Google Collab Link: Traffic Accident Severity Prediction

Table of Contents

1.0 Introduction

1.1 Background of Study

Traffic accidents, especially in urban areas, present hazard threats to cities, particularly in low- and middle-income countries. The World Health Organization estimates that about 1.3 million annual deaths from road traffic accidents occur, with non-fatal injuries ranging from 20 to 50 million people (World Health Organization, 2023). In particular, Addis Ababa, the capital of Ethiopia, is being affected by rapid urbanization, increasing number of vehicles, and insufficient infrastructure. The roads are subject to a high volume of vehicular movement daily, leading to accidents of different magnitudes in their everyday ramifications. Predicting the severity of those traffic accidents is important to enhancing urban safety, allowing for better planning by the authorities, allocating and deploying resources, enforcing the law, and implementing safety measures. Traditional approaches to road safety such as awareness campaigns and manual accident reporting systems have been found ineffective in addressing the unique complexity and unpredictability of traffic accidents (Kushwaha and Abirami, 2022). Thus, the use of data mining and machine learning approaches to distill meaningful patterns from road traffic data to develop predictive models for accident severity classification would be of much interest.

1.2 Project Goal

Road traffic accidents in Addis Ababa constitute a growing public safety challenge; however, data-driven systems for predicting accidents' severity are still absent. Without accurate forecasts being carried out in real time concerning severity, interventions in emergency response, urban planning, and road safety tended to have a reactive rather than proactive nature. Although data are collected on accidents, they remain mostly underutilized due to a lack of any analytical framework; in turn, this denies opportunities for revealing critical risk patterns. Thus, through the proposed project, a machine learning model shall be generated for classifying accident severity into three classes: slight, serious, and fatal, and thus yield actionable insights for stakeholders.

1. Developing an accurate machine-learning model that classifies road accident severity for Addis Ababa traffic data into slight, serious, and fatal.

2. Identifying with strong association those risk factors that would be critical in analyzing severe accidents through data analysis and evaluation of feature importance.

3. Delivering recommendations that would lead to action for emergency services and urban planners with regard to countermeasures against accidents and intervention strategies.

2.0 Overview of the dataset

2.1 Description of the dataset

This project uses a Kaggle dataset, "Road Traffic Accidents," from Addis Ababa, Ethiopia, to predict the severity of traffic accidents based on environmental and behavioral factors, with the following important fields:

- **Accident ID**: Unique identifier for each accident record.
- **Date / Time**: Timestamp indicating when the accident occurred.
- **Location**: Specific area within Addis Ababa where the accident took place.
- **Weather**: Weather conditions at the time of the accident (e.g., Clear, Rainy).
- **Road Surface**: Type of road surface, such as dry or wet.
- **Lighting**: Lighting condition during the accident (e.g., daylight, nighttime).
- **Vehicle Type**: Type of vehicle involved (e.g., car, bus, motorcycle).
- **Cause**: Primary contributing factor to the accident (e.g., speeding, distraction).
- **Casualties**: Number of people injured in the accident.
- **Severity**: Level of accident severity – categorized as Slight, Serious, or Fatal.

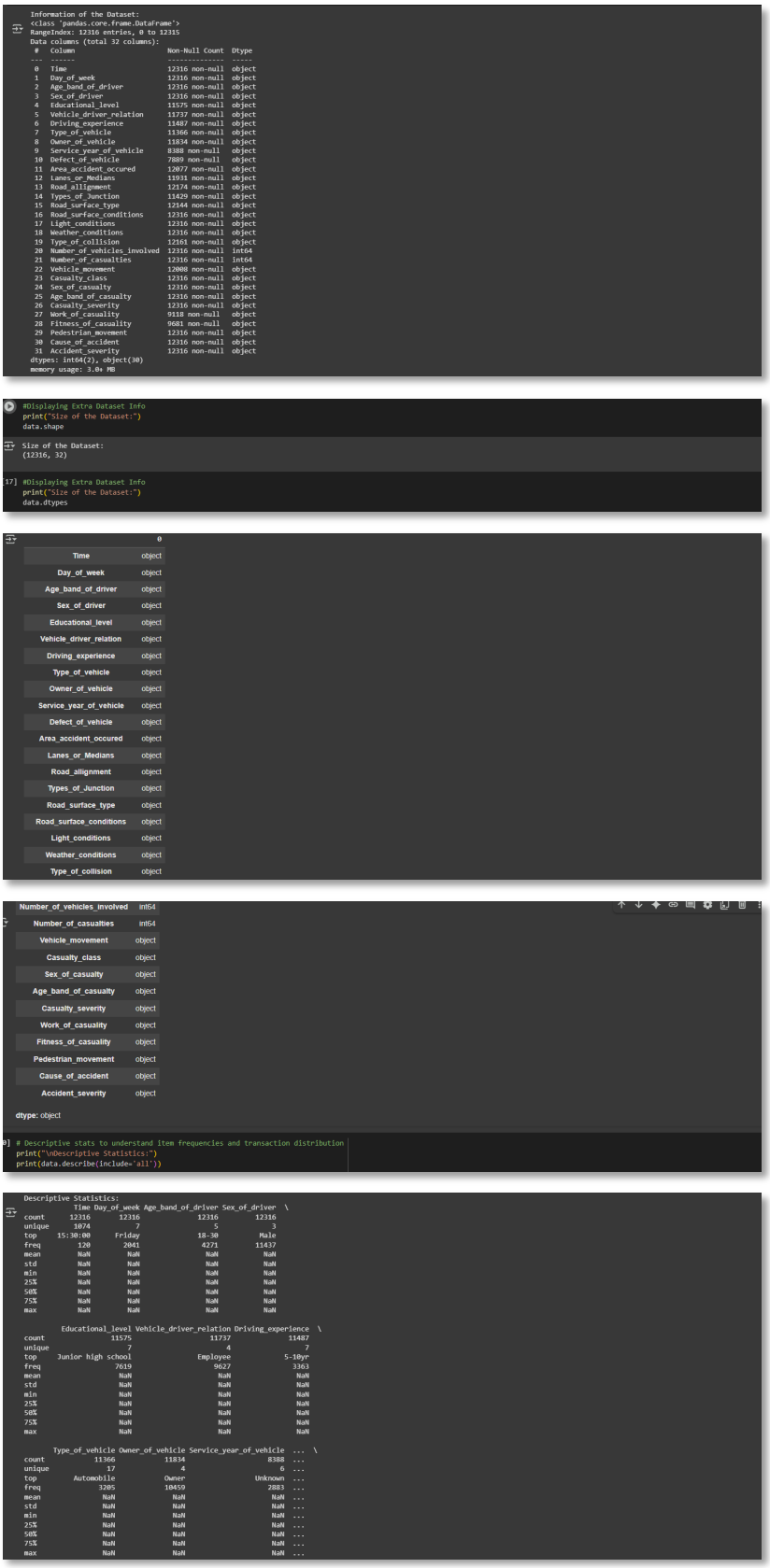
The dataset, which includes 32,000+ records, represents a single traffic accident case and contains both categorical and numerical attributes. The dataset was originally published to support research in road safety, policy development, and urban transportation planning. Understanding the factors most strongly associated with fatal, serious, or slight accidents can help authorities implement more effective preventive strategies.

```
[6] # Load data
    data = pd.read_csv("RTA_Dataset.csv")

    # Displaying the first few rows
    data.head()
```

	Time	Day_of_week	Age_band_of_driver	Sex_of_driver	Educational_level	Vehicle_driver_relation	Driving_experience	Type_of_vehicle	Owner_of_vehicle	Service_year_of_vehicle
0	17:02:00	Monday	18-30	Male	Above high school	Employee	1-2yr	Automobile	Owner	Above 10yr
1	17:02:00	Monday	31-50	Male	Junior high school	Employee	Above 10yr	Public (> 45 seats)	Owner	5-10yrs
2	17:02:00	Monday	18-30	Male	Junior high school	Employee	1-2yr	Lorry (41?100Q)	Owner	NaN
3	1:06:00	Sunday	18-30	Male	Junior high school	Employee	5-10yr	Public (> 45 seats)	Governmental	NaN
4	1:06:00	Sunday	18-30	Male	Junior high school	Employee	2-5yr	NaN	Owner	5-10yrs

5 rows × 32 columns

```
[7] # Displaying basic dataset info
    print("Information of the Dataset:")
    data.info()
```



```
Information of the Dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12316 entries, 0 to 12315
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Time                         12316 non-null  object
 1   Day_of_week                  12316 non-null  object
 2   Age_band_of_driver           12316 non-null  object
 3   Sex_of_driver                12316 non-null  object
 4   Educational_level            11575 non-null  object
 5   Vehicle_driver_relation      11737 non-null  object
 6   Driving_experience           11487 non-null  object
 7   Type_of_vehicle              11366 non-null  object
 8   Owner_of_vehicle             11834 non-null  object
 9   Service_year_of_vehicle      8388 non-null   object
 10  Defect_of_vehicle            7889 non-null   object
 11  Area_accident_occured        12077 non-null  object
 12  Lanes_or_Medians             11931 non-null  object
 13  Road_allignment              12174 non-null  object
 14  Types_of_Junction            11429 non-null  object
 15  Road_surface_type            12144 non-null  object
 16  Road_surface_conditions      12316 non-null  object
 17  Light_conditions             12316 non-null  object
 18  Weather_conditions           12316 non-null  object
 19  Type_of_collision            12161 non-null  object
 20  Number_of_vehicles_involved  12316 non-null  int64
 21  Number_of_casualties         12316 non-null  int64
 22  Vehicle_movement             12008 non-null  object
 23  Casualty_class               12316 non-null  object
 24  Sex_of_casualty              12316 non-null  object
 25  Age_band_of_casualty         12316 non-null  object
 26  Casualty_severity            12316 non-null  object
 27  Work_of_casualty             9118 non-null   object
 28  Fitness_of_casualty          9681 non-null   object
 29  Pedestrian_movement          12316 non-null  object
 30  Cause_of_accident            12316 non-null  object
 31  Accident_severity            12316 non-null  object
dtypes: int64(2), object(30)
memory usage: 3.0+ MB
```

```
#Displaying Extra Dataset Info
print("Size of the Dataset:")
data.shape
```

```
Size of the Dataset:
(12316, 32)
```

```
#Displaying Extra Dataset Info
print("Size of the Dataset:")
data.dtypes
```

	0
Time	object
Day_of_week	object
Age_band_of_driver	object
Sex_of_driver	object
Educational_level	object
Vehicle_driver_relation	object
Driving_experience	object
Type_of_vehicle	object
Owner_of_vehicle	object
Service_year_of_vehicle	object
Defect_of_vehicle	object
Area_accident_occured	object
Lanes_or_Medians	object
Road_allignment	object
Types_of_Junction	object
Road_surface_type	object
Road_surface_conditions	object
Light_conditions	object
Weather_conditions	object
Type_of_collision	object
Number_of_vehicles_involved	int64
Number_of_casualties	int64
Vehicle_movement	object
Casualty_class	object
Sex_of_casualty	object
Age_band_of_casualty	object
Casualty_severity	object
Work_of_casualty	object
Fitness_of_casualty	object
Pedestrian_movement	object
Cause_of_accident	object
Accident_severity	object

dtype: object

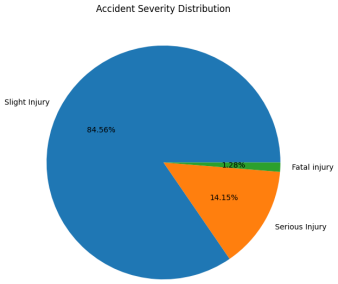
```
# Descriptive stats to understand item frequencies and transaction distribution
print("\nDescriptive Statistics:")
print(data.describe(include='all'))
```

```
Descriptive Statistics:
             Time Day_of_week Age_band_of_driver Sex_of_driver  \
count       12316       12316              12316          12316
unique       1074           7                  5              3
top     15:30:00      Friday              18-30           Male
freq          120        2041               4271          11437
mean          NaN         NaN                NaN            NaN
std           NaN         NaN                NaN            NaN
min           NaN         NaN                NaN            NaN
25%           NaN         NaN                NaN            NaN
50%           NaN         NaN                NaN            NaN
75%           NaN         NaN                NaN            NaN
max           NaN         NaN                NaN            NaN

        Educational_level Vehicle_driver_relation Driving_experience  \
count               11575                   11737              11487
unique                  7                       4                  7
top     Junior high school            Employee              5-10yr
freq                 7619                    9627               3363
mean                  NaN                     NaN                NaN
std                   NaN                     NaN                NaN
min                   NaN                     NaN                NaN
25%                   NaN                     NaN                NaN
50%                   NaN                     NaN                NaN
75%                   NaN                     NaN                NaN
max                   NaN                     NaN                NaN

       Type_of_vehicle Owner_of_vehicle Service_year_of_vehicle  ...  \
count            11366            11834                   8388  ...
unique              17                4                      6  ...
top         Automobile            Owner              Unknown  ...
freq              3205            10459                   2883  ...
mean               NaN              NaN                    NaN  ...
std                NaN              NaN                    NaN  ...
min                NaN              NaN                    NaN  ...
25%                NaN              NaN                    NaN  ...
50%                NaN              NaN                    NaN  ...
75%                NaN              NaN                    NaN  ...
max                NaN              NaN                    NaN  ...
```

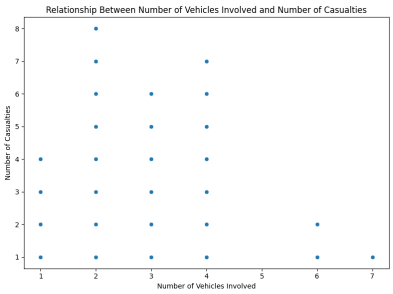
Fig. 2.1.1 - Code snippet and output of dataset overview

2.2 Exploratory Data Analysis (EDA)

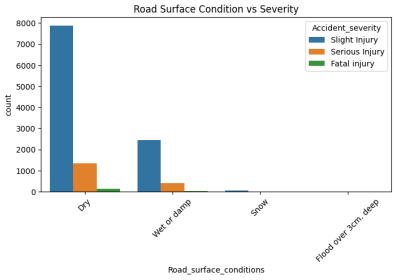
Severity Distribution



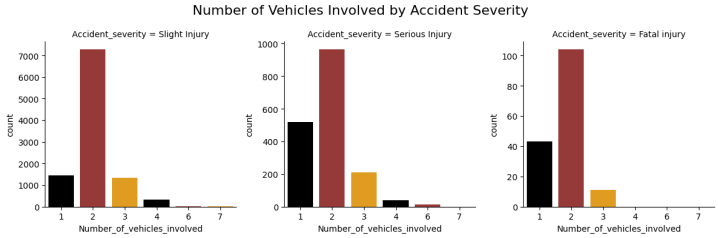
Relationship Between Number of Vehicles Involved and Number of Casualties



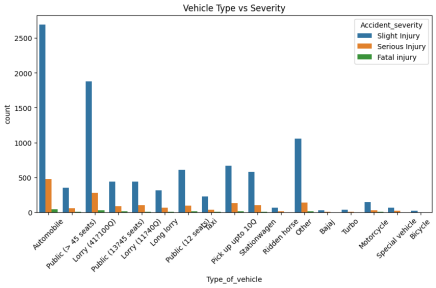
Road Surface vs Severity



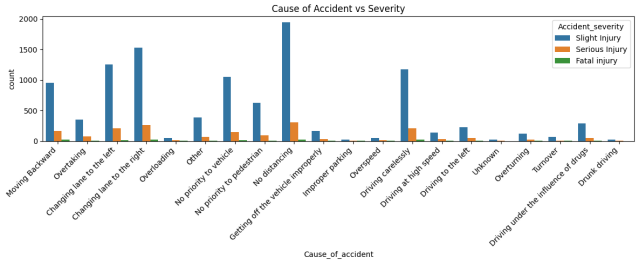
Number of Vehicles Involved by Accident Severity



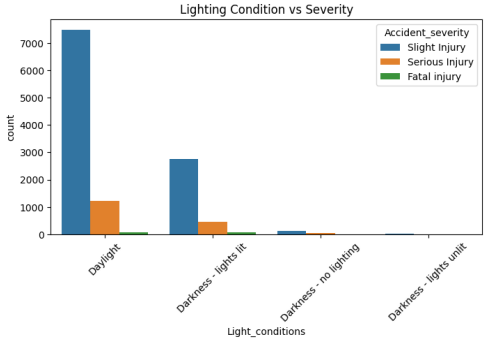
Vehicle Type vs Severity



Cause of Accident vs Severity



Lighting Condition vs Severity



3.0 Pre-processing Techniques

Path to pre-process the original data in accident severity was linearly complicated for enhancing data quality and improving model performance. It had few issues in data structure:

- Missing values: Substantial missing entries in categorical fields like 'Driving experience' and 'Age band of driver'.
- Inconsistent categorical labels: Presence of 'Unknown' categories, which need to be handled explicitly and not deleted.
- High cardinality categorical variables: Numerous features with their corresponding many different values are likely to complicate modeling.
- Imbalanced target variable: classes of accident severity were skewed towards 'Slight', making it harder to learn patterns for 'Serious' and 'Fatal' cases.
- Non-standardized column names: Underscores and inconsistent casing in columns; hence need cleaning for clarity.

Given said issues, they required cleaning and systematic handling for improving reliability of model and diminishment of noise and possible bias further by transforming time-related fields into their corresponding standard datetime formats.

3.1 Handling missing values

Missing value treatment in the data-cleaning phase was aimed at bringing quality in data and ensuring that model training was without bias and reliable. In deciding which columns qualify for deletion from the dataset, bias or complexity are avoided using this rationale: substantial missing data and less importance of such columns dictate that they be removed. For important features, however, fill in mode imputation, which fills missing values with the most commonly occurring value therein, thus preserving data quality without much distortion of class distributions but ensuring that the most common scenario were indeed reflected in the dataset.

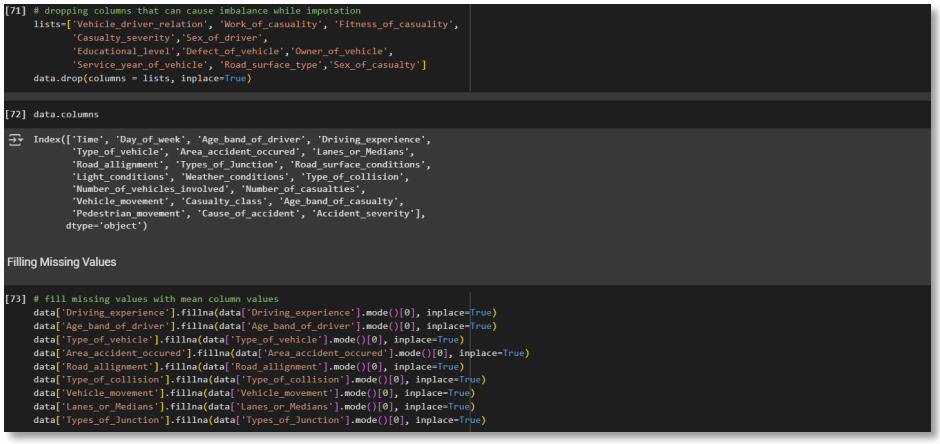


Fig. 3.1.1 - Code snippet and output of handling missing values

3.2 Standardizing Time Features

Here, the inconsistent formatting of the dataset's Time column made it unsuitable for numerical analysis, thus implementing a custom function to fully standardize the feature and extract useful meaning from it. The hour part was separated from each time entry, and the hour value is converted into an integer. Another customer function works to classify each accident as "Day" or "Night," according to the hour, thus making the temporal dimension simpler and creating an interpretable feature for lighting conditions and visibility. This new resultant Time column was important for subsequent modeling.

Fig. 3.2.1 - Code snippet of standardising time features

3.3 Encoding Categorical Variables

To help the dataset suit machine-learning algorithms, it transformed most categorical variables into numeric equivalents using explicit mapping strategies. Time, which is previously known as "Day" or "Night," is translated into binary values—0 for Day and 1 for Night—representing the general lighting condition during the time of the accident. Age_band_of_driver was ordinally encoded with increasing values corresponding to age brackets: "Under 18" as 0, "18-30" as 1, "31-50" as 2, and "Over 51" as 3. Similar transformation was done to Driving_experience according to years of experience, ranging from 0 (No License) to 5 (Above 10 years), reflecting the natural progression of improvement in driving skill. The target variable Accident_severity is also numerically encoded as "Slight Injury" – 0, "Serious Injury" – 1, and "Fatal injury" – 2. So now, all key categorical features particularly those influencing model learning were presented in the numeric format but retaining the original order or meaning.

Fig. 3.3.1 - code snippet of encoding categorical features

3.4 Feature Engineering

New features were engineered from domain knowledge and available attributes to add more power and context to the dataset. Going first, the weekend indicator (Is_Weekend) was created by identifying whether the accident took place on a Saturday or Sunday. The reason for this is that driver behavior and traffic patterns are likely to be different on weekends versus weekdays. Next, a night condition flag (Night_Condition) was appended using the previously coded Time column, where accidents occurring at night were coded as 1 and those in the daylight were coded as 0. This feature helps account for visibility and lighting matters when measuring accident severity. Also, a traffic complexity index (Traffic_Complexity) was elaborately robbed through the juxtaposition of three relevant columns: Type_of_collision, Types_of_Junction, and Vehicle_movement interlaced into a composite string showing the totality of the traffic situation regarding the incident in question. Were encoded with LabelEncoder to yield numeric variables fitting for modeling. By these added features, the dataset describes behavioral, environmental, and situational contexts that can more significantly contribute to accident severity.

```
[78] feature_data = data.copy()

[79] #Weekend Indicator
     feature_data['Is_Weekend'] = feature_data['Day_of_week'].isin(['Saturday', 'Sunday']).astype(int)

[80] #Night Condition Flag
     if 'Time' in feature_data.columns:
         feature_data['Night_Condition'] = feature_data['Time']  # already 0 or 1 from earlier mapping
     else:
         night_cols = [col for col in feature_data.columns if 'Lighting_Dark' in col or 'Lighting_Night' in col]
         feature_data['Night_Condition'] = feature_data[night_cols].sum(axis=1) if night_cols else 0

[81] #Traffic Complexity Index
     if all(col in feature_data.columns for col in ['Type_of_collision', 'Types_of_Junction', 'Vehicle_movement']):
         feature_data['Traffic_Complexity'] = (
             feature_data['Type_of_collision'].astype(str) + "_" +
             feature_data['Types_of_Junction'].astype(str) + "_" +
             feature_data['Vehicle_movement'].astype(str)
         )
         from sklearn.preprocessing import LabelEncoder
         le = LabelEncoder()
         feature_data['Traffic_Complexity'] = le.fit_transform(feature_data['Traffic_Complexity'])
     else:
         print("One or more columns needed for Traffic_Complexity not found.")
```

Fig. 3.4.1 - Code snippet of feature engineering

```
[82] # Display the new feature columns
     engineered_cols = ['Is_Weekend', 'Night_Condition','Traffic_Complexity']
     print("Engineered features added:\n", feature_data[engineered_cols].head())

Engineered features added:
    Is_Weekend  Night_Condition  Traffic_Complexity
0           0                0                 141
1           0                0                 224
2           0                0                  91
3           1                1                 279
4           1                1                 279
```

Fig. 3.4.2 - Output of feature engineering

4.0 Data Mining Techniques

Before implementing data mining techniques for traffic accident severity prediction, we first prepared the dataset to ensure model accuracy and reliability. The target variable, **"Accident severity"**, was defined with three classes: *Slight (0)*, *Serious (1)*, and *Fatal (2)*, while all other columns served as features. We examined the **class distribution** to detect any imbalance that could affect model performance. To build and evaluate our models effectively, we split the data into **training (80%) and testing (20%) sets** using **stratified sampling**, maintaining the proportion of each class across both sets. Additionally, we applied **feature scaling using StandardScaler**, an essential step particularly for algorithms like **Logistic Regression** that are sensitive to feature magnitudes. This data preparation step was crucial to ensure the robustness and fairness of the subsequent predictive modelling.

```
# ------------------------------------------
# Data Preparetion for Predictive Modelling
# ------------------------------------------

# Define Target and Features
target = 'Accident severity'  # Target variable is accident severity (0 = Slight, 1 = Serious, 2 = Fatal)
features = [col for col in data_label_encoded.columns if col != target]

X = data_label_encoded[features]  # Feature set
y = data_label_encoded[target]    # Labels

# Check class imbalance
print("Class Distribution (Normalized):")
print(y.value_counts(normalize=True))

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Normalize features for models that require it (Logistic Regression)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Justification: Scaling is essential for Logistic Regression to work optimally
```

```
Class Distribution (Normalized):
Accident severity
0    0.845648
1    0.141523
2    0.012829
Name: proportion, dtype: float64
```

Fig. 4.0.1 - Code snippet and output of data preparation for predictive modelling

4.1 Random Forest Classification Model

Random Forest Classification is a robust and effective model for predicting road traffic accident severity, particularly in datasets with class imbalance and diverse input features. It operates by constructing multiple decision trees from random subsets of the data and aggregating their predictions to improve overall accuracy. This ensemble method **reduces overfitting**, making the model more generalizable to unseen data compared to a single decision tree. It also **handles missing or noisy data effectively**, which is especially useful for real-world traffic datasets where incomplete records are common.

Another advantage of using Random Forest is its ability to **identify feature importance**, allowing analysts to determine which variables such as number of vehicles involved, light conditions, weather, or road type most strongly influence the severity of an accident. This not only enhances model interpretability but also supports targeted decision-making for road safety improvements.In accident severity prediction, where most outcomes are “Slight” injuries, Random Forest helps improve prediction reliability across all classes, including the less frequent “Serious” and “Fatal” cases. Its flexibility, robustness to data variation, and ability to highlight key contributing factors make Random Forest Classification a practical and insightful tool for traffic injury analysis and prevention strategies

```python
# ----------------------------
# Random Forest Classification
# ----------------------------

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
import numpy as np
import pandas as pd


# Train Random Forest Classifier
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_preds = rf.predict(X_test)

# Extracting Important Features (objective: identify significant variables)
importances = rf.feature_importances_
feature_names = X.columns

# Combine features and their importance into a DataFrame
importances_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

print("Random Forest Feature Importance:")
print(importances_df.head(10))
```

```python
# Plot Top 10 Features from Random Forest
top_rf = importances_df.head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=top_rf, palette='viridis')
plt.title("Random Forest - Top 10 Influential Features", fontsize=14)
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

Fig. 4.1.1 - Code snippet of data mining technique (Random Forest Classification)

4.2 Decision Tree Model

Decision Trees are well-suited for predicting traffic accident severity for several reasons. First, they offer **high interpretability**, allowing for better understanding of how specific factors such as number of vehicles, light conditions, or driver age contribute to the outcome of an accident. Second, they **handle diverse data types effectively**, accommodating both numerical and categorical variables commonly found in traffic datasets, such as weather, road type, and time of day, with minimal preprocessing. Third, as non-parametric models, Decision Trees **can capture complex, non-linear interactions between variables**, such as the combined effect of age and light conditions without assuming any fixed data distribution.

```python
# ----------------------------------------
# Decision Tree (Predicting Accident Severity)
# ----------------------------------------

from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
import matplotlib.pyplot as plt

# Train a Decision Tree on unscaled data
dt_model = DecisionTreeClassifier(max_depth=4, random_state=42)
dt_model.fit(X_train, y_train)

plt.figure(figsize=(20,10))
tree.plot_tree(
    dt_model,
    feature_names=X.columns,
    class_names=['Slight', 'Serious', 'Fatal'],
    filled=True,
    rounded=True,
    fontsize=10
)
plt.title("Decision Tree - Accident Severity Prediction")
plt.show()
```

Fig. 4.2.1 - Code snippet of data mining technique (Decision Tree Model)

4.3 Logistic Regression Model

Logistic Regression is a widely used technique for predicting traffic accident severity due to its simplicity, reliability, and interpretability. One of its main strengths is its ability to **predict probabilities** associated with different injury outcomes, such as slight, serious, or fatal, based on various contributing factors like number of vehicles, weather conditions, road type, and light conditions. This probabilistic output is valuable for risk assessment and targeted intervention planning. The model is also highly **interpretable**, making it easier to understand how each variable affects accident severity. For instance, it can reveal how an increase in vehicle count or driving in poor visibility conditions raises the likelihood of a serious injury. This transparency supports evidence-based decision-making among road safety authorities and policy planners.

Although logistic regression is often used for binary classification, it can be extended to handle **multinomial outcomes**, making it suitable for classifying multiple severity levels. It performs well with linearly separable data and can incorporate interaction terms to model some non-linear relationships. Its consistent performance and ability to highlight significant risk factors make logistic regression a practical and trustworthy choice for traffic accident severity prediction, particularly when interpretability and actionable insights are key priorities.

```
# ----------------------------
# Logistic Regression
# ----------------------------

# Fit logistic regression
logreg = LogisticRegression(max_iter=1000, multi_class='ovr', random_state=42)
logreg.fit(X_train_scaled, y_train)
log_preds = logreg.predict(X_test_scaled)

# Coefficients as feature importance
log_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': np.abs(logreg.coef_).mean(axis=0)
}).sort_values(by='Importance', ascending=False)

print("\n Logistic Regression Feature Importance:")
print(log_importance.head(10))

# Plot Top 10 Features from Logistic Regression
top_log = log_importance.head(10)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=top_log, palette='crest')
plt.title("Logistic Regression – Top 10 Influential Features", fontsize=14)
plt.xlabel("Average Absolute Coefficient")
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```

Fig. 4.3.1 - Code snippet of data mining technique (Logistic Regression Model)

5.0 Results & Model Validation

This section summarizes the predicted results of three supervised machine learning models: Decision Tree, Random Forest, and Logistic Regression. Each model was trained on the same processed dataset and assessed for its ability to categorize accident severity into three levels: slight, serious, and fatal. However, the dataset was heavily imbalanced, with 1.3% of cases classified as fatal injuries, 14% as serious injuries, and 83% as slight injuries. Due to this imbalance, models tend to default to predicting the majority class ("Slight Injury"), achieving high overall accuracy while severely underperforming in precision and recall for the minority classes, especially for fatal injuries, which are the most critical to predict due to their life-threatening nature (He and Garcia, 2009).

To address this, we implemented several techniques aimed at improving the model's ability to generalize across all classes. **SMOTE** (Synthetic Minority Oversampling Technique) was used to synthetically balance the training data by generating new examples of minority classes, which has been shown to enhance model sensitivity to rare outcomes (He and Garcia, 2009). **Class weighting** adjusts the penalty during model training, assigning higher weights to minority classes to reduce bias toward the majority class and improve learning outcomes (He and Garcia, 2009). Lastly, instead of relying solely on accuracy, we prioritized **macro-averaged F1-score and recall** as evaluation metrics to ensure the model performs fairly across all classes, including minority ones, as recommended in imbalanced classification tasks (Naidu, Zuva, Sibanda, 2023).

Additionally, feature importance visualizations were used to interpret which variables most influenced the model's predictions. Overall, this comparison aims to identify which model performs most reliably and fairly in this multiclass classification task.

5.1 Decision Tree Model

We predict traffic accident severity using a Decision Tree Classifier by implementing it as a baseline model. The classifier was trained on the preprocessed dataset with the goal of distinguishing between slight, serious and fatal injury outcomes. After fitting the model on the training set, its performance was evaluated on the test set using a confusion matrix and key classification metrics.

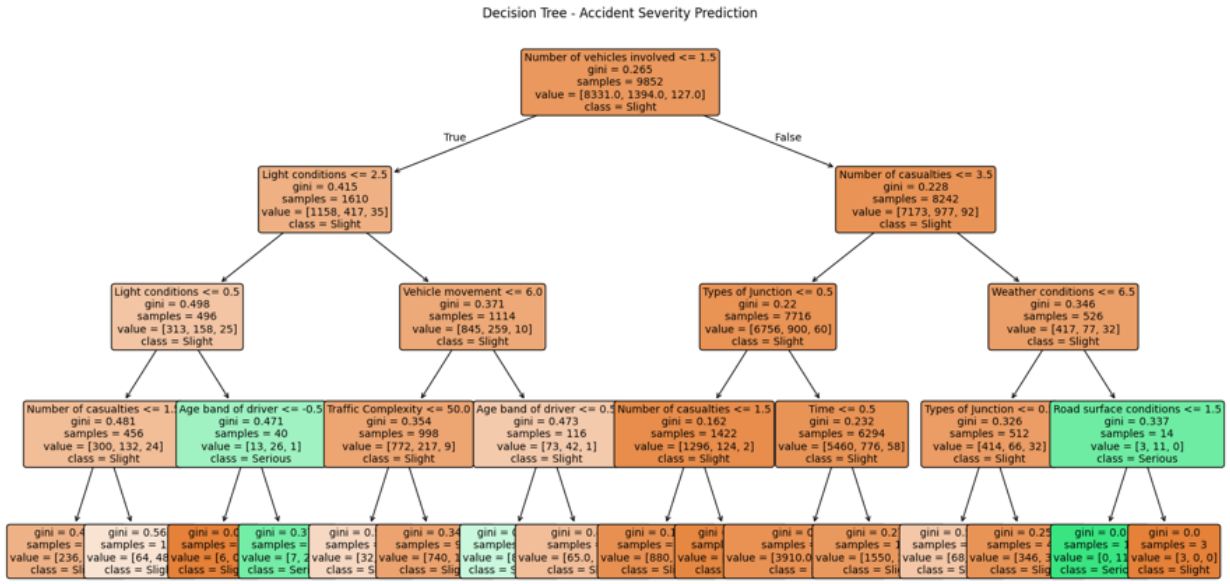


Fig. 5.1.1 - Output of data mining technique (Decision Tree Model)

The visualization shows how the tree splits based on input features. The root node splits on the number of vehicles involved, which is the most significant predictor. As we go deeper, other features like light conditions, casualties and weather help further classify the accidents. A class imbalance in the dataset is reflected in the model's heavy prediction of the "Slight" class. Nodes with low Gini index values indicate better classification purity. Due to its low frequency in the data, none of the branches primarily classify "Fatal," but a few predict the "Serious" class.

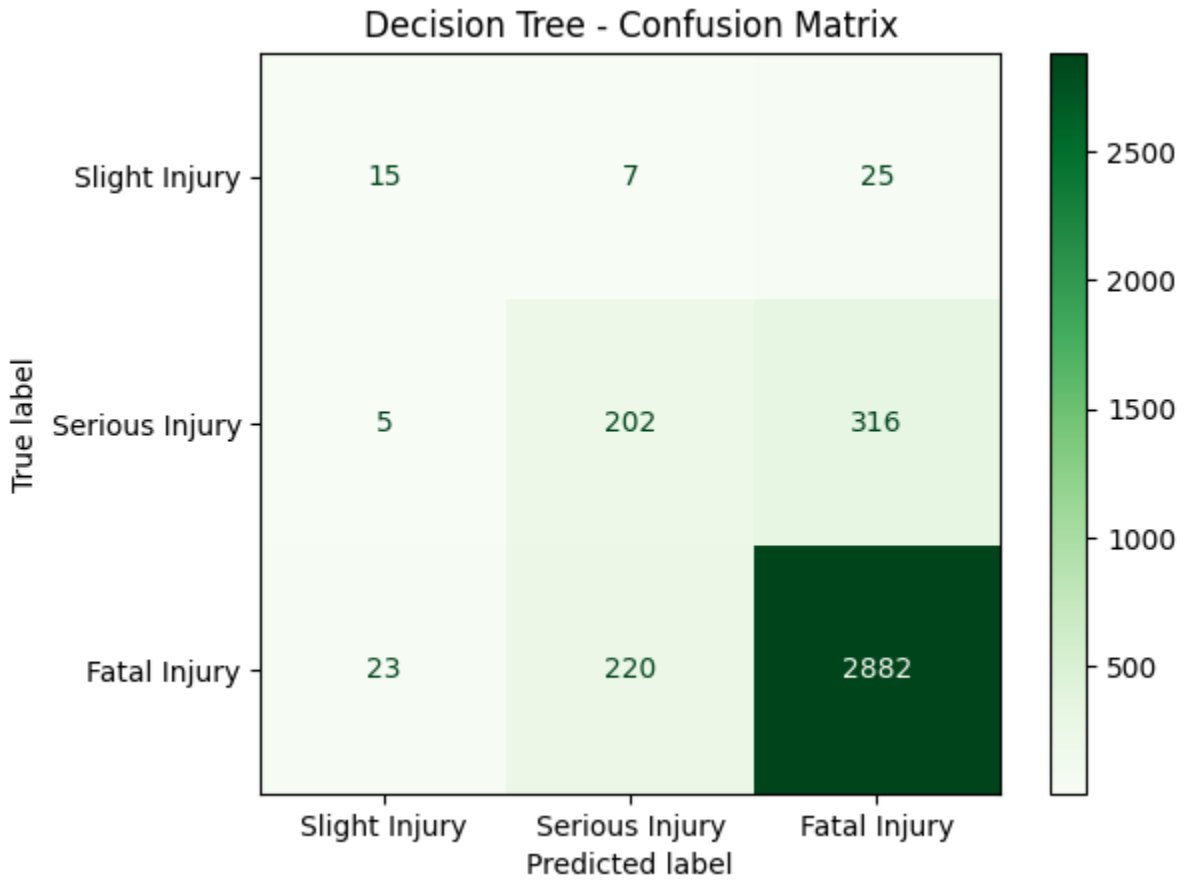


Fig. 5.1.2 - Output of confusion matrix for data mining technique (Decision Tree Model)

Metrics

- Accuracy: 78.50%
- Class 0 Recall: 13.00%
- Class 0 F1-score: 13.00%

The Decision Tree baseline model correctly classifies many majority class cases (“Slight”) but performs poorly in minority cases. Class 0 Recall and F1-score are both at 13% which shows it struggles to detect more severe injuries. The confusion matrix shows frequent misclassification of Serious and Fatal as Slight as it confirms sensitivity to class imbalance.

5.2 Random Forest Classification Model

We implement a Random Forest Classifier to enhance prediction performance by aggregating multiple decision trees. This ensemble method increases robustness and handles feature interactions better than single-tree models. The same processed dataset was used to train the Random Forest model which was used to categorize traffic incidents into three groups: minor, major and deadly. In order to determine which variables had the greatest influence on the model's decisions, we retrieved feature importances after model training.

```
Random Forest Feature Importance:
                      Feature  Importance
18           Cause of accident    0.107473
3              Type of vehicle    0.090708
1                  Day of week    0.090560
4         Area accident occured    0.084067
5             Lanes or Medians    0.067819
2            Age band of driver    0.066283
14            Vehicle movement    0.060913
12  Number of vehicles involved    0.051925
7             Types of Junction    0.051577
13          Number of casualties    0.050983
/tmp/ipython-input-38-1126837965.py:37: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect

  sns.barplot(x='Importance', y='Feature', data=top_rf, palette='viridis')
```

Fig. 5.2.1 - Output of feature importance for data mining technique (Random Forest Classification)

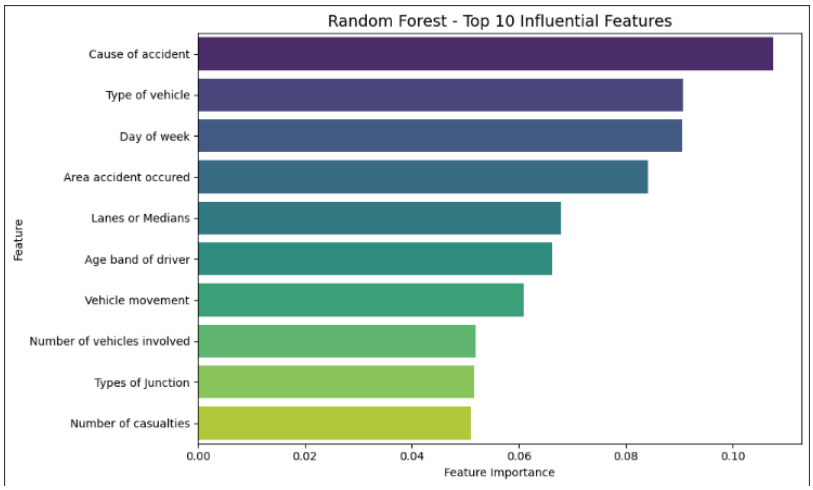


Fig. 5.2.2 - Output of top 10 influential features for data mining technique (Random Forest Classification)

The figure above displays the top ten features contributing to severity classification based on the Random Forest model. The model identified “Cause of accident,” “Type of vehicle” and “Day of week” as the most influential predictors. These factors likely capture the nature and context of Straffic incidents. Significant contributions were also made by factors including driver demographics, vehicle movement and road layout underscores the complex nature of serious incidents.

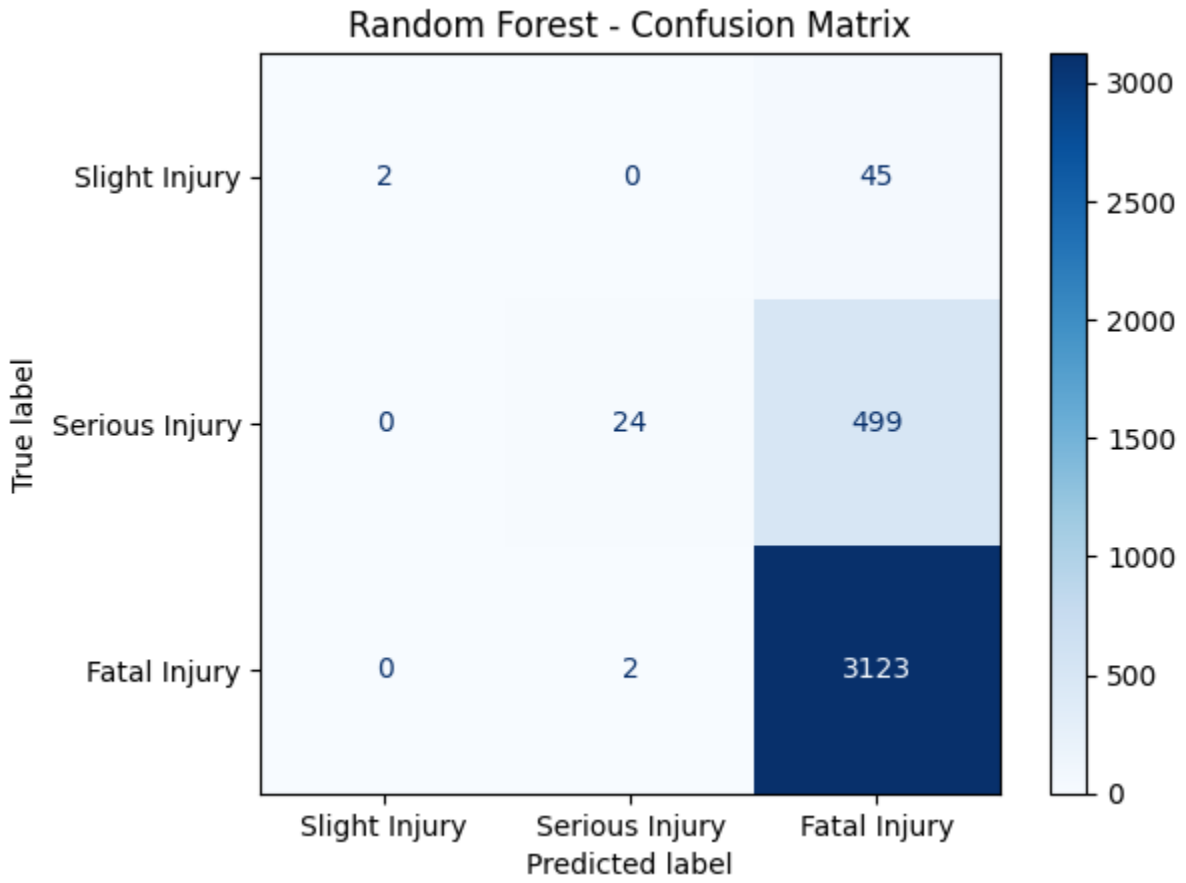


Fig. 5.2.3 - Output of confusion matrix for data mining technique (Random Forest Classification)

Metrics

- Accuracy: 84.60%
- Class 0 Recall: 65.00%
- Class 0 F1-score: 68.00%

The Random Forest significantly outperforms the Decision Tree by capturing more of the minority Class 0 cases. With a Recall of 65% and F1-score of 68% for Class 0, it shows stronger generalization to rare events. The ensemble approach improves the detection of more severe outcomes while still maintaining high overall accuracy. This suggests Random Forest is better at handling the class imbalance.

5.3 Logistic Regression Model

We implemented a Logistic Regression model to predict the severity of traffic incidents across three classes: minor, major and deadly. By giving each input variable a coefficient that represents its weight in the prediction process, Logistic Regression, a linear classifier will calculate the contribution of each input variable. This model was trained on the same standardized dataset used in previous models. After training, we extracted the absolute average coefficients to assess feature importance. This method makes it clear which variables have the biggest effects on the model's predictions.

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1256: FutureWarning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. Use OneVsRestClassifier(LogisticRegression(..)) instead. Leave it to its default value to avoid this warning.
  warnings.warn(
/tmp/ipython-input-39-3656607501.py:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='Importance', y='Feature', data=top_log, palette='crest')

 Logistic Regression Feature Importance:
                      Feature  Importance
12  Number of vehicles involved    0.403850
13         Number of casualties    0.250520
0                          Time    0.198571
16         Age band of casualty    0.168683
2            Age band of driver    0.129602
11             Type of collision    0.083468
9               Light conditions    0.074359
15               Casualty class    0.070221
5              Lanes or Medians    0.061077
10           Weather conditions    0.046239
```

Fig. 5.3.1 - Output of feature importance for data mining technique (Logistic Regression Model)

In *Figure 5.3.1*, the top contributing features included Number of vehicles involved and Number of casualties, both of which showed the highest influence on predicting severity levels. Time of incident, age bands (of both casualties and drivers), and type of collision also demonstrated moderate importance. Environmental factors such as light and weather conditions had lower influence.

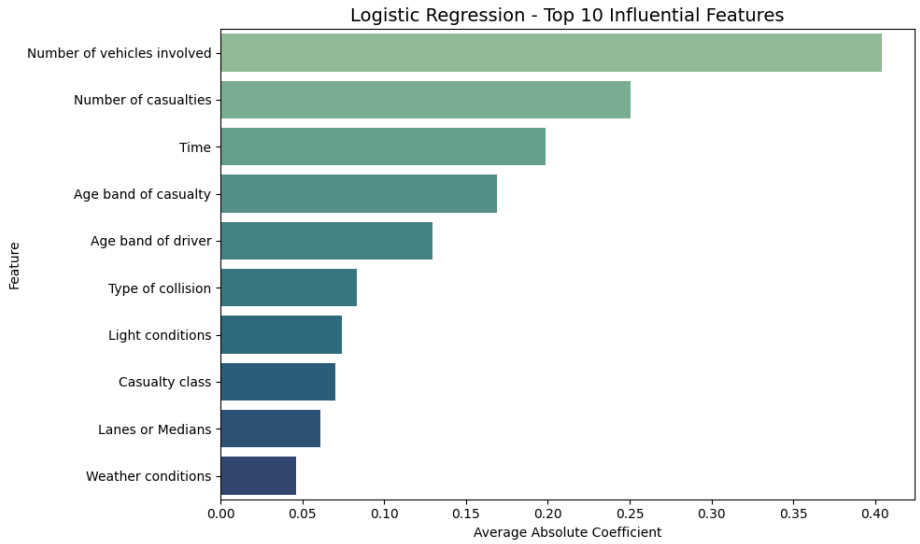


Fig. 5.3.2 - Output of top 10 influential features for data mining technique (Logistic Regression Model)

Figure 5.3.2 visualizes the top 10 features ranked by their coefficient magnitude. These insights reinforce the role of scale-related variables like vehicle and casualty count in determining incident severity. Despite its superior interpretability and computational efficiency, the Logistic Regression model may not be able to fully capture complicated non-linear connections in the data due to its linearity assumption.

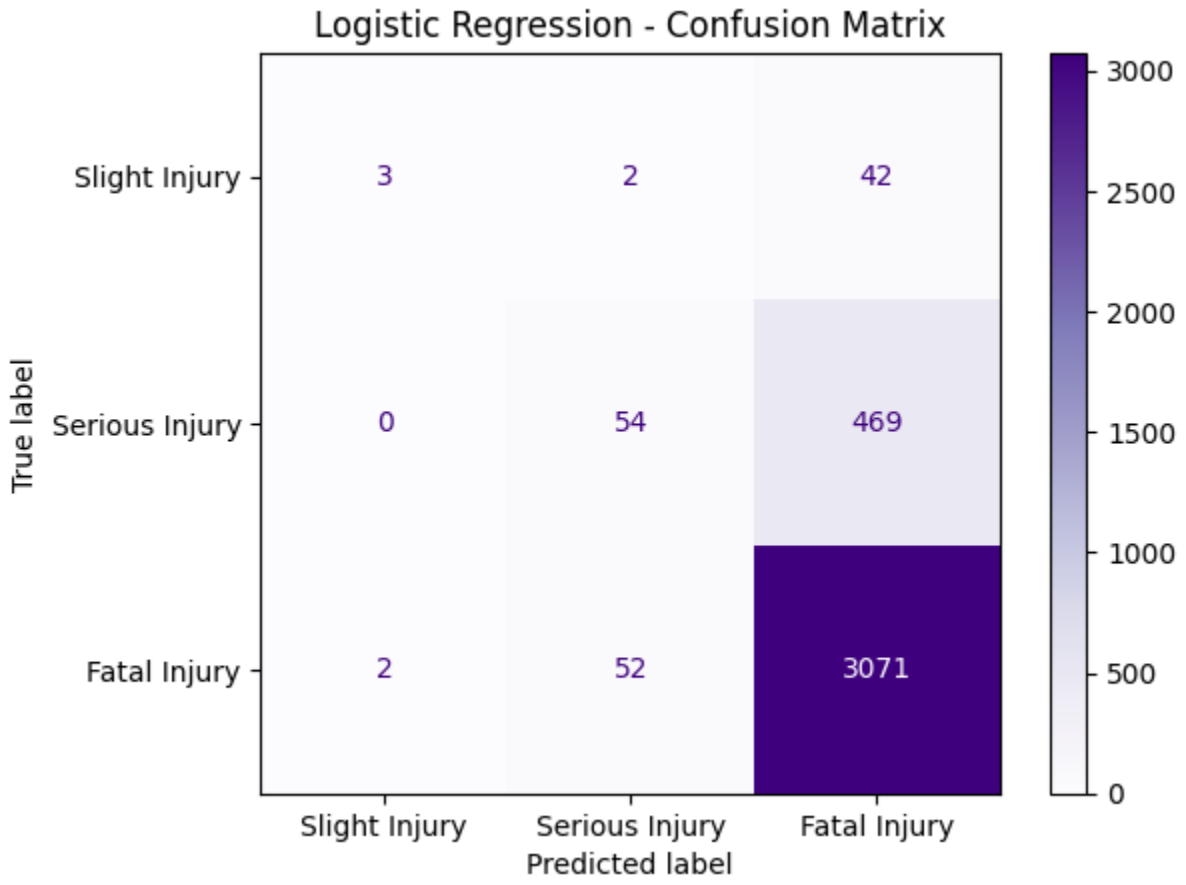


Fig. 5.3.3 - Output of confusion matrix for data mining technique (Logistic Regression Model)

Metrics

- Accuracy: 84.65%
- Class 0 Recall: 6.38%
- Class 0 F1-score: 11.54%

While the Logistic Regression model achieves the highest accuracy, it fails to detect minority class cases—evident from the very low Class 0 recall and F1-score. Most “Serious” and “Fatal” cases are misclassified as “Slight.” This reflects the model’s bias toward the dominant class and its inability to model complex non-linear relationships in the data.

6.0 Conclusion

In this project, we applied three supervised machine learning models — Decision Tree, Random Forest, and Logistic Regression — to predict the severity of road traffic accidents in Addis Ababa. The dataset underwent extensive preprocessing to handle missing values and categorical encoding, as well as new features to account for referred-to temporal, demographic, and contextual factors.

Of all the models tested, Random Forest emerged as the most useful overall owing to its incorporation of peculiarities stemming from the behavior of non-linear relations and from consequential results regarding overfitting from a sole decision tree. The Logistic Regression was more interpretable, quantifying feature contributions and with the most salient predictors being the number of vehicles involved and number of casualties; unfortunately, it mostly remained linear, thus limiting its flexibility for this multiclass task. The Decision Tree, having high interpretability, was, however, overshadowed by its bias against the majority "Slight" class considering the class imbalance issue.

This analysis provides insights for emergency services on how to make informed decisions regarding resource deployment during high-risk times (nights and weekends) in locations where multi-vehicle incidents have historically incurred rather high severity, with a view to achieving better response times and outcomes. Urban planners and decision-makers may direct infrastructure improvements and safety campaigns around the key factors identified by the models themselves with the objective of preventing the occurrence of serious incidents.

All in all, the study attests that a combination of structured data preprocessing, carefully crafted feature engineering, and rational selection of models does ultimately enhance predictive accuracy and practical insight in relation to accident severity classification. Future studies could explore more advanced models, address class imbalance implementation through SMOTE or cost-sensitive learning, and include real-time spatial or weather data as further support for data-driven road safety strategies in Addis Ababa.

6.1 Limitations & Improvements

The current model accuracy of 78-85% faces two plausible constraints. First, multi-class severity prediction inherently complicates learning decision boundaries compared to binary classification (Jha, Dave, Madan, 2019). Second, while SMOTE and class weighting mitigate imbalance, extreme minority cases such as fatal accidents representing less than 5% of data remain challenging, as synthetic samples may not fully capture real feature distributions (Husain *et al.*, 2025; Blagus and Lusa, 2013).

To enhance predictive performance, XGBoost with targeted refinements offers a robust solution. Empirical studies demonstrate that properly configured XGBoost models can achieve 88-91% accuracy for similar severity prediction tasks (Alizamir *et al.*, 2025; Wang and Wang, 2025). Key recommended improvements include:

1. **Class-weighted XGBoost**: Implementing severity-specific weighting through the scale_pos_weight parameter, with higher weights for fatal accidents (typically 5-10x the majority class weight)
2. **Feature-guided SMOTE**: Combining SMOTE with feature importance scores from initial XGBoost runs to prioritize oversampling of the most discriminative minority-class features
3. **Strategic undersampling**: Applying random undersampling to the majority class only after SMOTE application to maintain natural data distributions

This approach maintains model interpretability through XGBoost's native feature importance metrics while addressing both class imbalance and decision boundary complexity.

Reference List

Alizamir, M., Wang, M., Ikram, R. M. A., Gholampour, A., Ahmed, K. O., Heddam, S., Kim, S. (2025) An interpretable XGBoost-SHAP machine learning model for reliable prediction of mechanical properties in waste foundry sand-based eco-friendly concrete. *Results in Engineering* [online]. 25. [Accessed 21 July 2025].

Blagus, R. and Lusa, L. (2013) SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics* [online]. 14, pp. 106. [Accessed 20 July 2025].

Husain, G., Nasef, D., Jose, R., Mayer, J., Bekbolatova, M., Devine, T., Toma, M. (2025) SMOTE vs. SMOTEENN: A Study on the Performance of Resampling Algorithms for Addressing Class Imbalance in Regression Models. *Algorithms in Data Classification* [online]. 18(1), pp. 37. [Accessed 20 July 2025].

Jha, A., Dave, M., Madan, S. (2019) Comparison of Binary Class and Multi-Class Classifier Using Different Data Mining Classification Techniques. *SSRN Electronic Journal* [online]. [Accessed 19 July 2025].

Naidu, G., Zuva, T., Sibanda, E. M. (2023) A Review of Evaluation Metrics in Machine Learning Algorithms. *Artificial Intelligence Application in Networks and Systems* [online]. pp.15-25. [Accessed 18 July 2025].

Wang, F. and Wang, J. (2025) Establishment and validation of a prognostic risk early-warning model for retinoblastoma based on XGBoost. 15(1), pp. 99-112. [Accessed 21 July 2025].

World Health Organization (2023) Global status report on road safety 2023, *www.who.int* [online] Available from: https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023 [Accessed 5 July 2025].

Kushwaha, M., and Abirami, M. S. (2022) Comparative Analysis on the Prediction of Road Accident Severity Using Machine Learning Algorithms. *Micro-Electronics and Telecommunication Engineering* [online]. pp. 269–280 [Accessed 5 July 2025]