**PHASE-2**

**STUDENT NAME:**

Gracey.M

**REGISTER NUMBER:**

422223106013

**INSTITUTION:**

Surya Group Of Institutions

**DEPARTMENT:**

B.E “Electronics And Communication Engineering”

**DATE OF SUBMISSION:**

**02/05/2025**

**GITHUB REPOSITORY LINK:**

[**https://github.com/gracey-antony/NM-Files.git**](https://github.com/gracey-antony/NM-Files.git)

**Project : ENHANCING ROAD SAFETY WITH AI-DRIVEN TRAFFIC ACCIDENT ANALYSIS AND PREDICTION**

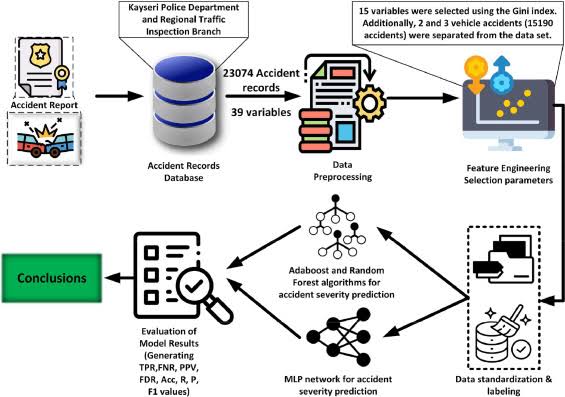
**1. Problem Statement**

* + Revisit and refine the problem based on additional understanding of the dataset. (e.g., Specific types of accidents, high-risk locations identified during initial data exploration).
  + Clearly define the type of problem (e.g., Classification: predicting accident severity; Regression: predicting the number of accidents in a specific area; Clustering: identifying accident hotspots).
  + Explain why solving this problem matters (impact, relevance, or application area). (e.g., Reducing fatalities and injuries, improving traffic flow, informing policy decisions for safer roads).

**2. Project Objectives**

* + Define the key technical objectives (e.g., Build a classification model to predict accident severity with X% accuracy; Develop a regression model to forecast accident frequency in specific zones; Cluster high-accident risk areas based on historical data).
  + Specify what the model aims to achieve (e.g., High accuracy in predicting severe accidents, interpretable factors contributing to accidents, real-world applicability in a traffic management system).
  + Mention if the goal has changed or evolved after data exploration. (e.g., Initially aimed for regression but shifted to classification based on data distribution and insights).

**3. Flowchart of the Project Workflow**

****

**4. Data Description**

* + Dataset name and origin (e.g., "Tamil Nadu Road Accident Dataset 2020-2024," sourced from a government transportation authority).
  + Type of data: structured (e.g., tabular data with features like location, time, vehicle type, road conditions, driver demographics).
  + Number of records and features (e.g., 10,000 accident records with 50 relevant features).
  + Static or dynamic dataset (e.g., Static dataset for the analysis period, but potentially dynamic for future real-time prediction).
  + Target variable (if supervised learning) (e.g., Accident Severity - [Minor, Moderate, Severe]; Number of Accidents in a Zone).
  + Dataset Link:
  + <https://www.data.gov.in/keywords/Traffic>

**5. Data Preprocessing**

* + Handle missing values (e.g., Imputed missing weather conditions using the most frequent value, removed records with excessive missing location data if justifiable).
  + Remove or justify duplicate records (e.g., Checked for and removed exact duplicate entries if they existed, justified keeping near-duplicates if they represent separate incidents).
  + Detect and treat outliers (e.g., Identified and capped extreme values for speed or vehicle age based on statistical analysis).
  + Convert data types and ensure consistency (e.g., Converted date/time columns to appropriate formats, standardized categorical entries).
  + Encode categorical variables (e.g., Used one-hot encoding for vehicle type and road conditions, label encoding for ordered categories like day of the week).
  + Normalize or standardize features where required (e.g., Standardized numerical features like speed and age for models sensitive to feature scaling).
  + Document the preprocessing steps and reasoning behind each decision.

import pandas as pd

from sklearn.model\_selection

import train\_test\_split

from sklearn.linear\_model import

LogisticRegression

from sklearn.metrics

import accuracy\_score,

classification\_report,

confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

#1. Create a synthetic dataset (replace with your actual data loading)

data = {

'speeding': [0, 1, 0, 1, 0, 1, 0, 1, 1, 1],

'drunk\_driving': [0, 0, 1, 1, 0, 0, 1, 1, 0, 1],

'red\_light\_violation': [0,0, 0, 1, 1, 0, 1, 0, 1, 1],

'weather\_bad': [0, 0, 0, 0, 1, 1, 1, 0, 1, 1],

'accident': [0, 1, 1, 1, 1, 0, 1, 1, 1, 1] # 0: No Accident, 1:

Accident

}

df = pd.DataFrame(data)

# 2. Separate features (X) and target (y)

X = df [['speeding',

'drunk\_driving',

'red\_light\_violation',

'weather\_bad']]

y = df ['accident']

# 3. Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

#4. Train a Logistic Regression model model = LogisticRegression()

model.fit(X\_train, y\_train)

#5. Make predictions on the test set

y\_pred model.predict(X\_test)

# 6. Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")

print(classificati

on\_report(y\_test,

y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True,

[10:01 PM, 5/2/2025] Gracey: cm = confusion\_matrix(y\_test,

y\_pred)

sns.heatmap(cm, annot=True,

fmt='d', cmap='Blues')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

# 7. (Optional) Use the model to predict on new data

new\_data = pd. DataFrame({

'speeding': [1, 0],

'drunk\_driving': [0, 1],

'red\_light\_violation': [1, 0],

'weather\_bad': [0, 1]

})

predictions =

model.predict(new\_data)

print("\nPredictions for new

data:")

print(predictions) # Output: [1 1] (meaning the model predicts an accident in both scenarios)

**6. Exploratory Data Analysis (EDA)**

* + Summary statistics of key features.
  + Visualizations of feature distributions (histograms, box plots).
  + Relationships between features and the target variable (scatter plots, bar charts).
  + Correlation analysis (heatmaps).
  + Identify initial trends and potential factors contributing to accidents (e.g., Higher accident rates during specific times, in certain locations, or involving particular vehicle types).

**7. Feature Engineering**

* + Generate relevant features from existing data (e.g., Time of day categories [morning, afternoon, evening, night] from timestamp, day of the week, month, season).
  + Create interaction features if they seem relevant (e.g., Interaction between road condition and weather condition).
  + Handle categorical features with high cardinality (e.g., Grouping less frequent categories).
  + Consider dimensionality reduction techniques (e.g., PCA) if the number of features is very high (and justify its use).
  + Justify each feature added or removed based on domain knowledge or EDA insights.

**8. Model Building**

* + Select and implement at least 2 machine learning models (e.g., Logistic Regression for predicting accident severity, Random Forest for its ability to handle non-linear relationships and feature importance).
  + Justify why these models were selected (based on problem type - classification/regression/clustering - and characteristics of the data).
  + Split data into training and testing sets (e.g., 80% training, 20% testing, with stratification if the target variable is imbalanced).
  + Train models on the training data.
  + Evaluate initial performance using appropriate metrics:
    - For classification (if predicting severity): accuracy, precision, recall, F1-score, AUC-ROC curve.
    - For regression (if predicting number of accidents): MAE, RMSE, R^{2} score.

**9. Visualization of Results & Model Insights**

* + Confusion matrix to visualize the performance of a classification model.
  + ROC curve and AUC score to assess the trade-off between true positive and false positive rates.
  + Feature importance plot (e.g., from Random Forest) to show which features have the most influence on the model's predictions.
  + Residual plots to evaluate the performance of a regression model.
  + Include visual comparisons of the performance of the different models.
  + Interpret the top features influencing the outcome in the context of road safety.
  + Clearly explain what each plot shows and how it supports your conclusions about the models' performance and the factors influencing road accidents.

**10. Tools and Technologies Used**

* + Programming Language: Python
  + IDE/Notebook: Google Colab
  + Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn

**11. Team Members and Contributions**

* + Clearly mention who worked on:
    - Data cleaning and EDA: P.ABI
    - Feature engineering: A.TAMILSELVI
    - Model Development, Documentation and reporting, guiding the team is done by M.GRACEY