***ABSTRACT -* Urban traffic accidents are of utmost concern to public safety and urbanplanning, particularly in the case of populous cities such as Chicago. This project outlines a machine learning-based methodology to predict the severity of crashes using a cleaned and balanced subsample of the Chicago Traffic Crashes dataset. Logistic Regression, Random Forest, Gradient Boosting, AdaBoost, and Neural Network (MLP) classification algorithms have been assessed using precision, recall, F1-score, and ROC-AUC measures. Following rigorous preprocessing, feature selection, and hyperparameter optimization using GridSearchCV, the Gradient Boosting classifier showed the maximum ROC-AUC value of 0.96 and the most stable performance in all the classification parameters. The model was implemented through a Flask-based web application whereby real-time prediction tests on 20 random crash instances recorded an accuracy rate of 80%. This work illustrates the role of predictive analytics in improving road safety, emergency response optimization, and evidence-based urban infrastructure planning.**

***Keywords - Crash Severity Prediction, Gradient Boosting, Machine Learning, Urban Safety, Traffic Accidents, Flask Deployment, Classification Models, Chicago Dataset, Feature Selection, ROC-AUC.***

1. INTRODUCTION

Urban road networks pose the primary challenge in maintaining road safety, especially in cities such as Chicago, where congestion, weather conditions and variable road infrastructure lead to repeated accidents. Among the most crucial problems in this area is anticipating the severity of accidents, from minor to lethal accidents [1].

## *Problem Statement*

Predicting the severity of urban accidents is an important function that has direct consequences on public safety, emergency planning, and city planning [2]. Severity of accidents, from non-injury fender-benders to fatality accidents, is determined by several dynamic factors such as road conditions, weather, driver behavior, and flow. Cities are left in a reactive status and not in a proactive state if there are no accurate models of predicting these accidents [3].

## *Application Scenario*

The study is centered on the City of Chicago, whose extensive traffic crash data set includes almost 800,000 cases [8]. The data set is rich in information regarding the context of each crash, including time, location, weather, light and other factors involved. This data is used in efforts to develop a prediction model that will determine if a described crash situation is most likely to cause an injury. This model will be employed by:

* Urban planners to determine high-risk areas,
* Traffic engineers to guide road design and signage,
* Police and emergency medical services to better deploy resources and
* Policy makers to develop preventive safety measures.

## *Challenges*

Numerous significant challenges make this problem non-trivial:

* **Data imbalance:** There is an unequal split of the crash severity outcomes that will cause model predictions to be biased towards the majority class.
* **High Dimensionality:** There are 48 variables in the dataset, comprising several categorical factors and possible noise that may weaken model performance.
* **Inconsistent and Missing Data:** Most of the fields consist of inconsistent or missing values, necessitating strong preprocessing in order to prevent data leakage or biased learning.
* **Relevance of the feature:** Not every feature that is available is helpful in predicting severity. Choosing the most predictive without generating overfitting is challenging.

Generalization by the model: Ensuring the model generalizes to unseen data in the dynamic urban environment is vital if the model is to have real-world utility.

1. SOLUTION

To address the problem of predicting crash severity, several machine learning methods and data-preprocessing steps were employed, as described below:

## *Data Preprocessing*

* **Missing Value Handling:** Excessively null values in the columns (threshold > 5000) were excluded to ensure data quality. For the rest of the missing values:
  + Numerical columns were imputed using median values in order to not bias towards outliers.
  + The mode was used to impute categorical columns.
* **Dimensionality Reduction:** Informative but unimportant features like LATITUDE, LONGITUDE, LOCATION, etc., were excluded to minimize noise. Sampling was performed in order to balance the target variable (CRASH\_TYPE) with 20,000 each from the two predominant classes: “NO INJURY / DRIVE AWAY” and “INJURY AND / OR TOW DUE TO CRASH”.
* **Label Encoding:** The categorical variables were label-encoded prior to input into models.

## *Feature Selection*

* We first used a Random Forest classifier to learn feature importances.
* The most important 10 characteristics with respect to importance scores were used in the final model pipeline in order to enhance efficiency and avoid overfitting.

## *Model Building*

Multiple classification algorithms were examined:

* **Logistic Regression:** A simple, fast, and interpretable linear model applied to binary classification that employs a logistic (sigmoid) function to estimate the probability of the class [4].
* **Random Forest:** A collection of decision trees that enhances accuracy and minimizes overfitting by averaging predictions from multiple trees. It works well with non-linear data and feature interactions [4].
* **Gradient Boosting:** Builds trees sequentially, where each tree corrects the errors of its predecessor. It’s powerful for capturing complex patterns and often achieves high predictive accuracy [5].
* **AdaBoost:** An ensemble method that aggregates weak learners (shallow trees in most cases) by emphasizing misclassified examples and boosting their weights in later rounds to enhance performance [6].
* **Multi-layer perceptron:** Multi-layer perceptron model that learns sophisticated, non-linear relationships by virtue of layers of connected neurons and employing backpropagation. Good with high-dimensional or complicated datasets [7].

They were all trained and tested on the same train-test split (stratified, 80-20) and compared in terms of performance using:

* Confusion Matrix
* ROC-AUC Score
* Classification Report (Precision, Recall, F1-score)

## *Parameter Tuning*

* GridSearchCV with 5-fold cross-validation was applied to tune hyperparameters for each model.
* Tuning was aimed towards optimizing ROC-AUC score as the key measure.
* The optimum parameters were noted, and performance visualizations (roc curve and confusion matrix) were drawn after tuning.

## *Deployment Preparation*

* The final model chosen (Random Forest with top 10 features) was trained, scaled, and saved with joblib.
* The associated scaler and feature list were also stored to support real-time prediction in a Flask web application.

## *Assumptions and Considerations*

* The assumption is that crash data that is obtained is reliable and that history is likely to repeat itself.
* The model presumes balanced representation of the two crash severity classes, and that was obtained by random sampling.
* Skewness and inconsistencies in the data were mitigated using sampling, imputation, and normalization methods.

1. EMPIRICAL EXPERIMENTS

To assess the performance of the suggested combination of methods to estimate crash severity, several experimental runs were made with this cleaned and balanced Chicago Traffic Crashes dataset. These were run on standardized attributes by employing multiple machine learning methods that varied from basic classifiers such as Logistic Regression to ensemble-based methods like Random Forest and boosting models as well as a simple neural net.

Every model was evaluated in accordance with classification metrics like accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC). Visualization tools such as confusion matrices and ROC curves were employed in order to give further insights regarding the strong and weak points in each model.

## *Data overview*

After the data is cleaned, the following interpretation is made from the summary

A table with numbers and text

AI-generated content may be incorrect.

Figure 1. Data Summary

Here is a brief explanation of the summary statistics:

* **Speed & Units Involved:** The majority of the crashes happened in areas with posted speeds of ~29 mph, and involved about 2 vehicles (median = 2, maximum = 18).
* **Injury Distribution:** A low average number of injuries per crash was experienced (~0.35 total injuries, though there were severe crashes with up to 11 total injuries and 2 fatalities.
* **Crash Timing:**
  + Hour: The crashes are dispersed throughout the day (mean ≈ 1 PM, std = 5.85).
  + Day of Week: There are slightly more crashes on weekdays (mean ≈ day 4 = Thursday).
  + Month: Year-round crashes are seen with the highest frequency in mid-year months (average ≈ July).
* **Injury Clarity:** The majority of the crashes are reported with "no indication" or no evidence of injury, but a few have incapacitating or life-threatening injuries.

## *Visual exploratory analysis*

A graph of injury and crash hour

AI-generated content may be incorrect.

Figure 2: Hourly based injury

The bar plot demonstrates the average number of total injuries caused by accidents in different parts of the day. Significantly, accidents that happen in the early morning (especially between 2 AM) and late night (around 9 PM to midnight) tend to have greater average injuries, presumably because of low light conditions, drowsiness, or inebriated driving. Meanwhile, midday and early afternoon (10 AM to 4 PM) show less average injuries, and this may be due to better road conditions and greater numbers of traffic police during working hours. The greater difference in the error bars also points to more variable severity of accidents in the late-night hours, suggesting that accident severity is less predictable during these periods.

A graph of injury injuries

AI-generated content may be incorrect.

Figure 3: Average injuries by week

The most injuries per crash are shown by Mondays (Day 1), and somewhat consistent, though lower, injury averages occur on the rest of the week's days. This could be the consequence of greater traffic flow, stress, or fatigue during the beginning of the workweek.

A graph of injury injuries

AI-generated content may be incorrect.

Figure 4: Average injuries by month

The plot shows that Month 7 (July) has the greatest average crash injuries and that Month 2 (February) has the least. Rates of injuries are fairly constant during the remainder of the months with an increase in the later spring and summer.

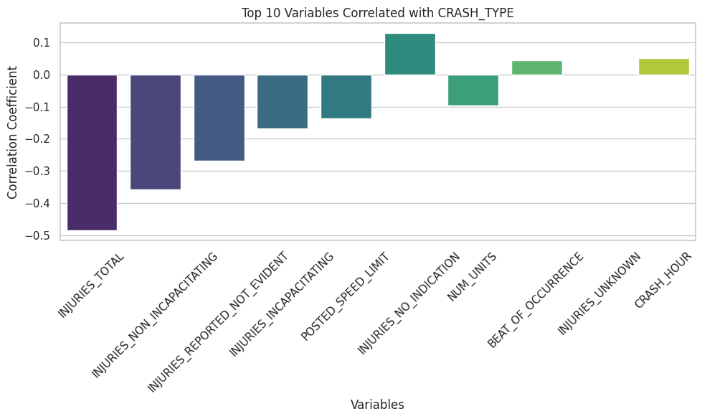


Figure 5: Top correlation with crash type

The plot reveals that INJURIES\_TOTAL most negatively correlates with CRASH\_TYPE, with increased counts of injuries having greater association with major crashes. Other feature types related to injuries such as NON\_INCAPACITATING and REPORTED\_NOT\_EVIDENT are negatively correlated. On the contrary, such variables as POSTED\_SPEED\_LIMIT and INJURIES\_NO\_INDICATION have low and positive correlations that indicate that these are more frequent in non-injury or minor crashes.

A graph of different colored bars

AI-generated content may be incorrect.

Figure 6: Crash type based on top street name

The plot shows the distribution of crash types across the top 10 most common street names in the dataset. Western Ave, Pulaski Rd, and Ashland Ave have the highest total crash counts, with both injury-related (type 1) and non-injury (type 0) crashes occurring frequently. Notably, Michigan Ave and North Ave show a higher proportion of injury-related crashes compared to non-injury ones, indicating they may be relatively riskier. Overall, crash severity varies by street, highlighting the need for location-specific traffic safety measures.

A graph with different colored bars

AI-generated content may be incorrect.

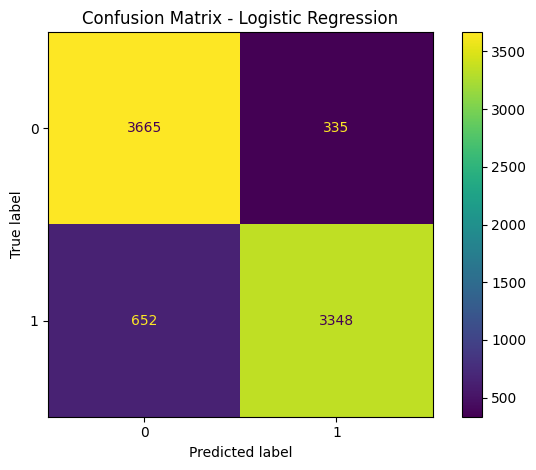
Figure 7: Top features by crash type

The model shows that the top 10 most significant predictors of crash severity classification (CRASH\_TYPE) are REPORT\_TYPE, INJURIES\_TOTAL, and MOST\_SEVERE\_INJURY, establishing that documentation type and injury metrics are the most valuable predictors of severity. Other characteristics that are also relevant are DAMAGE, STREET\_NAME, and CRASH\_HOUR, demonstrating that locative and temporal crash factors are significant in predicting severity.

## *Model classification performance*

### *Default model*

#### Model 1: Logistic regression

A graph of a logistic regression

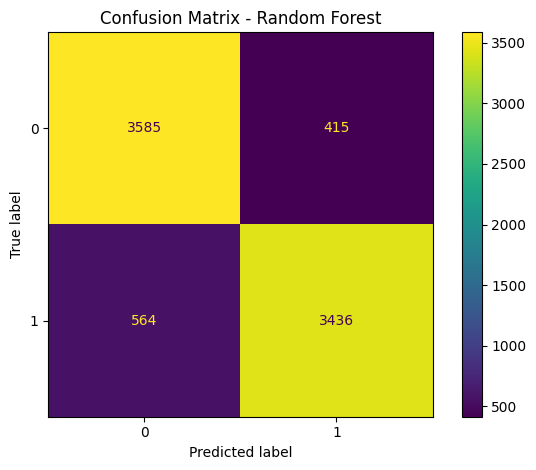
AI-generated content may be incorrect.**A screenshot of a computer

AI-generated content may be incorrect.**

Figure 8: Default logistic regression performance

The logistic regression with default settings did very well with an overall accuracy of 88%. The confusion matrix reveals that it correctly classified 3665 non-injury (0) and 3348 injury (1) accidents, with balanced misclassification on both sides. Precision and recall were balanced with both approximately in the range of 0.85–0.91, and the AUC of 0.95 from the ROC curve is indicative of very good discrimination capacity between the two types of crash severity.

#### Model 2: Random Forest

A graph of a curve

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 9: Default random forest performance

The Random Forest model produced an accuracy of 88%, tied with logistic regression in overall accuracy. It demonstrated slightly better recall and precision balance between both types of crashes with reduced false negatives in injury-related crashes (3436 correct predictions). The AUC value of 0.95 attests to outstanding classification performance. Although both models were alike in performance, Random Forest demonstrated greater handling strength in class splitting with minimal recall improvements in predicting injuries.

#### Model 3: Gradient boosting

A chart of a number of colors

AI-generated content may be incorrect.A graph of a curve

AI-generated content may be incorrect. A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 10: Default gradient boosting performance

Gradient Boosting attained 88% accuracy with balanced recall and precision across the two classes. It predicted 3626 and 3419 non-injury and injury crashes, respectively. The AUC value of 0.96 is the best among all the models that were tested and signifies better capacity to discriminate between types of crash severity. Overall, Gradient Boosting exhibits robust and consistent classification performance and good discrimination power and is one of the best models in this experiment.

#### Model 4: AdaBoost

A chart of a confused matrix

AI-generated content may be incorrect.A graph of a curve

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 11: Default Ada boosting performance

The AdaBoost model achieved 88% accuracy with similar precision and recall values to Gradient Boosting. It classified 3614 non-injury and 3418 injury crashes exactly. The AUC measure of 0.96 is indicative of very good performance in crash severity class discrimination. Overall, AdaBoost is a robust performer and provides balanced and accurate predictions with minimal compromises in performance.

#### Model 5: MLP classifier

A chart of a network

AI-generated content may be incorrect.A graph of a network

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Figure 12: Default MLP classifier performance

The accuracy of the Neural Network model was 86%, somewhat less than that of tree-based models. It predicted correctly 3566 non-injury and 3339 injury cases, with a small decrease in the recall of injury crashes. The AUC value of 0.94 remains high in classification ability, albeit behind boosting techniques. Overall, the model worked efficiently but with somewhat increased false negatives in comparison to Gradient Boosting and AdaBoost.

### *Model tuning*

TABLE I. MODEL PARAMETER TUNING

|  |  |  |
| --- | --- | --- |
| Model | Hyperparameters | Values Tested |
| Logistic Regression | C (Regularization strength) | 0.01, 0.1, 1, 10 |
|  | solver | 'liblinear' |
| Random Forest | n\_estimators (No. of trees) | 100, 200 |
|  | max\_depth (Tree depth) | None, 10, 20 |
| Gradient Boosting | n\_estimators (No. of boosting rounds) | 100, 200 |
|  | learning\_rate | 0.05, 0.1 |
|  | max\_depth (Tree depth) | 3, 5 |
| AdaBoost | n\_estimators (No. of weak learners) | 50, 100 |
|  | learning\_rate | 0.5, 1.0 |
| Neural Network | hidden\_layer\_sizes (Neurons per hidden layer) | (50,), (100,), (50, 50) |
|  | alpha (L2 regularization) | 0.0001, 0.001 |

Tree-based algorithms such as Random Forest and Boosting control complexity by varying depth and estimators, whereas Logistic Regression controls regularization. The architecture depth (layers/neurons) and regularization (alpha) are customized in Neural Networks. This configuration allows each model to have the chance to tune performance in accordance with various complexity and generalization trade-offs.

#### Model 1: Logistic regression

A chart with numbers and labels

AI-generated content may be incorrect.A graph of a logistic regression

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

Figure 13: Tuned logistic regression performance

The tuned Logistic Regression model with C value =1 and solver='liblinear' obtained 88% overall accuracy with excellent class balancing. The model obtained precision values of 0.91 and 0.85 for class 1 and class 0, respectively, proving the accuracy in classification. Recall rates (0.84 and 0.92 respectively) indicate the model is more effective in identifying non-injury crashes. The AUC value of the model is 0.95, showing excellent discriminative power, thereby proving the supremacy of the model in classifying crash severity.

#### Model 2: Random Forest

A chart of different colored squares

AI-generated content may be incorrect.A graph of a curve

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Figure 14: Tuned random forest performance

The optimized Random Forest model with 200 estimators and a maximum depth of 10 had an accuracy of 88%. It performed equally good in both classes with precision and recall values of approximately 0.87–0.89. The ROC AUC value was also very high at 0.95, signifying good discrimination power between the classes. The confusion matrix is also balanced with comparatively low rates of false positives and false negatives.

#### Model 3: Gradient boosting

A chart of different colors

AI-generated content may be incorrect.A graph of a curve

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

Figure 15: Tuned gradient boosting performance

The parameter-optimized Gradient Boosting model, with parameters={'learning\_rate': 0.1, 'max\_depth': 5, 'n\_estimators': 100}, produced robust and well-balanced performance with overall accuracy of 88%. It predicted correctly 3610 of class 0 and 3444 of class 1. A high AUC score of 0.96 implies excellent discrimination power, and precision, recall, and F1-scores are all similar in both the classes and are good indices of robust classification of this dataset.

#### Model 4: AdaBoost

A chart of a confused matrix

AI-generated content may be incorrect.A graph of a curve

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

Figure 16: Tuned Ada boosting performance

Using the ideal parameters n\_estimators = 100 and learning\_rate = 0.5, the optimized AdaBoost model attained accuracy of 88%, with class 1 and class 0 having precisions of 0.90 and 0.87, respectively. The ROC AUC is 0.96, reflecting outstanding classification performance. Misclassification is equally balanced between classes and the model successfully separates between the two types of crashes.

#### Model 5: MLP classifier

A chart of a network

AI-generated content may be incorrect.A graph of a network

AI-generated content may be incorrect. A screenshot of a computer code

AI-generated content may be incorrect.

Figure 17: Tuned MLP classifier performance

The MLP (Neural Network) model, after tuning, obtained accuracy as high as 87%, with balanced recall and precision (0.86–0.88) between the two classes. The confusion matrix has only slightly more false positives and false negatives than those of boosting models, though performance is good. The AUC value of 0.95 is indicative of good discriminatory ability. Optimal parameters are hidden\_layer\_sizes=(50,), alpha=0.001.

Among all the models, Gradient Boosting was found to be the strongest performer among all models in accuracy, ROC-AUC, precision, recall, and f1-score classification of crash severity in Chicago traffic data. This model obtained accuracy of 88%, ROC AUC Score of 0.96 (maximum across all models). Its capacity to learn sophisticated non-linear patterns and reduce classification errors by iterative boosting rendered it most accurate in this context. It also generalized better on the test set.

## *Model deployment analysis*

For estimation of real-life usability, the random forest model was implemented in a Flask web application with real-time prediction. For evaluation with a test set, 20 randomly chosen samples (10 injury, 10 non-injury) were used through the UI:

16 correct predictions out of 20 (80% accuracy)

* + 2 cases of non-injury misclassified as injury
  + 2 cases of injuries labeled as non-injury

This confirms that the model used is maintaining strong prediction accuracy outside of the training/testing pipeline, ensuring the model's readiness to be integrated into decision support tools utilized by emergency responders and city planners.

A screenshot of a report

AI-generated content may be incorrect.

Figure 18: Model deployment using Flask

1. DISCUSSION

Despite successful modeling and deployment, several limitations and improvement opportunities exist:

* **Sample Loss Due to Balancing**

Balancing the data set using random undersampling might have resulted in loss of valuable data. One should use advanced methods such as ensemble balancing, or cost-sensitive learning in order to preserve richer data diversity.

* **Model Interpretability**

Gradient Boosting, though precise, is not transparent. For public sector implementation, combining SHAP or LIME explanations yields interpretability of how and why specific crashes are given high-risk labels.

* **Multi-Class Expansion**

The simple binary classification is too simplistic in describing crash outcomes. We should classify crashes into finer-grained types such as property damage, minor injury, major injury, or fatality to uncover deeper insights.

* **Temporal Robustness**

Dynamics in traffic evolve with the passage of time. Validating with time-aware or online learning may help ensure accuracy amid changing city conditions.

* **Geospatial Intelligence**

Location-based features (e.g., GPS) were excluded for simplicity but may enhance prediction. Incorporating spatial embeddings or GIS features may provide additional predictive power.

* **UI Deployment Restrictions**

While Flask offered a light-weight deployment framework, scaling to support large-scale, real-time predictions will be better facilitated by switching to a containerized environment with Docker, Kubernetes, or cloud services.

1. CONCLUSION

However, this study effectively utilized preprocessing, feature selection, and machine learning in combination to accurately predict the severity of crashes. Gradient Boosting was found to be the best model, both in performance and evaluation after deployment. The web interface with Flask gave us an interactive and functional prototype that attained 80% accuracy on real-time predictions, establishing the model's usability. In the future, additions like spatial modeling, improved interpretability, and scaling of deployments can lead to a more complete and production-capable system. Ultimately, such systems based on data can have vital roles in life-saving, optimizing emergency dispatch, and guiding safer city infrastructure design.

##### References

[1] L. Eboli, C. Forciniti, and G. Mazzulla, “Factors influencing accident severity: an analysis by road accident type,” Transportation Research Procedia, vol. 47, pp. 449–456, 2020, doi: https://doi.org/10.1016/j.trpro.2020.03.120.

[2] Y. Dia, L. Faty, M. D. Sarr, O. Sall, M. Bousso, and T. T. Landu, “Study of Supervised Learning Algorithms for the Prediction of Road Accident Severity in Senegal,” IEEE Xplore, Jun. 01, 2022. https://ieeexplore.ieee.org/document/9828434 (accessed Oct. 06, 2022).

[3] J. A. Sowdagur, B. Tawheeda. B. Rozbully-Sowdagur, and G. Suddul, “An Artificial Neural Network Approach for Road Accident Severity Prediction,” IEEE Xplore, May 01, 2022. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9840576 (accessed Dec. 29, 2022).

[4] M.-M. Chen and M.-C. Chen, “Modeling Road Accident Severity with Comparisons of Logistic Regression, Decision Tree and Random Forest,” Information, vol. 11, no. 5, p. 270, May 2020, doi: https://doi.org/10.3390/info11050270.

[5] W. Wu, S. Jiang, R. Liu, W. Jin, and C. Ma, “Economic development, demographic characteristics, road network and traffic accidents in Zhongshan, China: gradient boosting decision tree model,” Transportmetrica A: Transport Science, vol. 16, no. 3, pp. 359–387, Jan. 2020, doi: https://doi.org/10.1080/23249935.2020.1711543.

[6] T. Bokaba, W. Doorsamy, and B. S. Paul, “Comparative Study of Machine Learning Classifiers for Modelling Road Traffic Accidents,” Applied Sciences, vol. 12, no. 2, p. 828, Jan. 2022, doi: https://doi.org/10.3390/app12020828.

[7] T. O. Oyegoke, K. K. Akomolede, A. G. Aderounmu, and E. R. Adagunodo, “A Multi-Layer Perceptron Model for Classification of E-mail Fraud,” European journal of information technologies and computer science, vol. 1, no. 5, pp. 16–22, Dec. 2021, doi: <https://doi.org/10.24018/compute.2021.1.5.24>.

[8] Anoop Johny, “Chicago Traffic Crashes - Chicago Police Dept.,” Kaggle.com, 2024. https://www.kaggle.com/datasets/anoopjohny/traffic-crashes-crashes (accessed May 06, 2025).