Urban Crime Data Classification

Michael Boon, Xin Jin, Aleksei Luchinsky, Dan Mrachko

April 28, 2023

Abstract

This paper is the final project report for the CS5630/4630 course. In our work, the statistics of San Francisco crimes during the years 2003-2013 are analyzed. After downloading, cleaning, and wrangling the data, we have tried to predict the type of crime (i.e., violent vs. non-violent) using various ML models. Our analysis shows that the Random Forest method gives the best prediction accuracy.

1 Introduction

An emerging area of research focuses on criminal classification methods for crime forecasting. By studying and comparing well-known prediction algorithms, our main objective is to predict crime categories based on a given collection of geographical and time-based factors, such as the district and address of the city where the crime was committed, the latitude and longitude of the crime position, dates, and the day of the week on which the crime occurred. For security organizations to allocate resources effectively, it is both fascinating and important to study and understand the distribution of various crime types throughout the city based on their occurrence times and locations. Crime statistics can help officers share observations and enhance early warning systems to keep residents safe in their communities.

The objective of this report is to present a predictive model for forecasting crime in San Francisco based on past criminal records. The model is built using multiple techniques and evaluated through accuracy, precision, and recall metrics. The data is subject to descriptive analysis, and the statistical distribution of crime over space and time is visualized to uncover potential patterns. Features are extracted from the original dataset, and classification is performed using Naive Bayes, Random Forest, Decision Tree, SVM Linear Kernel, Logistic Regression, Neural Network, and Gradient Boosting Decision Tree techniques.

In Section 2, we will give some information on the data set and describe how it was cleaned and prepared for future usage. Section 3 provides a brief description of the methods used in our analysis and results. Last section is reserved for final discussion and conclusion.

2 Data Description and Cleaning

In this project we will consider data sets that describe crime statistics in such cities as San-Francisco. You can find this data set on Kaggle [1]. It contains information about 878049 cases and for each case the following features are stored:

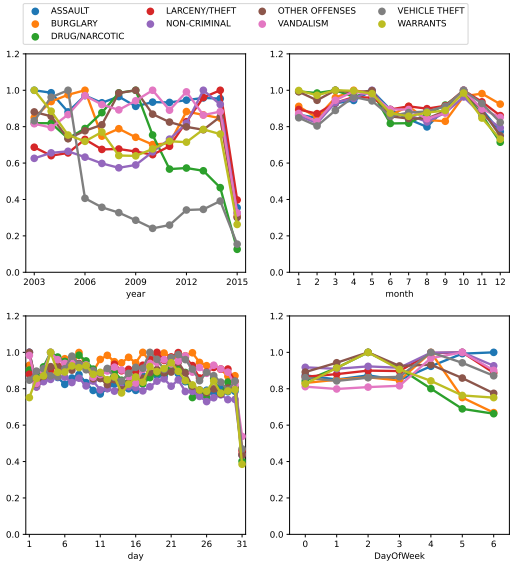

Figure 1: Distribution of 9 most popular crime categories by year, month, day, and day of week, scaled by the maximum value

- **Dates**: The date of the case, starting from Jan 2003 up to May 2015;
- **Category**: crime category, such as ROBBERY, ASSAULT, or even KIDNAPPING. In total there are 39 such categories;
- **Descript**, **Resolution**: brief description and resolution of the case;
- **DayOfWeek**: string representation of the weekday (like "Monday", "Wednesday", etc.);
- **PdDistrict**: one of 10 police department districts;
- **Address**: adress of the case;
- **X**, **Y**: geographical coordinates.

Let us talk in little more details about each of these fields and describe the data cleaning and preparation process.

It turns out, that **Descript** and **Resolution** fields do not store any important information for our problem, so they will be dropped out from the analysis.

Dates field contains full information about the date of the crime. For our usage we decided to split it into **year**, **month**, and **day**, which are now separate quantitate variables. After that **Dates** field itself was dropped out from the dataset. The **DayOfWeek** field was transformed from categorical to numerical form. In figure 1 you can see the distributions of 9 most popular crime categories by these variables. To make the comparison more simple, we have normalized these distributions by their mean values.

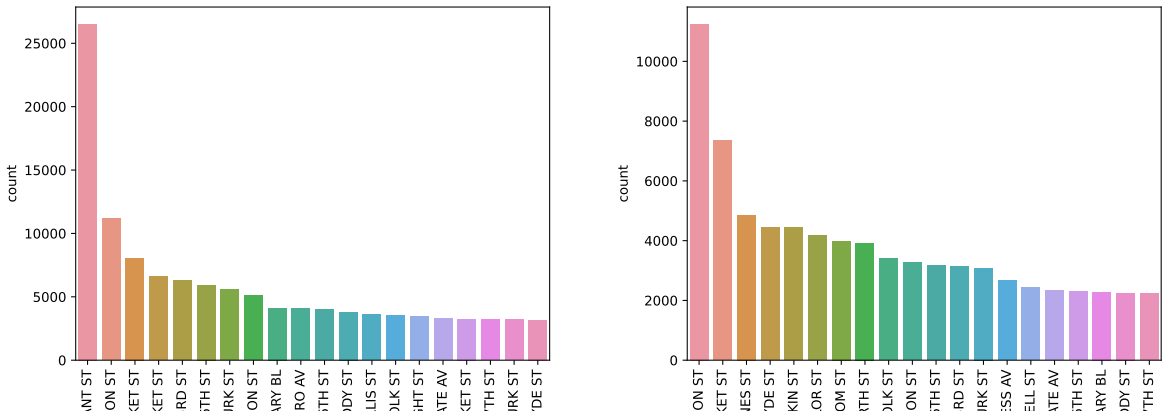

Figure 2: Streets histograms

For the majority of cases the **Address** field has the form [street1] /[street2], showing the intersection of streets, at which an incident has occurred. In total, there are 23228 unique adresses in the data set, while number of unique streets are 12552 and 1695 respectively. For this reason we have decided to split the **Address** field into **street1** and **street2**. Moreover, only a limited number of streets occur often in the data set (see histograms, shown in Fig. 2), so we have left only 200 street of each type (the others are denoted as "other").

Talking about the geographical coordinates **X** and **Y**, it turns out that the majority of the cases have $-122.5 < X < -122.25$, $37.7 < Y < 40$. There are, however, 67 cases with $Y = 90.0$ and unusually large **X**. It seems, that these points are just some outlier error, so we have dropped them from the analysis. Using all other cases we can plot a map of the city, which is shown in figure 3.

Let us now talk about the response variable. Following article [2] we divided all 39 crime categories into violent and non-violent groups, trying to match total numbers of cases for each crime type to number, that can be extracted from presented in [2] confusion matrices. Finally we stopped on the following choice:

- **Violent categories** WARRANTS, VEHICLE THEFT, VANDALISM, ROBBERY, ASSAULT, WEAPON LAWS, BURGLARY, SUSPICIOUS OCC, FORGERY, NARCOTIC, TRESPASS, MISSING PERSON, FRAUD, KIDNAPPING, SEX OFFENSES FORCIBLE, DISORDERLY CONDUCT, ARSON, SEX OFFENSES NON FORCIBLE

- **Non-violent**: THER OFFENSES, LARCENY/THEFT, NON-CRIMINAL, DRUNKENNESS, STOLEN PROPERTY, SECONDARY CODES, RUNAWAY, DRIVING UNDER THE INFLUENCE, PROSTITUTION, FAMILY OFFENSES, LIQUOR LAWS, BRIBERY, EMBEZZLEMENT, SUICIDE, LOITERING, EXTORTION, GAMBLING, BAD CHECKS, TREA, RECOVERED VEHICLE, PORNOGRAPHY/OBSCENE MAT.

With this assignment there are 433408 cases of violent crimes and 444574 cases of non-violent crimes in total, the data set is balanced. It should be noted, however, that the choice we

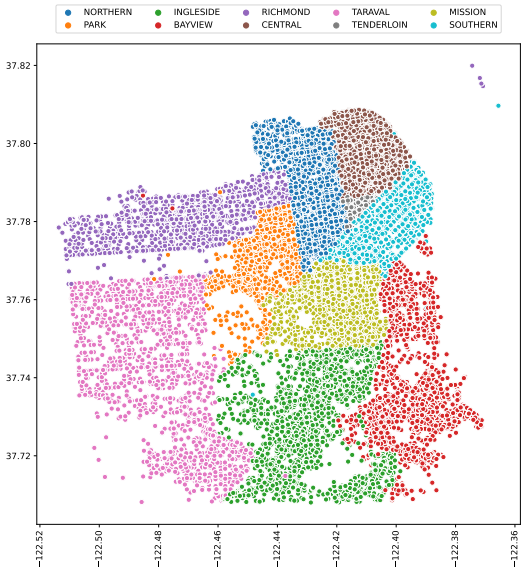

Figure 3: Map

made is not unique. We asked the corresponding author of the paper [2] for details, but have not received any reply.

To summarize, in our report we are using three versions of the dataset:

- The smallest one, including DayOfWeek, X, Y, year, month, day, hour, and PdDistrict as numerical variable. This choice of the predictors seems to be most close to one used in [2]. In the following it will be referred to as [paper]
- The same set of predictors, but PdDistrict is considered as categorical variable. This correction adds 9 more dummy variables to the dta set, so it is expected to be more informative. In the following it will be named [catPdDistrict]
- In addition to predictors from the previous data set we also included dummy variables, generated from str1 and str2 categorical fields. This adds us 400 more features and gives even more predictive power to the model. This set will be referred to as [catPdDistrict+strs].

Using each if these three versions of the data set we will try to predict the type of the crime case, determining, whether it will be violent, or not. Figure 4 shows aggregative distribution of violent and non-violent crimes by different variables.

3 Description of the Classification Methods

3.1 Naive Bayes

First of the methods, that we used for analysis of San-Francisco data set, is the Naive Bayes algorithm (see [3] for details). Within this approach the decision of the class assignment

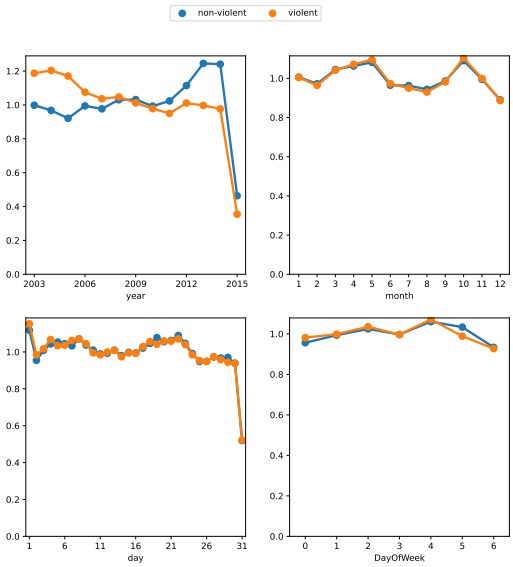

Figure 4: Comparing violent and non-violent crimes

is done by maximizing the posteriorly probability for a point with given set of predictor variables $\mathbf{x} = \{x_1, \dots x_i\}$ to be assigned to a specific class C_k:

$$k = \arg\max_k P\left(C_k|\mathbf{x}\right) \tag{1}$$

The conditional probability in the equation above can be calculated using the Bayes relation

$$P\left(C_k|\mathbf{x}\right) = \frac{P\left(C_k\mathbf{x}\right)}{P\left(\mathbf{x}\right)} \sim P\left(C_k\mathbf{x}\right), \tag{2}$$

where the last transformation is legal since the omitted denominator does not depend on the class number k. Further simplification is possible if we assume the conditional of the predictors:

$$\begin{aligned} P\left(C_k|x_1 \dots x_i\right) \sim & P\left(C_k x_1 \dots x_i\right) = P\left(x_1|C_k x_2 \dots x_i\right) P\left(C_k x_2 \dots x_i\right) = \\ & P\left(x_1|C_k\right) P\left(C_k x_2 \dots x_i\right) = \dots = P\left(C_k\right) \prod_{i=1}^{n} P\left(x_i|C_k\right) \end{aligned} \tag{3}$$

All the probabilities on right hand size of the above equation can be calculated from the training part of the data set and then used to make predictions for the test part.

From presented above description it is clear, that this type of the classification algorithm does not require any tuning of the hyper parameters. This makes it very convenient for quick estimates, but diminishes the prediction accuracy.

Let us consider first the results for the simplest data set, which is most close to one, used in paper [2]. The confusion matrix that we have received in our analysis is shown in figure 5. The values of different accuracy metrics for training and test data sets in this case are

$$\text{accuracy} : 0.573, \quad \text{sensitivity} : 0.828, \quad \text{specificity} : 0.312, \quad \text{kappa} : 0.141$$

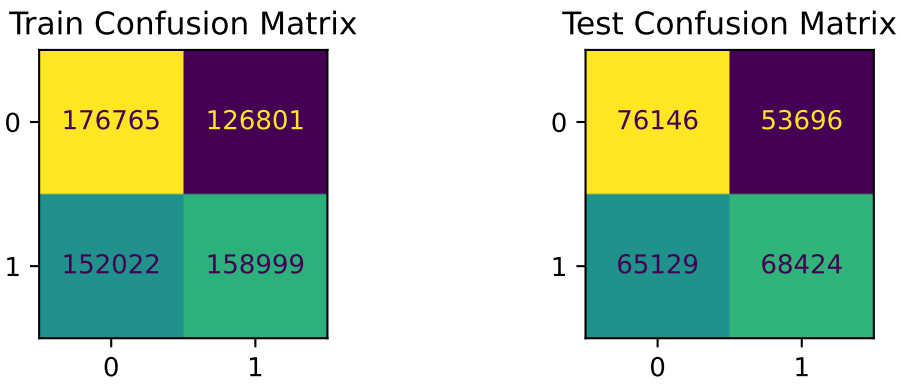

Figure 5: Confusion Matrix for [paper] Training and Test Sets

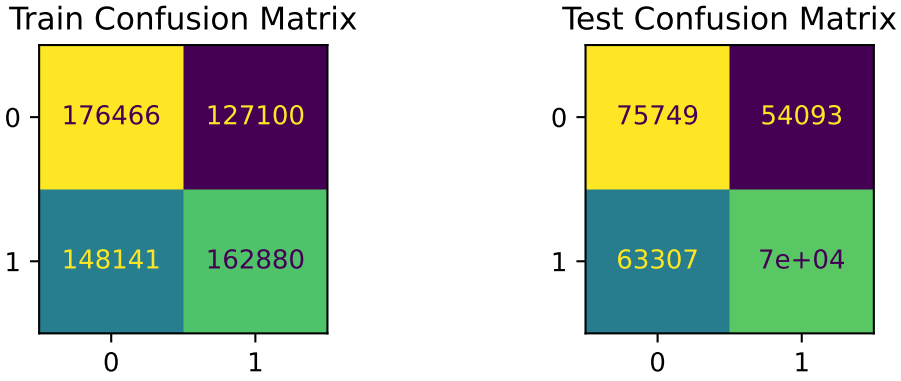

Figure 6: Confusion Matrix for [catPd] Training and Test Sets

and

accuracy : 0.549, sensitivity : 0.512, specificity : 0.586, kappa : 0.099

respectively. As you can see, prediction accuracy on the test set is smaller, than on the train set, which is expected, but they are comparable with each other. It turns out also, that our results are significantly smaller, that published in [2]. According to the confusion matrices, presented in this paper, accuracies for training and test sets are equal to 0.658 and 0.643.

One of the ways to increase the prediction accuracy is to consider a more complicated data set. In figure 6 you can see our results for confusion matrix of the NB model, trained and tested on the [catPdDistrict] data set. Values of different accuracy metrics for this choice are

accuracy : 0.552, sensitivity : 0.524, specificity : 0.581, kappa : 0.105

and

accuracy : 0.554, sensitivity : 0.526, specificity : 0.583, kappa : 0.109

for train and test subsets respectively.

For our final choice we have also included dummy variables, generated from categorical fields **str1** and **str2**. This increases total number of predictors by approximately 400 and

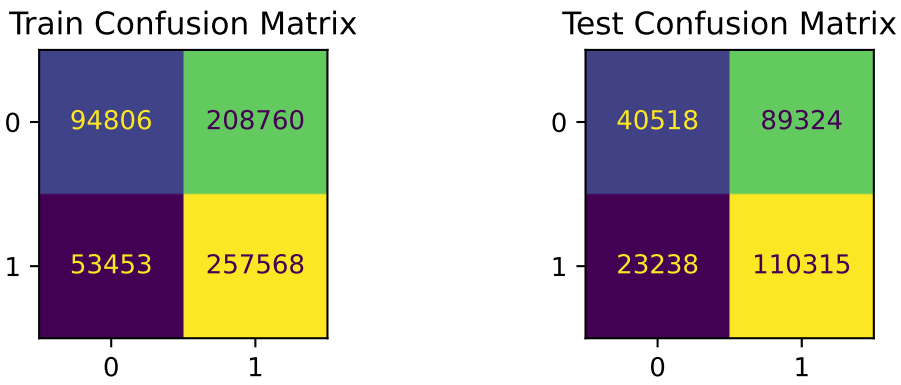

Figure 7: Confusion Matrix for [strs] Training and Test Sets

makes the model even more complicated. In figure 7 you can see resulting confusion matrices and accuracy metrics are

$$\text{accuracy} : 0.573, \quad \text{sensitivity} : 0.828, \quad \text{specificity} : 0.312, \quad \text{kappa} : 0.141$$

and

$$\text{accuracy} : 0.573, \quad \text{sensitivity} : 0.826, \quad \text{specificity} : 0.312, \quad \text{kappa} : 0.139.$$

It is clear, that even with the last, most complicated and informative dataset, our preformance metrics are not as good as in [2]. One of the reasons for this differense is that we have selected wrong seperartion of the crime categories into violent and non-violent groups. Unfortunatelly, the authors of the original article did not give us their version of the assignment.

3.2 Decision Tree

In addition to Naive Bayes, we applied decision trees. Decision trees are less complex than ensemble machine learning methods such as random forest, take less time to train, and are known for being easy to describe and explain to persons outside the technical domain. Classification trees are represented as a series of choices, with thresholds of choice determined at training. Each split or branch in the tree seeks to partition the data in such a way that reduces the impurity of each grouping of information, for example splitting a branch so all samples with value x¡6 go to one branch, and all samples with value x¿6 go to another branch. The training algorithm will finish when the final impurity of each terminal node, or group at the end of a branch, is within an allowable tolerance. The goal is for each terminal node of the tree to contain an exclusive group of data that can be ascribed to one of the output labels [4]. To implement decision trees, our team used the Sci-Kit Learn implementation of decision trees in python.

There are multiple criteria available to use as a measure of impurity. Entropy is the default criteria for the sci-kit learn implementation of decision trees in python. For m samples and p_k being the proportion of records belonging to the k^{th} class, entropy is defined

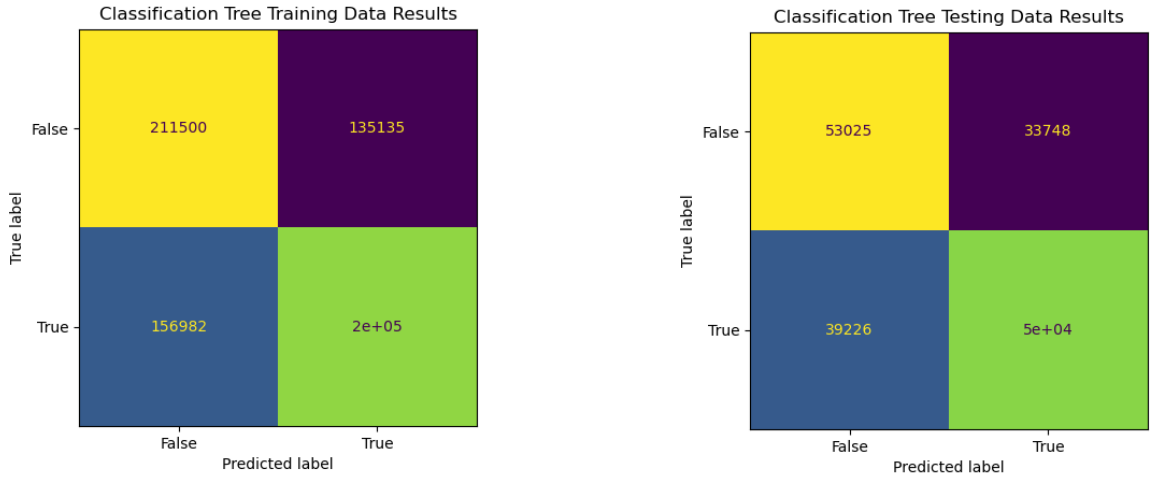

Figure 8: Classification Tree: Confusion Matrix for prediction on Training and Testing Sets

as:

$$entropy = -\sum_{k=1}^{m} p_k \log_2(p_k) \tag{4}$$

When all records belong to the same class, $entropy = 0$, the desired value. When all classes are equally represented, for example $p_k = 0.5$ for a two-class problem, $entropy = log_2(m)$.

Another popular criteria for decision trees is Gini Impurity index. The Gini Impurity index $I(A)$ is defined as:

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2 \tag{5}$$

Similar to entropy, when all records belong to the same class, $I(A) = 0$. When all classes are equally represented, $I(A) = 1 - 1/m$.

Additional parameters that can be used to tune a decision tree include the maximum depth of the tree, the minimum number of samples required to split the tree, and the maximum number of data features to use for determining splits. Hyper-parameter tuning was performed using a cross-validation approach, and the best parameters were determined to be:

- **Criteria**: Gini Impurity Index
- **Max Tree Depth**: 5
- **Max Features**: All features
- **Min Samples to Split**: 2

To train the model, the data was randomly split into 80% training and 20% testing. The results are depicted as confusion matrices in figure 8. The training set prediction performance metrics are:

accuracy : 0.584, sensitivity : 0.559, specificity : 0.559, kappa : 0.169

The testing set prediction performance metrics are:

accuracy : 0.584, sensitivity : 0.558, specificity : 0.558, kappa : 0.169

The final tree is shown in figure 9. It is interesting to see that the initial split was on the secondary street classification of "other". This means this was the strongest predictor of violent or non-violent crime for this model with this data set. When str2_other is 1, this is more likely to predict non-violent crime than violent crime. The performance metrics for the testing set compared to the training set show there is no overfitting occurring, however it cannot be ignored that the overall accuracy is very low at less than 60%.

3.3 Support Vector Machine

Support vector machines are similar to decision trees because they attempt to classify data samples by developing a rule for partitioning incoming samples. A tree does this with a set of branching decisions or rules based on impurity measurements in groups of samples. Support vector machines do this by mapping the data to a kernel function, then solving for coefficients to separate or partition the data as best as possible. This is done by solving for a nxn matrix of support vectors between the samples in the data, where n is the number of samples, then constructing a hyperplane in m dimensions corresponding to m variables to separate the samples based on these vectors [5]. The value of C in the algorithm is used to determine how much impact the support vectors have on the definition of the hyperplane. Output classes can be weighted, which has the possibility to assist with predicting the important class or dealing with an imbalanced dataset. However since the algorithm must compute the nxn matrix many times, this method can be expensive to implement for large datasets. Giving the algorithm twice as much data results in four times as much computation time [5].

In this project we employed a Linear Kernel Support Vector Machine for its simplicity and speed when training with student hardware. After performing model parameter tuning with cross-validation approach, the optimal parameters were determined to be:

- **C**: 0.1
- **Class Weight**: None

Just as with decision tree model, the Linear Kernel SVM model was trained using an 80% training set, and prediction accuracy was tested with a 20% testing set. The results of this model are depicted in figure 10. The prediction performance metrics for the training set are:

accuracy : 0.595, sensitivity : 0.680, specificity : 0.680, kappa : 0.188

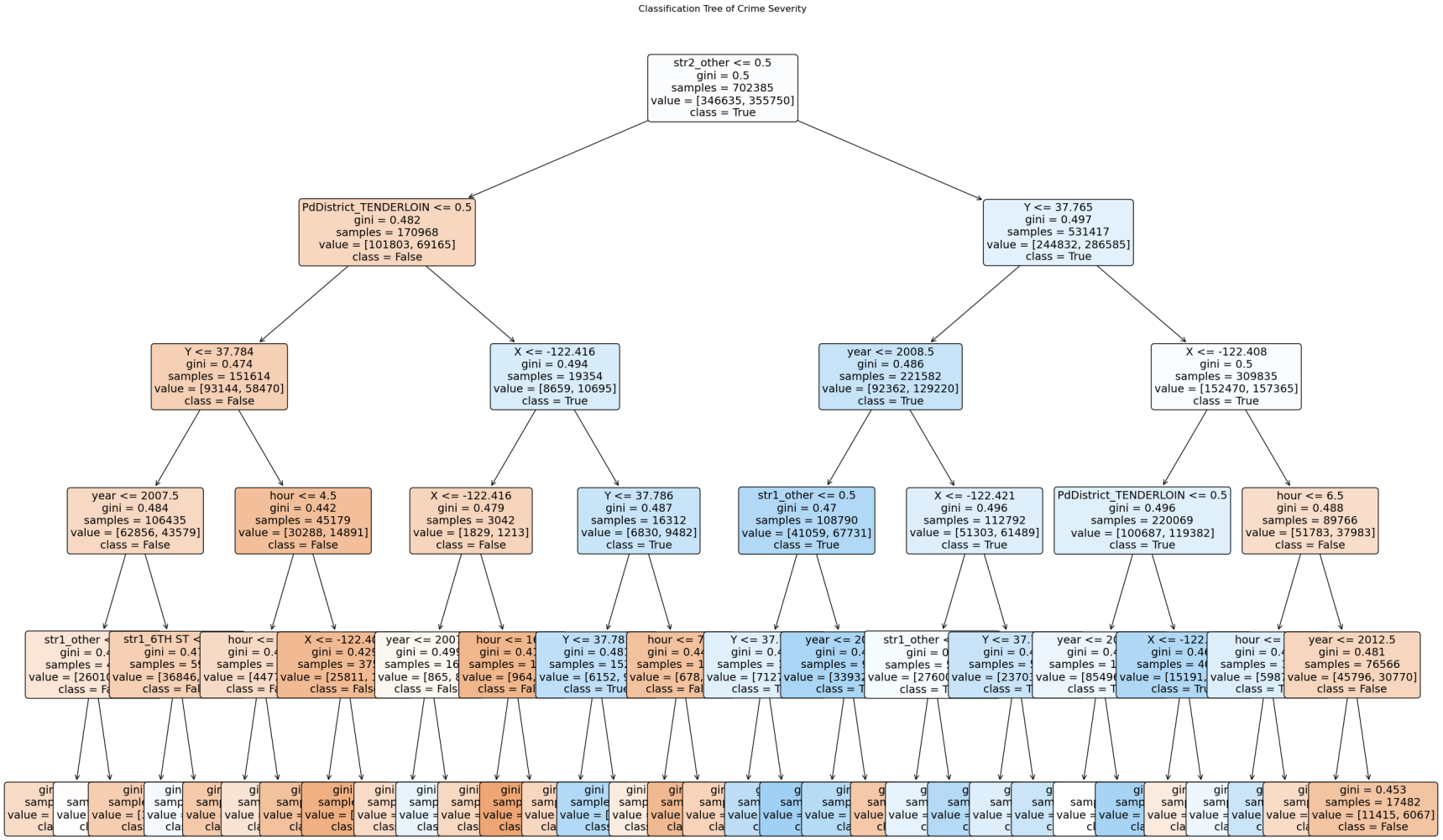

Figure 9: Classification Tree for Prediction of Violent Crime

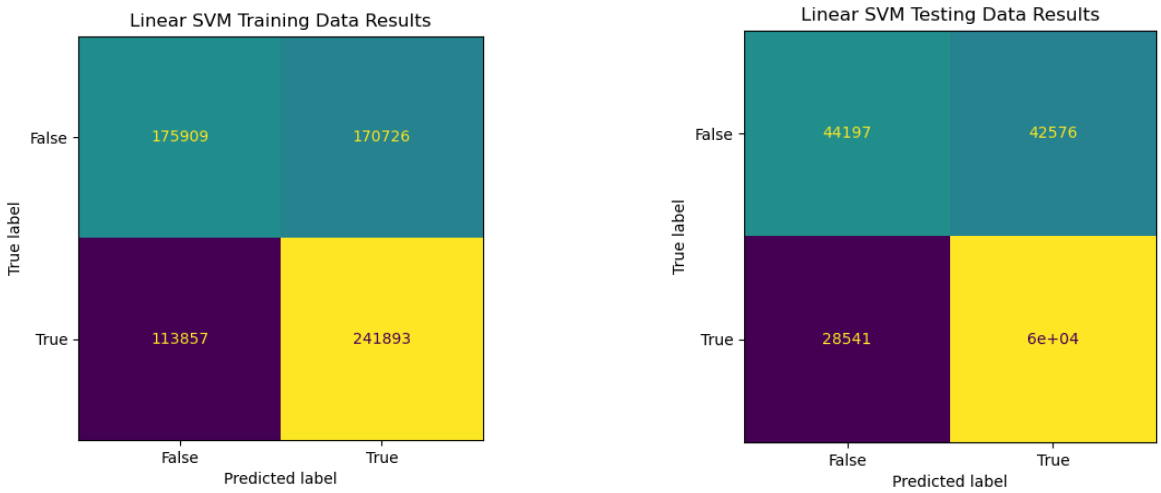

Figure 10: Linear Kernel SVM: Confusion Matrix for prediction on Training and Testing Sets

The prediction performance metrics for the testing set are:

$$\text{accuracy} : 0.595, \quad \text{sensitivity} : 0.679, \quad \text{specificity} : 0.679, \quad \text{kappa} : 0.188$$

Similar to decision tree, there was little to no bias towards the training set, the model did not over-fit to the training samples. However the accuracy is quite poor, less than 60%.

3.4 Random Forest

The next classification method to discuss here is random forest. As it follows from its title, random forest is collection of (decision) trees, where each tree uses some random subset of the original data set's points and prediction features. The final decision for class assignment is selected fro decisions of all the trees using by using simple majority votes. Details of the method can be found in [6]. Some of the hyper-parameters, that can be determined in the tuning step are number of trees in the random forest, maximal depth of each tree and minimal number of splittings. Our search has shown, that the optimal parameters are

$$\text{max\_depth} = 100, \text{min\_salpes\_split} = 50$$

It is clear, that random forest is an expansion of the decision tree method, so it is more complicated. On the other hand, we could expect more precise predictions from it.

With these parameters random forest model, trained on the simplest data set has shown results, shown in figure 11. This matrix correspond to

$$\text{accuracy} : 0.746, \quad \text{sensitivity} : 0.779, \quad \text{specificity} : 0.712, \quad \text{kappa} : 0.491$$

on training data set and

$$\text{accuracy} : 0.603, \quad \text{sensitivity} : 0.643, \quad \text{specificity} : 0.562, \quad \text{kappa} : 0.205$$

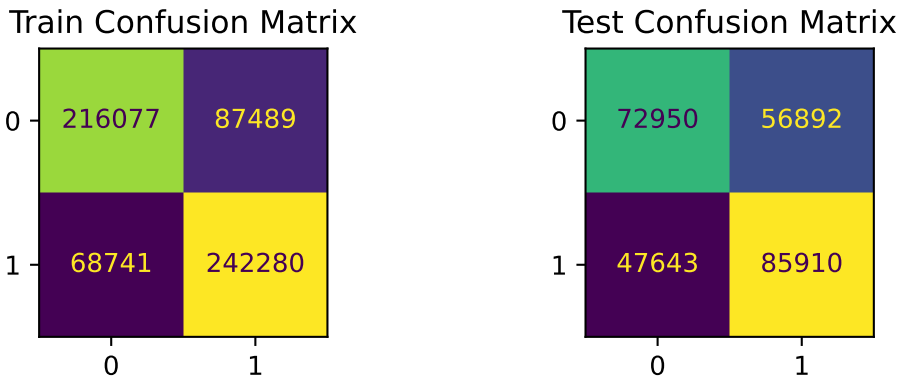

Figure 11: Random Forest: confusion Matrix for [paper] Training and Test Sets

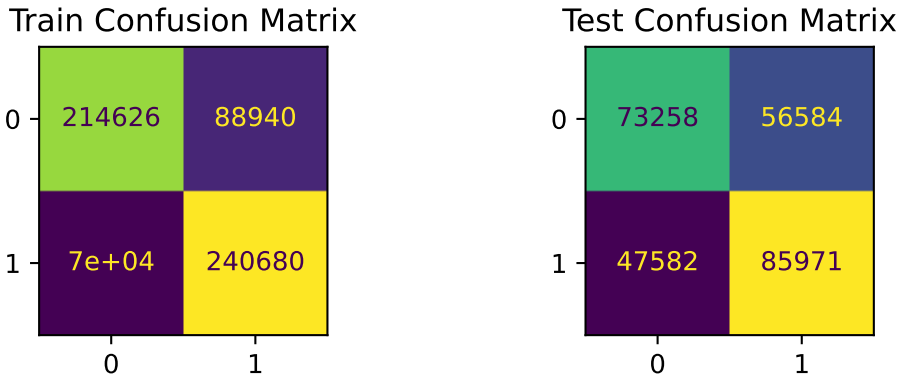

Figure 12: Random Forest: confusion Matrix for [paper] Training and Test Sets

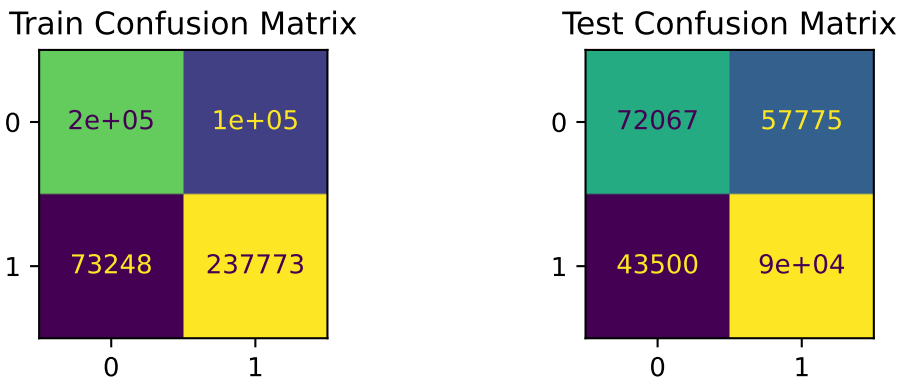

Figure 13: Random Forest: confusion Matrix for [catPdDistrict_strs] Training and Test Sets

on test subset.

In the case of more complicated set of predictors, [catPdDistrict] the confusion matrix is shown in figure 12, and precisions metrics are

$$\text{accuracy} : 0.741, \quad \text{sensitivity} : 0.774, \quad \text{specificity} : 0.707, \quad \text{kappa} : 0.481$$

and

$$\text{accuracy} : 0.605, \quad \text{sensitivity} : 0.644, \quad \text{specificity} : 0.564, \quad \text{kappa} : 0.208$$

For [catPdDistrict_strs] data set you can find the confusion matrix in figure 13, while the results for various precision metrics are

$$\text{accuracy} : 0.711, \quad \text{sensitivity} : 0.764, \quad \text{specificity} : 0.656, \quad \text{kappa} : 0.421$$

and

$$\text{accuracy} : 0.616, \quad \text{sensitivity} : 0.674, \quad \text{specificity} : 0.555, \quad \text{kappa} : 0.230$$

3.5 Logistic Regression

Assumptions: The two classes are coded as 0/1. Thus, the response $Y \sim$ Bernoulli (p), where $p = \mathbf{P}(Y = 1) = \mathbf{E}(Y)$. Let p depend on the covariates X, then we have $p(x) = \mathbf{P}(Y = 1|X = x) = \mathbf{E}(Y|X = x)$ [7].

Use the logit link, then $\operatorname{logit} p(x) = x^\top \beta$, where $\operatorname{logit} p(x) = \log\left(p(x)/[1 - p(x)]\right)$, i.e.,

$$\log\left\{\frac{p(x)}{1-p(x)}\right\} = x^\top \beta.$$

As p increases in $(0, 1)$, odds ratio $p(x)/[1-p(x)]$ increases in $(0, \infty)$. The logistic regression function is given by

$$p(x) = \frac{\exp\{x^\top \beta\}}{1 + \exp\{x^\top \beta\}}.$$

A logistic regression model has an S-shaped curve and predicted response Y lies between 0 and 1. We are supposed to pick a cutoff value between 0 and 1 to determine the categories of the response and 0.5 is the most commonly used.

We perform the hyper-parameter tuning procedure for two parameters by using the cross validation method: solvers, which specifies the algorithm to use in the optimization problem, and c_values, which refers to the inverse of regularization strength; smaller values specify stronger regularization. We set up the penalty to be the L2 norm.

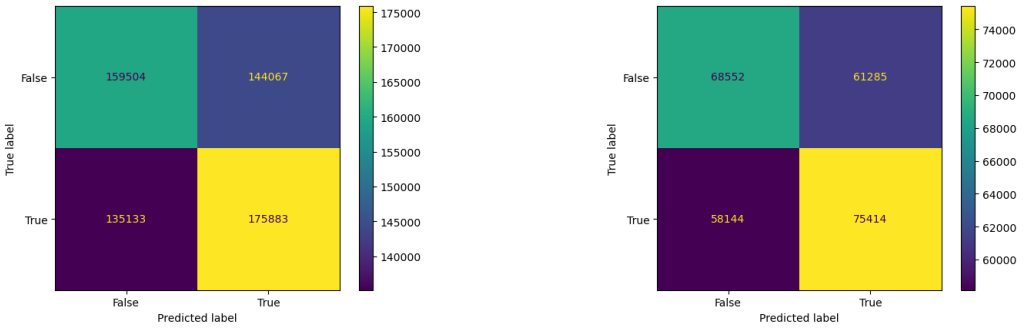

Figure 14: Confusion Matrix for [paper] Training and Test Sets

For the dataset [paper], the optimal c_values is 100 and solver is 'newton-cg'. The performance of the model on classifying the violent and non-violent crimes are similar for the training set and test set and the model performs poorly with an accuracy of 0.547, see Figure 14.

For the dataset [catPdDistrict], the optimal solver is 'liblinear'. This is a little surprising because 'liblinear' is a good choice for small datasets. This model performs slightly better than the first one with an accuracy of 0.561, see Figure 15, sensitivity of 0.585, specificity of 0.536 and Kappa of 0.121.

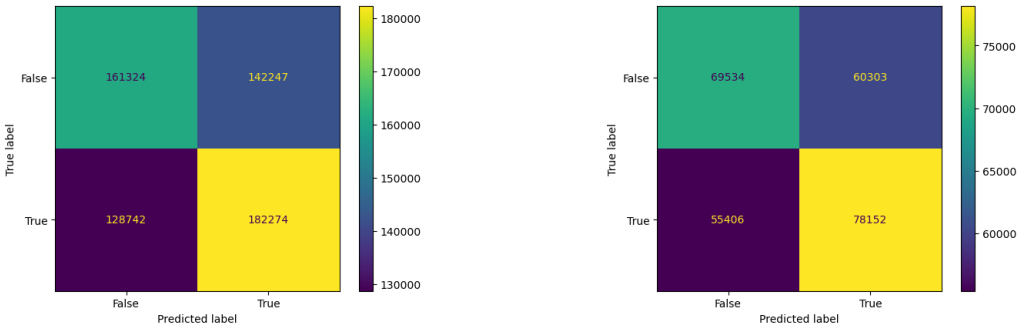

Figure 15: Confusion Matrix for [catPdDistrict] Training and Test Sets

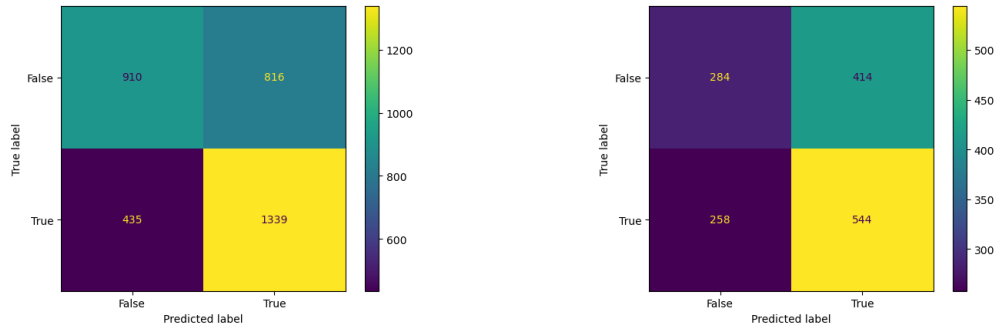

Figure 16: Confusion Matrix for [catPdDistrict+strs] Training and Test Sets

For the dataset [catPdDistrict+strs], take the expensive computational cost into the consideration, we randomly select 5000 observations from the whole dataset to do the analysis. From Figure 16, this model performs the best among the three models with an accuracy of 0.573.

3.6 Neural Network

Neural networks are algorithms that can establish correlations between datasets, similar to the way the human brain works. By processing information through layers of interconnected nodes, neural networks can adapt to changes in input data and produce the best possible results without needing to change the output criteria.

In this technique, mathematical functions known as neurons collect and organize data according to a specific architecture. Each node in the network is a perceptron, which works like multiple linear regression to process data and make decisions [8].

One of the most significant advantages of neural networks is their ability to model nonlinear processes. This makes them highly effective for problem-solving tasks, such as regression, pattern recognition, clustering, and anomaly detection. With these capabilities, neural networks have become an essential tool in fields such as machine learning and artificial intelligence.

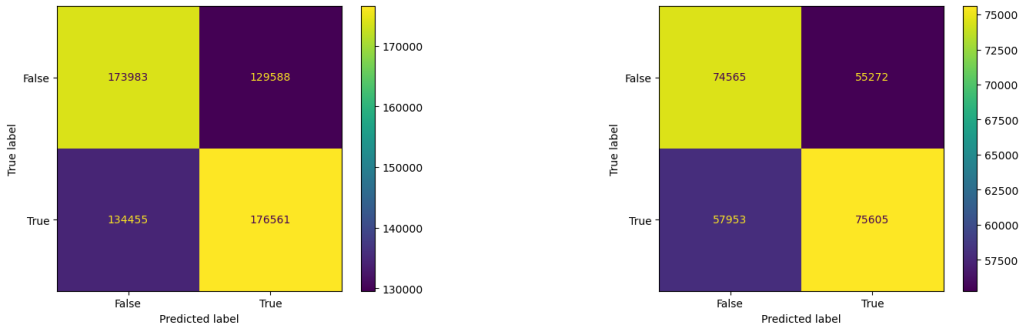

Figure 17: Confusion Matrix for [paper] Training and Test Sets

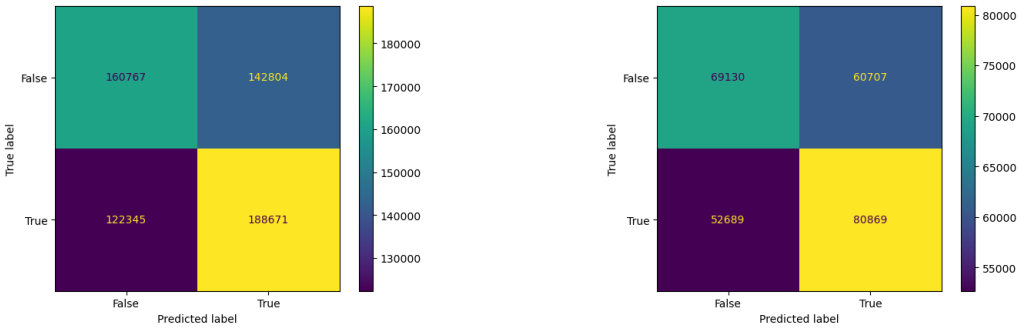

Figure 18: Confusion Matrix for [catPdDistrict] Training and Test Sets

To prepare the data for the neural network model, we use the Min-Max Normalization to scale the values between 0 and 1. The model consists of two hidden layers, with 64 and 32 neurons, respectively. The last layer produces a score that indicates whether the response belongs to one of two categories.

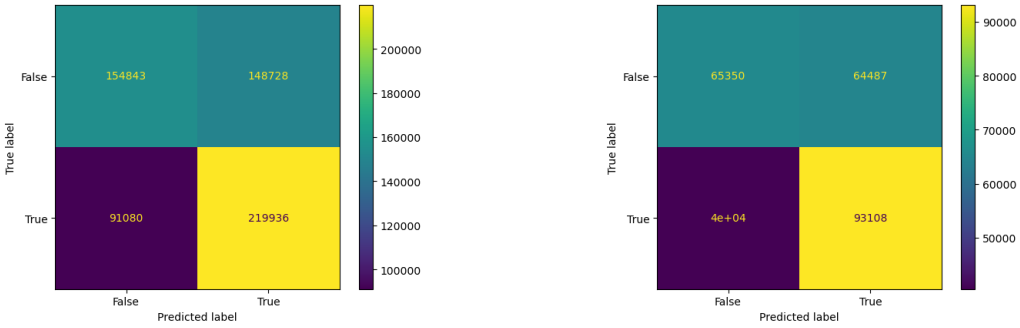

Figure 19: Confusion Matrix for [catPdDistrict+strs] Training and Test Sets

However, the first and second models perform poorly on both the training and test sets, despite their similar accuracy. The third model, which incorporates more information about the crime location, performs slightly better with an accuracy of 0.602. This suggests that the model benefits from additional data about the location of the crimes.

3.7 Gradient Boosting Decision Trees

Gradient Boosting is a technique that works by combining a large number of weak predictor models in order to "learn" from their mistakes and create a much stronger predictive model. In this case, decision trees are being used as the "weak" model.

The first step in implementing this model is deciding on a few parameters. The number of trees is set to 100, meaning the boosting algorithm will be learning from a pool of 100 decision trees. Learning Rate, which is a gauge of how quickly the model learns from the data set, is set to 0.1 which is the default value. A higher value for learning rate would make the model run quicker, however, it would jeopardize accuracy in the long run. The final parameter is max depth, which is the depth of the decision trees being generated. After

running a test, it was determined that the decision trees were able to achieve their highest accuracy score when max depth was set to 14.

The first model is built on the simplest version of the data set using no address information. The accuracy for the training set is 0.794, while the accuracy for the testing set is 0.605

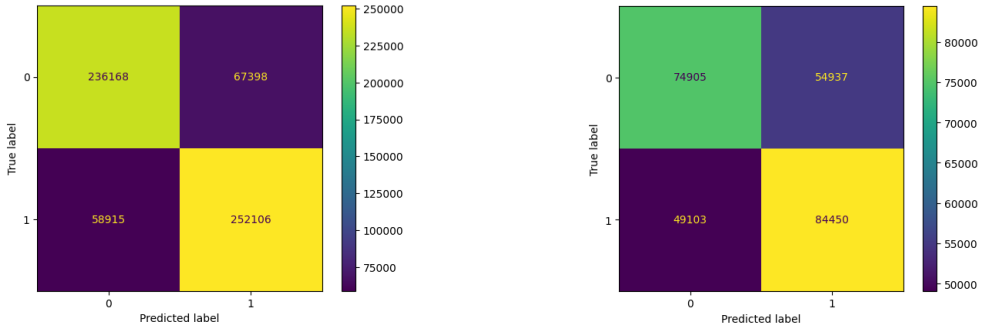

Figure 20: Confusion Matrix for [paper] Training and Test Sets

The second model uses the data set that includes PdDistrict. The confusion matrices for this model are seen below. On the training set, accuracy was 0.800, while on the testing set accuracy was 0.605. This shows no noticeable improvement compared to the original data set.

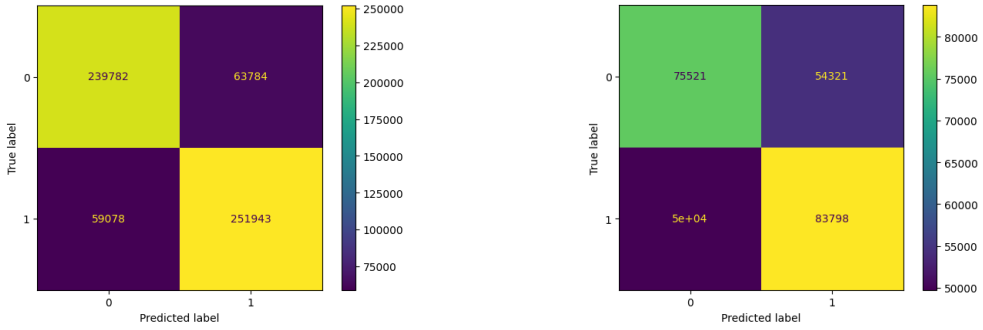

Figure 21: Confusion Matrix for [catPdDistrict] Training and Test Sets

The final model used the most complex data set, now including str1 and str2 as well. On the training set, accuracy was 0.736, while on the testing set accuracy was 0.615, which is a very slight improvement.

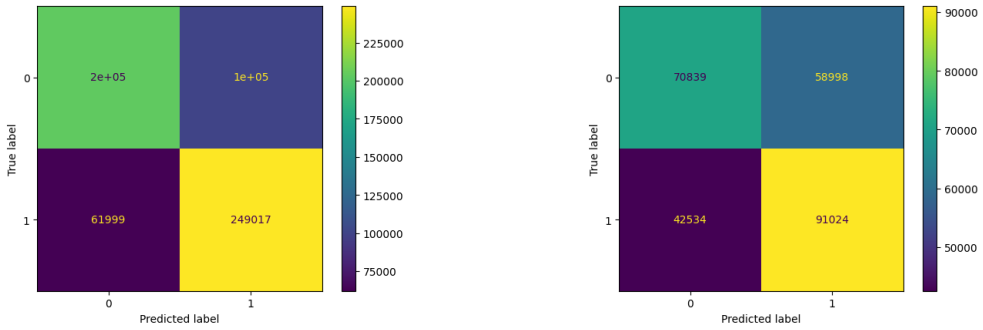

Figure 22: Confusion Matrix for [catPdDistrict+strs] Training and Test Sets

This model proved to be one of the more accurate models, and while the measurements are close, the model performed the best on the final data set which included the most amount of data about location.

4 Conclusions

In the presented project our group was considering crime statistics, collected in 2003-2013 years in San Francisco [1]. Various ML models were trained on cleaned and wrangled data set to predict the crime type, i.e. will it be violent or non-violent.

For all considered in our report models main performance metrics, measured on the test data set, are presented in table 1. As you can see, accuracies of all models are comparable with each other, but we can notice that Random Forest and Naive Bayes models give the best and the worst results respectively. This is not surprising since the Naive Bayes model seems to be the most simple approach of all used here. As for the Random Forest, this model is pretty complicated, joins the power of decision tree approach and all ensemble methods, so we could expect some good results from it. The fact that NN network model is not very accurate is a little bit strange, but we see that it has almost best Sensitivity from all models

	Accuracy	Sensitivity	Specificity	Kappa
Naive Bayes	0.573	0.826	0.312	0.139
Random Forest	0.616	0.674	0.555	0.230
Decision Tree	0.584	0.558	0.598	0.169
SVM Linear Kernel	0.595	0.678	0.678	0.188
Logistic Regression	0.573	0.668	0.469	0.137
Neural Network	0.602	0.721	0.478	0.199
Gradient Boosting	0.614	0.680	0.546	0.227

Table 1: Results

In [2] each crime was labeled as either non-violent or violent. Unfortunately these labels were not provided as previously stated, and we were left to determine them ourselves through deduction from the data and the confusion matrices presented in the paper. This means

there is a possibility our classifications were incorrect[1]. However it is also possible that these categories were tailored or arbitrarily chosen by the original authors. A more objective approach could involve labeling each crime based on how it was tried in court, as a felony or a misdemeanor. This would require additional data not contained in this dataset, but may lead to important inferences concerning society. For instance, it could be interesting to explore the relationship between geographical location and crime label as a felony or misdemeanor.

Extending the work to other datasets from different cities such as Montreal [9], Denver [10], and Philadelphia [11] could provide valuable insights into how crime types vary across different regions. However, it is important to note that each city may have unique characteristics and crime patterns that could impact the performance of the predictive model.

To extend the work to these new datasets, several steps could be taken. First, the relevant datasets for each city should be preprocessed in a similar manner as the San Francisco dataset. This may involve dealing with missing values, encoding categorical variables, and scaling numerical variables. Next, the model developed using the San Francisco dataset could be applied to these new datasets. However, it is likely that the model would need to be refined and optimized for each city, as the crime patterns and variables that contribute to crime may differ across the different regions. Once the model has been refined and optimized for each city, it should be evaluated using appropriate performance metrics to assess its accuracy and precision. This could involve using techniques such as cross-validation. Finally, the results and insights gained from applying the model to these new datasets could be discussed and compared to those obtained from the San Francisco dataset. This could help identify similarities and differences in crime patterns across the different cities, and provide valuable insights into how crime types vary across different regions.

Overall, extending the work to other datasets from different cities has the potential to provide valuable insights into the generalizability and applicability of the predictive model, and could help inform policy and decision-making in the field of crime prevention and law enforcement.

Our code can be found in GitLab repository [12].

5 Acknowledgements

The authors would like to thank Dr. Tian for the useful discussions and help provided during our work on this project.

References

[1] Kaggle. San francisco crime classification. https://www.kaggle.com/competitions/sf-crime/data.

[1] We tried to contact the authors and ask them for labelling scheme, that they were using, but did not receive any meaningful reply.

[2] Muzammil Khan, Azmat Ali, and Yasser Alharbi. Predicting and preventing crime: A crime prediction model using san francisco crime data by classification techniques. *Complexity*, 2022, 2022.

[3] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

[4] Radhwan H. A. Alsagheer, Abbas F. H. Alharan, and Ali S. A. Al-Haboobi. Popular decision tree algorithms of data mining techniques: A review. *International Journal of Computer Science and Mobile Computing*, 6:133–142, 2017.

[5] William Noble. What is a support vector machine? *Nature Biotechnology*, 24:1565–1567, 2006.

[6] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[7] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[8] Kevin Gurney. *An introduction to neural networks*. CRC press, 1997.

[9] Kaggle. Montreal crime data. https://www.kaggle.com/datasets/stevieknox/montreal-crime-data.

[10] Kaggle. Denver crime data. https://www.kaggle.com/datasets/paultimothymooney/denver-crime-data.

[11] Kaggle. Philadelphia crime data. https://www.kaggle.com/datasets/mchirico/philadelphiacrimedata.

[12] Aleksei Luchinsky Dan Mrachko Michael Boon, Xin Jin. Cs5630_sp2023_finalproject. `https://gitlab.com/dmrachk/cs5630_sp2023_finalproject`.