Dissertation

KF7029
MSc Computer Science and Digital Technologies Project

Student Name: Arka Mandol
Student ID: w23023023

Supervisor Name: Dr. Mehmet Fatih Tuysuz
Second Supervisor Name: Dr. Biju Issac

MSc Programme: Data Science

Dissertation Title:
Time Series Crime Analysis in the UK: Integrating Explainable AI and Causal Inference

Month and Year of Submission: 09/2024

Academic Publication Venue: IEEE Transactions on Artificial Intelligence

Declaration

I declare the following:

1) That the material contained in this dissertation is the end result of my own work and that due acknowledgment has been given in the bibliography and references to ALL sources, be they printed, electronic, or personal.

2) The Word Count of this Dissertation is 7944

3) That unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on the eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.

4) I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other Department or from other institutions using the service. In the event of the service detecting a high degree of similarity between content within the service, this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

5) I have read the Northumbria University Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated, and appropriately addressed in this research.

SIGNED: Arka Mandol
DATE: 12-09-2024

Time Series Crime Analysis in the UK: Integrating Explainable AI and Causal Inference

Arka Mandol
MSc Data Science
Northumbria University
Newcastle Upon Tyne, United Kingdom
arka.mandol@northumbria.ac.uk

***Abstract*—In this paper, I propose a novel framework that combines Long Short-Term Memory (LSTM) and Traditional models with Explainable AI (XAI) techniques and causal inference to examine UK crime time series data. The author used the LSTM model to predict during periods of crime and further enhanced interpretability by applying SHAP or LIME. SHAP gave an overall perspective, and LIME was per-instance. The study found that the extent of unemployment and education are major contributors to crime. Used Bayesian Networks for the inferential task of discovering causal mechanisms—or network configurations—between these factors and crime rates. The findings demonstrate how the integration of predictive modeling and causal inference can yield interpretable results for informed crime prediction. Our future work will include streaming external causal factors and combining with advanced causal models to further enhance the prediction accuracy, making our results more useful for policy.**

***Index Terms*—crime prediction, LSTM, Explainable AI, SHAP, LIME, Bayesian Networks, causal inference, Random Forest ,Linear Regression.**

CONTENTS

I. Introduction

A. Explainable AI and Causal Inference in UK Crime Time Series Data Analysis

A gradual tendency is a more significant incorporation of Artificial Intelligence (AI) into crime prediction to help ensure public safety and law enforcement activities are done with more accuracy and speed [1]. This is a field that has recently been penetrated by the researchers. Traditional crime analytical methods such as statistical models and regression analyses to the use of Artificial Intelligence (AI) still encounter problems despite the fact that we live in complex realities behind samples, which are affected by various socio-economic, environmental, and temporal factors [2]. The emergence of machine learning, and more particularly, deep learning techniques such as Long Short-Term Memory (LSTM) networks can improve the predictive ability of crime analysis in style. LSTMs, which are able to learn long-range dependencies in sequential data, are particularly suited to issue forecasts of crime trends based on previous data [3]. But still, the models' results are very often "black boxes" even those with quite high accuracy. The practitioners and policymakers cannot easily understand the generated predictions, do not rely on them, and cannot act upon them.

Explainable AI (XAI) methods are a necessity to solve this issue and, thus, to improve the transparency and interpretability of machine learning models. For instance, Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanationS (SHAP) are such methods that give insights into the decision-making process of models when predicting the target variable [4]. Interpretability is of great importance in crime analysis. The reason is that the decisions based on AI predictions can have serious societal consequences [4].

The need for more accurate and timely insights to support law enforcement and public safety activities has led to a growing trend of integrating Artificial Intelligence (AI) into crime prediction [1]. Conventional crime analysis techniques like statistical models and regression analyses often struggle in capturing the complicated nature of crime data, which is impacted by different socio-economic, environmental, and temporal factors [2]. The emergence of machine learning, especially deep learning methods such as Long Short-Term Memory (LSTM) networks, has created new opportunities for improving the predictive power of crime analysis. LSTM models, known for their capability to identify long-term dependencies in sequential data, are a good choice for forecasting crime trends from past data [3]. However, despite their accuracy, these models frequently function as "black boxes," making their predictions challenging to understand and rely on by practitioners and policymakers.

To address this issue, explainable AI (XAI) methods have become essential for improving the interpretability and transparency of machine learning models. Methods like Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) provide insights into how models make decisions, thereby helping analysts comprehend the fundamental elements influencing predictions [4]. In crime analysis, where decisions based on AI predictions can have major societal implications, interpretability is very essential [4].

Another vital aspect of crime data analysis is comprehending the causal relationships between different factors and crime rates. Although predictive models like LSTM can predict trends, they do not inherently reveal the underlying reasons for these trends. Causal inference techniques, like Bayesian Networks and Structural Equation Modeling (SEM), provide an organized way to unravel the complex interactions between various variables that contribute to crime [5]. Combining causal analysis with predictive modeling can give a more holistic comprehension of crime dynamics, facilitating informed decision-making [6].

B. Problem Statement

Despite advancements in AI and machine learning, there are still a number of issues with the current crime prediction models [7]. The main issue is that complex models like LSTM provide predictions that are challenging to understand [8], which restricts the actual use of these models in law enforcement [9]. Furthermore, recent models fail to consider the causal relationships between different factors that impact crime, bringing about predictions that may be accurate but not insightful [10]. One way to fix this issue is to integrate XAI methodologies with that of causal inference [6], but this combination has not been thoroughly researched in the field of crime data analysis. Furthermore, despite a growing body of research on the use of AI to crime prediction, there is a noticeable lack of knowledge regarding the difficulties of crime data in the United Kingdom in the literature [11]. A customized approach to predictive modeling and causal analysis is important because of the varied and dynamic nature of crime in the UK, which is impacted by social, economic, and regional factors [10]. This paper tries to bridge this gap by focusing on the analysis of UK crime time series data employing LSTM models together with XAI and causal inference approaches.

C. Research Objectives

The main aim of this research is to create and evaluate a predictive framework that integrates LSTM models with XAI methods and causal inference approaches to

analyze UK crime data. Specifically, this paper will concentrate on the following main objectives:

- To utilize LSTM models trained on historical crime data to forecast trends in crime throughout the United Kingdom.
- To compare the effectiveness of LIME and SHAP in offering interpretable insights for the predictions that LSTM and Traditional models make.
- To find and analyze the causal relationships between various socio-economic indicators and crime rates using Bayesian Networks or SEM.

D. Significance of the Study

This study is anticipated to make several significant contributions to the field of AI and crime analysis. By combining XAI approaches with LSTM models, the research intends to improve the interpretability of crime predictions and make them more useful for law enforcement agencies. The combination of causal inference techniques will yield insights that go beyond simple prediction and a deeper comprehension of the variables influencing crime. These contributions are not only pertinent for academic study, but also hold practical significance for enhancing public safety strategies in the UK [12]. Furthermore, the emphasis on UK crime data, which has distinct regional and socioeconomic features, offers a level of specificity that is often lacking in general AI crime prediction studies. The result of this research could inform policy decisions and resource allocation in law enforcement, which might ultimately result in safer communities.

II. Literature Review

A. Historical Context and Evolution

Crime prediction has long been an area that social sciences and criminology have been interested in [13] [14]. It has traditionally depended on statistical techniques like regression analysis and spatial grouping [14]. Early models concentrated on finding patterns in historical crime data to predict future occurrences. For example, it is common practice to anticipate crime hotspots by doing spatial analysis of crime data using techniques like Kernel Density Estimation (KDE) and hot spot mapping [15].

The transition from traditional statistical methods to machine learning techniques marked a major change in the domain. Machine learning has shown promise in increasing the accuracy of crime forecasts, especially in its capacity to model intricate and non-linear relationships. Research shows that machine learning techniques like Random Forest, Support Vector Machines (SVM), and Gradient Boosting Machines (GBM) can help predict crime hotspots [16].

B. Deep Learning in Time Series Forecasting

Deep Learning models, especially Long Short-Term Memory, have become increasingly popular these days in time series forecasting than other machine learning techniques because they can capture temporal connections in sequential data [17] [18]. [19] put forward LSTM models, a type of Recurrent Neural Networks (RNNs), to address the reducing gradient problem in standard RNNs. These models are particularly suited for tasks that require the use of long-term memory, making them perfect for predicting crime trends that rely on historical patterns [20].

Recent literature has shown promising results in the application of LSTM models in crime prediction. Research by [21] utilized LSTM networks to predict crime rates in smart cities, demonstrating that LSTM models could outperform traditional ARIMA models in terms of accuracy by achieving a validation predictive accuracy of 87% on a large dataset of historical crime data. Similarly, [22] demonstrates that an LSTM encoder-decoder architecture produced greater performance in forecasting crime numbers throughout 83 regions of Costa Rica, with the model performing especially well in areas with higher crime frequency. These results reveal promising outcomes when employing LSTM models to crime prediction.

Though LSTM models are very predictive, they have been criticized for their ”black box” nature that makes their internal workings and decision-making processes opaque. This lack of interpretability is a huge obstacle to their widespread use in critical domains like crime prediction, where grasping the factors driving predictions is important for actionable insights [23].

C. Explainable AI (XAI) Techniques

1) The Need for Interpretability in AI: The growing complexity of AI models, particularly deep learning networks, has resulted in escalated concerns about their interpretability. In the realm of crime prediction, where AI-driven judgments can have huge societal impacts, the capability to clarify and justify predictions is essential. This demand for interpretability has led to the development of Explainable AI (XAI), which aims to make AI models more transparent and easier for people to understand [12].

Although the terms interpretability and explainability are frequently used interchangeably in the field of AI, they have different meanings: interpretability relates to comprehending the internal logic and mechanics of the model, whereas explainability is concerned with understanding the intuition behind a model's outputs and cause-and-effect relationships. As highlighted by [12], interpretability alone is insufficient without explainabil-

ity, both are necessary in high-stakes applications like crime prediction to ensure openness and accountability.

There are two primary approaches to Explainable AI (XAI): ante-hoc and post-hoc methods. After a model is developed, post-hoc XAI uses methods to explain its predictions. For complicated models such deep learning networks, it uses techniques like GradCAM, LIME, and SHAP [24]. These methods might not, however, always faithfully capture the model's decision-making process. Ante-hoc XAI, on the other hand, concentrates on creating models that are naturally interpretable from the start, including decision trees and linear models, which offer transparent insights into their predictions. Ante-hoc models allow users to understand the logic behind the model's decisions, even though they may trade some predictive accuracy in the name of openness [25]. Depending on the required balance between accuracy and interpretability, one can choose amongst these strategies.

2) Local Interpretable Model-agnostic Explanations (LIME): LIME, developed by Ribeiro, Singh, and Guestrin in 2016 [26], is a widely used technique that approximates the decision boundary of a complicated model locally around a given prediction. LIME creates a simple, interpretable model (often a linear model) that replicates the behavior of the complicated model around the instance by perturbing the input data and tracking the changes in the model's predictions. With this method, users can comprehend which features had the most influence and why the model predicted a certain outcome [27].

3) SHapley Additive exPlanations (SHAP): SHAP, developed by Lundberg and Lee in 2017 [27], is another prominent technique for explaining how AI models make predictions. Based on principles of cooperative game theory, SHAP calculates the contribution of each feature to a prediction by considering all possible combinations of features. This method gives a consistent and theoretically based way to rank the value, making it particularly suitable for understanding complex models.

SHAP has been utilized in different domains, including crime prediction, to provide insight on how models reach their decisions. In a recent study, researchers employed SHAP, to analyze and evaluate the importance of various features within clustered groups of data. By combining k-means clustering, PCA, and a LightGBM classifier during the preparation of data, the study has grouped features into multiple clusters before using SHAP. The simulation results show that SHAP values play an important role in developing accurate and meaningful representations of each cluster, showing the most significant features influencing the model's predictions [28]. However, despite how good they are, both have limitations. While SHAP's processing complexity may be unaffordable for very big datasets or extremely complicated models like LSTMs, LIME's local explanations may occasionally be deceptive if the model acts nonlinearly in other regions of the input space [28].

D. Causal Inference in Crime Data Analysis

1) Importance of Causal Understanding in Crime Analysis: Predictive models are useful for predicting crime trends based on historical data, but they frequently don't provide details about the underlying causes of these trends. Understanding the causal relationships between various factors and crime rates is pivotal for effective crime prevention measures. For instance, knowing that greater unemployment rates lead to higher crime rates can guide policymakers in designing targeted interventions. Causal inference techniques offer a body for studying these relationships, moving beyond simple correlation to comprehend causality. This distinction is crucial in crime analysis, where interpositions that rely only on correlations may result in ineffective or counterproductive policies.

2) Bayesian Networks in Crime Analysis: Bayesian Networks are robust graphical models that encode probabilistic relationships between variables, making them better for analyzing complex interactions in crime data. These networks are quite beneficial for data analysis, especially when it comes to modeling and understanding the causal processes that can result in illegal activities. By encoding variables and their dependencies using a directed acyclic graph (DAG), Bayesian Networks allow researchers to investigate and predict the effects of several factors impacting crime.

Bayesian networks have been used in several studies to analyze crime data. For instance, [29] showed how these networks can capture small correlations between offender characteristics and crime scene evidence by modeling criminal profiling from limited data using Bayesian networks. The study demonstrated how well Bayesian Networks could manage vagueness and missing data in criminal investigations. Similarly, [30] used a spatiotemporal Bayesian model to show trends in burglaries, investigate correlations between crimes and socioeconomic indicators, and examine distribution patterns, demonstrating how useful the model is for figuring out causal relationships in safety-related data.

E. Identified Gaps and Research Contributions

Even with the advances in causal inference, XAI, and AI, there are still several gaps in the literature. Firstly, although LSTM models have showed potential in the field of crime prediction, their interpretability issues contain their real world use. Secondly, there is a shortage of research on the application of XAI approaches to causal inference methods in the study of crime data,

especially when it comes to UK crime data. Finally, more targeted research is required to address the difficulties in analysing crime statistics in the UK while considering particular socioeconomic and regional aspects.

By creating a comprehensive framework that combines LSTM models, XAI approaches, and causal inference methods to analyse UK crime time series data, this research aims to close these gaps. A more understandable and practical crime prediction model, insights into the causal connections underlying crime patterns, and a tailored method for handling the inherent challenges of UK crime data analysis. This framework will be judged based on its capacity to support both academic research and real-world applications in public safety by offering precise forecasts, comprehensible insights, and a thorough grasp of the causative variables impacting crime.

F. Dataset Overview

1) Data on Migrants: The ONS's migration dataset, which records UK and international migration, is essential for studying demographic changes and local crime rates.

2) Data on Education Level: This ONS dataset categorises people by their highest education level, helping analyse educational disparities' impact on crime and apply explainable AI to find patterns.

3) Data on Employment History: ONS data classifies people into employed, unemployed, or outside the labour force, allowing causal inference to examine crime rates and employment movements.

4) Data on Ethnic Groups: This ONS dataset divides the UK population by ethnicity, allowing crime statistics to be analysed across ethnic groups and causal connections to be drawn.

5) Data on Economic Status: The ONS's economic status dataset looks at people's economic activity, which is important for using causal analysis to investigate the connection between economic disparity and crime.

6) Crime Information: The main dataset for crime prediction and causal inference in this project is the "UK Police Street Crime" dataset from Kaggle, which covers street-level offences from 2018 to 2021.

G. Integration and Analytical Approach

All datasets are combined in this project to provide a comprehensive examination of the factors that contribute to crime in the UK. The objective is to identify and quantify how demographic, educational, economic, and ethnic factors impact crime using causal inference methodologies. Applying explainable AI (XAI) techniques will ensure that the models developed are understandable and accurate, enabling stakeholders to comprehend the fundamental causes of crime and make informed decisions.

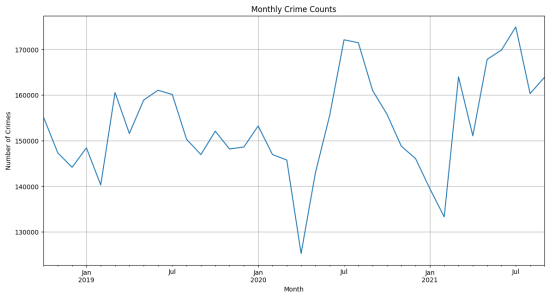


Fig. 1. TimeSeriesVisualizationofCrimeData

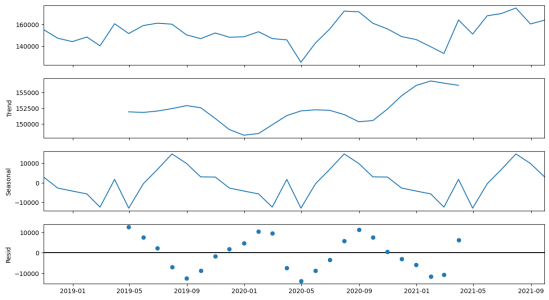


Fig. 2. Time Series Decomposition

III. Methodology

A. Crime Data Analysis and Forecasting using LSTM and Explainability Techniques

This section presents the approach utilized to forecast crime data employing interpretable machine learning algorithms and a Long Short-Term Memory (LSTM) model.

1) Data loading and Preprocessing: UK crime dataset was loaded and then filtered by "Violence and sexual offenses" and checked for missing values.

Time Series Transformation: To aggregate crime counts, the data was resampled to a monthly frequency and the 'Month' column was transformed into a datetime format to capture broader trends and seasonal variations, simplifying complexity and aligning with typical crime reporting frequencies.

Resampling: To capture broader trends and seasonal fluctuations, monthly aggregate was selected.

2) Time Series Visualization and Decomposition: Line plots were used to analyze the cleaned time series data to understand general crime patterns.

The `seasonal_decompose` method from statsmodels split the time series into trend (long-term crime changes), seasonal (recurring yearly patterns), and residual (unexplained noise) to uncover deeper data insights [31].

$$X_t = T_t + S_t + R_t \tag{1}$$

Where: - X_t is the observed time series. - T_t is the trend component. - S_t is the seasonal component. - R_t is the residual component.

Decomposition helps in the identification of long-term trends and seasonal patterns that can inform proactive crime prevention intervention tactics [32].

3) LSTM Model Development and Training: The LSTM model was used because of its ability to capture long-term dependencies in time series data [33]. The following steps were used:

Data Preparation: To forecast the number of crimes for the upcoming month, the data was transformed into a supervised learning format using 12-month sequences. To ensure robust training, features were normalized to the range [0, 1] using MinMaxScaler.

Model Architecture: To avoid overfitting, dropout layers (rate 0.3) and two LSTM layers, each with 50 units, were employed. The estimated crime count was obtained by adding a fully connected dense layer.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2}$$

Where: - C_t is the cell state. - f_t is the forget gate. - i_t is the input gate. - $\tilde{C}_t$ is the candidate cell state.

Model Compilation and Training: The adaptive learning rate and Mean Squared Error (MSE) loss function of the model were compiled using the Adam optimizer. To improve learning and provide frequent weight updates, it was trained for 500 epochs with a batch size of 1.

Incorporating lag features, seasonal features, and temporal covariates can improve the LSTM model's ability to capture long-term dependencies and seasonal crime trends [34].

4) Model Predictions and Evaluation: Predictions were evaluated using MSE, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE):

- Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values [35].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3}$$

  Where: - n is the number of observations. - y_i is the actual value. - $\hat{y}_i$ is the predicted value.
- Root Mean Squared Error (RMSE): Provides an interpretable metric in the same units as the data.

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{4}$$

  Where: - MSE is the Mean Squared Error.
- Mean Absolute Error (MAE): Measures the average absolute difference between predicted and actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{5}$$

  Where: - n is the number of observations. - Y_i is the actual value. - $\hat{Y}_i$ is the predicted value.

To improve model generalization and mitigate overfitting, L1/L2 regularization, dropout, and data augmentation methods like synthetic data generation or bootstrapping can be implemented to enhance the model's ability to learn diverse patterns [36], [37].

5) Incorporating External Factors for Robust Predictions: To improve robustness, external factors such as economic shifts, public events, and policy changes can be integrated as additional features in the LSTM model to capture their temporal effects on crime rates [16].

6) Model Interpretability with LIME and SHAP: Two interpretability techniques were used to improve the predictability of LSTMs:

LIME: LIME highlighted which months (time steps) had the greatest influence on a particular forecast by approximating the LSTM model with a simpler one for individual predictions. For expeditious criminal assessments, this method provides rapid, actionable information.

SHAP: This method provided both local and global interpretability by quantifying the contribution of each time step to the forecasts. To ensure consistency and fairness in feature attribution, SHAP summary plots were used to identify the most influential months for each forecast.

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|N|!}{|S|!(|N| - |S| - 1)!}[f(S \cup \{i\}) - f(S)] \tag{6}$$

Where: - ϕ_i is the SHAP value for feature i. - S is a subset of features excluding i. - N is the total set of features. - $f(S)$ is the model output for subset S.

Both methods offered vital information on how historical crime data affected model projections, facilitating quick decision-making and guaranteeing impartiality in research.

B. Random Forest and Linear Regression, along with model interpretability techniques SHAP and LIME

In addition to SHAP and LIME for interpretability, this section describes how to create and evaluate crime prediction models using Random Forest and Linear Regression.

1) Data loading and Preprocessing: UK Police crime data for 2021, filtered for "Violence and sexual offenses." The 'Month' column was converted to datetime, and the 'LSOA name' was renamed. The data was combined by month and renamed column.

Socio-economic indicators were standardized by geographical identifiers and merged with the crime data using an outer join. SHAP and LIME were applied to the

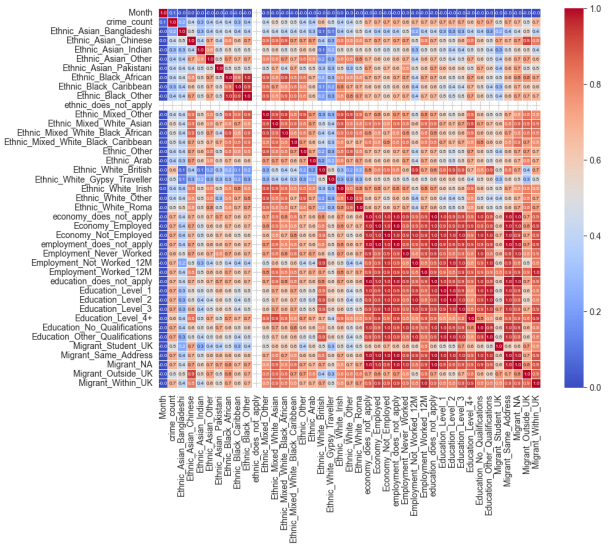


Fig. 3. Correlation Matrix of All Data Variables

models for interpretability to identify key socioeconomic factors influencing crime predictions.

2) Merging Datasets and Feature Engineering: The socioeconomic and criminal databases were combined, with relevant columns being renamed and redundant ones being eliminated. Among the features engineered were:

Feature Selection: Choosing the socioeconomic features that have the strongest correlation with crime count.

Feature Scaling: Apply StandardScaler to standardize a subset of features.

Time Features: To capture seasonality, convert the 'Month' column into a category variable.

3) Visualization of Correlation Matrix: The most important variables were highlighted in the correlation matrix and heatmap that were made to illustrate the connections between socioeconomic characteristics and crime rates.

4) Model Training and Train-Test Split: Models were trained on the training data and tested on the reserved data to assess performance. The data was divided into 70% training and 30% testing subsets.

a) Model Selection: Random Forest Regressor: The Random Forest Regressor is a popular choice for complicated datasets due to its exceptional capacity to capture non-linear relationships and feature interactions.

Linear Regression: Linear Regression was selected as a benchmark model because it is straightforward and can produce comprehensible linear connections between characteristics and crime counts.

5) Model Evaluation: To evaluate the models, two key metrics were utilized:

Mean Squared Error (MSE) was used, It facilitates the quantification of prediction error.

R-squared (R^2): Represents the proportion of the variance in the dependent variable that is predictable from the independent variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{7}$$

Where: - n is the number of observations. - y_i is the actual value. - $\hat{y}_i$ is the predicted value. - $\bar{y}$ is the mean of actual values.

The test data was used to evaluate both models, and the models' MSE and R^2 scores were compared to ascertain which one performed better.

6) Feature Importance Visualization: The linear link between characteristics and crime counts was demonstrated by plotting absolute values of the coefficients that were computed and visualized for Random Forest and Linear Regression Feature Importance.

7) SHAP Values for Global and Local Interpretation: SHAP values were used for interpretability in both models. TreeExplainer revealed global feature importance in Random Forest and shap.LinearExplainer gave local and global insights for Linear Regression.

8) LIME for Local Interpretation: LIME provided local insights for both models, simplifying Random Forest predictions and clarifying individual feature contributions in Linear Regression, making each prediction easier to understand.

C. Bayesian Network for Causal Inference on Crime Data

The process for building and evaluating a Bayesian network to determine the causal links between socioeconomic characteristics and crime rates is described in this section.

1) Imports and Setup: Main Python libraries utilized for data manipulation are pandas and numpy, for visualization are matplotlib and seaborn, and for Bayesian Network development pgmpy & networkx was used.

2) Crime Data Preparation: Converted 'Month' to datetime format and filtered for relevant 2021 crime data, focusing on "Violence and sexual offenses."

3) Data Cleaning and Integration: Aggregated crime data by location, integrated socioeconomic data (migration, employment, etc.), and cleaned for modeling.

4) Correlation and Distribution Analysis: Visualized correlation relationships with a heatmap and analyzed key variable distributions using histograms to identify trends for Bayesian modeling.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}} \tag{8}$$

Where: - x_i and y_i are individual data points. - $\bar{x}$ and $\bar{y}$ are the means of the x and y variables.

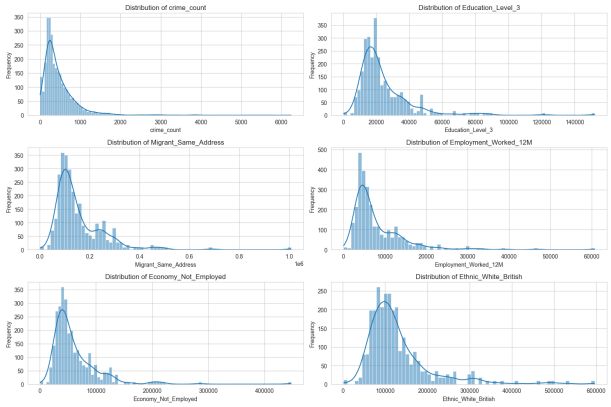


Fig. 4. Distribution of key variables related to crime.

5) Data Transformation for Bayesian Modeling: Logarithmic transformation was applied to normalize skewed variables, reducing the influence of outliers for more robust Bayesian Network modeling. Quantile binning then simplified the log-transformed data, categorizing variables like crime counts into "Low," "Medium," and "High," aiding discrete relationship modeling. Additionally, the 'Month' column was encoded categorically to capture seasonal crime trends.

$$y' = \log(y + 1) \tag{9}$$

Where: - y is the original variable. - y' is the log-transformed variable.

6) Bayesian Network Structure Learning: Using the K2 score method to guide the search, the Hill-Climb Search algorithm was used to discover the structure of the Bayesian Network. 'Month_Encoded' was one of the variables that the BIC score rejected owing to its complexity penalty, hence K2 was selected because it permitted the inclusion of all pertinent variables.

7) Fitting and Evaluating Bayesian Network: The Bayesian Estimator was used to fit the learnt structure to the data. BIC and K2 scores were used to assess the model:

- BIC: model fit is measured, and complexity is penalized.
- K2: maximizes data fit while lessening the penalty for complexity.

The dual evaluation aids in achieving a trade-off between complexity and model accuracy.

8) Inference and Analysis: Key factors contributing to high crime rates were determined by inference utilizing the Variable Elimination approach, including socioeconomic status and employment.

$$P(C \mid E) = \frac{P(E \mid C) \cdot P(C)}{P(E)} \tag{10}$$

Where:

- $P(C \mid E)$ is the probability of the crime count being in a specific bin given evidence E.
- $P(E \mid C)$ is the probability of observing the evidence given the crime count.
- $P(C)$ is the prior probability of the crime count bin.
- $P(E)$ is the probability of observing the evidence.

9) Visualization of Bayesian Network: The networkx package was used to visualize the topology of the Bayesian network in a circular form, with larger nodes and thicker edges signifying important interactions.

D. LSTM Hotspot Mapping

1) Data Collection and Preprocessing: The UK's crime dataset was utilized, which includes details of crime occurrences that contain geo-coordinates latitude and longitude as well as timestamps in monthly intervals. The data was imported from a CSV file and processed for time-series modeling.

- **Date Formatting and Sorting**: Converting the 'Month' column to datetime format, and sorting the dataset chronologically was critical so that the temporal order was maintained. This step guarantees proper time-series forecasting by the LSTM model.
- **Aggregating Crime Data**: By grouping the dataset by 'Month', 'Latitude', and 'Longitude', a spatial-temporal dataset was created, with each row showing the crime count for a specific location and time. This aggregation step is important for preparing the data, which helps the LSTM model to understand spatial and temporal crime patterns [38].

2) Data Normalization: MinMax() scaling normalized crime counts between 0 & 1, ensuring consistent feature scaling for stable & efficient LSTM model training.

$$x_i' = \frac{x_i - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \tag{11}$$

Where:

- x_i is the crime count for location i.
- x_{min} and x_{max} are the minimum and maximum crime counts in the dataset.

3) Sequence Creation for LSTM: The dataset underwent the process of transforming it to a series of input-output pairs, using a sliding window approach. The model used a sliding window of the past three months' crime counts to predict the next month, with input for each time step derived.

$$X_t = [C_{i,t-2}, C_{i,t-1}, C_{i,t}] \quad \text{to predict} \quad y_t = C_{i,t+1} \tag{12}$$

Where: - X_t is the feature vector containing past crime counts. - $C_{i,t-2}, C_{i,t-1}, C_{i,t}$ are the crime counts for location i in previous months. - $y_t = C_{i,t+1}$ is the crime count for location i at time $t+1$.

4) Building and Compiling the LSTM Model: Using Keras, the LSTM model was built with 50 units for capturing temporal crime patterns and a single-neuron dense layer was used to predict the next month's crime count.

5) Model Training: A balanced approach between computational efficiency and model efficiency was adopted for the model training utilizing 10 epochs and batch size of 512. The decision of a smaller number of epochs was adopted so as to prevent overfitting while ensuring that the model learns from the data well.

6) Predictions and Inverse Transformation: Subsequently, the model predicted the crime counts for the next year's of the training period. In the case where the data were normalized before training, thus the predicted values were converted back to the original scale through inverse MinMax scaling.

$$p_i = p_i' \cdot (x_{\max} - x_{\min}) + x_{\min} \tag{13}$$

Where: - p_i' is the normalized prediction. - $x_{\min}$ and $x_{\max}$ are the minimum and maximum crime counts in the dataset.

7) Visualization with Heatmaps: To visualize the predicted crime hotspots, a heatmap was created using the Folium library. A crime count was plotted on the map for each longitude and latitude. Each location's predicted crime count is shown by the intensity of the heatmap.

8) Model Evaluation using RMSE: The model's effectiveness was measured by Root Mean Squared Error (RMSE), which reflects the average prediction error. This metric is crucial for understanding how well the model predicts actual crime occurrences.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{14}$$

The closer the model's predictions are to the actual values, the lower the RMSE score and the better the model performance [39]. This metric is particularly valuable in evaluating the accuracy of forecasted crime counts, providing a clear measure of the model's predictive power and guiding further refinement.

IV. Results and Discussion

A. LSTM Crime Trend Prediction and Model Performance

The LSTM model, trained on past monthly crime data, leverages historical trends to forecast future crime patterns.

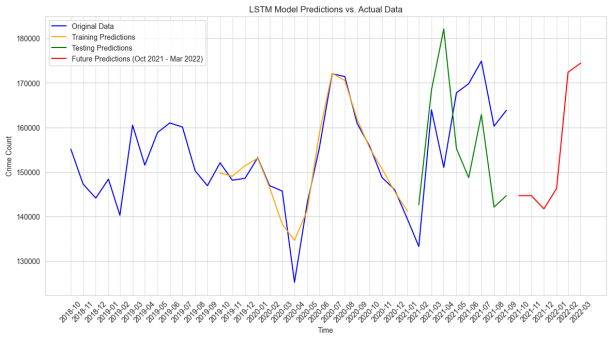


Fig. 5. LSTM Model Predictions vs. Actual Crime Data (Train, Test, and Future Predictions)

1) LSTM Model Performance: Training and Testing: During the training phase, the LSTM model demonstrated good performance, accurately capturing temporal changes in crime. The model was able to fit the training data accurately, as evidenced by its Train Mean Squared Error (MSE) of 11,264,257.07 and Train Root Mean Squared Error (RMSE) of 3,356.23. On the test set, however, the model's performance declined, with Test MSE of 314,326,459.66 and Test RMSE of 17,729.25, indicating a notable rise in error when exposed to unseen data.

The significant rise in errors during the testing period can probably be attributed to external interruptions like the COVID-19 pandemic, which brought unusual patterns of criminality [40]. The pandemic caused significant changes in the UK's crime rates, including sudden declines and increases that the model found difficult to account for.

2) Discussion: Impact of External Disruptions on LSTM Models: The pandemic drastically affected crime rates introducing noise & unpredictability, the LSTM model struggled to handle effectively. This begs the questions of how resilient classic time series models like LSTM are to such disturbances and if model resilience may be enhanced by including external influences (such economic shifts or public events).

The model's predictions and actual crime counts during the test phase differed, as shown by the visualization in Figure 6. This was especially true during the pandemic months when lockdowns caused a dramatic decline in crime rates.

3) Model Interpretability: SHAP vs. LIME: Enhancing the interpretability of the LSTM crime trend predictions using Explainable AI (XAI) techniques—specifically, SHAP and LIME—was one of the main goals of this work. These techniques were employed to clarify the ways in which various time lags (prior crime data points) influenced the predictions made by the LSTM model.

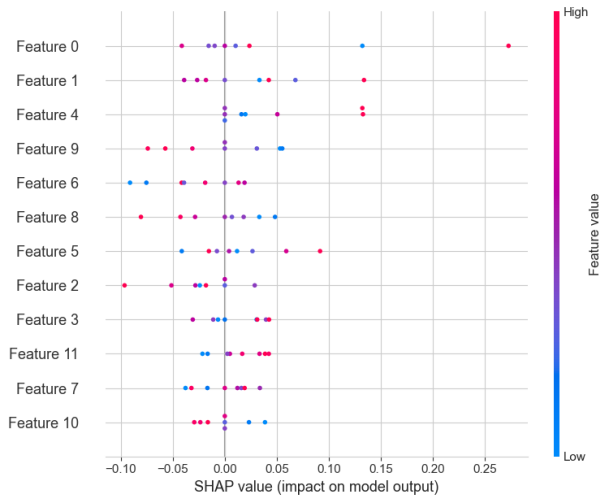


Fig. 6. SHAP Summary Plot Showing Global Feature Importance in LSTM Predictions

a) SHAP: Global and Local Explanations: SHAP was used to give both global and local interpretability by calculating the contribution of all features (time lag) to the model's predictions. Figure **??** SHAP summary plot emphasizes how crucial recent crime data points—like t-0, t-1, and t-4—are to the model's ability to make predictions. These characteristics consistently had the highest SHAP values, suggesting that the most important factors for predicting future crime trends are current criminal incidents. The impact of earlier time lags (t-7 to t-9) on the predictions was decreasing.

Understanding the global trends in how the LSTM model uses historical crime data is made possible by SHAP's capacity to consistently and reliably deliver feature importance throughout the whole dataset. For example, SHAP disclosed that the most recent time point, t-0, positively impacted crime predictions. This implies that a high crime rate in the current month would probably lead to a higher prediction for the following month.

b) LIME: Instance-Specific Explanations: For each prediction, instance-specific explanations were produced using LIME. Here, LIME modifies the input data and tracks how slight modifications impact the model's output, yielding a breakdown of feature contributions for each prediction.

For instance, LIME breaks clearly how recent crime counts (t-0, t-1) contribute to an LSTM prediction for a particular instance in Figure **??**. LIME demonstrated how recent crime spikes affected the model's output by capturing the local significance of characteristics like t-0 and t-1. To make short-term crime estimates, the LSTM model relies heavily on current data points; older time points (such as t-8 and t-9) contributed less to this.

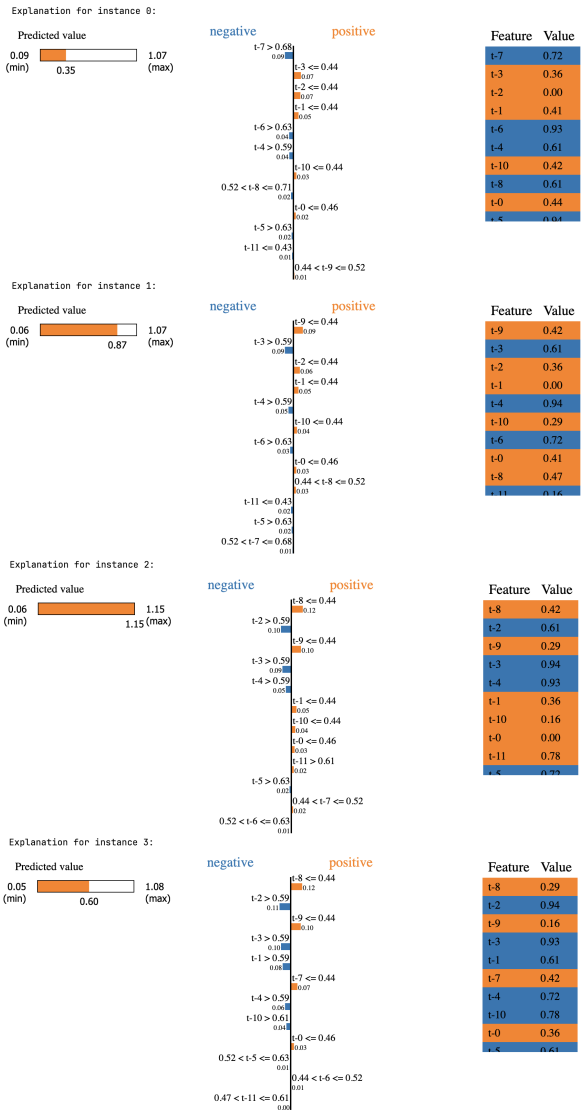


Fig. 7. LIME Explanation for a Specific LSTM Prediction (First 4 Instance)

Although LIME's instance-specific insights are helpful in comprehending specific predictions, small variations between runs may result from the randomness induced by disturbance. Because of this, LIME is better suited for local interpretability than for identifying broad patterns in the dataset.

c) SHAP vs LIME, which is Better?: When the two methods were compared, SHAP proved to be the most effective way to understand the crime predictions made by the LSTM model. SHAP was better suited for comprehending the behavior of the model over the entire dataset because of its capacity to offer consistent, theoretically supported global and local insights. In order to explain time-series models like LSTM, where the link between time lags and future predictions might be

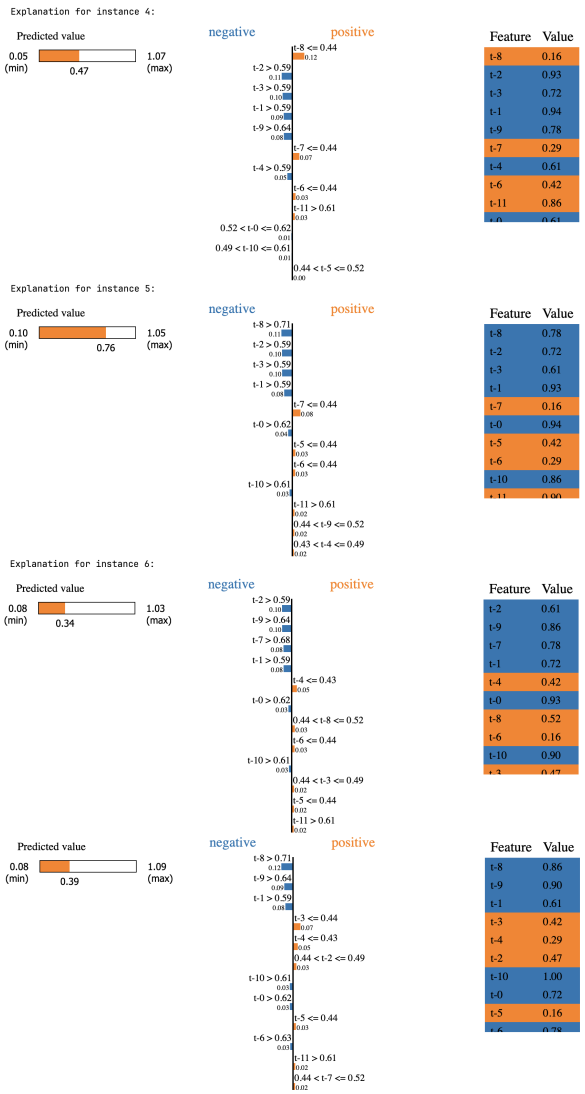


Fig. 8. LIME Explanation for a Specific LSTM Prediction (Last 4 Instance)

complex and non-linear, SHAP also makes sure that the significance values are constant across instances.

Here, SHAP should be preferred for interpretability as it provided clearer, more consistent insights into how time lags influenced the LSTM predictions.

B. LSTM Crime Hotspot Prediction

The findings of predicting crime hotspots in different UK regions using the LSTM model are shown in this section. Finding locations with a high likelihood of increased criminal activity was the main objective in order to help law enforcement organizations allocate resources more wisely.

1) Model Performance: Root Mean Squared Error (RMSE): The Root Mean Squared Error (RMSE) metric

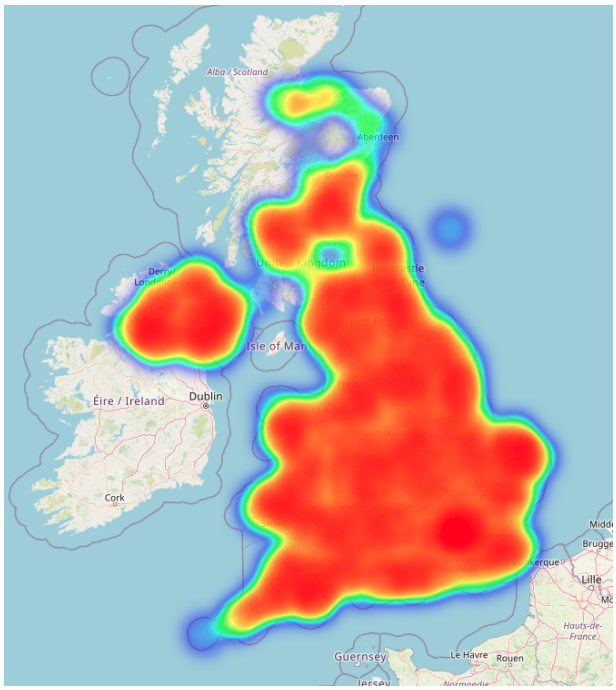


Fig. 9. Crime Hotspot Prediction Heatmap for the First Month

was used to assess the LSTM model's hotspot prediction performance. With an RMSE of 1.572, the model demonstrated a respectable degree of accuracy in estimating the number of crimes per site for the first month of the study. According to this RMSE value, there was an average deviation of 1.57 crimes per region between the model's predictions and the actual crime counts.

Although this performance is encouraging, there is still space for improvement, particularly in areas where crime rates are lower and the model shows somewhat higher prediction errors. This is in line with the difficulties time series models frequently encounter when forecasting results in low-variance areas.

2) Crime Hotspot Predictions Visualization: Figure 9 shows the expected crime hotspots for the first month. Areas with higher predicted crime activity are shown by red shading. The algorithm identified major hotspots as urban centers like London, Birmingham, and Manchester, as anticipated, which is in line with past trends in crime (Hart, 2020).

3) Practical Implications for Law Enforcement: Law enforcement organizations can take action based on the predictions made by the LSTM model, especially when it comes to allocating resources. Law enforcement can more efficiently allocate resources, speeding up reaction times and possibly averting crime in high-crime regions discovered by the model.

4) Limitations and Potential Improvements: Although the low RMSE suggests that the LSTM model for hotspot

prediction did quite well, there are a few issues that could be fixed in next iterations of the model:

- **Prediction Granularity:** At the moment, the model offers predictions at the regional level. More accurate insights might be obtained by predicting at the neighborhood level, for example, to increase the granularity of the projections.
- **External Factors and Long-Term Validation:** The model, relying solely on past crime data, overlooks external factors like economic shifts or public events, which could enhance accuracy. Longer validation periods may offer deeper insight into its predictive abilities.

C. Crime Counts and Socio-Economic Factors Prediction

This section compares crime forecasts from Linear Regression, a simple baseline, and Random Forest, which captures non-linear patterns, using SHAP and LIME to reveal the impact of socioeconomic and demographic factors.

1) Model Performance: Linear Regression vs. Random Forest: The Random Forest model, with an MSE of 7421.12 and R^2 of 0.9603, explained 96% of the crime count variance, handling complex non-linear relationships effectively. In contrast, the Linear Regression model, with a much higher MSE of 85258.72 and R^2 of 0.5433, struggled to capture these interactions, performing significantly worse.

TABLE I
PERFORMANCE METRICS FOR RANDOM FOREST AND LINEAR REGRESSION MODELS

Model	MSE	R^2
Random Forest	7421.12	0.9603
Linear Regression	85258.72	0.5433

2) Feature Importance Analysis: SHAP and LIME: The Random Forest and Linear Regression models' predictions were interpreted using both SHAP and LIME, revealing which socioeconomic and demographic variables had the biggest impact on crime rates.

a) SHAP Analysis: International and Regional Perspectives: Figure 10 shows SHAP values indicated that Employment_Worked_12M, which positively impacted crime rates, was the most significant feature for the Random Forest model. This indicates that areas with more employment during the previous 12 months may have had higher crime rates as a result of sporadic or insecure work circumstances.

Other notable characteristics, such Economy_Not_Employed and Education_Level_3, also made a substantial contribution to the forecasts.

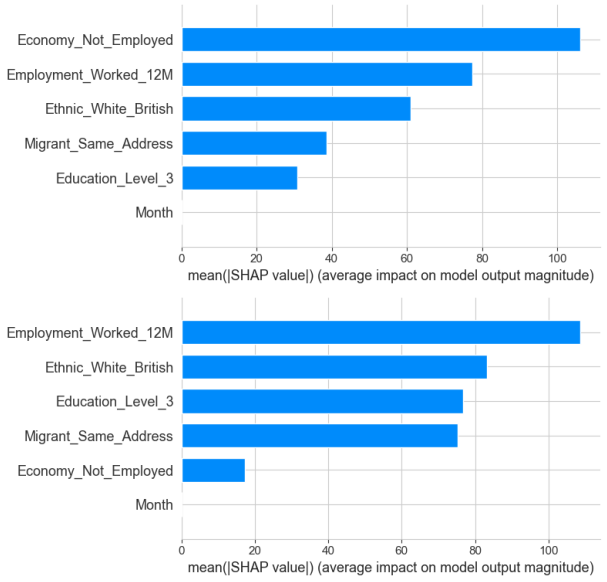


Fig. 10. SHAP Summary Plot for Feature Importance in Random Forest(top) and Linear Regression(bottom) Models

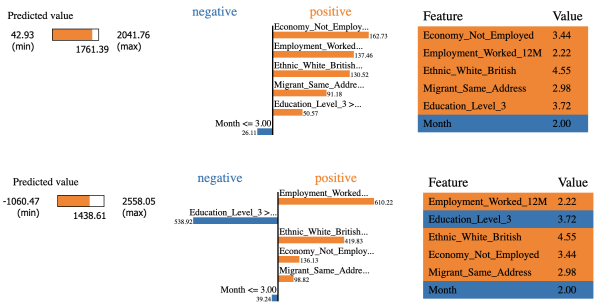


Fig. 11. LIME Explanation for Random Forest Prediction and Linear Regression

b) LIME Analysis: Instance-Level Explanations: For cases in both models, local explanations were produced using LIME. The contribution of features like Employment_Worked_12M and Ethnic_White_British to a specific Random Forest model prediction is displayed in Figure 11 by LIME. For instance, Education_Level_3 contributed negatively by lowering the projected crime count, whereas Employment_Worked_12M contributed favourably to a crime prediction of 1438.61.

3) SHAP vs LIME, which is Better?: SHAP and LIME both provided insightful analyses of the models' interpretability. SHAP, by offering both global and local feature importance, helps explain how socioeconomic factors shape crime rates, providing the clarity and consistency needed for stakeholders to trust model insights. LIME was very helpful in providing instance-specific explanations, including brief descriptions of the meth-

ods used to make individual predictions. Nevertheless, because feature importance might fluctuate somewhat between runs, its dependence on perturbation rendered it less trustworthy for identifying global patterns.

4) Discussion: Socio-Economic Factors and Crime Prediction: The Random Forest model's findings point to a number of important socioeconomic variables that affect crime rates. The robust significance of Employment_Worked_12M implies a close relationship between crime rates and employment patterns, especially in areas where there have been recent swings in employment. This result can be a reflection of the erratic nature of contract or temporary work, which can put financial strain on people and raise crime rates.

Higher education levels were found to be linked to decreased crime rates, whereas Education_Level_3 was found to have a protective impact against crime. This is consistent with other research that indicates increased education increases opportunities and lowers the risk of criminal activity.

The more conventional criminological theory that points to unemployment as a major contributing factor to crime is reflected in the Linear Regression model's emphasis on Economy_Not_Employed.

Policymakers can better target crime prevention initiatives by concentrating on communities with lower levels of education and insecure employment, according to these insights.

D. Discussion: Socio-Economic Factors and Crime Prediction

The Random Forest model's findings point to a number of important socioeconomic variables that affect crime rates. The robust significance of Employment_Worked_12M implies a close relationship between crime rates and employment patterns, especially in areas where there have been recent swings in employment. This result can be a reflection of the erratic nature of contract or temporary work, which can put financial strain on people and raise crime rates.

Higher education levels were found to be linked to decreased crime rates, whereas Education_Level_3 was found to have a protective impact against crime. This is consistent with other research that indicates increased education increases opportunities and lowers the risk of criminal activity.

The more conventional criminological theory that points to unemployment as a major contributing factor to crime is reflected in the Linear Regression model's emphasis on Economy_Not_Employed.

Policymakers can better target crime prevention initiatives by concentrating on communities with lower levels of education and insecure employment, according to these insights.

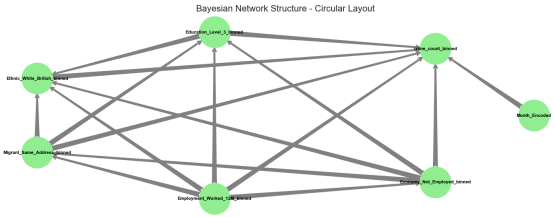


Fig. 12. Advance Bayesian Network Structure

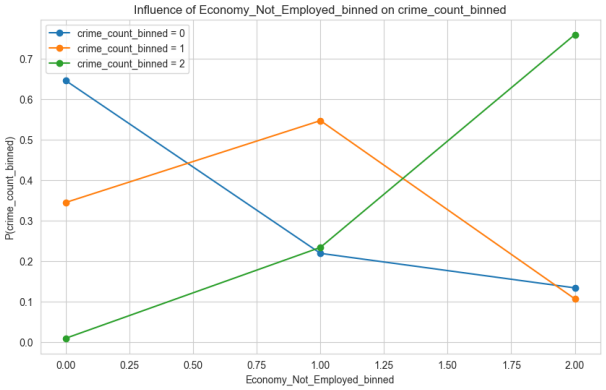


Fig. 13. Bayesian Network - Influence of "Economy_Not_Employed_binned" on Crime Counts

E. Bayesian Network Analysis on Crime and Socio-Economic Factors

The Bayesian Network analysis, which was done to determine the probability correlations between crime rates and several socioeconomic indicators such as unemployment, education, migratory stability, and ethnic composition, is shown in this section. These components and their conditional dependencies are represented graphically by the Bayesian technique, which aids in the discovery of causal correlations that can guide crime prevention strategies.

1) Influence of "Economy_Not_Employed_binned" on Crime: Figure 13 shows low crime rates dropped steadily as unemployment increased, but the likelihood of high crime rates increased in regions with high unemployment.

2) Influence of "Education_Level_3_binned" on Crime: Figure 14 illustrates how, according to conventional wisdom, higher education levels were associated with a higher likelihood of rising crime rates. This unexpected finding raises the possibility that additional variables, such as urbanization, are present in these areas.

3) Influence of "Migrant_Same_Address_binned" on Crime: Figure 15 shows that areas with greater migrant stability—that is, people who lived at one residence for a prolonged amount of time—were also more likely to have higher rates of crime.

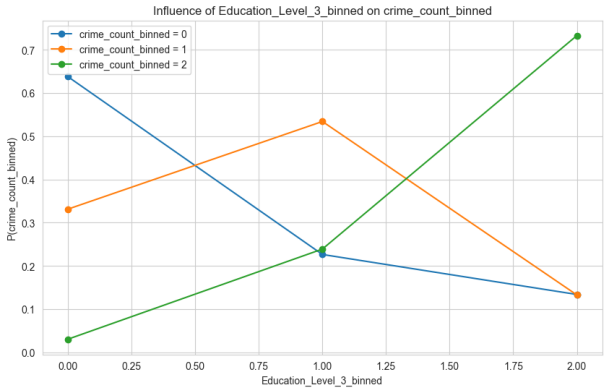


Fig. 14. Bayesian Network - Influence of "Education_Level_3_binned" on Crime Counts

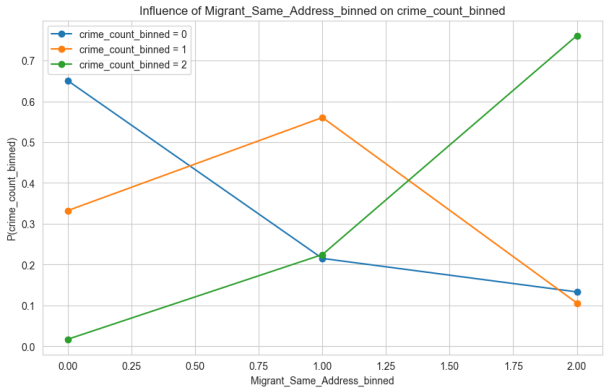


Fig. 15. Bayesian Network - Influence of "Migrant_Same_Address_binned" on Crime Counts

4) Influence of "Ethnic_White_British_binned" on Crime: It was discovered that there was a positive correlation between the percentage of Ethnic White British people living in a given area and the crime rate (Figure 16).

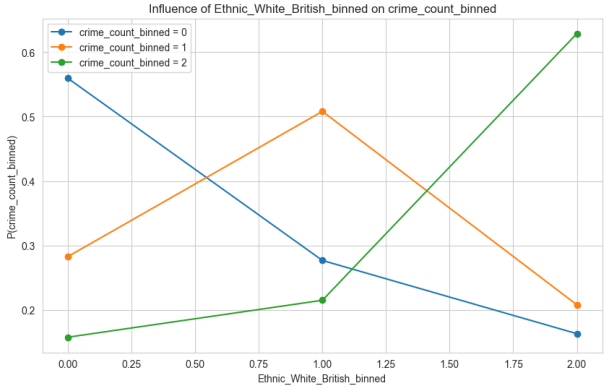


Fig. 16. Bayesian Network - Influence of "Ethnic_White_British_binned" on Crime Counts

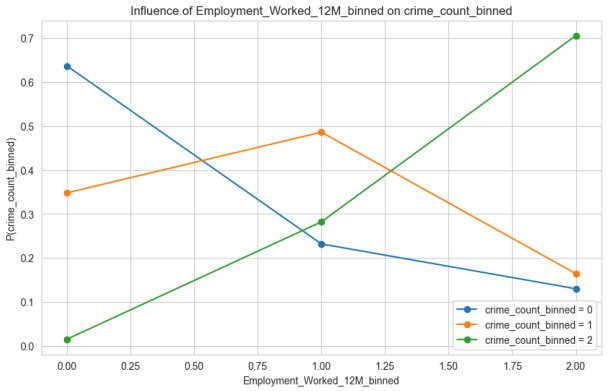


Fig. 17. Bayesian Network - Influence of "Employment_Worked_12M_binned" on Crime Counts

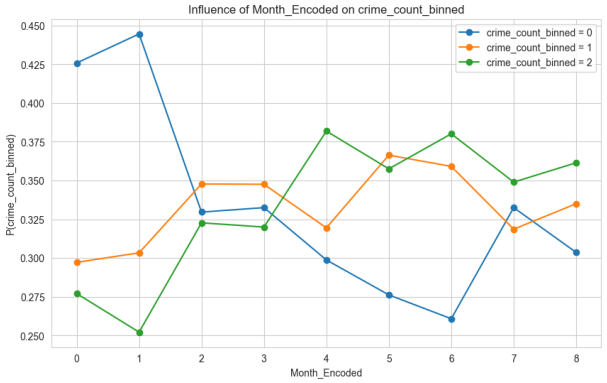


Fig. 18. Bayesian Network - Influence of "Month_Encoded" on Crime Counts

5) Influence of "Employment_Worked_12M_binned" on Crime: As seen in Figure **??**, higher employment levels surprisingly correlate with an increase in high crime counts. This contradicts traditional expectations that higher employment reduces crime.

6) Influence of "Month_Encoded" on Crime: Figure 18 reveals seasonal trends in crime, with notable fluctuations throughout the year.

7) Discussion: Socio-Economic Factors Driving Crime: The investigation of Bayesian networks shed light on the intricate connections between socioeconomic variables and criminal activity. Policies targeted at lessening financial hardship as a method of preventing crime are supported by the high correlation between unemployment and crime rates.

8) Policy Implications: When developing crime prevention methods, the Bayesian Network analysis emphasizes how crucial it is to take socioeconomic issues like unemployment, education, and stable migration into account. Given the higher likelihood of rising crime in places with high unemployment and erratic job patterns, policymakers ought to concentrate their efforts in these

areas.

F. Ethical Considerations

By coming up with trustworthy protection measures over data, adherence to regulations such as GDPR ensures the personal information of a person regarding their privacy and security. Equality can only be guaranteed through identification of the errors in data and in the models. Trust of the people can only be maintained by ensuring that predictive models are built with much accuracy and their usage is being given proper attention. Interventions should not disproportionately hurt the vulnerable populations.

V. Conclusion and Future Works

This study successfully integrated LSTM models, Explainable AI (XAI) techniques, and causal inference to analyze UK crime time series data. The LSTM model showed good performance in crime trend forecasting but faced generalization challenges during external disruptions like the COVID-19 pandemic. SHAP & LIME were employed to improve model interpretability. SHAP offering comprehensive global insights & LIME providing quick instance-specific explanations. Random Forest outperformed Linear Regression in predicting crime counts, highlighting unemployment, education, and employment instability as significant socio-economic factors influencing crime rates. Bayesian Network analysis confirmed causal relationships, identifying unemployment and education as key drivers of crime. For future work, incorporating external factors, which may responsible for crimes, such as whether data or public events or CCTV positions [41] [42], into the LSTM model could enhance robustness. Further exploration of hybrid models and advanced causal inference techniques, such as Structural Equation Models (SEM) [43], along with more granular socio-economic data, will improve prediction accuracy and provide deeper insights into crime dynamics.

References

[1] O. Z. Apene, N. V. Blamah, and G. I. Aimufua, "Advancements in crime prevention and detection: From traditional approaches to artificial intelligence solutions," *European Journal of Applied Science, Engineering and Technology*, vol. 2, no. 2, pp. 285–297, 2024.

[2] Y. Zhou, F. Wang, and S. Zhou, "The spatial patterns of the crime rate in london and its socio-economic influence factors," *Social Sciences*, vol. 12, no. 6, p. 340, 2023.

[3] E. Cesario, P. Lindia, and A. Vinci, "Multi-density crime predictor: An approach to forecast criminal activities in multi-density crime hotspots," *Journal of Big Data*, vol. 11, no. 1, 2024.

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?"," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[5] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, 2009.

[6] A. I. Weinberg *et al.*, "Causality from bottom to top: A survey," 2024. [Online]. Available: https://arxiv.org/html/2403.11219v1

[7] X. Zhang *et al.*, "Interpretable machine learning models for crime prediction," *Computers, Environment and Urban Systems*, vol. 94, p. 101789, 2022.

[8] J. Zeng, B. Ustun, and C. Rudin, "Interpretable classification models for recidivism prediction," *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 180, no. 3, p. 689–722, 2016.

[9] E. Parkar *et al.*, "Comparative study of deep learning explainability and causal ai for fraud detection," *International Journal on Smart Sensing and Intelligent Systems*, vol. 17, no. 1, 2024.

[10] D. Wang *et al.*, "Understanding the spatial distribution of crime based on its related variables using geospatial discriminative patterns," *Computers, Environment and Urban Systems*, vol. 39, p. 93–106, 2013.

[11] Office for Statistics Regulation, "The quality of police recorded crime statistics for england and wales," 2024. [Online]. Available: https://osr.statisticsauthority.gov.uk/publication/the-quality-of-police-recorded-crime-statistics-for-england-and-wales

[12] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *MDPI*, 2020. [Online]. Available: https://www.mdpi.com/1099-4300/23/1/18

[13] O. Kounadi *et al.*, "A systematic review on spatial crime forecasting," *Crime Science*, vol. 9, no. 1, 2020.

[14] J. Wu, S. M. Abrar *et al.*, "Enhancing short-term crime prediction with human mobility flows and deep learning architectures," *EPJ Data Science*, vol. 11, no. 1, 2022.

[15] S. Chainey, L. Tompson, and S. Uhlig, "The utility of hotspot mapping for predicting spatial patterns of crime," *Security Journal*, vol. 21, no. 1-2, pp. 4–28, 2008.

[16] H.-W. Kang and H.-B. Kang, "Prediction of crime occurrence from multi-modal data using deep learning," *PLOS ONE*, vol. 12, no. 4, 2017.

[17] A. Casolaro *et al.*, "Deep learning for time series forecasting: Advances and open problems," *Information*, vol. 14, no. 11, p. 598, 2023.

[18] V. Mandalapu *et al.*, "Crime prediction using machine learning and deep learning: A systematic review and future directions," *IEEE Access*, vol. 11, p. 60153–60170, 2023.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, 1997.

[20] D. Paul and D. Yeasin, "Tslstm: Long short term memory (lstm) model for time series forecasting," *CRAN: Contributed Packages*, 2022.

[21] U. M. Ramirez-Alcocer, E. Tello-Leal, and J. A. Mata-Torres, "Smart 2019: The eighth international conference on smart cities, systems, devices and technologies," 2019. [Online]. Available: https://personales.upv.es/thinkmind/SMART/SMART_2019/smart_2019_2_30_40010.html

[22] M. Solís and L.-A. Calvo-Valverde, "Deep learning for crime forecasting of multiple regions, considering spatial–temporal correlations between regions," in *ITISE 2024*, 2024.

[23] Y. Rayhan and T. Hashem, "Aist: An interpretable attention-based deep learning model for crime prediction," *ACM Transactions on Spatial Algorithms and Systems*, vol. 9, no. 2, p. 1–31, 2023.

[24] A. Chaddad *et al.*, "Survey of explainable ai techniques in healthcare," *Sensors*, vol. 23, no. 2, p. 634, 2023.

[25] C. O. Retzlaff *et al.*, "Post-hoc vs ante-hoc explanations: Xai design guidelines for data scientists," *Cognitive Systems Research*, vol. 86, p. 101243, 2024.

[26] A. Holzinger *et al.*, "Causability and explainability of artificial intelligence in medicine," *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 4, 2019.

[27] A. Gramegna and P. Giudici, "Shap and lime: An evaluation of discriminative power in credit risk," *Frontiers in Artificial Intelligence*, vol. 4, 2021.

[28] M. Louhichi *et al.*, "Shapley values for explaining the black box nature of machine learning model clustering," *Procedia Computer Science*, vol. 220, pp. 806–811, 2023.

[29] K. Baumgartner, S. Ferrari, and G. Palermo, "Constructing bayesian networks for criminal profiling from limited data," *Knowledge-Based Systems*, vol. 21, no. 7, pp. 563–572, 2008.

[30] T. Hu *et al.*, "Urban crime prediction based on spatio-temporal bayesian model," *PLOS ONE*, vol. 13, no. 10, 2018.

[31] Q. Wen *et al.*, "Robuststl: A robust seasonal-trend decomposition algorithm for long time series," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, p. 5409–5416, 2019.

[32] A. Sohrabbeig, O. Ardakanian, and P. Musilek, "Decompose and conquer: Time series forecasting with multiseasonal trend decomposition using loess," *Forecasting*, vol. 5, no. 4, p. 684–696, 2023.

[33] C. Huang *et al.*, "Deepcrime," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[34] O. Bespalova, "Gdp forecasts: Informational asymmetry of the spf and fomc minutes," *International Journal of Forecasting*, vol. 36, no. 4, pp. 1531–1540, 2020.

[35] D. R. Setiadi *et al.*, "Digital image steganography survey and investigation," *Signal Processing*, vol. 206, p. 108908, 2023.

[36] Q. Wen *et al.*, "Time series data augmentation for deep learning: A survey," *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, p. 4653–4660, 2021.

[37] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 1970. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[38] N. Jiang, K. Miao, Y. Chai, D. Lu, and J. Wu, "Spatio-temporal prediction of crime based on data mining and lstm network," *2023 IEEE 6th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 6, pp. 672–676, 2023.

[39] X. Zhang, L. Liu, L. Xiao, and J. Ji, "Comparison of machine learning algorithms for predicting crime hotspots," *IEEE Access*, vol. 8, pp. 181 302–181 310, 2020.

[40] T. P. Lowe *et al.*, "Violent crime and covid-19 in england and wales," *Studies of Organized Crime*, pp. 279–294, 2022.

[41] M. Ranson, "Crime, weather, and climate change," *Journal of Environmental Economics and Management*, vol. 67, no. 3, pp. 274–302, 2014.

[42] B. C. Welsh and D. P. Farrington, "Public area cctv and crime prevention: An updated systematic review and meta-analysis," *Justice Quarterly*, vol. 26, no. 4, pp. 716–745, 2009.

[43] A. S. Albahri *et al.*, "Hybrid artificial neural network and structural equation modeling techniques: A survey," *Complex & Intelligent Systems*, vol. 8, no. 2, pp. 1781–1801, 2021.

APPENDIX A
DATASET IMAGES

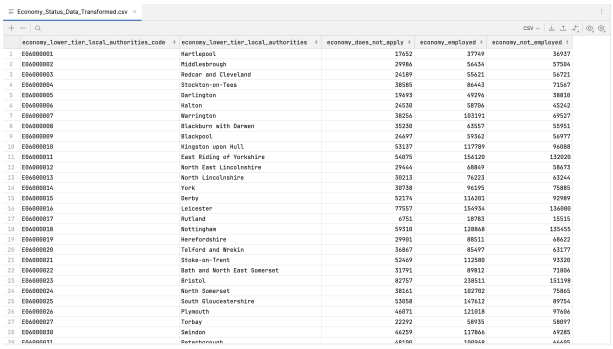


Fig. 19. Economy status

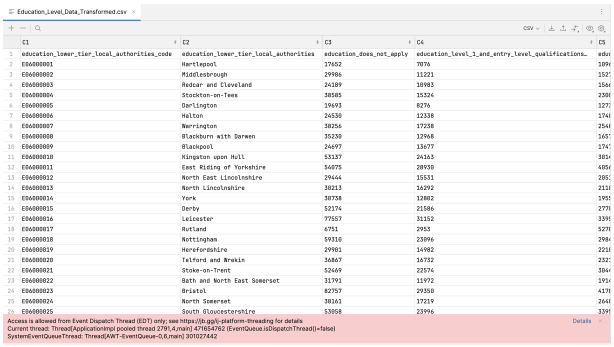


Fig. 20. Education Level

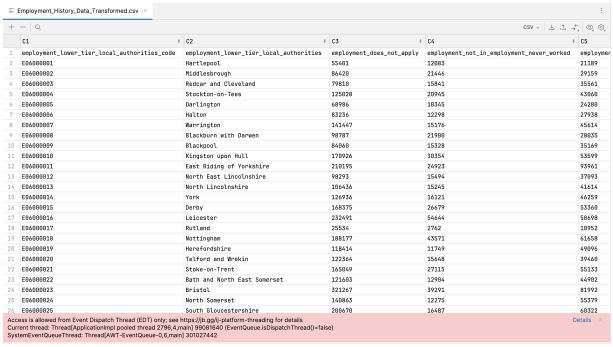
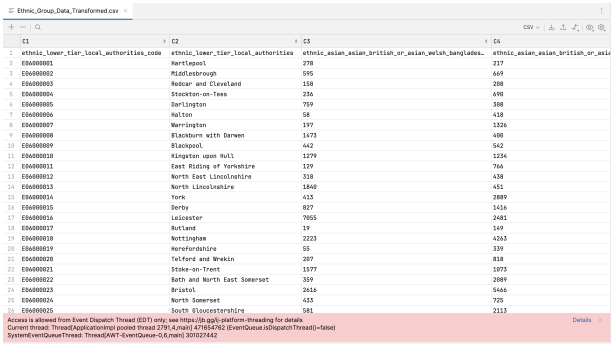


Fig. 21. Employment History Data



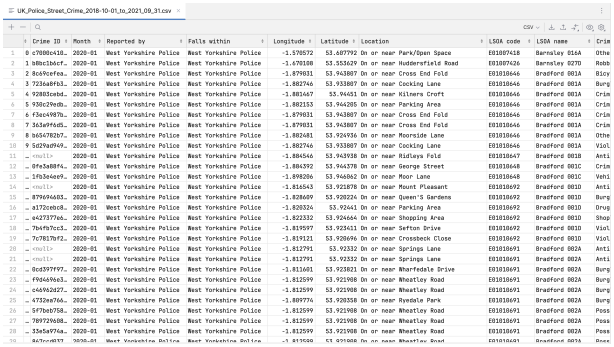
Ethnic_Group_Data_Transformed.csv

	C1	C2	C3	C4
1	ethnic_lower_tier_local_authorities_code	ethnic_lower_tier_local_authorities	ethnic_asian_asian_british_or_asian_welsh_banglades…	ethnic_asian_asian_british_or_asia
2	E06000001	Hartlepool	278	217
3	E06000002	Middlesbrough	595	669
4	E06000003	Redcar and Cleveland	158	208
5	E06000004	Stockton-on-Tees	236	490
6	E06000005	Darlington	759	308
7	E06000006	Halton	58	418
8	E06000007	Warrington	197	1326
9	E06000008	Blackburn with Darwen	1473	488
10	E06000009	Blackpool	442	542
11	E06000010	Kingston upon Hull	1279	1234
12	E06000011	East Riding of Yorkshire	129	766
13	E06000012	North East Lincolnshire	318	438
14	E06000013	North Lincolnshire	1840	451
15	E06000014	York	413	2889
16	E06000015	Derby	827	1416
17	E06000016	Leicester	7055	2481
18	E06000017	Rutland	19	149
19	E06000018	Nottingham	2223	4263
20	E06000019	Herefordshire	55	339
21	E06000020	Telford and Wrekin	207	818
22	E06000021	Stoke-on-Trent	1577	1073
23	E06000022	Bath and North East Somerset	359	2089
24	E06000023	Bristol	2616	5466
25	E06000024	North Somerset	433	725
26	E06000025	South Gloucestershire	581	2113

Access is allowed from Event Dispatch Thread (EDT) only; see https://jb.gg/ij-platform-threading for details
Current thread: Thread[ApplicationImpl pooled thread 279],4,main] 471654762 [EventQueue.isDispatchThread()=false]
SystemEventQueueThread: Thread[AWT-EventQueue-0,6,main] 301027442

Details

Fig. 22. Ethnic Data



UK_Police_Street_Crime_2018-10-01_to_2021_09_31.csv

		Crime ID	Month	Reported by	Falls within	Longitude	Latitude	Location	LSOA code	LSOA name	Crim
1	0	c7000c410…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.570572	53.607792	On or near Park/Open Space	E01007418	Barnsley 016A	Othe
2	1	b8bc1b6cf…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.670108	53.553629	On or near Huddersfield Road	E01007426	Barnsley 027D	Robb
3	2	8c69cefea…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.879031	53.943807	On or near Cross End Fold	E01010646	Bradford 001A	Bicy
4	3	7216a6fb1…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.882746	53.933807	On or near Cocking Lane	E01010646	Bradford 001A	Burg
5	4	92803cebd…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.881467	53.94451	On or near Kilnera Croft	E01010646	Bradford 001A	Crim
6	5	930c29edb…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.882153	53.944205	On or near Parking Area	E01010646	Bradford 001A	Crim
7	6	f3dc4987b…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.879031	53.943807	On or near Cross End Fold	E01010646	Bradford 001A	Crim
8	7	3d3a9fbd5…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.879031	53.943807	On or near Cross End Fold	E01010646	Bradford 001A	Crim
9	8	be5478b2e…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.882481	53.924936	On or near Moorside Lane	E01010646	Bradford 001A	Othe
10	9	5d29ad949…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.882746	53.933807	On or near Cocking Lane	E01010646	Bradford 001A	Viol
11	…	<null>	2020-01	West Yorkshire Police	West Yorkshire Police	-1.884544	53.943938	On or near Ridleys Fold	E01010647	Bradford 001B	Anti
12	…	0fe3a884c…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.884392	53.944378	On or near George Street	E01010648	Bradford 001C	Crim
13	…	1fb3e4ae9…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.898286	53.946862	On or near Moor Lane	E01010648	Bradford 001C	Vehi
14	…	<null>	2020-01	West Yorkshire Police	West Yorkshire Police	-1.816543	53.921878	On or near Mount Pleasant	E01010692	Bradford 001D	Anti
15	…	87969da65…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.820609	53.920226	On or near Queen'S Gardens	E01010692	Bradford 001D	Burg
16	…	a172cebc8…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.820324	53.92441	On or near Parking Area	E01010692	Bradford 001D	Drug
17	…	e4273776e…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.822332	53.924664	On or near Shopping Area	E01010692	Bradford 001D	Shop
18	…	7b4fb7cc3…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.815597	53.923411	On or near Sefton Drive	E01010692	Bradford 001D	Viol
19	…	7c7817bf2…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.819121	53.928696	On or near Crossbeck Close	E01010692	Bradford 001D	Viol
20	…	<null>	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812791	53.92332	On or near Springs Lane	E01010691	Bradford 002A	Anti
21	…	<null>	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812791	53.92332	On or near Springs Lane	E01010691	Bradford 002A	Anti
22	…	0c6377f97…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.811601	53.923021	On or near Wharfedale Drive	E01010691	Bradford 002A	Burg
23	…	f9464a9a1…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812599	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Burg
24	…	c4496d627…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812599	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Burg
25	…	4732ee766…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.809774	53.920358	On or near Ryedale Park	E01010691	Bradford 002A	Burg
26	…	5f7bdd758…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812599	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Poss
27	…	789729688…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812599	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Poss
28	…	33e5a974a…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.812599	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Poss
29	…	847ce4837…	2020-01	West Yorkshire Police	West Yorkshire Police	-1.813500	53.921908	On or near Wheatley Road	E01010691	Bradford 002A	Poss

Fig. 23. UK Crime Data

APPENDIX B
ANNOTATED PAPERS

You can access the folder containing the papers used in the study at the following link:

Click here to view the papers

APPENDIX C
LSTM MODEL CODE

```
#%% md
# Crime Data Analysis and Forecasting using LSTM and Explainability Techniques

This notebook demonstrates the process of analyzing crime data, forecasting future trends using an LSTM model, and interpreting the model's predictions using LIME and SHAP.
#%% md
# 1. Imports

Let's begin by collecting all the import statements together at the beginning. This ensures that all dependencies are loaded before any logic is executed.

#%%
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
from statsmodels.tsa.seasonal import seasonal_decompose
from lime import lime_tabular
import shap
import os  # Added to handle file paths more robustly

#%% md
# 2. Loading and Preparing the Dataset

Now, I'll load the dataset and perform some initial data cleaning and transformation:

#%%
# Define the file path more robustly using os.path
file_path = os.path.join('/Users', 'arkamandol', 'DataspellProjects', 'Desertation_arka_23023023', 'data_files', 'UK_Police_Street_Crime_2018-10-01_to_2021_09_31.csv')

# Load the data from the specified file path
try:
    df = pd.read_csv(file_path)
except FileNotFoundError:
    print(f"File not found: {file_path}")
    raise
except Exception as e:
    print(f"An error occurred while loading the file: {e}")
    raise

# Display the initial rows of the dataset to understand its structure
df.head()

# 1. Remove any rows where 'Longitude' or 'Latitude' values are missing
df_cleaned = df.dropna(subset=['Longitude', 'Latitude'])
```

```

# 2. Focus the dataset on "Violence and sexual offences" crimes
df_cleaned = df_cleaned[df_cleaned['Crime type'] == "Violence and sexual offences"]

# 3. Keep only the necessary columns for analysis
columns_to_keep = ['Month', 'Longitude', 'Latitude']
df_cleaned = df_cleaned[columns_to_keep]

#%% md
**Explanation**:
- **File Path Handling**: Improved the file path handling with `os.path.join()` for better cross-platform compatibility.
- **Error Handling**: Added exception handling during file loading to catch and report issues like file not found or other IO errors.
- **Data Cleaning**: The data is filtered to remove rows with missing geographical information and focus on specific crime types.

#%% md
# 3. Time Series Transformation

Here I transform the data into a time series format suitable for analysis.

#%% md

#%%
# Convert the 'Month' column to datetime format for time series analysis
data = df_cleaned.copy()
data['Month'] = pd.to_datetime(data['Month'])

# Set the 'Month' column as the index
data.set_index('Month', inplace=True)

# Aggregate the data to get the number of crimes per month
monthly_crime_count = data.resample('M').size()

# Display the first few entries of the aggregated data
monthly_crime_count.head()

#%% md
**Explanation**:
- **Datetime Conversion**: Converts the 'Month' column into a datetime format, which is essential for time series operations.
- **Indexing**: The 'Month' column is set as the index to facilitate resampling and time series analysis.
- **Aggregation**: The number of crimes per month is calculated.

#%% md
# 4. Visualization of Time Series Data

Next, visualize the monthly crime counts over time.

#%%
# Generate a line plot to visualize monthly crime counts over time
plt.figure(figsize=(14, 7))
monthly_crime_count.plot()
plt.title('Monthly Crime Counts')
plt.xlabel('Month')
plt.ylabel('Number of Crimes')
plt.grid(True)
plt.show()

#%% md
**Explanation**:
- **Visualization**: A line plot is generated to visualize the crime trend over time, which helps in understanding the data distribution.

#%% md
# 5. Seasonal Decomposition of Time Series

Decompose the time series to analyze the underlying components:

#%%
# Decompose the time series into trend, seasonal, and residual components
decomposition = seasonal_decompose(monthly_crime_count, model='additive')

# Plot the decomposed components to better understand the underlying patterns
fig = decomposition.plot()
fig.set_size_inches(14, 7)
plt.show()

#%% md
**Explanation**:
- **Decomposition**: The time series is decomposed into trend, seasonal, and residual components to identify underlying patterns in the data.

#%% md
# 6. LSTM Model for Time Series Forecasting

Now need to build and train an LSTM model to predict future crime counts:

#%%
# Convert the series to a supervised learning dataset
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step):
        X.append(data[i:(i + time_step), 0])
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

# Reshape the data into a format suitable for LSTM
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(monthly_crime_count.values.reshape(-1, 1))

# Create sequences of 12 months (1 year)
time_step = 12
X, Y = create_dataset(scaled_data, time_step)

# Reshape input to be [samples, time steps, features]
X = X.reshape(X.shape[0], X.shape[1], 1)

# Split data into training and testing sets
```

```
train_size = int(len(X) * 0.7)
test_size = len(X) - train_size
X_train, X_test = X[:train_size], X[train_size:]
Y_train, Y_test = Y[:train_size], Y[train_size:]

#%% md
**Explanation**:
- **Supervised Dataset Creation**: The `create_dataset` function converts the time series data into sequences for supervised learning.
    - **Normalization**: The data is scaled between 0 and 1 using MinMaxScaler, which helps in stabilizing the LSTM model training.
- **Train-Test Split**: The dataset is split into training and testing sets, maintaining 70% of the data for training.

#%% md
# 7. Building the LSTM Model

Let's define, compile, and train the LSTM model:

#%%
# Define the LSTM model with dropout
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(Dropout(0.3))  # Adding dropout after the first LSTM layer
model.add(LSTM(50, return_sequences=False))
model.add(Dropout(0.3))  # Adding dropout after the second LSTM layer
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, Y_train, batch_size=1, epochs=500)

#%% md
**Explanation**:
- **Model Architecture**: A Sequential LSTM model is defined with two LSTM layers, each followed by a Dropout layer to prevent overfitting. The model ends with Dense layers to produce the final output.
- **Compilation**: The model is compiled with the Adam optimizer and mean squared error loss function.
- **Training**: The model is trained with a small batch size and a large number of epochs for better learning.

#%% md
# 8. Making Predictions and Evaluating the Model

After training,predict and evaluate the model:

#%%
# Predict using the trained model
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# Inverse transform the predictions to original scale
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
Y_train = scaler.inverse_transform([Y_train])
Y_test = scaler.inverse_transform([Y_test])

# Calculate MSE, RMSE and MAE for training and testing data
train_mse = mean_squared_error(Y_train.T, train_predict)
train_rmse = np.sqrt(train_mse)
train_mae = mean_absolute_error(Y_train.T, train_predict)
test_mse = mean_squared_error(Y_test.T, test_predict)
test_rmse = np.sqrt(test_mse)
test_mae = mean_absolute_error(Y_test.T, test_predict)

print(f'Train MSE: {train_mse}')
print(f'Train RMSE: {train_rmse}')
print(f'Train MAE: {train_mae}')
print(f'Test MSE: {test_mse}')
print(f'Test RMSE: {test_rmse}')
print(f'Test MAE: {test_mae}')

#%% md
**Explanation**:
- **Predictions**: The model is used to predict both the training and testing data.
- **Inverse Scaling**: The predictions are transformed back to the original scale.
- **Evaluation**: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) are calculated to evaluate model performance.

#%% md
# 9. Future Predictions

Predicting crime counts for the next 6 months:

#%%
# Predict the next 6 months (October 2021 to March 2022)
future_steps = 6
future_predictions = []
last_sequence = X[-1]  # The last sequence of the test set

for _ in range(future_steps):
    next_pred = model.predict(last_sequence.reshape(1, time_step, 1))
    future_predictions.append(next_pred[0, 0])
    last_sequence = np.append(last_sequence[1:], next_pred, axis=0)

# Inverse transform future predictions
future_predictions = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))

# Generate date range for predictions
last_date = monthly_crime_count.index[-1]
future_dates = pd.date_range(last_date + pd.DateOffset(months=1), periods=future_steps,
```

```
    freq='M')

#%% md
**Explanation**:
- **Future Predictions**: The model is used to predict the next 6 months of crime counts.
- **Date Range Generation**: A corresponding date range is generated for the predicted values.

#%% md
# 10. Visualization of Predictions

Plotting actual vs predicted crime counts:

#%%
# Plot the results
plt.figure(figsize=(14, 7))
plt.plot(monthly_crime_count.index, monthly_crime_count.values, label='Original Data', color='blue')
plt.plot(monthly_crime_count.index[time_step: time_step + len(train_predict)], train_predict,
         label='Training Predictions', color='orange')
plt.plot(monthly_crime_count.index[time_step + len(train_predict):time_step + len(train_predict) + len(test_predict)],
         test_predict, label='Testing Predictions', color='green')
plt.plot(future_dates, future_predictions, label='Future Predictions (Oct 2021 - Mar 2022)', color='red')

plt.title('LSTM Model Predictions vs. Actual Data')
plt.xlabel('Time')
plt.ylabel('Crime Count')
plt.xticks(rotation=45)
plt.grid(True, which='both', axis='x', linestyle='--', linewidth=0.7)
plt.legend()

# Set the x-ticks to be all months in the range
plt.gca().set_xticks(pd.date_range(start=monthly_crime_count.index[0], end=future_dates[-1], freq='M'))

# Format the x-ticks to show month and year
plt.gca().set_xticklabels([date.strftime('%Y-%m') for date in
                           pd.date_range(start=monthly_crime_count.index[0], end=future_dates[-1], freq='M')])

plt.show()

#%% md
**Explanation**:
- **Visualization**: The plot compares the original data with both the training/testing predictions and future predictions, showing how well the model fits and forecasts.

#%% md
# 11. LIME for Model Interpretability

Using LIME to explain the predictions of the LSTM model:

#%%
# Define the prediction function for LIME
def lstm_predict(input_data):
    """ Wrapper function to adapt input data for the LSTM model's expected input shape """
    reshaped_data = input_data.reshape(input_data.shape[0], time_step, 1)  # Reshape to [samples, time steps, features]
    return model.predict(reshaped_data)

# Reshape X_train for LIME
X_train_2d = X_train.reshape(X_train.shape[0], X_train.shape[1])

# Initialize the LIME Explainer
explainer = lime_tabular.LimeTabularExplainer(
    training_data=X_train_2d,
    feature_names=['t-' + str(i) for i in range(X_train.shape[1])],  # Create feature names for each time step
    mode='regression'
)

# Loop through each instance in X_test
lime_explanations = []

for i in range(X_test.shape[0]):
    # Reshape test instance to 2D for LIME
    test_instance = X_test[i].reshape(1, -1)[0]

    # Generate explanation for the current instance
    exp = explainer.explain_instance(
        data_row=test_instance,
        predict_fn=lstm_predict,
        num_features=12  # Set this to ensure all features are considered
    )

    lime_explanations.append(exp)

    # Optionally, print or plot each explanation
    print(f"Explanation for instance {i}:")
    exp.show_in_notebook(show_table=True, show_all=False)

    # Plotting each explanation
    features, weights = zip(*[(i[0], i[1]) for i in exp.local_exp[0]])
    features = [f"t-{X_train.shape[1] - idx}" for idx in features]

    plt.figure(figsize=(10, 5))
    plt.bar(features, weights, color='skyblue')
    plt.xlabel('Features (Time Steps)')
    plt.ylabel('Weights')
    plt.title(f'LIME Feature Contributions for Instance {i}')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

#%% md
**Explanation**:
- **LIME Explanation**: The code uses LIME to interpret the LSTM models predictions, identifying the most influential features for each prediction.
```

```

#%% md
# 12. SHAP for Model Interpretability

Lastly, SHAP is used to interpret the model:

#%%
# Define a custom prediction function that reshapes the input before predicting
def predict_reshaped(data):
    reshaped_data = data.reshape(data.shape[0], -1, 1)
    return model.predict(reshaped_data)

# Use a small subset of training data as background for KernelExplainer
background = X_train.reshape(X_train.shape[0], -1)  # Flatten to 2D for KernelExplainer

# Initialize KernelExplainer with the custom predict function
explainer = shap.KernelExplainer(predict_reshaped, background)

# Compute SHAP values for test set (reshaped to 2D)
X_test_2d = X_test.reshape(X_test.shape[0], -1)  # Flatten to 2D
shap_values = explainer.shap_values(X_test_2d, nsamples=100)

# Reshape SHAP values from (8, 12, 1) to (8, 12)
shap_values_reshaped = np.array(shap_values).reshape(X_test.shape[0], X_test.shape[1])

# Now use SHAP's summary plot, passing in the original X_test without the last dimension
X_test_reshaped = X_test.reshape(X_test.shape[0], X_test.shape[1])  # Remove the last dimension
shap.summary_plot(shap_values_reshaped, X_test_reshaped)
```

Listing 1. LSTM Model Implementation

APPENDIX D
BAYESIAN NETWORK CODE

```

#%% md

#%% md
# Crime Data Analysis and Bayesian Network Modeling

#%% md
## 1. Imports and Setup

#%% md
### Imports and Setup

- **pandas**: For data manipulation and analysis.
- **numpy**: For numerical operations.
- **matplotlib and seaborn**: For visualization.
- **sklearn**: For machine learning models and preprocessing.
- **pgmpy**: For building and analyzing Bayesian networks.
- **shap and lime**: For model explainability (although not used explicitly in the code).

will add the imports at the beginning of the script to ensure all dependencies are loaded before executing any logic.

#%%
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pgmpy.estimators import HillClimbSearch, BicScore, BayesianEstimator, K2Score
from pgmpy.models import BayesianNetwork
from pgmpy.inference import VariableElimination
import networkx as nx

#%% md
## 2. Loading and Preprocessing Crime Data
#%% md
### Loading and Preprocessing Crime Data

- **Loading Crime Data**: load the dataset containing crime data from a specified CSV file.
- **Date Conversion**: The 'Month' column is converted to datetime format for easier filtering and manipulation.
    - **Filtering Data**: The data is filtered to retain only records from the year 2021 and those related to the "Violence and sexual offences" crime type.

#%%
# Load the dataset containing crime data
crime_data_path = "/Users/arkamandol/Dataspell Projects/Desertation_arka_23023023/data_files/UK_Police_Street_Crime_2018-10-01_to_2021_09_31.csv"
crime_df = pd.read_csv(crime_data_path)

# Convert 'Month' to datetime and filter for 2021 and specific crime type
crime_df['Month'] = pd.to_datetime(crime_df['Month'])
```

```
filtered_crime_df = crime_df[(crime_df['Month'].dt.year == 2021) &
                             (crime_df['Crime type'] == "Violence and sexual offences")]

#%% md
## 3. Data Cleaning and Aggregation

#%% md
### Data Cleaning and Aggregation

- **Modifying 'LSOA name'**: modify the 'LSOA name' column to remove the last 5 characters, which might represent redundant information.
- **Renaming Columns**: The 'LSOA name' column is renamed to 'lower_tier_local_authorities' for clarity.
- **Aggregating Data**: The data is aggregated by 'Month' and 'lower_tier_local_authorities', and count the occurrences of crimes within each group.

#%%
# Modify 'LSOA name' to remove the last 5 characters
filtered_crime_df['LSOA name'] = filtered_crime_df['LSOA name'].str[:-5]

# Rename 'LSOA name' to 'lower_tier_local_authorities'
filtered_crime_df.rename(columns={'LSOA name': 'lower_tier_local_authorities'}, inplace=True)

# Aggregate the data by 'Month' and 'lower_tier_local_authorities' and count occurrences
aggregated_crime_data = filtered_crime_df.groupby(['Month', 'lower_tier_local_authorities']).size().reset_index(name='crime_count')

#%% md
## 4. Loading and Merging Socioeconomic Data

#%% md
### Loading and Merging Socioeconomic Data

- **Loading Socioeconomic Datasets**: load multiple CSV files containing different socioeconomic data such as ethnic groups, economy status, employment history, education level, and migrant data.
- **Renaming Columns**: In each dataset, the locality code columns are renamed to 'area_code' for consistency.
- **Merging Datasets**: All the socioeconomic datasets are merged into a single DataFrame based on the 'area_code' column.

#%%
# Load datasets
ethnic_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Ethnic_Group_Data_Transformed.csv")
economy_status_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Economy_Status_Data_Transformed.csv")
employment_history_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Employment_History_Data_Transformed.csv")
education_level_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Education_Level_Data_Transformed.csv")
migrant_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Migrant_Data_Transformed.csv")

# Rename the locality code columns to 'area_code'
ethnic_data.rename(columns={'ethnic_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
economy_status_data.rename(columns={'economy_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
employment_history_data.rename(columns={'employment_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
education_level_data.rename(columns={'education_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
migrant_data.rename(columns={'migrant_lower_tier_local_authorities_code': 'area_code'}, inplace=True)

# Merge all datasets on the 'area_code' column
merged_data = ethnic_data.merge(economy_status_data, on='area_code', how='outer')
merged_data = merged_data.merge(employment_history_data, on='area_code', how='outer')
merged_data = merged_data.merge(education_level_data, on='area_code', how='outer')
merged_data = merged_data.merge(migrant_data, on='area_code', how='outer')

#%% md
## 5. Cleaning and Merging Final Dataset

#%% md
### Cleaning and Merging Final Dataset

- **Dropping Redundant Columns**: identify and drop any redundant 'lower_tier_local_authorities' columns from the merged dataset.
- **Final Merging**: The cleaned socioeconomic data is merged with the aggregated crime data using the 'lower_tier_local_authorities' column.
- **Initial Data Inspection**: inspect the first few rows and the structure of the final merged dataset to ensure correctness.

#%%
# Load the additional dataset
additional_df = merged_data.copy()

# Identify and drop redundant 'lower_tier_local_authorities' columns
columns_to_drop = [col for col in additional_df.columns if 'lower_tier_local_authorities'
```

```
    in col]
columns_to_drop.remove('ethnic_lower_tier_local_authorities')  # Keep one relevant column

# Drop the identified columns
df_cleaned = additional_df.drop(columns=columns_to_drop)

# Rename the relevant column to 'lower_tier_local_authorities'
df_cleaned.rename(columns={'ethnic_lower_tier_local_authorities': 'lower_tier_local_authorities'}, inplace=True)

# Merge the crime data with the cleaned dataset on 'lower_tier_local_authorities'
merged_data = aggregated_crime_data.merge(df_cleaned, on='lower_tier_local_authorities', how='inner')

# Display basic information and initial rows of the merged dataset
print(merged_data.head())
print(merged_data.info())

#%% md
## 6. Data Preparation for Analysis

#%% md
### Data Preparation for Analysis

- **Dropping Unnecessary Columns**: drop columns that are not needed for further analysis, like 'lower_tier_local_authorities' and 'area_code'.
- **Renaming Columns**: rename certain columns to shorter, more intuitive names for easier reference.

#%%
# Create a copy for processing
data = merged_data.copy()

# Drop columns not needed for analysis
data.drop(columns=['lower_tier_local_authorities', 'area_code'], inplace=True)

# Mapping new column names for easier reference
new_column_names = {
    'ethnic_asian_asian_british_or_asian_welsh_bangladeshi': 'Ethnic_Asian_Bangladeshi',
    'ethnic_asian_asian_british_or_asian_welsh_chinese': 'Ethnic_Asian_Chinese',
    'ethnic_asian_asian_british_or_asian_welsh_indian': 'Ethnic_Asian_Indian',
    'ethnic_asian_asian_british_or_asian_welsh_other': 'Ethnic_Asian_Other',
    'ethnic_asian_asian_british_or_asian_welsh_pakistani': 'Ethnic_Asian_Pakistani',
    'ethnic_black_black_british_black_welsh_caribbean_or_african': 'Ethnic_Black_African',
    'ethnic_black_black_british_black_welsh_caribbean_or_african': 'Ethnic_Black_Caribbean',
    'ethnic_black_black_british_black_welsh_caribbean_or_african': 'Ethnic_Black_Other',
    'ethnic_mixed_or_multiple_ethnic_groups_other_mixed_or_mult': 'Ethnic_Mixed_Other',
    'ethnic_mixed_or_multiple_ethnic_groups_white_and_asian': 'Ethnic_Mixed_White_Asian',
    'ethnic_mixed_or_multiple_ethnic_groups_white_and_black_afri': 'Ethnic_Mixed_White_Black_African',
    'ethnic_mixed_or_multiple_ethnic_groups_white_and_black_cari': 'Ethnic_Mixed_White_Black_Caribbean',
    'ethnic_other_ethnic_group_any_other_ethnic_group': 'Ethnic_Other',
    'ethnic_other_ethnic_group_arab': 'Ethnic_Arab',
    'ethnic_white_english_welsh_scottish_northern_irish_or_briti': 'Ethnic_White_British',
    'ethnic_white_gypsy_or_irish_traveller': 'Ethnic_White_Gypsy_Traveller',
    'ethnic_white_irish': 'Ethnic_White_Irish',
    'ethnic_white_other_white': 'Ethnic_White_Other',
    'ethnic_white_roma': 'Ethnic_White_Roma',
    'economy_employed': 'Economy_Employed',
    'economy_not_employed': 'Economy_Not_Employed',
    'employment_not_in_employment_never_worked': 'Employment_Never_Worked',
    'employment_not_in_employment_not_worked_in_the_last_12_mont': 'Employment_Not_Worked_12M',
    'employment_not_in_employment_worked_in_the_last_12_months': 'Employment_Worked_12M',
    'education_level_1_and_entry_level_qualifications_1_to_4_gcs*_to_c_any_gcses_at_other_grades_o_levels_or_cses_any_grade': 'Education_Level_1',
    'education_level_2_qualifications_5_or_more_gcses_a*_to_c_or_9_to_4_o_levels_passes_cses_grade_1_school_certifi': 'Education_Level_2',
    'education_level_3_qualifications_2_or_more_a_levels_or_vces;_advanced_gnvq_city_and_guilds_advanced_craft_onc_ond_btec_': 'Education_Level_3',
    'education_level_4_qualifications_or_above_degree_ba_bsc_hig': 'Education_Level_4+',
    'education_no_qualifications': 'Education_No_Qualifications',
    'education_other_apprenticeships_vocational_or_work-related_qualifications_other_qualifications_achieved_in_engla': 'Education_Other_Qualifications',
    'migrant_address_one_year_ago_is_student_term-time_or_boarding_school_address_in_the_uk'
```

```
    : 'Migrant_Student_UK',
    'migrant_address_one_year_ago_is_the_same_as_th': 'Migrant_Same_Address',
    'migrant_does_not_apply': 'Migrant_NA',
    'migrant_migrant_from_outside_the_uk_address_on': 'Migrant_Outside_UK',
    'migrant_migrant_from_within_the_uk_address_one': 'Migrant_Within_UK'
}

# Rename columns according to the mapping
data.rename(columns=new_column_names, inplace=True)

#%% md
## 7. Correlation Analysis and Visualization

#%% md
### Correlation Analysis and Visualization

- **Correlation Matrix Calculation**: calculate the correlation matrix for the dataset to identify relationships between variables.
- **Heatmap Visualization**: A heatmap is plotted to visualize the correlation matrix, making it easier to spot strong correlations.

#%%
# Calculate the correlation matrix
correlation_matrix = data.corr()

# Set the size of the plot
plt.figure(figsize=(10, 8))  # Increase the figure size for better readability

# Create the heatmap with annotations
heatmap = sns.heatmap(correlation_matrix, annot=True, fmt=".1f", annot_kws={'size': 6}, cmap='coolwarm', linewidths=.5)

# Rotate the x and y labels for better readability
plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
plt.yticks(rotation=0)   # y-axis labels can be left horizontal or adjusted as needed

# Show the plot
plt.show()

#%% md
## 8. Distribution Analysis

#%% md
### Distribution Analysis

- **Variable Selection**: select key variables related to crime, education, migration, employment, economy, and ethnicity for further analysis.
    - **Distribution Plots**: plot the distribution of these variables using histograms with kernel density estimation (KDE) to understand their underlying distribution.

#%%
# List of key variables
variables = ['crime_count', 'Education_Level_3', 'Migrant_Same_Address', 'Employment_Worked_12M', 'Economy_Not_Employed', 'Ethnic_White_British']

# Plotting the distributions
plt.figure(figsize=(15, 10))
for i, var in enumerate(variables, 1):
    plt.subplot(3, 2, i)
    sns.histplot(data[var], kde=True)
    plt.title(f'Distribution of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

#%% md
## 9. Logarithmic Transformation

#%% md
### Logarithmic Transformation

- **Log Transformation**: apply a logarithmic transformation to the selected variables to normalize their distribution and reduce skewness. This is particularly useful when variables have outliers or are highly skewed.
- **Log-Transformed Distribution Plots**: plot the distributions of the log-transformed variables to compare them with the original distributions.

#%%
# Applying logarithmic transformation, adding 1 to avoid log(0) issue
log_transformed_data = data.copy()
log_transformed_data['crime_count_log'] = np.log1p(log_transformed_data['crime_count'])
log_transformed_data['Education_Level_3_log'] = np.log1p(log_transformed_data['Education_Level_3'])
log_transformed_data['Migrant_Same_Address_log'] = np.log1p(log_transformed_data['Migrant_Same_Address'])
log_transformed_data['Employment_Worked_12M_log'] = np.log1p(log_transformed_data['Employment_Worked_12M'])
log_transformed_data['Economy_Not_Employed_log'] = np.log1p(log_transformed_data['Economy_Not_Employed'])
log_transformed_data['Ethnic_White_British_log'] = np.log1p(log_transformed_data['Ethnic_White_British'])

# Plotting the log-transformed distributions
plt.figure(figsize=(15, 10))
for i, var in enumerate(['crime_count_log', 'Education_Level_3_log', 'Migrant_Same_Address_log', 'Employment_Worked_12M_log', 'Economy_Not_Employed_log', 'Ethnic_White_British_log'], 1):
    plt.subplot(3, 2, i)
    sns.histplot(log_transformed_data[var], kde=True)
    plt.title(f'Log-Transformed Distribution of
```

```
    {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')

plt.tight_layout()
plt.show()

#%% md
## 10. Quantile Binning

#%% md
### Quantile Binning

- **Quantile Binning**: The log-transformed variables are further processed using quantile binning to categorize them into discrete categories (e.g., 'Low', 'Medium', 'High').
- **Binned Data**: The binned data provides a simplified view of the variables, making it easier to analyze and interpret their relationships.

#%%
# Applying quantile binning on log-transformed variables
log_binned_data = log_transformed_data.copy()

# Defining the number of bins (e.g., 3 bins for low, medium, high)
n_bins = 3

# Applying quantile binning
log_binned_data['crime_count_binned'] = pd.qcut(log_binned_data['crime_count_log'], q=n_bins, labels=['Low', 'Medium', 'High'])
log_binned_data['Education_Level_3_binned'] = pd.qcut(log_binned_data['Education_Level_3_log'], q=n_bins, labels=['Low', 'Medium', 'High'])
log_binned_data['Migrant_Same_Address_binned'] = pd.qcut(log_binned_data['Migrant_Same_Address_log'], q=n_bins, labels=['Low', 'Medium', 'High'])
log_binned_data['Employment_Worked_12M_binned'] = pd.qcut(log_binned_data['Employment_Worked_12M_log'], q=n_bins, labels=['Low', 'Medium', 'High'])
log_binned_data['Economy_Not_Employed_binned'] = pd.qcut(log_binned_data['Economy_Not_Employed_log'], q=n_bins, labels=['Low', 'Medium', 'High'])
log_binned_data['Ethnic_White_British_binned'] = pd.qcut(log_binned_data['Ethnic_White_British_log'], q=n_bins, labels=['Low', 'Medium', 'High'])

# Display the first few rows of the binned data
print(log_binned_data[['crime_count_binned', 'Education_Level_3_binned', 'Migrant_Same_Address_binned', 'Employment_Worked_12M_binned', 'Economy_Not_Employed_binned', 'Ethnic_White_British_binned']].head())

#%% md
## 11. Categorical Encoding of Date Variables

#%% md
### Categorical Encoding of Date Variables

- **Month Encoding**: The 'Month' column, which is initially in datetime format, is converted into a categorical variable representing just the month. This is useful for modeling purposes where month-based seasonality might be relevant.

#%%
# Convert 'Month' to a datetime object if it isn't already
log_binned_data['Month'] = pd.to_datetime(log_binned_data['Month'])

# Extract the month part and encode it as a categorical variable
log_binned_data['Month_Encoded'] = log_binned_data['Month'].dt.month.astype('category')

# Display the first few rows to verify the changes
print(log_binned_data[['Month', 'Month_Encoded']].head())

#%% md
## 12. Bayesian Network Structure Learning

#%% md
### Bayesian Network Structure Learning

- **Data Preparation for Bayesian Network**: select relevant binned and encoded variables to build a Bayesian Network model. Categorical variables are converted into numerical codes required for Bayesian modeling.
- **Structure Learning**: The structure of the Bayesian Network is learned using Hill-Climb Search with BIC (Bayesian Information Criterion) and K2 scoring methods. The learned structure provides insights into the relationships between different variables.

#%%
# Selecting the relevant binned and encoded variables for the Bayesian network
bayesian_data = log_binned_data[['Month_Encoded', 'crime_count_binned', 'Education_Level_3_binned', 'Migrant_Same_Address_binned', 'Employment_Worked_12M_binned', 'Economy_Not_Employed_binned', 'Ethnic_White_British_binned']]

# Encoding categorical variables (Month_Encoded is already categorical)
for col in bayesian_data.columns:
    if bayesian_data[col].dtype.name == 'category':
        bayesian_data[col] = bayesian_data[col].cat.codes

# Performing structure learning using Hill-Climb Search and BIC scoring
hc = HillClimbSearch(bayesian_data)
best_model = hc.estimate(scoring_method=K2Score(bayesian_data))
```

```

# Display the structure of the learned Bayesian network
print(best_model.edges())

#%% md
## 13. Fitting and Evaluating Bayesian Network

#%% md
### Fitting and Evaluating Bayesian Network

- **Model Fitting**: The learned Bayesian Network structure is fitted to the data using the Bayesian Estimator.
- **Model Evaluation**: The Bayesian Network is evaluated using the BIC and K2 scores to assess its performance and adequacy in representing the relationships in the data.

#%%
# Fit the Bayesian network
model = BayesianNetwork(best_model.edges())
model.fit(bayesian_data, estimator=BayesianEstimator)

# Evaluate the model using BIC score
bic_score = BicScore(bayesian_data).score(model)
print(f"BIC Score: {bic_score}")

# Evaluate the model using K2 score
k2_score = K2Score(bayesian_data).score(best_model)
print(f"K2 Score: {k2_score}")

#%% md
## 14. Inference and Analysis

#%% md
### Inference and Analysis

- **Variable Elimination**: perform inference using the Variable Elimination method to determine the most probable causes of high crime rates (categorized as 'crime_count_binned').
- **Crime Cause Analysis**: The results of the inference provide insights into the likely causes of high crime rates based on the relationships captured in the Bayesian Network.

#%%
# Perform inference to determine the most probable causes of high crime_count_binned
inference = VariableElimination(model)
crime_cause = inference.map_query(variables=['crime_count_binned'])
print(f"Most likely cause of crime: {crime_cause}")

#%% md
## 15. Visualization of Bayesian Network

#%% md
### Visualization of Bayesian Network

- **Network Structure Visualization**: use the `networkx` library to visualize the structure of the Bayesian Network. The circular layout helps in clearly displaying the relationships between variables.

#%%
# Extracting the structure of the Bayesian Network
edges = model.edges()

# Create a directed graph from the Bayesian network structure
G = nx.DiGraph()
G.add_edges_from(edges)

# Set node positions using a circular layout for better organization
pos = nx.circular_layout(G)  # Circular layout for a more organized appearance

# Draw the network
plt.figure(figsize=(15, 5))  # Increase the figure size for a more spacious view
nx.draw(
    G, pos,
    with_labels=True,
    node_size=3000,  # Even larger nodes
    node_color='lightgreen',  # Softer, calming color for nodes
    font_size=8,  # Slightly larger font for labels
    font_weight='bold',
    edge_color='gray',  # Softer edge color
    linewidths=2,  # Thicker node borders for emphasis
    arrowsize=20,  # Larger arrows for better visibility
    arrowstyle='fancy',  # Different arrow style for a unique look
)

plt.title("Bayesian Network Structure - Circular Layout", fontsize=16)  # Larger title font size
plt.show()

#%% md
## 16. Analysis of Variable Influence on Crime

#%% md
### Analysis of Variable Influence on Crime

- **Influence Plotting**: A custom function is created to plot the influence of different variables on the probability distribution of the target variable (`crime_count_binned`). This helps in understanding how changes in one variable affect the likelihood of different crime levels.

#%%
import matplotlib.pyplot as plt
from pgmpy.inference import VariableElimination

def plot_variable_influence(model, data, target_variable='crime_count_binned', variables_to_plot=None):
    """
    Plots the influence of different variables on the target variable's (crime_count_binned) probability distribution.

    :param model: The fitted Bayesian network
```

```
    model.
    :param data: The DataFrame used to fit the model.
    :param target_variable: The target variable to analyze (default is 'crime_count_binned').
    :param variables_to_plot: A list of variables for which to plot influence on the target variable.
    """
    if variables_to_plot is None:
        variables_to_plot = data.columns.tolist()
        variables_to_plot.remove(target_variable)

    # Initialize inference object
    inference = VariableElimination(model)

    for var in variables_to_plot:
        # Get unique values of the variable to plot
        unique_vals = sorted(data[var].unique())
        probabilities = {bin_label: [] for bin_label in data[target_variable].cat.categories}

        for val in unique_vals:
            # Query the model for the probability distribution of the target variable given the current variable's value
            query_result = inference.query(variables=[target_variable], evidence={var: val})
            for bin_label in probabilities:
                probabilities[bin_label].append(query_result.values[bin_label])

        # Plot the results
        plt.figure(figsize=(10, 6))
        for bin_label, prob_values in probabilities.items():
            plt.plot(unique_vals, prob_values, marker='o', label=f'{target_variable} = {bin_label}')

        plt.title(f'Influence of {var} on {target_variable}')
        plt.xlabel(var)
        plt.ylabel(f'P({target_variable})')
        plt.legend()
        plt.grid(True)
        plt.show()

# Ensure the relevant columns are converted to categorical dtype
categorical_columns = ['Month_Encoded', 'crime_count_binned', 'Education_Level_3_binned',
    'Migrant_Same_Address_binned', 'Employment_Worked_12M_binned',
    'Economy_Not_Employed_binned', 'Ethnic_White_British_binned']

# Convert each column to a categorical dtype
for col in categorical_columns:
    bayesian_data[col] = bayesian_data[col].astype('category')


plot_variable_influence(model, bayesian_data, target_variable='crime_count_binned',
                        variables_to_plot=['Economy_Not_Employed_binned',
    'Education_Level_3_binned', 'Migrant_Same_Address_binned',
    'Employment_Worked_12M_binned', 'Ethnic_White_British_binned','Month_Encoded'])
```

Listing 2. Bayesian Network Implementation

Appendix E
Feature Importance Analysis Code

```
#%% md
# 1. Importing Libraries

**Explanation**:
- **pandas** and **numpy**: For data manipulation and numerical operations.
- **matplotlib** and **seaborn**: For data visualization.
- **sklearn.preprocessing.StandardScaler**: To scale features before feeding them into the model.
- **sklearn.model_selection.train_test_split**: To split the dataset into training and testing subsets.
- **sklearn.ensemble.RandomForestRegressor**: For training a Random Forest model.
- **sklearn.linear_model.LinearRegression**: For training a Linear Regression model.
- **sklearn.metrics.mean_squared_error, r2_score**: To evaluate the models using metrics.
- **shap** and **lime.lime_tabular**: For model interpretability.

#%%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import shap
import lime.lime_tabular

#%% md
# 2. Loading and Preparing the Crime Data

**Explanation**:
- **crime_df**: Load the CSV file containing crime data.
- **'Month' to datetime**: Convert the 'Month' column to a datetime object for easier filtering.
  - **Filter the DataFrame**: Keep only the records from 2021 related to "Violence and sexual offences".
- **Modify 'LSOA name'**: Truncate the last 5 characters from 'LSOA name' to generalize it.
- **Rename column**: Change the column name to 'lower_tier_local_authorities'.
- **Aggregate Data**: Group by 'Month' and 'lower_tier_local_authorities', counting the number of crimes.

#%%
# Load the dataset containing crime data
crime_data_path = "/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/UK_Police_Street_Crime_2018-10-01_to_2021_09_31.csv"
crime_df = pd.read_csv(crime_data_path)

# Convert 'Month' to datetime and filter for 2021 and specific crime type
crime_df['Month'] = pd.to_datetime(crime_df['Month'])
filtered_crime_df = crime_df[(crime_df['Month'].dt.year == 2021) &
                             (crime_df['Crime type'] == "Violence and sexual offences")]

# Modify 'LSOA name' to remove the last 5 characters
filtered_crime_df['LSOA name'] = filtered_crime_df['LSOA name'].str[:-5]

# Rename 'LSOA name' to 'lower_tier_local_authorities'
filtered_crime_df.rename(columns={'LSOA name': 'lower_tier_local_authorities'}, inplace=True)

# Aggregate the data by 'Month' and 'lower_tier_local_authorities' and count occurrences
aggregated_crime_data = filtered_crime_df.groupby(['Month', 'lower_tier_local_authorities']).size().reset_index(name='crime_count')

#%% md
# 3. Loading and Cleaning the Additional Dataset

**Explanation**:
- **file_path**: Load another dataset containing demographic and socio-economic data.
- **columns_to_drop**: Identify columns containing redundant 'lower_tier_local_authorities' labels.
- **Drop columns**: Remove all but one 'lower_tier_local_authorities' column to avoid duplication.
- **Rename column**: Standardize the column name for consistency.

#%%
# Load datasets
ethnic_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Ethnic_Group_Data_Transformed.csv")
economy_status_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Economy_Status_Data_Transformed.csv")
employment_history_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Employment_History_Data_Transformed.csv")
education_level_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Education_Level_Data_Transformed.csv")
migrant_data = pd.read_csv("/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Migrant_Data_Transformed.csv")

# Rename the locality code columns to '
```

```
    area_code'
ethnic_data.rename(columns={'ethnic_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
economy_status_data.rename(columns={'economy_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
employment_history_data.rename(columns={'employment_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
education_level_data.rename(columns={'education_lower_tier_local_authorities_code': 'area_code'}, inplace=True)
migrant_data.rename(columns={'migrant_lower_tier_local_authorities_code': 'area_code'}, inplace=True)

# Merge all datasets on the 'area_code' column
merged_data = ethnic_data.merge(economy_status_data, on='area_code', how='outer')
merged_data = merged_data.merge(employment_history_data, on='area_code', how='outer')
merged_data = merged_data.merge(education_level_data, on='area_code', how='outer')
merged_data = merged_data.merge(migrant_data, on='area_code', how='outer')

# Save the merged dataset to a CSV file (optional)
# merged_data.to_csv('path_to/merged_dataset.csv', index=False)
merged_data
#%%
# Load the additional dataset
# file_path = '/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/Merged_Dataset.csv'
# additional_df = pd.read_csv(file_path)
additional_df = merged_data.copy()
# Identify and drop redundant 'lower_tier_local_authorities' columns
columns_to_drop = [col for col in additional_df.columns if 'lower_tier_local_authorities' in col]
columns_to_drop.remove('ethnic_lower_tier_local_authorities')  # Keep one relevant column

# Drop the identified columns
df_cleaned = additional_df.drop(columns=columns_to_drop)

# Rename the relevant column to 'lower_tier_local_authorities'
df_cleaned.rename(columns={'ethnic_lower_tier_local_authorities': 'lower_tier_local_authorities'}, inplace=True)

#%% md
# 4. Merging the Datasets

**Explanation**:
- **Merge DataFrames**: Combine the crime data and the cleaned demographic data based on 'lower_tier_local_authorities'.
- **Display Info**: Print the first few rows and the structure of the merged DataFrame to verify the merge.

#%%
# Merge the crime data with the cleaned dataset on 'lower_tier_local_authorities'
merged_data = aggregated_crime_data.merge(df_cleaned, on='lower_tier_local_authorities', how='inner')

# Display basic information and initial rows of the merged dataset
print(merged_data.head())
print(merged_data.info())

#%% md
# 5. Preprocessing and Feature Engineering

**Explanation**:
- **Drop columns**: Remove irrelevant columns like 'lower_tier_local_authorities' and 'area_code'.
- **Column mapping**: Simplify column names for easier reference.
  - **Select features**: Choose the most correlated features within each category (e.g., Migrant, Education) with the target variable, `crime_count`.
- **Scale Features**: Standardize the selected features for modeling.
  - **Convert 'Month'**: Transform the 'Month' column into a numeric format for use in models.

#%%
# Create a copy for processing
data = merged_data.copy()

# Drop columns not needed for analysis
data.drop(columns=['lower_tier_local_authorities', 'area_code'], inplace=True)

# Mapping new column names for easier reference
new_column_names = {
    'ethnic_asian_asian_british_or_asian_welsh_bangladeshi': 'Ethnic_Asian_Bangladeshi',
    'ethnic_asian_asian_british_or_asian_welsh_chinese': 'Ethnic_Asian_Chinese',
    'ethnic_asian_asian_british_or_asian_welsh_indian': 'Ethnic_Asian_Indian',
    'ethnic_asian_asian_british_or_asian_welsh_other_asian': 'Ethnic_Asian_Other',
    'ethnic_asian_asian_british_or_asian_welsh_pakistani': 'Ethnic_Asian_Pakistani',
    'ethnic_black_black_british_black_welsh_caribbean_or_africar': 'Ethnic_Black_African',
    'ethnic_black_black_british_black_welsh_caribbean_or_african': 'Ethnic_Black_Caribbean',
    'ethnic_black_black_british_black_welsh_caribbean_or_african': 'Ethnic_Black_Other',
    'ethnic_mixed_or_multiple_ethnic_groups_other_mixed_or_mult': 'Ethnic_Mixed_Other',
```

```
    '
    ethnic_mixed_or_multiple_ethnic_groups_white_a
    ': 'Ethnic_Mixed_White_Asian',
    '
    ethnic_mixed_or_multiple_ethnic_groups_white_a
    ': 'Ethnic_Mixed_White_Black_African',
    '
    ethnic_mixed_or_multiple_ethnic_groups_white_a
    ': 'Ethnic_Mixed_White_Black_Caribbean',
    '
    ethnic_other_ethnic_group_any_other_ethnic_gro
    ': 'Ethnic_Other',
    'ethnic_other_ethnic_group_arab': '
    Ethnic_Arab',
    '
    ethnic_white_english_welsh_scottish_northern_
    ': 'Ethnic_White_British',
    'ethnic_white_gypsy_or_irish_traveller': '
    Ethnic_White_Gypsy_Traveller',
    'ethnic_white_irish': 'Ethnic_White_Irish',
    'ethnic_white_other_white': '
    Ethnic_White_Other',
    'ethnic_white_roma': 'Ethnic_White_Roma',
    'economy_employed': 'Economy_Employed',
    'economy_not_employed': '
    Economy_Not_Employed',
    'employment_not_in_employment_never_worked'
    : 'Employment_Never_Worked',
    '
    employment_not_in_employment_not_worked_in_the
    ': 'Employment_Not_Worked_12M',
    '
    employment_not_in_employment_worked_in_the_las
    ': 'Employment_Worked_12M',
    '
    education_level_1_and_entry_level_qualification
    *
    _to_c_any_gcses_at_other_grades_o_levels_or_cs
    ': 'Education_Level_1',
    '
    education_level_2_qualifications_5_or_more_gcs
    *
    _to_c_or_9_to_4_o_levels_passes_cses_grade_1_
    ': 'Education_Level_2',
    '
    education_level_3_qualifications_2_or_more_a_le
    ;
    _advanced_gnvq_city_and_guilds_advanced_craft_
    ': 'Education_Level_3',
    '
    education_level_4_qualifications_or_above_degre
    ': 'Education_Level_4+',
    'education_no_qualifications': '
    Education_No_Qualifications',
    '
    education_other_apprenticeships_vocational_or_v
    _
    related_qualifications_other_qualifications_achi
    ': 'Education_Other_Qualifications',
    '
    migrant_address_one_year_ago_is_student_term
    -time_or_boarding_school_address_in_the_uk'
    : 'Migrant_Student_UK',
    '
    migrant_address_one_year_ago_is_the_same_as_th
    ': 'Migrant_Same_Address',
    'migrant_does_not_apply': 'Migrant_NA',
    '
    migrant_migrant_from_outside_the_uk_address_on
    ': 'Migrant_Outside_UK',
    '
    migrant_migrant_from_within_the_uk_address_one_year_ago_was
    ': 'Migrant_Within_UK'
}

# Rename columns according to the mapping
data.rename(columns=new_column_names, inplace=
    True)

# Select columns for further analysis
prefixes = ['Migrant', 'Education', 'Economy',
    'Ethnic', 'Employment']
selected_columns = []

for prefix in prefixes:
    prefix_columns = [col for col in data.
    columns if col.startswith(prefix)]
    corr_with_target = data[prefix_columns].
    corrwith(data['crime_count']).abs()
    best_column = corr_with_target.idxmax()
    selected_columns.append(best_column)

# No need to reload the data, proceed with the
    selected columns
selected_columns_with_month = ['Month'] +
    selected_columns + ['crime_count']
final_data = data[selected_columns_with_month]

# Scale features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(
    final_data.drop(columns=['Month', '
    crime_count']))
scaled_features_df = pd.DataFrame(
    scaled_features, columns=selected_columns)

# # Combine scaled features with 'Month' and '
    crime_count'
# final_data = pd.concat([final_data[['Month
    ']], scaled_features_df, final_data['
    crime_count']], axis=1)
#
# # Convert 'Month' to a numerical format (Unix
     timestamp)
# final_data['Month'] = pd.to_datetime(
    final_data['Month']).astype(int) // 10**9
# Extract year and month from the 'Month'
    column and treat them as categorical
# Extract month from the 'Month' column and
    treat it as categorical
final_data['Month'] = pd.to_datetime(final_data
    ['Month']).dt.month.astype('category')

# Combine scaled features with 'Month' and '
    crime_count'
final_data = pd.concat([final_data[['Month']],
    scaled_features_df, final_data['crime_count
    ']], axis=1)

#%%
final_data
#%% md
# 6. Correlation Matrix Visualization

**Explanation**:
- **Correlation Matrix**: Display the
    correlation matrix to understand the
    relationships between features and the
    target variable.
- **Heatmap**: Visual representation using
    Seaborn's heatmap for easier interpretation
```

```
    .

#%%
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the correlation matrix
corr_matrix = final_data.corr()

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a heatmap with annotations and a stylish color map
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='magma', linewidths=0.5, linecolor='white',
            cbar_kws={'shrink': 0.8, 'aspect': 20, 'pad': 0.02})

# Customize the plot with a title, better layout, and larger font sizes
plt.title('Correlation Matrix of Features and Crime Count', fontsize=18, pad=20)
plt.xticks(fontsize=12, rotation=45, ha='right')
plt.yticks(fontsize=12)

# Remove the top and right spines for a cleaner look
sns.despine()

# Adjust layout for better spacing
plt.tight_layout()

# Display the heatmap
plt.show()

#%% md
# 7. Train-Test Split and Model Training

**Explanation**:
- **Train-test split**: Split the data into 70% training and 30% testing subsets.
- **Model Training**: Train both a Random Forest model and a Linear Regression model on the training data.

#%%
# Split the data into training and testing sets
X = final_data.drop(columns=['crime_count'])
y = final_data['crime_count']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train Random Forest model
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

# Train Linear Regression model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

#%% md
# 8. Model Evaluation

**Explanation**:
- **Predict**: Generate predictions for the test set using both models.
- **Evaluate**: Calculate and print the Mean Squared Error (MSE) and R-squared ($R^2$) values for both models to assess performance.

#%%
# Predict and evaluate Random Forest
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# Predict and evaluate Linear Regression
y_pred_linear = linear_model.predict(X_test)
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)

# Print evaluation metrics
print("Random Forest Evaluation:")
print(f"R-squared (R²): {r2_rf:.4f}")
print(f"Mean Squared Error (MSE): {mse_rf:.4f}\n")

print("Linear Regression Evaluation:")
print(f"R-squared (R²): {r2_linear:.4f}")
print(f"Mean Squared Error (MSE): {mse_linear:.4f}\n")

#%% md
# 9. Visualization of Predictions vs Actual Values

**Explanation**:
- **Plot**: Create scatter plots comparing actual vs predicted values for both models to visually assess their performance.

#%%
# Visualization: Predicted vs Actual for Random Forest and Linear Regression
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_rf, alpha=0.6, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Random Forest: Actual vs Predicted')
plt.xlabel('Actual Crime Count')
plt.ylabel('Predicted Crime Count')

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_linear, alpha=0.6, color='red')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Linear Regression: Actual vs Predicted')
plt.xlabel('Actual Crime Count')
plt.ylabel('Predicted Crime Count')

plt.tight_layout()
plt.show()

#%% md
# 10. Feature Importance Visualization

**Explanation**:
- **Feature Importance**: Calculate and visualize feature importance from the Random Forest model and Linear Regression
```

```
    model coefficients.

#%%
# Random Forest Feature Importance
importances_rf = rf_model.feature_importances_
importance_df_rf = pd.DataFrame({'Feature':
    X_train.columns, 'Importance':
    importances_rf}).sort_values(by='Importance
    ', ascending=False)

# Linear Regression Coefficients
coefficients_lr = linear_model.coef_
importance_df_lr = pd.DataFrame({'Feature':
    X_train.columns, 'Importance':
    coefficients_lr}).sort_values(by='
    Importance', ascending=False)

# Visualization of Feature Importances
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.barh(importance_df_rf['Feature'],
    importance_df_rf['Importance'], color='
    skyblue')
plt.title('Random Forest Feature Importance')
plt.xlabel('Importance')

plt.subplot(1, 2, 2)
plt.barh(importance_df_lr['Feature'], abs(
    importance_df_lr['Importance']), color='
    lightcoral')
plt.title('Linear Regression Feature Importance
    ')
plt.xlabel('Absolute Coefficient Value')

plt.tight_layout()
plt.show()

#%% md
# 11. SHAP Values for Interpretation

**Explanation**:
- **SHAP Values**: Use SHAP to interpret
    feature importance and impact on
    predictions for both the Random Forest and
    Linear Regression models.

#%%
# SHAP interpretation for Random Forest
explainer_rf = shap.TreeExplainer(rf_model)
shap_values_rf = explainer_rf.shap_values(
    X_test)
shap.summary_plot(shap_values_rf, X_test,
    plot_type="bar")

# SHAP interpretation for Linear Regression
explainer_lr = shap.LinearExplainer(
    linear_model, X_train)
shap_values_lr = explainer_lr.shap_values(
    X_test)
shap.summary_plot(shap_values_lr, X_test,
    plot_type="bar")

#%%

#%% md
# 12. LIME for Interpretation

**Explanation**:
- **LIME**: Generate local interpretable model-
    agnostic explanations using LIME for both
    models on a specific instance from the test
     set.

#%%
# LIME interpretation for Random Forest
explainer_rf_lime = lime.lime_tabular.
    LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X_train.columns,
    mode='regression'
)
instance_to_explain = X_test.iloc[0]
explanation_rf = explainer_rf_lime.
    explain_instance(
    data_row=instance_to_explain.values,
    predict_fn=rf_model.predict
)
explanation_rf.show_in_notebook(show_table=True
    )

# LIME interpretation for Linear Regression
explainer_lr_lime = lime.lime_tabular.
    LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X_train.columns,
    mode='regression'
)
explanation_lr = explainer_lr_lime.
    explain_instance(
    data_row=instance_to_explain.values,
    predict_fn=linear_model.predict
)
explanation_lr.show_in_notebook(show_table=True
    )

# Save LIME interpretation to HTML
rf_html_file = 'lime_rf_explanation.html'
explanation_rf.save_to_file(rf_html_file)
# Save LIME interpretation to HTML
lr_html_file = 'lime_lr_explanation.html'
explanation_lr.save_to_file(lr_html_file)
```

Listing 3. Feature Importance Analysis

APPENDIX F
LSTM HOTSPOT MAPPING

```
#%%
import pandas as pd
import numpy as np

# Path to your dataset
file_path = '/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/uk_crime_lat_long.csv'
data = pd.read_csv(file_path)

#%%
data.head()

#%%
# Convert 'Month' to datetime format for better date handling
data['Month'] = pd.to_datetime(data['Month'])

# Sort the DataFrame by the 'Month' column
data.sort_values('Month', inplace=True)
data.head()
#%%
# Group data by Month, Latitude, and Longitude and count occurrences
monthly_data = data.groupby(['Month', 'Latitude', 'Longitude']).size().reset_index(name='Count')
monthly_data.head()
#%%
from sklearn.preprocessing import MinMaxScaler

# Initialize the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

# Fit and transform the 'Count' data to scale it
monthly_data['Normalized_Count'] = scaler.fit_transform(monthly_data[['Count']])

#%%
# Function to create input sequences for LSTM
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back):
        a = dataset[i:(i + look_back)]
        dataX.append(a)
        dataY.append(dataset[i + look_back])
    return np.array(dataX), np.array(dataY)

# Define the number of past months data to consider for predicting the next month
look_back = 3
X, y = create_dataset(monthly_data['Normalized_Count'].values, look_back)

# Reshape input to be [samples, time steps, features] for LSTM
X = np.reshape(X, (X.shape[0], look_back, 1))

#%%
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Define the LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(look_back, 1)))  # 50 LSTM units
model.add(Dense(1))  # Output layer that predicts the future value
model.compile(loss='mean_squared_error', optimizer='adam')

#%%
# Fit the model on the dataset
model.fit(X, y, epochs=10, batch_size=512, verbose=1)

#%%
# Predict future values
predictions = model.predict(X)

#%%
# Inverse transform to get predictions in the original count scale
predictions = scaler.inverse_transform(predictions)

#%%
# Add predictions to the monthly_data DataFrame
monthly_data['Predicted_Count'] = np.concatenate([np.zeros(look_back), predictions.flatten()])

#%%
import folium
from folium.plugins import HeatMap

# Create a map centered around an average location
center_lat, center_lon = monthly_data['Latitude'].mean(), monthly_data['Longitude'].mean()
map = folium.Map(location=[center_lat, center_lon], zoom_start=6)

# Add a heatmap to the map using predicted crime counts
heat_data = [[row['Latitude'], row['Longitude'], row['Predicted_Count']] for index, row in monthly_data.iterrows()]
HeatMap(heat_data).add_to(map)

# Save or display the map
map.save('/Users/arkamandol/DataspellProjects/Desertation_arka_23023023/data_files/crime_hotspots.html')

#%%
from sklearn.metrics import mean_squared_error
from math import sqrt

# Calculate RMSE
rmse = sqrt(mean_squared_error(y, predictions))
print('Root Mean Squared Error:', rmse)

#%%


#%%
```

Listing 4. LSTM Hotspot Mapping Script