FACULTY OF COMPUTATIONAL AND SOFTWARE ENGINEERING

Course: Big Data Analytics &

Business Intelligence

Course Code: SEng-9251

Group B Name	ID
1. Nancy Tesfaye	NSR/1221/14
2. Yordanos Eshetu	NSR/2772/14
3. Dibora Shibeshi	NSR/2277/14
4. Bontu Abera	NSR/139/14
5. Ashenafi Aberham	NSR/076/14
6. Gulelat Erena	NSR/917/14

Instructor: **Mr. Anteneh.T**

Arba Minch, Ethiopia

December, 2026

Contents

Executive Summary

This project presents an end-to-end Big Data and Machine Learning pipeline designed to analyze and forecast crime trends using multi-million-record datasets from Chicago and New York City. The work integrates large-scale data engineering, distributed processing, statistical modeling, and interactive visualization into a unified analytical ecosystem.

The project begins by acquiring raw crime datasets spanning nearly two decades, followed by a full ETL pipeline built using Apache Spark to clean, normalize, and aggregate the data into a structured time-series format. Advanced machine learning models—including Random Forest, Gradient Boosting, Linear Regression, ARIMA-based methods, and a finalized Random Forest pipeline—were evaluated to identify the strongest predictive approach. Results demonstrated that the Random Forest model outperformed all others, achieving an R^2 of 0.983, making it the primary model for deployment.

To support real-time usage, a Streamlit dashboard was developed, enabling users to explore trends, view model performance, and generate daily crime predictions based on temporal features. The project holds academic value by demonstrating mastery of Big Data ETL methods and practical value by illustrating how predictive analytics can support public safety and resource planning. Throughout the process, ethical guidelines were applied to ensure responsible use of crime data without profiling individuals or communities.

1. Introduction

Modern cities generate an enormous volume of crime-related information every single day. Police departments, emergency call centers, city surveillance systems, and public reporting portals continuously record incidents, arrests, complaints, and location-based activities. As these data streams accumulate over years, they form massive historical crime datasets that contain valuable patterns about how crime evolves across time, space, and socio-environmental contexts. However, traditional policing strategies—largely dependent on human judgment, manual reports, and reactive patrols—are not equipped to extract insights from such large-scale, high-dimensional data.

Predictive Policing/Crime Analytics focuses on leveraging Big Data technologies to transform these large datasets into meaningful intelligence. This project specifically examines two of the most detailed and widely used public crime datasets in the world:

- **Chicago Crime Dataset**, which contains over 8 million incident records collected since 2001.
- **NYC Crime Dataset**, extracted from the NYPD Complaint Data Historic dataset, containing several million reports dating back to 2006.

Both datasets are longitudinal, high-volume, and geographically rich, which makes them ideal for applying Big Data processing frameworks such as **Apache Hadoop** and **Apache Spark**. Because these datasets exceed the scale of typical classroom CSV files, they allow us to demonstrate true large-scale data engineering, advanced analytics, and machine learning.

This project aims to design and implement a **complete data analytics pipeline** that can process raw crime data, engineer relevant features, detect temporal trends, and ultimately generate forecasting models. By applying modern time-series techniques such as SARIMA and Prophet, the system will be able to predict crime evolution on a daily, weekly, or monthly basis.

The focus on Chicago and NYC is intentional: both cities have dense populations, diverse neighborhoods, and strong public datasets. They also differ in policing strategies, demographics, and urban design, which creates opportunities to compare patterns across cities.

With the rise of **data-driven governance**, predictive policing systems are becoming essential for resource planning, patrol scheduling, hotspot detection, and policy evaluation. However, building such systems requires more than simply loading a CSV file. It requires:

- **Big Data ingestion techniques**
- **Distributed computation using Spark**

- **Data cleaning and preprocessing at scale**
- **Aggregation of millions of records by date**
- **Statistical stationarity testing**
- **Parameter tuning for forecasting models**
- **Visualization and interpretation of results**

This project therefore simulates a professional end-to-end crime analytics workflow as would be implemented in real police departments or city analytics teams.

Ultimately, the purpose of this research is not only to forecast crime but also to demonstrate how Big Data technologies can enhance modern policing. By integrating historical datasets with scalable ETL pipelines and predictive modeling, the system offers a data-driven approach that can support decision-makers, improve public safety strategies, and contribute to the growing field of urban analytics.

2. Problem Statement

Major cities such as **Chicago** and **New York City** generate extremely large volumes of crime records every year. These datasets are not small or neatly structured; instead, they consist of millions of semi-structured entries that must be cleaned, standardized, and transformed before any meaningful analysis can be performed. Because of their size, complexity, and irregular formatting, traditional tools (Excel, small Python scripts, basic databases) cannot process them efficiently.

To analyze crime patterns over long time periods and support *Predictive Policing/Crime Analytics*, a scalable Big Data approach is required.

The core problem addressed in this project is therefore twofold:

1. Data Engineering Problem

The raw Chicago and NYC crime datasets contain:

- inconsistent date formats
- missing or malformed entries
- noise and redundant fields
- millions of rows requiring distributed processing

Before the data can be used for prediction, it must be extracted, cleaned, and aggregated using Big Data technologies such as **Hadoop** and **Apache Spark**. Without this preprocessing, statistical insights and forecasting models cannot be generated.

2. Predictive Modeling Problem

Once cleaned and structured, the data must be modeled to identify:

- temporal crime patterns
- seasonal fluctuations
- long-term trends
- daily/weekly/monthly crime cycles

This requires advanced time-series forecasting methods capable of handling large datasets, irregular patterns, and seasonality. The project aims to build models that can forecast future crime levels, supporting data-driven decision-making in policing.

The project seeks to solve the challenge of transforming massive, real-world crime datasets from Chicago and NYC into structured analytical formats and developing predictive models that reveal how crime evolves over time. This requires both **Big Data processing** and **statistical forecasting**, forming a complete end-to-end crime analytics pipeline.

3. Project Objectives

Core Objective

To develop a scalable **Big Data–powered predictive crime analytics system** that processes, analyzes, and forecasts crime patterns using the **Chicago Crime** and **NYC Crime** datasets from the *Predictive Policing/Crime Analytics (Group B)* project scope.

Specific Objectives

1. Data Acquisition

- Gather multi-year crime records from official open-data portals for Chicago and New York City, focusing on datasets large enough to demonstrate true Big Data processing requirements.

2. Distributed Data Storage

- Store the raw datasets within a distributed file system (e.g., HDFS storage) to support large-scale processing and parallel computation.

3. Big Data ETL Pipeline

- Develop an Apache Spark-based ETL workflow that:
  - parses and standardizes inconsistent date/time fields

- handles missing or malformed entries
- extracts relevant attributes (location, offense type, timestamps)
- aggregates the data into structured analytical formats

4. Construction of Time-Series Datasets

- Transform the cleaned data into daily, weekly, and monthly crime counts for both cities, enabling temporal analysis and forecasting.

5. Exploratory Crime Analysis

- Perform descriptive analytics to reveal:
  - long-term crime trends
  - seasonal variations
  - peak activity periods
  - anomalies and irregular patterns

6. Predictive Modeling

- Apply advanced time-series forecasting models, including SARIMA or similar statistical methods, to predict future crime levels for each city.

7. Cross-City Comparative Analysis

- Compare crime patterns between Chicago and NYC to identify:
  - shared seasonal behaviors
  - differences in long-term trends
  - city-specific crime dynamics

8. Reporting and Visualization

- Present findings through:
  - interactive dashboards
  - charts and time-series visualizations
  - written analytical reports
  - model performance summaries

4. Data Description

4.1 Chicago Crime Dataset

The **Chicago Crime dataset** is one of the largest publicly available municipal crime datasets in the United States. It contains:

- **More than 7 million crime incident records** spanning from **2001 to the present**
- Semi-structured entries with inconsistencies typical of long-term public datasets
- Key fields such as:
    - **Date and time** of the offense
    - **Primary crime category** (e.g., theft, assault, narcotics)
    - **Arrest indicator** showing whether an arrest was made
    - **Geographical coordinates** (latitude, longitude)
    - **Police district and community area identifiers**

The dataset's size, continuous updates, and diverse field formats make it a strong candidate for Big Data techniques, particularly distributed processing and large-scale time-series construction.

4.2 NYC Crime Dataset

The **NYC (New York City) Complaint-Level Crime dataset** contains:

- **Several million crime complaint reports** from **2006 to the present**
- Records that vary by year in format and completeness, which requires careful cleaning and standardization
- Key fields including:
    - **Incident date and reporting time**
    - **Offense classification** (felony, misdemeanor, violation)
    - **Borough information** (e.g., Manhattan, Bronx, Queens)
    - **Precinct and jurisdiction details**

Like the Chicago dataset, its size, longitudinal structure, and variability require scalable Big Data pipelines for processing and analysis.

4.3 Big Data Qualification

Both datasets clearly meet Big Data criteria due to:

- **Volume:** Millions of rows spanning decades
- **Variety:** Mixed data types, inconsistent formatting, and evolving schemas
- **Velocity:** Frequently updated and continuously growing datasets
- **Complexity:** Requires distributed systems to process, transform, and model efficiently

This project treats these datasets not as simple CSV tables but as **massive, evolving data streams** requiring Hadoop/Spark-level computation to extract meaningful trends and predictive insights.

5. Project Scope

5.1 In-Scope Activities

This project focuses on building a complete Big Data analytical pipeline for crime forecasting using Chicago and NYC crime datasets. The scope includes the following components:

1. Big Data Engineering with Apache Spark

Implementation of distributed data processing workflows capable of handling multi-million-record datasets. Spark will be used to clean, transform, aggregate, and prepare data for analytics at scale.

2. Data Storage and Management

Raw and processed datasets will be stored in a Hadoop-based environment. This ensures scalability, accessibility, and the ability to reprocess or extend the dataset as needed.

3. ETL (Extract–Transform–Load) Pipeline Development

A full ETL sequence will be designed to:

- Ingest raw CSV or parquet files
- Standardize formats across both cities
- Handle missing, duplicated, or inconsistent records
- Generate unified, analysis-ready time-series datasets

This pipeline will be automated so new data can be incorporated without redesign.

4. Time-Series Modeling and Forecasting

The project will construct analytical models for daily, weekly, and monthly crime counts. Techniques may include:

- Seasonal ARIMA (SARIMA)
- Exponential Smoothing
- Trend and seasonality decomposition

These models will be used to identify long-term tendencies, cyclical patterns, and future projections.

5. Visualization and Reporting

Interactive and static visualizations will be developed to present:

- Historical crime patterns
- Seasonal effects
- Cross-city comparisons
- Forecasted future crime trends

A final report and presentation will summarize findings, limitations, and real-world implications.

5.2 Out-of-Scope Activities

To maintain ethical and academic boundaries, the following activities are **explicitly excluded** from the project:

1. Identification of Individual Offenders

No attempt will be made to track, classify, or identify specific persons involved in criminal incidents.

2. Predictive Profiling

The project will not develop models that infer characteristics about individuals, groups, or communities beyond aggregated crime patterns.

3. Geospatial Policing Recommendations

No operational policing decisions—such as officer deployment, hotspot targeting, or patrol routing—will be generated unless considered as an optional extension for general academic analysis only.

6. Methodology (Step-by-Step)

6.1 Week 1 – Project Proposal & Conceptual Foundation

Week 1 focused on developing the conceptual and structural foundation of the entire Crime Prediction Project. Before any coding or technical implementation began, it was essential to establish a clear understanding of the project's goals, relevance, scope, architecture, and expected deliverables. This week laid the groundwork for the Big Data system that would be built in the following weeks.

1. Project Proposal Development

The week began with a comprehensive proposal that justified the significance of crime prediction in modern urban environments. The proposal outlined:

- The importance of analyzing crime patterns using data science
- The challenges of dealing with multi-million–row datasets
- The need for scalable computing (Hadoop + Spark)
- The benefits of predictive modeling for future crime estimation
- Ethical considerations guiding the entire project

The proposal clearly stated that the project would integrate **Big Data Engineering**, **Exploratory Data Analysis**, **Machine Learning Modeling**, and **Dashboard Deployment** into a single end-to-end system.

2. Research & Requirement Analysis

A requirements study was conducted to identify:

- Available crime datasets (Chicago & NYC)
- Data formats, volumes, and schema inconsistencies
- Tools necessary for handling distributed processing
- Required Python libraries for ML development
- Expected outputs such as processed datasets, visual reports, and ML predictions

This stage ensured that the project was feasible within the timeline and the computational tools available.

3. Big Data Architecture Planning

A preliminary architecture was drafted to illustrate how data would flow through the system:

1. **Raw Layer** — Original CSVs in HDFS
2. **Processing Layer** — Spark ETL cleaning scripts
3. **Aggregation Layer** — Daily crime counts per city
4. **Feature Engineering Stage** — Creation of temporal variables
5. **Modeling Stage** — Training and evaluation
6. **Deployment Stage** — Streamlit dashboard

This architecture ensured the system remained modular, scalable, and easy to extend in the future.

6.2 Week 2 – ETL Pipeline & Data Engineering

Week 2 represented the start of hands-on technical development. The main objective of this week was to build a reliable **Extract–Transform–Load (ETL)** pipeline capable of ingesting, cleaning, and preparing large-scale crime datasets from Chicago and New York.

The tasks were divided into five major components:

6.2.1 Environment Setup

A fully functional Big Data environment was prepared to support large-scale distributed processing.

1. Java Installation & Configuration

Apache Spark and Hadoop require Java (JDK).

- Installed **OpenJDK/Oracle JDK 17**
- Configured system variables:
- JAVA_HOME = C:\Program Files\Java\jdk-17
- PATH = %JAVA_HOME%\bin;%PATH%
- Verified installation using:
- java -version

2. Hadoop Installation (Single-Node HDFS)

To simulate a distributed storage system:

- Downloaded Hadoop binary compatible with Spark
- Configured essential files:
    - core-site.xml … defines Namenode host
    - hdfs-site.xml … sets replication factor and data directories
- Initialized Namenode and launched HDFS services:
    - cd C:\hadoop\sbin
    - start-dfs.cmd
- Confirmed HDFS availability using:
    - hdfs dfs -ls /

3. Spark Installation

Spark was installed to serve as the main engine for ETL operations.

- Extracted Spark 4.0.1
- Set the SPARK_HOME and updated system PATH

- Tested Spark installation:
    - spark-submit --version

4. Python & PySpark Setup

Python is required for writing ETL scripts.

- Installed Python 3.10
- Installed PySpark using pip:
    - pip install pyspark
- Verified PySpark functionality:
    - from pyspark.sql import SparkSession
    - spark = SparkSession.builder.appName("test").getOrCreate()

This environment setup ensured the system was ready for dataset ingestion and large-scale cleaning.

6.2.2 Data Ingestion into HDFS

With the environment ready, Week 2 continued with moving raw datasets into HDFS.

1. Creating necessary directories

HDFS directories were created to organize the data pipeline:

- hdfs dfs -mkdir -p /user/nan25/raw
- hdfs dfs -mkdir -p /user/nan25/processed
- hdfs dfs -mkdir -p /user/nan25/data

2. Uploading raw crime datasets

Two major datasets were uploaded:

- Chi_Crimes_2001_to_Present.csv
- nypd_raw_data.csv

Commands used:

- hdfs dfs -put Chi_Crimes_2001_to_Present.csv /user/nan25/raw/
- hdfs dfs -put nypd_raw_data.csv /user/nan25/raw/

3. Verification

Contents confirmed using:

- hdfs dfs -ls /user/nan25/raw

Outcome:
Raw datasets were successfully stored in HDFS, enabling distributed cleaning using PySpark.

6.2.3 Chicago ETL Pipeline

Chicago ETL was the first city-specific cleaning process.

ETL Script Location:

- etl_code/nan25_chicago/cleaning_chicago.py

Steps Executed

1. Spark Session Creation

A SparkSession allows distributed operations on large CSV files.

2. Using Hadoop Java FileSystem API (Py4J)

Used to handle Spark's part-file outputs and rename them into clean CSV files.

3. Selecting essential columns

Extracted fields:

- **ID**
- **Date**
- **Primary Type** (crime category)
- **Description**

These were standardized into lowercase and cleaned formats.

4. Data cleaning & preprocessing

- Removed records with missing essential values
- Standardized datetime formats (Spark timestamp parsing)
- Normalized crime type categories for consistency

5. Saving cleaned dataset into HDFS

Output file:

- /user/nan25/processed/chicago_cleaned.csv

Commands executed

- spark-submit C:\crime_project\etl_code\nan25_chicago\cleaning_chicago.py
- hdfs dfs -ls /user/nan25/processed

Result

Chicago dataset successfully transformed into a clean, structured CSV ready for aggregation and EDA.

6.2.4 NYC ETL Pipeline

NYC dataset required additional transformations because its structure differs significantly from Chicago's.

ETL Script Location:

- etl_code/nan25_nyc/cleaning_nyc.py

Steps Executed

1. Reading the raw NYC dataset

Headers analyzed to identify differences and inconsistencies.

2. Mapping columns to a unified schema

NYC headers (e.g., *ARREST_KEY*, *ARREST_DATE*) were converted to a consistent schema with Chicago.

3. Standardizing offense descriptions

Converted crime descriptions to uppercase to avoid duplicates caused by text variations.

4. Parsing datetime fields

Converted arrest/incident dates into a uniform timestamp format.

5. Saving cleaned dataset

Stored cleaned file:

- /user/nan25/processed/nyc_cleaned.csv

Commands executed

- spark-submit C:\crime_project\etl_code\nan25_nyc\cleaning_nyc.py
- hdfs dfs -ls /user/nan25/processed

Notes

- Encountered a temporary SafeModeException (HDFS still initializing).
- Issue resolved by restarting Hadoop services and re-running the script.

6.2.5 Daily Aggregation

Daily aggregation created the analytical foundation for predictive modeling.

Objective

Generate **daily crime counts**, per crime type and per city.

Chicago Aggregates produced

- battery_occurrence_per_day.csv
- theft_occurrence_per_day.csv
- crime_occurrence_per_day.csv

NYC Aggregates produced

- nyc_assault_occurrence_per_day.csv
- nyc_larceny_occurrence_per_day.csv
- nyc_crime_occurrence_per_day.csv

Methodology

1. Filtering specific crime types

Crime types were normalized (uppercase) to avoid inconsistent matches.

2. Grouping by date

Spark's groupBy() function was used:

- df.groupBy("date").count()

3. Coalescing to a single output file

Spark outputs multiple part-files by default.
Used:

- df.coalesce(1)

4. Renaming part-files via FileSystem API

Py4J Hadoop API used to rename part-0000 files into readable CSVs.

5. Saving aggregated data into HDFS

Files stored in:

- /user/nan25/data/

Verification

- hdfs dfs -ls /user/nan25/data

Example output paths

- /user/nan25/data/battery_occurrence_per_day.csv
- /user/nan25/data/nyc_assault_occurrence_per_day.csv

6.3 Week 3 – Exploratory Data Analysis (EDA) & Early BI Visuals

Week 3 focused on examining the cleaned datasets produced during Week 2 and extracting meaningful insights through Exploratory Data Analysis (EDA). This stage is essential because statistical summaries and visual patterns guide the selection of features, models, and evaluation strategies in later phases. The tasks in Week 3 involved data merging, in-depth analysis of Chicago and NYC datasets, cross-city comparisons, and development of initial BI (Business Intelligence) visuals.

6.3.1 Data Merging Across Cities

Objective

The primary goal was to create a unified dataset containing crime incidents from both Chicago and New York City. A merged dataset supports:

- Cross-city comparisons
- Combined temporal analyses
- Harmonized crime categories
- Broader generalization for future predictive models

Implementation Plan

A dedicated merge script was created at:
etl_code/nan25_merge/merge_city_crimes.py

The merging process involved the following structured steps:

Step 1: Load Cleaned Datasets from HDFS

The Chicago and NYC cleaned CSV files were retrieved from:

/user/nan25/processed/

Both files had already undergone cleaning in Week 2, ensuring consistent timestamp formats and selected attributes.

Step 2: Standardize Column Names

To enable seamless concatenation, all columns were standardized using the following unified schema:

Column	Description
id	Unique crime ID / arrest key
date	Incident date and time
type	Primary offense category
desc	Detailed description
city	City label (“Chicago” or “NYC”)

Standardization ensures both cities follow the same structure.

Step 3: Add "City" Column

A new column was appended to each dataset:

- "Chicago" for Chicago records
- "NYC" for New York records

This enables grouped visualizations and filtering.

Step 4: Concatenate DataFrames

Spark's union function was used to vertically combine both datasets into a single integrated DataFrame.

Step 5: Save Final Merged Dataset

Output file:

- /user/nan25/processed/merged_city_crimes.csv

Current Status

- Chicago cleaned dataset ready
- NYC cleaned dataset ready
- Merge script written
- Pending: final execution

Once executed, the merged dataset serves as the foundation for cross-city EDA and future modeling.

6.3.2 EDA on Single-City Datasets

EDA was performed separately for Chicago and NYC data using Jupyter notebooks stored under:
ana_code/

Notebooks Include:

- **analysis_chicago_all.ipynb** – Comprehensive Chicago EDA
- **analysis_chicago_type1.ipynb** – Category-focused Chicago analysis
- **analysis_chicago_type2.ipynb** – Category comparative analysis

- **analysis_nyc_all.ipynb** – Full NYC EDA
- **analysis_nyc_type1/type2.ipynb** – Category-specific NYC analysis

Key EDA Tasks

1. Dataset Loading

Cleaned CSVs were loaded either:

- directly from local copy (downloaded from HDFS), or
- via PySpark read functions if using distributed analysis.

2. Structural Inspection

Basic data exploration included:

- df.info() → data types, memory usage
- df.describe() → statistical summaries
- df.isnull().sum() → missing value assessment
- Duplicate detection

This ensured readiness for deeper analysis.

3. Time-Series Analysis

Crime is inherently temporal. Methods used:

- **Daily crime counts**
- **7-day rolling averages** to smooth short-term fluctuations
- **Monthly aggregates** for long-term seasonality
- **Yearly trends** comparing crime over multiple years

These revealed recurring patterns such as weekend spikes, seasonal increases, or long-term declines.

4. Crime Category Insights

Generated category-level insights by calculating:

- Top 10 crimes in each city
- Percentage distribution between categories
- Frequency patterns of specific crimes such as assault, theft, battery, and larceny

Whenever possible, frequencies were normalized per population to obtain *per-capita* comparisons.

5. Visual Export

All generated charts were saved to:

- output/screenshots/

This allowed integrating visuals and later into the final report.

6.3.3 EDA on Merged Datasets

Once the unified dataset was ready, deeper cross-city exploration was performed using the notebooks:

- **analysis_merged_all.ipynb**
- **analysis_merged_type1.ipynb**
- **analysis_merged_type2.ipynb**

Cross-City EDA Tasks

1. Cross-City Comparative Trends

The merged dataset allowed comparing how different crimes behave across cities:

- Assault trends: NYC consistently higher
- Theft trends: Chicago showing more variation
- Seasonal differences between cities

This revealed structural differences in crime reporting.

2. Combined Time-Series Aggregations

Aggregations were performed on both cities fused together:

- Combined daily totals
- Combined monthly heatmaps
- Side-by-side trendlines showing both cities

These visualizations highlighted how crime patterns shift across urban environments.

3. Visualization of Cross-City Crime Composition

Charts created included:

- Stacked bar charts showing proportion of crime types
- Side-by-side category comparisons
- Multi-line plots comparing aligned crime trends (e.g., theft Chicago vs. theft NYC)

This enabled understanding of which city exhibits higher risk for specific crime categories.

6.3.4 Early Visuals & Dashboard v1

The visualizations generated in Week 3 formed the starting point for the BI dashboard that would later be deployed in Streamlit.

Visuals Produced

1. Time-Series Line Charts

Charts showed:

- Daily crime counts
- Rolling averages
- Seasonal patterns

Useful for identifying spikes, anomalies, and long-term trends.

2. Crime Category Bar Charts

Displayed:

- Most frequent offense types
- Percentage distribution
- Comparison across months or years

These highlight category dominance (e.g., theft and larceny).

3. Monthly Heatmaps

Heatmaps visualized crime intensity across:

- Months (rows)
- Years (columns)

This makes seasonal variations immediately visible.

Dashboard Version 1 Options

The project considered three dashboard development options:

1. **Plotly Dash** – Python-based, more customizable
2. **Streamlit** – later used for ML prediction

6.3.5 Files, Paths & Command Reference

HDFS Raw Input Files:

- /user/nan25/raw/Chi_Crimes_2001_to_Present.csv
- /user/nan25/raw/nypd_raw_data.csv

Cleaned Outputs:

- /user/nan25/processed/chicago_cleaned.csv
- /user/nan25/processed/nyc_cleaned.csv

Daily Aggregations:

- /user/nan25/data/battery_occurrence_per_day.csv
- /user/nan25/data/theft_occurrence_per_day.csv
- /user/nan25/data/nyc_assault_occurrence_per_day.csv
- /user/nan25/data/nyc_larceny_occurrence_per_day.csv
- /user/nan25/data/nyc_crime_occurrence_per_day.csv

Planned Merge Output:

- /user/nan25/processed/merged_city_crimes.csv

Spark Execution Commands:

- spark-submit C:\crime_project\etl_code\nan25_chicago\cleaning_chicago.py
- spark-submit C:\crime_project\etl_code\nan25_nyc\cleaning_nyc.py
- spark-submit C:\crime_project\etl_code\nan25_merge\merge_city_crimes.py

HDFS Folder Checks:

- hdfs dfs -ls /user/nan25/processed
- hdfs dfs -ls /user/nan25/data

6.4 Week 4 – Feature Engineering & Machine Learning Model Training

Week 4 marked the transition from data engineering and exploratory analysis into predictive modeling. The primary goal this week was to transform aggregated crime data into machine-learning-ready features and train several baseline models that could later be evaluated and refined in Week 5. This stage involved designing reproducible scripts,

organizing folders for modularity, creating temporal features, validating data quality, training models, and saving serialized model artifacts.

6.4.1 Folder Structure Overview

The project was organized into a modular and scalable directory structure to ensure clarity, maintainability, and reproducibility. Below is the structured layout:

```
crime_project/
│
├── ml_code/
│   ├── features/
│   │   └── build_features.py
│   ├── training/
│   │   └── train_models.py
│   ├── models/
│   │   ├── RandomForest.pkl
│   │   ├── GradientBoosting.pkl
│   │   ├── LinearRegression.pkl
│   │   └── final_model.pkl
│   └── streamlit_app/
│       └── app.py
│
└── data/
    └── daily_features.csv
```

Explanation of Key Project Components

Component	Description
ml_code/features/build_features.py	Script responsible for transforming the raw aggregated dataset (daily_features.csv) into a machine-learning-ready dataset by generating temporal features such as day, month, year, day_of_week, and is_weekend.
ml_code/training/train_models.py	Trains three foundational regression models—Random Forest, Gradient Boosting, and Linear Regression—and stores them in serialized .pkl files.
ml_code/models/	Directory where all trained machine learning models are saved, including the baseline models and the final optimized model (final_model.pkl).
ml_code/streamlit_app/app.py	A fully functional Streamlit application used for interactive visualization and real-time prediction using the trained model.
data/daily_features.csv	Core input dataset containing the aggregated daily crime counts along with their respective dates.

This structure ensures clean separation of responsibilities—feature engineering, training, modeling, and deployment—making the project easy to scale and maintain.

6.4.2 Methodology

Week 4 focused on two main activities:

1. **Feature Engineering (creating model inputs)**
2. **Model Training (building baseline ML models)**

These steps are essential for producing a predictive model that is both accurate and generalizable.

6.4.2.1 Feature Engineering

Objective

To convert daily crime count data into a structured format with usable features that capture time-based patterns influencing crime trends.

Data Source

The input dataset was:

- data/daily_features.csv

This dataset contained two columns:

- date — the day-level timestamp
- count — number of crimes recorded on that day

Steps in Feature Engineering

1. Extract Temporal Features

Crime patterns are heavily dependent on temporal cycles such as weekdays, weekends, and seasonal variations. Week 4 introduced the following features:

Feature Name	Description
day	Day of month (1–31)
month	Month of year (1–12)

year	Full year extracted from date
day_of_week	Day index (0 = Monday, 6 = Sunday)
is_weekend	Binary flag: 1 for Saturday/Sunday, else 0

These features help models learn patterns such as:

- Higher crime on weekends
- Seasonal peaks (e.g., summer months)
- Year-over-year shifts

2. Prepare Target Variable

The target variable is:

- count

This is the numeric value the model attempts to predict.

3. Data Validation

Before saving the processed dataset, the following checks were performed:

- Missing values in date or count
- Incorrect datetime formats
- Columns containing unexpected symbols or strings
- Consistency of weekday and weekend labeling
- Data-type enforcement (integers for features, datetime for dates)

Data validation ensured that models trained on the dataset would not encounter errors or performance issues.

4. Feature Engineering Script

File: ml_code/features/build_features.py

Python Implementation:

```
import pandas as pd
from pyspark.sql import SparkSession
from pyspark.sql.functions import to_date, col
import os

# ------------------------------
```

```
# 1. Spark session
# ------------------------------
spark = SparkSession.builder.appName("Build_Features").getOrCreate()

# ------------------------------
# 2. Load merged dataset from HDFS
# ------------------------------
MERGED_HDFS_PATH                                                    =
"hdfs://localhost:9000/user/nan25/processed/merged_city_crimes.csv"

df = spark.read.option("header", True).csv(MERGED_HDFS_PATH)

# ------------------------------
# 3. Basic Cleaning
# ------------------------------
df = df.withColumn("date", to_date(col("date_parsed")))
df = df.dropna(subset=["date"])
```

Outcome

A new machine-learning-ready dataset:

- daily_features_processed.csv

This dataset contains engineered temporal features suitable for model training in the next phase.

6.4.2.2 Machine Learning Model Training

After generating the processed feature dataset, the next step was to develop a suite of baseline predictive models.

Objective

Train multiple regression algorithms to determine which model best captures the relationship between temporal features and daily crime counts.

Algorithms Used

1. **Random Forest Regressor**
    - Handles nonlinear relationships
    - Robust to overfitting
    - Performs well on time-related tabular data
2. **Gradient Boosting Regressor**

- Builds sequential models that correct errors of previous ones
- Often highly accurate

3. **Linear Regression**
   - Simple baseline model
   - Helps evaluate dataset linearity
   - Useful for interpretability

Training Steps

1. Load Processed Dataset

```
df = pd.read_csv("data/daily_features_processed.csv")
X = df[["day", "month", "year", "day_of_week", "is_weekend"]]
y = df["count"]
```

2. Initialize Models

Three models were defined in a dictionary for clean iteration:

```
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
import pickle
```

3. Train & Save Models

```
models = {
    "Random Forest": RandomForestRegressor(random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(random_state=42),
    "Linear Regression": LinearRegression()
}

for name, model in models.items():
    model.fit(X, y)
    with open(f"ml_code/models/{name.replace(' ', '')}.pkl", "wb") as f:
        pickle.dump(model, f)
```

Each model was trained on the same input features and then serialized as a .pkl file for later evaluation.

Outcome

By the end of Week 4, the following baseline models were successfully trained and stored:

- RandomForest.pkl
- GradientBoosting.pkl
- LinearRegression.pkl

6.5 Week 5 – Model Evaluation, Pipeline Finalization & Dashboard Deployment

Week 5 marked the final phase of the Crime Prediction Project, transitioning from model training (completed in Week 4) into rigorous model evaluation, selection of the best-performing algorithm, and full deployment of the Machine Learning pipeline. This phase also included development of an interactive Streamlit dashboard to provide real-time prediction capabilities.

6.5.1 Model Evaluation & Comparison

Objective

The main objective of the evaluation stage was to assess the predictive performance of the three trained models—Random Forest, Gradient Boosting, and Linear Regression—using statistical metrics. These evaluations ensure that the chosen model generalizes well and produces reliable crime predictions.

1. Evaluation Metrics

To ensure a fair and consistent comparison, three industry-standard regression metrics were used:

• R^2 Score (Coefficient of Determination)

Measures how much variance in crime counts is explained by the model.

- R^2 close to **1.0** → excellent fit
- R^2 near **0.0** → weak model

• MAE (Mean Absolute Error)

Measures the average magnitude of errors without considering direction.

- Lower MAE indicates more accurate predictions
- Less sensitive to outliers than RMSE

• RMSE (Root Mean Squared Error)

Square root of the average squared differences between predicted and actual values.

- Strongly penalizes large errors
- Lower RMSE → better model reliability

These three metrics together provide a balanced view of accuracy and robustness.

2. Evaluation Process

The evaluation used the processed feature dataset generated in Week 4:

```
df = pd.read_csv("data/daily_features.csv")
X = df[["day","month","year","day_of_week","is_weekend"]]
y = df["count"]
```

For reproducibility, the evaluation script iteratively loaded each trained model, generated predictions, calculated metrics, and stored results in a structured format:

```
results = []
for model_name in ["RandomForest", "GradientBoosting",
"LinearRegression"]:
    with open(f"ml_code/models/{model_name}.pkl", "rb") as f:
        model = pickle.load(f)
    y_pred = model.predict(X)
    results.append([
        model_name,
        r2_score(y, y_pred),
        mean_absolute_error(y, y_pred),
        mean_squared_error(y, y_pred, squared=False)
    ])
```

The final results were presented in a DataFrame for easy comparison.

3. Model Performance Table

Model	R^2 Score	MAE	RMSE
Random Forest	**0.983**	**23.93**	**36.61**
Gradient Boosting	0.941	50.65	68.42
Linear Regression	0.765	109.24	136.66

Interpretation of Results

- **Random Forest clearly outperformed all other models**, achieving:
  - Highest R^2 score (0.983) – meaning it explains nearly all variance in crime data
  - Lowest MAE and RMSE – meaning predictions were very close to actual values
- **Gradient Boosting performed reasonably well** but produced higher errors than Random Forest.
- **Linear Regression performed the weakest**, indicating that crime patterns are not strictly linear and require more expressive algorithms.

Conclusion of Evaluation

Random Forest was selected as the final model for deployment due to its superior performance and stability across all evaluation metrics.

6.5.2 ML Pipeline Finalization

After selecting the Random Forest model as the best-performing algorithm, the next critical step was to encapsulate the entire machine learning workflow into a **reproducible pipeline**.

Using a pipeline ensures that:

- Feature scaling is consistently applied
- The model can be deployed easily
- Input preprocessing and prediction occur in one step
- The system is modular and maintainable

1. Pipeline Structure

The final pipeline included:

Pipeline Step	Purpose
StandardScaler	Scales numerical features to stabilize variance and improve model performance
RandomForestRegressor	Final selected model for prediction

The pipeline was defined as:

```
pipeline = Pipeline([
```

```
    ('scaler', StandardScaler()),
    ('model', RandomForestRegressor(n_estimators=100, random_state=42))
])
```

2. Training the Final Pipeline

- pipeline.fit(X, y)
- joblib.dump(pipeline, "ml_code/models/final_model.pkl")

Features Used

- day
- month
- year
- day_of_week
- is_weekend

Output

A serialized pipeline file:

- final_model.pkl

This file would later be loaded into the Streamlit app for real-time prediction.

6.5.3 Streamlit Dashboard Development

Objective

To create an interactive and user-friendly **web dashboard** that allows non-technical users to:

- View crime trends
- Examine model performance
- Generate predictions based on custom dates
- Analyze KPIs in real time

The dashboard was implemented using **Streamlit**, a Python library ideal for rapid app development and data visualization.

1. Streamlit Application Structure

Location:

- ml_code/streamlit_app/app.py

The dashboard includes the following functional components:

1. KPIs (Key Performance Indicators)

Displayed metrics include:

- Total number of daily records analyzed
- Model files available in the project
- Full date range of the dataset

These give users an immediate overview of system scale and data coverage.

2. Real-Time Crime Prediction

Users select a date from a sidebar calendar widget. The system automatically extracts temporal features and predicts crime count:

```
selected_date = st.sidebar.date_input("Select Date", value=datetime.today())
```

3. Prediction Logic

The dashboard transforms the selected date into features compatible with the model:

```
X_input = pd.DataFrame([{
    "day": selected_date.day,
    "month": selected_date.month,
    "year": selected_date.year,
    "day_of_week": selected_date.weekday(),
    "is_weekend": 1 if selected_date.weekday() in [5,6] else 0
}])
```

4. Crime Trend Visualization

Line charts display:

- Historical crime counts
- Moving averages
- Seasonal or weekly variations

5. Model Performance Table

The dashboard includes a detailed model comparison table presenting R^2, MAE, and RMSE.

Outcome

The dashboard was successfully deployed and operates fully using the trained model. Users can:

- Choose any date
- Generate prediction instantly
- Explore trends visually
- Interpret model performance clearly

This makes the project practical, interactive, and accessible.

6.5.4 Results & Discussion

1. Model Performance

The Random Forest model delivered exceptional accuracy ($R^2 = 0.983$), demonstrating its ability to capture non-linear temporal patterns and seasonal cycles.

2. Pipeline Strengths

- Fully reproducible
- Simple to deploy
- Automatically processes inputs
- Handles feature scaling internally

3. Dashboard Impact

The dashboard transforms complex ML workflows into a tool usable by students, analysts, or policymakers. It enables:

- Real-time predictions
- Clear visual insights
- Easy interpretation of model performance

4. Insights Observed

- Weekends often show higher predicted crime counts

- Crime follows strong monthly seasonality
- The merged city dataset reveals structural differences between Chicago and NYC

6.5.5 Conclusion

Week 4 and Week 5 successfully completed the Machine Learning component of the project.

Week 4 Achievements

- Generated high-quality temporal features
- Produced a fully cleaned and validated ML dataset
- Trained three baseline models
- Saved serialized versions for evaluation

Week 5 Achievements

- Evaluated models using rigorous metrics
- Selected Random Forest as the final model
- Built a complete ML pipeline for deployment
- Developed and deployed an interactive Streamlit dashboard
- Enabled real-time daily crime prediction

Overall Project Outcome

The Crime Prediction Project now offers a fully functional Big Data → EDA → ML → Deployment pipeline capable of:

- Processing multi-million–row datasets
- Generating actionable insights
- Predicting crime counts accurately
- Presenting results through a user-friendly interface

7. Significance

7.1 Academic Significance

This project holds substantial academic value because it integrates a wide range of concepts taught in modern data science, Big Data analytics, and machine learning curricula. Through the use of real, large-scale crime datasets from Chicago and New York City, the project accomplishes the following:

1. Demonstrates mastery of Big Data engineering concepts

The work involves data ingestion, distributed processing, and large-scale file management—key components of Big Data ecosystems. By implementing HDFS-style directory structures and Spark-based ETL operations, the project replicates workflows typically encountered in enterprise data engineering roles.

2. Integrates multiple technical disciplines into a cohesive pipeline

The project goes beyond simple analysis by connecting the entire data lifecycle:

- Raw data ingestion
- Data cleaning and transformation
- Feature engineering
- Exploratory data analysis
- Machine learning model development
- Model evaluation and comparison
- Dashboard deployment

This offers a complete end-to-end demonstration of analytical engineering practices.

3. Strengthens applied machine learning skills

Students gain hands-on experience in time-series prediction using models such as Random Forest and Gradient Boosting. They also apply evaluation metrics like R^2, MAE, and RMSE, reinforcing understanding of model accuracy and generalization.

4. Provides real-world exposure to data pipeline development

The project simulates professional workflows typically found in organizations dealing with structured and semi-structured data. This includes:

- Automated processing scripts
- Scalable computation using Spark
- Job orchestration through ETL steps
- Data storage and retrieval management

This exposure prepares students for advanced academic research or industry-level data engineering roles.

5. Aligns with key learning outcomes in Big Data and Analytics courses

The project reflects the academic objectives of courses focusing on:

- Big Data Architecture
- Distributed Computing
- Predictive Analytics
- Machine Learning Methods
- Business Intelligence and Visualization

In summary, the academic significance lies in the project's ability to merge theory with practice, demonstrating both technical competency and analytical thinking.

7.2 Practical Significance

Beyond its academic contributions, the project carries meaningful societal and real-world relevance, especially in the context of public safety and urban management.

1. Enhances understanding of temporal crime behavior

The project identifies and visualizes patterns in crime occurrences across time. These insights include:

- Seasonal variations
- Monthly and yearly crime trends
- Differences between weekdays and weekends
- Historical spikes and declines in specific crime categories

Such insights help uncover underlying behavioral and societal drivers.

2. Provides actionable forecasting for stakeholders

Using machine learning predictions, the system estimates daily crime counts. These forecasts can support:

- Police patrol scheduling
- Emergency response planning
- Allocation of manpower and resources
- Risk assessment for neighborhoods or districts

Even though the dashboard is academic, the predictive logic reflects real tools used in crime analytics units around the world.

3. Demonstrates practical value of data-driven decision-making

The project exemplifies how cities can leverage historical data to anticipate future needs. Predictive analytics allow local governments to:

- Reduce crime through proactive measures
- Improve deployment of safety initiatives
- Identify high-risk times and locations
- Enhance overall public safety strategies

4. Supports ethical and informed urban planning

Understanding crime trends assists policymakers and urban planners in creating safer communities. Forecasts can inform:

- Street lighting improvements
- CCTV placement
- Community awareness programs
- Social interventions during high-risk periods

5. Bridges technical capability with real societal impact

The project not only showcases advanced engineering and analytical techniques but also demonstrates how such technologies can benefit society. It highlights the rising role of data science in public policy, law enforcement, and community welfare.

8. Big Data Requirement Justification

The crime datasets used in this project fully meet the classification of **Big Data**. This classification is not solely based on size, but also on the complexity, diversity, and computational demands required to process them. The following characteristics justify the use of Big Data frameworks such as Hadoop-style storage and Apache Spark.

8.1 Dataset Volume

The datasets involved contain extremely large record counts and span decades:

Chicago Crime Dataset

- Contains **7+ million individual crime reports**
- Covers **more than two decades**, from 2001 to the present
- Raw CSV file size reaches several gigabytes

NYC Crime Dataset

- Contains **millions of complaint/arrest records**
- Updated annually, covering **15–20 years**

- Includes multiple tables with varying structures and fields

These datasets significantly exceed the capabilities of traditional tools such as:

- Microsoft Excel (which can only handle ~1 million rows)
- Baseline Python/pandas (limited by system RAM)
- Simple desktop applications that lack memory-efficient processing

Operations like merging datasets, cleaning millions of rows, or recalculating daily time-series statistics would be extremely slow or entirely infeasible without distributed processing.

8.2 Velocity and Variety

Velocity (Data Update Frequency)

Although historical, these datasets are updated **monthly or yearly**, meaning an analytical system must support:

- Frequent ingestion of new data
- Automated re-processing
- Incremental aggregation

Variety (Format and Structure Differences)

The datasets exhibit high structural variability:

- Different timestamp formats (e.g., "MM/DD/YYYY", ISO-8601)
- Variations in crime type labels across cities
- Missing, incomplete, or noisy records
- Mixed data types (categorical strings, numerical IDs, timestamps)

This requires flexible and scalable cleaning pipelines capable of:

- Schema normalization
- Type casting
- Deduplication
- Multi-source merging

Basic tools are not sufficient for such preprocessing at scale.

8.3 Why Big Data Tools Are Required

Memory Limitations

A standard laptop cannot load 8–12 million rows into memory all at once without slowdowns or crashes.

Spark Advantages

Apache Spark is essential because it:

1. Enables Distributed Computation

Spark breaks tasks into parallel operations across:

- Multiple CPU cores
- Multiple nodes (if on a cluster)

This improves speed dramatically for:

- Filtering millions of rows
- Grouping and aggregating
- Joining multiple datasets
- Generating daily, monthly, and yearly statistics

2. Efficient Memory Management

Spark processes data in chunks instead of loading everything into RAM.

3. Fast Transformations

Operations that take minutes or hours in pandas can be done in seconds with Spark.

4. Fault Tolerance

Spark's RDD system handles:

- Node failures
- Network delays
- Interrupted computations

9. Ethical Considerations

Working with crime data requires careful attention to ethical safeguards. Although the datasets are publicly available, responsible use is essential to avoid misuse, bias reinforcement, or privacy violations. This project incorporates multiple layers of ethical practice.

9.1 No Personal Identifiers

The datasets used contain **no personally identifiable information (PII)**. They exclude:

- Names
- Phone numbers
- Addresses
- National IDs
- Social security numbers
- Photos

All records are aggregated crime logs, not individual profiles. This ensures compliance with privacy requirements and ethical research standards.

9.2 Aggregated, Not Individual-Level Analysis

All analytical methods focus on **patterns and trends**, not on individuals.

Examples of aggregate-level analyses:

- Daily crime counts
- Monthly trends
- Crime type distributions
- Weekday vs. weekend comparisons

This avoids:

- Tracking specific people
- Identifying individual offenders or victims
- Any form of targeted profiling

9.3 Non-Prescriptive Predictions

The machine learning model uses historical patterns to produce **statistical forecasts**. These predictions are:

- ❌ NOT meant to predict individual behavior
  ❌ NOT meant for operational policing decisions
  ❌ NOT intended to direct surveillance toward demographics
- ✓ ONLY meant to demonstrate academic forecasting techniques
  ✓ ONLY used for high-level pattern prediction

The dashboard is an educational tool, not a decision-making system.

9.4 Responsible Use of Public Data

The datasets are:

- Released by government agencies
- Intended for research and transparency
- Openly accessible to the public

This project respects data availability conditions by:

- Not redistributing modified datasets
- Ensuring transformations remain ethical
- Using the data strictly for academic purposes

9.5 Bias Awareness and Mitigation

Crime data often reflects systemic biases:

- Over-policing in certain neighborhoods
- Underreporting in others
- Historical inequalities in justice systems

To prevent reinforcing bias:

- The project focuses on **temporal** patterns (time-based) instead of **spatial** or **demographic** patterns
- No race, ethnicity, socioeconomic status, or neighborhood-level profiling is conducted
- Interpretations are conservative and avoid stereotype-based conclusions

10. Timeline

The project follows a structured, multi-phase timeline designed to ensure organized progress from raw data acquisition to final reporting. Each stage is intentionally

sequenced to reflect industry-standard data science workflows and academic expectations in Big Data and Machine Learning projects.

Phase 1 — Dataset Acquisition (Week 1)

This phase focuses on collecting and validating all necessary datasets before any processing begins.

Tasks

- Download multi-year crime datasets from:
  - Chicago Data Portal
  - NYC Open Data Portal
- Examine raw files to verify:
  - Schema structure
  - Timestamp formats
  - Crime type consistency
  - File size and memory requirements
- Organize datasets into a structured directory hierarchy suitable for Big Data pipelines, e.g.:
  - /raw_data/chicago/
  - /raw_data/nyc/

Outcome

A well-organized dataset repository, ready for ingestion into the ETL pipeline.

Phase 2 — ETL Pipeline Implementation (Weeks 2–3)

This stage focuses on building a scalable Big Data pipeline using Apache Spark.

Tasks

- Set up the Big Data environment:
  - Install and configure Apache Spark
  - Configure local Hadoop/HDFS for file distribution
- Develop ETL scripts to:
  - Load multi-million-row datasets efficiently
  - Clean missing, inconsistent, or malformed records
  - Normalize timestamps and crime categories
  - Remove duplicates and enforce schema consistency
- Generate structured outputs:
  - Daily crime counts
  - Weekly aggregated tables

    - Monthly trend datasets
- Validate the ETL outputs:
    - Check for missing days
    - Ensure correct date ranges
    - Confirm aggregation accuracy

Outcome

High-quality structured datasets prepared for time-series analysis.

Phase 3 — Time-Series Construction (Week 4)

This phase prepares modeling-ready datasets by transforming aggregated outputs into complete time-series structures.

Tasks

- Merge ETL outputs into unified daily time-series tables
- Handle gaps (missing days) by forward-filling or zero-padding
- Resolve timestamp inconsistencies between Chicago and NYC
- Generate final feature-engineered datasets for modeling, including:
    - Day, month, year
    - Weekday/weekend
    - Crime count

Outcome

Two clean, continuous time-series datasets ready for forecasting models.

Phase 4 — Modeling and Predictions (Weeks 5–6)

This stage focuses on performing statistical analysis and building forecasting models.

Tasks

- Conduct diagnostics:
    - Stationarity tests (ADF)
    - Seasonality decomposition
    - Autocorrelation and partial autocorrelation analysis
- Train forecasting models:
    - ARIMA
    - SARIMA
    - Holt–Winters (Exponential Smoothing)

- Compare models based on:
  - RMSE
  - MAE
  - AIC/BIC (for ARIMA/SARIMA)
- Produce short-term and long-term forecasts for both cities
- Document insights and differences between model predictions

Outcome

Validated crime prediction models and documented comparative findings.

Phase 5 — Visualization and Final Report Writing (Weeks 7–8)

The final phase converts the analytical work into visual and written deliverables.

Tasks

- Create visualizations:
  - Long-term trend lines
  - Seasonal decomposition charts
  - Forecast graphs
- Build the final project report including:
  - Introduction and background
  - Big Data methodology
  - ETL pipeline explanation
  - Time-series modeling approach
  - Forecast results and interpretation
  - Significance and ethical considerations
  - Limitations and recommendations
- Prepare final deliverables:
  - Dashboard (Streamlit)
  - Forecast charts
  - Final write-up (PDF or DOCX)
  - Full GitHub repository

Outcome

A fully completed academic project with professional-level documentation, visualizations, and predictions ready for submission and presentation.

11. Limitations

Despite the successful implementation of a complete Big Data and Machine Learning pipeline, several limitations remain that influence the accuracy, interpretability, and applicability of the results:

1. Data Quality Limitations

- The Chicago and NYC datasets contain missing values, inconsistent timestamps, duplicated records, and occasional reporting errors.
- Crime categories are broad and sometimes inconsistently labeled across years.
- Some dates have incomplete reporting due to system outages or delayed uploads.

2. Temporal Limitations

- Crime data is influenced by unpredictable external factors such as policy changes, major events, public health crises, or socioeconomic shifts.
- Models assume stable crime patterns over time, which may not hold during abrupt structural changes.

3. Modeling Limitations

- The Random Forest model captures nonlinear patterns but does not inherently incorporate long-range temporal dependencies as well as deep learning methods (e.g., LSTM).
- The selected time-based features (day, month, year, day_of_week, is_weekend) are informative but may not capture more complex seasonal or sociological factors.

4. Spatial Limitations

- This project focuses on citywide aggregated counts.
- Crime varies significantly by neighborhood, district, or street block, and aggregated models cannot capture:
    - localized hotspots
    - community-level variations
    - spatial correlations

5. Ethical Limitations

- Crime data may contain historical biases driven by enforcement patterns rather than true crime incidence.

- Forecasting models trained on biased data risk reproducing these patterns unless spatial and demographic context is included with caution.

6. Technical Limitations

- Although Spark handles large data efficiently, the local Hadoop/Spark environment cannot match the scalability of full distributed clusters in production environments.
- The final Streamlit dashboard performs well but may slow down with extremely large datasets or extended time ranges.

Conclusion

This project successfully demonstrates the construction of a complete Big Data and Machine Learning system for crime analysis and prediction. Through Spark-powered ETL pipelines, extensive feature engineering, comparative model evaluation, and dashboard deployment, the project integrates all major components of modern data science workflows.

The Random Forest model emerged as the most accurate and reliable method for short-term crime prediction based on daily temporal features. The Streamlit dashboard further translates this modeling work into an intuitive interface that allows end-users to visualize crime trends and generate real-time forecasts.

Beyond technical execution, the project underscores the importance of ethical practices, recognizing the sensitivities and biases often inherent in crime datasets. While results are promising, further work is needed to incorporate spatial context, broader socioeconomic variables, and advanced forecasting techniques to improve robustness and applicability.

Overall, the project provides both academic value through the application of Big Data methodologies and societal relevance by illustrating how data-driven insights can support informed public safety planning.