Project Report

Chicago Crimes Data Analysis and Prediction

Iscte - University Institute of Lisbon
Algorithms for Big Data 2024/2025
Vaag Mkrtchyan

Summary:
This project investigates the use of machine learning to predict whether a crime in Chicago will result in an arrest based on historical police data. The study applies a complete data science workflow, including data cleaning, exploratory analysis, model development, evaluation, and real-time prediction using Apache Kafka. Three machine learning models — Logistic Regression, Support Vector Machine, and Random Forest — were tested, revealing trade-offs between precision, recall, and overall accuracy, with SVM achieving the best precision and AUC, and Logistic Regression excelling in recall. The real-time pipeline demonstrates how batch-trained models can be deployed for near real-time predictions using Kafka consumers without distributed stream processing.

Introduction

In recent years, the use of data-driven approaches in law enforcement has become increasingly essential. The ability to analyze historical crime patterns and predict future occurrences enables police departments to optimize resource allocation, enhance operational efficiency, and improve public safety.

This project focuses on the analysis of crime data from the Chicago Police Department with the primary objective of building a machine learning model capable of predicting whether a given crime will result in an arrest. The problem has been identified as a **binary classification task**, where the target variable indicates whether an arrest has been made (Arrest = 1) or not (Arrest = 0).

Beyond traditional batch data analysis, this project introduces a near real-time data processing pipeline using **Apache Kafka** and **Apache Spark**. This streaming architecture allows for ingesting crime event data as it occurs and applying the trained machine learning model in real time to generate instant predictions.

This report covers the full lifecycle of the project, including data exploration, preprocessing, model development, evaluation, and deployment of a real-time prediction pipeline. The goal is not only to build an accurate predictive model but also to showcase how such models can be operationalized in dynamic data environments.

Dataset description

Data source

The dataset has been retrieved from an open source. The data includes detailed information about each incident, such as the type of crime, the time and date of occurrence, location, whether an arrest was made, and other contextual attributes.

Dataset overview

- **Number of records:** Approximately 6.8 million crime reports
- **Time range:** From 2001 to 2024
- **Number of features:** 22 columns initially

Key features

- ***ID:*** Unique identifier for each record
- ***Case Number:*** Official case number assigned by the police
- ***Date:*** Date and time when the crime occurred
- ***Block:*** Block address of the crime occurrence
- ***IUCR:*** Illinois Uniform Crime Reporting code
- ***Primary Type:*** Primary classification of the crime (e.g., THEFT, BATTERY)
- ***Description:*** Detailed description of the crime
- ***Location Description:*** Specific location (e.g., STREET, RESIDENCE, SCHOOL)
- ***Arrest:*** Whether an arrest was made (True/False) - **target variable**
- ***Domestic:*** Indicates whether the crime was domestic-related (True/False)
- ***Beat:*** Smallest unit of police patrol area
- ***District:*** Police district
- ***Ward:*** Political district
- ***Community Area:*** Geographical community area
- ***FBI Code:*** FBI crime classification code
- ***Year:*** Year when the crime occurred
- ***Updated On:*** Date when the record was last updated
- ***Latitude / Longitude:*** Geographical coordinates of the crime

Initial Data Assessment

During the initial exploration, several data quality issues were identified:

- **Missing Values:** Some records contained missing values in fields such as Location Description, Latitude, and Longitude.
- **Duplicated Entries:** Multiple rows shared the same Case Number. This occurs when a single incident involves multiple offenses, multiple suspects, or different locations within the same case.

- **Data Formatting Issues:** Some categorical variables, such as IUCR, contained values with leading/trailing spaces, affecting their uniqueness.

Data Characteristics

- **Class Imbalance:** The target variable Arrest is significantly imbalanced. Around 77% of crimes do not result in an arrest (Arrest = 0) whereas only 23% result in an arrest (Arrest = 1)
- **Mix of Feature Types**, such as categorical, numerical, and Boolean.
- **Temporal Information:** The Date column contains precise timestamps, which are useful for extracting temporal features such as month, day, and hour to capture time-based crime patterns.

Data processing

Before any modeling could be deployed, it is essential to prepare the dataset to ensure its quality and suitability for machine learning. The raw crime data, while comprehensive, contained several inconsistencies and imperfections typical for real-world datasets.

Data cleaning

Upon inspection, it became evident that some records contained missing values, particularly in the fields related to location descriptions and geographic coordinates. Given that location data was not the focus of this specific classification task, missing coordinates were not considered critical. However, other columns required more careful attention.

A significant issue was the presence of duplicated case numbers. These duplications occur in the source data when multiple offenses, suspects, or locations are recorded under a single official case number. Since it was not feasible within the scope of this project to investigate the precise nature of each duplication, the decision was made to remove all cases where a case number appeared more than once. This ensured that each row in the dataset represented a unique crime event.

Another important part of the cleaning process was handling categorical data inconsistencies. Some categorical fields, such as the IUCR code, contained unexpected whitespace or mixed formats that affected data integrity. These issues were corrected by standardizing the string formats across relevant columns.

Feature engineering

A key transformation involved working with temporal data. The Date column, initially stored as a timestamp string, was parsed to extract granular time-based features. New variables were created to represent the year, month, day, and hour of the crime, allowing the models to capture potential temporal patterns in criminal activity. Once these features were extracted, the original timestamp column was removed, as it was no longer necessary.

Boolean variables such as Arrest and Domestic were converted into binary numerical representations (1 and 0) to be compatible with the requirements of machine learning algorithms. Categorical attributes like Primary Type, IUCR, and Location Description were encoded using Spark's StringIndexer, which converts string labels into numeric indices while preserving the categorical nature of the data.

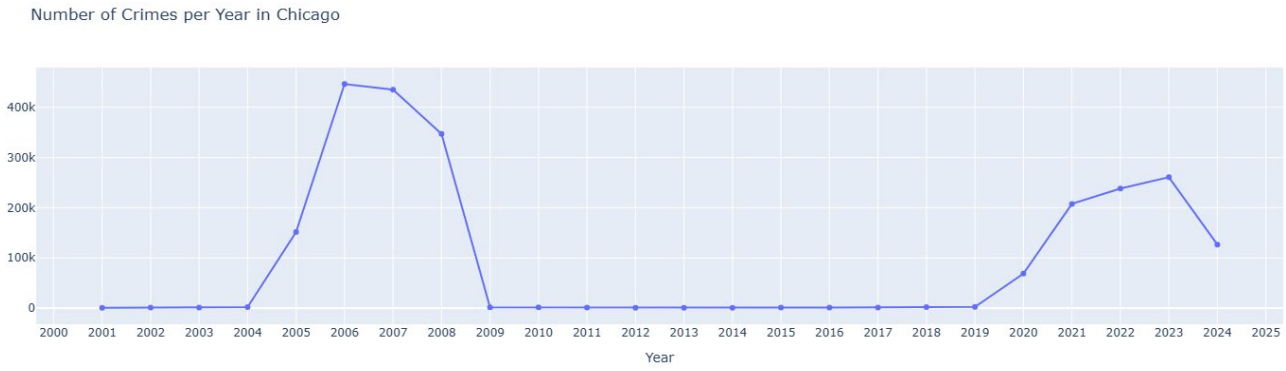
Features related to geographical and administrative divisions, such as Beat, District, Ward, and Community Area, were retained without transformation. These variables are critical for understanding the spatial distribution of crimes and potentially contribute to the model's predictive capabilities.

Columns that were irrelevant to the prediction task, such as ID, Case Number, and Updated On, were removed from the dataset. These fields either served purely administrative purposes or had no predictive value.

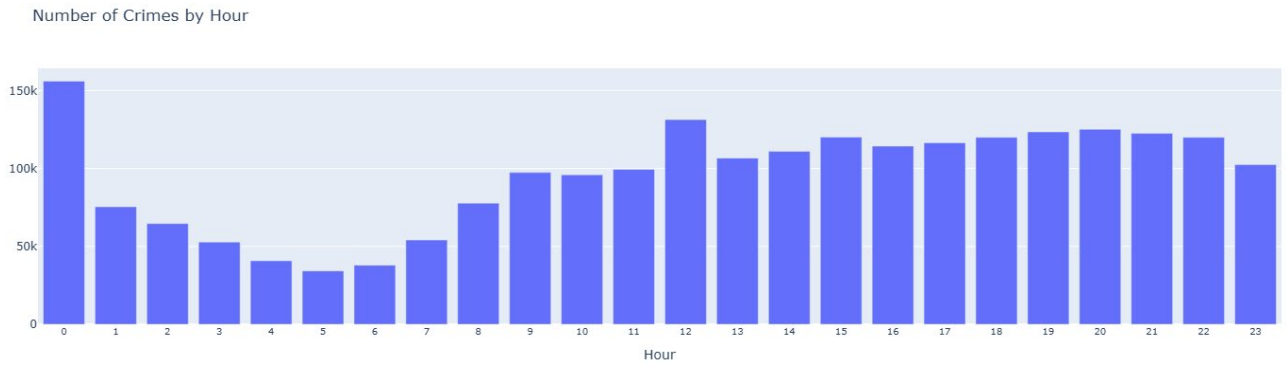
By the end of the preprocessing phase, the dataset was fully cleaned, transformed, and ready for both statistical analysis and model training. The final dataset contained a combination of numerical, categorical, and binary features, all formatted appropriately for input into machine learning algorithms.

Exploratory Data Analysis

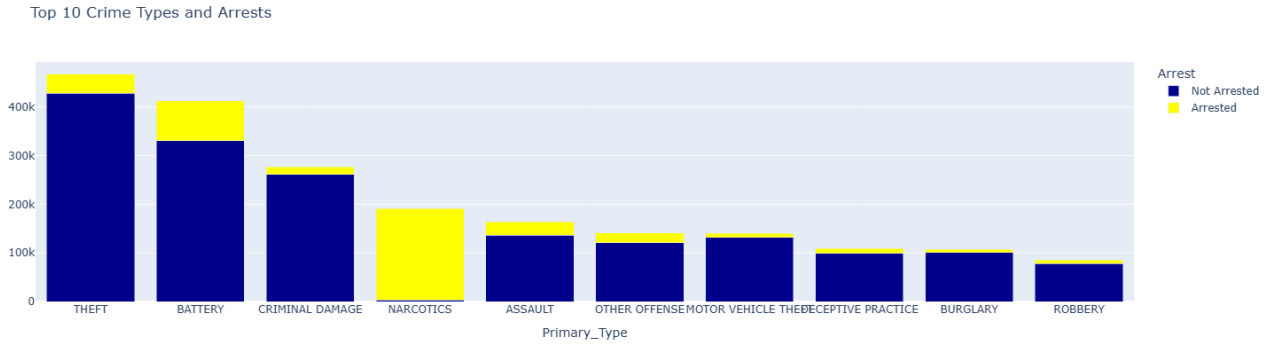
With the dataset cleaned and properly structured, the next step was to conduct an exploratory data analysis to better understand the underlying patterns, relationships, and distributions within the data. This phase is essential not only for gaining insights but also for informing the selection of features and the design of machine learning models.



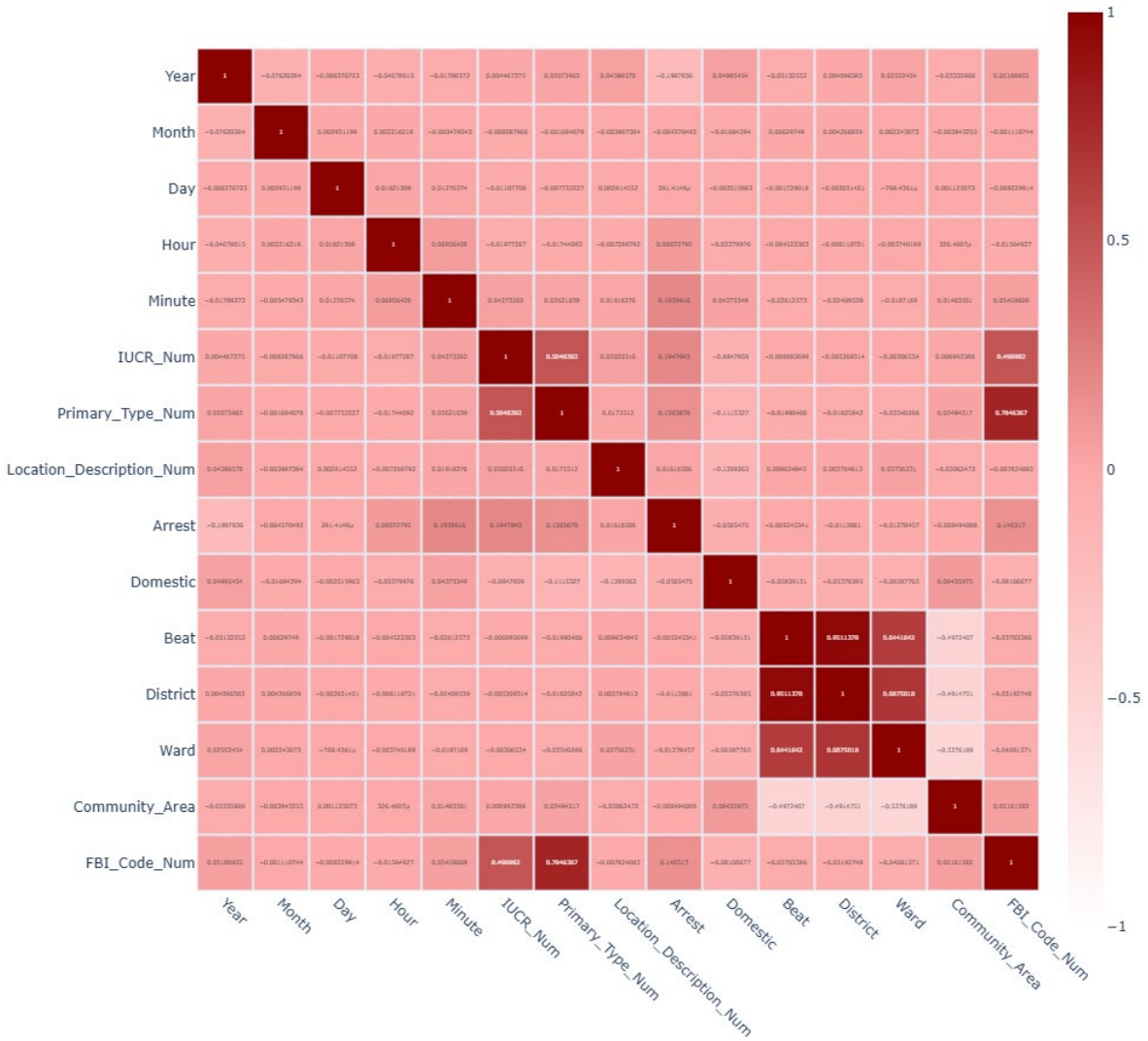
The first area of exploration focused on understanding the distribution of crimes over time. By analyzing the number of crimes per year, it became apparent that crime rates in Chicago have fluctuated over the years, with notable declines in certain periods. Such a difference may signal a large amount of missing data.



Further temporal analysis examined the distribution of crimes throughout the day. The data showed clear patterns, with certain hours experiencing higher crime rates than others. Late evening and nighttime tended to have a higher frequency of offenses, particularly those associated with thefts, assaults, and disturbances. This observation justified the inclusion of the ‘hour’ feature as a potential contributor to the predictive model.



A critical part of the analysis involved examining the distribution of crime types. The Primary Type variable revealed that the most common offenses were theft, battery, criminal damage, and narcotics-related crimes. However, the arrest rates varied significantly across crime categories. For instance, crimes like narcotics violations had a relatively high proportion of arrests, while property-related offenses, such as theft, rarely resulted in an arrest. This imbalance indicated that the nature of the crime itself is one of the strongest indicators for predicting an arrest.



Another important component of the EDA was assessing correlations between variables. A correlation matrix revealed that, surprisingly, almost all of the variables have weak direct linear

correlations between themselves and with the target variable Arrest. However, this does not exclude the possibility that these variables contribute non-linearly, especially when used with tree-based models such as Random Forest.

Overall, the exploratory analysis confirmed that several variables — particularly crime type, time of occurrence, and location — are likely to be strong predictors of whether an arrest occurs. It also highlighted the necessity of using modeling techniques capable of handling both categorical and imbalanced data, setting the stage for the subsequent machine learning phase.

Machine Learning Model Development

Following the insights gained from exploratory data analysis and assessment, the next phase of the project involved developing machine learning models to predict whether a reported crime would result in an arrest. The problem was framed as a supervised binary classification task, where the target variable indicated whether an arrest occurred (1) or not (0).

An immediate challenge faced during model development was the class imbalance within the dataset. As previously observed, only about 23% of crimes resulted in an arrest, while the vast majority did not. Ignoring this imbalance would lead to biased models that predominantly predict the majority class ('no arrest') while failing to accurately detect instances where arrests actually occur. To mitigate this issue, **class weighting** was employed. This approach increases the penalty for misclassifying the minority class (arrests), ensuring that the model pays adequate attention to both classes during training.

The dataset was split into training and testing subsets in ratio of 70/30 to evaluate the model's performance reliably. A random split ensured that both subsets maintained similar distributions of the target variable, preserving the inherent class imbalance that the models needed to learn to handle.

Three distinct machine learning algorithms were selected for this task, each offering complementary strengths. The first was **Logistic Regression**, a widely used baseline for binary classification tasks. Despite its simplicity, logistic regression provides interpretable results and can be surprisingly effective when relationships between features and the target are mostly linear.

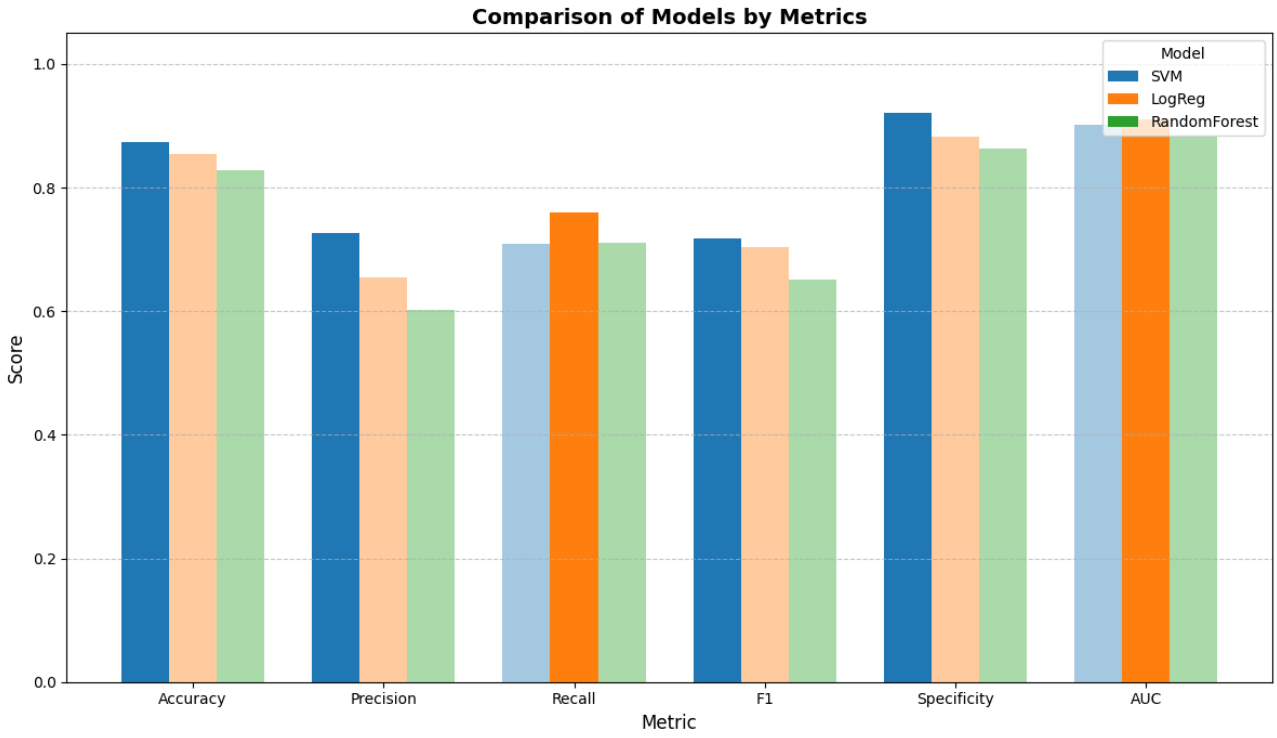
The second algorithm implemented was a **Linear Support Vector Machine (SVM)**. SVMs are known for their robustness in handling high-dimensional data and their ability to find the optimal decision boundary by maximizing the margin between classes. While SVMs are inherently binary classifiers, they are particularly sensitive to well-structured feature spaces, making them a strong candidate for this problem after appropriate data transformation.

The third and final algorithm was a **Random Forest Classifier**, an ensemble-based method that constructs multiple decision trees and aggregates their outputs. Random forests are particularly effective in handling categorical variables, capturing complex non-linear relationships, and are generally resilient to overfitting when tuned appropriately. Given the diversity of features in the crime dataset — including categorical, numerical, and temporal variables — random forests were expected to perform well.

All models were trained using the Spark MLlib framework, which allowed the training to scale efficiently across large datasets. Prior to training, categorical variables were encoded into numerical indices, and all features were assembled into feature vectors as required by the Spark machine learning pipeline.

Model Evaluation

Once the models were trained, the next step was to rigorously evaluate their performance on unseen data. Given the highly imbalanced nature of the dataset, relying solely on accuracy as a performance metric would have been misleading. A naive classifier predicting the majority class ('no arrest') for every instance could still achieve a deceptively high accuracy of around 77%, while failing entirely to detect cases where an arrest actually occurred. Therefore, the evaluation focused on more informative metrics: precision, recall, and F1-score, in addition to overall accuracy.



The evaluation revealed that none of the models universally outperformed the others across all metrics. The **Support Vector Machine (SVM)** demonstrated the highest overall accuracy, precision, specificity, and AUC, suggesting that it is particularly effective at correctly identifying non-arrest cases while maintaining a strong overall discriminative ability. This model achieved the best balance in terms of avoiding false positives.

On the other hand, **Logistic Regression** excelled in recall, indicating that it was the most effective model at capturing actual arrests. However, this came at the cost of reduced precision, meaning it also produced a higher number of false positives relative to the other models. This behavior is typical for models tuned toward maximizing recall in imbalanced datasets — they become more aggressive in predicting the minority class but sacrifice precision.

The **Random Forest Classifier**, despite its reputation for handling complex, non-linear relationships, performed somewhat conservatively in this particular task. It achieved moderate results across most metrics without leading in any specific category. Its recall was reasonably

high but lower than that of Logistic Regression, and its precision was lower than that of the SVM. This suggests that while Random Forest was able to capture some patterns in the data, it did not fully capitalize on its strength in this context, possibly due to parameter settings, limited depth, or the nature of the dataset itself.

A detailed comparison of the models showed that while all three achieved comparable overall accuracy, the random forest's superior performance in terms of both precision and recall made it the most reliable for practical applications. Particularly in contexts like law enforcement decision support, where failing to identify an arrest-worthy event could have operational consequences, this balance is crucial.

The comparison also highlights an inherent trade-off between precision and recall among the models. SVM prioritized precision and specificity, making it better suited for scenarios where minimizing false positives is crucial. Conversely, Logistic Regression prioritized recall, which may be preferable in situations where failing to identify an arrest is more costly than generating false positives.

Overall, the results indicate that the optimal model choice depends heavily on the specific goals and risk tolerance of the application context. If the priority is to ensure that as many potential arrests are identified as possible, even at the risk of some false alarms, Logistic Regression is preferable. If the goal is to minimize false positives and maintain high confidence in predicted arrests, SVM stands out as the better choice. Random Forest, while balanced, does not lead in any metric and would likely require further tuning or feature engineering to improve its competitiveness in this task.

Real-Time Data Streaming Pipeline

Beyond batch data analysis and modeling, this project includes a real-time data streaming component designed to classify crime events as they arrive. This pipeline is built entirely on Apache Kafka, without the use of Spark Streaming or any other stream processing frameworks.

The core concept behind the streaming pipeline is straightforward: a pre-trained machine learning model, developed during the batch learning phase, is loaded into a Python-based Kafka consumer. This consumer listens to a Kafka topic where crime event data is published and applies the model in real time to predict whether each crime is likely to result in an arrest.

The pipeline architecture is composed of two main components:

- **Kafka Producer:**

A Python-based producer that reads the crime dataset — either the entire dataset or a subset prepared for streaming — and sends records one by one to a Kafka topic. This simulates a real-time data feed, where each crime event is transformed into a JSON message or similar serialized format.

- **Kafka Consumer with Embedded Model:**

The consumer, also written in Python, continuously subscribes to the Kafka topic. Upon receiving each message, it deserializes the crime event, applies the same data preprocessing steps that were used during model training (such as categorical encoding and scaling if applicable), and then passes the processed input into the pre-loaded machine learning model (SVM in this case). The output is the predicted label indicating whether an arrest is expected (1) or not (0).

The prediction results are printed to the console, but in a production setup, they could be forwarded to another Kafka topic, a database, or an API endpoint for further integration.

This implementation demonstrates how a machine learning model trained in a batch setting can be effectively used for real-time inference without the need for heavy distributed stream processing frameworks. Kafka alone, combined with lightweight Python consumers, is sufficient to process incoming messages with low latency.

While simple in architecture, this approach is powerful for scenarios where the volume of streaming data is moderate, and the primary need is near real-time prediction rather than large-scale distributed computation. It also offers a scalable foundation — if needed, multiple consumers can be deployed to handle higher throughput by leveraging Kafka's partitioning capabilities.

The pipeline serves as a proof of concept for how crime prediction models could be operationalized in a streaming context, providing instant classification for each incoming crime event based on historical patterns learned during the batch training phase.

Discussion

While the project successfully demonstrates an end-to-end machine learning pipeline for predicting arrests based on crime data — including data preprocessing, model training, evaluation, and real-time inference via Kafka — there are several limitations and areas for improvement that should be acknowledged.

The project does not include hyperparameter tuning. All models were trained using default or minimally adjusted parameters. This inevitably limits their performance, particularly for models like Random Forest, which are highly sensitive to the number of trees, depth, and feature sampling parameters. Applying grid search, randomized search, or Bayesian optimization could lead to significant improvements in model accuracy, recall, and overall robustness.

Another constraint is the simplicity of the real-time pipeline architecture. While using Kafka consumers for real-time predictions is perfectly valid for prototypes or small-scale deployments, it lacks the scalability, fault tolerance, and robustness needed for production-level systems. In an enterprise or mission-critical context, integrating Spark Structured Streaming or a similar distributed stream processing framework would provide significant benefits. Spark Streaming can process data at scale, handle failures gracefully, and seamlessly integrate with both Kafka and the broader Spark ML ecosystem, allowing for distributed model inference on large volumes of streaming data.

Lastly, it is important to consider the ethical implications of predictive policing models. Relying on historical crime data — which may contain embedded biases related to over-policing in certain communities — can unintentionally perpetuate those biases if not carefully audited and managed. Any deployment of such models in real-world decision-making must be accompanied by transparency, bias mitigation strategies, and continuous monitoring to ensure fairness.

Conclusion

This project successfully demonstrates the application of machine learning techniques to predict whether a crime reported in Chicago is likely to result in an arrest. It encompasses the full data science pipeline — from data inspection and cleaning to model development, evaluation, and real-time inference using Kafka.

The results show that no single model universally outperforms the others; rather, different models present trade-offs between precision, recall, and overall accuracy. Support Vector Machines proved effective in minimizing false positives, whereas Logistic Regression achieved the highest recall, making it more suitable for scenarios where capturing all potential arrests is the priority. Random Forest, despite being a powerful ensemble method, performed moderately in this particular task without leading in any specific metric.

The real-time component of the project demonstrates how batch-trained models can be deployed for near real-time prediction using Kafka alone, without the need for complex stream processing frameworks. While effective as a proof of concept, this architecture has limitations in terms of scalability and fault tolerance, which would need to be addressed for production-grade deployments.

Several areas for improvement have been identified. The absence of advanced feature engineering, feature selection, and hyperparameter tuning limits the models' current performance. Furthermore, adopting distributed processing tools like Spark Structured Streaming could greatly enhance the scalability and resilience of the streaming pipeline.

In addition, it is crucial to acknowledge the ethical implications of using predictive models in policing. Historical crime data may reflect systemic biases, and any operational use of such models must be accompanied by rigorous fairness assessments, transparency, and bias mitigation strategies.

Overall, this project highlights both the potential and the challenges of applying machine learning to crime data analysis. With further development, including more sophisticated modeling techniques, enriched data sources, and scalable real-time architectures, the approach demonstrated here could serve as a valuable foundation for data-driven decision support systems in public safety and beyond.