System for Analysis and Predictions of Crimes in Los Angeles

Diogo Pinto: 2024156583
Giovanni Maffeo: 2024232210
João Carvalho: 2024173154

May 17, 2025

Source code: Access Link to Github

1 Requirements

1.1 Dataset

First of all, it is important to recall some important points about the dataset that were already presented in our first submission.

The dataset that is going to be utilized is available on Kaggle [1]. It contains records of registered crimes committed in the city of Los Angeles between January 2020 and June 2023. The dataset contains 743,816 samples and 28 features.

Although the dataset is hosted on Kaggle, one of the leading platforms for accessing and downloading open datasets, its description is quite general. To gain a deeper understanding of the available features and the dataset's potential, we searched for another repository [2] that provides a detailed description of each feature. This additional information was crucial for our project, as it enhances our knowledge of both the domain and the dataset. Ultimately, this detailed description originates from the Los Angeles Police Department, which is also responsible for data collection.

Finally, it is important to highlight the primary stakeholders of our application: the Los Angeles Police Department, the Los Angeles Government and the citizens of Los Angeles. Since these stakeholders have different profiles and interests, we must balance highly technical analyses, capable of generating valuable insights for public policies, with analyses that are relevant to an ordinary citizen who seeks to understand the security situation in their city.

1.2 Big questions

In this context, following the dataset description, the set of relevant questions that can be answered based on our application are:

- What will be the crime reporting time, in days?
- What will be the severity of a crime based on the register descriptions?
- What is the most likely final status of a crime based on its characteristics?

From a more technical perspective, specifically targeting BI analysis, these questions can be answered as follows:

- The first question clearly represents a regression problem, since the output is a continuous value. To solve it, we implemented regression techniques in Machine Learning such as Linear Regression and Random Forest Regressor, whose results will be compared in Section 5.
- The second question aims to predict the severity of a crime, categorized as low, medium or high. Therefore, we chose to approach it using an unsupervised classification technique through Clustering.
- The third question focuses on predicting the most likely final status of a crime based on the characteristics of the report. These statuses include: JA, AO, JO, CC and IC, which will be explained in more detail in Section 4. To address this problem, we selected a range of classification methods studied in class to compare their performance.

1.3 Data mining process

The Data Mining process we followed is illustrated in the figure below.

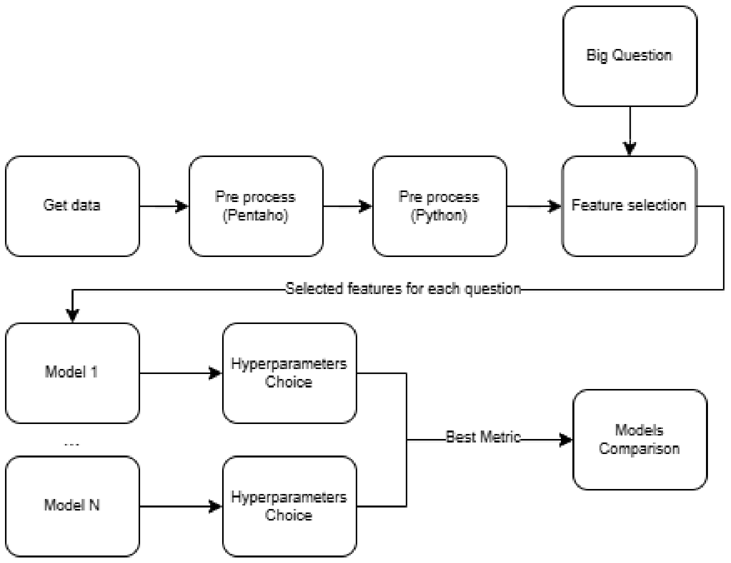

Figure 1: Data mining process

For each model, we first performed exploratory runs to define the best combination of hyperparameters through grid search. This step was essential to tune the model to our specific dataset and improve its generalization capacity.

Once the best configuration was found, we executed 15 runs with different random seeds and data splits to evaluate model stability and performance. This strategy allowed us to simulate the effect of using different training and testing sets, ensuring that our comparison was based on the average performance across multiple scenarios. Finally, the final metric used to compare models varied depending on the Big Question.

2 Pre processing

For the preprocessing stage, we decided to divide it into two phases: the first one performed in Pentaho and the second in Python. We adopted this approach because, in the first part of the project, which focused on Data Warehousing during the transformation stage, we implemented several important steps on the transformation of the original raw dataset. Thus, these steps were preserved for the dataset used in this part of the project, which focuses on Data Mining, especially those related to the removal of invalid data, data cleaning and feature addition, which we will explain below.

Therefore, we applied some adjustments that were unnecessary for this stage, mainly by removing Load steps from the ETL flow in the Pentaho '.ktr' file and excluding parts that were not relevant to the Data Mining approach. An example is the "Descent Code Conversion," where we performed a lookup to replace codes with their corresponding descriptions and included them in the dimension table, which was important for Data Warehousing analysis but makes no difference for this stage, as Machine Learning models do not distinguish between a description or a code.

After that, based on the CSV file generated by Pentaho, we implemented the preprocessing techniques required for the Data Mining approach using Python, such as additional data cleaning and formatting, label encoding, feature selection and others.

2.1 Pentaho

For the preprocessing stage in Pentaho, we performed data cleaning and feature addition, as detailed in the following subsections.

2.1.1 Cleaning data

In the first stage, we analyzed the percentage of missing values in each column. Based on this analysis, we decided not to use the following features: `Crm Cd 3` and `Crm Cd 4`, which are two of four columns in the dataset representing the type of crime (`Crm Cd 1` to `Crm Cd 4`). These columns capture cases where multiple crimes of different severity were recorded in a single register. However, columns 3 and 4 contained a very high percentage of missing values, with missing in 99% of the samples, so we chose to remove them. We also excluded `Cross Street`, which had 83.90% of missing values. These fields are optional and rarely filled in the dataset, so their removal was guided by data quality and intuition. Additionally, we chose to rely on other features to represent the location where the crime was committed.

Furthermore, we decided to remove every sample that contained invalid or missing values in the features we intended to keep. This included cases where the victim's age was null or less than or equal to zero, and cases where the victim's sex or descent were missing.

2.1.2 Feature Addition

For feature addition, we added some key attributes that were considered crucial:

- **Severity Classification**: Each crime record is associated with a crime type, which allows us to infer its severity level. To achieve this, we performed a classification process using a combination of manual classification and large Language Models (LLMs), specifically ChatGPT 4o.
  - First, we used an LLM to classify the 137 unique crime types according to their severity.
  - Then, we manually reviewed and adjusted the classifications with our interpretation.
  - Finally, the generated classification file was used to associate each crime record with its corresponding severity level.

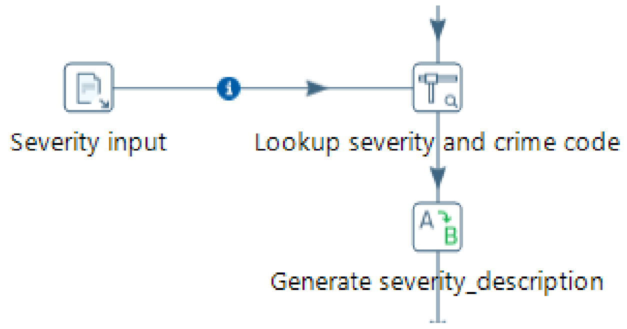

Figure 2: Add severity to crime

- **Add Weekdays**: We added the day of the week for each crime occurrence by using the Calculator step, converting the numeric weekday index (generated by Pentaho) into its corresponding name.

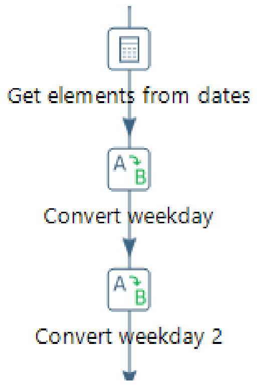

Figure 3: Add weekdays to crime

- **Days Difference**: Another important feature was the difference in days between the occurrence date and the registration date. This addition allows us to compute the time victims take to report a crime, in days. We had to use days instead of hours because the dataset does not include the time of the occurrence or the report, only the dates.

The complete preprocessing workflow performed in Pentaho is illustrated below:

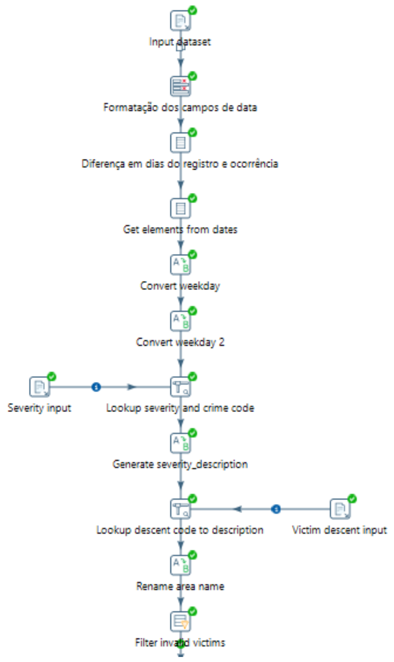
Input dataset
Formatação dos campos de data
Diferença em dias do registro e ocorrência
Get elements from dates
Convert weekday
Convert weekday 2
Severity input
Lookup severity and crime code
Generate severity_description
Lookup descent code to description
Victim descent input
Rename area name
Filter invalid victims

Figure 4: Pentaho Pre processing 1

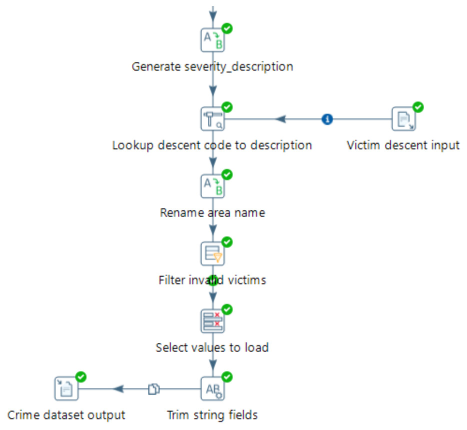

Figure 5: Pentaho Pre processing 2

2.2 Python

2.2.1 Preventing Bias in Model Training

One of the most important preprocessing steps was the analysis of which records or features should be removed depending on the specific questions we aimed to answer. This was crucial to avoid leading the models toward biased interpretations or learning shortcuts that would compromise the analysis.

The first case relates to the question described in Section 4, where we aim to predict the severity of a crime. However, this column was created by us, as detailed in Subsection 2.1, based on the `crime_type_code` column. We manually mapped each crime type to a severity level. Therefore, it makes no sense to use the original `crime_type_code` as an input to predict the severity, since this would allow the model to simply learn the same mapping, making the task trivial. This would result in a biased model that does not generalize, as all records with the same crime type are associated with the same severity label.

The second case relates to the question described in Section 5, where we aim to predict the final status of a crime. However, many records in the dataset are labeled with the status `IC` (Investigation Continued), meaning the investigation is still ongoing and the case has not yet received a final status. These records are effectively unlabeled and, including them in training or validation would compromise the learning process. For this reason, we removed all records with the status `IC` when training and evaluating the models for this specific task, because it is a temporary label.

2.2.2 Cleaning data

The first preprocessing step was splitting a column named `mo_codes` in our dataset. This step had not been implemented during the Data Warehousing phase because it was not considered relevant to the big questions at that point.

The `mo_codes` column contains information about the *Modus Operandi* (MO), a Latin term meaning "mode of operation." In law enforcement, it refers to a criminal's behavioral pattern, including tactics and recurring strategies. This number is part of a classification system used by the Los Angeles Police Department (LAPD) to categorize a suspect's methods or behavior when committing a crime. For example:

- **2105**: Draws diagrams or takes notes;
- **2107**: Abandons vehicle in a restricted area

All MO codes associated with a crime were stored in a single column, separated by spaces. Our task was to split this information into multiple columns. We identified that the maximum number of MO codes per record was 10, so we performed a space-based split and assigned up to 10 separate MO code columns for all records.

Another preprocessing step was the addition of a rule to replace null values in the `weapon_code` and `crm_cd_2` columns. This was necessary because these fields often contained missing values when a given crime did not involve a weapon or when no secondary, less severe crime was recorded, respectively. In this sense, these missing values also represent relevant information for the model, as they may indicate less serious crimes or lead to different final status for the crime, for example.

To handle these missing values uniformly, we replaced null entries with the code `0`, which was not used in the original dataset for these columns. This allows Machine Learning models to interpret the absence of a weapon as just another valid category, alongside others such as "knife" or "firearm" for example.

2.2.3 Encoding

Next, since we have many features that represent categorical values (such as codes), we needed to apply encoding to these labels. However, it is important to handle them carefully, as there are two types of categorical features: those that involve an inherent order and those that do not. This distinction is crucial because they require different encoding techniques. Encoding is important in Machine Learning because models typically cannot process non-numerical values directly.

For example, career growth within a company is a clear case of an ordinal categorical feature, as it follows a progression. In such cases, the best strategy is label encoding, where the lowest rank could be encoded as 0 and higher levels as increasing integer values.

On the other hand, eye color is a categorical feature without any inherent order. In this case, models interpret the data better when we transform each category into separate binary columns such as `is_blue` or `is_green`, one for each label. This approach is known as One-Hot Encoding.

For our features, we applied the following encoding strategies:

Table 1: Features encoded using One-Hot Encoding

Feature	Description
`department_code`	Police department code
`vict_sex`	Victim's gender
`vict_descent_code`	Victim's descent category
`occurence_date_weekday`	Weekday of crime occurrence
`register_date_weekday`	Weekday of crime registration

Table 2: Features encoded using Label Encoding

Feature	Description
`status_code`	Final status of the crime
`structure_code`	Type of structure, vehicle or location
`weapon_code`	Weapon used in the crime
`severity_code`	Crime severity classification
`crime_type_code`	Main crime type
`crime_type_code2`	Secondary crime type
`mo_code1` to `mo_code10`	Modus Operandi codes per record

It is important to note that some features such as `structure_code`, `weapon_code`, `crime_type_code` and `mo_code` do not represent ordinal categories. In theory, One-Hot Encoding would be more appropriate for these cases, as it prevents the model from learning artificial order in the values. However, using One-Hot Encoding on such features would drastically increase the number of dimensions in the dataset, making it more computationally expensive and potentially harder for models to learn effectively.

For example, there are 137 unique values in `crime_type_code`, which would generate 137 new columns if One-Hot Encoding were applied. For this reason and to maintain computational efficiency, we opted for Label Encoding even in cases where the categories were nominal.

2.3 Feature selection

Finally, it is also important to mention that, in the part of the project focused on Data Warehousing, we implemented feature deletion directly in Pentaho, with steps that were also removed in this Data Mining phase. In that context, we excluded features that were not aligned with the big questions of that part of the project and were therefore considered useless.

However, in a Data Mining approach, it is difficult to determine in advance which features are relevant to our big questions, especially when we are not domain experts. As such, feature deletion should

rely on a feature selection or feature reduction algorithm, which in our project is detailed in this section.

2.3.1 Feature Selection Based on Correlation

In this Data Mining phase, we decided to consider all features by default before applying any selection or reduction techniques. There are several possible approaches for this task, ranging from entropy-based feature selection to feature reduction techniques based on variance or label separability.

For this project, we chose a simple correlation-based method. The main idea is to analyze the features that show the highest correlation with the target label, which varies depending on the specific question. Features with low correlation were discarded.

To avoid discarding potentially relevant information due to the simplicity of this method, we established a low threshold of 5%. That is, we kept all features with an absolute correlation value of at least 0.05 with the respective target.

Based on this analysis, we generated the following correlation heatmap:

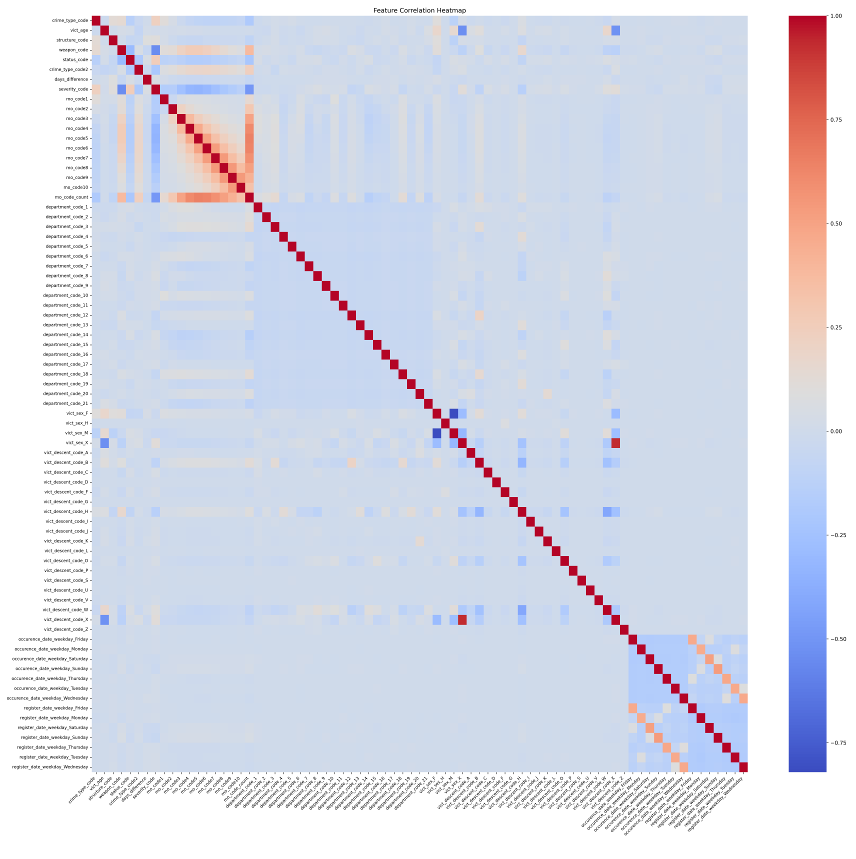

Figure 6: Correlation heatmap

As the target variable differs for each question, the set of selected features also varied accordingly. The tables below list the features selected for each case, along with their correlation values.

Table 3: Correlated features for **Question 1** (target: `days_difference`)

Feature	**Correlation**
`structure_code`	0.0880
`weapon_code`	-0.0585
`severity_code`	0.0578
`mo_code_count`	-0.0644

Table 4: Correlated features for **Question 2** (target: `severity_code`)

Feature	**Correlation**
`structure_code`	0.1031
`weapon_code`	-0.5411
`status_code`	0.2539
`days_difference`	0.0578
`mo_code1`	-0.1620
`mo_code2`	-0.1555
`mo_code3`	-0.2580
`mo_code4`	-0.3292
`mo_code5`	-0.3399
`mo_code6`	-0.3122
`mo_code7`	-0.2657
`mo_code8`	-0.2148
`mo_code9`	-0.1681
`mo_code10`	-0.1268
`mo_code_count`	-0.4932
`department_code_8`	0.0605
`department_code_12`	-0.0737
`department_code_14`	0.0501
`department_code_18`	-0.0623
`vict_sex_X`	0.1143
`vict_descent_code_B`	-0.0977
`vict_descent_code_H`	-0.1362
`vict_descent_code_W`	0.0946
`vict_descent_code_X`	0.1135

Table 5: Correlated features for **Question 3** (target: `status_code`)

Feature	**Correlation**
`crime_type_code`	-0.1358
`structure_code`	-0.0608
`weapon_code`	-0.3006
`crime_type_code2`	-0.1859
`severity_code`	0.2539
`mo_code1`	-0.1127
`mo_code2`	-0.0961
`mo_code3`	-0.1307
`mo_code4`	-0.1599
`mo_code5`	-0.1628
`mo_code6`	-0.1552
`mo_code7`	-0.1328
`mo_code8`	-0.1099
`mo_code9`	-0.0834
`mo_code10`	-0.0615
`mo_code_count`	-0.2149
`vict_sex_F`	-0.0697
`vict_sex_M`	0.0506
`vict_descent_code_H`	-0.0880
`vict_descent_code_W`	0.0574

Finally, it is worth noting that only the features listed above were considered for training the models developed for each of our Big Questions.

3 Question 1

3.1 Question goals

For each crime in the dataset, we have both the date when the incident occurred and the date when it was reported to the police. The time difference between these two dates—referred to here as the "reporting time" can vary significantly depending on multiple factors.

In certain high-risk areas, for example, residents may delay reporting crimes, or even stop reporting them altogether, due to a perceived lack of response or effectiveness from law enforcement. If victims feel that reporting a crime will not lead to improved safety or justice, the incentive to report diminishes. In other cases, the severity of the incident might prevent the victim from reporting it promptly due to trauma or medical incapacitation. Additionally, logistical factors such as overcrowded police departments, limited access to police stations or inefficient reporting channels may also contribute to delays.

Thus, the objective of this question is to estimate the expected reporting time for a given crime based on contextual features. This information is particularly relevant for stakeholders such as the Los Angeles Police Department and local government authorities. Understanding the patterns behind delayed reporting can inform strategic decisions, such as reallocating police resources, improving accessibility to law enforcement services, redesigning reporting mechanisms or reevaluating security policies in specific areas.

By predicting the expected reporting time, policymakers and law enforcement agencies may identify underserved communities, improve response planning and ultimately enhance the overall efficiency of the public safety system.

3.2 Evaluation metrics

In Machine Learning, regression and classification tasks require different evaluation metrics, as they aim to solve fundamentally distinct types of problems. Classification models predict discrete categories, so they are typically evaluated using metrics such as accuracy, precision, recall or F1-score. In contrast, regression models predict continuous numerical values and are evaluated using error-based metrics.

For regression, two of the most commonly used metrics are:

- **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and actual values. It provides a clear interpretation in the same unit as the target variable and is less sensitive to outliers.
- **Mean Squared Error (MSE)**: Calculates the average of the squared differences between predicted and actual values. It penalizes larger errors more strongly than MAE due to the squaring operation, making it useful when large deviations are particularly undesirable.

In our case, since we aim to predict the number of days between a crime occurrence and its report, MAE is especially suitable. It offers an interpretable average of the prediction error in days, which is directly meaningful to our stakeholders such as LAPD and government. Understanding how many days, on average, our predictions deviate from reality allows for practical assessments of reporting behavior and system efficiency.

3.3 Algorithms choice

The first question clearly characterizes a regression problem, as the target variable is a continuous value—the number of days between the occurrence of a crime and its report to the police. To address this, we implemented supervised Machine Learning techniques, specifically *Linear Regression* and *Random Forest Regressor*. At the end of the process, the results from both models will be compared, since different ML methods may perform better depending on the nature of the data and the problem structure. Therefore, evaluating multiple predictors is essential.

The first algorithm, **Linear Regression**, is a supervised learning method that models the relationship between the target variable and one or more input features by fitting a linear equation. It assumes that the influence of each feature on the target is additive and constant, which makes it well suited for problems where the relationship between input variables and the output is approximately linear. This simplicity leads to fast training and easy interpretability, although it may not perform well in the presence of complex, nonlinear interactions. In our project, we implemented this algorithm using the following import: `from sklearn.linear_model import LinearRegression`.

The second algorithm, **Random Forest Regressor**, is also a supervised learning method, based on an ensemble of multiple decision trees. It builds several simple decision trees, referred to as weak learners, each trained on a random subset of the data and features to introduce diversity. Although each individual tree may have limited predictive power, their combined predictions (through averaging) result in a robust and accurate model. This approach is especially effective for capturing complex, nonlinear patterns in the data. The method is closely related to the `Random Forest Classifier` explored in class, but adapted to continuous outputs using the CART (Classification and Regression Trees) algorithm. We implemented it using: `from sklearn.ensemble import RandomForestRegressor`.

3.4 Hyperparameter choice

An essential step in the development of Machine Learning models is the choice of hyperparameters, also known as model fine-tuning. Many algorithms are highly sensitive to these parameters, which cannot be estimated directly from the data and are often not easily interpretable. Hyperparameter tuning is critical to ensure the model's ability to generalize well to unseen data and to avoid issues such as overfitting, when a model learns the noise or specific patterns of the training data rather than the underlying structure. It also helps prevent underfitting (poor learning capacity), excessive training time and instability in predictions.

It is important to emphasize that the comparison of model performance is conducted under fair conditions, meaning each model is evaluated using the best hyperparameters it can achieve within the defined testing range. In other words, the comparison follows a *best vs. best* approach.

To achieve this, for each algorithm requiring hyperparameters for each Question, we performed a Grid Search, which explores all combinations within a predefined parameter space. The selection of optimal parameters is guided by an evaluation metric appropriate to the problem at hand, since different tasks aim to minimize or maximize different aspects of performance.

Below, we present the tested hyperparameter values and the results obtained according to the most suitable evaluation metric for each question.

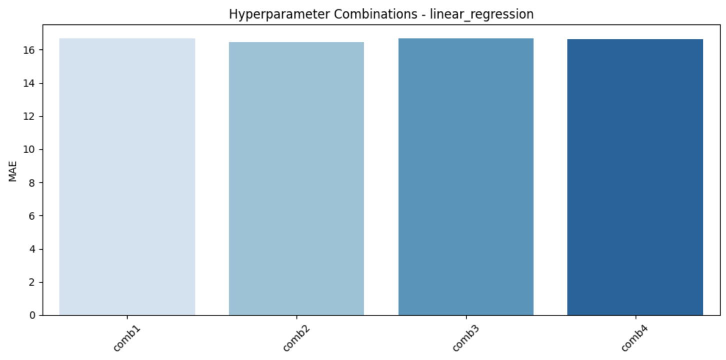

Figure 7: Linear regression

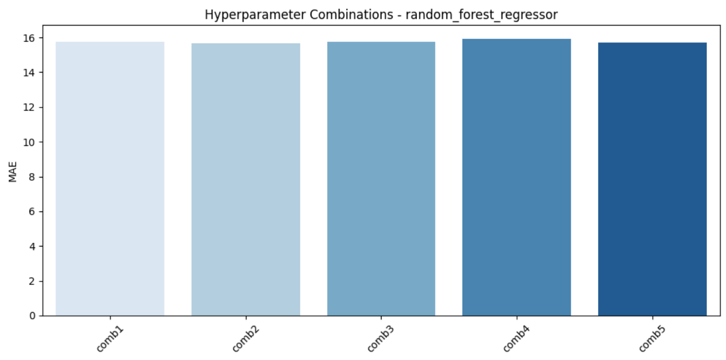

Figure 8: Random forest regression

The best hyperparameters used in each algorithm, as well as their corresponding results, are presented in the tables below. For each case, the combination that yielded the highest evaluation metric was selected for the model comparison shown in Subsection 3.5.

Table 6: Best MAE Scores and Hyperparameters for Each Regression Model

Model	**Best MAE Score**	**Best Hyperparameters**
Linear Regression	16.4353	{fit_intercept: False, positive: False}
Random Forest Regressor	15.6750	{n_estimators: 7, max_depth: 10, min_samples_split: 4}

3.5 Results

As shown in the chart below, and following the experimental procedure described in Subsection 1.3, we observed that the **Linear Regression** model outperformed the **Random Forest Regressor**. This result suggests that the relationship between the input features and the target variable, reporting time in days, is well approximated by a linear function, without the need for complex nonlinear modeling.

This result can also be explained by limitations of the Random Forest model in this context. When the underlying data relationships are mostly linear and low in variance, Random Forest may overfit, build unnecessarily complex trees or fail to capture consistent patterns due to random data and feature splitting.

Therefore, in scenarios where the relationship between predictors and response is straightforward, Linear Regression not only offers better interpretability and lower computational cost but can also outperform more complex models like Random Forest.

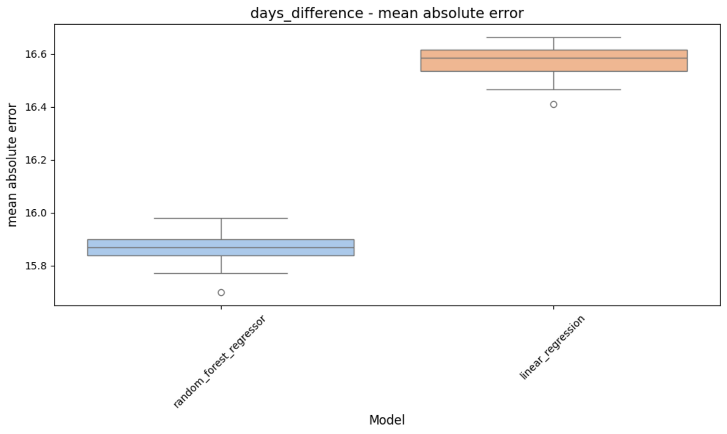

Figure 9: MAE Boxplot

4 Question 2

4.1 Question goals

This question aims to predict the severity of a crime, classified as low, medium or high, based on key characteristics of the incident. In particular, features such as victim demographics, the police department responsible for the area (`department_code`), the structure where the crime occurred (e.g., street, commercial property or vehicle), the use of weapons and the modus operandi (MO) provide valuable signals. For instance, recent trends may reveal that crimes committed in vehicles or involving certain victim profiles tend to be more violent.

In practice, these predictions allow us to capture behavioral and contextual patterns that are often associated with more serious crimes. Crimes involving specific MOs or occurring in high-risk zones can signal a greater likelihood of escalation. Moreover, variables like the final status of the crime, can be related to its severity and further inform predictive accuracy.

This question is highly relevant for stakeholders such as the Los Angeles Police Department (LAPD), public safety organizations and government decision-makers. By anticipating the potential severity of an incident, law enforcement agencies can proactively allocate resources, adjust patrol strategies and prioritize the protection of more vulnerable areas or groups.

Additionally, this information can be of great value to the general public. Citizens can become more aware of which areas present higher risks and which types of crimes or circumstances tend to be more dangerous. This can support more informed behavior, such as avoiding certain regions or recognizing high-risk situations, thereby contributing to personal safety and community awareness.

4.2 Algorithms choice

We opted to approach it using an unsupervised classification technique through Clustering.

To implement the model, we used the `KMeans` algorithm from the `sklearn.cluster` module, combined with additional logic to simulate classification behavior. The idea was to confirm that clustering methods such as K-Means could group the data based on similarity and allow us to later classify new samples by evaluating their proximity to each cluster's centroid.

After clustering, we assigned a severity label to each cluster by identifying the most frequent severity class among its constituent samples. This label assignment enabled us to treat the cluster as a class proxy for prediction purposes.

To streamline integration with our Machine Learning pipeline, we implemented a custom classifier based on K-Means clustering that follows the structure of scikit-learn models. This class includes `fit()` and `predict()` methods, allowing it to be used like any other supervised model within our framework. The pseudocode is presented below:

```
Class KMeansClassifier:
    Initialize with random seed and optional number of clusters

    Method fit(X_train, y_train):
        - Identify the unique labels in y_train
        - Set number of clusters (equal to number of labels, if not specified)
        - Train a KMeans model with the defined number of clusters
        - For each cluster:
            - Count how many samples of each true label fall into the cluster
            - Assign the cluster the most frequent label (majority vote)
            - Optionally, save statistics about cluster purity
```

4.3 Evaluation metrics

Although all standard classification metrics were computed during evaluation, the metric chosen to guide model selection was the **F-score**. This metric represents the harmonic mean between precision and recall, and is particularly suitable when there is a need to balance the impact of false positives and false negatives.

In the context of predicting crime severity, both types of errors carry significant implications. False positives, cases where a crime is incorrectly predicted as more severe than it actually is, may lead to unnecessary allocation of police resources or disproportionate prioritization. On the other hand, false negatives, cases where a severe crime is predicted as mild, can be even more critical, as they may result in insufficient response, delayed action or underestimation of risk.

Therefore, the F-score was chosen as the most appropriate metric because it does not favor only precision or only recall. Instead of tying our evaluation to philosophical preferences, such as whether it is worse to over-allocate police resources or to underestimate the severity of a crime and potentially put a victim at risk, we opted for a balanced metric. The F-score offers a middle ground by equally penalizing both false positives and false negatives, making it a fair and pragmatic choice for this type of classification problem.

4.4 Results

Based on the execution described in Subsection 1.3, we evaluated the model's performance using the F1 Score metric. The results are summarized as follows:

- **Mean F1 Score**: 0.52933
- **Standard Deviation**: 0.01089

As shown in the graph below:

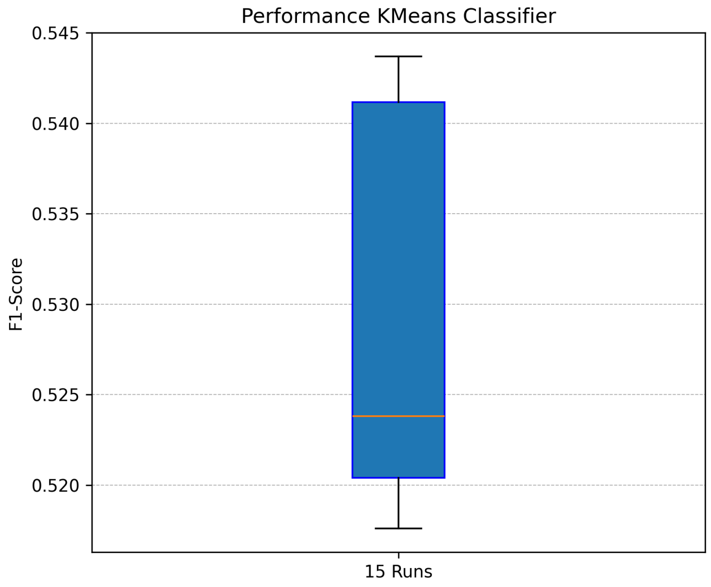

Figure 10: KMeans Classifier Results

This performance can be further understood through the analysis of Figure 11, which illustrates the percentage of the majority class within each cluster across different random seeds. As shown in the figure, most clusters rarely surpass the 60% mark—represented by the dashed red line—indicating that no single class dominates the cluster composition.

This suggests that the clusters formed by K-Means do not strongly align with a single severity label, which impacts the classifier's ability to clearly separate the data. In other words, the lack of cluster purity explains the moderate F1 Score achieved, as the predicted labels based on cluster majority are not consistently representative of the actual labels. This reinforces the inherent challenge of applying unsupervised clustering techniques to problems where class separation is complex.

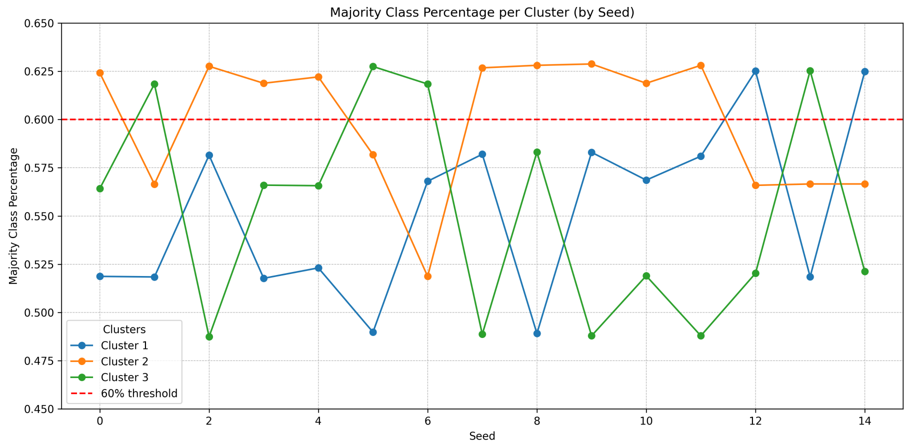

Figure 11: KMeans majority class

5 Question 3

5.1 Question goals

For this question, we decided to implement with classification algorithms, since its a discrete output space. This question focuses on predicting the final `status_code` of a given crime. The status indicates the procedural outcome of the crime report and is essential for understanding how cases progress through the justice system. Below are the definitions of each status code:

Table 7: Status codes and their meanings

Code	Description
AA	**Adult Arrest** — An adult was arrested in connection with the reported crime.
JA	**Juvenile Arrest** — A minor was arrested in connection with the reported crime.
AO	**Adult Other** — Involves an adult but did not result in an arrest; may include citations or other legal actions.
JO	**Juvenile Other** — Involves a minor but did not result in an arrest; may include warnings or diversion programs.
CC	**Case Closed** — The case was closed without resolution, meaning no arrest or other legal action occurred.

The main objective of this predictor is to estimate the final status of a crime based on its characteristics. For instance, if a firearm was involved, it is likely that an adult was implicated in the case; or if the crime was reported with a significant delay, it may increase the chances of the case ending without resolution.

This prediction is particularly valuable for the police department, as one of the key stakeholders. As discussed in Subsubsection 2.2.1, all records labeled with status `IC` ("Investigation Continued") were removed from training, as they represent ongoing investigations and account for approximately 80% of the dataset. Therefore, this classifier could support the LAPD by estimating likely outcomes for these incomplete records, enabling more informed decision-making—even within the current dataset.

5.2 Algorithms choice

For the algorithms we implemented using libraries of sklearn, random forest, decision tree, adaboost and xgboost.

- **Decision Tree**: A simple model that splits the data into decision rules based on feature thresholds.
- **Random Forest**: An ensemble of decision trees trained on random subsets of data to improve generalization.
- **AdaBoost**: A boosting method that combines weak classifiers sequentially, focusing on correcting previous errors.
- **XGBoost**: An optimized and regularized boosting technique that builds trees to minimize prediction error efficiently.

It was also implemented the algorithm of Ensemble of models by hand (even tho random forest, etc was previously utilized) to also compare the performance. For the basic "weak" models it was utilized Decision Trees. The difference between this implementation and the traditional Random Forest is the absence of randomly selection a subset of features. The algorithm is the following:

Algorithm 1 Trees Ensemble training

```
for number of trees do
    Choose a random subset (with replacement) of samples
    Train a Decision Tree with that subset
    Save the Tree
end for
```

Algorithm 2 Trees Ensemble prediction

```
for Tree in Trees do
    Predict the result using Tree
end for
return The most frequent answer
```

5.3 Evaluation metrics

For similar reasons discussed in this Big Question, described in Section 4.3 F-Score was also used to guide both model comparison and hyperparameter selection for this task.

5.4 Hyperparameter choice

The choice of hyperparameters was implemented as described in Subsection 1.3 for each model that required fine-tuning. The importance of this testing strategy is further discussed in Subsection 3.4.

The results are presented below:

Table 8: Best F1 Scores and Hyperparameters for Each Model

Model	**Best F1 Score**	**Best Hyperparameters**
Random Forest	0.6447	{n_estimators: 15, max_depth: 15, min_samples_split: 3}
Decision Tree	0.6411	{max_depth: 10, min_samples_split: 2}
AdaBoost	0.5994	{n_estimators: 5, learning_rate: 1.0}
XGBoost	0.6356	{n_estimators: 10, learning_rate: 0.2, max_depth: 4}

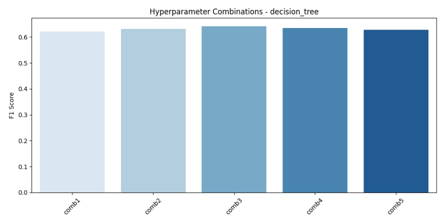

Figure 12: Hyperparameters – Decision Tree

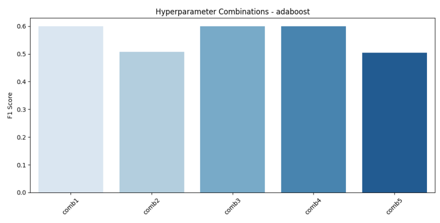

Figure 13: Hyperparameters – AdaBoost

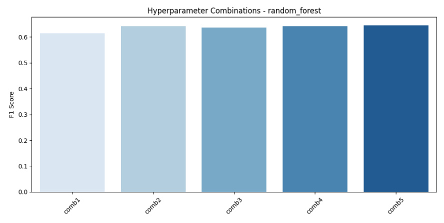

Figure 14: Hyperparameters – Random Forest

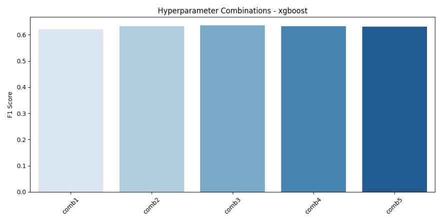

Figure 15: Hyperparameters – XGBoost

5.5 Results

We used the F1 Score as our main evaluation metric, as previously discussed. All evaluations followed the same methodology outlined in Subsection 1.3 and the results are presented similarly to the analysis in Section 3.5. The results are below:

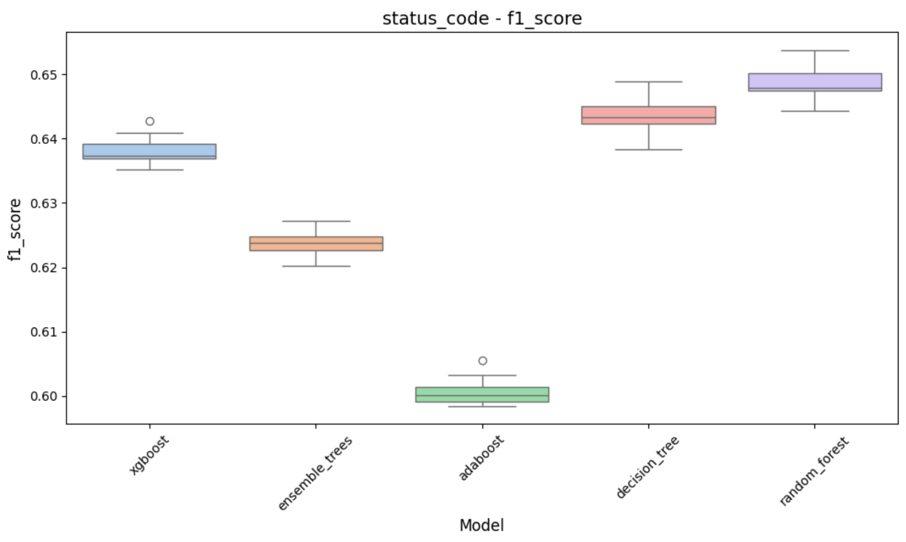

Figure 16: Status code boxplot

The model that achieved the highest overall F1 Score for predicting the `status_code` was the Random Forest. This result reinforces the strength of ensemble-based approaches when there is enough variability in the dataset. Since Random Forest builds multiple diverse decision trees using different subsets of features and samples, it benefits from aggregating multiple weak learners into a robust model. This diversity helps reduce overfitting and leads to better generalization on unseen data.

It also reinforces the importance of testing multiple algorithms, as different models can yield significantly different results depending on the nature of the problem. In our case, while Random Forest performed best here, other models performed better in different Big Questions, demonstrating that there is no one-size-fits-all solution in Machine Learning.

6 Availability and integration

The models developed can be integrated into the BI workflow by exporting predictions to structured tables that are connected to Tableau dashboards. This allows end users, such as LAPD analysts and citizens, to visualize and interact with model outputs, such as crime severity or reporting time, alongside historical data, supporting more informed and strategic decision-making.

7 Responsible AI

One important concern related to responsible AI in our project is the potential for the model to learn and reinforce existing biases present in the dataset. Since our features include sensitive attributes such as victim sex and descent code, there is a risk that the model may associate these variables with the severity or outcome of crimes in a way that reflects historical discrimination or systemic bias.

Security is a particularly sensitive topic in this context because it involves people's lives and the consequences of decisions driven by model predictions. As discussed in Section 4, questions such as whether it is better to allocate more resources unnecessarily or risk underestimating the severity of a crime are not purely technical, they are ethical dilemmas. These decisions require careful reflection before being operationalized in real-world policies or actions.

Additionally, privacy enhancing techniques, such as differential privacy, could be further implemented to ensure the privacy of the users to avoid membership inference attacks.

8 Conclusion

In this project, we worked with a complex and unstructured dataset derived from crime reports in a global and diverse city like Los Angeles. As expected, the dataset presented numerous challenges, including a high volume of missing data, a large number of features and inconsistencies typical of real-world public safety records.

Despite these difficulties, we chose to tackle this dataset and proposed predictive tasks that were both meaningful and technically demanding. We applied a range of classification and regression models, both supervised and unsupervised, and observed that different models were better suited to different types of problems. This highlighted the importance of model selection and adaptation to the specific nature of each task.

Due to the challenging nature of the dataset and the inherent complexity of the public safety domain, the models did not achieve very high performance scores. However, these results are consistent with the difficulty of the task, where noisy, incomplete and sensitive data make predictive modeling particularly demanding.

Throughout the project, we were able to apply and consolidate many key concepts covered in class, such as building supervised and unsupervised predictive models, implementing ensemble techniques, evaluating multiple metrics and comparing model performance. The results reinforced the idea that no single algorithm is universally best and that thoughtful experimentation is essential to finding the right fit for each problem.

Finally, we touched on one of the most recent and important topics discussed in class: Responsible AI. Our work showed how ethical considerations, such as bias and the consequences of automated decisions, must be integrated into any data-driven solution, especially in domains as sensitive as public safety.

9 Acknowledgments

ChatGPT-4o and ChatGPT-3.5 were utilized in almost all sections with the intention of polishing the English and also for creating code in general and especially for bug fixing.

References

[1] Kaggle Dataset: Crime Data from 2020 to Present, `https://www.kaggle.com/datasets/shubhamgupta012/crime-data-from-2020-to-present`

[2] Los Angeles Police Department: Description of Crime Data from 2020 to Present, `https://data.lacity.org/Public-Safety/Crime-Data-from-2020-to-Present/2nrs-mtv8/about_data`

[3] BI Dashboards, `https://public.tableau.com/views/LACrimeAnalysis_17421632867310/Victimscharacteristicspercrime?:language=pt-BR&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link`