Chicago Crime Prediction Using Machine Learning

Lakshmanan Subramanian
A20539733
lsubramnanian@iit.hawk.edu

Jasti bhuvan sai
A20555206
Bjasti@hawk.Iit.edu

Mohan sasank mannava
A20554781
mmannava@hawk.iit.edu

***Abstract*—The project focuses on the utilization of machine learning to conduct an analysis and make predictions regarding the patterns of criminal activity that occur inside the city of Chicago. Through the utilization of sophisticated data preprocessing methods and the utilization of a variety of prediction models, the objective is to uncover insights about criminal activities, with a particular emphasis on the likelihood of arrests and the categorization of different types of crimes.**

I. GitHub Repository

The code and additional materials related to this research can be found on our GitHub repository:
https://github.com/Lakshman1000/ML-FinalProject-Grp-7

II. Background

The project's foundation rests on a thorough examination of Chicago's crime patterns. Understanding the dynamics and underlying variables that contribute to criminal behaviours is the goal of crime analysis, which is an essential part of urban studies and law enforcement. Here, Chicago stands out as a dynamic and distinctive location, offering a wealth of crime data to delve into. The project's dataset is derived from the Chicago Police Department's criminal reports. The data set contains a wide variety of information, such as the nature of the crime, the place where it occurred, the time and date, and indications of domestic violence and arrests. With its diversified population, unique neighbourhoods, and wide range of socioeconomic situations, Chicago provides an ideal setting for a comprehensive analysis of crime in metropolitan areas.
Dataset URL: https://data.cityofchicago.org/resource/crimes.json

III. Problem Statement:

The project's overarching goal is to address the pressing problems of crime type classification and arrest prediction in Chicago. The goal is to create reliable prediction models that can identify trends in crime data, which will help with things like classifying crimes and estimating the probability of arrests in response to recorded incidences. Important obstacles include fixing dataset imbalances, accurately identifying different sorts of crimes, improving the model, and understanding the results to draw conclusions that can help with city planning and police enforcement. In the field of urban security in Chicago, we aim to provide useful resources, insights, and tools to improve public safety, guide resource allocation, and assist decision-making

	id	case_number	date	block	iucr	primary_type	description	location_description	arrest	domestic	...	fbi_code	x_coordir
0	13204489	JG416325	2023-09-06T11:00:00.000	0000X E 8TH ST	0810	THEFT	OVER $500	PARKING LOT / GARAGE (NON RESIDENTIAL)	False	False	...	06	1176
1	13045102	JG226663	2023-03-30T09:16:00.000	080XX S DREXEL AVE	1544	SEX OFFENSE	SEXUAL EXPLOITATION OF A CHILD	APARTMENT	False	True	...	17	1183
2	13074891	JG262771	2023-05-10T12:43:00.000	028XX N MANGO AVE	1754	OFFENSE INVOLVING CHILDREN	AGGRAVATED SEXUAL ASSAULT OF CHILD BY FAMILY M...	RESIDENCE	False	True	...	02	1137
3	13099339	JG291745	2023-04-01T11:13:00.000	020XX N LAPORTE AVE	1751	OFFENSE INVOLVING CHILDREN	CRIMINAL SEXUAL ABUSE BY FAMILY MEMBER	RESIDENCE	False	True	...	17	1143
4	13121127	JG313964	2023-06-22T18:52:00.000	015XX W NORTH AVE	1153	DECEPTIVE PRACTICE	FINANCIAL IDENTITY THEFT OVER $ 300	AUTO / BOAT / RV DEALERSHIP	True	False	...	11	1165
...	...	...	...	...	...	...	...	...	...	...	...	...	...
16995	13226506	JG443096	2023-09-28T23:08:00.000	071XX S INDIANA AVE	0810	THEFT	OVER $500	STREET	False	False	...	06	1178
16996	13225395	JG441907	2023-09-28T01:58:00.000	011XX S TROY ST	0560	ASSAULT	SIMPLE	APARTMENT	False	True	...	08A	1155
16997	13226002	JG442565	2023-09-28T14:23:00.000	051XX W CONCORD PL	051A	ASSAULT	AGGRAVATED - HANDGUN	SIDEWALK	True	False	...	04A	1142
16998	13226464	JG443005	2023-09-28T09:40:00.000	048XX N WINTHROP AVE	0560	ASSAULT	SIMPLE	RESIDENCE - PORCH / HALLWAY	False	True	...	08A	1168
16999	13225516	JG441946	2023-09-28T03:55:00.000	026XX W 103RD PL	0910	MOTOR VEHICLE THEFT	AUTOMOBILE	STREET	False	False	...	07	1160

Fig. 1. Filtered Dataset

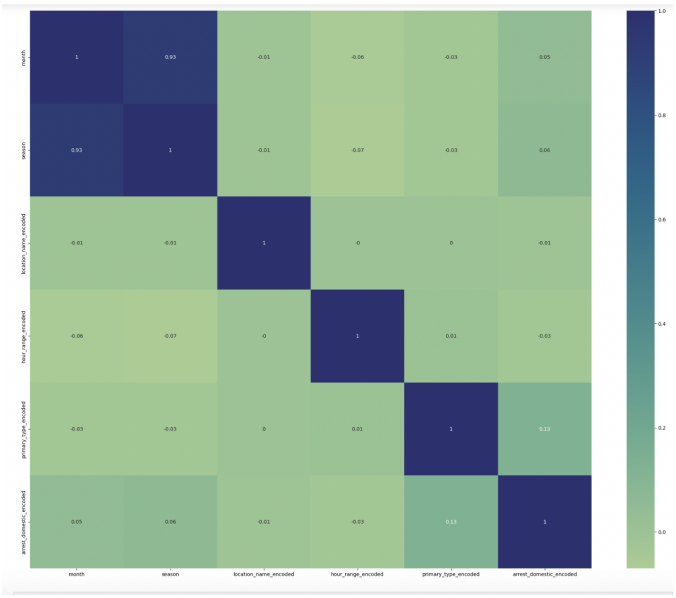

Fig. 2. Heat Map

IV. Model Selection

A wide range of machine learning techniques are utilized to tackle the difficulties of forecasting arrests and categorizing crime kinds in the city of Chicago. The **Random Forest classifier** was selected as the major method due of its adaptability and effectiveness. In addition, various algorithms such as **Decision Tree, XGBoost, and KNeighbors Classifier** are used for comparison and to establish a baseline for assessment.

A. *Model 1: Random Forest*

The Random Forest algorithm is well-suited for these tasks due to its ensemble learning nature, which combines multiple decision trees to improve overall predictive performance. This algorithm is robust against overfitting, handles both classification and regression tasks effectively, and accommodates a variety of data types.

Fig. 3. Random Forest Model

B. *Model 2: KNeighbors Classifier*

the KNeighbors Classifier is employed, offering a non-parametric methodology for classification applications. This technique use a measure of similarity between data points to generate predictions, providing an alternative viewpoint in contrast to models based on trees.

Fig. 4. KNeighbors Classifier

C. *Model 3: XGBoost algorithm*

The project utilizes the XGBoost algorithm, a gradient boosting approach renowned for its effectiveness and precision, to achieve improved performance. XGBoost is highly effective in dealing with imbalanced classes, making it especially suitable for crime prediction challenges where certain types of crimes may occur less frequently.

D. *Model 4: Decision Tree*

In addition, The Decision Tree method is utilized in conjunction with Random Forest. Decision Trees offer transparency and comprehension on the significance of features, facilitating the comprehension of the factors that impact forecasts. Nevertheless, Decision Trees are susceptible to overfitting, therefore the incorporation of Random Forest to alleviate this problem.

V. Experimental Methodology

A. *Data Properties:*

- Location Information: The **'location-name'** attribute provides specific information regarding the whereabouts of reported criminal incidents. The data has been encoded using label encoding to simplify the training of the machine learning model.
- Time-related data: The **'month'** attribute denotes the specific month when the crime took place, reflecting temporal patterns.
- The **'hour-range'** attribute denotes time periods during the day, offering more temporal context.
- The **'season'** attribute is obtained from the 'month' column and functions as an indicator of the season.
- Discrete Variables: The attribute **'primary-type'** denotes the category of crime and is used as the target variable for classifying crime types.
- **'arrest-domestic'**: Indicates if there was an arrest and if the incident involved domestic matters. This is utilized to generate the 'arrest-domestic' attribute.
- Label Encoding: Categorical variables, including 'location-name,' 'hour-range,' 'primary-type,' and 'arrest-domestic,' have undergone label encoding to transform them into numerical format for the purpose of model training.
- Addressing Imbalanced Classes: Classes that have a singular occurrence in the 'primary-type' column have been eliminated in order to rectify disparities in the dataset.
- Dependent Variables: **'arrest-domestic-encoded'**: This is the encoded version of the probability of being arrested after an incident, which is generated from 'arrest-domestic.'
- The **'primary-type-encoded'** refers to the encoded representation of crime types used for classification purposes.
- Size of the data: The dataset consists of **17,000** entries documenting crime events in Chicago throughout the year 2023.

B. *Hyperparameter Tuning*

The hyperparameter settings play a crucial role in the performance of machine learning models. In this project, Random Forest and KNeighbors Classifier are the two models for which hyperparameter tuning has been conducted, and they exhibit the highest accuracy.

Random Forest :

Arrest Model :

```
In [101]: param_dist = {
              'n_estimators': randint(100, 1000),
              'max_depth': randint(3, 20)
          }


          random_search = RandomizedSearchCV(estimator=random_forest_arrest_model, param_distributions=param_dist, n_iter=100,


          random_search.fit(X_val_arrest, y_val_arrest)

          # Get the best parameters and the best score
          best_params_rand = random_search.best_params_
          best_score_rand = random_search.best_score_

          print("Best Parameters (Randomized Search):", best_params_rand)
          print("Best Score (Randomized Search):", best_score_rand)

          Best Parameters (Randomized Search): {'max_depth': 4, 'n_estimators': 396}
          Best Score (Randomized Search): 0.7285882973207322
```

Primary Model :

```
In [102]: param_dist = {
              'n_estimators': randint(100, 1000),
              'max_depth': randint(3, 20)
          }


          random_search = RandomizedSearchCV(estimator=random_forest_primary_model, param_distributions=param_dist, n_iter=100


          random_search.fit(X_val_primary, y_val_primary)

          # Get the best parameters and the best score
          best_params_rand = random_search.best_params_
          best_score_rand = random_search.best_score_

          print("Best Parameters (Randomized Search):", best_params_rand)
          print("Best Score (Randomized Search):", best_score_rand)

          Best Parameters (Randomized Search): {'max_depth': 5, 'n_estimators': 138}
          Best Score (Randomized Search): 0.2321289917788526
```

Fig. 5. Hypermeter Tuning - Random Forest

KNeighbors :

Arrest Model :

```
In [103]: param_dist = {
              'n_neighbors': randint(1, 50),  # Number of neighbors
              'weights': ['uniform', 'distance'],  # Weight function used in prediction
              'p': [1, 2]  # Power parameter for Minkowski distance
          }


          # Perform randomized search using RandomizedSearchCV
          random_search = RandomizedSearchCV(
              knn_arrest_model,
              param_distributions=param_dist,
              n_iter=100,  # Number of parameter settings sampled
              cv=5,  # Number of cross-validation folds
              scoring='accuracy',  # Evaluation metric
              random_state=42  # Random seed for reproducibility
          )

          # Fit the randomized search to your data
          random_search.fit(X_val_arrest, y_val_arrest)  # Replace X_train and y_train with your data

          # Print the best parameters and best score
          print("Best Parameters: ", random_search.best_params_)
          print("Best Score: ", random_search.best_score_)

          Best Parameters:  {'n_neighbors': 39, 'p': 2, 'weights': 'uniform'}
          Best Score:  0.7274070491453271
```

Primary Model :

```
In [104]: param_dist = {
              'n_neighbors': randint(1, 50),  # Number of neighbors
              'weights': ['uniform', 'distance'],  # Weight function used in prediction
              'p': [1, 2]  # Power parameter for Minkowski distance
          }


          # Perform randomized search using RandomizedSearchCV
          random_search = RandomizedSearchCV(
              knn_primary_model,
              param_distributions=param_dist,
              n_iter=100,  # Number of parameter settings sampled
              cv=5,  # Number of cross-validation folds
              scoring='accuracy',  # Evaluation metric
              random_state=42  # Random seed for reproducibility
          )

          # Fit the randomized search to your data
          random_search.fit(X_val_primary, y_val_primary)  # Replace X_train and y_train with your data

          # Print the best parameters and best score
          print("Best Parameters: ", random_search.best_params_)
          print("Best Score: ", random_search.best_score_)

          Best Parameters:  {'n_neighbors': 41, 'p': 2, 'weights': 'uniform'}
          Best Score:  0.21027219687760074
```

Fig. 6. Hypermeter Tuning - KNeighbour classifer

*1) Random Forest : Number of Estimators (**n-estimators**) A key hyperparameter representing the number of decision trees in the forest. It is set using a Randomized Search over a range from 100 to 1000 to find the optimal value:* :
(**max-depth**): Another critical hyperparameter specifying the maximum depth of each decision tree. It is set using a Randomized Search over a range from 3 to 20 to find the optimal value.

2) KNeighbors Classifier : The major hyperparameters of the KNeighbors classifier are n-neighbors (the number of neighbors), weights (the weighting of neighbor contributions), algorithm (the method used for neighbor calculation), leaf-size (the size of tree leaf nodes), p (the power of the Minkowski distance), and metric (the distance metric). Grid search and cross-validation are widely used methods for efficiently optimizing these parameters. For example, when we investigate values such as 3, 5, and 7 for the number of neighbors, different weights, and various distance metrics, it improves the performance of the model. The procedure entails assessing several combinations of hyperparameters to determine the ideal configuration, hence enhancing the classifier's accuracy in predicting outcomes.: Check Fig 4 and Fig 5 for Reference

C. Testing and Training:

Divide the dataset into separate sets for training and testing purposes. The allocation of data is as follows: **60 perecent is used for training**, **20 percent for testing**, and **20 percent for validation**. The training set is utilized to instruct the machine learning model, whereas the testing set is employed to assess its performance. Aside from the training and testing sets, a validation set was utilized to refine the hyperparameters of the model.
The cross-validation outcomes for the chosen models in forecasting arrests ('arrest-domestic-encoded') demonstrate diverse levels of accuracy among various algorithms. The Random Forest model exhibits the highest average cross-validation accuracy (**72.4 percent**), demonstrating its efficacy in regularly forecasting arrests. Conversely, the Decision Tree model exhibits a lower average accuracy rate of **55.99 percent**, indicating the possibility of overfitting. The mean accuracies of XGBoost and KNeighbors Classifier are **66.13 percent** and **72.50 percent**, respectively, indicating that they perform at a similar level. These findings offer valuable information on the ability of the models to apply their knowledge to new situations, which helps in choosing the most dependable algorithms for forecasting arrests in Chicago.

VI. Result Analysis

A. Cross Validation:

The result analysis of the cross-validation scores provides valuable insights into the performance of the selected models for predicting arrests in the city of Chicago. Here are the key observations:

1) Random Forest:: Demonstrates the highest mean cross-validation **accuracy of 72.4 and F1- Score of 61.8 percent**. Consistent performance across folds suggests robustness and reliability.

2) Decision Tree:: Exhibits a lower mean cross-validation **accuracy of 55.99 and F1- Score of 57.6 percent** .
The variance in scores across folds indicates potential overfitting, highlighting limitations.

3) XGBoost:: Shows a competitive mean cross-validation **accuracy of 66.13 F1- Score of 61.9 percent**.
Offers a balanced performance, indicating effectiveness in predicting arrests.

4) KNeighbors Classifier:: Achieves a strong mean cross-validation **accuracy of 72.50 F1- Score of 61.9 percent** Consistent and high scores across folds demonstrate reliability and effectiveness.

VII. Conclusion

Eventually, the research use machine learning models to forecast arrests and categorize crime categories in Chicago using crime data. The dataset is prepared for training by undergoing thorough preprocessing, feature engineering, and label encoding. Four models, specifically Random Forest, Decision Tree, XGBoost, and KNeighbors Classifier, are chosen and assessed using cross-validation.

The findings suggest that both Random Forest and KNeighbors Classifier have strong and reliable performance, with mean cross-validation accuracies of 72.4 percent and 72.5 percent respectively. These models exhibit a high degree of accuracy in forecasting arrests, hence demonstrating their efficacy in dealing with the intricacies of urban crime data. To improve the performance and interpretability of the model, additional investigation could include hyperparameter tuning and study of feature importance. Refer Fig 9 for Predicted Output

In essence, the project serves to enhance comprehension of crime patterns in Chicago, offering valuable insights that might assist law enforcement and urban planning endeavors.

A. Contribution:

1. Lakshmanan focused on Data Preprocessing, Model Implementation, Hyperparameter Tuning and in Report Documentation
2. Jasti Bhuvan Sai focused on Data Collection, Model Implementation and in Report Documentation
3. Mohan sasank mannava focused on specific models for cross-validation, evaluating the models on different subsets of the data

Cross-Validation :

```
In [100]: models = [
              ('Random Forest', random_forest_arrest_model),
              ('Decision Tree', decision_tree_arrest_model),
              ('XGBoost', xgb_arrest_model),
              ('KNeighbors Classifier', knn_arrest_model)
          ]

          # Iterate through each model and perform cross-validation
          for model_name, model in models:
              # Perform cross-validation with 5 folds
              cv_scores = cross_val_score(model, X_val_arrest, y_val_arrest, cv=5)

              print(f"Model: {model_name}")
              print("Cross-validation scores:", cv_scores)
              print("Mean CV Accuracy:", cv_scores.mean())
              print("==============================")

Model: Random Forest
Cross-validation scores: [0.72123894 0.72230428 0.7267356  0.72082718 0.72968981]
Mean CV Accuracy: 0.7241591613181526
==============================
Model: Decision Tree
Cross-validation scores: [0.55752212 0.55391433 0.58197932 0.53323486 0.57311669]
Mean CV Accuracy: 0.5599534646605926
==============================
Model: XGBoost
Cross-validation scores: [0.64896755 0.65140325 0.6676514  0.66026588 0.67799114]
Mean CV Accuracy: 0.6612558441501853
==============================
Model: KNeighbors Classifier
Cross-validation scores: [0.72271386 0.72082718 0.7267356  0.7267356  0.7282127 ]
Mean CV Accuracy: 0.7250449885186687
==============================
```

Fig. 7. Cross Validation

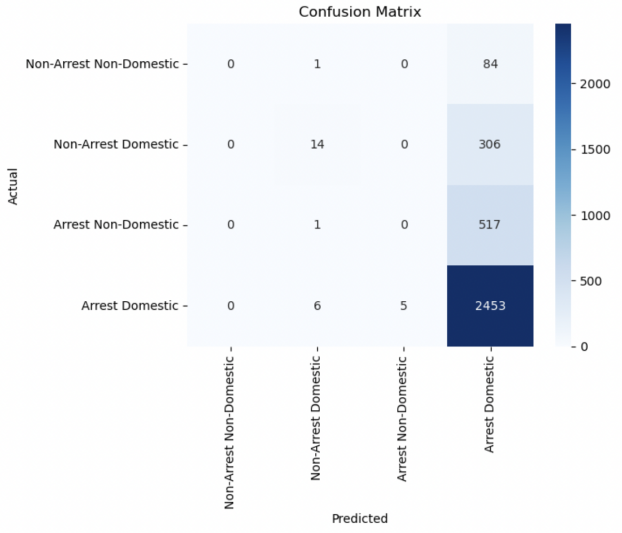

Fig. 8. Confusion Matrix

```
Location: E 8TH ST - Time Range: 11 am - 12 pm - Month: 1 - Season: 1
Top 10 Primary Types with Probabilities:
STALKING: 1.03%
CRIMINAL DAMAGE: 1.31%
KIDNAPPING: 2.46%
PUBLIC PEACE VIOLATION: 4.82%
BURGLARY: 5.17%
DECEPTIVE PRACTICE: 6.76%
OBSCENITY: 10.16%
NARCOTICS: 10.93%
SEX OFFENSE: 21.57%
CRIMINAL SEXUAL ASSAULT: 33.82%
Arrest_Domestic Probabilities:
Arrest_Domestic: 4.36%
Non_Arrest_Domestic: 14.43%
Arrest_Non_Domestic: 8.95%
Non_Arrest_Non_Domestic: 72.26%
```

Fig. 9. Predicted Output