LEVERAGING MACHINE LEARNING FOR ADVANCED CRIME PREDICTION IN LOS ANGELES

A Thesis

Submitted to the Graduate Faculty of the

National University, Department of Engineering & Computing

in partial fulfillment of the requirements for the degree of

Masters of Science in Data Science

Prepared By:

Magnus Aghe

Heng Ly Aun

Elijah Walker

March 2024

MASTER'S THESIS APPROVAL FORM

We certify that we have read the project of Magnus Aghe, Heng Ly Aun, and Elijah Walker entitled LEVERAGING MACHINE LEARNING FOR ADVANCED CRIME PREDICTION IN LOS ANGELES and that, in our opinion, it is satisfactory in scope and quality as the thesis for the degree of Master of Science in Data Science at National University.

Approved:

Faye Anderson 4/3/2024

Dr. Faye Anderson, Capstone Project Sponsor Date
Professor, Department of Data Science
National University

Shatha Jawad 4/14/2024

Dr. Shatha Jawad, Capstone Project Advisor Date
Professor, School of Technology & Engineering
National University

 4/14/2024

Dr. Mario Missakian, Capstone Instructor Date
Professor, School of Technology & Engineering
National University

Acknowledgements

We wish to express our sincere gratitude to our capstone committee and the entire faculty at National University's Data Science program for providing us with a rigorous and enriching academic journey. Special thanks to Dr. Shatha Jawad, Dr. Faye Anderson, and Dr. Mario Missakian for their invaluable support and mentorship throughout the completion of our thesis. Your expertise and guidance were instrumental in enabling us to enhance our research capabilities and to evolve as data scientists. We are also profoundly grateful to Dr. Jodi Reeves, the Master's in Data Science Academic Program Director, for maintaining the highest standards of excellence for both the program and its students. Your commitment to quality education has significantly contributed to our professional growth and readiness to contribute to the field of data science.

Abstract

This study employs machine learning to enhance predictive policing by analyzing Los Angeles Police Department crime reports and weather data from 2010 to 2023. It aims to forecast crime locations and categorize crimes based on weather conditions and victim demographics. Employing Long Short-Term Memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA) for temporal and spatial crime analysis, alongside a suite of algorithms for crime classification, the research covers data preparation, exploratory analysis, feature selection, and model evaluation. The findings underscore the benefits of integrating diverse data sources for improved prediction and classification accuracy. Offering actionable insights for law enforcement, this work demonstrates machine learning's role in advancing predictive policing, ultimately contributing to more effective crime prevention and enhanced public safety.

Keywords: Predictive Policing, Machine Learning, Crime Forecasting, Spatiotemporal Analysis, Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), Crime Classification

Table of Contents

List of Tables

List of Illustrations

Chapter 1: Introduction

1.1 Background

Los Angeles (LA) has witnessed a significant escalation in crime rates, from 270,000 reported incidents in 2010 to an alarming 400,000 by 2023 (Los Angeles Police Department, 2024). This rise is attributed to the impact of the pandemic, as pre-pandemic levels were approximately 200,000 cases per year. This surge, particularly noticeable in the latter part of 2023, with an average of 800 to 900 daily crimes, places considerable strain on law enforcement resources. In response, the Los Angeles Police Department (LAPD) augmented its Hollywood division by 200 officers (Hayes, 2022). This move raises questions about the adequacy of such measures and whether a broader, more strategic allocation of resources is required citywide.

Our research adopts a data-driven approach to bolster predictive policing in LA, merging LAPD crime data with weather statistics to potentially preempt crime occurrences. Our goal is to furnish the LAPD with actionable insights for resource deployment, transitioning from reactive to proactive crime prevention strategies. The inclusion of weather data is predicated on the hypothesis that certain environmental conditions might significantly influence crime rates, thus adding a valuable dimension to predictive models.

This study aims to refine the operational framework of crime prediction, focusing on the accuracy of our analytics. By examining historical crime data across LA's diverse neighborhoods, including Hollywood, we seek to uncover patterns that could enhance predictive models. The integration of weather data aims to augment the model's predictive power. While our findings will not directly dictate police deployment strategies, they offer law enforcement valuable insights for informed strategic planning.

1.2 Problem Statement

Despite the Los Angeles Police Department's efforts to combat rising crime rates, including the augmentation of its Hollywood division, the persistent surge in criminal incidents underscores a critical gap in current predictive policing strategies. Traditional reactive approaches remain limited in their ability to adapt to the dynamic and complex nature of urban crime, reflecting a pressing need for a more nuanced, proactive methodology (Ensign et al., 2018).

The central challenge lies in the absence of a comprehensive model that integrates diverse yet significant factors influencing crime rates, such as temporal patterns, geographic distributions, and particularly, the impact of weather conditions. While the LAPD has access to vast amounts of crime data, the potential of this data remains underutilized due to a lack of advanced analytical tools capable of synthesizing these multifaceted elements into actionable predictions.

This research seeks to bridge this gap by developing an innovative predictive model that combines LAPD's extensive crime data with detailed meteorological information. The core hypothesis posits that a deeper understanding of the relationship between weather patterns and crime occurrences can significantly enhance the predictive accuracy of crime models. Such a model not only promises to revolutionize the existing framework of predictive policing in Los Angeles but also offers a template for other metropolitan areas grappling with similar challenges. By addressing this problem, the study aims to provide the LAPD and potentially other law enforcement agencies with a powerful tool for strategic resource allocation, ultimately contributing to more effective crime prevention and improved public safety. The anticipated outcome is a shift from a predominantly reactive policing model to one that is anticipatory, informed by a rich analysis of historical crime trends and environmental conditions.

1.3 Objectives

Our research is guided by a set of well-defined objectives aimed at exploring the complex dynamics of crime in Los Angeles. We adopt a data-driven methodology to refine predictive policing strategies, leveraging an extensive LAPD dataset spanning from 2010 to 2023, in addition to a weather dataset, analogous to the methodology of Stec & Klabjan (2018), this study incorporates weather data to enhance the accuracy of crime prediction models. Central to our project are three pivotal questions that guide our analysis, each contributing to our overarching goal of developing a sophisticated crime prediction model.

Firstly, we aim to identify and analyze the hotspots of crime occurrences within LA, recognizing that crime is not uniformly distributed across the city. By examining factors such as the time of day, day of the week, and geographical locations, coupled with the types of crime prevalent in these hotspots, we intend to uncover underlying patterns that can inform predictive models. This objective is crucial for deploying police resources more effectively, enabling a targeted approach to crime prevention in areas most in need.

Secondly, the role of victim demographics in crime prediction constitutes a vital area of our investigation. We hypothesize that certain demographic variables may correlate with specific crime types, suggesting patterns or trends that could preemptively inform law enforcement strategies. This objective involves a detailed analysis of demographic data in conjunction with crime statistics, aiming to identify potential risk factors and vulnerabilities within different communities' segments.

Lastly, we explore the influence of weather conditions on crime patterns and frequencies in LA. This objective involves constructing a model that integrates meteorological data with crime records to ascertain whether certain weather conditions act as catalysts for specific types of crime. The insights gleaned from this analysis could significantly enhance the LAPD's ability to

anticipate crime occurrences based on weather forecasts, optimizing resource planning and deployment.

Together, these objectives form the foundation of our research, each contributing to a holistic understanding of crime dynamics in Los Angeles. By achieving these goals, we anticipate providing the LAPD with a multifaceted predictive tool that not only forecasts crime occurrences but also offers strategic insights for preemptive action. Our aim is to contribute to the creation of safer communities through the application of advanced analytics and machine learning in the realm of predictive policing.

1.4 Significance of the Study

The study holds potential for advancing predictive policing strategies by developing a model that integrates extensive crime data with meteorological information. It seeks to provide law enforcement agencies, such as the LAPD, with enhanced capabilities for strategic resource allocation and crime prevention, thereby improving public safety. This model's predictive accuracy, informed by the relationship between weather patterns and crime occurrences, could enhance predictive policing not only in Los Angeles but potentially in other metropolitan areas facing similar challenges.

Its implications propose a scalable and adaptable approach to predictive policing. By moving towards anticipatory crime prevention informed by advanced analytics and machine learning, the study represents a shift in law enforcement methodologies. The aim is to transition from reactive to proactive policing models, leveraging historical crime trends and environmental conditions to create safer communities through strategic, data-driven insights.

1.5 Research Hypothesis

The underlying premise of this investigation posits that the random patterns of criminal behavior are subject to prediction through the application of sophisticated machine learning techniques. The study is predicated on several interrelated hypotheses, each aiming to dissect the intricate web of crime predictability. The first hypothesis asserts that spatial-temporal crime hotspots within the district of the Los Angeles Police Department (LAPD) can be accurately forecasted using historical crime data. Recent studies have shown that machine learning models, such as LSTM and RNN, effectively conduct spatiotemporal analyses, demonstrating the capability to uncover long-term dependencies and trends in crime data (Yao et al., 2020; Liang et al., 2022). This is grounded in the notion that crime, although seemingly random, follows discernible patterns influenced by many factors that can be decoded through rigorous data analysis.

Table 1 captures this crime hotspot prediction hypothesis. The target variable is Crime_Incidence, which indicates the amount of crime committed daily in a geographical area. It is derived in our dataset by grouping AREA_NAME which is a feature representing the 21 geographical areas in Los Angeles with DATE_OCC, the date that the crime occurred. These variables will be discussed in detail in our methodology chapter. Through time series machine learning models such as ARIMA and LSTM, we aim to predict crime occurrences in each of the 21 areas of Los Angeles, plus the city as one big hotspot.

Table 1

Hypothesis 1: Crime Hotspot Prediction

Target Variable	Target Type	Data Type	Predictor Variables	Data Type	Statistical / ML Method	Number of Models	Student
Crime_Incidence	Calculated field = Daily Amount of Crime grouped by Area	int64	DATE_OCC	datetime	Time Series - ARIMA	22	Magnus Aghe
					Time Series - LSTM	22	Magnus Aghe
44 Models in total, 1 for each ML method for each of the 21 Areas, plus the aggregate L.A area.							

The second hypothesis proposes that weather conditions correlate with crime incidence and typology, offering predictive insights that can be instrumental in identifying future crime hotspots. Studies performed in Little Rock, Arkansas, by Sabakhtarishvili et al. (2023) indicate a quantifiable relationship between weather and crime rates. It suggests that integrating meteorological data into predictive models can enhance their accuracy. Table 2 lists the predictor variables that will predict categories or types of crime. We shall test the effect of weather features on crime-category classification in L.A using a variety of machine learning models.

Table 2

Hypothesis 2: Effect of Weather on Crime Type Prediction

Target variable: Crime_Category_Code
Derived field = Crime_Category (8 classes) mapped from Crm_Cd_Desc (144 classes)
Data type: Categorical

Predictor Variables and Data Type		**Algorithms, number of models, and student**		
Predictor Variables	Data Type	Statistical / ML Method	Number of Models	Student
TIME_OCC	categorical (4)	Decision Trees	2	Elijah Walker
AREA	int64	XGBoost	2	Elijah Walker
Part_1-2	categorical (2)	Random Forest	2	Heng Ly Aun
Vict_Age	float64	Logistic Regression	2	Heng Ly Aun
Vict_Sex	categorical (3)			
Region_Ethnic_Origin	categorical (6)			
Avg_Temp	float64			
Avg_Dewpoint	float64			
Avg_Humidity	float64			
Avg_Windspeed	float64			
Avg_Pressure	float64			
Total_Precipitation	float64			
Weapon_Reported	categorical (2)			
Year	categorical (14)			
Month	categorical (12)			
Day_of_week	categorical (7)			
18 variables	**58 variables**			

Model 1 - With weather predictor features (4 models, one for each statistical/ML method)

Model 2 - Without weather predictor features (4 models, one for each statistical/ML method)

8 models in total

In our third hypothesis, we argue for the significance of victim demographics in patterning crime, contending that these factors, in conjunction with others, are pivotal in refining the accuracy of crime forecasts. During a study conducted between 2015 to 2018 in southeast China, victim demographic data was identified as a quantifiable factor that intersects with criminal occurrences, with machine learning algorithms elucidating statistically significant patterns that can refine crime forecasts (Zhang et al., 2020). It is predicated on the idea that

demographic factors intersect with criminal occurrences in statistically significant ways that can be elucidated through machine learning algorithms and applied to other locations like Los Angeles. Just like in Table 2, the target variable here is Crime_Category_Code as shown in Table 3. The presence and absence of victim demographics in the models built will test for the effect of victim demographics on crime type prediction.

Table 3

Hypothesis 3: Effect of Victim Demographics on Crime Type Prediction

Target variable: Crime_Category_Code
Derived field = Crime_Category (8 classes) mapped from Crm_Cd_Desc (144 classes)
Data type: Categorical

Predictor Variables and Data Type		**Algorithms, number of models, and student**		
Predictor Variables	Data Type	Statistical / ML Method	Number of Models	Student
TIME_OCC	categorical (4)	Decision Trees	2	Elijah Walker
AREA	int64	XGBoost	2	Elijah Walker
Part_1-2	categorical (2)	Random Forest	2	Heng Ly Aun
Vict_Age	float64	Logistic Regression	2	Heng Ly Aun
Vict_Sex	categorical (3)			
Region_Ethnic_Origin	categorical (6)			
Avg_Temp	float64			
Avg_Dewpoint	float64			
Avg_Humidity	float64			
Avg_Windspeed	float64			
Avg_Pressure	float64			
Total_Precipitation	float64			
Weapon_Reported	categorical (2)			
Year	categorical (14)			
Month	categorical (12)			
Day_of_week	categorical (7)			
18 variables	**58 variables**			

Model 1 - With victim demographics predictor features (4 models, one for each statistical/ML method)

Model 2 - Without victim demographics predictor features (4 models, one for each statistical/ML method)

8 models in total

To substantiate these hypotheses, the research will undertake a comprehensive analysis using a large dataset from the LAPD, which includes over 2,886,530 crime records with 28 distinctive attributes. This will be augmented by a detailed examination of a separate weather dataset containing 5,113 entries with 17 variables to identify potential correlations with crime rates. Additionally, the study will analyze victim demographic data to determine its statistical relevance to crime patterns and assess how much it can enhance predictive models.

The study's methodological framework will incorporate advanced machine learning models such as Recurrent Neural Networks' (RNN) Long Short-Term Memory (LSTM) to conduct spatio-temporal analyses, leveraging their capacity to detect long-term dependencies and trends. This will be complemented using the Autoregressive Integrated Moving Average (ARIMA) model, renowned for its time-series forecasting accuracy. It focuses on univariate crime statistics to predict future trends from historical data. LSTM and ARIMA models have been shown to improve crime prediction accuracy in Chicago (Safat et al, 2021).

A suite of machine learning algorithms, including decision trees, random forests, gradient boosting machines, and logistic regression, will be deployed to assess the integrity of these hypotheses. The selection of algorithms may expand as more analysis is performed. Many different models have shown success in various parts of crime prediction, for example, decision trees, bagging, and AdaBoost achieving accuracies as high as 99% when predicting crime based on time and location data (Yuki et al, 2019). The performance of these models will be evaluated considering their ability to manage class imbalance and high dimensionality, with an expectation that balanced datasets and optimized feature sets will enhance the performance of the predictive models.

This study seeks to forge a significant advancement in predictive policing by systematically validating machine learning to improve the precision of crime prediction models.

By incorporating complex variables such as weather and demographics, this research aims to refine and elevate law enforcement tactics through data-driven insights, potentially setting a benchmark for predictive policing models that can be adopted in various areas and cities.

1.6 Limitations of the Study

This study faces several notable limitations in its pursuit of harnessing machine learning for crime prediction. The stable weather conditions characteristic of Los Angeles may impede the discovery of meaningful correlations between weather variables and crime rates, as the lack of variability in climatic factors could result in limited predictive insights. Additionally, the dataset's omission of perpetrator demographics means that the predictive models may miss critical factors influencing crime, such as the relationship between offender profiles and criminal activities. Last, using advanced artificial intelligence and machine learning techniques creates a tradeoff between model transparency and accuracy.

While providing in-depth local analysis, the geographical focus on Los Angeles may translate poorly to other regions where different environmental, social, or economic conditions prevail. The uniqueness of the city's demographic composition and urban infrastructure may lead to predictive models that are only universally applicable or scalable with significant adaptation.

From a methodological perspective, limiting the data from the years 2010 to 2023 imposes a limitation on the study. Machine learning models, particularly the Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNN) excel in environments where they can learn from sequential and temporally rich data. The lack of such dynamism in historical datasets may prevent these models from realizing their full predictive potential. While effective for many time-series forecasting applications, the Autoregressive Integrated Moving Average (ARIMA)

model might need to be more suitable for capturing the complex, multi-variable dependencies in crime data, which often exhibit non-linear and non-stationary behaviors (Kim, 2023).

Although robust in many respects, the chosen machine learning algorithms—Multinomial Logistic Regression, Decision Trees, Random Forest, and XGBoost—have limitations. Random Forest and XGBoost may offer better performance with unbalanced datasets, but they can be computationally intensive and may overfit if not correctly tuned. XGBoost has been shown to complement LSTM and ARIMA models in crime prediction as well (Safat et al, 2021). Creating a symbiotic relationship between multiple different models could pose quite a challenge when hyperparameter tuning, and later, model transparency. The topic of model accuracy and transparency may be a bit more nuanced than previously thought, and the distinction between black-box and interpretable models may not always align with their respective levels of accuracy and transparency (Bell et al, 2022).

The dataset may not represent all crime types or contain biases in crime reporting and recording practices. It is critical to maintain as much model transparency in mind, especially with current issues such as racial profiling being prevalent. Issues such as reporting delays, misclassification of crimes, and incomplete records can introduce noise into the dataset, potentially affecting the models' accuracy. Class balancing techniques aim to mitigate the influence of imbalanced data. Still, they can also introduce their own biases, particularly if minority classes are oversampled or if majority classes are under-sampled to the extent that critical information is lost.

1.7 Definition of Terms

Spatiotemporal Data Analysis

This refers to the analysis of data collected across both space and time. Spatiotemporal crime hotspots will uncover patterns of crime across geographic locations over a certain period of time.

Long Short-Term Memory (LSTM)

LSTM is a deep learning, sequential neural network that allows information to persist. It is a special type of recurrent neural network (RNN) that addresses the vanishing gradient problem faced by RNNs.

Recurrent Neural Network (RNN)

A recurrent neural network is a type of artificial neural network that uses sequential data or time series data.

Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a statistical analysis model that uses time series data to better understand the data set or predict future trends.

Time Series Forecasting

This refers to forecasting or predicting the future value over a period of time. It entails developing models based on previous data and applying them to make observations and guide future strategic decisions.

Decision Tree Algorithm

A decision tree algorithm is a supervised learning method that uses a tree-like model of decisions and their possible consequences to make predictions.

Random Forest (RF)

This is an ensemble learning algorithm that uses a collection of decision trees to make predictions.

Gradient Boosting Machine

This is a powerful boosting algorithm that combines several weak learners into strong learners, where each new model is trained to minimize the loss function of the previous model using gradient descent. XGBoost refers to extreme gradient boosting machines.

Class Imbalance

This refers to the situation in a dataset where the distribution of categories or classes is highly uneven.

Model Transparency

This refers to the degree to which a model's inner workings can be explained and/or understood.

Model Accuracy

Model accuracy refers to the proportional measure of the number of correct predictions over all predictions.

1.8 Summary

The escalation of crime rates in Los Angeles, with incidents rising from 270,000 in 2010 to 400,000 by 2023, necessitates a shift from traditional policing strategies to predictive policing. This study proposes a data-driven model leveraging LAPD crime data and weather statistics to predict crime occurrences, aiming for a proactive approach to crime prevention. It seeks to analyze historical crime data, incorporating weather data to enhance model accuracy, and provide actionable insights for strategic law enforcement planning. Addressing the challenge of integrating various influential factors on crime rates, such as victim demographics, temporal patterns, and weather conditions, the research aims to develop a predictive model to assist in strategic resource allocation by predicting crime hotspots and improving public safety, underscoring the need for advanced analytical tools in predictive policing.

Chapter 2: Literature Review

2.1 Introduction

As we delve into the literature on crime prediction, the transition from traditional statistical methodologies to advanced ML and AI techniques emerges as a pivotal paradigm shift in predictive policing. This evolution, driven by technological progress since the 1960s underscores the integration of sophisticated AI technologies into law enforcement practices (Hernandez, 1990). A close examination of recent studies by Zhang et al. (2023) and Khan et al. (2022) illuminates the influential role of cutting-edge algorithms like Long Short-Term Memory (LSTM) and Gradient Boosting Decision Trees in enhancing crime prediction methodologies. Through a synthesis of research findings, it becomes evident that the strategic incorporation of various variables, including victim demographics, temporal attributes, weather conditions, crime hotspots, and environmental data, holds promise for improving the accuracy and efficacy of predictive policing models. This review underscores the transformative impact of data-driven insights and innovative methodologies in shaping the landscape of predictive policing toward more efficient and informed crime prevention strategies.

2.2 Crime and Crime Prediction: Influencing Factors & Perspectives

Several factors, including socio-economic conditions, demographics, access to resources, law enforcement presence, community dynamics, and environmental factors influence crime. These elements can contribute to the occurrence and patterns of criminal activities within a given area. According to Saradha et al. (2021), criminal behavior is influenced by multiple variables such as demographics, socio-economic factors, environmental aspects, and network social data. It also highlights that criminals tend to choose environments with which they are familiar and that crime hotspots, areas with high crime concentrations, can be predicted based on spatial and

temporal patterns with accuracy as high as 99.25% (Yuki et al., 2019). Hardyns and Khalfa (2022) discuss the significance of the environment in generating or deterring crime. They highlight how crime is not randomly distributed but is influenced by interactions between personal and environmental characteristics, emphasizing the spatial-temporal clustering of crime and the concept of crime hot spots.

Ying-Lung et al (2018) discuss factors that inform crime such as population density, the presence of gangs or parolees, geographical characteristics of the grid, and environmental features like weather and temperature. Wu et al. (2022) discuss the factors influencing crime such as routine activities theory, social disorganization theory, and opportunity theories. The routine activities theory emphasizes mobility and social characteristics of micro-places, while social disorganization theory focuses on neighborhood-level social interactions. Additionally, the opportunity theory suggests that the presence of targets like people and property can impact the likelihood of crime occurring.

In historical context, crimes were traditionally treated reactively, with law enforcement relying on time-consuming methods. However, with the rise of criminal activities over the years, there has been a growing need for innovative technologies and novel techniques to enhance crime analytics to safeguard communities (Saradha et al., 2021). Hardyns et al. (2022) discuss the historical perspective of crime concerning predictive modeling and spatial-temporal criminology. The authors highlight the importance of understanding crime due to interactions between personal and environmental characteristics. They emphasize the spatial-temporal clustering of crime and the significance of the environment in generating or discouraging criminal behavior. This historical background provides insights into the spatial-temporal

dimensions of criminal opportunity and the non-random distribution of crime in specific geographical settings over time.

2.3 Understanding the Fundamentals of Crime Prediction

Crime prediction involves the application of machine learning algorithms to analyze historical crime data in different geographical locations, enabling the anticipation of potential crime locations and types in the future. By employing Artificial Intelligence and Machine Learning methodologies, researchers and analysts strive to construct models that aid law enforcement agencies in identifying high-crime areas, predicting criminal incidents, and implementing preemptive measures to deter them (Sharma et al., 2021, p. 23). The primary objective of crime prediction is to curtail criminal activities, mitigate economic losses, and enhance community safety and well-being (Saradha et al., 2021).

The overarching goal is to forecast the timing and locations of potential crimes through the analysis of historical crime data, environmental conditions, socio-economic factors, and social network data. Ying-Lung et al. (2018) emphasize that crime prediction models are influenced by several factors such as time, space, crime type, and geographic attributes. Notably, geographic features play a pivotal role in grid-based crime prediction models by simulating and predicting crime displacement, enabling law enforcement to forecast high-risk zones and bolster crime prevention strategies. Their research integrates experiential insights with additional geographic data sourced from the Google Places API to enhance predictive accuracy.

According to Roshankar et al. (2023), crime prediction involves the examination of crime data patterns and trends to pinpoint areas and times with a higher likelihood of experiencing criminal activities. The goal is to accurately forecast the occurrence and characteristics of criminal activities in specific locations and time periods. Factors such as spatial dependencies

and temporal dynamics are crucial in understanding and predicting crime activities. Saradha et al. (2021) highlights the importance of analyzing and understanding the complex relationships between diverse types of crimes and their occurrences in specific locations and times. Crime prediction models leverage machine learning algorithms and advanced data analytics to enhance law enforcement efforts in crime detection, prevention, and control.

2.4 Challenges of Law Enforcement in Combating Crime

Acknowledging the challenges encountered by law enforcement in combating crime, Sharma et al. (2021) emphasizes the complexities associated with statistical analysis of crime data and the necessity for more precise predictive models. Prior to the implementation of the Uniform Crime Reporting (UCR) system, the statistical analysis of crime datasets presented significant difficulties, hampering law enforcement efforts due to the lack of comprehensive analytical insights into crime data (Sharma et al., 2021, p. 16). The challenges in fighting crime vary across geographical locations.

In Little Rock, Arkansas, Sabakhtarishvili et al. (2023) identify the hurdles of effectively managing limited resources, reducing crime rates in a city consistently ranked among the top 15 most dangerous in the United States, and addressing the inefficiencies and high costs associated with traditional crime prevention methods such as increased patrols and targeted investigations. Additionally, Raji et al. (2023) emphasize the importance of understanding and considering several factors influencing crime, including neighborhood characteristics, law enforcement resources, and the quality and diversity of data used for analysis, as pivotal challenges for law enforcement. The authors also draw attention to the utilization of machine learning algorithms as a potential tool to streamline data analysis for crime prevention while cautioning about limitations linked to false positive rates that could lead to social and ethical implications.

Exploring crime prediction using geographical features in Taoyuan, one of Taiwan's major cities, Ying-Lung et al. (2018) highlighted how traditional crime prevention strategies heavily lean on the experiential knowledge of senior law enforcement officers, posing challenges in automating the storage, transfer, and application of such expertise. Wu et al. (2022) scrutinized the hurdles faced by law enforcement in addressing crime in their research on short time crime prediction with human mobility flows and deep learning architectures. One significant concern they addressed is the issue of data bias in historical crime data used for predictive modeling. Key socio-economic factors such as income, unemployment rates, race, and gender are linked to biases in the reporting and recording of crimes, especially when historic crime data is included (Yao et al, 2020), which can impact the training of predictive models. Studies such as Rummens and Hardyns (2021) and May et al. (2021) have shown that models trained on biased data could perpetuate and exacerbate such biases, potentially resulting in discriminatory practices targeting specific demographic groups. Because of this, methodological rigor and strategic feature selection are critical for minimizing bias and deploying ethical models. These challenges underscore the critical importance of mitigating data bias and ensuring algorithmic fairness in predictive policing efforts.

2.5 Machine Learning Algorithms Review for Crime Prediction

The application of Artificial Intelligence and Machine Learning algorithms is increasingly essential for analyzing crime data accurately and predicting future criminal incidents based on crime pattern recognition (Sharma et al., 2021, p. 16). Sharma et al. (2021) conducted a study reviewing algorithms such as Decision tree, Random Forest, Decision tree with PCA (Principal Component Analysis), and Random Forest with PCA for crime prediction in Boston city based on a geospatial dataset. Their findings revealed that applying these models

improved the prediction and classification accuracy by 4-10%. The Decision tree model achieved an accuracy of 53%, while the Random Forest model reached 52% accuracy. Integrating PCA enhanced the accuracy of the Decision tree model to 56%, and the Random Forest model achieved the highest accuracy of 60%. These results were compared with existing implementations, demonstrating enhanced accuracy levels (Sharma et al., 2021, p. 3).

Time series analysis is a popular and rapidly growing topic in crime prediction, especially using techniques like recurrent neural networks (RNNs) and Long-short-term memory (LSTM) techniques. Saradha et al. (2021) reviewed several machine learning models for crime prediction: RNN, LSTM, and Parallel LSTM. The study demonstrated that the Parallel LSTM model achieved the highest validation accuracy of 94%, indicating its superior performance compared to the RNN (66.59%) and traditional LSTM models (72.63%) in predicting crime occurrences. Another study by Stec and Klabjan (2018) showed an increase in prediction accuracy from 65.3% to 75.6% when utilizing RNNs, complemented by CNNs, to predict crime in Chicago and Portland.

ML models like Facebook Prophet (FBProphet) and NeuralProphet were used to predict criminal activities in Little Rock, Arkansas (Sabakhtarishvili et al., 2023). NeuralProphet is a powerful Python library for modeling time-series data on neural networks. Facebook Prophet is a proprietary algorithm used for the same modeling problems using a Bayesian approach. The results indicated that the Facebook Prophet model performed better than NeuralProphet for the dataset analyzed. The Facebook Prophet model achieved lower Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE) values compared to the NeuralProphet model.

Simpler machine learning models often demonstrate their value in crime prediction. Research by Kim et al. (2019) and Babakura et al. (2021) has been pivotal in demonstrating the efficacy of algorithms such as K-nearest-neighbor (KNN), boosted decision trees, and Naïve Bayesian models in urban crime analysis, highlighting their potential in achieving remarkable prediction accuracies. In another study by Raji et al. (2022), two models - Linear Regression and Decision Tree Regression - were evaluated for their effectiveness in predicting crime. For Linear Regression, the researchers compared two models: one considering only local variables and another incorporating national indicators. Both models exhibited a reasonable goodness of fit, with national indicator coefficients proving significant across various cities. Including national indicators notably enhanced prediction accuracy in cities such as Los Angeles, Austin, and Chicago. Similarly, the performance of crime prediction models using Decision Tree Regression improved upon incorporating national indicators.

Using multiple models is a prevalent strategy in applying machine learning for predicting crime. In Alghamdi's (2017) study on predicting burglary crime rates using a data mining approach, various machine learning models such as ridge regression, linear regression, random forest, and gradient boosting were employed. The findings indicated that the Random Forest-based model achieved the highest accuracy compared to the other models, boasting an accuracy rate of 79%. Notably, incorporating new data concerning house characteristics and crime history led to significant enhancements in accuracy. Specifically, including house characteristics data resulted in a 12% accuracy improvement, while integrating crime history data contributed to a substantial 40% increase in accuracy.

Integrating sophisticated neural networks with conventional machine learning models offers potential in creating a comprehensive approach to extracting intricate details from crime

data. While simpler models can be highly effective, the enhanced accuracy of neural networks and other complex artificial intelligence techniques may be necessary in some instances. Ying-Lung et al. (2018) examined various machine learning models for crime prediction, including Naïve Bayes, Ensemble, Deep Learning Structure, and Deep Neural Networks (DNN). Their study revealed that DNN yielded superior results in prior experiments. Through tuning, the researchers identified DNN as the top-performing model, surpassing Random Forest, Support Vector Machine, and K-Nearest Neighbor algorithms. Performance evaluations showed a 7% improvement in F1 score on 100-by-100 grids compared to the baseline 11-month moving average. In a separate study, Roshankar et al. (2023) assessed a range of machine learning models for crime prediction, such as logistic regression, decision trees, and the Spatial-Temporal Meta-path Guided Explainable Crime Prediction (STMEC) model. The research indicated that the graph-based Spatio-Temporal Graph Neural Network (STGNN) exhibited superior accuracy, recall, and F1 scores compared to conventional models, notably outperforming the STMEC model. Specifically, the STGNN model achieved an 82% accuracy rate, with precision at 0.78 and recall at 0.85, indicating balanced positive event prediction. The F1-score, a combined measure of precision and recall, was calculated as 0.81.

Herath et al. (2021), in a crime prediction study using San Francisco Police Incident Report (SFPIR) datasets, reviewed machine learning models such as SVM (Support Vector Machine), LightGBM (Light Gradient Boosting Machine), Random Forest, XGBoost, AdaBoost, Stochastic Gradient Descent, and multi-layer perceptron neural networks with different numbers of layers. The results from the evaluations showed varying testing accuracy and log loss scores for each model, indicating their performance in predicting crime events based on spatial, temporal, and demographic features. For example, the LightGBM model achieved a testing

accuracy of 29.38% with a testing log loss of 2.291, while the Deep Learning models yielded testing accuracies ranging from 22.09% to 25.65% with corresponding log loss scores. Additionally, the TabNet architecture model outperformed other models with a testing accuracy of 41% and a testing log loss of 1.8941, showing improved performance in crime prediction tasks. However, when Synthetic Minority Oversampling Technique (SMOTE) was introduced to train LightGBM and TabNet models, their testing accuracy increased to (32% and 72% for LightGBM) and (46% and 81% for TabNet) in 38 and 4 crime classes, respectively.

In their examination of machine learning models for crime hotspot prediction in Medellin, Columbia, Munoz et al. (2021) assessed Decision Trees, Logistic Regression, and MLP Classifier. The Decision Trees model stood out with an F1-Score of 88.2%, surpassing the performances of Logistic Regression and MLP neural networks, boasting an accuracy of 87.6% and a recall of 90%. A separate study by Yan et al. (2022) delved into predicting multi-class theft crimes using XGBoost, OVR-XGBoost, and OVO-XGBoost models. Their research highlighted that both OVR-XGBoost and OVO-XGBoost models exhibited higher prediction accuracy compared to the baseline XGBoost model. Notably, the OVO-XGBoost model, optimized with the SMOTENN algorithm, highlighted superior performance and accuracy among the models analyzed.

In a study focusing on predicting burglaries based on multiple contextual factors, Solomon et al. (2022) examined a range of machine learning models, including XGBoost, Logistic Regression, SVM, Random Forest, MLP, GRU, LSTM, and the new DeePrison framework. Their research evaluated burglary prediction using datasets from New York City and Israel. In another investigation, Wu et al. (2022) reviewed machine learning models like Historical Average Logistics Regression (HALR), Gated Recurrent Unit (GRU), Attention

(Attn), Graph Convolution Network (GCN), Graph-Graph Convolution (GGConv), and Neighbor Convolution (NbConv). Their results indicated that integrating human mobility flow features with historical crime data (C + M) consistently enhanced short-term crime prediction across different cities and crime types. Incorporating mobility features improved the F1 scores of various neural short-term crime prediction models, resulting in accuracy improvements ranging from 2% to 7%. Additionally, the study suggested that certain neural architectures outperformed regression models by up to 2% in terms of crime prediction performance.

2.6 The Effect of Weather on Crime Prediction

Incorporating weather data into crime prediction models is crucial for understanding the multifaceted influences on criminal activities, as highlighted by Sabakhtarishvili et al. (2023). The researchers emphasized the valuable role of external factors, such as weather conditions, in shaping crime patterns. Weather data sourced from the Little Rock Airport Adams Field weather station via the National Oceanic and Atmospheric Administration's National Centers for Environmental Information (NCEI) database served as a pivotal component in their analysis. Parameters including average wind speed, precipitation, snow depth, snowfall, and maximum and minimum temperatures were integrated as additional regressors in machine learning models, notably Facebook Prophet and NeuralProphet, to enhance their predictive capacity. The inclusion of weather variables enabled a comprehensive examination of how fluctuations in weather elements, such as precipitation, temperature, and wind speed, could impact crime incidents. This amalgamation of weather data culminated in a more sophisticated and accurate forecast of crime patterns, advancing the effectiveness of crime prediction in the Little Rock area (Sabakhtarishvili et al., 2023, p. 14).

In exploring multi-class theft crime prediction, Yan et al. (2022) analyzes the integration of weather conditions as predictive features within theft cases. The study incorporates weather factors, including sunny, cloudy, and rainy conditions, during the data processing phase alongside other predictors such as location, timing, and stolen items to forecast theft types. While the research addresses the inclusion of weather conditions in the predictive model, it does not delve into the specific impact of weather conditions on the accuracy of crime prediction.

The impact of weather on crime prediction is a focal point in the study by Solomon et al. (2022). Weather variables play a significant role as contextual factors in forecasting crime rates. The research encompasses diverse weather characteristics like minimum and maximum temperatures, precipitation levels, and other meteorological data obtained from reliable sources such as the US National Weather Service. By integrating weather data into the predictive framework, the study aims to comprehend the correlations between weather conditions and burglary incidents to augment the precision of crime prediction.

2.7 Summary

In conclusion, the evolution from traditional statistical methodologies to advanced machine learning (ML) and artificial intelligence (AI) techniques has significantly transformed the landscape of crime prediction and predictive policing. By integrating cutting-edge algorithms like Long Short-Term Memory (LSTM) and Gradient Boosting Decision Trees, researchers have been able to enhance the accuracy and efficacy of crime prediction models. Factors such as victim demographics, temporal attributes, weather conditions, crime hotspots, and environmental data have been identified as crucial variables influencing criminal activities in specific geographical locations. The strategic incorporation of these variables has shown promise in improving crime prediction models and informing more efficient crime prevention strategies.

Challenges faced by law enforcement in combating crime include the complexities associated with statistical analysis of crime data, limited resources, and biases in historical crime data that can impact the training of predictive models. Machine learning algorithms have played a vital role in addressing these challenges, with studies highlighting the effectiveness of various models such as Decision Trees, Random Forest, LSTM, and Neural Networks in predicting crime occurrences and improving law enforcement efforts.

The integration of weather data into crime prediction models has emerged as a valuable component for understanding the multifaceted influences on criminal activities. By incorporating weather variables as regressors in machine learning models, researchers have been able to enhance the predictive capacity of these models and provide more accurate forecasts of crime patterns. Overall, the advancement of machine learning algorithms and the inclusion of diverse factors like weather conditions have paved the way for more informed, data-driven approaches to predictive policing, aiming to curtail criminal activities, mitigate economic losses, and enhance community safety and well-being.

Chapter 3: Methodology

3.1 Introduction

This chapter delineates the methodological framework utilized in our investigation, laying the groundwork for a comprehensive understanding of the predictive modeling of criminal behavior in Los Angeles. Initially, we introduce the study population, which serves as the cornerstone of our research. Subsequently, we delve into the intricacies of the data sources employed and elucidate the features and variables that form the bedrock of our analysis. A meticulous process of data preparation, cleaning, and merging is detailed, ensuring a robust foundation for the subsequent application of advanced machine learning algorithms.

Our exploration includes carefully selecting variables guided by their relevance and predictive power. The ensemble of machine learning algorithms chosen for this study is discussed, emphasizing their strengths and potential synergies when applied in concert. Data division into training and testing sets is outlined, adhering to best practices, and adjusted to find the best model performance.

The tools and techniques employed for model training, tuning, and evaluation demonstrate our commitment to methodological rigor. Each step, from initial data handling to the final model evaluation, is crafted to adhere to the highest standards of data science, ensuring that our conclusions are both reliable and actionable when discussed in Chapter 4.

3.2 Study Population

The subject of investigation in this research comprises the residents of Los Angeles, the most populous city in California, with an estimated population exceeding 3.8 million (Census.gov, 2022). Understanding the diverse characteristics of this population is fundamental to gaining insights into crime prediction. Los Angeles, a major hub for commerce, finance, and

culture, exhibits ethnic and cultural diversity, as reflected in its demographic composition. Notably, 41.2% are White alone, 8.6% are Black or African American alone, 1% are American Indian and Alaska Native alone, 11.8% are Asian alone, and 0.1% are Native Hawaiian and Other Pacific Islander alone. Additionally, 12.7% identify with two or more races, 48.1% are Hispanic or Latino, and 28.1% are White alone, not Hispanic or Latino (Census.gov, 2022). The gender distribution is equal, with a median age of 36.2 (Data USA, 2021). Subpopulations include 5.3% under 5 years old, 19.7% under 18 years old, and 13.4% aged 65 years and over (Census.gov, 2022).

Los Angeles, the fourth most visited city in the United States, experienced over 2.7 million visitors in 2022 (WorldAtlas, n.d.). For this study's purposes, we focus solely on the city with its 3.8 million residents, excluding its larger metropolitan area. The city is recognized for its economic significance, being home to the Hollywood Film Industry, and forms part of a metropolitan area with a total of 18 million residents and a GDP of $1 trillion (2018) according to U.S. Bureau of Economic Analysis, ranking as the third-largest city by GDP globally, following New York City and Tokyo (World Population Review, 2024).

As of 2022, Los Angeles County, in which the city is the county seat, boasts an estimated 9.86 million residents. The city, renowned for its Mediterranean climate, has 1.4 million households, averaging 2.7 persons per household. Educational attainment statistics indicate that 78.7% of the population are high school graduates or higher, while 36.7% possess a bachelor’s degree or higher. Median household income stands at $76,244, per capita income at $43,257, and 16.6% of the population lives in poverty. The city experiences a Mediterranean climate with dry summers, mild winters, and infrequent rainfall, characterized by an average of only 35 days of measurable precipitation annually (Weatherbase, n.d.).

Notably, Los Angeles faces challenges related to crime, with a crime rate of 36 per 1,000 residents, higher than many communities across the United States (NeighborhoodScout, n.d.). The city is divided into districts and neighborhoods, aligning with the Los Angeles Police Department's 21 community police stations. The study specifically considers the city's twenty-one geographic areas, each associated with a landmark or surrounding community.

For this research, the Los Angeles Crime dataset from 2010 to 2023 and the corresponding weather dataset will be employed. Crime perpetrator demographics are excluded for privacy considerations, while victim demographics are retained. The study aims to predict spatial-temporal crime hotspots, analyze victim demographics in relation to crime type, investigate daily and temporal patterns of crime, and assess the correlation between crime incidence and typology with weather predictions, providing predictive insights.

3.3 Data Sources

Our analysis is anchored on two critical datasets illuminating the interplay between crime occurrences and weather conditions in Los Angeles. The first dataset comprises Los Angeles Crime Data, derived from LAPD crime reports and made accessible through the city's Open Data portal. The second dataset is historical weather data sourced from Weather Underground, leveraging a global network of weather stations to provide comprehensive climatic insights.

3.3.1 Los Angeles Crime Data

3.3.1.1 Overview

The Los Angeles Crime Data is a rich repository of crime incident records, sourced from LAPD crime reports. These records offer firsthand accounts of various incidents, painting a detailed picture of the crime landscape in Los Angeles from 2010 to the present. The dataset

categorizes crimes based on their severity, with Crime Code 1 representing the most serious offenses, followed by less severe classifications for Crime Codes 2 through 4.

3.3.2 Data Collection Process

The dataset is built on individual crime reports by LAPD officers, capturing essential details such as the incident's date, time, and location, alongside the nature of the crime and victim demographics. Missing location data is marked with (0°, 0°), and missing demographic data is left blank or labeled as unknown. The collection process occurs from 2010 to 2023, and the total dataset consists of 2,886,530 records and 28 variables.

3.3.3 Data Accuracy and Privacy Considerations

While the transition from paper-based reports to a digital dataset enhances accessibility, it introduces potential for minor inaccuracies. Moreover, to safeguard individual privacy, exact addresses within the dataset are obfuscated, presented only to the nearest hundred blocks. This obfuscation is intentional to help maintain victim confidentiality in the dataset. Other location variables such as latitude, longitude, and area, are available in the dataset and are missing less data.

3.3.4 Data Accessibility and Update Frequency

The dataset's maintenance involves weekly updates, with occasional bi-weekly intervals, ensuring its currency and relevance. Available in various formats (CSV, Excel, RDF, RSS, XML, TSV for Excel), the dataset caters to a wide range of technical preferences, ensuring broad accessibility for diverse research endeavors.

3.3.5 Weather Data

3.3.5.1 Data Collection

Weather Underground's dataset is compiled from many sources, including airport-based Automated Surface Observation Systems and Personal Weather Stations. This diverse collection mechanism ensures a rich dataset that captures detailed weather conditions across Los Angeles. The data was manually collected by scraping monthly data from Weather Underground (2024), requiring 144 separate data pulls to concatenate chronologically into one dataset for use in the study.

3.3.5.2 Variables

The dataset provides weather parameters, such as temperature, humidity, precipitation levels, and dewpoint. These variables offer a detailed portrayal of the climatic conditions in Los Angeles. The variables are further divided into minimum, maximum, and average. Year and month are added manually as they are not inherently present to allow for merging with the crime dataset.

3.3.5.3 Spatiotemporal Coverage

Our analysis focuses on Los Angeles, specifically leveraging weather data from Los Angeles International Airport Station and crime data spanning the same period, 2010 to 2023. This congruence ensures that our examination of climatic conditions and crime rates are aligned. The dataset encompasses 5,113 rows and 17 variables in its final state.

3.3.5.4 Data Quality and Accessibility

The integrity of the weather data is maintained through stringent quality controls and regular updates, reflecting Weather Underground's commitment to data precision. For our study,

the data was systematically compiled monthly into Excel, subsequently merged, and converted into a CSV format, exemplifying our meticulous approach to data preparation.

3.4 Features

3.4.1 LA Crime Data Features

Los Angeles Crime Data has 28 features. These variables represent details of crime reports and are of various data types, some of which will be converted. The following is a breakdown of what each feature represents.

3.4.2 Division of Records Number

Denoted by DR_NO, this is an official file number that is made up of a two-digit year, area ID, and five digits. The data type will be converted to a nominal categorical variable. There are 2,886,530 records in the dataset of which 2,557,558 records are distinct values, representing 88.6% of the data. There are no missing values.

3.4.3 Date Reported

This is the date that the crime was reported. It is denoted by Date_Rptd and will be converted to datetime format (YYYY-MM-DD). There are 2,886,530 records in the dataset of which 5,129 records are distinct values, representing 0.18% of the data. There are no missing values.

3.4.4 Date Occurred

This is the date that the crime occurred. DATE_OCC denotes it and will be converted to DateTime format (YYYY-MM-DD). There are 2,886,530 records in the dataset, of which 5,113 records are distinct values, representing 0.18% of the data. There are no missing values.

3.4.5 Time Occurred

This is the time that the crime occurred. It is denoted by TIME_OCC and is represented in 24-hour military time. There are 2,886,530 records in the dataset of which 1,439 are distinct values, representing 0.05% of the data. There are no missing values.

3.4.6 Area

Area represents the geographical area where the crime took place. The Los Angeles Police Department has 21 community police stations referred to as geographic areas or patrol divisions within the department. These geographic areas are sequentially numbered from 1 to 21. Denoted by AREA, this is a categorical variable. There are 2,886,530 records in the dataset and only 21 distinct values, representing 0.0007% of the data. There are no missing values.

3.4.7 Area Name

'Area Name' refers to the name designation of 'AREA.' The name designation references a landmark or surrounding community that it is responsible for. An example is 77th Street Division, located at the intersection of South Broadway and 77th Street, serving neighborhoods in South Los Angeles. It is denoted by AREA_NAME and is a nominal categorical variable. There are 2,886,530 records in the dataset and 21 distinct values representing 0.0007% of the data. There are no missing values.

3.4.8 Report District Number

This is a four-digit code that represents a sub-area within a geographical area. It is denoted by Rpt_Dist_No and references the reporting district that the crime occurred in. This is a nominal categorical variable. There are 2,886,530 records in the dataset of which 1,308 are distinct values, representing 0.045% of the data. There are no missing values.

3.4.9 Part 1-2

This variable refers to the nature of the offence committed, whether they are Part 1, or Part 2. Part 1 refers to serious crimes, and there are two types of Part 1 offences namely violent crimes and property crimes. Part 2 refers to less serious or non-violent crimes. According to the Uniform Crime Reporting (UCR) program of the Criminal Justice Information Services Division of the Federal Bureau of Investigation (FBI), Part 1 offences include criminal homicide, rape, robbery, aggravated assault, burglary, larceny-theft (except motor vehicle theft), motor vehicle theft, arson, human trafficking – commercial sex acts, human trafficking – involuntary servitude. Part 2 offences encompass all other reportable classifications outside those defined as Part 1. The variable is denoted by Part_1-2 and is categorical. There are 2,886,530 records in the dataset and only 2 distinct values (1 or 2), representing 0.00007% of the data. There are no missing values.

3.4.10 Crime Code

Crime code indicates the crime committed. It is the same as Crime Code 1, another feature of the LA crime data. Denoted by Crm_Cd, a three-digit code represents it and is a nominal categorical variable. There are 2,886,530 records in the dataset and 144 distinct values, representing 0.005% of the data. There are no missing values.

3.4.11 Crime Code Description

This defines the crime code provided. It describes the crime committed and is denoted by Crm_Cd_Desc. The data type is a character variable. There are 2,886,530 records in the dataset and 144 distinct values, representing 0.005% of the data. There are no missing values.

3.4.12 Modus Operandi Codes (mocodes)

Modus Operandi Codes (mocodes) are four-digit codes that refer to the activities associated with the suspect in the commission of the crime. This is a nominal categorical variable and is denoted by mocodes. There are 626,550 distinct values, representing 21.7% of the data. This implies that there are 626,550 distinct activities associated with the suspect at the incident of the crime, highlighting its variability. 330,174 values are missing, representing 11.44% of the data.

3.4.13 Victim Age

This refers to the age of the victim of the crime. It is a 2-digit numeric variable and is denoted by Vict_Age. It is part of the victim demographics in our dataset. There are 2,886,530 records and 113 distinct values, representing 0.004% of the data. There are no missing values. Because there are 113 distinct values for 2-digit records, we can already sense that there are some errors in the records, hence we would need to conduct data integrity checks and clean our data properly. This is highlighted in our data preparation and cleaning section.

3.4.14 Victim Sex

Victim Sex refers to the victim's gender of the crime and has three entries: Female = F, Male = M, or Unknown = X. It is a nominal categorical variable and is denoted by Vict_Sex. It is part of the victim demographics. There are seven distinct values, representing 0.00024% of the data. Data integrity checks will be conducted as there should be only three distinct values. This is discussed in the data preparation and cleaning section. 291,361 values are missing, representing 11.44% of the data.

3.4.15 Victim Descent

This refers to the race or ethnicity of the victim and is also a part of the victim demographics of our crime dataset. It is a one-letter code representing the diverse racial profile of Los Angeles residents. The 19 descendant codes are A - Other Asian, B – Black, C – Chinese, D – Cambodian, F – Filipino, G – Guamanian, H - Hispanic/Latin/Mexican, I - American Indian/Alaskan Native, J – Japanese, K – Korean, L – Laotian, O – Other, P - Pacific Islander, S – Samoan, U – Hawaiian, V – Vietnamese, W – White, X – Unknown, Z - Asian Indian. Victim Descent is a nominal categorical variable and is denoted by Vict_Descent. There are 21 distinct values, representing 0.0007% of the data. 291,418 values are missing, representing 10.1% of the data.

3.4.16 Premise Code

Premise code is a three-digit code that refers to the type of premises, building, structure, vehicle, or location that the crime took place in. It is a categorical variable and is denoted by Premis_Cd. There are 330 distinct values, representing 0.01% of the data. 73 values are missing, representing 0.0025% of the data.

3.4.17 Premise Description

This defines the premise code provided. It describes the premises or location where the crime was committed (e.g., Hotel, Sidewalk, Department Store etc.) and is denoted by Premis_Desc. The data type is a character variable. There are 321 distinct values, representing 0.01% of the data. 669 values are missing, representing 0.023% of the data.

3.4.18 Weapon Used Code

This is a three-digit code that refers to the type of weapon used in the crime. It is a categorical variable and is denoted by Weapon_Used_Cd. There are 81 distinct values, representing 0.0028% of the data. 1,909,507 values are missing, representing 66.15% of the data.

3.4.19 Weapon Description

This defines the weapon used code provided. It describes the weapon(s) used in the crime (e.g., Handgun, Strong-arm, Folding Knife, etc.) and is denoted by Weapon_Desc. The data type is character variable. There are 80 distinct values, representing 0.0028% of the data. 1,909,508 values are missing, representing 66.15% of the data.

3.4.20 Status

This refers to the status of the case, whether investigation is ongoing/continuing, or the case has been closed/resolved. Status denotes it; entries are two letter words such as IC, AO, AA. IC is the default entry. There are 10 distinct values, representing 0.00035% of the data. 4 values are missing, representing 0.00014% of the data.

3.4.21 Status Description

This defines the status code provided. It describes the status of the case (e.g., Adult Other, Invest Cont, Adult Arrest, etc.) and is denoted by Status_Desc. The data type is a character variable. There are 6 distinct values, representing 0.0002% of the data. There are no missing values.

3.4.22 Crime Code 1

This indicates the crime that was committed. Crime Code 1 is the primary and most serious one. The lower the crime class number, the more serious the crime. It is a three-digit code denoted by Crm_Cd_1 and is a nominal categorical variable. There are 151 distinct values, representing 0.005% of the data. 20 values are missing, representing 0.0007% of the data.

3.4.23 Crime Code 2

This indicates the crime that was committed. It is usually present when an additional crime is committed, less severe than Crime Code 1. It is a three-digit code denoted by Crm_Cd_2 and is a nominal categorical variable. There are 146 distinct values, representing 0.005% of the data. 2,693,306 values are missing, representing 93.3% of the data.

3.4.24 Crime Code 3

This indicates the crime that was committed. It is usually present when an additional crime is committed, less severe than Crime Code 1 and Crime Code 2. It is a three-digit code denoted by Crm_Cd_3 and is a nominal categorical variable. There are 65 distinct values, representing 0.002% of the data. 2,881,468 values are missing, representing 99.8% of the data.

3.4.25 Crime Code 4

This indicates the crime that was committed. It is usually present when an additional crime was committed, less severe than Crime Code 1, 2 and 3. It is a three-digit code denoted by Crm_Cd_4 and is a nominal categorical variable. There are 13 distinct values, representing 0.00045% of the data. 2,886,379 values are missing, representing 99.99% of the data.

3.4.26 Location

This refers to the street address of the crime incident. It is rounded to the nearest hundredth block to maintain anonymity for data privacy concerns. It is a character variable and is denoted by LOCATION. There are 80,302 distinct values, representing 2.78% of the data. There are no missing values.

3.4.27 Cross Street

This refers to the cross street of the rounded address where the crime occurred. It is the closest street that intersects the rounded address or location of the crime. It is a character variable and is denoted by Cross_Street. There are 14,336 distinct values, representing 0.5% of the data. 2,412,813 values are missing, representing 83.6% of the data.

3.4.28 Latitude

This measures the distance north or south of the equator. Latitude lines circle the planet from east to west. It pinpoints the geographic location of the crime in conjunction with longitude. It is a continuous numerical variable and is denoted by LAT. There are 5,635 distinct values, representing 0.2% of the data. There are no missing values.

3.4.29 Longitude

This measures the distance east or west of the prime meridian. They run north to south from pole to pole but measure the distance east or west. Longitude is measured in degrees, minutes, and seconds. It pinpoints the geographic location of the crime in conjunction with latitude. It is a continuous numerical variable and is denoted by LON. There are 5,132 distinct values, representing 0.18% of the data. There are no missing values. Table 4 summarizes the characteristics of the features in the LA Crime data.

Table 4

Features of the 2010-2023 Los Angeles Crime Dataset

Variable	Full Variable Name	Data Type	No of Records	No of Distinct Values	Distinct Values (%)	No of Missing Values	Missing Values (%)
DR_NO	Division of Records Number	string	2886530	2557558	88.60%	0	0.00%
Date_Rptd	Date Reported	date	2886530	5129	0.18%	0	0.00%
DATE_OCC	Date Occurred	date	2886530	5113	0.18%	0	0.00%
TIME_OCC	Time Occurred	string	2886530	1439	0.05%	0	0.00%
AREA	Area	string	2886530	21	0.00%	0	0.00%
AREA_NAME	Area Name	string	2886530	21	0.00%	0	0.00%
Rpt_Dist_No	Report District Number	string	2886530	1308	0.05%	0	0.00%
Part_1-2	Part 1-2	number	2886530	2	0.00%	0	0.00%
Crm_Cd	Crime Code	string	2886530	144	0.00%	0	0.00%
Crm_Cd_Desc	Crime Code Description	character	2886530	144	0.00%	0	0.00%
Mocodes	Mocodes	string	2556356	626550	21.71%	330174	11.44%
Vict_Age	Victim Age	number	2886530	113	0.00%	0	0.00%
Vict_Sex	Victim Sex	string	2595169	7	0.00%	291361	10.09%
Vict_Descent	Victim Descent	string	2595112	21	0.00%	291418	10.10%
Premis_Cd	Premise Code	string	2886457	330	0.01%	73	0.00%
Premis_Desc	Premise Description	string	2885861	321	0.01%	669	0.02%
Weapon_Used_Cd	Weapon Used Code	string	977023	81	0.00%	1909507	66.15%
Weapon_Desc	Weapon Description	string	977022	80	0.00%	1909508	66.15%
Status	Status	string	2886526	10	0.00%	4	0.00%
Status_Desc	Status Description	string	2886530	6	0.00%	0	0.00%
Crm_Cd_1	Crime Code 1	string	2886510	151	0.01%	20	0.00%
Crm_Cd_2	Crime Code 2	string	193224	146	0.01%	2693306	93.31%
Crm_Cd_3	Crime Code 3	string	5062	65	0.00%	2881468	99.82%
Crm_Cd_4	Crime Code 4	string	151	13	0.00%	2886379	99.99%
LOCATION	Location	string	2886530	80302	2.78%	0	0.00%
Cross_Street	Cross Street	string	473717	14336	0.50%	2412813	83.59%
LAT	Latitude	float	2886530	5635	0.20%	0	0.00%
LON	Longitude	float	2886530	5132	0.18%	0	0.00%

3.4.30 LA Weather Data

Los Angeles Weather Data has 17 features. These features can be classified in seven (7) groups. Apart from Date and Total Precipitation, the other five groups of features each are divided into three (3) subgroups: Minimum, Average, and Maximum. They are Temperature, Dewpoint, Humidity, Windspeed, and Pressure. These variables represent details of weather recorded from January 1, 2010, to December 31, 2023. The following is a breakdown of what each group of features represents.

3.4.31 Date Occurred

This refers to the date that the weather data was recorded. It is denoted by DATE_OCC and will be converted to datetime format (YYYY-MM-DD). There are 5,113 records in the dataset all of which are distinct values, representing 100% of the data. There are no missing values.

3.4.32 Temperature – Maximum, Minimum, Average

This is a measure of hotness or coldness. It is denoted by Max_Temp, Avg_Temp, and Min_Temp. Each of these three values are continuous numeric variables. There are 53 distinct values, representing 0.0018% of the data. There are no missing values.

3.4.33 Dewpoint – Maximum, Minimum, Average

This is the temperature to which the air needs to be cooled for relative humidity to reach 100% (when a cloud would form). It is denoted by Max_Dewpoint, Avg_Dewpoint, and Min_Dewpoint. Each of these three values are continuous numeric variables. There are 53 distinct values, representing 0.0018% of the data. There is 1 missing value, representing 0.02% of the data.

3.4.34 Humidity – Maximum, Minimum, Average

Relative humidity is the ratio of water vapor contained in the air to the maximum amount of water vapor that can be contained in the air at the current temperature. It is denoted by Max_Humidity, Avg_Humidity, and Min_Humidity. Each of these three values are continuous numeric variables. There are 74 distinct values, representing 0.0026% of the data. There is 1 missing value, representing 0.02% of the data.

3.4.35 Windspeed – Maximum, Minimum, Average

Wind speed is the distance covered by air at a particular time. It is denoted by Max_Windspeed, Avg_Windspeed, and Min_Windspeed. Each of these three values are continuous numeric variables. There are 35 distinct values, representing 0.0012% of the data. There is 1 missing value, representing 0.02% of the data.

3.4.36 Pressure – Maximum, Minimum, Average

Pressure is defined as the force per unit area of the weight of air above an object. It is denoted by Max_Pressure, Avg_Pressure, and Min_Pressure. Each of these three values are continuous numeric variables. There are 12 distinct values, representing 0.0004% of the data. There is 1 missing value, representing 0.02% of the data.

3.4.37 Total Precipitation

Total precipitation is the total amount of rainfall or snow that falls from the clouds toward the ground. It is denoted by Total_Precipitation and is a continuous numeric variable. There are 99 distinct values, representing 0.003% of the data. There is 1 missing value, representing 0.02% of the data. Table 5 summarizes the characteristics of the features in the LA Weather data.

Table 5

Features of the L.A. Weather Dataset

Variable	Full Variable Name	Data Type	No of Records	No of Distinct Values	Distinct Values (%)	No of Missing Values	Missing Values (%)
DATE_OCC	Date Occurred	date	5113	5113	100.00%	0	0.00%
Max_Temp	Maximum Temperature	float	5113	53	1.04%	0	0.00%
Avg_Temp	Average Temperature	float	5113	53	1.04%	0	0.00%
Min_Temp	Minimum Temperature	float	5113	53	1.04%	0	0.00%
Max_Dewpoint	Maximum Dewpoint	float	5113	53	1.04%	1	0.02%
Avg_Dewpoint	Average Dewpoint	float	5113	53	1.04%	1	0.02%
Min_Dewpoint	Minimum Dewpoint	float	5113	53	1.04%	1	0.02%
Max_Humidity	Maximum Humidity	float	5113	74	1.45%	1	0.02%
Avg_Humidity	Average Humidity	float	5113	74	1.45%	1	0.02%
Min_Humidity	Minimum Humidity	float	5113	74	1.45%	1	0.02%
Max_Windspeed	Maximum Windspeed	float	5113	35	0.68%	1	0.02%
Avg_Windspeed	Average Windspeed	float	5113	35	0.68%	1	0.02%
Min_Windspeed	Minimum Windspeed	float	5113	35	0.68%	1	0.02%
Max_Pressure	Maximum Pressure	float	5113	12	0.23%	1	0.02%
Avg_Pressure	Average Pressure	float	5113	12	0.23%	1	0.02%
Min_Pressure	Minimum Pressure	float	5113	12	0.23%	1	0.02%
Total_Precipitation	Total Precipitation	float	5113	99	1.94%	1	0.02%

3.5 Data Preparation

Building on the foundational datasets outlined in Section 3.3 and to preprocess our data for analysis, this segment delves into the critical stages of data preparation, cleaning, and merging.

3.5.1 Variable Name Standardization

As part of the data preparation process for both datasets, we aimed to standardize the variable names by introducing underscores to replace spaces. This was done to ensure consistency with Python naming conventions and to enhance the readability and accessibility of the dataset for both data manipulation and analysis. Variables containing two or more words, such as 'Date Rptd' and 'Area Name', were transformed into 'Date_Rptd' and 'Area_Name', respectively.

3.5.2 Crime Data Preparation

The initial stage of preparing the crime datasets for analysis involved two distinct datasets: one covering the period from 2010 to 2019, and the other, from 2020 to 2023. Both datasets contain the same column names and number (28), encapsulating various aspects of each crime incident reported in Los Angeles.

3.5.3 Crime Data from 2010 to 2019

The dataset spanning 2010 to 2019 consists of 2,122,469 records, each representing a unique crime incident. The data includes a range of information, from the division of records number (DR_NO) and the date the crime was reported (Date Rptd), to more detailed fields such as the crime code (Crm Cd) and the description of the crime (Crm Cd Desc). Victim information, including age (Vict Age), sex (Vict Sex), and descent (Vict Descent), provide demographic insights into the affected individuals. Location details are given through the area name (AREA NAME), reporting district number (Rpt Dist No), and precise latitude (LAT) and longitude (LON) coordinates, ensuring a comprehensive geographical mapping of crime incidents.

During the initial review of the dataset covering the period from 2010 to 2019, it was observed that the "AREA" column name included a trailing space, rendering it as "AREA "; thus, the extra space was removed. Such discrepancies in column naming conventions could potentially lead to complications during the data merging, especially when columns from different datasets are intended to align precisely.

3.5.4 Crime Data from 2020 to 2023

Our updated dataset covers 2020 to 2023 and includes over 760,000 records. It has the same format as our previous dataset, making it seamless to combine the data and work with both datasets. The dataset from 2020 to 2023 was then reviewed to ensure that no similar issues were present, confirming that the "AREA" column was correctly named without any trailing spaces, thereby aligning with the adjusted column name in the 2010-2019 dataset.

3.5.5 Crime Data Concatenation

Integrating the crime data from two distinct periods, 2010-2019 and 2020-2023, was the first necessary step for analytical preparation. This merger was executed to create a single, cohesive dataset that would allow for a continuous and comprehensive analysis of crime trends over fourteen years. Before merging, both datasets were closely examined to ensure uniformity in structure. This involved confirming that the data types were consistent across both datasets and that the column names were precisely aligned, including the removal of any trailing spaces.

The merging process used a concatenation function within the Python Pandas data manipulation library, allowing the datasets to be combined end-to-end. Special attention was paid to maintaining the integrity of the index to prevent any misalignment of records. After merging the datasets, checks were conducted to validate the operation's success. The total

number of rows in the merged dataset matched the sum of the individual datasets, amounting to 2,886,530 entries, and the number of columns remained unchanged at 28.

3.5.6 Time Stamp Reformatting

As part of the data synchronization process between the crime and weather datasets, it is important to align the DATE_OCC column from the crime dataset with the corresponding date column in the weather dataset. This step ensures that the date of occurrence in the crime data matches precisely with the date in the weather data, allowing for an accurate merger.

The DATE_OCC variable, which initially contains both date and time information, must be converted to a date-only data type. This conversion simplifies the data and aligns it with the weather dataset, which contains date information without the time. The time component of the DATE_OCC variable must be removed. This involves truncating the time from the DateTime object or extracting just the date part. The goal is to ensure that the variable focuses solely on the date, as the time stamp is not required for the intended date-by-date comparison with weather data.

3.5.7 Weather Dataset Preparation

As presented on the website, the original dataset included comprehensive meteorological data for each day, structured into columns such as Temperature, Dew Point, Humidity, Wind Speed, Pressure, and Precipitation. Each of these columns was further broken down into subcategories indicating the day's maximum, average, and minimum readings. As each month of weather data was downloaded separately and concatenated chronologically, the years were manually added, and the day column required manipulation into a proper DateTime data type labeled “DATE_OCC” to match the crime dataset.

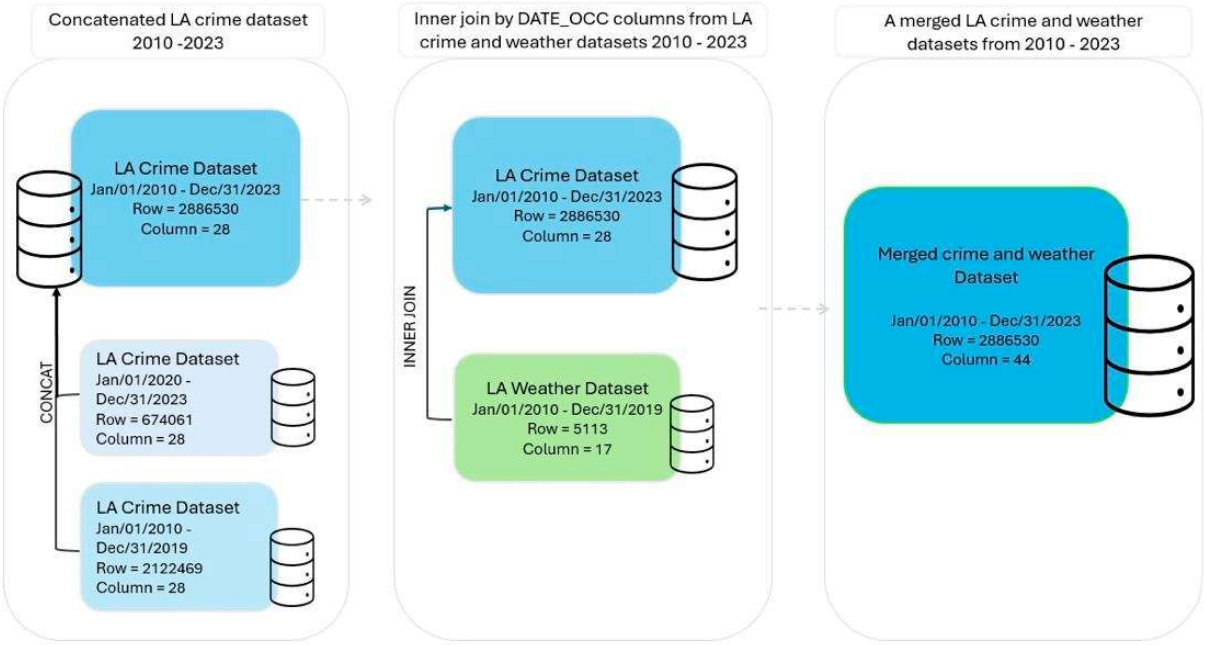
3.5.8 Weather and Crime Data Merge

Before the data merge, the 'DATE_OCC' column in both datasets was formatted identically as the date without timestamps. This alignment was crucial as it is the primary key that links weather conditions to the corresponding dates of the crime incidents.

For merging we are using Python's Pandas library, an inner join was performed to combine the datasets. This method retains records matching both datasets, ensuring that every available day reporting crime is aligned with weather data. Figure 1 shows a diagram illustrating the concatenation and merge of all datasets to the final, comprehensive dataset.

Figure 1

Crime and Weather Data Merging Diagram 2010–2023.



3.5.9 Data Cleaning

3.5.9.1 Missing Values

In the comprehensive study of crime data spanning from 2010 to 2023, our initial step involved a thorough data cleaning process to ensure the integrity and usability of the dataset for

proper analyses. This multifaceted process began with a thorough assessment of the dataset, after all three datasets were merged, to identify and rectify issues such as missing values, outliers, and inaccuracies that could potentially skew results or cause models to train poorly.

As discussed in section 3.4, several variables contained significant amounts of missing values. Some of these variables were deemed non-critical to the analysis and were removed altogether. Due to missing well over 90 percent of their values, crime codes 2, 3, and 4 were removed. Next, cross street is missing 83.6 percent of its values and was deemed necessary to remove, as there are more precise location variables to use that do not have as much missing data, such as latitude and longitude. Other variables such as premise code, premise description, crime code 1, and status were missing smaller amounts of values, so the missing rows were removed for these.

The weapon used code and weapon description variables were missing 66 percent of their values, but they could provide valuable crime insights. For this reason, weapon used code was turned into a binary variable known as weapon reported, where 1 indicates a weapon being used, and 0 indicates the absence of a weapon. Having information on weapons during a crime analysis can provide practical insights into how offenses are committed.

The original weather dataset contained one row missing all parametric data except temperature. The issue becomes slightly exacerbated when merging into the larger dataset, increasing the missing values of these variables to 79. Upon further investigation, on November 20th, 2020, these weather parameters were not recorded, and 79 crimes were committed that day. A similar issue appears with the total precipitation feature in the weather dataset on April 4th, 2020, creating 144 missing values in the merged dataset. Due to the mostly stable weather in Los Angeles and the small number of missing values, all features had missing values imputed using their median values.

The research question suggesting that victim demographics play a role in crime prediction prods us to minimize as much data loss as possible with these features. To maintain data integrity, missing values for victim sex and descent were imputed as 'unknown.' Modus Operandi codes are missing just over 10 percent of values and are also imputed as 'unknown'.

Last, according to the data card for the LAPD dataset, values of 0 for latitude and longitude are the equivalent of missing values. The location of latitude and longitude equaling 0 is appropriately referred to as "Null Island" and appears to be a common issue in data collection. With 1,907 total entries of both values being 0, these rows are removed to prevent issues when performing crime hotspot predictions.

3.5.9.2 Outliers

Outliers can be more appropriately addressed now that all missing data is accounted for in the combined dataset. There are apparent discrepancies when observing the descriptive statistics of the numeric values in the dataset. For example, temperatures of close to 140 and 160 degrees are present. While this may seem like a data entry issue, the highest recorded temperature in Los Angeles was 135 degrees in Beverly Hills during 2017, according to The Los Angeles Almanac (n.d), potentially validating the near 140-degree temperature observation but not the 160-degree temperature. These values are far beyond the normal interquartile range for temperature, and their respective rows are removed to maintain data integrity.

Other observed outliers include an entry for wind speed of over 800 miles per hour. This is a straightforward data entry issue where an extra 0 was added to the value as an 80-mile-per-hour seems far more feasible during severe weather. In addition, the average maximum wind speed is around 13 miles per hour, which may suggest the 800-mile-per-hour entry was intended to be 8 miles per hour. The action deemed necessary to address this data anomaly is to drop the observation. Additional anomalies include a minimum temperature and pressure of 0 and a

maximum dewpoint over 140 degrees Fahrenheit. One final issue is victim ages; some victim ages are negative while others are close to 120 years of age. Due to the small population of supercentenarians, while possible, their presence in the dataset seems unlikely. Rows with age outliers and negative ages are few and, therefore dropped.

3.5.9.3 Feature Engineering

Due to many classes being present within the target variable and their apparent imbalances, the deemed necessary approach was to group related crimes into a new categorical feature known as the crime category code. The resulting categories are assaults, battery, and violent crimes, thefts and burglaries, vandalism and property damage, sexual offenses, robbery and extortion, court violations, fraud and financial crimes, and all other crimes. Several crimes had few reports and balancing each class through weights or oversampling proved too computationally expensive to be practical. The resulting class, the crime category code, has eight categories representing generalized and common crime types.

The same issue presents itself with victim descents. Some ethnic origins are grouped in a general fashion, such as white, black, and Hispanic. At the same time, the observed Asian population is separated down to the individual country such as Guamanian, Vietnamese, Chinese, Japanese, Korean, Cambodian, and Laotian. To create a better balance between the victim descent categories, all Asian countries were combined in a comparable way as black and white, which is simply the category Asian. Other descent classes, such as American Indian/Alaskan Native, Hawaiian, and Samoan, had very few observations and were listed as "Other." The resulting feature is listed as region of ethnic origin and contains six categories instead of the previous 20. Additionally, after removing outliers, victim ages are categorized into those under 18 (minors), between 18 and 65 (adults) and 65 or older (seniors).

Minimizing the number of specific and niche classes allows machine learning algorithms and neural networks to train more efficiently while subjecting them to much less class imbalance. With such a large dataset and so many categories, feature reduction and engineering are a practical way to reduce model complexity and hopefully reduce overfitting due to the more general nature of the modified data.

3.6 Variable Selection

The choice of variables in our study are carefully selected after preparing and merging the data to best support our research hypotheses, ensuring that each feature chosen offers valuable insights into crime prediction in Los Angeles. The hypothesis posits that victim demographics, weather conditions, and temporal patterns influence crime occurrences. Consequently, we select variables that best represent these domains.

In our methodology, the primary variable of interest, crime code, is categorical and reduced to eight categories from 144, represented by numerical codes from 0 to 7. Each code correlates with a group of similar crimes, which conveys the crime's description. The crime category code category is the target variable for two of three research hypotheses: victim demographic and weather’s influence on crime in Los Angeles.

From the crime data, we prioritize the victim ages, sex, and region of ethnic origin, which encapsulate the demographic profile of victims present in the data, relating to our hypothesis regarding the role of victim demographics in crime prediction. These variables are expected to offer a nuanced understanding of crime patterns when analyzed alongside variables Part 1-2, and weapon reported, as well as the date reported, date of occurrence, time of occurrence, and area, which provide temporal and spatial dimensions to the incidents, thus addressing our third

hypothesis about crime hotspots and their prediction using time series analysis (Bhadri, 2023, p. 13).

For weather influences, as we merge the crime data with our weather dataset, drawing on date of occurrence as the key linking variable, average temperature, humidity, wind speed, and total precipitation from the weather data are selected for their potential impact on crime occurrence, reflecting our second hypothesis. Their respective maximum and minimum variables are dropped to avoid multicollinearity. Average weather metrics are anticipated to correlate with the frequency and type of crimes, (Sabakhtarishvili et al, 2023) and improve the predictive model's accuracy.

Given the extensive nature of the datasets, variables with high percentages of missing data, such as cross street, crime codes 2,3, and 4, are excluded from the initial model to maintain data integrity and analytical precision. Other variables are removed due to overall irrelevance to the three research questions, collinearity to existing variables, or feature engineering redundancy. These removed variables include crime code, crime code description, status, status description, victim descent, victim age, weapon used code, weapon description, mocodes, premise code, and premise description.

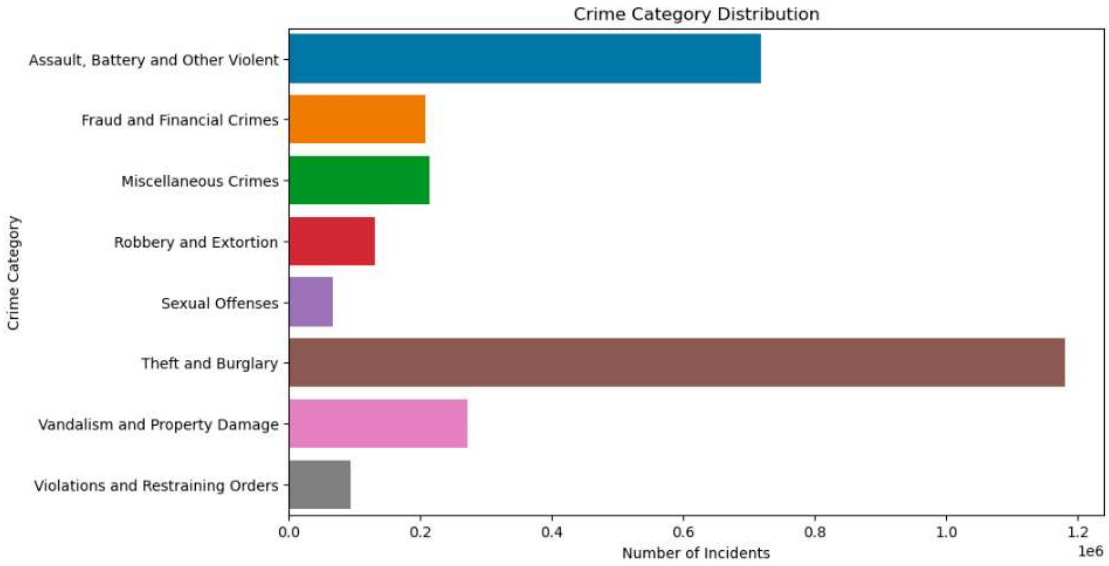
For the final research hypothesis on predicting future crime hotspots in Los Angeles, the feature selection mostly remains the same, with the inclusion of features such as area, report district number, time of day, day of the week, month, and year. The target variable aggregates crime category codes to total crimes by area predicted by time of day, day of the week, month, and year to identify seasonality trends in complement to the prediction variables used in the other two research hypotheses.

3.7 Exploratory Data Analysis

This section explores key features of our clean dataset and displays their distribution characteristics. 144 different crime categories present in the original dataset, are reduced to eight (8), because of the imbalance of categories and potential classification challenges of models. Figure 2 shows the class distribution of the newly grouped crime category codes.

Figure 2

Crime Category Distribution (Target Variable)



The Los Angeles crime dataset contains 21 unique areas. Figure 3 shows a bar graph with all categorical areas and their respective crime amounts. The graph is arranged in descending order from left to right, showing crime distribution by area in a concise way.

Figure 3

Crimes by Area from 2010-2023

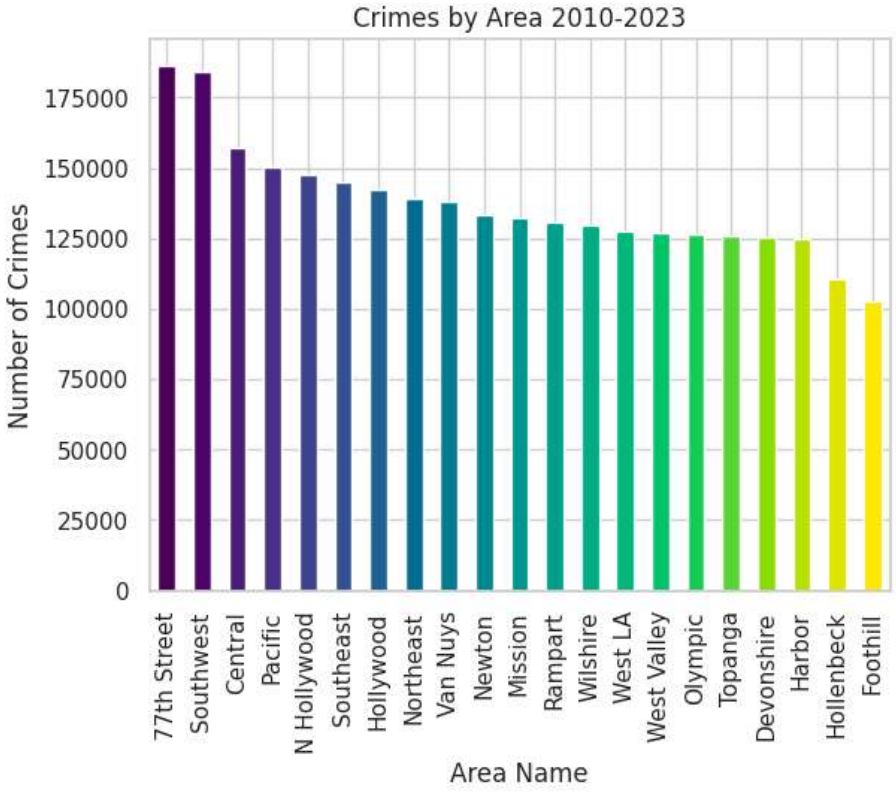


Figure 4 shows the distribution of crimes by year. The number of crimes per year is relatively stable, with notable declines from 2020-2022 and a large surge of crimes in 2023. Normal years range between 200,000 and 300,000 crimes per year.

Figure 4

Number of Crimes by Year

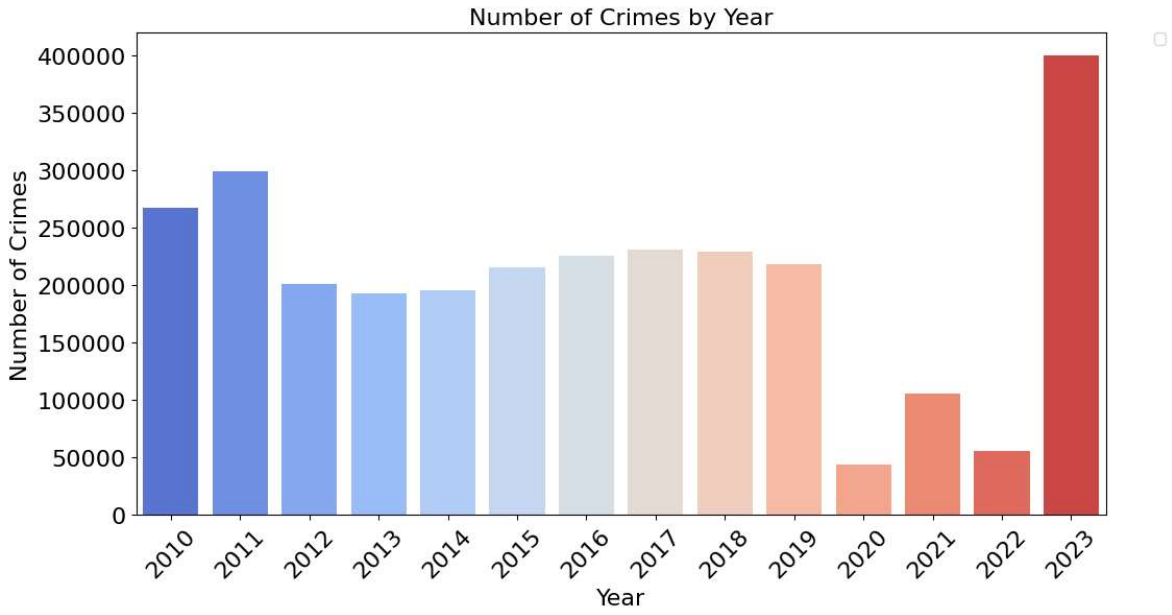


Figure 5 shows a colored plot of crime category codes plotted by latitude and longitude across Los Angeles. Each plot shows the location of crime reported by the LAPD.

Figure 5

Crime Spread Plot by Latitude and Longitude

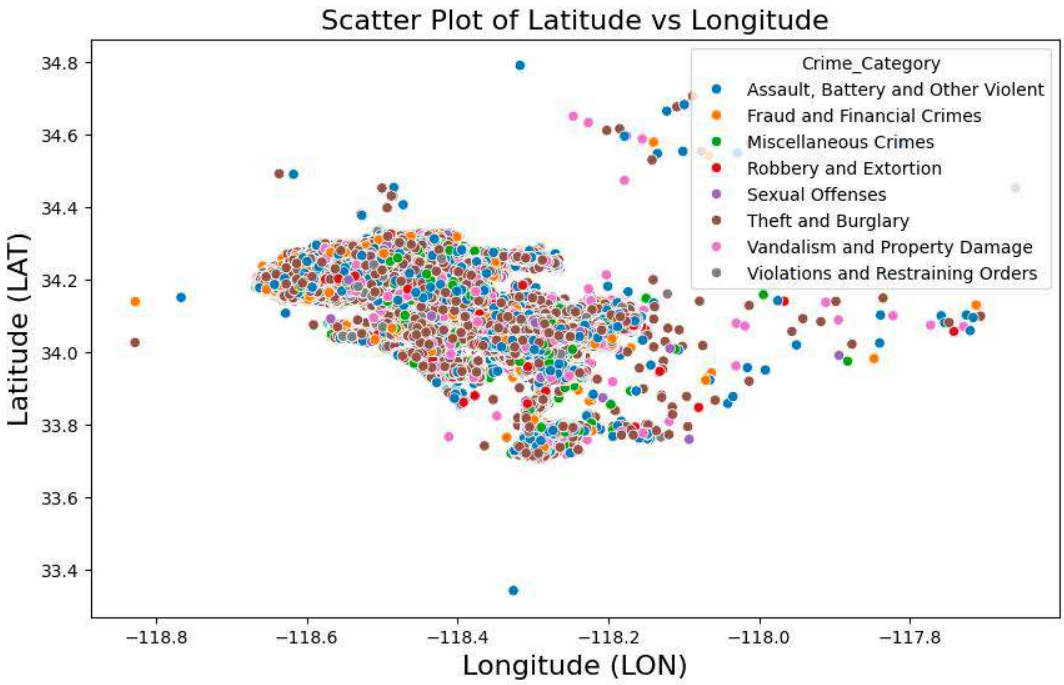


Figure 6 shows the average type of weather in Los Angeles from 2010 to 2023. The noisy sinusoidal patterns are in lockstep with seasonal changes apart from total precipitation, showing large surges of precipitation amounts. They do appear to happen closer to the middle of each year after 2013. Overall, the weather in Los Angeles shows clear patterns.

Figure 6

Weather Distribution by Year

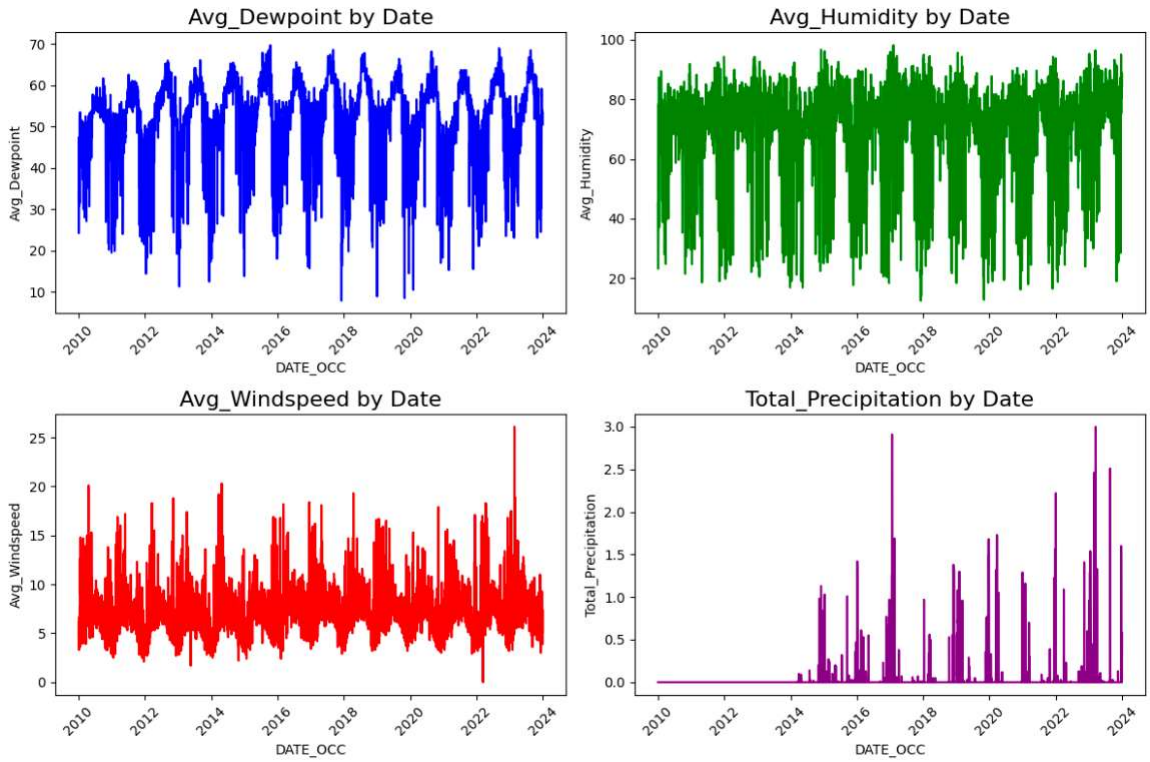


Figure 7 shows a line graph of the average weather measurements, temperature, humidity, wind speed, and air pressure for each year from 2010 through 2023. All weather measurements remain consistent for the entire duration data used in the study.

Figure 7

Climate Measures by Year

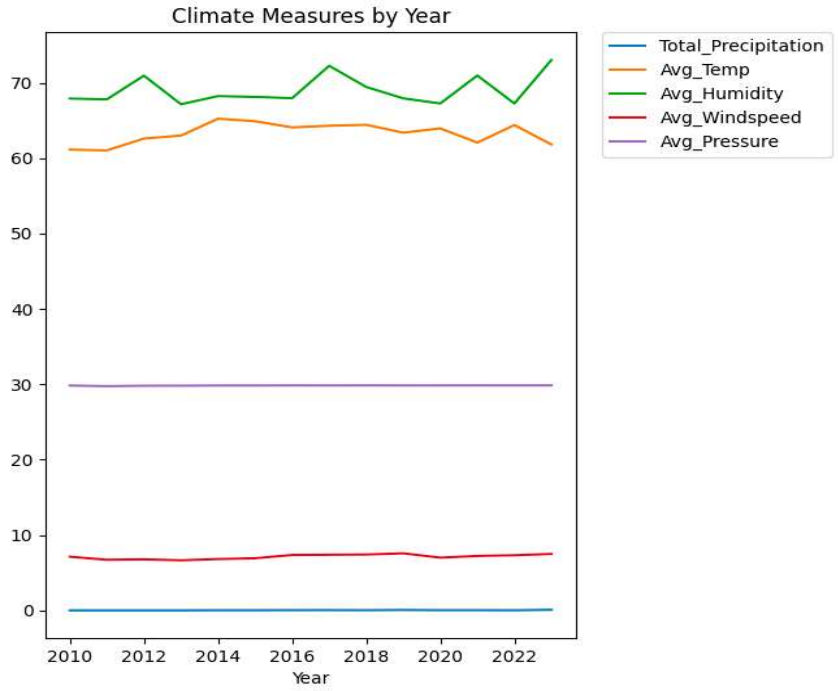
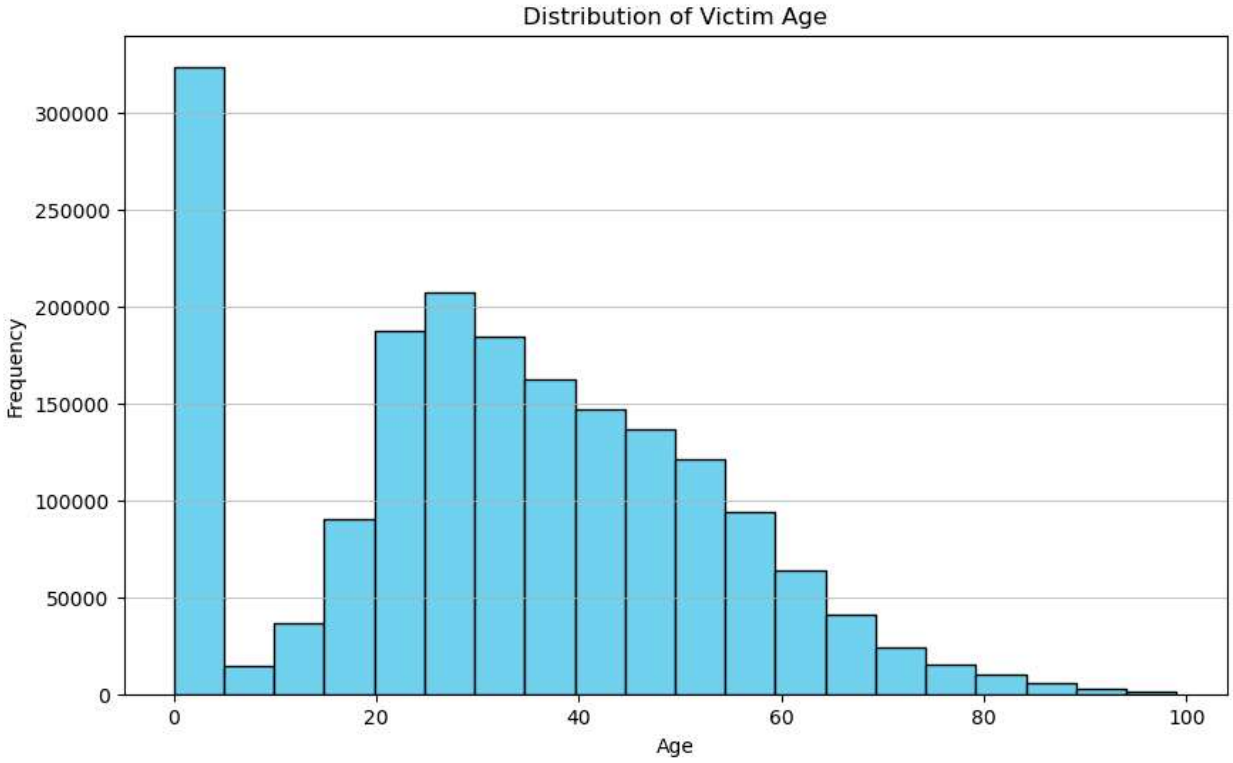


Figure 8 shows the distribution of victim ages in Los Angeles from 2010 to 2023. A large concentration of age 0 indicates a possible data entry issue or some crime trend against very young victims. Otherwise, the distribution shows a slight right skew, with most victims out of the 0-age value at around 25 to 30 years old.

Figure 8

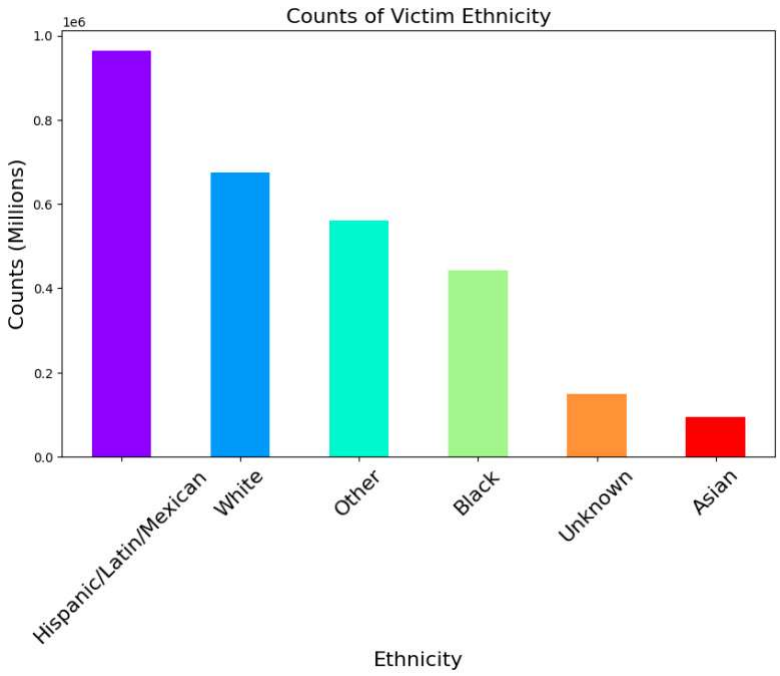
Count of Victim Age Groups



The Los Angeles crime reports encompass a classification of victim ethnicities, initially enumerating twenty-one distinct groups. For analytical clarity, this study consolidates several of these ethnicities into broader, related categories, simplifying the original complexity. Pre-existing classifications such as 'other' have been retained, whereas data with indiscernible ethnicity have been ascribed to an 'unknown' category. Illustrated in Figure 9 is a bar graph delineating six primary ethnic classifications, which quantifies the crime incidents corresponding to victims from these aggregated ethnic groups.

Figure 9

Counts of Victim Ethnicity



3.8 Machine Learning Algorithms

3.8.1 Multinomial Logistic Regression

Logistic Regression (LR) is a statistical method for analyzing datasets with one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (where there are only two outcomes). LR is used extensively in various fields, including medicine, social sciences, and machine learning, particularly for binary classification problems. LR estimates the probability that a given input point belongs to a specific class, making it applicable for predictive analytics in crime prediction (Hosmer et al., 2013). The core of LR lies in its logit function, which is the natural logarithm of the odds of the dependent variable. This feature allows LR to handle binary outcomes effectively by projecting the predictive values onto a logistic curve, ensuring the output probabilities range between 0 and 1.

In crime prediction, logistic regression can be used to predict the likelihood of a crime occurring based on various predictors such as time of day, location, economic conditions, and more. The method is well-suited for datasets where the dependent variable is categorical.
In recent studies, Logistic Regression has been effectively applied in predictive policing and crime hotspot detection. For example, Lin et al. (2018) explored crime hotspots using logistic regression, among other techniques, to assist in crime prevention by identifying areas with a high likelihood of crime. Similarly, Rummens and Hardyns (2021) explored the use of Logistic Regression for spatiotemporal crime prediction, demonstrating its utility in identifying potential crime hotspots and contributing to proactive policing strategies.

Additionally, Mandalapu (2020) highlighted the application of Logistic Regression in predicting crime rates, emphasizing its effectiveness in handling binary outcome variables, such as the presence or absence of crime. Despite its advantages, LR is not without limitations. High-dimensional spaces and multicollinearity among predictors can pose significant challenges, potentially leading to overfitting and making the model less interpretable (Ziegler, 2015).

Moreover, the linear nature of LR implies that it assumes a linear relationship between the independent variables and the log odds of the dependent variable, which may not always be held in complex crime datasets. Therefore, careful feature selection, regularization techniques, and consideration of nonlinear relationships through polynomial features or interaction terms are crucial to enhance LR's applicability and reliability in crime prediction.

3.8.2 Decision Trees

Decision Trees, also known as Classification and Regression Trees (CART), represent a collection of rules structured as "if-then-else" statements, providing an easily interpretable model for both classification and regression tasks (Bruce et al., 2020). Originally introduced by Leo

Breiman et al. (1984), Decision Trees have become a fundamental tool in predictive modeling in data science.

The construction of a decision tree involves recursive partitioning, a straightforward and intuitive algorithm. This process entails iteratively partitioning the data based on predictor values that effectively segregate it into homogeneous subsets. Two common metrics used to measure the impurity of a partition are the Gini impurity and entropy of information.

The Gini impurity index for a rectangle A is defined by

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

where p_k is the proportion of records in rectangle A that belong to class k (Shmueli et al, 2020, p.223).

The entropy for a rectangle A is defined by

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

Each partition or split refers to a specific value of a predictor variable, dividing the data into records where that predictor value is either above or below the split value. At each stage, the algorithm selects the split that minimizes the outcome impurity within each sub-partition. When no further splits are possible, the tree is fully grown, and each terminal node, or leaf, contains records of a single class. However, a fully grown tree tends to overfit the data, necessitating pruning to ensure it captures signal rather than noise.

While single Decision Trees offer rule-based interpretability, advanced algorithms like random forests and boosted trees, which incorporate multiple trees, often provide enhanced predictive performance at the expense of some interpretability.

3.8.3 Random Forest Classifier

Random Forest, introduced by Breiman and Cutler (Shmueli et al., 2020, p.243), is a widely adopted multi-tree approach. The term "Random Decision Tree" was coined by Tin Kam Ho of Bell Labs in 1995 (Khan et al., 2022). This technique builds on the concept of bagging, where "bootstrap aggregating" is employed to enhance predictive power by amalgamating multiple classifiers or prediction algorithms. Bagging, introduced by Leo Breiman in 1994 (Bruce et al., 2020, p.260), serves as the foundation for Random Forest.

In the Random Forest framework, the fundamental concept involves drawing multiple random samples, with replacement, from the dataset. For each sample, a classification or regression tree is fitted, utilizing a random subset of the data at each stage. The result is a forest of trees. The predictions or classifications from individual trees are then combined to generate more robust predictions. For classification tasks, voting is employed, while averaging is used for prediction tasks.

One valuable output provided by Random Forest is a measure of variable importance, ranking predictors based on their contribution to model accuracy. However, a key distinction is that the outcomes of a random forest cannot be illustrated in a tree-like diagram, leading to a loss of the interpretability characteristic of a single tree. To optimize the performance of Random Forest, a set of hyperparameters exists that should be fine-tuned using cross-validation, preventing issues like overfitting.

3.8.4 XGBoost

Extreme Gradient Boosting, commonly known as XGBoost, stands out as the most extensively used open-source software for boosting in data science. Originally conceived by Tianqi Chen and Carlos Guestrin at the University of Washington, XGBoost has gained widespread adoption and is available as a package for major data science languages, including Python and R (Bruce et al., 2020).

XGBoost belongs to the family of boosting algorithms, specifically implementing stochastic gradient boosting. In boosting, weak models are additively generated, and XGBoost formalizes this process as a gradient descent algorithm over an objective function. This algorithm aims to iteratively minimize errors by setting targeted outcomes for the subsequent model. These targeted outcomes are determined based on the gradient of the error, hence the name "gradient boosting."

Gradient Boosting Decision Trees (GBDTs), a form of gradient boosting, sequentially trains a collection of weak decision trees. In each iteration, the residuals of the errors from the preceding model are employed to train the next model (Sezgin, 2023). The overall prediction is a weighted combination of predictions from all the trees. Unlike random forests which tackle variance and overfitting through "bagging," GBDT is specifically designed to minimize bias and counteract underfitting through the technique of "boosting."

XGBoost has many advantages compared to many machine learning algorithms due to its ability to handle missing data, and large, complex datasets. Its high accuracy, efficiency and adaptability also makes it a desirable choice for the development of predictive models.

3.8.5 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory Networks (LSTMs) are a type of recurrent neural network architecture designed to overcome the limitations of traditional RNNs, particularly the vanishing gradient problem (Rehman & Kroll, 2021, p. 1). LSTMs can learn long-term dependencies in sequence data, making them highly effective for sequential prediction, natural language text generation, and complex time series analysis.

LSTMs achieve this capability through a complex architecture of gates – the input gate, output gate, and forget gate. These gates regulate the flow of information into and out of the cell, allowing the network to maintain or discard information across long sequences. This mechanism enables LSTMs to capture long-term dependencies and patterns in data, which traditional RNNs struggle with (Sherstinsky, 2021).

While LSTMs represent a significant advancement in sequence modeling, they come with increased model complexity and computational cost. Training LSTMs requires more computational resources and time compared to simpler models. Additionally, like other neural network models, they need large datasets to train effectively and can be prone to overfitting without proper regularization (Sherstinsky, 2021).

3.8.6 Autoregressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model, despite its foundational principles being established over four decades ago, continues to be a pivotal analytical tool in modern statistical analysis and forecasting of time series data. Emphasizing the integration of autoregression (AR), differencing (I), and moving average (MA) components, ARIMA excels in modeling and predicting the future values of time series by examining the inherent dynamics within the data (Hyndman & Athanasopoulos, 2018). This method stands out

for its applicability to a wide range of scenarios, particularly in forecasting non-seasonal time series data where linear relationships are predominant.

ARIMA's methodology involves transforming the series to stationary, exhibiting constant statistical properties over time through differencing. The model is then defined by three key parameters: p (the order of the autoregressive part), d (the degree of differencing), and q (the order of the moving average part). This notation, ARIMA(p, d, q), encapsulates the model's structure, offering a systematic approach to understanding and forecasting time series data.

Identifying the optimal parameters for p, d, and q is critical to the model's success and often involves iterative experimentation and testing, such as the Box-Jenkins methodology. This approach ensures that the model closely captures the temporal dependencies within the data, providing accurate forecasts. However, the application of ARIMA models is constrained by their linear nature and the necessity for the time series to be stationary. These requirements limit the model's effectiveness in handling data with strong seasonal patterns or nonlinear characteristics without modifications or extensions, such as the Seasonal ARIMA (SARIMA) model or integration with non-linear modeling techniques.

Recent advancements and studies within the last decade highlight ARIMA models' ongoing relevance and adaptation in various fields, including economics, environmental science, and beyond, highlighting their versatility and enduring value in time series analysis (Hyndman & Athanasopoulos, 2018). Despite the emergence of more complex and computationally intensive models, ARIMA's simplicity, interpretability, and robustness make it an indispensable tool in statistical forecasting, especially for applications where understanding the underlying model and its parameters is crucial.

3.8.7 Tools, Modeling, and Training

This study's primary tool for analysis and modeling is the Python programming language, with support from modules NumPy, Pandas, Scikit-Learn, XGBoost, TensorFlow, Keras, and Seaborn. These modules provide robust modeling frameworks that can be tuned to improve their respective prediction capabilities. Scikit-Learn and XGBoost both offer machine learning model frameworks with tunable parameters and cover the spread of models used in the classification portion of this study. TensorFlow provides customizable neural network frameworks to support the time series analysis using its built-in LSTM, and ARIMA classes and methods. Keras acts as an overlay API to simplify TensorFlow usage. Pandas, NumPy, and Seaborn are used for manipulating and visualizing the data.

3.8.8 Modeling

To properly train and test each model, the dataset is split into training and testing sets, where 80% of the data is allocated to training, while the remaining 20% is used to check for accuracy, precision, recall, and F1-score, as well as overfitting. Generic model templates are used for each algorithm listed in section 3.7 and tuned to improve performance. Initially, all predictor variables, such as weather, demographics, weapons, time, and location, are used. The goal is to establish a baseline for each model using all predictors, then remove victim demographics and weather data individually to see if model performance is significantly affected. Using hyperparameter grid searches for each model provided a computationally expensive, but thorough investigation of how hyperparameters were affecting each model and allowed for each model to be optimized. The results of these models will be discussed in chapter 4.

Modeling for crime hotspot prediction requires a slightly different approach. The target variable becomes an aggregate total of crimes by specific area. This is achieved by grouping the data by area and date the crime occurred. The created target variable is called Crime Incidence, which means the number of crimes that occurred daily per area. It is denoted as Crime_Incidence. The goal is to predict the number of future crimes by area by assessing seasonal trends and comparing each area's results using moving averages and neural networks that work well with time series data. Crime Incidence aggregates all types of crimes committed in each area per day, for the 14-year span of our dataset. The entire Los Angeles is also analyzed to predict total crime rate. A total of twenty-two models are generated representing these areas and the city of L.A. Two algorithms, Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) are used to each build these models, making a total of 44 models. The dataset is sorted by date to ensure chronological order. The dataset is then split into training and test sets in the ratio 80:20, respectively.

To build our ARIMA model, there are 22 datasets, 21 of which are sliced from the main crime dataset, with each representing an area, the 22nd being the full crime dataset. Our ARIMA model fits the training data with a specified order (p, d, q). Upon trying out different combinations of p, d, q, we settled for parameters (p=1, d=0, q=2) as they gave optimal RMSE values compared to at least 15 other combinations where each of p, d, q was between 0 and 3.

To build the LSTM models, there are 22 datasets, just like with ARIMA. For each model, the size of the sliding window is set to a default of 40. The LSTM model architecture consists of two LSTM layers with 256 and 128 units respectively, a dense layer with one output neuron is added for regression, and the model is compiled using mean squared error loss and “Adam”

optimizer, for hyperparameter tuning. A default batch size of 32 is used and the model is trained for 25 epochs. Training is performed with early stopping to prevent overfitting.

3.8.9 Model Performance Evaluation

To evaluate the results of this study, the team broke the problem down into two separate sections: classification problems and time-series predictions by area. The research hypothesis questions pertaining to the correlation of victim demographics and weather on crime category were treated as classification problems while predicting future crime amounts by area to determine hotspots was treated as a time series problem. The evaluation metrics for each hypothesis are discussed in this section.

3.8.10 Classification of Crime Code Categories Model Evaluation

A variety of performance metrics were used to assess each classification model. Four different models will be created for every classification algorithm to examine the impact of victim demographics and weather on crime prediction accuracy. These models will include one with all variables, one without weather features, one without victim demographic features, and another without demographic and weather features. Accuracy is the principal measure, showing the overall rate of correct predictions. Precision and recall provide more nuanced insights, showing the model's accuracy in its positive predictions and its ability to identify all relevant cases within each crime category. The F1 Score combines precision and recall into a metric that captures each model's accuracy and thoroughness.

The Confusion Matrix enhances the evaluation by giving a detailed picture of the model's performance, showing correct and incorrect predictions for each crime category. To counteract any imbalance in crime category representation, down sampling is employed on the dataset to

balance it. The adjusted dataset then serves as a basis for evaluating whether class imbalance significantly impacts the model's predictive accuracy. Other metrics such as Area under the Curve (AUC) and Receiver Operating Characteristic (ROC) are more attuned to binary classification problems (Nahm, 2022, p. 26) and are purposely omitted from this evaluation, as eight classes are being predicted with no one crime category code class being more important than another.

Additionally, the Matthews Correlation Coefficient and Cohen's Kappa are robust measures of model quality (Chicco et al., 2021). The former is valuable for assessing classification quality in the presence of class imbalances, and the latter offers a normalized score that reflects the level of agreement between predicted and actual classifications, adjusted for chance. Log Loss is also used to assess the confidence of the model's probabilistic predictions, placing value on both the precision of the prediction and its accuracy with binary and multiclass datasets (Kumar & Sastry, 2018, p. 687). After developing the four model variations for each algorithm, these metrics are compared to determine whether victim and weather demographics influence model performance in both the complete and balanced datasets.

3.8.11 Future Crime Rates and Hotspots Model Evaluation

To evaluate our ARIMA and LSTM models, a common metric, root mean squared error (RMSE) is used on the validation set. In ARIMA, the fitted model is evaluated on the test set, and RMSE is calculated to quantify the ARIMA model's prediction accuracy. A visualization plot is generated to visualize the training data, the actual data, and the predicted data. The x-axis will represent the DATE_OCC, and the y-axis will represent the crime incidence, as we shall see in the next chapter. Plots and RMSE values will be generated for each of the 22 ARIMA models.

For LSTM, crime incidence predictions are made on the validation set using the trained model. Predictions are inverse transformed to obtain actual crime incidence values. Upon model training, training and validation loss curves are plotted to monitor model performance. This could raise issues such as overfitting or underfitting or accentuate model complexity if there are disparities between the training loss and validation loss. Since a regularization method such as early stopping is introduced which prevents the model from learning noise in the training data, we expect our trained models to improve in its generalization to unseen data. The root mean squared error (RMSE) evaluates the model's performance on the validation set. It is a measure of the average deviation of the predicted values from the actual values. A visualization function plots the training data, the actual crime incidence, and the predicted crime incidence for visualization and comparison by interpretation and analysis. This process is repeated for each of the 22 LSTM models. Both ARIMA and LSTM models are compared with one another. The lower the RMSE value, the better the model.

3.9 Conclusion

A significant amount of preparation was required to begin creating predictive models for the final dataset. Manually gathering 168 monthly weather datasets and concatenating them chronologically to align correctly for inner joining with two crime datasets, also merged chronologically, required attention to detail to avoid introducing data errors. Auditing the final dataset revealed many missing values, outliers, and unwieldy amounts of categories in certain variables that would create dimensional spaces far too high for practical computing. To alleviate these issues, several variables with missing values were removed. Afterward, removing outliers and imputations to correct smaller data issues, like negative ages, allowed the final dataset to be free of outliers and missing values.

Heavy class imbalances also presented themselves during integrity checks of the final datasets, including within the target variable crime code. Crime code contained 144 categories, with some having several hundred thousand instances and others containing only two instances. The action deemed necessary to put the target variable in a more usable format was to group similar crimes and reduce the categories from 144 to 8. Victim descent showed a more detailed separation of victims from Asia and was broken down into specific countries. Considering other descents were grouped into large, nonspecific groups such as white, black, and Hispanic/Latino, all categories representing Asian countries were relabeled as "Asian" to improve the racial class imbalance.

With exploration and preprocessing completed, modeling variables were identified and manipulated. Using the target variable crime category code and creating dummy variables for each categorical predictor variable were the final steps before implementing machine learning models and neural networks to begin assessing the hypothesis questions. Two variants of each machine learning model will be created for each hypothesis, one with victim demographics, one without, and one with weather information and one without, to compare their predictive accuracies and determine the result of the research hypothesis. For neural networks, the dataset is divided by area, and crime amounts are totaled to begin implementing ARIMA and LSTM networks to support the research hypothesis of whether future crime hotspots can be predicted. The results of these methods are discussed in Chapter 4.

Chapter 4: Results

4.1 Introduction

Results of the spatiotemporal analysis to predict crime hotspots and the crime code category classification hypotheses are presented in this chapter. To determine the impact of victim demographics and weather impact on crime category code prediction, four separate datasets were used to evaluate the change in predictive accuracy. These datasets included the full (cleaned) dataset, the dataset with weather information removed, the dataset with victim demographic information removed, and the dataset with both weather and victim demographic information removed. For spatiotemporal analysis, to predict crime hotspots in Los Angeles effectively, we predicted crime incidences for each geographical area of Los Angeles, including the full city, making 22 crime hotspot predictions. Each area is time-series analyzed using LSTM (Long Short-Term Memory) and ARIMA (Autoregressive Integrated Moving Average) to determine future crime amounts per locations. The models are compared with one another in this chapter.

4.2 Hypothesis 1: Crime Hotspot Prediction Results

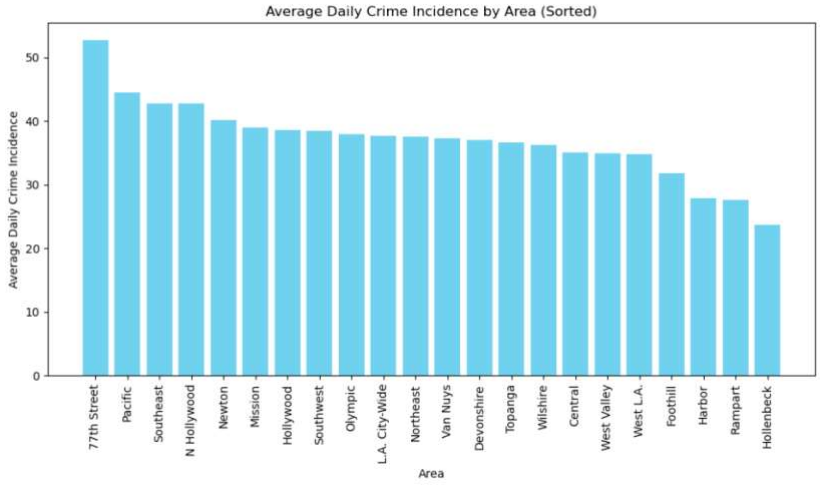
The results of our crime hotspot prediction will be discussed in this section. 22 ARIMA models and 22 LSTM models representing the 22 crime hotspots across Los Angeles (21 areas and the full city of L.A [Los Angeles]) were trained and evaluated. The root mean squared error (RMSE) of each of these 44 models and their visualization charts showing the trained data, actual data and predicted data will be compared.

4.2.1 Average Daily Crime Incidence Distribution

Figure 10 shows the distribution of the average daily crime incidences by area. In the last 14 years (Jan 1, 2010 - Dec 31, 2023), an average of 37.61 crimes took place daily per area across Los Angeles, taking the full crime dataset into perspective. Narrowing down to individual slices of datasets by area, the data shows some areas with higher-than-average crime incidence rates while some are lower. An average of about 53 crimes took place daily in 77th Street area, higher than in any other area of Los Angeles. Pacific ranked 2nd in average daily crime incidence rates at 44. Southeast and North Hollywood tied in 3rd place with about 43 crimes per day. Newton rounded up the top five with 40 crime incidences per day. Hollenbeck had the least amount of average daily crime incidences with about 24 crimes per day.

Figure 10

Average Daily Crime Incidents by Area



4.2.2 Descriptive Statistics of Total Crime Incidence across all Areas

Table 6 visually represents the descriptive statistics of total crime incidence across each area. 77th Street had the highest total crime incidence count of 186,392 amongst the 21 areas with

a maximum value of 256 crimes per day. However, Foothill area, which had the lowest total crime incidence of 102,688, also recorded the highest daily crime rate ever, with 322 crimes. L.A. City-Wide, which encompasses all the areas, recorded a total of more than 2.8 million crime incidences. The top five areas with the highest crime rates over 14 years were77th Street, Southwest, Central, Pacific, and North Hollywood.

Table 6

Crime Rate Descriptive Statistics by Area

Area	Count	Mean	Std	Min	25%	50%	75%	Max
L.A. City-Wide	2883802	37.61	19.01	1	26	32	44	322
77th Street	186392	52.75	23.51	1	38	45	63	256
Southwest	184073	38.41	11.15	5	32	37	43	171
Central	156659	35.05	13.11	6	26	34	43	174
Pacific	150005	44.49	20.66	1	30	36	56	122
N Hollywood	147519	42.71	19.95	1	30	36	52	266
Southeast	145151	42.81	22.38	1	29	36	52	250
Hollywood	142079	38.56	18.77	1	26	33	46	202
Northeast	139159	37.57	18.24	1	26	31	46	195
Van Nuys	137837	37.28	18.63	7	26	31	46	234
Newton	133213	40.13	19.88	1	26	33	50	180
Mission	132084	39.01	21.75	1	27	33	46	286
Rampart	130369	27.58	9.46	4	22	26	31	161
Wilshire	129314	36.28	18.45	1	23	30	45	130
West L.A.	127442	34.73	16.09	5	24	29	42	120
West Valley	126732	34.98	17.24	4	23	29	44	172
Olympic	126603	37.94	18.92	1	25	31	46	194
Topanga	125854	36.62	16.54	1	26	31	44	186
Devonshire	125420	36.99	17.42	1	25	32	46	208
Harbor	124890	27.84	11.15	1	22	26	31	195
Hollenbeck	110319	23.66	8.54	4	19	23	27	130
Foothill	102688	31.78	22.96	1	20	26	38	322

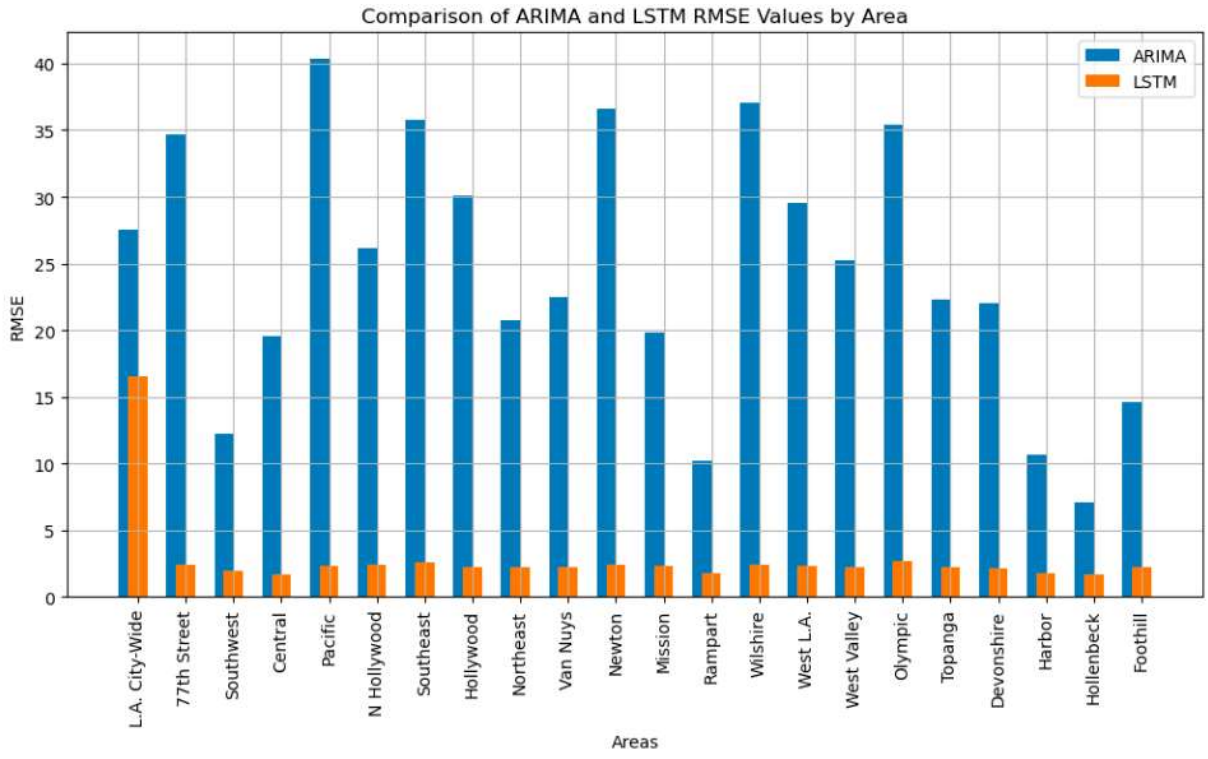
4.2.3 Modeling Results of RMSE across Algorithms (ARIMA and LSTM)

In Figure 11, the ARIMA/LSTM RMSE by Area bar chart compares the results of the Root Mean Squared Error (RMSE) on the validation set of the two time series algorithms used. The lower the RMSE, the better the model. For each area, LSTM model performed better than

ARIMA model. For the L.A. City-Wide area (full dataset), the RMSE for LSTM was 16.55 compared to the ARIMA model which had 27.57. LSTM performs even better when the individual areas are assessed. The disparity between the models is widest in Pacific area, where ARIMA has an RMSE of 40.33 but LSTM RMSE for the same area is 2.36. Hollenbeck has the lowest disparity between both RMSEs, with ARIMA RMSE being 7.05 compared LSTM with an RMSE of 1.69. Both models performed well in predicting crime incidences in Hollenbeck even though LSTM was far more accurate. An RMSE of 1.69 means that on average, the predicted number of crimes per day deviated from the actual number by approximately 1.69 incidences.

Figure 11

LSTM and ARIMA RMSE Results per Area



4.2.4 Modeling Results of RMSE: Comparison across Areas

Table 7 compares the results of the Root Mean Squared Error (RMSE) of the two time series algorithms used, ARIMA and LSTM across areas. Olympic area had the largest LSTM RMSE of 2.65, whereas Pacific area had the largest ARIMA RMSE of 40.33. Hollenback had the smallest RMSE across both LSTM and ARIMA, 1.69 and 7.05, respectively. The average root mean squared error (RMSE) across the 21 areas (excluding L.A. City-Wide) for LSTM is 2.21, and for ARIMA, is 24.40. This means that on average, the predicted number of crimes per day across all areas deviated from the actual number by approximately 2.21 incidences for LSTM model, and 24.4 incidences for ARIMA model.

Table 7

Modeling Results of RMSE across Algorithms

Area	ARIMA_RMSE	LSTM_RMSE
L.A. City-Wide	27.57	16.55
77th Street	34.64	2.46
Southwest	12.2	1.92
Central	19.53	1.72
Pacific	40.33	2.36
N Hollywood	26.15	2.4
Southeast	35.79	2.63
Hollywood	30.08	2.24
Northeast	20.78	2.22
Van Nuys	22.45	2.23
Newton	36.63	2.43
Mission	19.86	2.35
Rampart	10.17	1.73
Wilshire	37.02	2.45
West L.A.	29.57	2.32
West Valley	25.22	2.19
Olympic	35.4	2.65
Topanga	22.32	2.21
Devonshire	21.99	2.18
Harbor	10.71	1.74
Hollenbeck	**7.05**	**1.69**
Foothill	14.6	2.23
Average RMSE	24.4	2.21

4.2.5 Actual Prediction Charts of Each Area

Figure 12 shows the charts of the trained data, the actual data, and the predicted data for the Hollenbeck area. Hollenbeck had the least LSTM RMSE and least ARIMA RMSE; in other words, it performed best in predicting crime incidences/hotspots among all areas. Both ARIMA and LSTM models are compared side by side. For LSTM (on the right), the predicted data (yellow legend) almost perfectly matched the actual data (orange legend), showing remarkably high accuracy and unnoticeable marginal errors, compared to ARIMA model's predicted data (green legend on the left).

Figure 12

ARIMA and LSTM Prediction Chart for Hollenbeck Area

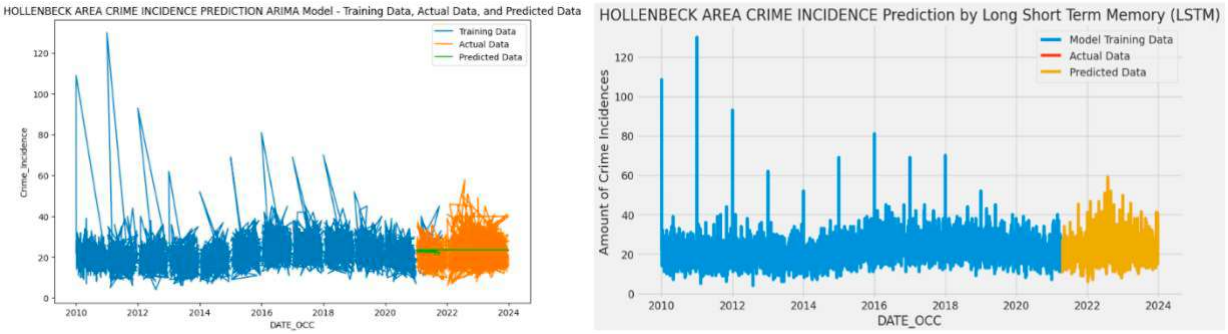


Figure 13 shows the prediction charts of the Olympic area. Olympic had the highest LSTM RMSE; in other words, compared to other areas (apart from the full Los Angeles area), its prediction of crime incidences/hotspots performed worse. However, it still performed much better than ARIMA RMSE, as the predicted data (yellow legend) nearly matched the actual data (orange legend), showing high accuracy and not as much deviation as ARIMA.

Figure 13

ARIMA and LSTM Prediction Chart for Olympic Area

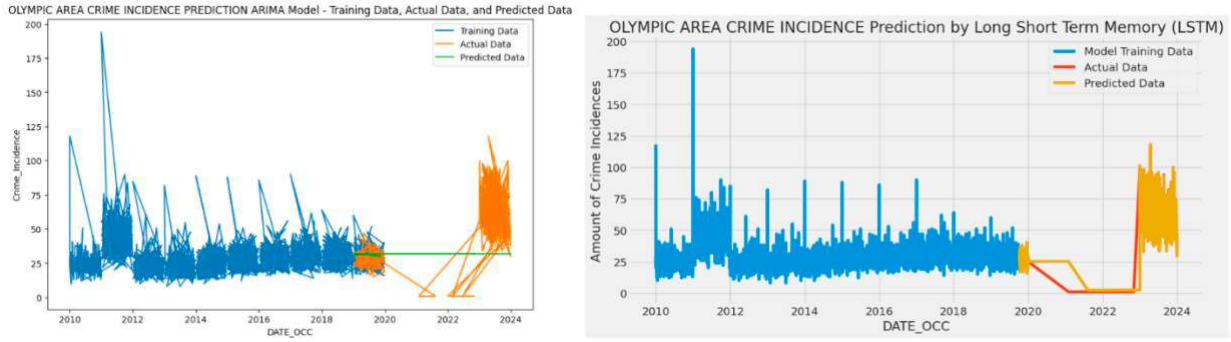
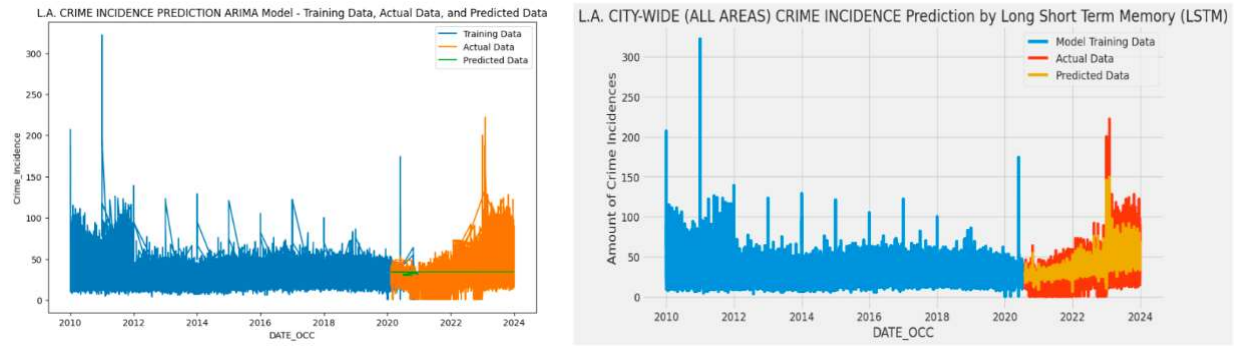


Figure 14 shows the prediction charts of the full Los Angeles area, comparing both ARIMA and LSTM models. Visually, for every area, including the full Los Angeles city, LSTM performed better than ARIMA in predicting crime incidences.

Figure 14

ARIMA and LSTM Prediction Chart for Los Angeles City-wide Area (all 21 areas)



Appendix C shows the prediction charts for all 22 areas and the full Los Angeles city-wide area. From these charts, LSTM outperformed ARIMA because the LSTM models are able to capture complex patterns better, especially when dealing with large datasets such as the Los

Angeles Crime Dataset from 2010 to 2023 which has over 2.88 million records. LSTM is a type of recurrent neural network (RNN) which can capture complex temporal dependencies in the data; they can also model non-linear relationships between variables. ARIMA however, assumes linear relationships between variables. This is seen by the linear movement of the predicted data of ARIMA (green legend) compared to the non-linear movement of the predicted data of LSTM (yellow legend). LSTM models can also handle irregularities, trends, and seasonality in the data without categorically stating these components, however, ARIMA models require manual specification of seasonal and trend components.

4.2.6 Training and Validation Loss Chart across areas for each LSTM model

In Figures 15 and 16, we see the plots of the training and validation loss for the best and worst LSTM Area models (Hollenbeck and Olympic). The exceptionally low difference between the training and validation loss implies that the models are performing well and are generalizing effectively to unseen data. It also shows that the number of layers being used is appropriate for the complexity of the L.A. Crime dataset. The models are neither underfitting nor overfitting. Underfitting occurs when both training and validation loss remain high and do not decrease significantly throughout training. In such cases, the model cannot learn from the training data due to its simplistic nature of being unable to capture underlying patterns in the data, resulting in poor performance on training and unseen data. Overfitting, on the other hand, is indicated by a large gap between training and validation loss, where training loss decreases but validation loss increases or remains high. In such cases, the model performs well on the training data but poorly on unseen data due to inability to generalize from it. Figures 15 and 16 show a good fit. Here, we see the training losses decrease steadily while the validation losses decrease initially but stabilize

or start increasing slightly after a certain point. This suggests that the models are learning the underlying patterns in the data without overfitting thereby generalizing well to unseen data.

Early stopping, an effective regularization technique which helps avoid overfitting, was applied effectively to all the LSTM models. This helps prevent the models from fitting the noise in the training data while still capturing relevant patterns. Also, the low RMSE values of the LSTM model for each area show how effective the models can be with performance on unseen data. Figure 17 shows the plot of the loss chart for the full Los Angeles area. The complete loss charts of all areas are provided in Appendix D.

Figure 15

LSTM Training and Validation Loss Chart for Hollenbeck Area

Figure 16

LSTM Training and Validation Loss Chart for Olympic Area

Figure 17

LSTM Training and Validation Loss Chart for Los Angeles City-wide Area (all 21 areas)

4.3 Hypothesis 2 & 3: Crime Classification & Impact of Weather and Demographics

4.3.1 Dataset Results

This section presents the results from the four different datasets used to evaluate the change in predictive accuracy concerning victim demographics and weather impact on crime category code prediction. For this analysis, datasets begin with the full, cleaned dataset and include:

1. The full dataset with all weather and victim information included.
2. A dataset with weather information excluded.
3. A dataset with victim demographic information excluded.
4. A dataset with both weather and victim demographic information excluded.

Results are displayed in Table 8, which contrasts each dataset variant against the four models based on various performance metrics.

Table 8

Model Performance Metrics on Each Dataset

Model and Dataset Variant	Accuracy	Precision	Recall	F1 Score	Matthews Coefficient	Cohens Kappa	Log Loss
Full Dataset Models							
Logistic Regression	0.74	0.74	0.74	0.72	0.64	0.63	0.78
Random Forest	**0.81**	**0.81**	**0.81**	**0.81**	**0.74**	**0.74**	**0.71**
Decision Tree	0.74	0.74	0.74	0.74	0.67	0.67	9.34
XGBoost	0.78	0.78	0.78	0.77	0.69	0.69	0.74
Models with Dataset Excluding Weather Features							
Logistic Regression	0.71	0.65	0.71	0.68	0.64	0.63	0.78
Random Forest	**0.81**	**0.81**	**0.81**	**0.81**	**0.74**	**0.74**	**0.73**
Decision Tree	0.74	0.75	0.74	0.74	0.68	0.68	9.24
XGBoost	0.76	0.77	0.76	0.74	0.69	0.69	0.74
Models with Dataset Excluding Victim Demographic Features							
Logistic Regression	0.63	0.59	0.64	0.55	0.63	0.62	**0.81**
Random Forest	**0.67**	**0.66**	**0.68**	**0.67**	**0.72**	**0.71**	0.82
Decision Tree	0.65	0.65	0.65	0.65	0.55	0.55	12.67
XGBoost	0.64	**0.62**	**0.64**	**0.57**	0.54	0.53	**0.94**
Models with Dataset Excluding Victim Demographic and Weather Features (Baseline)							
Logistic Regression	0.64	0.59	0.64	**0.67**	0.63	0.63	0.80
Random Forest	**0.65**	0.64	**0.65**	0.65	**0.70**	**0.73**	**0.77**
Decision Tree	**0.65**	**0.65**	**0.65**	0.65	0.56	0.56	12.5
XGBoost	0.64	0.60	0.64	0.57	0.54	0.53	0.94

4.3.2 Model Performance

This section evaluates the performance of various machine learning models across different dataset variants, focusing on metrics such as accuracy, precision, recall, F1 score, Matthew's correlation coefficient, Cohen's kappa, and log loss. The models examined include Logistic Regression, Random Forest, Decision Tree, and XGBoost, tested on four variations of

the dataset: the full dataset, datasets excluding weather features, excluding victim demographic features, and both weather and victim demographic features (baseline).

On the full dataset, the Random Forest model emerged as the standout performer, boasting the highest accuracy, precision, recall, and F1 score of 0.81, alongside the lowest log loss of 0.71. This model also achieved the highest Matthews correlation coefficient and Cohen's kappa, both at 0.74, indicating a strong and consistent performance across the board. XGBoost also displayed commendable performance with slightly lower metrics than Random Forest but still robust, demonstrating its efficacy in handling complex data structures.

When weather features were excluded from the dataset, the performance of all models slightly declined. Yet, Random Forest maintained its superiority with unchanged accuracy, precision, recall, and F1 score, albeit with a slight increase in log loss. This pattern highlights the robustness of Random Forest against dataset variations, particularly when environmental factors are not considered.

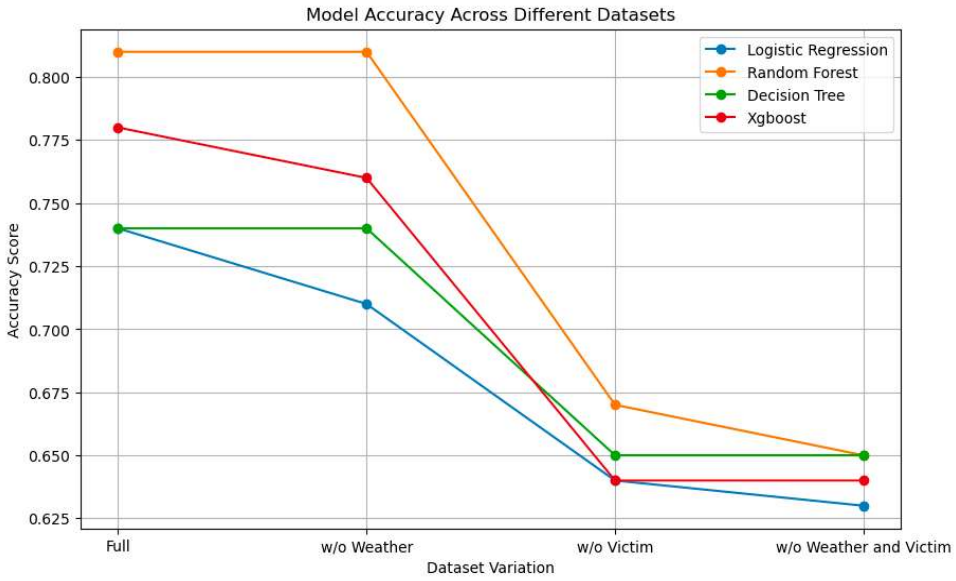
The exclusion of victim demographic features resulted in a more pronounced performance dip across all models. This variant demonstrated the importance of demographic information in predicting crime categories, as evidenced by significant declines in key metrics. For instance, Logistic Regression's accuracy dropped to 0.63 from 0.74 on the full dataset, and Random Forest decreased accuracy to 0.67 from 0.81.

The baseline dataset variant, which excludes weather and victim demographic features, further accentuated the performance challenges all models face. This variant underscored the critical role of comprehensive feature sets for model accuracy and efficiency, with all models showing reduced performance metrics compared to their results on the full dataset. The following figures show the performance change for each model when trained on each dataset

variant. The following figures show a clear decline in model performance when victim information is removed from the dataset and slight change when weather information is removed.

Figure 18

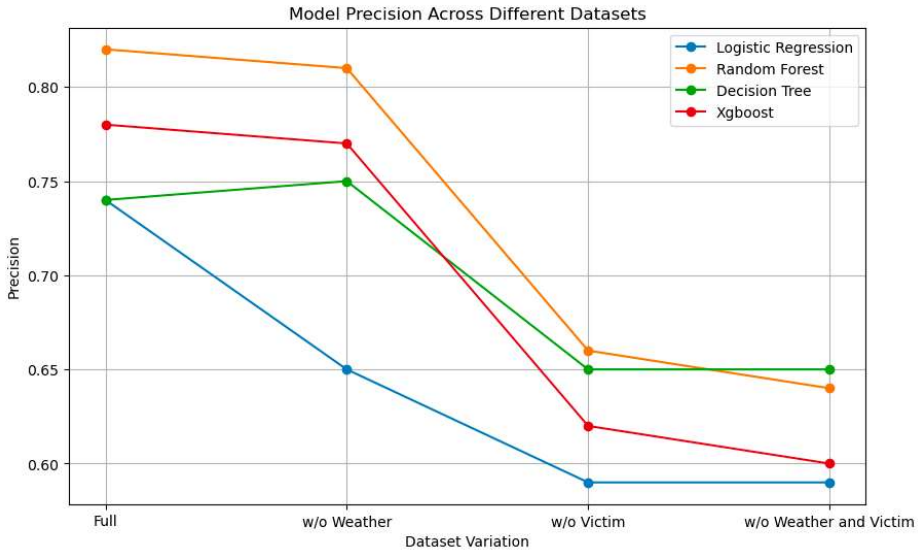
Model Accuracy for All Datasets



The predictive accuracy of each model shows a slight decline in XGBoost and Logistic Regression, but not Random Forests and Decision Trees when weather features are removed. Every model experiences a large loss of accuracy when victim information is removed compared to the full dataset.

Figure 19

Model Precision for All Datasets



The overall trend is present for model precision, except for logistic regression, which shows a larger decline in precision when weather is removed. Still, it remains stable for the datasets without victim and weather or victim features. Recall and F1 score show the same pattern of decline accuracy when the hypothesis variables are removed.

Figure 20

Model Recall for all Datasets

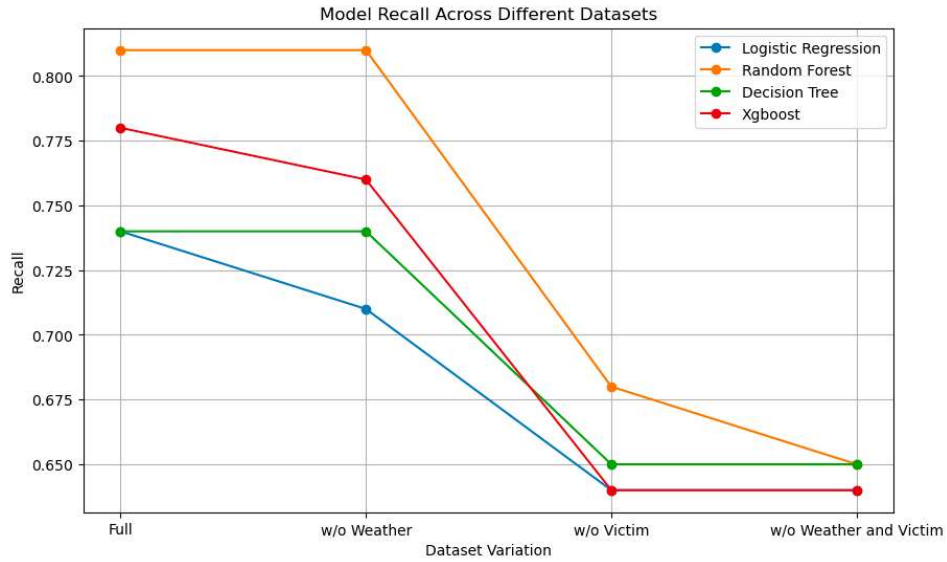
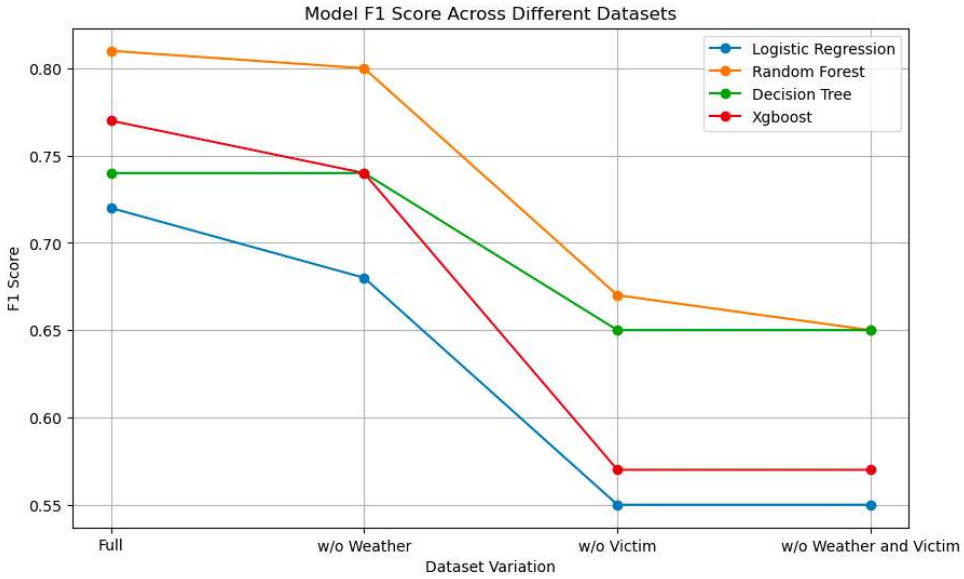


Figure 21

Model F1-Score for All Datasets



Cohen's Kappa and Matthew's Correlation Coefficient have slightly different outcomes with respect to random forests and logistic regression. XGBoost and Decision Trees are more impacted by these performance metrics and, again, follow the same decline in the model's ability to distinguish classes as victim and weather information is removed from the dataset.

Figure 22

Cohen's Kappa for All Algorithm and Datasets

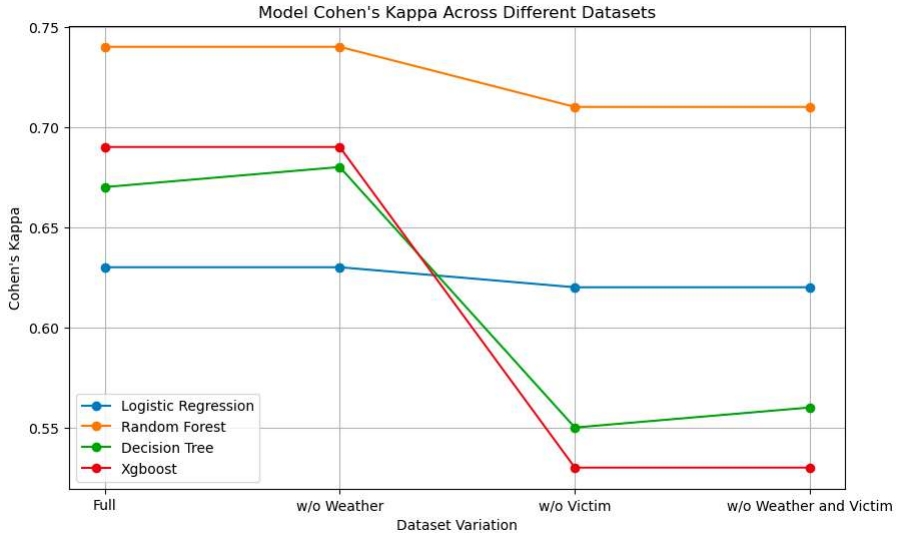
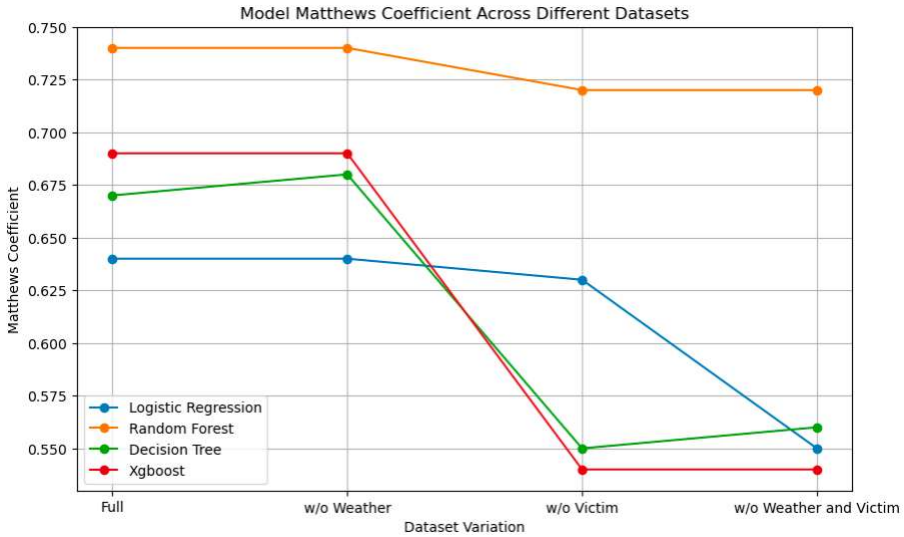


Figure 23

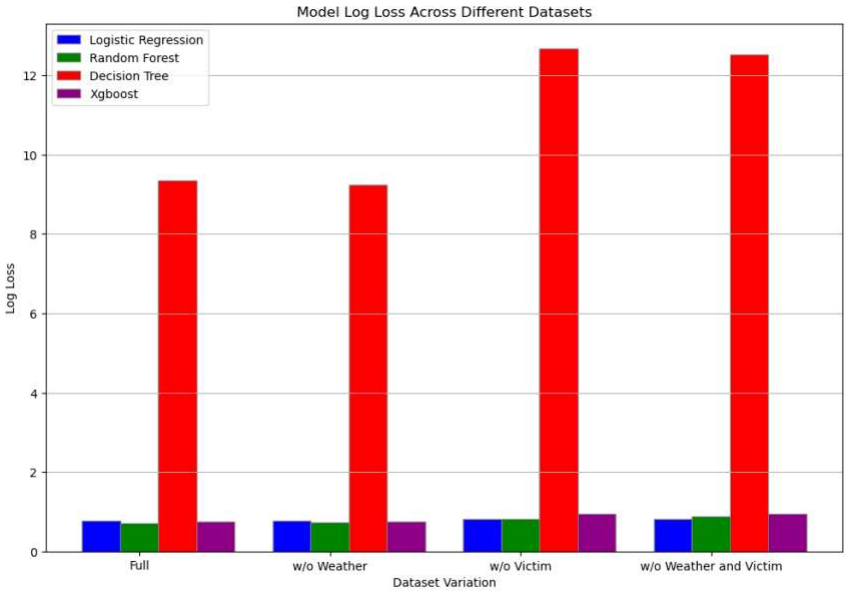
Matthew's Correlation Coefficient for All Models and Datasets



The last performance metric, log loss, showed very small changes across datasets for logistic regression, decision trees, and random forests but did increase as variables were removed, indicating a slight decline in model performance. Decision trees has a notably higher log loss than other models, which may indicate a lesser ability to discern between crime categories.

Figure 24

Log Loss for All Models



The Random Forest model consistently outperforms other models across different dataset variants, demonstrating its robustness and adaptability to varying feature sets. However, excluding critical features, especially victim demographic information, significantly impacts the performance of all models.

4.3.3 Impact of Weather Data

The analysis of the impact of weather data on predictive accuracy and other performance metrics of various models reveals an interesting insight into the role of environmental factors in crime prediction. Specifically, when weather features—total precipitation, average wind speed, average dew point, average humidity, and average pressure—are removed from the dataset, the performance of most models remains remarkably consistent with their performance on the full dataset. This includes metrics such as accuracy, precision, recall, and F1 score, which show minimal variation, typically within a one percent range of the models trained on the full dataset.

The slight changes in log loss for the Random Forest and Decision Tree models—increasing and decreasing, respectively, by as little as 0.1—point to minimal, if any, impact of weather data on these models' predictive efficiency. The Random Forest model exhibited a slight degradation, as evidenced by a minor increase in log loss, whereas the Decision Tree model showed a small improvement, indicated by a decreased log loss. Despite these slight variations, the overall negligible change in log loss further supports the conclusion that weather data do not significantly influence the prediction of crime categories.

These findings suggest that, at least within the context of Los Angeles—a city known for its generally stable weather conditions but also susceptible to severe weather events, rainy seasons, earthquakes, and other natural phenomena—weather factors have little to no impact on the ability of the analyzed models to predict crime category codes. This consistency in model performance, regardless of the inclusion or exclusion of weather data, underscores the potential for these models to effectively classify crime categories without relying on environmental factors, possibly due to the overriding influence of other, more determinant features present in the dataset.

4.3.4 Impact of Victim Demographic Data

The investigation into the impact of victim demographic data on crime code classification reveals a significant correlation between the presence of demographic information—such as age, sex, and ethnic origin—and the effectiveness of machine learning models in predicting crime categories. Notably, when demographic features are excluded, the Random Forest model experiences a notable decline in performance, with its accuracy dropping by 14% from an original 0.81 to 0.67. This significant decrease highlights the model's dependency on demographic data for accurate predictions.

Similarly, the Decision Tree model sees a 9% fall in accuracy, moving from 0.74 with the full dataset to 0.65 without victim demographic data. This reduction emphasizes the importance of including victim demographics in crime prediction models to maintain high levels of accuracy. The XGBoost model also exhibits a comparable decline in accuracy, decreasing by 14% from 0.78 to 0.64, underscoring the vital role demographic information plays in enhancing predictive performance across various models.

Removing victim demographic data impacts the accuracy and other crucial metrics, such as Cohen's Kappa and the Matthews correlation coefficient. For instance, the Logistic Regression model shows a slight decrease in Cohen's Kappa from 0.63 to 0.62, and a similar trend is observed in the Matthews correlation coefficient across models. The XGBoost model for example, sees a decrease in its Cohen's Kappa from 0.69 to 0.53, demonstrating a substantial decline in the model's ability to predict crime categories without demographic information accurately.

These observations confirm the hypothesis that victim demographic data significantly influences the accuracy of crime category code predictions. The stark reductions in performance

metrics such as accuracy, Cohen's Kappa, and the Matthews correlation coefficient across models like Random Forest, Decision Tree, and XGBoost highlight the essential nature of demographic features in crime prediction algorithms. Including these features enhances the models' ability to classify crimes more accurately and indicates the complexity and nuanced nature of crime prediction tasks, underscoring the need for comprehensive datasets that include demographic information for improved model performance.

4.3.5 Baseline Model Testing

In the final analysis phase, the impact of both victim demographic and weather data on crime category prediction was further examined by evaluating models developed using a baseline dataset. This dataset was stripped of all hypothesis variables, including the victim's age, sex, ethnic origin, total precipitation, average humidity, wind speed, dew point, and pressure, leaving only the core predictor variables. This approach aimed to isolate the effect of these removed variables on the models' predictive capabilities. The baseline models were then compared to models built on datasets with either the weather features removed, or the victim demographic features removed to assess their respective impacts on crime category classification.

The findings from this comparison were revealing. The models trained on the baseline dataset, which lacked both weather and victim demographic information, performed similarly to those trained on the dataset that excluded only victim demographic data. This observation was consistent across various performance metrics, including accuracy, precision, recall, F1 score, Matthew's correlation coefficient, Cohen's Kappa, and log loss. For instance, the Random Forest model's accuracy remained at 0.65, and the Decision Tree model's accuracy was also stable at 0.65, irrespective of whether the dataset included victim demographic data or was the baseline dataset. This similarity in model performance strongly suggests that the removal of weather data

had a negligible effect on the models' ability to classify crime categories, corroborating the earlier finding that weather data have minimal impact on crime prediction.

Conversely, the comparison underscores the significant role of victim demographic information in enhancing the predictive accuracy of models. The consistent performance metrics between models trained on datasets without victim demographics and the baseline dataset further indicate that including victim demographics substantially contributes to model efficacy. This finding aligns with the contrapositive analysis method, providing additional evidence that victim demographic data are crucial for accurate crime category prediction.

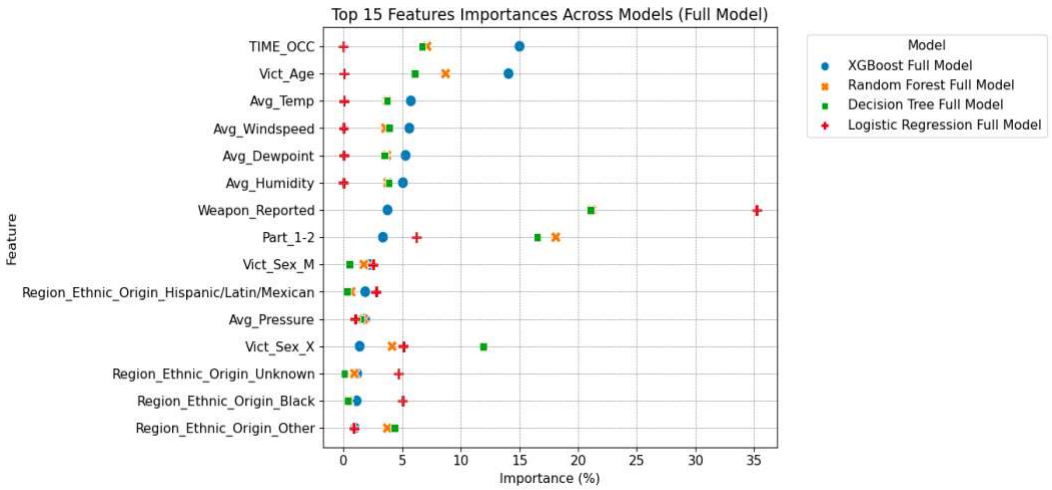
Overall, the baseline model testing phase reinforces the conclusion that while weather data does not significantly influence crime category code prediction, victim demographic information plays a critical role. The negligible differences in performance metrics between the baseline dataset and datasets excluding specific data types highlight the importance of including comprehensive demographic information to achieve optimal predictive performance in crime category classification tasks.

4.3.6 Feature Importance

Each algorithm's percentage of feature importance was compared to the other algorithms by dataset to assess feature importance. Appendix B contains tables showing each model's full breakdown of feature importance by dataset variant. For scaling purposes, feature importance for each variable is converted to percentages. Figure 25 shows the top 15 feature importance’s of the dataset containing all features.

Figure 25

Full Model Feature Importances

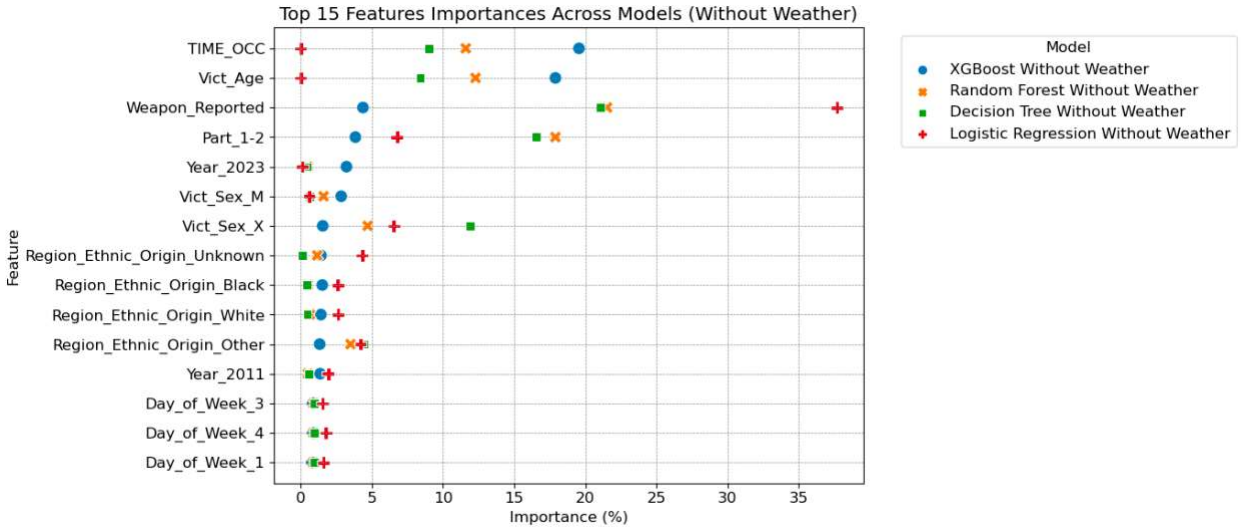


Multinomial logistic regression places much larger importance on whether a weapon is reported, and other features show minimal significance. Different models show high importance regarding weapons reported, time of occurrence, and victim age. Weather variables appear in the top 15 feature importance but contribute less than 10 percent of overall feature importance for every model. As shown in the upcoming chapters, as weather variables are removed, other features compensate for their absence in all models.

When weather variables are removed, victim demographics have a stronger presence in the top 15 features of importance for each model. Figures 26 and 27 show the top 15 features of models when weather and victim information is absent separately. The weapon reported and the time of occurrence are still of high importance. Still, victim demographic categories yield higher percentages, especially victim age, encompassing nearly 20 percent of importance for XGBoost and close to 13 percent for Random Forests feature importance. When weather is removed, victim demographic information accounts for most of the top 15 features.

Figure 26

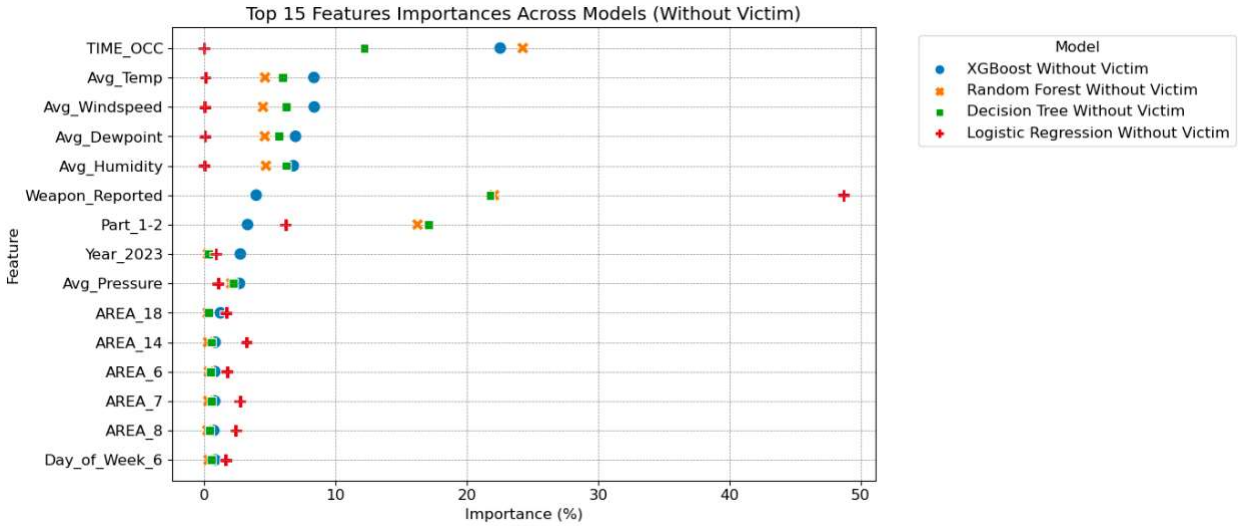
Model Performance Without Weather Features



The next dataset to analyze each model's feature importance is the variant lacking victim demographic information. Again, the time of occurrence and presence of a weapon account for a large percentage of importance, but when victim demographic information is absent, and weather information is present, each weather variable appears in the 15 most important features, each contributing less than 10 percent to any model.

Figure 27

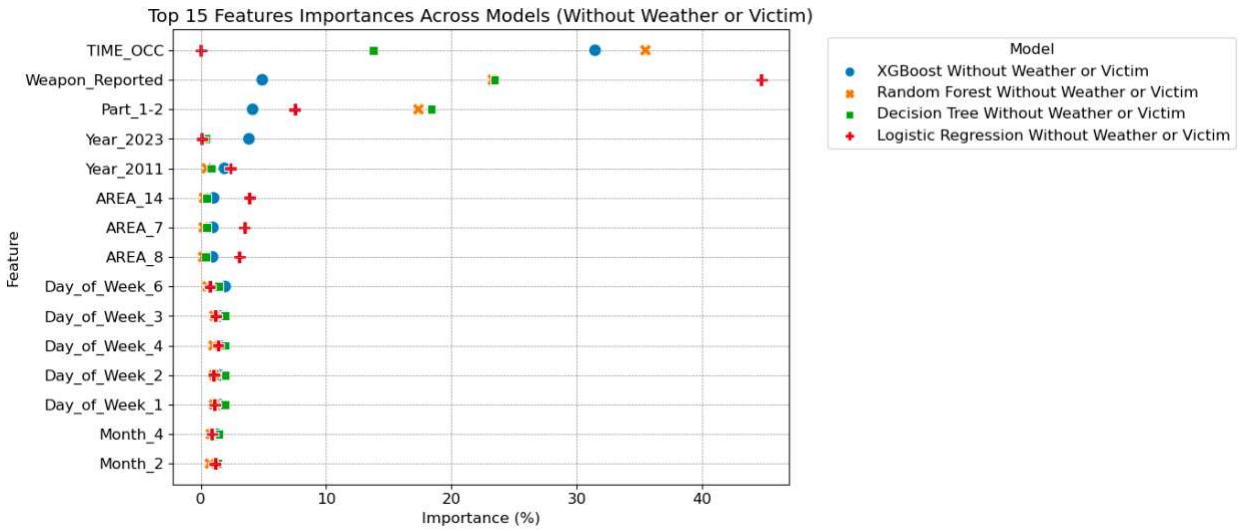
Model Performance Without Victim Features



When all hypothesis variables are removed, the baseline dataset follows the same trend with time of occurrence, weapon presence, and part 1-2 accounting for most of the overall feature importance for each model.

Figure 28

Model Performance Without Weather and Victim Features



In summary, as victim and weather information are removed, time-related variables now account for feature importance previously derived from the hypothesis variables, but minimally. The most important features in any dataset are time of occurrence, reporting of a weapon, and part 1-2. When victim demographic information is present, a large percentage of feature importance is composed of victim sex, ethnic origin, gender, and age. Compared to model accuracy across each dataset, the decline in accuracy when these variables are absent indicates their importance in predicting crime categories. For weather information, although the variables are present in the top 15 features in the full dataset and dataset without victim information, when these variables are removed, the predictive accuracy changes minimally, as shown in Appendix B.

4.4 Summary

In Chapter 4, the study employed advanced machine learning and time series forecasting techniques, notably Long Short-Term Memory (LSTM) networks, Autoregressive Integrated Moving Average (ARIMA) models, and Random Forest, Logistic Regression, XGBoost, and Decision Tree algorithms, to analyze crime prediction in Los Angeles. The chapter delved into the effectiveness of these models in predicting crime hotspots and categorizing crime types, with a specific focus on the impact of integrating diverse datasets. It was discovered that LSTM models, due to their ability to capture complex temporal dependencies, significantly outperformed ARIMA models in hotspot prediction accuracy.

Additionally, incorporating victim demographic information into the models markedly improved the overall performance of each model for crime category predictions. The role of weather information had a much lesser affect on each predictive model. Random Forests outperformed all other algorithms until training on datasets with fewer research hypothesis

variables present, which then showed each model performing about the same. Although each model favored different features, the most important ones generally remained within a few ranks of each other.

Based on these findings, we conclude that future crime hotspots can be predicted in Los Angeles, most accurately with LSTM over ARIMA. Additionally, weather information appears to have little to no impact on crime category prediction. Finally, since each model declines in performance when removing victim demographic information, it appears that victim demographic information plays a statistically significant role in crime category prediction in Los Angeles.

Chapter 5: Conclusion and Future Works

This thesis contributes to the understanding of crime prediction within Los Angeles by employing a range of machine learning and time series forecasting techniques, including Long Short-Term Memory (LSTM) networks, Autoregressive Integrated Moving Average (ARIMA) models, and algorithms such as Random Forest, XGBoost, Logistic Regression, and Decision Trees. This comprehensive approach has allowed for a nuanced examination of the intricate dynamics of crime prediction and underscored the essential role of diverse datasets in such analyses.

The investigation into time series forecasting revealed that LSTM models excel in predicting crime hotspots, a capability attributed to their ability to grasp complex temporal dependencies. This proficiency was particularly apparent in the spatiotemporal analysis conducted, where LSTM models were highly accurate in forecasting future crime incidences across Los Angeles. Refining their modeling scope to the 21 distinct areas of Los Angeles created much greater predictive accuracy and lower RMSE when compared to the five bureaus when predicting crime hotspots.

While the role of weather data in crime prediction was more subtle than that of victim demographics when classifying crime categories, it contributed to the refinement of predictive models. Regarding algorithmic performance, the Random Forest algorithm demonstrated notable accuracy, precision, and other performance metrics, delivering consistently superior outcomes across different dataset variations, and illustrating the efficacy of employing a varied suite of machine learning models in crime prediction efforts.

5.1 Future Work

To refine crime prediction models and methodologies, future research should prioritize the exploration of deep learning transformers. This involves a detailed comparison of their performance against established LSTM and ARIMA models. The objective is to evaluate whether transformers, known for their ability to handle large datasets and capture complex patterns through self-attention mechanisms, can outperform or complement existing time series and machine learning models in predicting crime patterns. Specifically, this comparison should focus on accuracy, computational efficiency, and the models' ability to integrate and analyze diverse data types, including temporal and spatial information.

Additionally, a crucial step forward is the practical application of these predictive systems within law enforcement workflows, particularly through collaboration with the Los Angeles Police Department (LAPD). This involves implementing the developed models to forecast crime hotspots and trends, followed by a structured process of gathering feedback from LAPD on the models' usability, accuracy, and operational impact. This feedback loop can identify gaps between model predictions and real-world applicability, offering insights into how models might be optimized for law enforcement use. It also serves as a platform for discussing and addressing any ethical concerns or potential biases inherent in predictive policing, ensuring that the deployment of such technologies advances public safety objectives without compromising fairness or community trust.

Engaging with LAPD and other relevant stakeholders should also include discussions on integrating real-time data feeds, such as weather changes, large public events, and demographic shifts, into the predictive models to enhance their relevance and timeliness. This collaborative approach not only tests the practical utility of the models but also aligns technological

advancements with the nuanced needs and ethical considerations of urban law enforcement agencies.

5.2 Limitations

This study presents a detailed evaluation of various machine learning and time series forecasting techniques for crime prediction in Los Angeles, focusing on models such as Long Short-Term Memory (LSTM) networks and Autoregressive Integrated Moving Average (ARIMA) models, among others. Despite offering valuable insights into the predictive capabilities of these models, a deeper interpretation and discussion of the results in relation to the initial research objectives and their potential applications in real-world settings could further enrich the report. Specifically, understanding how these models can be integrated into existing public safety strategies and the practical implications of deploying such technologies in urban environments would offer a more comprehensive view of their value.

Addressing the limitations of the study, it's clear that the reliance on Los Angeles-specific crime data may not fully represent the complexities and variations found in other urban settings, potentially limiting the generalizability of the findings. The study's dependency on historical data also raises concerns about its ability to adapt to rapid changes in crime patterns, underscoring the need for incorporating more timely and dynamic data sources. Furthermore, the research could benefit from a more granular analysis beyond the 21 area levels of Los Angeles to include the city's over 1,300 distinct districts. Such detail could significantly enhance the precision of crime hotspot predictions and, by extension, the efficacy of targeted interventions.

The integration of real-time data elements, such as weather changes and demographic shifts, was another area identified for improvement. The absence of these factors in the predictive models may detract from their applicability in quickly evolving scenarios, pointing to

a critical area for future development. Ethical considerations and the potential for biases in the machine learning models used for crime prediction also warrant a more thorough examination. Ensuring that these technologies are applied in a manner that is both responsible and equitable is paramount to their acceptance and effectiveness in public safety efforts.

5.3 Final Thoughts

Deploying advanced data science techniques in public safety and crime prediction is a rapidly evolving area with significant potential. This thesis not only illustrates the utility of machine learning and time series forecasting in deciphering and forecasting crime patterns but also emphasizes the critical importance of comprehensive and diverse datasets in such endeavors. Looking forward, the ongoing refinement of these models, alongside the exploration of new data dimensions, promises substantial improvements in the accuracy of crime predictions. Such advancements are poised to contribute to the creation of safer and more secure communities, highlighting the profound impact of data science on public safety and community well-being.

Appendix A: Dataset Descriptive Statistics

Table 9

Merged Dataset Descriptive Statistics – Numeric Features

Feature	Count	Mean	Std	Min	25%	50%	75%	Max	Skewness	Kurtosis	% Missing	Outliers
TIME_OCC	2886530	1355.5	648.3	1.0	922.0	1430.0	1900.0	2359.0	-0.5	-0.7	0.0	0
AREA	2886530	10.8	6.0	1.0	6.0	11.0	16.0	21.0	0.0	-1.2	0.0	0
Rpt_Dist_No	2886530	1128.3	603.9	100.0	629.0	1142.0	1638.0	2199.0	0.0	-1.2	0.0	0
Part_1-2	2886530	1.4	0.5	1.0	1.0	1.0	2.0	2.0	0.3	-1.9	0.0	0
Crm_Cd	2886530	505.9	210.2	110.0	330.0	442.0	626.0	956.0	0.5	-0.8	0.0	0
Vict_Age	2886530	31.2	20.9	-12.0	19.0	31.0	46.0	118.0	0.1	-0.6	0.0	9787
Premis_Cd	2886457	311.5	212.9	101.0	102.0	210.0	501.0	976.0	0.4	-1.2	0.0	0
Weapon_Used_Cd	977023	370.3	115.7	101.0	400.0	400.0	400.0	516.0	-1.2	0.5	66.2	399463
Crm_Cd_1	2886510	505.7	210.1	110.0	330.0	442.0	626.0	999.0	0.5	-0.8	0.0	0
Crm_Cd_2	193224	952.5	120.6	122.0	998.0	998.0	998.0	999.0	-3.3	11.3	93.3	39005
Crm_Cd_3	5062	975.4	78.3	93.0	998.0	998.0	998.0	999.0	-4.7	26.8	99.8	719
Crm_Cd_4	151	979.5	69.8	421.0	998.0	998.0	998.0	999.0	-5.9	39.7	100.0	24
LAT	2886530	34.1	0.9	0.0	34.0	34.1	34.2	34.8	-37.9	1458.5	0.0	63521
LON	2886530	-118.3	3.0	-118.8	-118.4	-118.3	-118.3	0.0	38.8	1505.0	0.0	1987
Max_Temp	2886530	70.0	7.4	52.0	65.0	70.0	74.0	162.0	0.9	4.8	0.0	52225
Avg_Temp	2886530	63.1	6.1	46.8	58.6	62.8	67.4	86.9	0.2	-0.2	0.0	11120
Min_Temp	2886530	56.7	9.6	0.0	53.0	57.0	63.0	77.0	-3.0	15.6	0.0	44371
Max_Dewpoint	2886451	56.0	7.0	15.0	52.0	56.0	61.0	145.0	-0.7	3.2	0.0	54462
Avg_Dewpoint	2886451	51.3	10.2	7.9	46.4	53.6	58.6	69.7	-1.1	0.9	0.0	113543
Min_Dewpoint	2886451	44.7	16.4	-7.0	38.0	50.0	56.0	67.0	-1.3	0.9	0.0	177380
Max_Humidity	2886451	85.1	12.1	19.0	81.0	87.0	93.0	100.0	-1.7	3.5	0.0	188092
Avg_Humidity	2886451	69.4	15.9	12.5	64.5	74.0	79.7	98.2	-1.3	1.0	0.0	277641
Min_Humidity	2886451	49.9	20.2	0.0	37.0	57.0	65.0	96.0	-0.8	-0.4	0.0	0
Max_Windspeed	2886451	15.5	13.3	0.0	13.0	15.0	17.0	807.0	54.2	3222.7	0.0	122935
Avg_Windspeed	2886451	7.2	2.6	0.0	5.7	6.9	8.1	94.7	10.4	317.4	0.0	112032
Min_Windspeed	2886451	0.5	1.4	0.0	0.0	0.0	0.0	17.0	3.6	18.4	0.0	337570
Max_Pressure	2886451	29.9	0.1	29.3	29.8	29.9	30.0	30.4	0.4	0.5	0.0	7758
Avg_Pressure	2886451	29.8	0.2	26.5	29.8	29.8	29.9	30.3	-5.6	56.1	0.0	221401
Min_Pressure	2886451	29.4	3.6	0.0	29.7	29.8	29.9	30.2	-8.0	61.3	0.0	51298
Total_Precipitation	2886386	0.0	0.2	0.0	0.0	0.0	0.0	3.0	10.5	136.4	0.0	192080

Table 10

Merged Dataset Descriptive Statistics – Categorical Features

	Count	Unique	Top	Frequency	% Missing
Date_Rptd	2886530	5129	2/2/23 0:00	1571	0
DATE_OCC	2886530	5113	1/1/11	4037	0
AREA_NAME	2886530	21	77th Street	186509	0
Crm_Cd_Desc	2886530	144	BATTERY - SIMPLE ASSAULT	255818	0
Mocodes	2556356	626351	344	252688	11.44
Vict_Sex	2595169	6	M	1296688	10.09
Vict_Descent	2595112	20	H	965038	10.10
Premis_Desc	2885861	320	STREET	663160	0.023
Weapon_Desc	977022	79	STRONG-ARM (HANDS, FIST, FEET OR BODILY FORCE)	577560	66.15

Status	2886526	9	IC		2253867	0.000
Status_Desc	2886530	6	Invest Cont		2253867	0
LOCATION	2886530	80302	6TH	ST	6136	0
Cross_Street	473717	14335	BROADWAY		8096	83.59

Table 11

Cleaned Dataset Descriptive Statistics – Numeric Features

Feature	Count	Mean	Std	Min	25%	50%	75%	Max	Skew	Kurtosis	% Missing	Outliers
TIME_OCC	2883802	1355.5	648.4	1.0	921.0	1430.0	1900.0	2359.0	-0.5	-0.7	0.00	0.0
AREA	2883802	10.8	6.0	1.0	6.0	11.0	16.0	21.0	0.0	-1.2	0.00	0.0
Rpt_Dist_No	2883802	1128.5	603.9	100.0	629.0	1143.0	1638.0	2199.0	0.0	-1.2	0.00	0.0
Part_1-2	2883802	1.4	0.5	1.0	1.0	1.0	2.0	2.0	0.2	-1.9	0.00	0.0
Crm_Cd	2883802	505.8	210.2	110.0	330.0	442.0	626.0	956.0	0.5	-0.8	0.00	0.0
Vict_Age	2883802	31.2	20.9	0.0	19.0	31.0	46.0	99.0	0.1	-0.6	0.00	9,774
LAT	2883802	34.1	0.1	33.3	34.0	34.1	34.2	34.8	-0.4	0.6	0.00	63439
LON	2883802	-118.4	0.1	-118.8	-118.4	-118.3	-118.3	-117.7	-0.7	-0.3	0.00	80
Avg_Temp	2883802	63.1	6.0	46.8	58.6	62.8	67.3	86.9	0.2	-0.2	0.00	12454
Avg_Dewpoint	2883802	51.3	10.2	7.9	46.4	53.6	58.6	69.7	-1.1	0.9	0.00	113495
Avg_Humidity	2883802	69.4	15.9	12.5	64.5	74.0	79.7	98.2	-1.3	1.0	0.00	277543
Avg_Windspeed	2883802	7.1	2.2	0.0	5.7	6.9	8.1	26.1	1.5	5.4	0.00	111260
Avg_Pressure	2883802	29.8	0.2	27.5	29.8	29.8	29.9	30.3	-4.8	39.3	0.00	220575
Total_Precipitation	2883802	0.0	0.2	0.0	0.0	0.0	0.0	3.0	10.5	136.3	0.00	192017
Crime_Category_Code	2883802	3.0	2.3	0.0	1.0	3.0	5.0	7.0	-0.1	-1.5	0.00	0.0
Weapon_Reported	2883802	0.3	0.5	0.0	0.0	0.0	1.0	1.0	0.7	-1.5	0.00	0.0

Table 12

Cleaned Dataset Descriptive Statistics – Categorical Features

Feature	Count	Unique	Top	Frequency	% Missing
Date_Rptd	2883802	5129	2/2/23 0:00	1571	0
DATE_OCC	2883802	5111	1/1/11	4031	0
AREA_NAME	2883802	21	77th Street	186392	0
Crm_Cd_Desc	2883802	144	BATTERY - SIMPLE ASSAULT	255465	0
Vict_Sex	2883802	3	M	1295425	0
Vict_Descent	2883802	21	H	964118	0
LOCATION	2883802	80242	6TH ST	6130	0
Crime_Category	2883802	8	Theft and Burglary	935482	0
Region_Ethnic_Origin	2883802	6	Hispanic/Latin/Mexican	964118	0

Appendix B: Feature Importance by Dataset

Table 13

Full Dataset Feature Importance (Percentages)

Feature	XGBoost	Random Forest	Decision Trees	Logistic Regression
TIME_OCC	15.014	7.135	6.719	0.001
Vict_Age	14.074	8.718	6.063	0.050
Avg_Temp	5.754	3.730	3.736	0.065
Avg_Windspeed	5.628	3.598	3.900	0.027
Avg_Dewpoint	5.305	3.709	3.483	0.045
Avg_Humidity	5.082	3.780	3.874	0.012
Weapon_Reported	3.766	21.162	21.039	35.209
Part_1-2	3.375	18.100	16.520	6.227
Year_2023	2.581	0.358	0.331	1.236
Vict_Sex_M	2.236	1.747	0.551	2.529
Region_Ethnic_Origin_Hispanic/ Latin/Mexican	1.869	0.669	0.350	2.796
Avg_Pressure	1.860	1.733	1.444	1.004
Total_Precipitation	1.440	0.454	0.389	0.253
Vict_Sex_X	1.393	4.156	11.895	5.108
AREA_18	1.258	0.261	0.260	2.508
Region_Ethnic_Origin_Unknown	1.188	0.947	0.098	4.694
Region_Ethnic_Origin_Black	1.134	0.459	0.369	5.053
Region_Ethnic_Origin_White	1.088	0.629	0.358	0.097
AREA_12	1.037	0.309	0.310	1.022
Year_2017	1.033	0.334	0.305	1.074
Region_Ethnic_Origin_Other	0.995	3.756	4.352	0.909
Year_2011	0.909	0.320	0.336	1.179
AREA_14	0.827	0.336	0.427	1.672
Year_2018	0.781	0.319	0.302	0.114
AREA_21	0.710	0.249	0.284	0.058
Year_2019	0.697	0.291	0.286	0.799
AREA_6	0.694	0.297	0.297	0.392
AREA_13	0.655	0.234	0.265	0.495
Year_2016	0.652	0.335	0.293	0.302
Year_2021	0.628	0.174	0.179	1.283
AREA_15	0.619	0.280	0.313	0.820
AREA_5	0.599	0.227	0.236	1.747
AREA_7	0.599	0.286	0.307	1.882
Year_2015	0.595	0.314	0.276	0.243
AREA_10	0.590	0.270	0.301	0.440

AREA_8	0.588	0.247	0.272	0.818
Year_2022	0.588	0.110	0.129	1.129
AREA_3	0.584	0.308	0.306	1.217
AREA_9	0.557	0.295	0.321	1.368
AREA_17	0.531	0.258	0.301	0.877
AREA_4	0.531	0.213	0.208	0.484
Day_of_Week_5	0.524	0.421	0.388	0.517
Day_of_Week_6	0.522	0.370	0.376	1.154
AREA_16	0.520	0.236	0.242	0.255
AREA_20	0.511	0.282	0.289	0.435
AREA_19	0.478	0.272	0.290	1.259
AREA_2	0.467	0.275	0.300	0.046
Year_2013	0.462	0.291	0.269	0.913
Year_2014	0.438	0.311	0.281	0.198
Month_12	0.411	0.310	0.246	0.923
AREA_11	0.411	0.300	0.306	0.606
Year_2012	0.407	0.296	0.280	0.845
Day_of_Week_3	0.396	0.511	0.448	0.238
Day_of_Week_4	0.378	0.515	0.455	0.388
Day_of_Week_2	0.363	0.510	0.436	0.138
Month_11	0.358	0.346	0.293	0.450
Day_of_Week_1	0.349	0.507	0.451	0.162
Year_2020	0.343	0.091	0.099	1.047
Month_4	0.341	0.358	0.316	0.223
Month_6	0.330	0.349	0.315	0.011
Month_10	0.314	0.363	0.300	0.143
Month_5	0.312	0.358	0.313	0.554
Month_2	0.290	0.310	0.248	0.515
Month_7	0.272	0.325	0.270	0.181
Month_3	0.272	0.347	0.291	0.670
Month_8	0.257	0.319	0.258	0.440
Month_9	0.232	0.325	0.257	0.451

Table 14

Feature Importance of the Baseline Dataset

Feature	XGBoost	Random Forests	Decision Trees	Logistic Regression
TIME_OCC	31.490	35.515	13.758	0.002
Weapon_Reported	4.907	23.325	23.468	44.763
Part_1-2	4.136	17.384	18.428	7.519
Year_2023	3.846	0.387	0.431	0.073
AREA_18	1.345	0.210	0.336	2.337
AREA_12	1.325	0.242	0.509	0.418
Year_2017	1.761	0.544	0.893	0.487
Year_2011	1.890	0.479	0.843	2.377
AREA_14	1.030	0.247	0.472	3.878
Year_2018	1.575	0.485	0.840	0.801
AREA_21	0.873	0.158	0.352	1.569
Year_2019	1.416	0.411	0.782	0.197
AREA_6	1.043	0.237	0.463	2.492
AREA_13	0.815	0.189	0.399	0.227
Year_2016	1.351	0.588	0.957	0.442
Year_2021	1.139	0.204	0.276	0.122
AREA_15	0.798	0.210	0.453	0.254
AREA_5	0.882	0.170	0.313	0.960
AREA_7	0.970	0.221	0.460	3.494
Year_2015	1.298	0.585	0.991	0.247
AREA_10	0.815	0.223	0.455	0.809
AREA_8	0.946	0.186	0.392	3.070
Year_2022	0.837	0.117	0.199	0.044
AREA_3	0.862	0.322	0.534	1.100
AREA_9	0.831	0.226	0.464	0.313
AREA_17	0.733	0.175	0.397	0.776
AREA_4	0.760	0.209	0.395	0.272
Day_of_Week_5	1.786	0.720	1.561	0.091
Day_of_Week_6	1.947	0.539	1.435	0.721
AREA_16	0.716	0.176	0.362	0.636
AREA_20	0.618	0.239	0.376	1.849
AREA_19	0.671	0.209	0.420	0.765
AREA_2	0.868	0.232	0.450	0.863
Year_2013	0.846	0.487	0.912	1.924

Year_2014	0.926	0.553	0.983	0.735
Month_12	1.411	0.749	1.391	0.711
AREA_11	0.627	0.259	0.467	0.770
Year_2012	1.072	0.488	0.910	1.819
Day_of_Week_3	1.495	1.073	1.929	1.182
Day_of_Week_4	1.487	1.005	1.940	1.378
Day_of_Week_2	1.460	1.060	1.962	1.035
Month_11	1.283	0.755	1.427	0.643
Day_of_Week_1	1.462	1.053	1.956	1.112
Year_2020	0.558	0.096	0.179	0.053
Month_4	1.272	0.792	1.461	0.887
Month_6	1.225	0.793	1.482	0.392
Month_10	1.285	0.796	1.436	0.524
Month_5	1.161	0.790	1.453	0.181
Month_2	1.230	0.728	1.332	1.144
Month_7	1.256	0.783	1.470	0.150
Month_3	1.298	0.792	1.498	0.664
Month_8	1.165	0.807	1.484	0.323
Month_9	1.199	0.778	1.468	0.405

Table 15

Model Feature Importance of the Baseline Dataset (percentage)

Feature	XGBoost	Random Forest	Decision Trees	Logistic Regression
TIME_OCC	22.545	24.262	12.192	0.001
Avg_Temp	8.360	4.639	5.998	0.101
Avg_Windspeed	8.384	4.486	6.274	0.041
Avg_Dewpoint	6.960	4.623	5.685	0.061
Avg_Humidity	6.797	4.707	6.250	0.014
Weapon_Reported	3.967	22.077	21.773	48.689
Part_1-2	3.310	16.253	17.097	6.196
Year_2023	2.760	0.284	0.316	0.910
Avg_Pressure	2.695	2.060	2.216	1.069
Total_Precipitation	1.949	0.496	0.567	0.302
AREA_18	1.240	0.268	0.344	1.678
AREA_12	1.130	0.303	0.560	0.416
Year_2017	1.202	0.315	0.364	1.415
Year_2011	1.119	0.298	0.381	1.369
AREA_14	0.841	0.297	0.571	3.226
Year_2018	1.000	0.299	0.361	0.011
AREA_21	0.774	0.231	0.446	0.750
Year_2019	0.843	0.270	0.346	0.827
AREA_6	0.825	0.360	0.505	1.751
AREA_13	0.695	0.242	0.412	0.620
Year_2016	0.886	0.338	0.398	0.510
Year_2021	0.834	0.146	0.176	1.326
AREA_15	0.657	0.343	0.586	0.644
AREA_5	0.662	0.248	0.334	1.968
AREA_7	0.810	0.316	0.526	2.729
Year_2015	0.823	0.320	0.385	0.597
AREA_10	0.653	0.337	0.541	0.101
AREA_8	0.736	0.253	0.433	2.411
Year_2022	0.693	0.089	0.135	1.188
AREA_3	0.608	0.500	0.596	0.352
AREA_9	0.715	0.357	0.591	1.326
AREA_17	0.567	0.268	0.516	0.081
AREA_4	0.590	0.283	0.370	0.614
Day_of_Week_5	0.666	0.420	0.573	0.768
Day_of_Week_6	0.841	0.333	0.554	1.639
AREA_16	0.632	0.280	0.391	0.199
AREA_20	0.484	0.378	0.409	1.142

AREA_19	0.558	0.328	0.469	1.751
AREA_2	0.675	0.347	0.454	0.040
Year_2013	0.518	0.286	0.356	0.958
Year_2014	0.498	0.303	0.364	0.036
Month_12	0.610	0.324	0.358	1.210
AREA_11	0.475	0.419	0.518	0.211
Year_2012	0.596	0.289	0.369	0.873
Day_of_Week_3	0.590	0.554	0.695	0.376
Day_of_Week_4	0.605	0.547	0.696	0.591
Day_of_Week_2	0.505	0.555	0.698	0.216
Month_11	0.536	0.358	0.421	0.585
Day_of_Week_1	0.545	0.553	0.701	0.236
Year_2020	0.431	0.073	0.105	1.211
Month_4	0.502	0.367	0.438	0.355
Month_6	0.388	0.348	0.420	0.084
Month_10	0.440	0.371	0.433	0.243
Month_5	0.478	0.359	0.429	0.814
Month_2	0.437	0.322	0.370	0.789
Month_7	0.345	0.323	0.368	0.208
Month_3	0.377	0.356	0.413	0.972
Month_8	0.305	0.316	0.366	0.584
Month_9	0.334	0.325	0.385	0.614

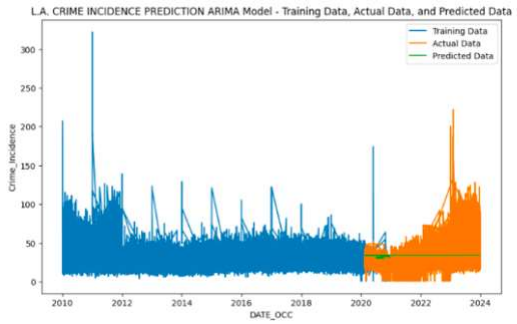
Table 16

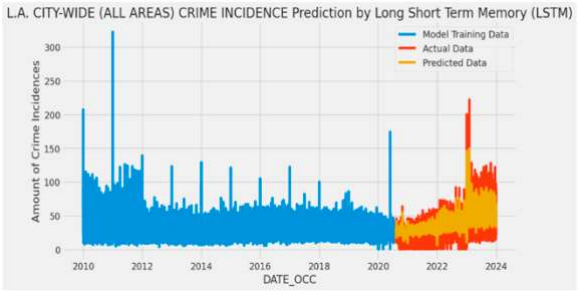
Dataset Without Weather Information Feature Importance (percentages)

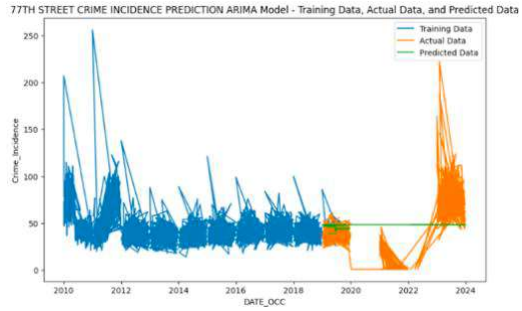
Feature	XGBoost	Random Forest	Decision Trees	Logistic Regression
TIME_OCC	19.559	11.599	9.005	0.001
Vict_Age	17.900	12.287	8.391	0.001
Weapon_Reported	4.373	21.527	21.052	37.689
Part_1-2	3.842	17.899	16.531	6.767
Year_2023	3.214	0.487	0.443	0.113
Vict_Sex_M	2.845	1.606	0.641	0.597
Region_Ethnic_Origin_Hispanic/Latin/Mexican	2.222	0.674	0.404	0.025
Vict_Sex_X	1.546	4.701	11.910	6.539
AREA_18	1.458	0.265	0.311	1.108
Region_Ethnic_Origin_Unknown	1.423	1.159	0.109	4.319
Region_Ethnic_Origin_Black	1.513	0.495	0.440	2.604
Region_Ethnic_Origin_White	1.417	0.643	0.467	2.632
AREA_12	1.148	0.324	0.368	0.854
Year_2017	1.373	0.539	0.554	0.491
Region_Ethnic_Origin_Other	1.332	3.502	4.457	4.204
Year_2011	1.364	0.500	0.578	1.945
AREA_14	0.984	0.353	0.479	1.526
Year_2018	1.122	0.493	0.538	0.659
AREA_21	0.848	0.251	0.358	0.650
Year_2019	0.999	0.435	0.496	0.305
AREA_6	0.792	0.319	0.379	0.526
AREA_13	0.711	0.238	0.321	0.520
Year_2016	0.940	0.547	0.558	0.873
Year_2021	0.811	0.245	0.276	0.015
AREA_15	0.641	0.285	0.370	0.423
AREA_5	0.750	0.244	0.298	0.223
AREA_7	0.728	0.295	0.372	1.257
Year_2015	0.942	0.526	0.541	0.715
AREA_10	0.673	0.270	0.365	0.362
AREA_8	0.717	0.253	0.340	0.940
Year_2022	0.669	0.142	0.182	0.016
AREA_3	0.818	0.329	0.383	0.882
AREA_9	0.593	0.313	0.402	0.203
AREA_17	0.656	0.258	0.367	0.541
AREA_4	0.632	0.226	0.277	0.000
Day_of_Week_5	0.947	0.691	0.788	0.762
Day_of_Week_6	1.039	0.562	0.712	0.119

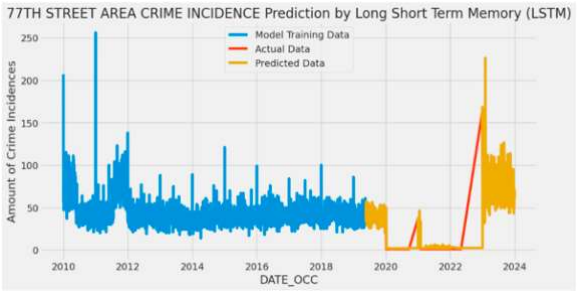
AREA_16	0.656	0.245	0.311	0.237
AREA_20	0.604	0.301	0.362	0.474
AREA_19	0.564	0.280	0.379	0.054
AREA_2	0.660	0.292	0.385	0.110
Year_2013	0.610	0.482	0.507	1.165
Year_2014	0.606	0.530	0.526	0.735
Month_12	0.767	0.684	0.730	0.867
AREA_11	0.536	0.316	0.379	0.469
Year_2012	0.658	0.482	0.515	1.205
Day_of_Week_3	0.840	0.905	0.930	1.539
Day_of_Week_4	0.859	0.904	0.955	1.763
Day_of_Week_2	0.752	0.897	0.935	1.536
Month_11	0.746	0.694	0.732	0.781
Day_of_Week_1	0.787	0.887	0.929	1.593
Year_2020	0.444	0.120	0.154	0.152
Month_4	0.676	0.725	0.746	0.783
Month_6	0.752	0.730	0.783	0.574
Month_10	0.757	0.737	0.768	0.669
Month_5	0.634	0.734	0.762	0.521
Month_2	0.691	0.658	0.698	0.985
Month_7	0.763	0.733	0.770	0.505
Month_3	0.660	0.724	0.749	0.735
Month_8	0.676	0.743	0.792	0.583
Month_9	0.761	0.714	0.739	0.561

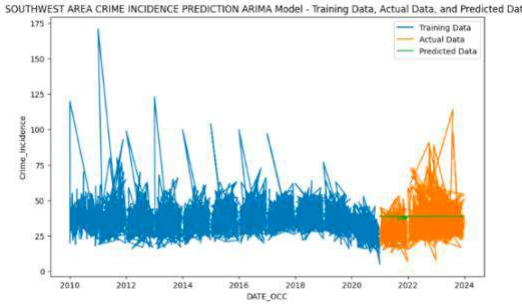
Appendix C: ARIMA and LSTM Prediction Charts

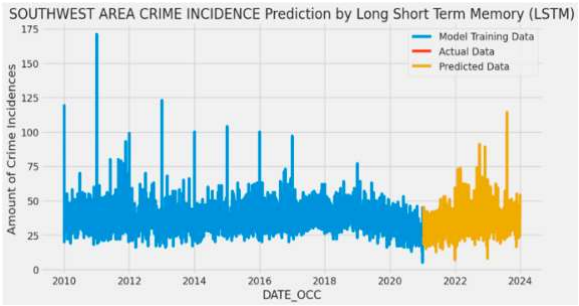


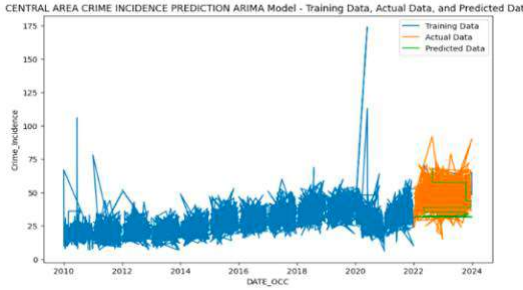


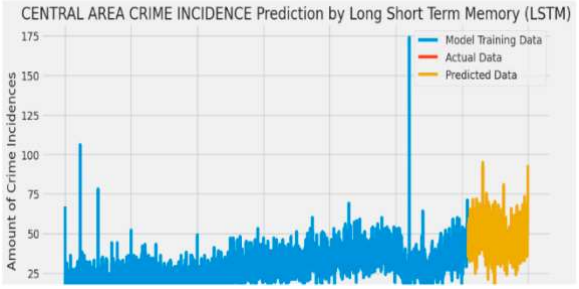


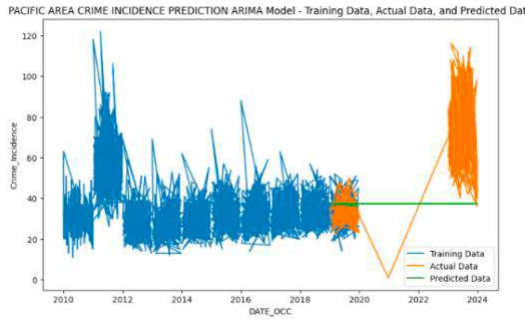


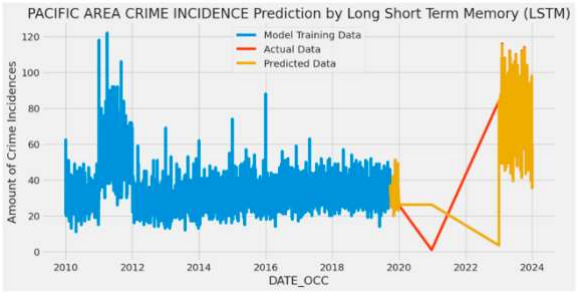


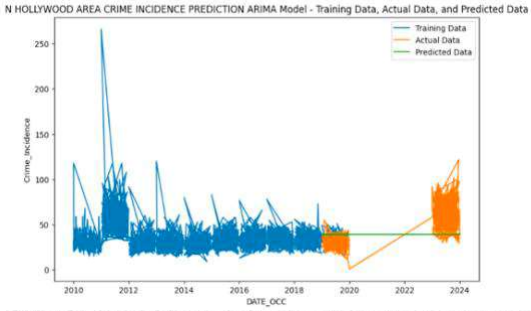


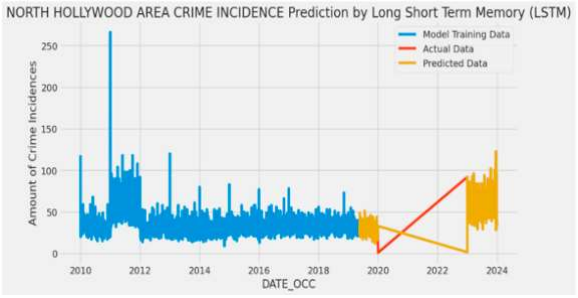


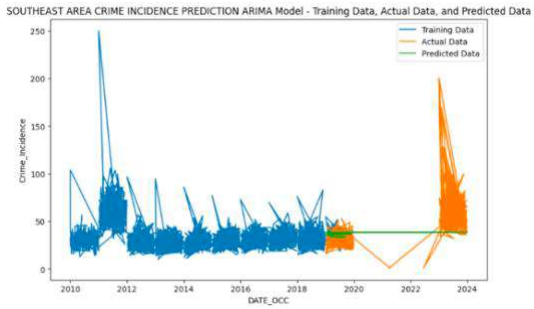


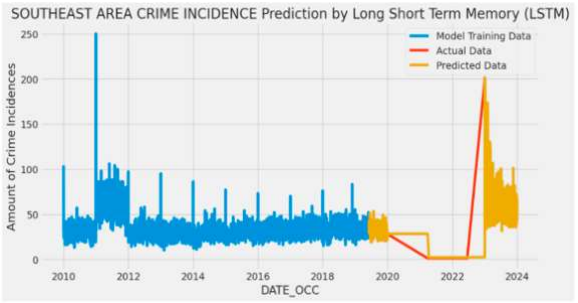


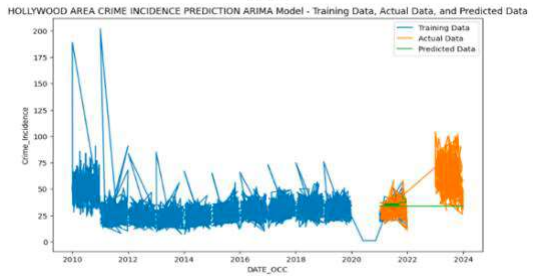


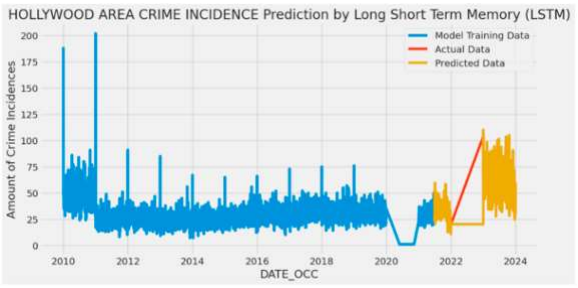

N HOLLYWOOD AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

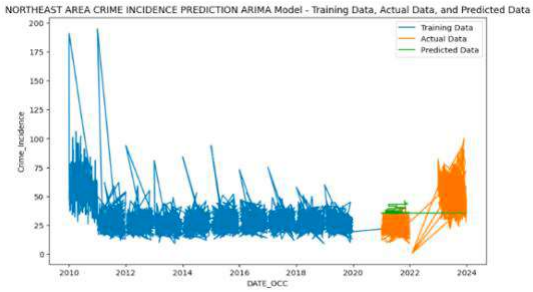

NORTH HOLLYWOOD AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

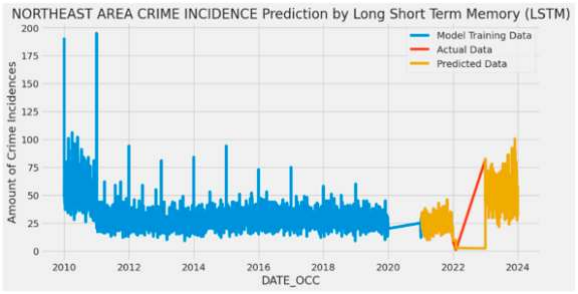

SOUTHEAST AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

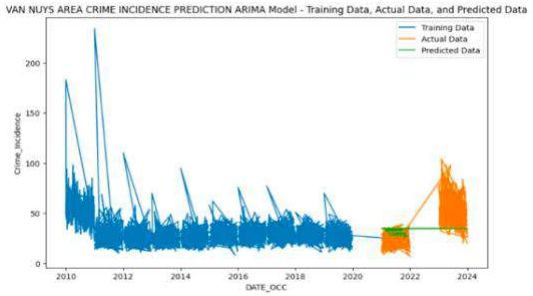

SOUTHEAST AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

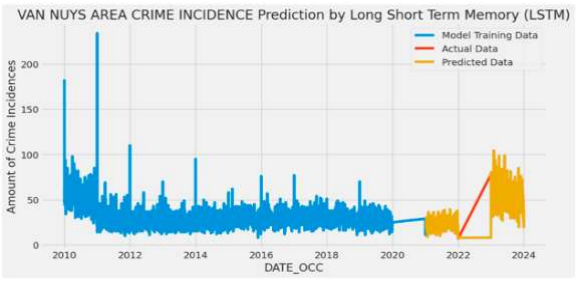

HOLLYWOOD AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

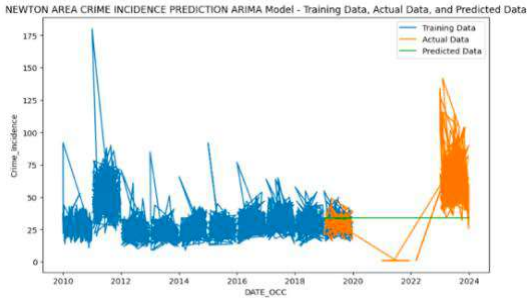

HOLLYWOOD AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

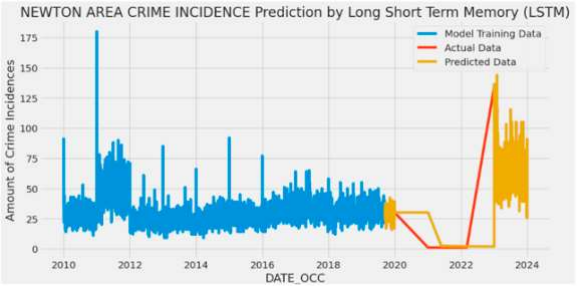

NORTHEAST AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

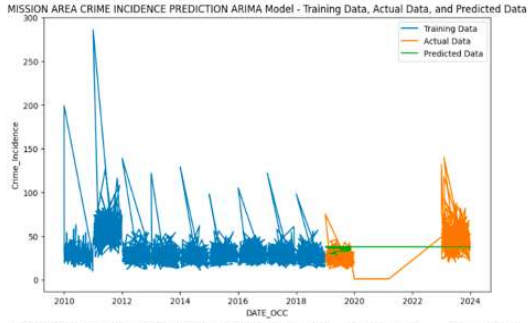

NORTHEAST AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

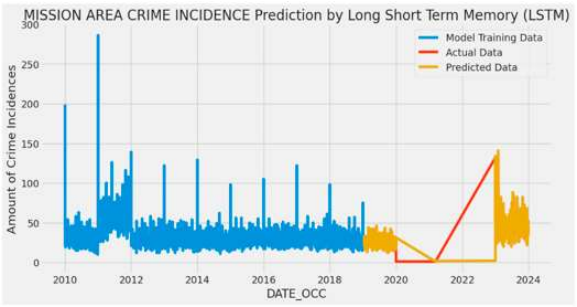

VAN NUYS AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

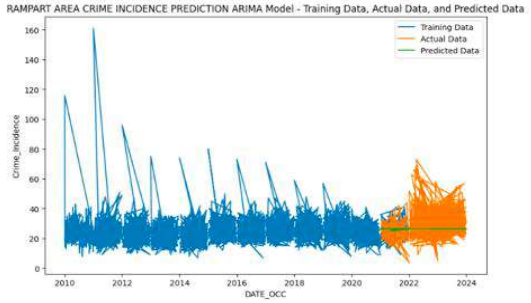

VAN NUYS AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

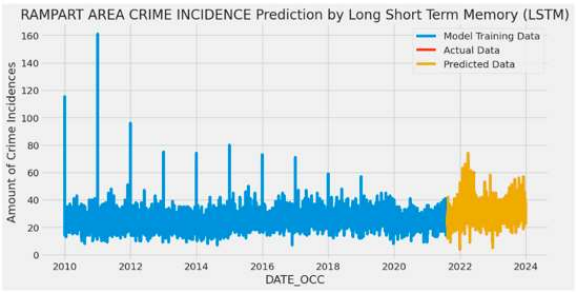

NEWTON AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

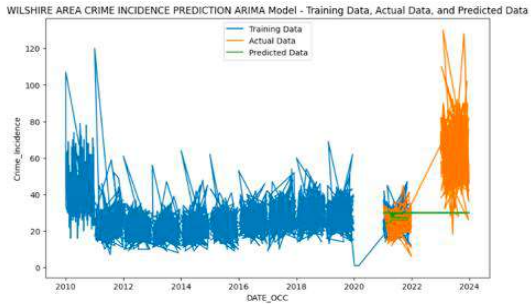

NEWTON AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

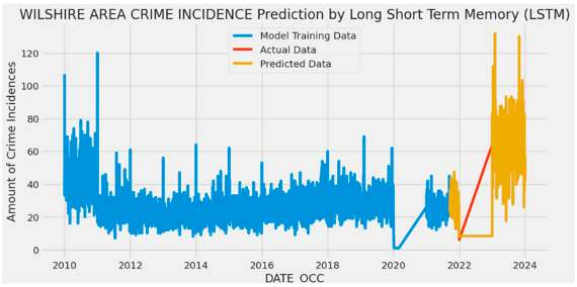

MISSION AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

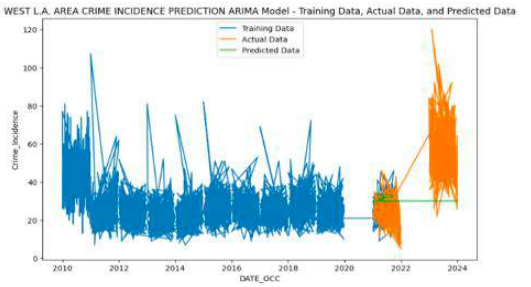

MISSION AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

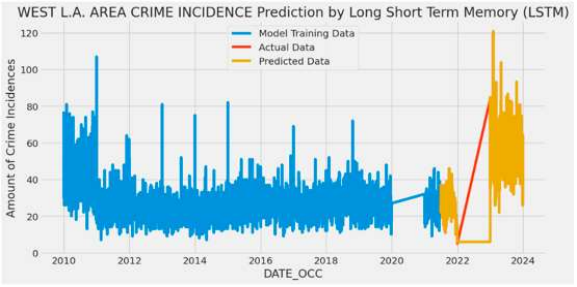

RAMPART AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

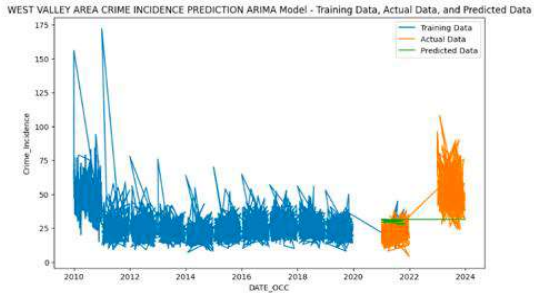

RAMPART AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

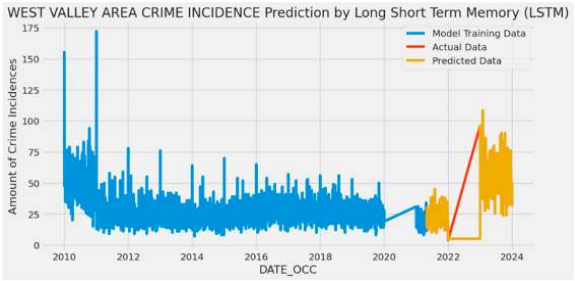

WILSHIRE AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

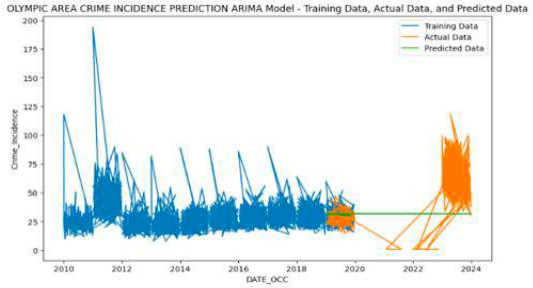

WILSHIRE AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

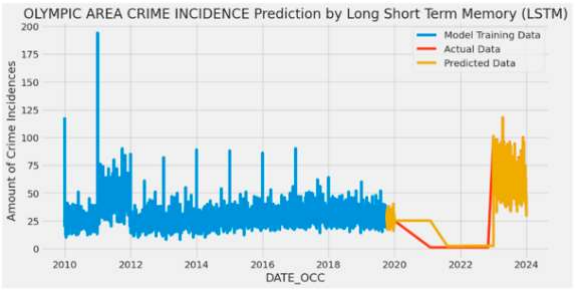

WEST L.A. AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

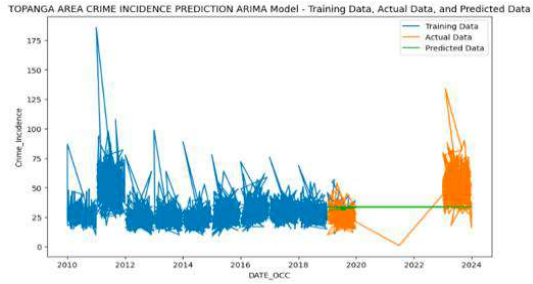

WEST L.A. AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

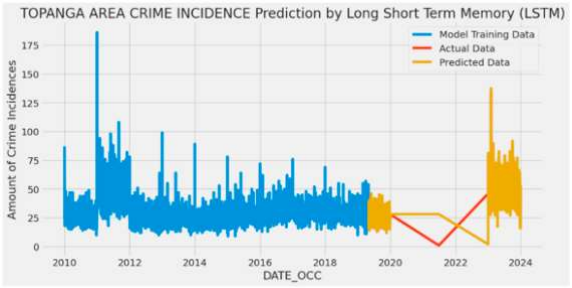

WEST VALLEY AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

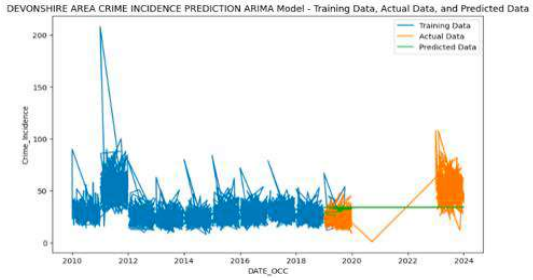

WEST VALLEY AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

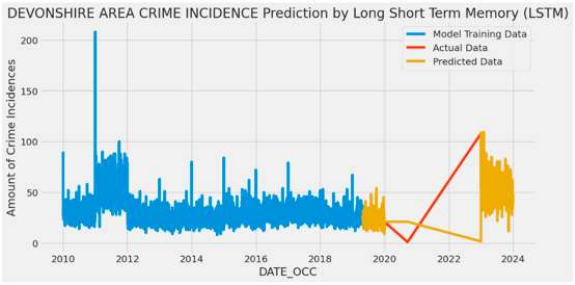

OLYMPIC AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

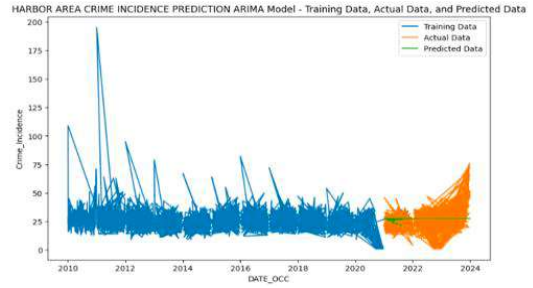

OLYMPIC AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

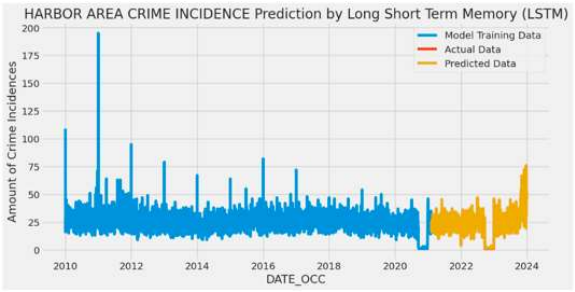

TOPANGA AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

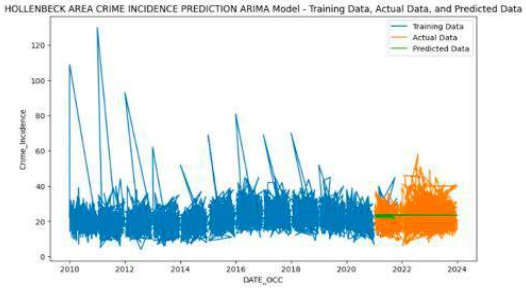

TOPANGA AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

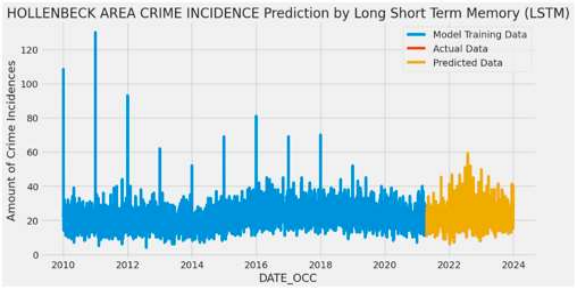

DEVONSHIRE AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC

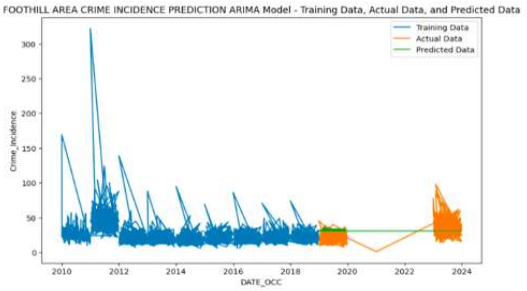

DEVONSHIRE AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

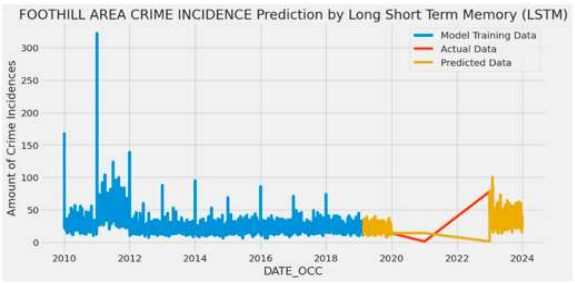

HARBOR AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC


HARBOR AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC


HOLLENBECK AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC


HOLLENBECK AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC


FOOTHILL AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data
Training Data
Actual Data
Predicted Data
Crime_Incidence
DATE_OCC


FOOTHILL AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)
Model Training Data
Actual Data
Predicted Data
Amount of Crime Incidences
DATE_OCC

Appendix D: LSTM Training and Validation Loss Charts

In chronological order from left to right, top to bottom:- 'L.A. City-Wide', '77th Street', 'Southwest', 'Central', 'Pacific', 'N Hollywood', 'Southeast', 'Hollywood', 'Northeast', 'Van Nuys', 'Newton', 'Mission', 'Rampart', 'Wilshire', 'West L.A.', 'West Valley', 'Olympic', 'Topanga', 'Devonshire', 'Harbor', 'Hollenbeck', 'Foothill'


Training and Validation Loss
train
validation
Loss
0.000250
0.000225
0.000200
0.000175
0.000150
0.000125
0.000100
0
2
4
6
8
10
Epochs


Training and Validation Loss
train
validation
Loss
0.00018
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0
1
2
3
4
5
Epochs


Training and Validation Loss
train
validation
Loss
0.000275
0.000250
0.000225
0.000200
0.000175
0.000150
0.000125
0.000100
0
1
2
3
4
5
6
7
Epochs


Training and Validation Loss
train
validation
Loss
0.00020
0.00018
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0
1
2
3
4
5
6
7
8
Epochs


Training and Validation Loss
train
validation
Loss
0.00018
0.00016
0.00014
0.00012
0.00010
0.00008
0.00006
0.00004
0
2
4
6
8
10
Epochs


Training and Validation Loss
train
validation
Loss
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0
5
10
15
20
25
Epochs


Training and Validation Loss
train
validation
Loss
0.00050
0.00045
0.00040
0.00035
0.00030
0.00025
0
1
2
3
4
5
6
7
Epochs


Training and Validation Loss
train
validation
Loss
0.000275
0.000250
0.000225
0.000200
0.000175
0.000150
0.000125
0
2
4
6
8
10
12
Epochs

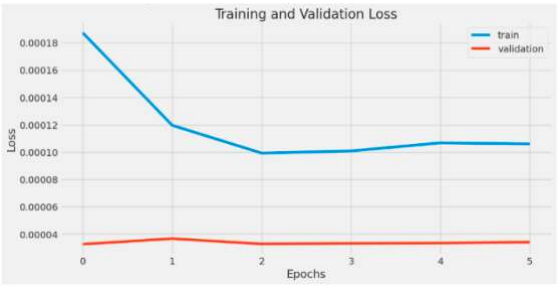

Training and Validation Loss
train
validation
Loss
0.00024
0.00022
0.00020
0.00018
0.00016
0.00014
0.00012
0.00010
0.00008
0
1
2
3
4
5
6
7
8
Epochs


Training and Validation Loss
train
validation
Loss
0.00030
0.00025
0.00020
0.00015
0.00010
0
2
4
6
8
10
12
14
Epochs


Training and Validation Loss
train
validation
Loss
Epochs


Training and Validation Loss
train
validation
Loss
Epochs


Training and Validation Loss
train
validation
Loss
Epochs


Training and Validation Loss
train
validation
Loss
Epochs

Appendix E: Python Code for ARIMA Crime Hotspot Analysis

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from statsmodels.tsa.seasonal import seasonal_decompose
import warnings
warnings.filterwarnings("ignore")

from google.colab import drive
drive.mount('/content/drive')

get_ipython().system('pip install category_encoders')
#Import dataset:

crime_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/crime_clean.csv')
print(crime_df.shape)


missing_values = crime_df.isnull().sum()
percentage_missing = (missing_values / len(crime_df)) * 100
unique_values = crime_df.nunique()
summary_df = print(pd.DataFrame({'Data_type': crime_df.dtypes,'Missing': missing_values, '%_Missing':
percentage_missing, 'Unique_values': unique_values}))
print(summary_df)


crime_df['DATE_OCC'] = pd.to_datetime(crime_df['DATE_OCC'])
crime_df.head(5)
crime_df.set_index('DATE_OCC', inplace=True)
```

```
# Convert the following into categorical

crime_df['AREA_NAME'] = crime_df['AREA_NAME'].astype('category')
crime_df['Crime_Incidence'] = crime_df.groupby(['AREA_NAME', 'DATE_OCC'])['AREA_NAME'].transform('count')
crime_df.head(20)
crime_df.shape
crime_df['AREA_NAME'].value_counts()

# Group by AREA and count the number of crimes
crimes_by_area = crime_df.groupby('AREA_NAME')['DR_NO'].count().reset_index(name='Count')

# Sort by crime count
crimes_by_area = crimes_by_area.sort_values('Count', ascending=False)

# Generate a list of colors
colors = plt.cm.viridis(np.linspace(0, 1, len(crimes_by_area)))

# Create a bar chart
plt.figure(figsize=(12, 6))
crimes_by_area.plot(kind='bar', x='AREA_NAME', y='Count', color=colors)

# Add labels and title
plt.title('Crimes by Area 2010-2023')
plt.xlabel('Area')
plt.ylabel('Number of Crimes')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the bar chart
plt.show()

split_datasets = {}

categories = crime_df['AREA_NAME'].unique()
for category in categories:
```

```
  split_datasets[category] = crime_df[crime_df['AREA_NAME'] ==category]

SeventhStreet_dataset = split_datasets['77th Street']
Southwest_dataset = split_datasets['Southwest']
Central_dataset = split_datasets['Central']
Pacific_dataset = split_datasets['Pacific']
N_Hollywood_dataset = split_datasets['N Hollywood']
Southeast_dataset = split_datasets['Southeast']
Hollywood_dataset = split_datasets['Hollywood']
Northeast_dataset = split_datasets['Northeast']
Van_Nuys_dataset = split_datasets['Van Nuys']
Newton_dataset = split_datasets['Newton']
Mission_dataset = split_datasets['Mission']
Rampart_dataset = split_datasets['Rampart']
Wilshire_dataset = split_datasets['Wilshire']
West_LA_dataset = split_datasets['West LA']
West_Valley_dataset = split_datasets['West Valley']
Olympic_dataset = split_datasets['Olympic']
Topanga_dataset = split_datasets['Topanga']
Devonshire_dataset = split_datasets['Devonshire']
Harbor_dataset = split_datasets['Harbor']
Hollenbeck_dataset = split_datasets['Hollenbeck']
Foothill_dataset = split_datasets['Foothill']


SeventhStreet_dataset.head()
crime_df.shape[0]


# **Using ARIMA to Predict Crime Incidence**
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from sklearn.metrics import mean_squared_error
from pandas.plotting import autocorrelation_plot
```

```
get_ipython().system('pip install pmdarima')
from pmdarima.arima import auto_arima

from pathlib import Path
import math
from math import floor,ceil,sqrt
import datetime as dt
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score, roc_curve, auc
import matplotlib.pylab as plt
import seaborn as sns
from sklearn import preprocessing


#  **22. Full Dataset (All Areas)**
crime_df.head(5)

# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions
```

```
# Split data into train and test sets
train_size = int(len(crime_df) * 0.8)  # Adjust the train size as needed
train_data, test_data = crime_df[:train_size]['Crime_Incidence'], crime_df[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(crime_df.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(crime_df.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(crime_df.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('L.A. CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')


# **1. ARIMA for 77th Street Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
```

```
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(SeventhStreet_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = SeventhStreet_dataset[:train_size]['Crime_Incidence'],
SeventhStreet_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(SeventhStreet_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(SeventhStreet_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(SeventhStreet_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
```

```
plt.title('77TH STREET CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **2. ARIMA for Southwest Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Southwest_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Southwest_dataset[:train_size]['Crime_Incidence'],
Southwest_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))
```

```
# Training Data
plt.plot(Southwest_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Southwest_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Southwest_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('SOUTHWEST AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **3. ARIMA for Central Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Central_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Central_dataset[:train_size]['Crime_Incidence'], Central_dataset[train_size:]['Crime_Incidence']
```

```
# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Central_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Central_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Central_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('CENTRAL AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **4. ARIMA for Pacific Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit
```

```
# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Pacific_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Pacific_dataset[:train_size]['Crime_Incidence'], Pacific_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Pacific_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Pacific_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Pacific_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('PACIFIC AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted
Data')
plt.show()
```

```
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **5. ARIMA for N Hollywood Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(N_Hollywood_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = N_Hollywood_dataset[:train_size]['Crime_Incidence'],
N_Hollywood_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(N_Hollywood_dataset.index[:train_size], train_data, label='Training Data')
```

```
# Actual Data
plt.plot(N_Hollywood_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(N_Hollywood_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('N HOLLYWOOD AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **6. ARIMA for Southeast Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Southeast_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Southeast_dataset[:train_size]['Crime_Incidence'],
Southeast_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
```

```
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Southeast_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Southeast_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Southeast_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('SOUTHEAST AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **7. ARIMA for Hollywood Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
```

```
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Hollywood_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Hollywood_dataset[:train_size]['Crime_Incidence'],
Hollywood_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Hollywood_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Hollywood_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Hollywood_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('HOLLYWOOD AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
```

```
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **8. ARIMA for Northeast Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Northeast_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Northeast_dataset[:train_size]['Crime_Incidence'],
Northeast_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Northeast_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
```

```
plt.plot(Northeast_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Northeast_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('NORTHEAST AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **9. ARIMA for Van Nuys Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Van_Nuys_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Van_Nuys_dataset[:train_size]['Crime_Incidence'],
Van_Nuys_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
```

```
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Van_Nuys_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Van_Nuys_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Van_Nuys_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('VAN NUYS AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **10. ARIMA for Newton Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
```

```
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Newton_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Newton_dataset[:train_size]['Crime_Incidence'], Newton_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Newton_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Newton_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Newton_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('NEWTON AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)
```

```
# **11. ARIMA for Mission Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Mission_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Mission_dataset[:train_size]['Crime_Incidence'], Mission_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Mission_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Mission_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
```

```
plt.plot(Mission_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('MISSION AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **12. ARIMA for Rampart Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Rampart_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Rampart_dataset[:train_size]['Crime_Incidence'],
Rampart_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
```

```
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Rampart_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Rampart_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Rampart_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('RAMPART AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **13. ARIMA for Wilshire Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
```

```
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Wilshire_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Wilshire_dataset[:train_size]['Crime_Incidence'],
Wilshire_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Wilshire_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Wilshire_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Wilshire_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('WILSHIRE AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)
```

```
# **14. ARIMA for West LA Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(West_LA_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = West_LA_dataset[:train_size]['Crime_Incidence'],
West_LA_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(West_LA_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(West_LA_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(West_LA_dataset.index[train_size:], forecast, label='Predicted Data')
```

```
plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('WEST L.A. AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **15. ARIMA for West Valley Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(West_Valley_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = West_Valley_dataset[:train_size]['Crime_Incidence'],
West_Valley_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)
```

```
# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(West_Valley_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(West_Valley_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(West_Valley_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('WEST VALLEY AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **16. ARIMA for Olympic Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions
```

```
# Split data into train and test sets
train_size = int(len(Olympic_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Olympic_dataset[:train_size]['Crime_Incidence'],
Olympic_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Olympic_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Olympic_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Olympic_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('OLYMPIC AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **17. ARIMA for Topanga Area**
```

```
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Topanga_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Topanga_dataset[:train_size]['Crime_Incidence'],
Topanga_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Topanga_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Topanga_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Topanga_dataset.index[train_size:], forecast, label='Predicted Data')
```

```
plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('TOPANGA AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **18. ARIMA for Devonshire Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Devonshire_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Devonshire_dataset[:train_size]['Crime_Incidence'],
Devonshire_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)
```

```
# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Devonshire_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Devonshire_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Devonshire_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('DEVONSHIRE AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **19. ARIMA for Harbor Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions
```

```
# Split data into train and test sets
train_size = int(len(Harbor_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Harbor_dataset[:train_size]['Crime_Incidence'], Harbor_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Harbor_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Harbor_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Harbor_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('HARBOR AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **20. ARIMA for Hollenbeck Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
```

```
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Hollenbeck_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Hollenbeck_dataset[:train_size]['Crime_Incidence'],
Hollenbeck_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))

# Training Data
plt.plot(Hollenbeck_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Hollenbeck_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Hollenbeck_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
```

```
plt.legend()
plt.title('HOLLENBECK AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)


# **21. ARIMA for Foothill Area**
# Function to fit ARIMA model and make predictions
def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

# Function to evaluate ARIMA model
def evaluate_arima(model_fit, test_data):
    predictions = model_fit.forecast(steps=len(test_data))
    mse = mean_squared_error(test_data, predictions)
    rmse = np.sqrt(mse)
    return rmse, predictions

# Split data into train and test sets
train_size = int(len(Foothill_dataset) * 0.8)  # Adjust the train size as needed
train_data, test_data = Foothill_dataset[:train_size]['Crime_Incidence'], Foothill_dataset[train_size:]['Crime_Incidence']

# Fit the ARIMA model with p=1, d=0, q=2 using the training data
order = (1, 0, 2)
model_fit = fit_arima(train_data, order)

# Evaluate the model on the test set
rmse, forecast = evaluate_arima(model_fit, test_data)

# Plot the results
plt.figure(figsize=(10, 6))
```

```
# Training Data
plt.plot(Foothill_dataset.index[:train_size], train_data, label='Training Data')

# Actual Data
plt.plot(Foothill_dataset.index[train_size:], test_data, label='Actual Data')

# Predicted Data
plt.plot(Foothill_dataset.index[train_size:], forecast, label='Predicted Data')

plt.xlabel('DATE_OCC')
plt.ylabel('Crime_Incidence')
plt.legend()
plt.title('FOOTHILL AREA CRIME INCIDENCE PREDICTION ARIMA Model - Training Data, Actual Data, and
Predicted Data')
plt.show()
# Print the Root Mean Squared Error (RMSE) on the test set
print(f'Root Mean Squared Error on Test Set: {rmse}')
print('RMSE: %.2f' % rmse)
```

Appendix F: Python Code for LSTM Crime Hotspot Analysis

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report
from statsmodels.tsa.seasonal import seasonal_decompose
import warnings
warnings.filterwarnings("ignore")

from google.colab import drive
drive.mount('/content/drive')
get_ipython().system('pip install category_encoders')

#Import dataset:
crime_df = pd.read_csv('/content/drive/My Drive/Colab Notebooks/crime_clean.csv')
print(crime_df.shape)
crime_df.columns
print(f"Dataset shape :{crime_df.shape}\n")

missing_values = crime_df.isnull().sum()
percentage_missing = (missing_values / len(crime_df)) * 100
unique_values = crime_df.nunique()
summary_df = print(pd.DataFrame({'Data_type': crime_df.dtypes,'Missing': missing_values, '%_Missing':
percentage_missing, 'Unique_values': unique_values}))
print(summary_df)

crime_df['DATE_OCC'] = pd.to_datetime(crime_df['DATE_OCC'])
crime_df.set_index('DATE_OCC', inplace=True)
```

```
# Convert the following into categorical
crime_df['AREA_NAME'] = crime_df['AREA_NAME'].astype('category')
crime_df['Crime_Incidence'] = crime_df.groupby(['AREA_NAME', 'DATE_OCC'])['AREA_NAME'].transform('count')
print(crime_df.columns)
crime_df.dtypes
crime_df.head(5)
crime_df.shape
crime_df['AREA_NAME'].value_counts()

# Group by AREA and count the number of crimes
crimes_by_area = crime_df.groupby('AREA_NAME')['DR_NO'].count().reset_index(name='Count')

# Sort by crime count
crimes_by_area = crimes_by_area.sort_values('Count', ascending=False)

# Generate a list of colors
colors = plt.cm.viridis(np.linspace(0, 1, len(crimes_by_area)))

# Create a bar chart
plt.figure(figsize=(12, 6))
crimes_by_area.plot(kind='bar', x='AREA_NAME', y='Count', color=colors)

# Add labels and title
plt.title('Crimes by Area 2010-2023')
plt.xlabel('Area')
plt.ylabel('Number of Crimes')

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Show the bar chart
plt.show()

split_datasets = {}

categories = crime_df['AREA_NAME'].unique()
for category in categories:
```

```
  split_datasets[category] = crime_df[crime_df['AREA_NAME'] ==category]

SeventhStreet_dataset = split_datasets['77th Street']
Southwest_dataset = split_datasets['Southwest']
Central_dataset = split_datasets['Central']
Pacific_dataset = split_datasets['Pacific']
N_Hollywood_dataset = split_datasets['N Hollywood']
Southeast_dataset = split_datasets['Southeast']
Hollywood_dataset = split_datasets['Hollywood']
Northeast_dataset = split_datasets['Northeast']
Van_Nuys_dataset = split_datasets['Van Nuys']
Newton_dataset = split_datasets['Newton']
Mission_dataset = split_datasets['Mission']
Rampart_dataset = split_datasets['Rampart']
Wilshire_dataset = split_datasets['Wilshire']
West_LA_dataset = split_datasets['West LA']
West_Valley_dataset = split_datasets['West Valley']
Olympic_dataset = split_datasets['Olympic']
Topanga_dataset = split_datasets['Topanga']
Devonshire_dataset = split_datasets['Devonshire']
Harbor_dataset = split_datasets['Harbor']
Hollenbeck_dataset = split_datasets['Hollenbeck']
Foothill_dataset = split_datasets['Foothill']

SeventhStreet_dataset.head()


# **Using Neural Network's LSTM in TensorFlow to Predict Crime Incidence (Hotspot)**
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(rc={'figure.figsize':(16,8)})
sns.set(font_scale=1.3)
plt.style.use('fivethirtyeight')
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense, Dropout, LSTM, RepeatVector, TimeDistributed
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import mean_squared_error
import math
from math import floor,ceil,sqrt
import datetime as dt
import warnings
warnings.filterwarnings('ignore')
from keras.optimizers import Adam


# **22. LSTM for Full Data (All Bureaus)**
def lstm_prediction(crime_df, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = crime_df.shape[0]
    df_new = crime_df[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------L.A. CITY-WIDE (ALL AREAS) CRIME INCIDENCE PREDICTION BY LONG SHORT TERM
MEMORY (LSTM)----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
```

```
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('L.A. CITY-WIDE (ALL AREAS) CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(crime_df)

# **1. LSTM for 77th Street Area**
def lstm_prediction(SeventhStreet_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = SeventhStreet_dataset.shape[0]
    df_new = SeventhStreet_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by
date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------77TH STREET AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
```

```
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('77TH STREET AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(SeventhStreet_dataset)


# **2. LSTM for Southwest Area**
def lstm_prediction(Southwest_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Southwest_dataset.shape[0]
    df_new = Southwest_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------------------')
    print('----------SOUTHWEST AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
```

```
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')
```

```
    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('SOUTHWEST AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Southwest_dataset)


# **3. LSTM for Central Area**
def lstm_prediction(Central_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Central_dataset.shape[0]
    df_new = Central_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]


    print('-----------------------------------------------------------------------------------------')
    print('----------CENTRAL AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)----
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

```
    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')
```

```
    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('CENTRAL AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Central_dataset)

# **4. LSTM for Pacific Area**
def lstm_prediction(Pacific_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Pacific_dataset.shape[0]
    df_new = Pacific_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------PACIFIC AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)------
---------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

```
    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')
```

```
    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('PACIFIC AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Pacific_dataset)


# **5. LSTM for N Hollywood Area**
def lstm_prediction(N_Hollywood_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = N_Hollywood_dataset.shape[0]
    df_new = N_Hollywood_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------NORTH HOLLYWOOD AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM
MEMORY (LSTM)----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

```
    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')
```

```
    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('NORTH HOLLYWOOD AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(N_Hollywood_dataset)

**6. LSTM for Southeast Area**
def lstm_prediction(Southeast_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Southeast_dataset.shape[0]
    df_new = Southeast_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------SOUTHEAST AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
```

```
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('SOUTHEAST AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Southeast_dataset)


# **7. LSTM for Hollywood Area**
def lstm_prediction(Hollywood_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Hollywood_dataset.shape[0]
    df_new = Hollywood_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------------------')
    print('-----------HOLLYWOOD AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
```

```
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('HOLLYWOOD AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Hollywood_dataset)


# **8. LSTM for Northeast Area**
def lstm_prediction(Northeast_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Northeast_dataset.shape[0]
    df_new = Northeast_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------------------')
    print('-----------NORTHEAST AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
```

```
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape
```

```
    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('NORTHEAST AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Northeast_dataset)
# **9. LSTM for Van Nuys Area**
def lstm_prediction(Van_Nuys_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Van_Nuys_dataset.shape[0]
    df_new = Van_Nuys_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------VAN NUYS AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)--
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
```

```
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------------')
    print('----------------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
```

```
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('VAN NUYS AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Van_Nuys_dataset)


# **10. LSTM for Newton Area**
def lstm_prediction(Newton_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Newton_dataset.shape[0]
    df_new = Newton_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('-----------NEWTON AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)-----
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
```

```
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------------')
    print('----------------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
```

```
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('NEWTON AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Newton_dataset)


# **11. LSTM for Mission Area**
def lstm_prediction(Mission_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Mission_dataset.shape[0]
    df_new = Mission_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('-----------MISSION AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)-----
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
```

```
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------------')
    print('----------------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
```

```
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('MISSION AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Mission_dataset)


# **12. LSTM for Rampart Area**
def lstm_prediction(Rampart_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Rampart_dataset.shape[0]
    df_new = Rampart_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('-----------RAMPART AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)---
-----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
```

```
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------------')
    print('----------------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
```

```
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('RAMPART AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Rampart_dataset)

# **13. LSTM for Wilshire Area**
def lstm_prediction(Wilshire_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Wilshire_dataset.shape[0]
    df_new = Wilshire_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('------------WILSHIRE AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)---
------------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
```

```
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('WILSHIRE AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Wilshire_dataset)


# **14. LSTM for West L.A. Area**
def lstm_prediction(West_LA_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = West_LA_dataset.shape[0]
    df_new = West_LA_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('----------------------------------------------------------------------------------------')
    print('------------WEST L.A. AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)---
------------')
    print('----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
```

```
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('WEST L.A. AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(West_LA_dataset)


# **15. LSTM for West Valley Area**
def lstm_prediction(West_Valley_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = West_Valley_dataset.shape[0]
    df_new = West_Valley_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------WEST VALLEY AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
```

```
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('WEST VALLEY AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(West_Valley_dataset)

# **16. LSTM for Olympic Area**
def lstm_prediction(Olympic_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Olympic_dataset.shape[0]
    df_new = Olympic_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------OLYMPIC AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)----
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
```

```
    plt.ylabel('Amount of Crime Incidences')
    plt.title('OLYMPIC AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Olympic_dataset)


# **17. LSTM for Topanga Area**
def lstm_prediction(Topanga_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Topanga_dataset.shape[0]
    df_new = Topanga_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]


    print('-------------------------------------------------------------------------------------------')
    print('------------TOPANGA AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)---
------------')
    print('-------------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
```

```
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('TOPANGA AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Topanga_dataset)


# **18. LSTM for Devonshire Area**
def lstm_prediction(Devonshire_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Devonshire_dataset.shape[0]
    df_new = Devonshire_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('----------------------------------------------------------------------------------------')
    print('----------DEVONSHIRE AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
```

```
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('DEVONSHIRE AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Devonshire_dataset)


# **19. LSTM for Harbor Area**
def lstm_prediction(Harbor_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Harbor_dataset.shape[0]
    df_new = Harbor_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------HARBOR AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)-----
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')
```

```
    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
    plt.plot(train['Crime_Incidence'])
```

```
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('HARBOR AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Harbor_dataset)


# **20. LSTM for Hollenbeck Area**
def lstm_prediction(Hollenbeck_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Hollenbeck_dataset.shape[0]
    df_new = Hollenbeck_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------------------')
    print('----------HOLLENBECK AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY
(LSTM)----------')
    print('-----------------------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
    model.add(Dense(1))
```

```
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()


    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
```

```
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('HOLLENBECK AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Hollenbeck_dataset)


# **21. LSTM for Foothill Area**
def lstm_prediction(Foothill_dataset, window_size=40, lstm_units=[256, 128], epochs=25, batch_size=32):
    shape = Foothill_dataset.shape[0]
    df_new = Foothill_dataset[['Crime_Incidence']].sort_values(by='DATE_OCC')  # Ensure data is sorted by date
    dataset = df_new.values.reshape(-1,1)
    train = df_new.iloc[:ceil(shape * 0.80)]
    valid = df_new.iloc[ceil(shape * 0.80):]

    print('-----------------------------------------------------------------------------------------')
    print('----------FOOTHILL AREA CRIME INCIDENCE PREDICTION BY LONG SHORT TERM MEMORY (LSTM)---
----------')
    print('-----------------------------------------------------------------------------------------')
    print('Shape of Training Set',train.shape)
    print('Shape of Validation Set',valid.shape)
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(dataset)
    x_train, y_train = [], []
    for i in range(40,len(df_new.iloc[:ceil(shape * 0.80)].reset_index()['DATE_OCC'].map(dt.datetime.toordinal))):
        x_train.append(scaled_data[i-40:i,0])
        y_train.append(scaled_data[i,0])
    x_train, y_train = np.array(x_train), np.array(y_train)
    x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))

    model = Sequential()
    model.add(LSTM(units=256, return_sequences=True, input_shape=(x_train.shape[1],1)))
    model.add(LSTM(units=128))
```

```
    model.add(Dense(1))
    model.compile(loss='mean_squared_error', optimizer='adam')

    early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    history = model.fit(x_train, y_train, epochs=25, batch_size=32, verbose=2, validation_split=0.2,
callbacks=[early_stopping])

    plt.figure(figsize=(12, 6))
    plt.plot(history.history['loss'], label='train')
    plt.plot(history.history['val_loss'], label='validation')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Validation Loss')
    plt.legend()
    plt.show()

    inputs = df_new[len(df_new) - len(df_new.iloc[ceil(shape *
0.80):].reset_index()['DATE_OCC'].map(dt.datetime.toordinal)) - 40:].values
    inputs = inputs.reshape(-1,1)
    inputs  = scaler.transform(inputs)
    X_test = []
    for i in range(40,inputs.shape[0]):
        X_test.append(inputs[i-40:i,0])
    X_test = np.array(X_test)
    X_test = np.reshape(X_test, (X_test.shape[0],X_test.shape[1],1))
    crime_predictions = model.predict(X_test)
    crime_predictions = scaler.inverse_transform(crime_predictions)

    rms=np.sqrt(np.mean(np.power((valid-crime_predictions),2)))
    print('RMSE value on validation set:',rms)
    print('----------------------------------------------------------')
    print('----------------------------------------------------------')

    valid['Predictions'] = crime_predictions.reshape(-1)  # Ensure predictions have the right shape

    plt.figure(figsize=(12, 6))
```

```
    plt.plot(train['Crime_Incidence'])
    plt.plot(valid[['Crime_Incidence','Predictions']])
    plt.xlabel('DATE_OCC')
    plt.ylabel('Amount of Crime Incidences')
    plt.title('FOOTHILL AREA CRIME INCIDENCE Prediction by Long Short Term Memory (LSTM)')
    plt.legend(['Model Training Data','Actual Data','Predicted Data'])
    plt.show()

lstm_prediction(Foothill_dataset)
```

Appendix G: Python Code for Crime Category Classification Models

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import cohen_kappa_score, matthews_corrcoef, log_loss
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import LinearSVC
from xgboost import XGBClassifier
import xgboost as xgb

import warnings
warnings.filterwarnings('ignore')

crime_df = pd.read_csv('LA_crime2010_2023.csv')
weather_df = pd.read_csv('LA_weather2010_2023')
## Data Cleaner
```

```
# ## Merge datasets on DATE_OCC, inner join so all rows correspond
crime_df['DATE_OCC'] = pd.to_datetime(crime_df['DATE_OCC'])
weather_df['DATE_OCC'] = pd.to_datetime(weather_df['DATE_OCC'])
merged_df = pd.merge(crime_df, weather_df, on='DATE_OCC', how='inner')

# ## Outlier removal and cleaning
merged_df = merged_df[~((merged_df['LAT'] == 0) & (merged_df['LON'] == 0))]
merged_df = merged_df[(merged_df['Avg_Windspeed'] <= 80) & (merged_df['Max_Temp'] >= 20) &
(merged_df['Vict_Age'] <= 100)]
merged_df['Vict_Sex'].fillna('Unknown', inplace=True)
merged_df['Vict_Descent'].fillna('Unknown', inplace=True)
merged_df['Total_Precipitation'].fillna(merged_df['Total_Precipitation'].median(), inplace=True)

# ## Initial Variable removal of collinear or unneeded variables
columns_to_drop = ['Max_Temp', 'Min_Temp', 'Max_Dewpoint', 'Min_Dewpoint', 'Max_Humidity', 'Min_Humidity',
                   'Max_Windspeed', 'Min_Windspeed', 'Max_Pressure', 'Min_Pressure', 'Weapon_Used_Cd',
'Weapon_Desc',
                   'Crm_Cd_1',  'Crm_Cd_2', 'Crm_Cd_3', 'Crm_Cd_4','Mocodes','Cross_Street', 'Premis_Cd', 'Premis_Desc',
                   'Status', 'Status_Desc']
merged_df.drop(columns=columns_to_drop, inplace=True)

# ## Convert Data into usable format for ML models
data=merged_df

# Convert 'Date_Rptd' and 'DATE_OCC' to datetime
data['Date_Rptd'] = pd.to_datetime(data['Date_Rptd'], format='%m/%d/%Y %I:%M:%S %p')
data['DATE_OCC'] = pd.to_datetime(data['DATE_OCC'], format='%Y-%m-%d')


# Extract day of week, month, and year from 'DATE_OCC'
data['Day_of_Week'] = data['DATE_OCC'].dt.dayofweek
data['Month'] = data['DATE_OCC'].dt.month
data['Year'] = data['DATE_OCC'].dt.year

# Drop the original 'Date_Rptd' and 'DATE_OCC' columns
data = data.drop(['Date_Rptd', 'DATE_OCC'], axis=1)
```

```
# Convert 'Vict_Sex' and 'Vict_Descent' to dummy variables
categorical_to_convert = ['Vict_Sex', 'Region_Ethnic_Origin']
data = pd.get_dummies(data, columns=categorical_to_convert, drop_first=True)
data['Day_of_Week'] = data['Day_of_Week'].astype('category')
data['Month'] = data['Month'].astype('category')
data['Year'] = data['Year'].astype('category')

# Create dummy variables for these columns
data = pd.get_dummies(data, columns=['Day_of_Week', 'Month', 'Year'], drop_first=True)


# We will not convert 'LOCATION' due to its high cardinality
data.drop(['LOCATION', 'Vict_Descent', 'DR_NO' ,'Crm_Cd_Desc', 'Crime_Category', 'Crm_Cd', 'AREA_NAME',
'Rpt_Dist_No', 'LAT', 'LON'], axis=1, inplace=True)
data.columns


## Full Dataset Models
X_full = data.drop('Crime_Category_Code', axis=1)  # Features
y_full = data['Crime_Category_Code']
X_full_train, X_full_test, y_full_train, y_full_test = train_test_split(X_full, y_full, test_size=0.2, random_state=42)


xgb_full_model = XGBClassifier(n_estimators = 100, n_jobs=-1, use_label_encoder = False, eval_metric = 'mlogloss')
xgb_full_model.fit(X_full_train,  y_full_train)
y_pred_xgb_full = xgb_full_model.predict(X_full_test)

# Accuracy
accuracy = accuracy_score(y_full_test, y_pred_xgb_full)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_full_test, y_pred_xgb_full))

# Confusion Matrix
cm = confusion_matrix(y_full_test, y_pred_xgb_full)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

rf_full_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_full_model.fit(X_full_train,  y_full_train)
y_pred_rf_full = rf_full_model.predict(X_full_test)

# Accuracy
accuracy = accuracy_score(y_full_test, y_pred_rf_full)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_full_test, y_pred_rf_full))

# Confusion Matrix
cm = confusion_matrix(y_full_test, y_pred_rf_full)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

lr_full_model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100, random_state=42)
lr_full_model.fit(X_full_train,  y_full_train)
y_pred_lr_full = lr_full_model.predict(X_full_test)

# Accuracy
accuracy = accuracy_score(y_full_test, y_pred_lr_full)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_full_test, y_pred_lr_full))

# Confusion Matrix
cm = confusion_matrix(y_full_test, y_pred_lr_full)
sns.heatmap(cm, annot=True, fmt="d")
```

```
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

dt_full_model = DecisionTreeClassifier(random_state=42)
dt_full_model.fit(X_full_train,  y_full_train)
y_pred_dt_full = dt_full_model.predict(X_full_test)

# Accuracy
accuracy = accuracy_score(y_full_test, y_pred_dt_full)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_full_test, y_pred_dt_full))

# Confusion Matrix
cm = confusion_matrix(y_full_test, y_pred_dt_full)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()


## Without victim demographics
X_wo_victim = data.drop(['Crime_Category_Code', 'Vict_Sex_M', 'Vict_Sex_X', 'Region_Ethnic_Origin_Black',
      'Region_Ethnic_Origin_Hispanic/Latin/Mexican',
      'Region_Ethnic_Origin_Other', 'Region_Ethnic_Origin_Unknown',
      'Region_Ethnic_Origin_White', 'Vict_Age'], axis=1)  # Features
y_wo_victim = data['Crime_Category_Code']
X_wo_victim_train, X_wo_victim_test, y_wo_victim_train, y_wo_victim_test = train_test_split(X_wo_victim, y_wo_victim,
test_size=0.2, random_state=42)

xgb_wo_victim_model = XGBClassifier(n_estimators = 100, n_jobs=-1, use_label_encoder = False, eval_metric =
'mlogloss')
xgb_wo_victim_model.fit(X_wo_victim_train,  y_wo_victim_train)
```

```
y_pred_xgb_wo_victim = xgb_wo_victim_model.predict(X_wo_victim_test)

# Accuracy
accuracy = accuracy_score(y_wo_victim_test, y_pred_xgb_wo_victim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_victim_test, y_pred_xgb_wo_victim))

# Confusion Matrix
cm = confusion_matrix(y_wo_victim_test, y_pred_xgb_wo_victim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

rf_wo_victim_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_wo_victim_model.fit(X_wo_victim_train,  y_wo_victim_train)
y_pred_rf_wo_victim = rf_wo_victim_model.predict(X_wo_victim_test)

# Accuracy
accuracy = accuracy_score(y_wo_victim_test, y_pred_rf_wo_victim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_victim_test, y_pred_rf_wo_victim))

# Confusion Matrix
cm = confusion_matrix(y_wo_victim_test, y_pred_rf_wo_victim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

lr_wo_victim_model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100, random_state=42)
```

```
lr_wo_victim_model.fit(X_wo_victim_train,  y_wo_victim_train)
y_pred_lr_wo_victim = lr_wo_victim_model.predict(X_wo_victim_test)

# Accuracy
accuracy = accuracy_score(y_wo_victim_test, y_pred_lr_wo_victim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_victim_test, y_pred_lr_wo_victim))

# Confusion Matrix
cm = confusion_matrix(y_wo_victim_test, y_pred_lr_wo_victim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

dt_wo_victim_model = DecisionTreeClassifier(random_state=42)
dt_wo_victim_model.fit(X_wo_victim_train,  y_wo_victim_train)
y_pred_dt_wo_victim = dt_wo_victim_model.predict(X_wo_victim_test)

# Accuracy
accuracy = accuracy_score(y_wo_victim_test, y_pred_dt_wo_victim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_victim_test, y_pred_dt_wo_victim))

# Confusion Matrix
cm = confusion_matrix(y_wo_victim_test, y_pred_dt_wo_victim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

```
## Without Weather
X_wo_weather = data.drop(['Crime_Category_Code', 'Avg_Temp',
    'Avg_Dewpoint', 'Avg_Humidity', 'Avg_Windspeed', 'Avg_Pressure',
    'Total_Precipitation'], axis=1)
y_wo_weather = data['Crime_Category_Code']
X_wo_weather_train, X_wo_weather_test, y_wo_weather_train, y_wo_weather_test = train_test_split(X_wo_weather,
y_wo_weather, test_size=0.2, random_state=42)

xgb_wo_weather_model = XGBClassifier(n_estimators = 100, n_jobs=-1, use_label_encoder = False, eval_metric =
'mlogloss')
xgb_wo_weather_model.fit(X_wo_weather_train,  y_wo_weather_train)
y_pred_xgb_wo_weather = xgb_wo_weather_model.predict(X_wo_weather_test)

# Accuracy
accuracy = accuracy_score(y_wo_weather_test, y_pred_xgb_wo_weather)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weather_test, y_pred_xgb_wo_weather))

# Confusion Matrix
cm = confusion_matrix(y_wo_weather_test, y_pred_xgb_wo_weather)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()


lr_wo_weather_model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100, random_state=42)
lr_wo_weather_model.fit(X_wo_weather_train,  y_wo_weather_train)
y_pred_lr_wo_weather = lr_wo_weather_model.predict(X_wo_weather_test)

# Accuracy
accuracy = accuracy_score(y_wo_weather_test, y_pred_lr_wo_weather)
print(f"Accuracy: {accuracy:.4f}")
```

```
# Classification report
print(classification_report(y_wo_weather_test, y_pred_lr_wo_weather))

# Confusion Matrix
cm = confusion_matrix(y_wo_weather_test, y_pred_lr_wo_weather)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

rf_wo_weather_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_wo_weather_model.fit(X_wo_weather_train,  y_wo_weather_train)
y_pred_rf_wo_weather = rf_wo_weather_model.predict(X_wo_weather_test)

# Accuracy
accuracy = accuracy_score(y_wo_weather_test, y_pred_lr_wo_weather)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weather_test, y_pred_lr_wo_weather))

# Confusion Matrix
cm = confusion_matrix(y_wo_weather_test, y_pred_lr_wo_weather)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

dt_wo_weather_model = DecisionTreeClassifier(random_state=42)
dt_wo_weather_model.fit(X_wo_weather_train,  y_wo_weather_train)
y_pred_dt_wo_weather = dt_wo_weather_model.predict(X_wo_weather_test)

# Accuracy
accuracy = accuracy_score(y_wo_weather_test, y_pred_dt_wo_weather)
print(f"Accuracy: {accuracy:.4f}")
```

```
# Classification report
print(classification_report(y_wo_weather_test, y_pred_dt_wo_weather))

# Confusion Matrix
cm = confusion_matrix(y_wo_weather_test, y_pred_dt_wo_weather)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

X_wo_weathervictim = data.drop(['Crime_Category_Code', 'Vict_Sex_M', 'Vict_Sex_X', 'Region_Ethnic_Origin_Black',
    'Region_Ethnic_Origin_Hispanic/Latin/Mexican',
    'Region_Ethnic_Origin_Other', 'Region_Ethnic_Origin_Unknown',
    'Region_Ethnic_Origin_White','Crime_Category_Code', 'Avg_Temp',
    'Avg_Dewpoint', 'Avg_Humidity', 'Avg_Windspeed', 'Avg_Pressure',
    'Total_Precipitation'], axis=1)  # Features

y_wo_weathervictim = data['Crime_Category_Code']
X_wo_weathervictim_train, X_wo_weathervictim_test, y_wo_weathervictim_train, y_wo_weathervictim_test =
train_test_split(X_svml_wo_weathervictim, y_svml_wo_weathervictim, test_size=0.2, random_state=42)
xgb_wo_weathervictim_model = XGBClassifier(n_estimators = 100, n_jobs=-1, use_label_encoder = False,
eval_metric = 'mlogloss')
xgb_wo_weathervictim_model.fit(X_wo_weathervictim_train,  y_wo_weathervictim_train)
y_pred_xgb_wo_weathervictim = xgb_wo_weathervictim_model.predict(X_wo_weathervictim_test)

# Accuracy
accuracy = accuracy_score(y_wo_weathervictim_test, y_pred_xgb_wo_weathervictim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weathervictim_test, y_pred_xgb_wo_weathervictim))

# Confusion Matrix
cm = confusion_matrix(y_wo_weathervictim_test, y_pred_xgb_wo_weathervictim)
sns.heatmap(cm, annot=True, fmt="d")
```

```
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()


lr_wo_weathervictim_model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=100,
random_state=42)
lr_wo_weathervictim_model.fit(X_wo_weathervictim_train,  y_wo_weathervictim_train)
y_pred_lr_wo_weathervictim = lr_wo_weathervictim_model.predict(X_wo_weathervictim_test)

# Accuracy
accuracy = accuracy_score(y_wo_weathervictim_test, y_pred_lr_wo_weathervictim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weathervictim_test, y_pred_lr_wo_weathervictim))

# Confusion Matrix
cm = confusion_matrix(y_wo_weathervictim_test, y_pred_lr_wo_weathervictim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

rf_wo_weathervictim_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_wo_weathervictim_model.fit(X_wo_weathervictim_train,  y_wo_weathervictim_train)
y_pred_rf_wo_weathervictim = rf_wo_weathervictim_model.predict(X_wo_weathervictim_test)

# Accuracy
accuracy = accuracy_score(y_wo_weathervictim_test, y_pred_rf_wo_weathervictim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weathervictim_test, y_pred_rf_wo_weathervictim))
```

```
# Confusion Matrix
cm = confusion_matrix(y_wo_weathervictim_test, y_pred_rf_wo_weathervictim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()

dt_wo_weathervictim_model = DecisionTreeClassifier(random_state=42)
dt_wo_weathervictim_model.fit(X_wo_weathervictim_train,  y_wo_weathervictim_train)
y_pred_dt_wo_weathervictim = dt_wo_weathervictim_model.predict(X_wo_weathervictim_test)

# Accuracy
accuracy = accuracy_score(y_wo_weathervictim_test, y_pred_dt_wo_weathervictim)
print(f"Accuracy: {accuracy:.4f}")

# Classification report
print(classification_report(y_wo_weathervictim_test, y_pred_dt_wo_weathervictim))

# Confusion Matrix
cm = confusion_matrix(y_wo_weathervictim_test, y_pred_dt_wo_weathervictim)
sns.heatmap(cm, annot=True, fmt="d")
plt.title('Confusion Matrix')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()


## Cohen's Kappa, Log Loss, and Matthews Correlation Coefficient

## Random Forests
calibrated_rf_full_model = CalibratedClassifierCV(rf_full_model, method='sigmoid', cv=5)
calibrated_rf_full_model.fit(X_full_train, y_full_train)

calibrated_rf_wo_weather_model = CalibratedClassifierCV(rf_wo_weather_model, method='sigmoid', cv=5)
calibrated_rf_wo_weather_model.fit(X_wo_weather_train, y_wo_weather_train)
```

```
calibrated_rf_wo_victim_model = CalibratedClassifierCV(rf_wo_victim_model, method='sigmoid', cv=5)
calibrated_rf_wo_victim_model.fit(X_wo_victim_train, y_wo_victim_train)

calibrated_rf_wo_weathervictim_model = CalibratedClassifierCV(rf_wo_weathervictim_model, method='sigmoid',
cv=5)
calibrated_rf_wo_weathervictim_model.fit(X_wo_weathervictim_train, y_wo_weathervictim_train)


# Full model
y_pred_rf_full = calibrated_rf_full_model.predict(X_full_test)
mcc = matthews_corrcoef(y_full_test, y_pred_rf_full)
cohen_kappa = cohen_kappa_score(y_full_test, y_pred_rf_full)
y_pred_proba = calibrated_rf_full_model.predict_proba(X_full_test)
logloss = log_loss(y_full_test, y_pred_proba)
print(f"Full Model - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss:
{logloss:.4f}")

# Model without weather
y_pred_rf_wo_weather = calibrated_rf_wo_weather_model.predict(X_wo_weather_test)
mcc = matthews_corrcoef(y_wo_weather_test, y_pred_rf_wo_weather)
cohen_kappa = cohen_kappa_score(y_wo_weather_test, y_pred_rf_wo_weather)
y_pred_proba_wo_weather = calibrated_rf_wo_weather_model.predict_proba(X_wo_weather_test)
logloss = log_loss(y_wo_weather_test, y_pred_proba_wo_weather)
print(f"Without Weather - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss:
{logloss:.4f}")

# Model without victim
y_pred_rf_wo_victim = calibrated_rf_wo_victim_model.predict(X_wo_victim_test)
mcc = matthews_corrcoef(y_wo_victim_test, y_pred_rf_wo_victim)
cohen_kappa = cohen_kappa_score(y_wo_victim_test, y_rf_pred_wo_victim)
y_pred_proba_wo_victim = calibrated_rf_wo_victim_model.predict_proba(X_wo_victim_test)
logloss = log_loss(y_wo_victim_test, y_pred_proba_wo_victim)
print(f"Without Victim - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss:
{logloss:.4f}")

# Model without weather or victim
y_pred_rf_wo_weathervictim = calibrated_rf_wo_weathervictim_model.predict(X_wo_weathervictim_test)
```

```
mcc = matthews_corrcoef(y_wo_weathervictim_test, y_pred_rf_wo_weathervictim)
cohen_kappa = cohen_kappa_score(y_wo_weathervictim_test, y_pred_rf_wo_weathervictim)
y_pred_proba_wo_weathervictim = calibrated_rf_wo_weathervictim_model.predict_proba(X_wo_weathervictim_test)
logloss = log_loss(y_wo_weathervictim_test, y_pred_proba_wo_weathervictim)
print(f"Without Weather or Victim - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f},
Log Loss: {logloss:.4f}")


# ## Multinomial Logistic Regression
calibrated_lr_full_model = CalibratedClassifierCV(lr_full_model, method='sigmoid', cv=5)
calibrated_lr_full_model.fit(X_full_train, y_full_train)

calibrated_lr_wo_weather_model = CalibratedClassifierCV(lr_wo_weather_model, method='sigmoid', cv=5)
calibrated_lr_wo_weather_model.fit(X_wo_weather_train, y_wo_weather_train)

calibrated_lr_wo_victim_model = CalibratedClassifierCV(lr_wo_victim_model, method='sigmoid', cv=5)
calibrated_lr_wo_victim_model.fit(X_wo_victim_train, y_wo_victim_train)

calibrated_lr_wo_weathervictim_model = CalibratedClassifierCV(lr_wo_weathervictim_model, method='sigmoid',
cv=5)
calibrated_lr_wo_weathervictim_model.fit(X_wo_weathervictim_train, y_wo_weathervictim_train)


# Full model
y_pred_lr_full = calibrated_lr_full_model.predict(X_full_test)
mcc = matthews_corrcoef(y_full_test, y_pred_lr_full)
cohen_kappa = cohen_kappa_score(y_full_test, y_pred_lr_full)
y_pred_proba = calibrated_lr_full_model.predict_proba(X_full_test)
logloss = log_loss(y_full_test, y_pred_proba)
print(f"Full Model - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss:
{logloss:.4f}")

# Model without weather
y_pred_lr_wo_weather = calibrated_lr_wo_weather_model.predict(X_wo_weather_test)
mcc = matthews_corrcoef(y_wo_weather_test, y_pred_lr_wo_weather)
cohen_kappa = cohen_kappa_score(y_wo_weather_test, y_pred_lr_wo_weather)
y_pred_proba_wo_weather = calibrated_lr_wo_weather_model.predict_proba(X_wo_weather_test)
logloss = log_loss(y_wo_weather_test, y_pred_proba_wo_weather)
```

```
print(f"Without Weather - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss: {logloss:.4f}")

# Model without victim
y_pred_lr_wo_victim = calibrated_lr_wo_victim_model.predict(X_wo_victim_test)
mcc = matthews_corrcoef(y_wo_victim_test, y_pred_lr_wo_victim)
cohen_kappa = cohen_kappa_score(y_wo_victim_test, y_pred_lr_wo_victim)
y_pred_proba_wo_victim = calibrated_lr_wo_victim_model.predict_proba(X_wo_victim_test)
logloss = log_loss(y_wo_victim_test, y_pred_proba_wo_victim)
print(f"Without Victim - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss: {logloss:.4f}")

# Model without weather or victim
y_pred_lr_wo_weathervictim = calibrated_lr_wo_weathervictim_model.predict(X_wo_weathervictim_test)
mcc = matthews_corrcoef(y_wo_weathervictim_test, y_pred_lr_wo_weathervictim)
cohen_kappa = cohen_kappa_score(y_wo_weathervictim_test, y_pred_lr_wo_weathervictim)
y_pred_proba_wo_weathervictim = calibrated_lr_wo_weathervictim_model.predict_proba(X_wo_weathervictim_test)
logloss = log_loss(y_wo_weathervictim_test, y_pred_proba_wo_weathervictim)
print(f"Without Weather or Victim - Matthews Correlation Coefficient: {mcc:.4f}, Cohen's Kappa: {cohen_kappa:.4f}, Log Loss: {logloss:.4f}")

# ## Decision Trees
mcc = matthews_corrcoef(y_dt_full_test, y_pred_dt_full)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_dt_full_test, y_pred_dt_full)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")
y_pred_proba = dt_full_model.predict_proba(X_full_test)
logloss = log_loss(y_full_test, y_pred_proba)
print(f"Log Loss: {logloss:.4f}")

mcc = matthews_corrcoef(y_wo_weather_test, y_pred_dt_wo_weather)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_weather_test, y_pred_dt_wo_weather)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")
y_pred_proba_wo_weather = dt_wo_weather_model.predict_proba(X_wo_weather_test)
logloss = log_loss(y_wo_weather_test, y_pred_proba_wo_weather)  # Use probabilities here
print(f"Log Loss: {logloss:.4f}")
```

```
mcc = matthews_corrcoef(y_wo_victim_test, y_pred_dt_wo_victim)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_victim_test, y_pred_dt_wo_victim)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")

# Correctly use the predict_proba method for log_loss calculation
y_pred_proba_wo_victim = dt_wo_victim_model.predict_proba(X_wo_victim_test)
logloss = log_loss(y_wo_victim_test, y_pred_proba_wo_victim)  # Use probabilities here
print(f"Log Loss: {logloss:.4f}")

mcc = matthews_corrcoef(y_wo_weathervictim_test, y_pred_dt_wo_weathervictim)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_weathervictim_test, y_pred_dt_wo_weathervictim)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")

# Correctly use the predict_proba method for log_loss calculation
y_pred_proba_wo_weathervictim = dt_wo_weathervictim_model.predict_proba(X_wo_weathervictim_test)
logloss = log_loss(y_wo_weathervictim_test, y_pred_proba_wo_weathervictim)  # Use probabilities here
print(f"Log Loss: {logloss:.4f}")


## XGBoost
mcc = matthews_corrcoef(y_full_test, y_pred_xgb_full)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_full_test, y_pred_xgb_full)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")
y_pred_proba = xgb_full_model.predict_proba(X_full_test)
logloss = log_loss(y_full_test, y_pred_proba)
print(f"Log Loss: {logloss:.4f}")
print('\nw/o weather:')
mcc = matthews_corrcoef(y_wo_weather_test, y_pred_xgb_wo_weather)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_weather_test, y_pred_xgb_wo_weather)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")
y_pred_proba_wo_weather = xgb_wo_weather_model.predict_proba(X_wo_weather_test)
logloss = log_loss(y_wo_weather_test, y_pred_proba_xgb_wo_weather)  # Use probabilities here
```

```
print(f"Log Loss: {logloss:.4f}")
print('\nw/o victim:')
mcc = matthews_corrcoef(y_wo_victim_test, y_pred_xgb_wo_victim)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_victim_test, y_pred_xgb_wo_victim)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")

# Correctly use the predict_proba method for log_loss calculation
y_pred_proba_wo_victim = xgb_wo_victim_model.predict_proba(X_wo_victim_test)
logloss = log_loss(y_wo_victim_test, y_pred_proba_xgb_wo_victim)  # Use probabilities here
print(f"Log Loss: {logloss:.4f}")

print('\nw/o weather or victim')
mcc = matthews_corrcoef(y_wo_weathervictim_test, y_pred_xgb_wo_weathervictim)
print(f"Matthews Correlation Coefficient: {mcc:.4f}")
cohen_kappa = cohen_kappa_score(y_wo_weathervictim_test, y_pred_xgb_wo_weathervictim)
print(f"Cohen's Kappa: {cohen_kappa:.4f}")

# Correctly use the predict_proba method for log_loss calculation
y_pred_proba_wo_weathervictim = xgb_wo_weathervictim_model.predict_proba(X_wo_weathervictim_test)
logloss = log_loss(y_wo_weathervictim_test, y_pred_proba_wo_weathervictim)  # Use probabilities here
print(f"Log Loss: {logloss:.4f}")
```

References

Ahmed, W., Biswas, S., & Nafis, T. (2017). Performance analysis of Naïve Bayes algorithm on crime data using Rapid Miner. International Journal of Advanced Research in Computer Science, 8(5), 683-687. https://doi.org/10.26483/ijarcs.v8i5.3393

Alghamdi, D. M. (2017, October 27). A Data Mining Based Approach For Burglary Crime Rate Prediction. Retrieved March 4, 2024, from indigo.uic.edu website: https://hdl.handle.net/10027/21894

Arpa, K. F., Mittra, T., Ferdous, T., Jahan, N., Khan Tayna, R. A., Hasan, M., … Ali, M. S. (2023). A Machine Learning and Deep Learning Integrated Model to Detect Criminal Activities. *2023 4th International Conference on Big Data Analytics and Practices (IBDAP)*. https://doi.org/10.1109/ibdap58581.2023.10272002

Babakura, A., Sulaiman, N., & Yusuf, M. (2014). *Improved Method of Classification Algorithms for Crime Prediction*. Retrieved from https://ieeexplore.ieee.org/abstract/document/7013130

Bell, A., Solano-Kamaiko, I., Nov, O., & Stoyanovich, J. (2022). It's Just Not That Simple: An Empirical Study of the Accuracy-Explainability Trade-off in Machine Learning for Public Policy. *2022 ACM Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3531146.3533090

Bhadri, M. (2023). The Effect of Demographic Imbalance on Crime. *Theses*. Retrieved from https://repository.rit.edu/theses/11496/

Bureau of Economic Analysis. (n.d.). *What is the Interactive Data Application?*. Wayback Machine. https://web.archive.org/web/20180913114415

Chicco, D., Tötsch, N., Jurman, G. (2021). The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining, 14*(1), Article 13. https://doi.org/10.1186/s13040-021-00244-z

City of Los Angeles. (n.d.). Meet your government. Departments & Bureaus. https://lacity.gov/government/departments-bureaus

Data USA. (n.d.). Los Angeles, CA. Retrieved March 4, 2024, from datausa.io website: https://datausa.io/profile/geo/los-angeles-ca#:~:text=1.89M%20people-

Ensign, D., Friedler, S. A., Neville, S., Scheidegger, C., & Venkatasubramanian, S. (2018, January 21). Runaway Feedback Loops in Predictive Policing. Retrieved from proceedings.mlr.press website: https://proceedings.mlr.press/v81/ensign18a.html

Hardyns, W., & Khalfa, R. (2022). Predicting Crime Across Cities and Regions: A Comparative Analysis of Predictive Modelling in Three Belgian Settings. *Applied Spatial Analysis and Policy*, *16*. https://doi.org/10.1007/s12061-022-09485-9

Hayes, R. (2022, August 4). LAPD shifts about 200 officers to Hollywood as residents report increase in crime. Retrieved March 4, 2024, from ABC7 Los Angeles website: https://abc7.com/hollywood-california-crime-prevention-los-angeles-police-department/12096655/

Herath, S., Dinalankara, R., Wijenayake, U. (2021). Crime Analysis, Prediction and Simulation Platform Based on Machine Learning. *2021 3rd International Conference on Advancements in Computing (ICAC)*. https://doi.org/10.1109/icac54203.2021.9671119

Hernandez, A. P. (1990). Artificial intelligence and expert systems in law enforcement: Current and potential uses. *Computers, Environment and Urban Systems*, *14*(4), 299–306. https://doi.org/10.1016/0198-9715(90)90004-d

Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). Applied Logistic Regression. In *ProQuest Ebook Central*. Newark, UNITED STATES: John Wiley & Sons, Incorporated. Retrieved from https://ebookcentral.proquest.com/lib/nu/reader.action?docID=1138225

Kanlanfeyi, S., Kishore, K., & Student, P. (2019). Using machine learning and recurrent neural networks for efficient crime detection and prevention. *Jetier*, *6*(6).

Karimi-Haghighi, M., & Castillo, C. (2021). Enhancing a recidivism prediction tool with machine learning. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*. https://doi.org/10.1145/3462757.3466150

Khalaf, E.F., Taresh, A.H. (2022). Survey: Crime Prediction using Machine Learning Approach. https://doi.org/10.29304/jqcm.2022.14.3.986

Khan, M., Ali, A., & Alharbi, Y. (2022). Predicting and Preventing Crime: A Crime Prediction Model Using San Francisco Crime Data by Classification Techniques. *Complexity*, *2022*, 1–13. https://doi.org/10.1155/2022/4830411

Kim, S., Joshi, P., Kalsi, P. S., & Taheri, P. (2018). Crime Analysis Through Machine Learning. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. https://doi.org/10.1109/iemcon.2018.8614828

Kim, S., & Lee, S. (2023). Nonlinear relationships and interaction effects of an urban environment on crime incidence: Application of urban big data and an interpretable machine learning method. *Sustainable Cities and Society*, *91*, 104419. https://doi.org/10.1016/j.scs.2023.104419

Kumar, H., & Sastry, P. S. (2018). Robust loss functions for learning multi-class classifiers. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 687–692). Miyazaki, Japan. https://doi.org/10.1109/SMC.2018.00125

Letourneau, S., Ell, N., Cheung, P., McCaskill, J., & El-Hajj, M. (2018). The effects of neighbourhood characteristics on crime incidence. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. https://doi.org/10.1109/ccwc.2018.8301675

Liang, W., Cao, J., Chen, L., Wang, Y., Wu, J., Beheshti, A., & Tang, J. (2023). Crime Prediction With Missing Data Via Spatiotemporal Regularized Tensor Decomposition. *IEEE Transactions on Big Data*, *9*(5), 1392–1407. https://doi.org/10.1109/tbdata.2023.3283098

Lin, X., Rivenson, Y., Yardimci, N. T., Veli, M., Luo, Y., Jarrahi, M., & Ozcan, A. (2018). All-optical machine learning using diffractive deep neural networks. *Science*, *361*(6406), 1004–1008. https://doi.org/10.1126/science.aat8084

Lin, Y., Chen, T., & Yu, L. (2017, July 1). Using Machine Learning to Assist Crime Prevention. https://doi.org/10.1109/IIAI-AAI.2017.46

Los Angeles, California Travel Weather Averages (Weatherbase). (n.d.). Retrieved March 12, 2024, from Weatherbase website: https://www.weatherbase.com/weather/weather.php3?s=159227&refer=

Los Angeles - Open Data Portal. (n.d.). Crime Data from 2010 to 2019. Retrieved from data.lacity.org website: https://data.lacity.org/Public-Safety/Crime-Data-from-2010-to-2019/63jg-8b9z/about_data

Los Angeles - Open Data Portal. (n.d.). Crime Data from 2020 to Present. Retrieved from data.lacity.org website: https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data

Mandalapu, V., Elluri, L., Vyas, P., & Roy, N. (2023). Crime Prediction Using Machine Learning and Deep Learning: A Systematic Review and Future Directions. *IEEE Access*, *11*, 60153–60170. https://doi.org/10.1109/access.2023.3286344

Mehsen, R. S. (2023). Deep Learning Algorithm for Detecting and Analyzing Criminal Activity. *International Journal of Computing*, *22*(2), 248–253. https://doi.org/10.47839/ijc.22.2.3095

Meskela, T. E., Afework, Y. K., Ayele, N. A., Teferi, M. W., & Mengist, T. B. (2020). Designing Time Series Crime Prediction Model using Long Short-Term Memory Recurrent Neural Network. *International Journal of Recent Technology and Engineering (IJRTE)*, *9*(4), 402–405. https://doi.org/10.35940/ijrte.d5025.119420

Muñoz, V., Vallejo, M., & Aedo, J. E. (2021, August 1). Machine Learning Models for Predicting Crime Hotspots in Medellin City. https://doi.org/10.1109/SCLA53004.2021.9540132

Nahm, F. S. (2022). Receiver operating characteristic curve: Overview and practical use for clinicians. *Korean Journal of Anesthesiology*, *75*(1), 25–36. https://doi.org/10.4097/kja.21209

Price, C. (2021, September 23). America's 10 Most Visited Cities. Retrieved from WorldAtlas website: https://www.worldatlas.com/cities/america-s-10-most-visited-cities.html

Raji, I., & Abass, A. (2023). Impact of National Indicators on Crime Prediction using Machine Learning Models. *ProQuest*. https://doi.org/10.21872/2023IISE_2069

Rehmer, A., & Kroll, A. (2020). On the vanishing and exploding gradient problem in Gated Recurrent Units. *IFAC-PapersOnLine*, *53*(2), 1243–1248. https://doi.org/10.1016/j.ifacol.2020.12.1342

Richest City in the World. (2024). Retrieved from worldpopulationreview.com website: https://worldpopulationreview.com/world-city-rankings/richest-city-in-the-world

Roshankar, R., Keyvanpour, M.R. (2023). Spatio-Temporal Graph Neural Networks for Accurate Crime Prediction. *IEEE*. https://doi.org/10.1109/iccke60553.2023.10326223

Rummens, A., & Hardyns, W. (2021). The effect of spatiotemporal resolution on predictive policing model performance. *International Journal of Forecasting*, *37*(1). https://doi.org/10.1016/j.ijforecast.2020.03.006

Sabakhtarishvili, Z., Panday, S., Jensen, C., & Ghosh, R. (2023). Crime Prediction Using Machine Learning: The Case of The City of Little Rock. *IEEE*. https://doi.org/10.1109/hora58378.2023.10156662

Safat, W., Asghar, S., & Gillani, S. A. (2021). Empirical Analysis for Crime Prediction and Forecasting Using Machine Learning and Deep Learning Techniques. *IEEE Access*, *9*, 1–1. https://doi.org/10.1109/access.2021.3078117

Samat, C., Chaijaroen, S., Kanjug, I., Vongtathum, P. (2017). Design and Development of Constructivist Multimedia Learning Environment Enhancing Skills in Computer Programming. *IEEE*. https://doi.org/10.1109/iiai-aai.2017.167

Saradha, M., Priyan, T.N., Shreeram, D. U., Viknesh, S. (2023). Crime Type Prediction based on Various Occurrence using Parallel LSTM. *IEEE*. https://doi.org/10.1109/icscss57650.2023.10169580

Sezgin, S. (2023, August 12). XGBoost. Retrieved March 4, 2024, from Medium website: https://medium.com/@sedasezgin/xgboost-c2eea332aaf2

Sharma, H. K., Choudhury, T., & Kandwal, A. (2021). Machine learning based analytical approach for geographical analysis and prediction of Boston City crime using geospatial dataset. *GeoJournal*. https://doi.org/10.1007/s10708-021-10485-4

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Solomon, A., Kertis, M., Shapira, B., & Rokach, L. (2022). A deep learning framework for predicting burglaries based on multiple contextual factors. *Expert Systems with Applications*, *199*, 117042. https://doi.org/10.1016/j.eswa.2022.117042

Stec, A., & Klabjan, D. (2018). *Forecasting Crime with Deep Learning*. Retrieved from https://arxiv.org/pdf/1806.01486.pdf

Thiele, F., Windebank, A. J., & Siddiqui, A. M. (2023). Motivation for using data-driven algorithms in research: A review of machine learning solutions for image analysis of micrographs in neuroscience. *Journal of Neuropathology and Experimental Neurology*, *82*(7), 595–610. https://doi.org/10.1093/jnen/nlad040

Travaini, G. V., Pacchioni, F., Bellumore, S., Bosia, M., & De Micco, F. (2022). Machine Learning and Criminal Justice: A Systematic Review of Advanced Methodology for Recidivism Risk Prediction. *International Journal of Environmental Research and Public Health*, *19*(17), 10594. https://doi.org/10.3390/ijerph191710594

U.S. Census Bureau QuickFacts: Los Angeles city, California. (2022). Retrieved from www.census.gov website: https://www.census.gov/quickfacts/fact/table/losangelescitycalifornia/PST045222

Vakalopoulou, M., Stergios Christodoulidis, Burgos, N., Olivier Colliot, & Lepetit, V. (2023). Deep Learning: Basics and Convolutional Neural Networks (CNNs). *Neuromethods*, *197*, 77–115. https://doi.org/10.1007/978-1-0716-3195-9_3

Varghese, R., & Prasanna, A. (2022). A Study Of Crime Analysis : A Systematic Review. *International Journal of Engineering Technology and Management Sciences*, *6*(5), 334–340. https://doi.org/10.46647/ijetms.2022.v06i05.049

Vinothkumar, K., Ranjith, K.S., Vikram, R.R., Mekala, N., Sasirekha, S.P., Reshma, R. (2023). Predicting High-Risk Areas for Crime Hotspot Using Hybrid KNN Machine Learning Framework. *IEEE Conference Publication. IEEE Xplore.* Retrieved August 28, 2023, from ieeexplore.ieee.org website: https://ieeexplore.ieee.org/document/10220738

Weather Underground. (n.d.). Los Angeles, CA Weather Conditions. Retrieved from www.wunderground.com website: https://www.wunderground.com/weather/us/ca/los-angeles

Wu, J., Abrar, S. M., Awasthi, N., Frias-Martinez, E., & Frias-Martinez, V. (2022). Enhancing short-term crime prediction with human mobility flows and deep learning architectures. *EPJ Data Science*, *11*(1). https://doi.org/10.1140/epjds/s13688-022-00366-2

Yao, S., Wei, M., Yan, L., Wang, C., Dong, X., Liu, F., & Xiong, Y. (2020). *Prediction of Crime Hotspots based on Spatial Factors of Random Forest*.

Yuki, J. Q., Sakib, Md. M. Q., Zamal, Z., Habibullah, K. M., & Das, A. K. (2019). Predicting Crime Using Time and Location Data. *Proceedings of the 2019 7th International Conference on Computer and Communications Management*. https://doi.org/10.1145/3348445.3348483

Zhang, X., Liu, L., Xiao, L., & Ji, J. (2020). Comparison of Machine Learning Algorithms for Predicting Crime Hotspots. *IEEE Access*, *8*, 181302–181310. https://doi.org/10.1109/access.2020.3028420

Ziegler, A. (2015). An Introduction to Statistical Learning with Applications. R. G. James, D. Witten, T. Hastie, and R. Tibshirani (2013). Berlin: Springer. 440 pages, ISBN: 978-1-4614-7138-7. *Biometrical Journal*, *58*(3), 715–716. https://doi.org/10.1002/bimj.201500224