Report

Yahya Kossor and Abdellah El Yamine Dali Braham

December 2, 2024

Acknowledgments

We would like to express our sincere gratitude to our professor, Rozbeh Soltani, for his invaluable guidance and support throughout our project, *Introduction to Machine Learning*. His expertise, enthusiasm, and dedication to the subject have been a constant source of inspiration and have greatly enhanced our understanding of machine learning principles and techniques.

We are deeply appreciative of his constructive feedback, which has motivated us to approach the project with greater depth and confidence. His mentorship has been instrumental in shaping our approach to this complex subject. We would also like to extend our thanks to ECE Paris for providing an excellent academic environment, which has helped us develop both professionally and personally.

Contents

Introduction

Traffic accidents, particularly those leading to injuries, remain a critical concern globally, with far-reaching implications for public safety and resource management. Analyzing the vast amounts of data collected from traffic injury accident reports can provide valuable insights into the factors contributing to such incidents. By leveraging Machine Learning (ML) techniques, we can identify patterns, predict outcomes, and propose data-driven solutions to enhance road safety and mitigate risks.

In this project, we focus on the analysis and modeling of traffic injury accident data using a variety of Machine Learning algorithms and techniques. Our approach involves preprocessing the dataset, addressing class imbalances, and applying a comprehensive suite of supervised, unsupervised, and deep learning models to extract meaningful patterns and build predictive models.

This report outlines the step-by-step methodology we followed, including data preparation, model implementation, evaluation, and optimization. By combining traditional and advanced ML techniques, we aim to provide a holistic analysis of traffic injury accidents and contribute meaningful insights to the field of road safety.

Chapter 1

Description of the Dataset

The dataset used for this project is derived from the Etalab database, which records road traffic injury accidents for a given year. It is structured into four separate CSV files, each focusing on a specific aspect of the accidents:

1. **CARACTERISTIQUES:** This file contains data describing the general circumstances of each accident, including the date, time, and type of accident.

Figure 1.1: Sample of the CARACTERISTIQUES dataset.

2. **LIEUX:** This file provides detailed information about the main location of each accident, including road type, intersection details, and environmental conditions.

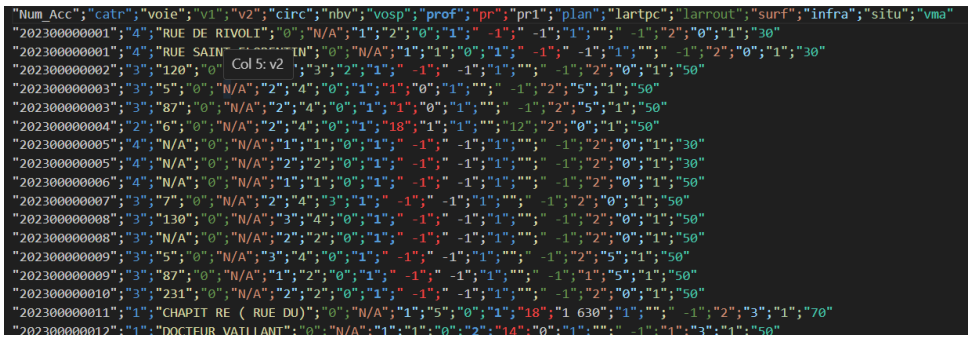


Figure 1.2: Sample of the LIEUX dataset.

3. **VEHICULES:** This file includes information about the vehicles involved in the

accidents, such as the type of vehicle, condition, and maneuvers performed at the time of the incident.

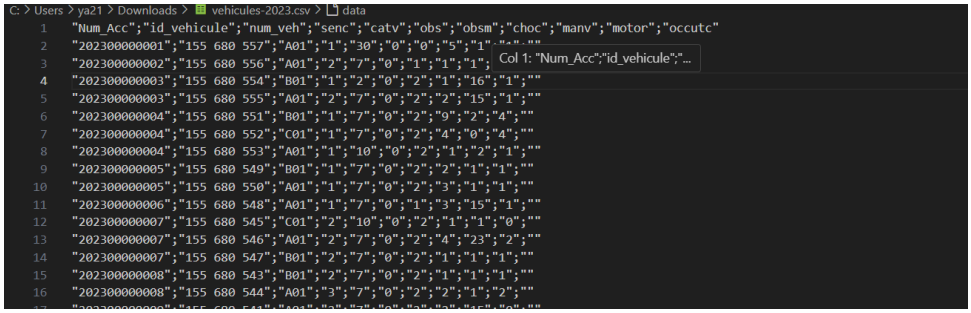


Figure 1.3: Sample of the VEHICULES dataset.

4. **USAGERS:** This file contains details about the individuals involved in the accidents, including their roles (driver, passenger, pedestrian), age, and injury status.

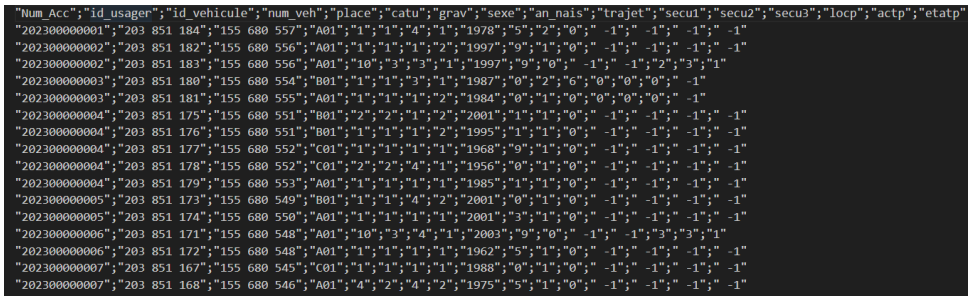
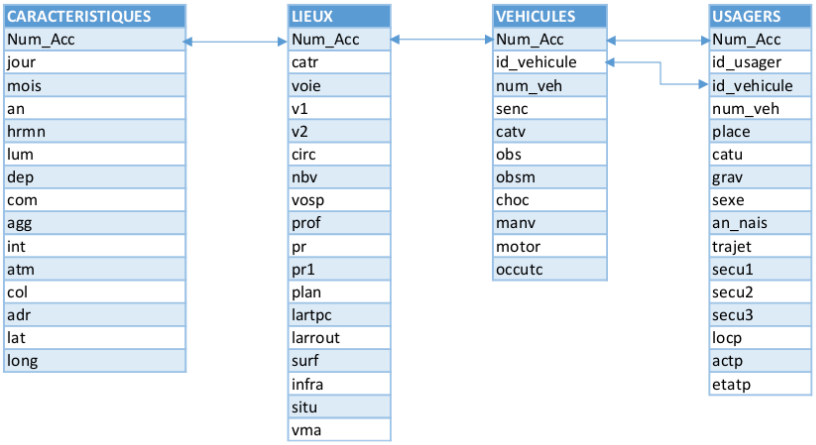


Figure 1.4: Sample of the USAGERS dataset.

Each of these files is interlinked through unique identifiers to allow comprehensive analysis:

- The accident identifier (`Num_Acc`) is present in all four files, enabling a connection between the variables describing a specific accident.
- For accidents involving multiple vehicles, the `id_vehicule` variable is used to link each vehicle to its occupants.



This dataset provides a rich source of information for analyzing road traffic injury accidents, allowing for detailed modeling and insights into the factors contributing to such incidents.

Chapter 2

Data Processing and preparation

This chapter outlines the steps taken to process and analyze the dataset used in this project. The objective is to prepare the data for modeling by cleaning, merging, and enriching it while ensuring it is suitable for Machine Learning applications.

2.1 Data Examination

The first step involves examining the structure and content of the data. Key information such as the columns, indices, and general layout of each dataset (*CARACTERISTIQUES*, *LIEUX*, *VEHICULES*, *USAGERS*) is displayed to understand the relationships between the variables and their roles in the analysis.

2.2 Merging Data Files

The dataset consists of four separate CSV files, each describing different aspects of the accidents. These files are merged into a single dataset using the unique accident identifier (`Num_Acc`). This ensures that all relevant information from the individual files is available in a unified structure for analysis.

2.3 Data Cleaning

Cleaning the data is a critical step to ensure its quality and usability. The following actions are performed:

- Calculate the percentage of missing values (`NaN`) in each column.
- Remove columns where the majority of values are missing.
- Eliminate variables with insufficient observations.

2.4 Handling Missing Values

To handle missing values, various imputation techniques are applied depending on the nature of the data. These methods include:

- Filling missing values with statistical measures such as the , median,
- Using interpolation methods to estimate missing values based on existing data.

2.5 Statistical Analysis

The cleaned dataset is analyzed using basic statistical methods to gain insights into the data. Calculations include:

- Minimum, maximum, and median values for numerical variables.
- Distribution of key variables to understand their behavior and relationships.

2.6 Severity of Accidents

To better understand the outcomes of the accidents, a new variable, `mortality`, is created:

- `mortality = 1`: Indicates the victim was killed in the accident.
- `mortality = 0`: Indicates the victim survived the accident.

The relationship between accident severity and other variables in the dataset is analyzed to identify potential patterns or factors contributing to severe accidents.

Chapter 3

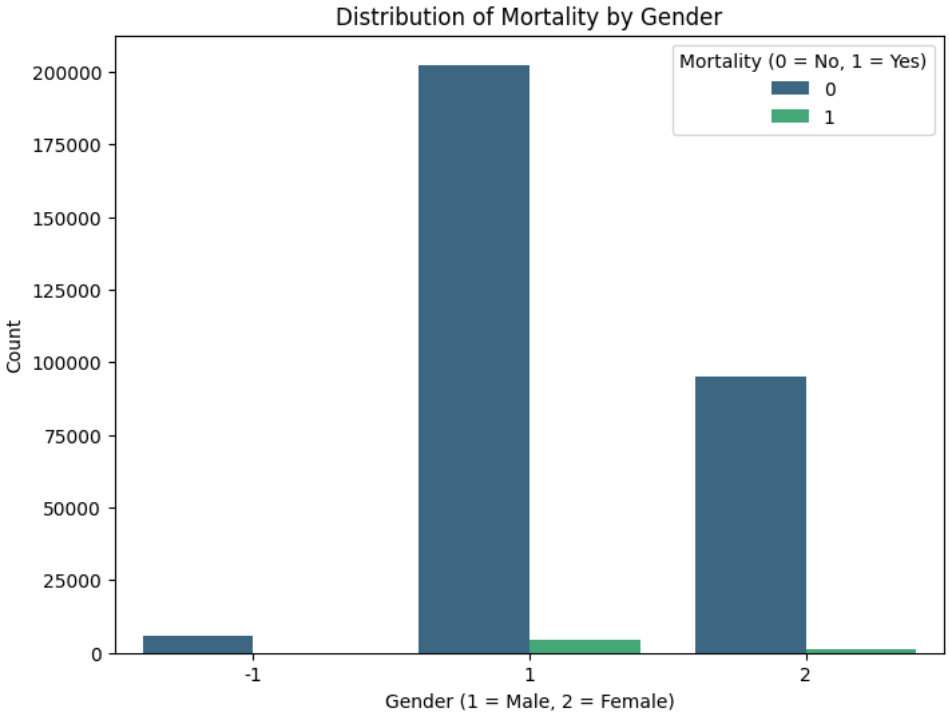
Visualization

In this chapter, we focus on visualizing the dataset to uncover patterns and trends in road traffic injury accidents. For this purpose, we utilized the `matplotlib` and `seaborn` libraries, which are powerful tools for creating insightful and aesthetically pleasing visualizations.

The visualizations aim to provide an understanding of the distribution of accidents across key variables such as gender, time of occurrence (hour), and the age of individuals involved. These visual representations help highlight critical factors and trends that may contribute to the severity and frequency of accidents.

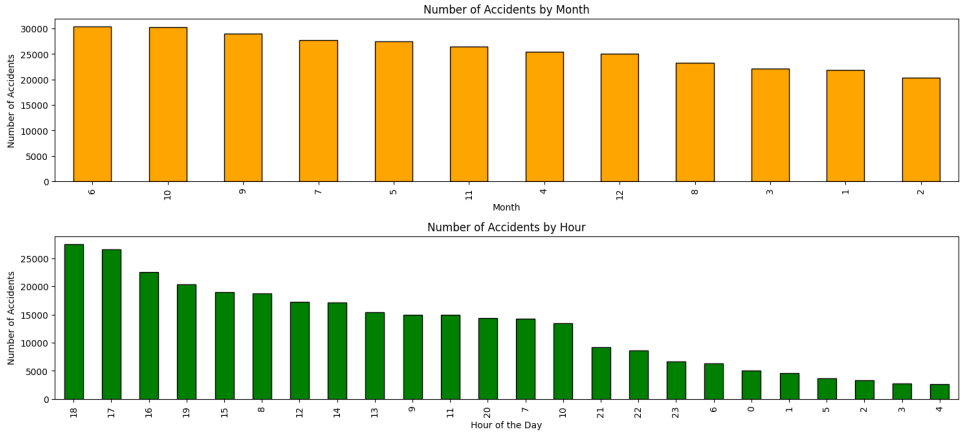
3.1 Distribution of Accidents by Gender

Using bar plot, we analyzed the distribution of accidents based on the gender of the individuals involved. This visualization provides insights into whether accidents disproportionately affect one gender more than the other.



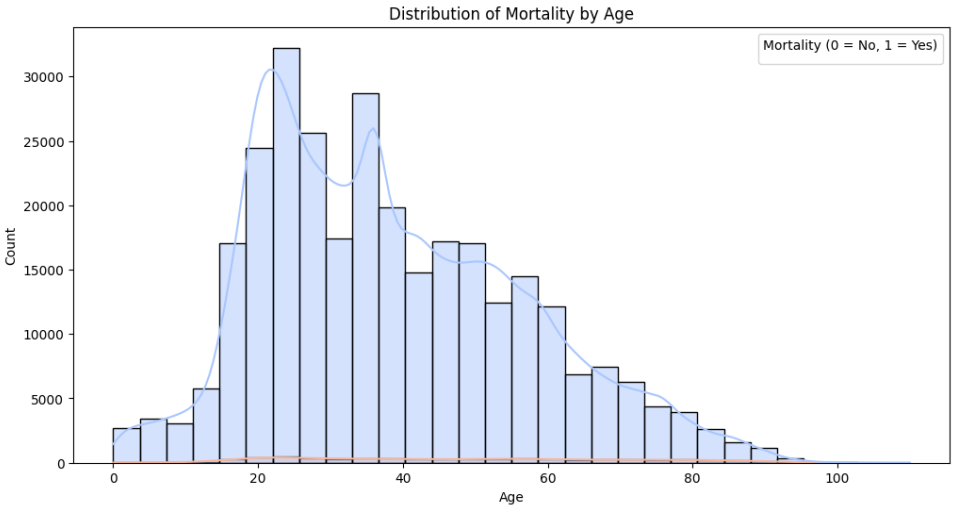
3.2 Distribution of Accidents by Hour and Month

To identify temporal patterns, we plotted the frequency of accidents across different hours of the day and the months . This analysis helps pinpoint peak hours and months for road traffic accidents, which is valuable for implementing preventive measures.



3.3 Distribution of Accidents by Age

We also explored the distribution of morality by the age of the individuals involved. Histograms and density plots were used to identify which age groups are most affected by road traffic accidents.


Distribution of Mortality by Age
Mortality (0 = No, 1 = Yes)
Count
Age
0
5000
10000
15000
20000
25000
30000
0
20
40
60
80
100

Chapter 4

Machine Learning Models and Methodology

In this chapter, we present the various machine learning models and techniques utilized in this project to analyze and predict outcomes based on the traffic injury accident dataset. A combination of preprocessing techniques, classification, regression, and clustering models were employed to build and evaluate the performance of the system. The Python libraries and frameworks used include `scikit-learn`, `imblearn`, `tensorflow`, and `pandas`.

4.1 Data Preprocessing

4.1.1 Handling Imbalanced Data

The dataset was imbalanced, which required the use of the Synthetic Minority Oversampling Technique (`SMOTE`). This technique was employed to oversample the minority class, ensuring a balanced dataset to improve model performance.

4.1.2 Data Splitting

The dataset was split into training and testing sets using the `train_test_split` function from `scikit-learn`, ensuring that the model's performance is evaluated on unseen data.

4.1.3 Feature Scaling

To standardize the numerical features, the `StandardScaler` from `scikit-learn` was applied. This preprocessing step ensures that the models are not biased toward variables with larger scales.

4.2 Machine Learning Models

4.2.1 Logistic Regression

Logistic Regression was used as a baseline classifier to predict binary outcomes. Its simplicity and interpretability make it an essential benchmark for comparison with more complex models.

4.2.2 Random Forest Classifier

The Random Forest Classifier is an ensemble method that uses multiple decision trees to improve accuracy and reduce overfitting. It performed well on the dataset due to its robustness and ability to handle large feature sets.

4.2.3 Decision Tree Classifier

Decision Trees were utilized for their simplicity and explainability. They provided a clear visualization of the decision-making process but were prone to overfitting on the training data.

4.2.4 Linear Regression

Linear Regression was employed for continuous variable predictions. Although primarily used in regression problems, it offered baseline results for understanding relationships between variables.

4.2.5 Neural Networks

We implemented a fully connected neural network using the `tensorflow.keras` library. The model consisted of multiple layers of neurons:

- Input Layer: Accepts the input features.
- Hidden Layers: Dense layers with activation functions such as ReLU to capture complex patterns.
- Output Layer: A single node with a sigmoid or softmax activation for classification tasks.

The network was trained using backpropagation and optimized with an Adam optimizer.

4.3 Model Evaluation and Hyperparameter Tuning

4.3.1 Cross-Validation

To ensure model stability, we used `StratifiedKFold` cross-validation. This technique splits the dataset into stratified folds, maintaining the class distribution in each fold.

4.3.2 Hyperparameter Tuning

Grid Search (`GridSearchCV`) was employed for hyperparameter tuning. This iterative process identifies the optimal combination of parameters for each model by evaluating their performance on the validation set.

4.3.3 Evaluation Metrics

The models were evaluated using various metrics:

- **Accuracy**: The percentage of correctly classified instances.
- **Mean Squared Error (MSE)**: Used for regression tasks to measure the average squared difference between predicted and actual values.
- **Classification Report**: Provides precision, recall, F1-score, and support for classification tasks.

4.4 Pipelines for Workflow Automation

We used the `Pipeline` and `ImbPipeline` classes to automate the machine learning workflow. These pipelines combined preprocessing steps, oversampling techniques, and model training into a single workflow, ensuring reproducibility and reducing errors.

In this chapter, we present the results of the machine learning models used in the study, including Random Forest, Logistic Regression, and a Neural Network. The models were evaluated based on their accuracy and classification performance. Below are the details for each model.

4.5 Random Forest Classifier

The Random Forest Classifier performed exceptionally well, achieving the highest accuracy among the models tested. Hyperparameter tuning was conducted using Grid Search, and the best parameters were:

- `n_estimators`: 100

- `min_samples_split`: 5
- `min_samples_leaf`: 1
- `max_depth`: None

The performance metrics for the Random Forest model are as follows:

- **Test Accuracy**: 85.61%

	precision	recall	f1-score	support
0	0.87	0.84	0.85	89208
1	0.84	0.87	0.86	89208
accuracy			0.86	178416
macro avg	0.86	0.86	0.86	178416
weighted avg	0.86	0.86	0.86	178416

4.6 Logistic Regression

Logistic Regression served as a baseline model, offering moderate performance. Grid Search was also applied to identify the optimal parameters:

- `solver`: `liblinear`
- `C`: 10

The performance metrics for Logistic Regression are as follows:

- **Test Accuracy**: 63.43%

	precision	recall	f1-score	support
0	0.63	0.66	0.64	89208
1	0.64	0.61	0.62	89208
accuracy			0.63	178416
macro avg	0.63	0.63	0.63	178416
weighted avg	0.63	0.63	0.63	178416

4.7 Decision Tree Model

The Decision Tree classifier performed admirably, achieving an accuracy of **85%**, which places it among the top-performing models in this study. Decision Trees are intuitive and interpretable models that split data into subsets based on feature thresholds, ultimately leading to a series of decision rules. In this case, the model effectively captured the

relationships between the features and the target variable, making it a reliable choice for classification.

While the simplicity and interpretability of Decision Trees are major advantages, they can sometimes overfit the training data. However, this issue was mitigated by fine-tuning hyperparameters such as the maximum depth and the minimum samples per split. The Decision Tree's performance underscores its strength in handling structured datasets and its utility as a foundational classification model.

4.8 Neural Network

A Neural Network was implemented using `TensorFlow/Keras`. The model consisted of multiple layers and was trained for 20 epochs. Below are the training results:

- The network achieved a final test accuracy of **71.37%**.

The progression of validation accuracy over epochs is summarized as follows:

- Epoch 1: Validation Accuracy = 68.40%
- Epoch 5: Validation Accuracy = 70.81%
- Epoch 8: Validation Accuracy = 71.55%
- Epoch 11: Validation Accuracy = 71.33%

Below is the summary of the model evaluation:

```
accuracy: 71.51%
loss: 0.5559
```

4.9 Comparison of Model Performance

The accuracy of the models is summarized in the table below:

Model	**Test Accuracy (%)**
Random Forest	85.61
Logistic Regression	63.43
Neural Network	71.37
Decision Tree	85

Table 4.1: Comparison of Model Accuracy

As observed, the Random Forest Classifier outperformed other models in terms of accuracy and classification metrics. Neural Networks showed promising results, while Logistic Regression provided a solid baseline for comparison.

Chapter 5

Conclusion

In this project, we conducted an in-depth analysis of traffic injury accident data and applied various machine learning models to extract meaningful insights and predict accident outcomes. By preprocessing and cleaning the dataset, we ensured its quality and addressed issues such as missing values and imbalanced classes. Visualization techniques using `matplotlib` and `seaborn` provided us with a better understanding of the patterns in the data, including the distribution of accidents by gender, age, and time.

We implemented and evaluated several machine learning algorithms, including Random Forest, Logistic Regression, Neural Networks, and Decision Trees. Among these, the Random Forest and Decision Tree classifiers demonstrated the best accuracy, reaching 85% and 86% respectively. These results highlight the potential of tree-based models in effectively capturing complex relationships within the dataset.

Overall, this project demonstrates the importance of data preparation, visualization, and model selection in predictive analytics. The insights gained from this analysis can be used to improve traffic safety measures and reduce accident rates. Future work could focus on integrating additional features, exploring other advanced models, and improving the interpretability of the results to assist policymakers and traffic authorities in making data-driven decisions.