geared towards transformation rather than pure understanding. Given the nature of our features and our intended use, cleaning meant encoding and sampling, as well as restructuring the data to be fed into our model. The process we underwent involved removing duplicate and redundant features, reshaping and parsing existing features, filling in missing values, and stitching together outside data to bring validity to our input, before running any sort of processing. Table 1 shows our final features after all preprocessing was done.

| Feature Name | Description |
|---|---|
| $\text{point}_x$ | latitude of the crime |
| $\text{point}_y$ | longitude of the crime |
| minute | from police report timestamp |
| hour | from police report timestamp |
| day | from police report timestamp |
| weekday | from police report timestamp |
| month | from police report timestamp |
| year | from police report timestamp |
| unemployment rate | from Philadelphia per month |

Table 1: Description of final dataset features used combining temporal and location features

## 4.2 Data Source

We sourced our main data from OpenDataPhilly, a copyright free and open-source collection of data in Philadelphia and surrounding regions. Our data spans from 2006 to 2024, and collected from the Crime Incidents dataset using the Carto API. We also used the unemployment rate over time for the city of Philadelphia from the Federal Reserve Bank of St. Louis.

## 4.3 Feature Preparation

1. Irrelevancy

   We found that isolating the dispatch date and time, location block, general crime code, and X & Y coordinates provided a clean starting point of features that directly contribute to the crime type, along with the type itself. The geometric representation, or other unique id's have no relevance in prediction or visualization of the data, so they were all dropped.

2. Redundancy

   In terms of repeat or redundant features, isolating only the dispatch DateTime combined string allowed us to streamline the time features of our data, so that we have a cleaner visual but can also encode the single column later in the preprocessing. Similar results were found in the district, service area, district keys, and UCR general code. These features overlapped, as well as incorrectly association out of range numeric values to important features, which would diminish the accuracy of our model.

3. Accuracy

   Since date and time are a strong predictor of a crime, we expanded our singular DateTime data into individual temporal components such as Minute, Day, Hour, Year, Weekday, and Month. Finally, we added unemployment rate to our data, which is the number of unemployed people as a percentage of the labor force. This percent describes the number of people in an area who are capable of working but choose not to. Unemployment rate has a tremendous effect on financial stability, health & well being, and importantly crime rates. By adding the unemployment rate of the community where a crime occurred to our dataset, we were able to increase accuracy and diversify our data.

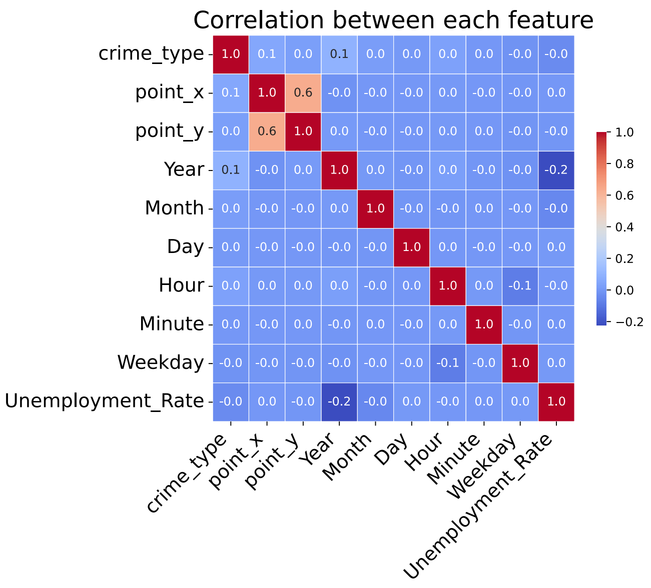


Figure 1: Heat-map of preprocessed features demonstrating lack of multicollinearity

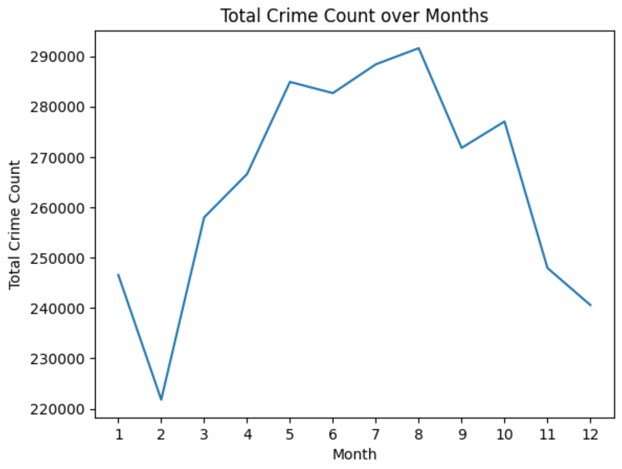
## 4.4 Crime Classifications

Figure 2 includes the original crime types that are associated with our dataset. The most frequent crime types are "All Other Offenses" and "Other Assaults", and the least frequent crime types are "Homicide - Justifiable" and "Homicide - Gross Negligence". The categories "All Other Offenses" and "Other Assaults" are an amalgamation of crime types either too rare or too common. For example, "All Other Offenses" is comprised of crimes like traffic infractions, kidnappings, quarantine violations, perjury, and other crimes on the frequency extremes. Due to the infrequency of homicides compared to other crimes, and that the distinction between negligence, justifiable, and criminal with our features being near infeasible, we decided to classify all homicides into one general category called Homicide. In fact, if there were 3500 homicides, on average 15 would be classified as justifiable with just 1 being considered grossly negligent. The other 3484 homicides would be classified as criminal, which further shows the disparity within the homicide subset. After the preprocessing, our data set has 3,177,852 samples.

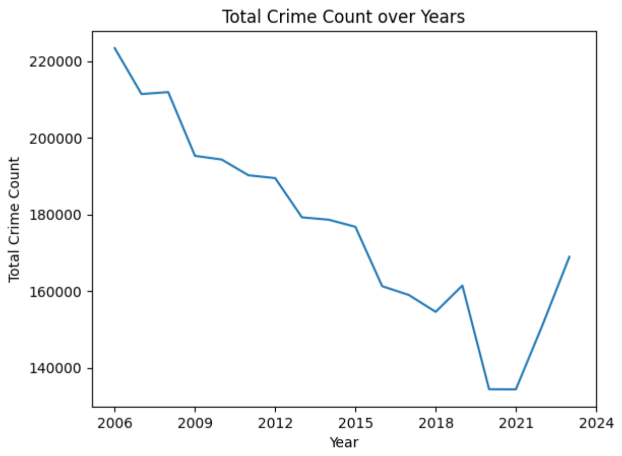
Figure 2: Crime Types and Frequency for Classification

### 4.5 Analysis of Final Features

In our final features, the location is measured by the latitude and longitude data of the crime, where the X values represent the latitude data and the Y values represent the longitude data. Latitude measures the distance north or south of the equator line, while longitude measures the distance east or west of the Prime Meridian. The range of longitudinal values is higher than the range of the latitudinal values in Philadelphia, thus making the Y features more important (as seen in the graph below).
There is a visible trend in the crime counts over the months. The number of crimes tend to increase after February till August, with the peak of crimes being August. After August, there is a visible drop in the total number of crimes, with the month with the lowest numbers of crime being February.



For years, there has been a visible downwards trend in the number of crime counts from 2006. The lowest number of crime counts from 2006 is 2020, which falls during the height of COVID-19. As cities like Philadelphia started to return to normal operations from 2021, we have started seeing an increase in number of crimes, which is the first substantial increase in our data.



Our final features, the X values for our model, along with their weighted importance against Crime Type are shown in Figure 2. These were chosen through a process of trial and error, with virtually all combinations being tested before determining that points X/Y, time from minute to year, unemployment rate, and day of the week had the lowest error percentage. The location coordinates allow the model to isolate which areas in the city are most likely for a particular crime to occur. Crimes are typically consistently committed in certain locations due to many factors, including but not limited to the distance from a police station, whether the area is residential or commercial, the population density, or average income. In addition to it's specificity, having the coordinates eliminates the need to have all aforementioned factors as features, as location is all-encompassing. Once the

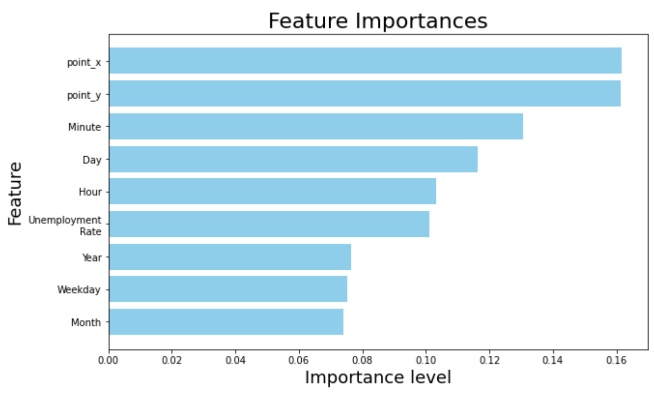


Figure 3: Feature Importances Relative to Predicting Crime Type

model knows the location, it uses the time of day to the minute to further predict the crime type because most crimes are carried out at consistent times. For example, crimes such as theft, robbery, and vandalism are most likely to be committed at night when it's dark out [1]. In contrary, crimes such as drug violations, fraud, or forgery are indiscriminate to the time of day. Unemployment rate is then factored in as it predicts the wealth of a particular location. Higher unemployment rates are inversely correlated with average family income, which in turn motivates individuals in these neighborhoods to look for alternative means of income such as theft, robbery, and burglary. Lastly day of the week is taken into account, as majority of crimes perpetrated increase in volume earlier in the week, especially Monday and Tuesday. Some crime types may spike during weekends such as weapon violations and liquor law violations, whereas crimes like burglary and auto theft occur most often on Monday and Tuesday.

## 5 Empirical Evaluation

### 5.1 Our Approach

With our data analysis in mind, we first preprocessed the data using standardization, one hot encoding, and SMOTE techniques. Then, we designed numerical features for our machine learning models to use during training. We then compared the performances of different machine learning models and deep neural networks.

### 5.2 Data Processing and Feature Engineering

Our pre-processing pipeline includes three key techniques: standardization, one-hot encoding and synthetic minority over-sampling technique (SMOTE). Additionally, we handled NaNs (missing values) which increased data integrity.

1. Handling Missing Values

   Before applying preprocessing techniques to our features we had to handle the NaN values. The features Point X and Point Y had a lot of missing values. The rows with missing X and Y coordinate values were dropped. Missing values in the Unemployment Rate feature, was handled by filling them with the most recent unemployment rate.

2. Standardization

   Standardization is a feature scaling technique used to center the data around the mean and scale it to have a standard deviation of one. In this study, standardization is applied to the numerical and temporal features, Point X, Point Y, Minute, Day, Hour, Unemployment Rate, Month, Weekday, and Year. These values vary in range and distribution. Furthermore, the temporal features are measured in different units, and without standardization, the year and day features dominate over the rest due to bigger values. Standardizing these values ensures that each of these features contribute to the model equally and improves the model's convergence. It also removes bias and prevents features with larger numerical ranges from disproportionately influencing the model's predictions.

3. One Hot Encoding for Multi-Class Classification

   One hot encoding is a representation of categorical variables as binary vectors. This can be useful for handling multi-class classification problems where the target variable has more than two classes and there are no ordinal relationship between them. Encoding was applied to the categorical features in our model such as Crime Types, Dispatch DateTime, UCR General Code and Police Service Area. While some crimes were more frequent than others, there was no way to rank these crimes. We used one hot encoding to convert these features into a set of binary columns.

4. SMOTE

   Oftentimes, there can be an imbalance in datasets. In the domain of crime prediction, certain types of crime are less frequent than others. For example, gambling violation was seen to be less frequent than thefts or assaults. This imbalance can lead to to biased model prediction that performs well mostly on the majority class. Moreover the location Point X and Point Y may have a different influence on Crime Type than time of day, Minute and Hour. SMOTE technique was applied to the entire feature set including Point X, Point Y, Minute, Day, Hour, Unemployment Rate, Month, Weekday and Year, to balance the training data.

It over-samples the minority by generating synthetic data. However, oversampling the minority class can reduce the relative frequency of patterns learned from majority class. This can be detrimental if these patterns are important for generalization.

### 5.3 Machine Learning Models

1. **K Nearest Neighbors** The k-nearest neighbors algorithm(KNN) is a non-parametric supervised learning classifier. It relies on proximity to categorize or predict the grouping of a specific data point. Although it can be applied to both regression and classification problems, it is predominantly utilized as a classification algorithm. The underlying principle is based on the idea that similar data points tend to cluster together.

   For classification:

   $$y_{\text{new}} = \text{argmax}_y \sum_{x_i \in N_k(x_{\text{new}})} I(y_i = y)$$

   For regression:

   $$y_{\text{new}} = \frac{1}{k} \sum_{x_i \in N_k(x_{\text{new}})} y_i$$

   In the above equations:

   - $y_{\text{new}}$ represents the predicted class label or value for the new data point $x_{\text{new}}$.
   - $N_k(x_{\text{new}})$ denotes the set of $k$ nearest neighbors of the new data point $x_{\text{new}}$ in the dataset.
   - $x_i$ represents a data point in the dataset.
   - $y_i$ is the class label (in classification) or the value (in regression) associated with the data point $x_i$.
   - $I(y_i = y)$ is the indicator function that returns 1 if the condition $y_i = y$ is true and 0 otherwise.
   - $\text{argmax}_y$ represents the value of $y$ that maximizes the expression following it. In the classification equation, it identifies the class with the highest count among the $k$ nearest neighbors.
   - $\frac{1}{k}\sum_{x_i \in N_k(x_{\text{new}})} y_i$ is the average (mean) of the values associated with the $k$ nearest neighbors in regression.

   In the context of crime prediction, KNN is employed to leverage spatial and temporal patterns from historical crime data, enabling the identification of localities with similar historical criminal activities. By considering the proximity of locations in the feature space, KNN provides an intuitive and interpretable framework for predicting potential crime hotspots based on the behaviors of neighboring areas.

2. **Decision Tree** Decision Trees are non-parametric supervised learning algorithms used for classification and regression tasks. They employ a hierarchical tree structure with root nodes, branches, internal nodes, and leaf nodes.

   The tree starts at the root, branching into decision nodes based on features. Each node evaluates conditions, leading to homogeneous subsets represented by leaf nodes, each indicating a possible outcome. The algorithm utilizes a divide-and-conquer approach, employing a greedy search to identify optimal split points in the tree. The recursive splitting process continues until most or all records are classified into specific class labels.

   Mathematically, Decision Trees minimize impurity at each node during the splitting process. Impurity is often measured by metrics like Gini impurity or entropy. Let $D$, $C_i$, and $p_i$ represent the data set at a node, the set of instances for class $i$, and the proportion of instances of class $i$ in $D$, respectively.

   The impurity ($I(D)$) is defined as:

   $$I(D) = 1 - \sum_i p_i^2$$

   Here:

   - $D$ represents the dataset at a given node.
   - $C_i$ is the set of instances belonging to class $i$.
   - $p_i$ is the proportion of instances of class $i$ in $D$.

   Decision Trees are well-suited for crime prediction tasks due to their flexibility in capturing intricate decision boundaries. The algorithm's interpret able nature, producing a transparent tree structure, allows us to comprehend and act upon factors influencing crime patterns.

3. **Random Forest** The Random Forest Algorithm serves as a sophisticated extension of decision trees, amalgamating inputs from multiple decision trees to generate a singular output. Functioning as a supervised algorithm, it adeptly addresses both regression and classification tasks.

While decision trees are ubiquitous in supervised learning, they exhibit susceptibility to issues such as bias and overfitting. However, the Random Forest Algorithm mitigates these concerns by aggregating multiple decision trees into an ensemble. This ensemble approach enhances predictive accuracy, particularly when the constituent trees are uncorrelated.

The algorithm is characterized by three principal hyper-parameters that necessitate pre-training configuration: node size, the quantity of trees, and the number of features sampled. Following parameterization, the Random Forest Classifier is poised to tackle regression or classification challenges.

Comprising a set of decision trees, each constructed from a data sample drawn with replacement from a training set (bootstrap sample), this ensemble method introduces variability and reduces correlation among trees through feature bagging. Notably, one-third of the training sample is earmarked as test data, referred to as the out-of-bag (oob) sample, a concept revisited later. For regression tasks, predictions involve averaging individual decision trees, while for classification tasks, a majority vote determines the predicted class. Subsequently, the oob sample is used for cross-validation.

The Random Forest algorithm can be mathematically described through its key steps:

Bootstrap Sampling: For each tree $i$ in the forest ($i = 1$ to $M$), a bootstrap sample $D_i$ is obtained by randomly selecting, with replacement, $n$ data points from the training set. This is represented as:

$$D_i = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Feature Randomization: $m$ features are randomly selected for each tree, forming the set $F_i$:

$$F_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,m}\}$$

Decision Tree Building: A decision tree $T_i$ is built for each bootstrap sample $D_i$ using the selected features $F_i$. Denoting the decision tree as $T_i(x)$, this process involves recursively partitioning the data based on the selected features to create a set of rules.

Voting (for Classification) or Averaging (for Regression): For classification, the final prediction $\hat{y}$ is determined by the majority vote of all trees:

$$\hat{y} = \text{argmax}_y \sum_{i=1}^{M} \mathbb{I}(T_i(x) = y)$$

For regression, the final prediction $\hat{y}$ is the average of the individual tree predictions:

$$\hat{y} = \frac{1}{M} \sum_{i=1}^{M} T_i(x)$$

Here, $x_i$ represents the feature vector of the $i$-th data point, $y_i$ is its corresponding label, $f_{i,j}$ is the $j$-th randomly selected feature for tree $i$, and $\mathbb{I}(\cdot)$ is the indicator function.

4. **Extreme Gradient Boosting (XGBoost)**

XGBoost, or Extreme Gradient Boosting, is a powerful ensemble learning algorithm that falls under the gradient boosting umbrella. It is renowned for its adaptability, excelling in both regression and classification tasks. The algorithm sequentially trains weak learners, usually decision trees, correcting errors with each iteration. To prevent over fitting, XGBoost employs regularization techniques and tree pruning. Mathematically, the algorithm minimizes a loss function that includes regularization terms, optimizing its predictive performance. In crime prediction, XGBoost proves invaluable by efficiently analyzing various data features, determining feature importance, and providing accurate predictions.

XGBoost minimizes an objective function that consists of a sum of a differentiable loss function and a regularization term. The objective function for XGBoost in a generic form can be written as follows:

$$\text{Objective}(\Theta) = \sum_{i=1}^{n} \ell(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Here:

- $\Theta$ represents the set of model parameters to be optimized.
- $n$ is the number of training samples.
- $\ell(y_i, \hat{y}_i)$ is the loss function, measuring the difference between the predicted $\hat{y}_i$ and the true $y_i$ for the $i$-th sample.
- $K$ is the number of trees (weak learners) in the ensemble.
- $\Omega(f_k)$ is the regularization term, penalizing the complexity of individual trees.

In practice, for regression problems, the loss function $\ell$ is often the mean squared error, and for classification, it may be the cross-entropy loss.

The regularization term $\Omega(f_k)$ typically takes the form of the sum of squared values of the leaf scores in each tree, penalizing complex trees:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Here:

- $T$ is the number of leaves in the tree.
- $w_j$ is the score assigned to leaf $j$.
- $\gamma$ and $\lambda$ are regularization parameters that need to be tuned.

Its speed and ability to handle extensive data sets make it a go-to choice for predictive policing, aiding law enforcement in identifying patterns and enhancing crime prevention strategies. The algorithm's flexibility and optimization capabilities, achieved through fine-tuning hyper-parameters, contribute to its widespread adoption across diverse applications, including the complex and critical domain of crime prediction.

### 5.4 Logistic Regression

Logistic Regression is a statistical model used primarily for binary classification tasks. It estimates the probability of an instance belonging to a default class (e.g., occurrence of a crime).

Mathematically, it is expressed as:

$$p = \frac{1}{1 + e^{-z}}$$

where $z$ is the linear combination of the input features:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Here, $p$ represents the probability that the outcome is the default class, $e$ is the base of the natural logarithm, $\beta_0, \beta_1, \ldots, \beta_n$ are coefficients representing the impact of the corresponding features $x_1, x_2, \ldots, x_n$.

### 5.5 Deep Neural Network

Deep Neural Networks (DNNs) are composed of multiple layers of neurons, each applying a nonlinear transformation to the inputs from the previous layer. They are used for complex pattern recognition.

Each neuron computes:

$$y = f(w \cdot x + b)$$

where:

- $x$ is the input vector to the neuron.
- $w$ is the weight vector associated with the inputs.
- $b$ is the bias term.
- $f$ is a nonlinear activation function, such as ReLU or sigmoid.
- $y$ is the output of the neuron.

### 5.6 Encoder Transformer

The Encoder part of a Transformer model processes sequences via self-attention mechanisms, encoding them into a continuous representation.

The self-attention for each head is calculated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

where:

- $Q$ are the query matrices derived from the input as concatenated vector representations of the input tokens.
- $K$ are the key matrices which are references to each of the input token embedding vectors.
- $V$ are the value matrices that represent the input token embedding vectors in relation to the previous tokens.
- $d_k$ is the dimension of the keys and queries, used for scaling.
- The softmax function is applied to normalize the weights.

Each component of the model is designed to capture different aspects of the input data, making Transformers powerful for tasks involving sequential data such as time-series analysis.

### 5.7 Experimental Results

For our experiments, we trained our models before and after rebalancing the data to compare the performances. The parameters we used for our model implementations are:

1. **K Nearest Neighbors**

   For the K Nearest Neighbors implementation, our model uses 5 neighbors in the feature space when making the classification decision. We use uniform weights for all 5 neighbors. The leaf size is set to 30, affecting the speed and memory efficiency. The distance metric used for our algorithm is Minkowski which defaults to the euclidean distance as our power parameter is set to 2.

2. **Decision Tree**

   The decision tree classifier is configured to choose the best split at each node. The maximum depth is set to None, allowing the nodes to expand until pure or reaches a minimum sample split of 2. Our model ensures each leaf has at least 1 sample and all features are considered for splitting. For cost complexity, pruning is set to 0.

3. **Random Forest**

   The Random forest classifier is an ensemble model made up of 75 decision trees each with the same configuration as above.

4. **XGBoost**

   The XGBoost classifier is likewise an ensemble of decision trees using gradient boosting. Our model uses the default settings as specified in the Python documentation. Noteable parameters are a learning rate of 0.3 and a max depth of 6 leaves for each decision tree.

5. **Logistic Regression**

   The Logistic Regression model used an l2 penalty for regularization, and uses the one-vs-rest (OvR) scheme for multiclass classification.

6. **Deep Neural Network**

   The neural network model is a sequential model built using TensorFlow's Keras API. The input features are treated as 1-dimensional vectors, with the input shape (batch, number of features). The first layer of the model is a dense (fully connected) layer consisting of 128 neurons with the ReLU activation function, which introduces non-linearity and helps the model learn complex patterns in the data. Following this, there is another dense layer with 64 neurons, also using the ReLU activation function, further enabling the model to capture intricate relationships within the input data. The final layer is an output layer with 26 neurons, corresponding to the 26 classes in the classification task, and utilizes the softmax activation function to output a probability distribution over the classes. The model is compiled with the Adam optimizer, which is an efficient variant of gradient descent, and the sparse categorical crossentropy loss function, suitable for multi-class classification problems where the target labels are integers. The model's performance is evaluated using accuracy as a metric.

7. **Encoder Transformer**

   For the encoder transformer, our features are treated as 1 dimensional vectors. The input shape of the model is (batch, feature dimension/sequence size, feature). Our model uses 2 multi-head attention mechanisms with 64 self-attention heads which are then connected to a dropout layer that drops 10% of the activations to improve generalization. After normalizing the outputs from the attention mechanism, they are passed into a feedforward layer with 128 neurons. Then, the feedforward layer is connected to a 1 dimensional Global Average Pooling layer which is connected to our output layer of 26 neurons for each of the 26 classes.

### 5.8 Results

Table 2 gives the accuracies obtained by each model before and after applying SMOTE to rebalance the data. It appears that applying SMOTE reduces model accuracy, and the Neural Network was the most accurate before SMOTE and XGBoost was the most accurate after applying SMOTE. Figure 4 shows the learning curves for the machine learning estimators we used with their accuracies plotted over time. It is seen that before SMOTE rebalancing, there is a higher variance of accuracies per training epoch. Whereas using SMOTE generally smoothens out the accuracy curves for the models except for Logistic Regression. Figure 5 shows the two deep learning models used and includes loss in addition to accuracy over training epochs. Despite the Transformer model utilizing feed-forward neural networks in its multiple self attention heads, the simpler deep neural network model outperformed likely due to its higher generalizability.

Table 2: Machine Learning Model Accuracies* before and after Rebalancing with SMOTE

| **Model** | **Original Data** | **SMOTE** |
|---|---|---|
| Random Forest | 0.257557 | 0.012270 |
| Decision Tree | 0.166031 | 0.012270 |
| K Nearest Neighbors | 0.205761 | 0.012270 |
| Logistic Regression | 0.249745 | 0.012270 |
| XGBoost | 0.245571 | 0.133506 |
| Neural Network | 0.303366 | 0.124924 |
| Transformer | 0.292365 | 0.098226 |
| Mean Accuracy | 0.245771 | 0.057105 |

*The accuracies for the validation/test set were used here.

## 6 Conclusion

From our experimental results, we find that Neural Network based Deep Learning models produced the highest accuracy for crime prediction. This is likely due to the neural network more effectively capturing the nonlinearities in the crime data through having a

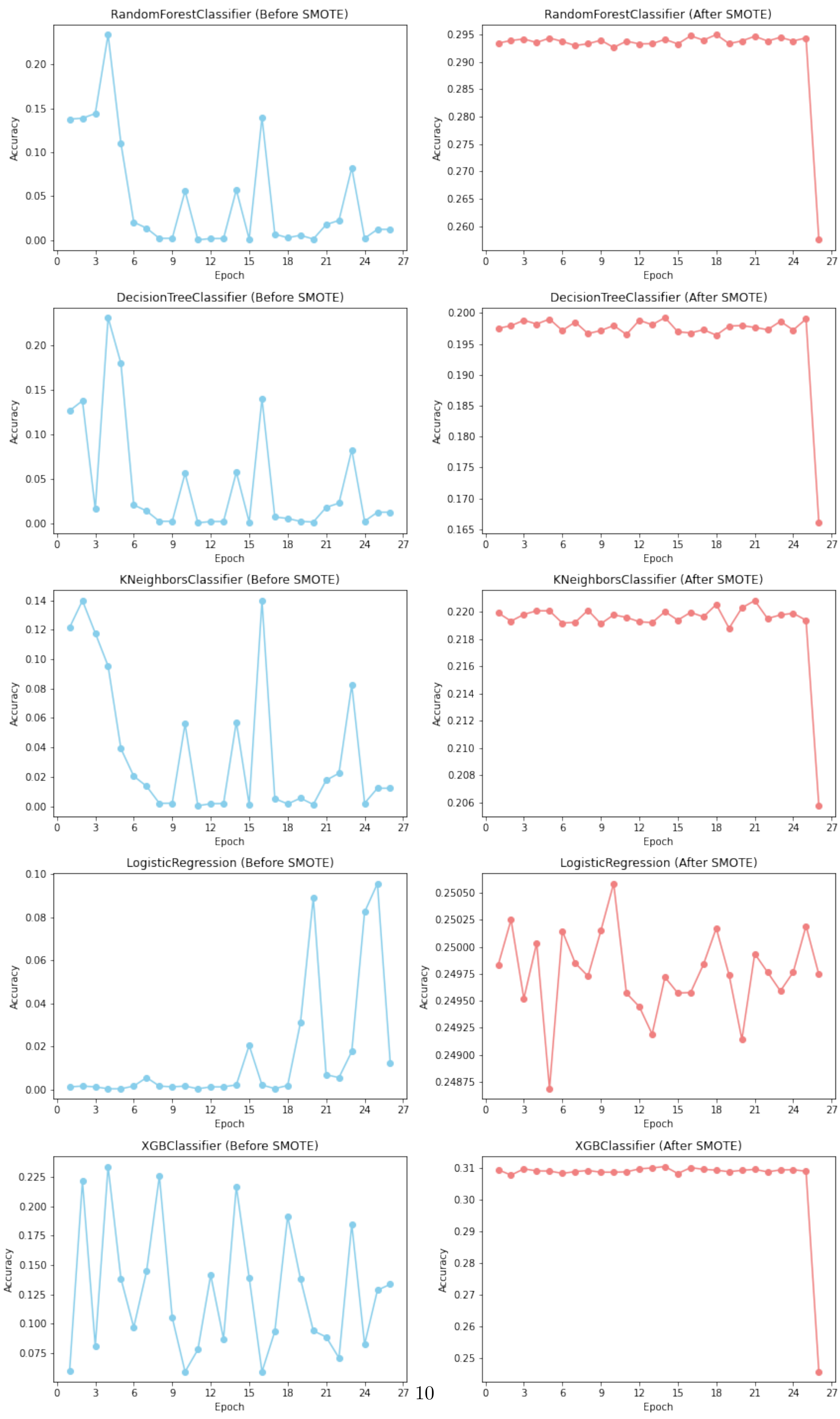


Figure 4: Accuracies of Machine Learning Estimators trained with and without SMOTE

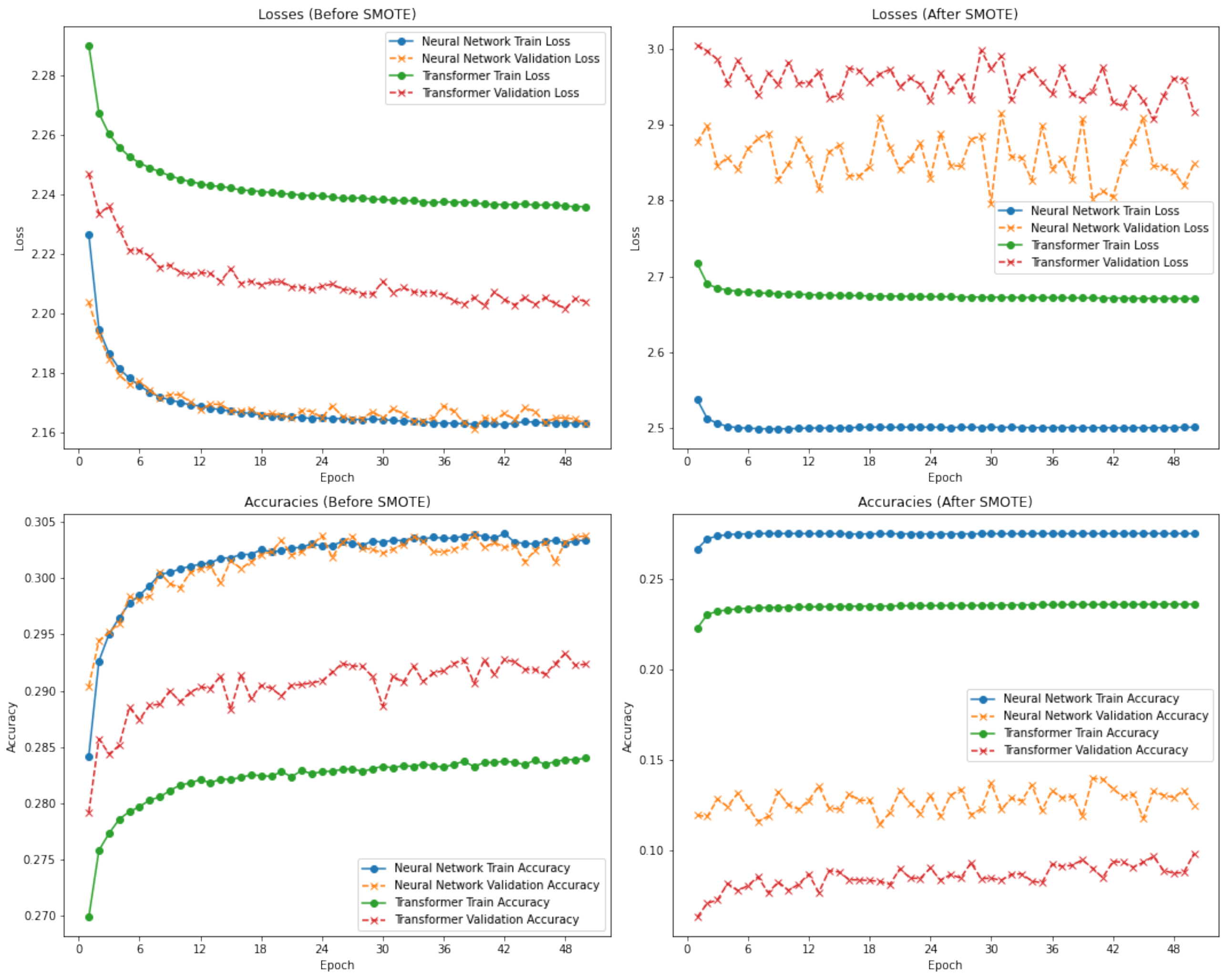


Figure 5: Accuracies of Deep Learning Models trained with and without SMOTE

larger number of trainable parameters. We also find that applying SMOTE to rebalance crime classification data decreases model prediction accuracy in half.

## 7 Future Work

This study focused on utilizing machine learning models for multiclass crime classification. There's many directions a future study from our work could expand upon such as applying our features and models to other cities in the United States as well as across the world and comparing performances. A future study could also reformat the tabular data into a windowed time series problem. Additionally, regression-based models can be used to forecast future locations of crimes as well as what times they will occur for a given day. Additional features to use could involve weather, more economic data, and pollution data. For implementing machine learning models, k-fold cross validation could also be used to see if the models overfit to the crime data.

## 8 Limitations and Ethics Statement

Crime is inherently very difficult to predict for diverse urban areas, and thus our models should not be used for legal decision making by the police. We have open sourced all our code and have made our model weights available online in order to allow other scientists and policy makers to help make data-driven decisions for the betterment of their community. Machine learning models are known to exhibit bias, and thus our work should not be used for making discriminatory policies.

## 9 Acknowledgements

We would like to acknowledge opendataphilly.org for providing us data for the majority of our features and for open sourcing datasets of Philadelphia for scientific use. We also would like to acknowledge the Federal Reserve Bank of St. Louis for providing us with the data for the unemployment rate feature we used for our model.

## References

[1] Robert J Sampson, Stephen W Raudenbush, and Felton Earls. "Neighborhoods and violent crime: A multilevel study of collective efficacy". In: *science* 277.5328 (1997), pp. 918–924.

[2] Leah H Schinasi and Ghassan B Hamra. "A time series analysis of associations between daily temperature and crime events in Philadelphia, Pennsylvania". In: *Journal of urban health* 94 (2017), pp. 892–900.

[3] Suhong Kim et al. "Crime analysis through machine learning". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE. 2018, pp. 415–420.

[4] Sherry Towers et al. "Factors influencing temporal patterns in crime in a large American city: A predictive analytics perspective". In: *PLoS one* 13.10 (2018), e0205151.

[5] Yigit Alparslan et al. "Perfecting the Crime Machine". In: *arXiv preprint arXiv:2001.09764* (2020).

[6] Ourania Kounadi et al. "A systematic review on spatial crime forecasting". In: *Crime science* 9 (2020), pp. 1–22.

[7] Muzammil Khan, Azmat Ali, and Yasser Alharbi. "Predicting and preventing crime: a crime prediction model using san francisco crime data by classification techniques". In: *Complexity* 2022.1 (2022), p. 4830411.

[8] Yingjie Du and Ning Ding. "A Systematic Review of Multi-Scale Spatio-Temporal Crime Prediction Methods". In: *ISPRS International Journal of Geo-Information* 12.6 (2023), p. 209.