### **Enhancing road safety with AI-driven traffic accident analysis and prediction**

### 

### **Student name: DIVYASRI.S**

### **Register name: 510923104015**

### **Institution: Global Institute Of Engineering And Technology**

### **Department: Computer science and Engineering**

### **Date of submission:07-05-2025**

### **Github Repository link:**[**https://github.com/saravanan1970/phase-3.git**](https://github.com/saravanan1970/phase-2.git)

### 

### **1. Problem Statement**

Road traffic accidents continue to be a critical global challenge, resulting in substantial human and economic losses. According to global health and safety reports, millions of people are injured or killed annually due to road accidents, and urban centers are particularly vulnerable due to growing populations, vehicle congestion, and infrastructure constraints. Traditional accident analysis and prevention strategies tend to be reactive—they often rely on post-incident reporting, manual assessment, and generalized safety interventions. These methods fall short in addressing the complex, dynamic, and context-sensitive nature of modern traffic environments.

This project aims to:

Analyze historical traffic collision data to identify trends and high-risk areas.

Uncover key contributing factors such as location, time, weather, road conditions, and vehicle types.

Develop machine learning models capable of predicting accident-prone zones and periods.

Provide actionable insights that aid in optimizing traffic control, designing targeted safety interventions, and allocating emergency resources more efficiently.

Ultimately, the goal is to improve urban road safety, reduce fatalities and injuries, and support smart city initiatives by leveraging AI for predictive traffic accident management.

## **2. Abstract**

Road traffic accidents pose a serious threat to public safety, leading to significant loss of life, injuries, and economic damage worldwide. Traditional approaches to accident prevention are largely reactive, relying on post-incident analysis and static intervention strategies that fail to adapt to the complex, evolving nature of urban traffic systems. This project proposes a data-driven, proactive framework for enhancing road safety by leveraging Artificial Intelligence (AI) and Machine Learning (ML) techniques. Using real-world accident datasets, such as the NYPD’s Motor Vehicle Collisions – Crashes data, the study conducts comprehensive data preprocessing, exploratory analysis, and pattern recognition to uncover key contributing factors and high-risk areas. Advanced ML models are developed to predict accident hotspots based on spatial, temporal, and contextual variables. The outcome is a predictive system that empowers traffic authorities and urban planners with actionable insights to optimize traffic management, deploy timely safety measures, and ultimately reduce the frequency and severity of road traffic accidents. This research supports the vision of intelligent, data-informed transportation systems within smart city framework.

Project Objectives.

## **3. System Requirements**

* **Software**

To develop an AI-based traffic accident prediction system, you'll need a robust software stack encompassing data processing, machine learning, real-time analytics, and user interfaces. Here's a comprehensive guide to the essential software components:

**1. Programming Languages**

* **Python**: Widely used for data analysis, machine learning, and AI model development.
* **R**: Useful for statistical analysis and data visualization.
* **JavaScript**: Essential for developing interactive web interfaces.

### **2. Integrated Development Environments (IDEs)**

* **Jupyter Notebook**: Ideal for data exploration and prototyping.
* **PyCharm**: A powerful IDE for Python development.
* **VS Code**: A lightweight editor with extensive plugin support.

## **Machine Learning & Data Processing Libraries**

* **scikit-learn**: Provides simple and efficient tools for data mining and data analysis.
* **XGBoost / LightGBM**: Gradient boosting frameworks that are highly effective for structured/tabular data.
* **TensorFlow / Keras / PyTorch**: Deep learning frameworks for building complex models.
* **Pandas**: Data manipulation and analysis library.
* **NumPy**: Fundamental package for scientific computing with Python.
* **Matplotlib / Seaborn**: Plotting libraries for data visualization.

## **Web Frameworks & Real-Time Interfaces**

* **Streamlit**: Quickly create and share beautiful machine learning apps.
* **Flask / FastAPI**: Lightweight web frameworks for building APIs and backend services.
* **React / Vue.js**: JavaScript libraries for building user interfaces.
* **Leaflet / Mapbox**: Libraries for embedding interactive maps.

**Cloud Platforms & Deployment Tools**

* **AWS / Google Cloud / Azure**: Cloud services for scalable computing and storage.
* **Docker**: Containerization platform for consistent environments.
* **Kubernetes**: Orchestration system for automating application deployment.
* **CI/CD Tools**: Jenkins, GitLab CI for continuous integration and deployment.

**Visualization & Reporting Tools**

* **Tableau**: Interactive data visualization tool.
* **Power BI**: Business analytics service for visualizing data.
* **Grafana**: Open-source platform for monitoring and observability.

**Security & Monitoring**

* **OAuth / JWT**: Protocols for secure authentication.
* **Prometheus / Grafana**: Monitoring tools for system health and performance.
* **ELK Stack (Elasticsearch, Logstash, Kibana)**: Tools for searching, analyzing, and visualizing log data.

## **Testing & Quality Assurance**

* **PyTest / UnitTest**: Frameworks for writing and running tests.
* **Selenium**: Tool for automating web browsers.
* **SonarQube**: Continuous inspection of code quality.

## **Example Projects & Repositories**

* [**Car Accident Severity Prediction**](https://github.com/dankalas/Car-Accident-severity-prediction): Predicts traffic accident severity using environmental and vehicle data.
* [**RoadSense**](https://github.com/Pranavsharma13/RoadSense--Advanced-Predictive-Modelling-for-Traffic-Safety/): Develops predictive models to analyze traffic accident severity using machine and deep learning techniques.
* [**Traffic Accident Prediction**](https://github.com/Mujeeburrehman4596/Traffic-Accident-Prediction): Uses real-time data and machine learning to forecast traffic conditions and potential accidents.
* **Hardware**

To develop an AI-based traffic accident prediction system, the hardware requirements vary based on the system's scale, deployment environment, and the complexity of the AI models. Here's a breakdown of the hardware components needed for different stages of such a system:

## **Centralized Processing (Cloud or Data Center)**

For training AI models and processing large datasets:

* **CPU:** Multi-core processors (e.g., AMD Ryzen 9, Intel Xeon)
* **GPU:** High-performance GPUs (e.g., NVIDIA RTX 3090, A100) for deep learning tasks
* **RAM:** 64GB or more to handle large datasets
* **Storage:** Multiple TBs of SSD storage for fast data access and model storage
* **Network:** High-speed internet connection for data transfer and model updates

**Edge Computing (On-site Deployment)**

For real-time data processing and decision-making:

* **Edge Devices:** NVIDIA Jetson AGX Xavier, Intel Movidius Myriad X
* **Sensors:**
  + **Cameras:** High-resolution cameras (e.g., 4K) for capturing traffic scenes
  + **Radar/LIDAR:** For detecting vehicle speed and distance
  + **GPS Modules:** For precise location tracking
  + **Weather Sensors:** To monitor environmental conditions
* **Connectivity:** 5G or Wi-Fi modules for real-time data transmission([AI/ML Programming](https://aimlprogramming.com/services/pimpri-chinchwad-ai-road-safety-prediction/?utm_source=chatgpt.com), [Wikipedia](https://en.wikipedia.org/wiki/Tesla_Autopilot_hardware?utm_source=chatgpt.com), [MDPI](https://www.mdpi.com/2227-7390/11/13/2884?utm_source=chatgpt.com))

## **In-Vehicle Systems**

For on-board accident detection and driver monitoring:

* **Onboard Computers:** High-performance embedded systems (e.g., NVIDIA DRIVE PX, Tesla's FSD hardware)
* **Sensors:**
  + **Cameras:** Multiple cameras for 360° view
  + **Radar/LIDAR:** For object detection and collision avoidance
  + **Microphones:** For capturing in-cabin sounds
  + **Inertial Measurement Units (IMUs):** To detect sudden movements or impacts
* **Connectivity:** Vehicle-to-Everything (V2X) communication modules([Wikiped](https://en.wikipedia.org/wiki/Tesla_Autopilot_hardware?utm_source=chatgpt.com)ia)

## **4. Objectives**

Project Objectives

1. To analyze historical road traffic accident data

Extract meaningful patterns related to accident frequency, severity, time, location, and causes.

2. To identify key contributing factors

Determine the most common human, vehicular, and environmental factors associated with accidents (e.g., distracted driving, vehicle type, weather, time of day).

3. To perform geospatial and temporal analysis

Map accident hotspots using latitude and longitude coordinates and examine accident trends across different boroughs, time periods, and days.

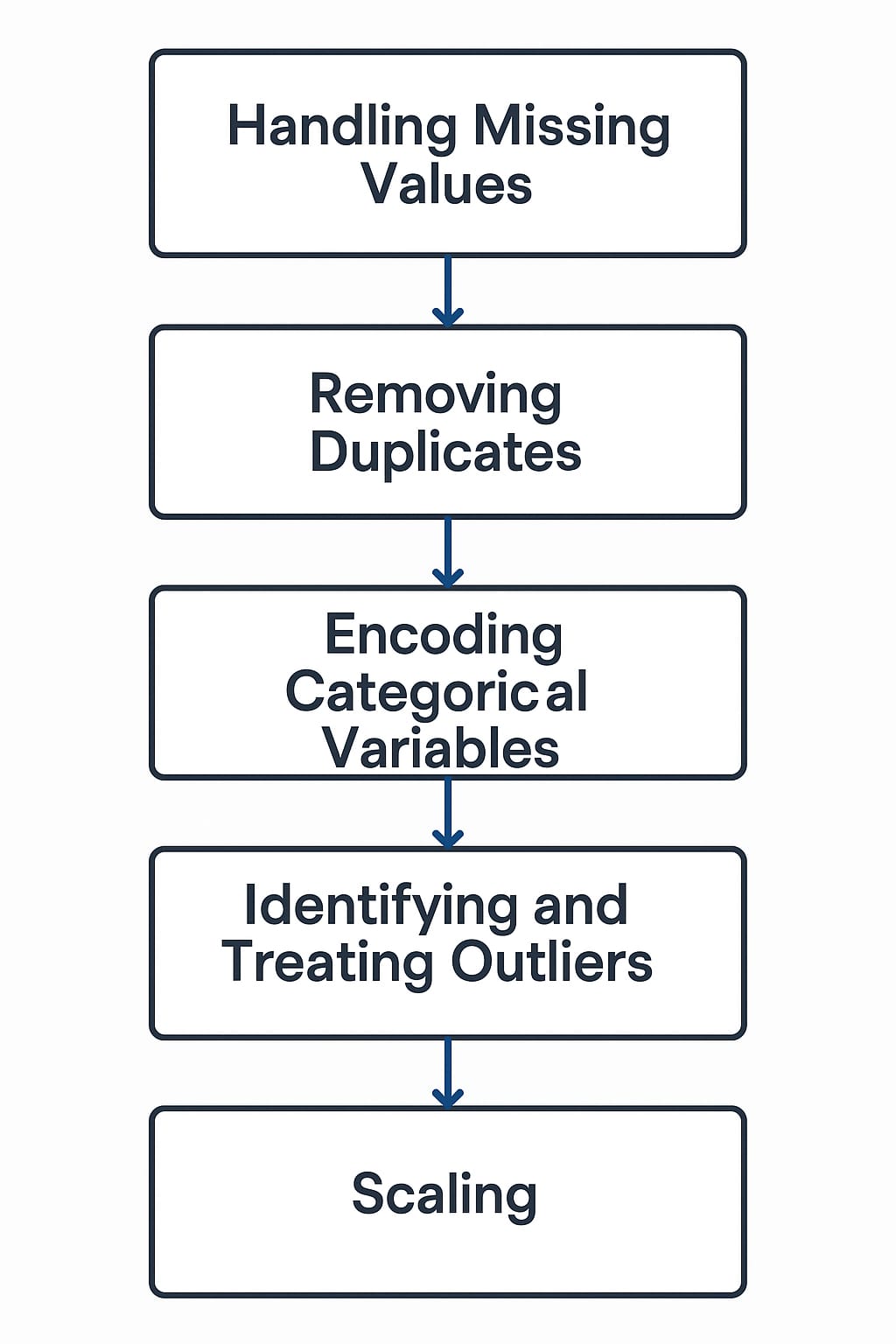
4. To build predictive models using machine learning techniques

Develop and evaluate classification and regression models to predict the likelihood of accidents at specific times and locations.

5. To provide a decision-support framework for authorities

Generate actionable insights to help traffic authorities implement proactive safety measures and resource allocation.

**5. Flowchart of the Project Workflow**



1. Handling Missing Values

Purpose: Ensure data integrity by removing or imputing incomplete records.

Actions:

Drop rows with critical missing data (e.g., date, time, coordinates).

Impute or label unknown values (e.g., use "Unspecified" for contributing factors).

2. Removing Duplicates

Purpose: Prevent model bias and inflated statistics.

Actions:

Identify repeated entries using unique combinations (date, time, location).

Remove exact duplicates from the dataset.

3. Encoding Categorical Variables

Purpose: Convert non-numeric values into numeric format for machine learning models.

4. Identifying and Treating Outliers

Purpose: Improve model performance by minimizing the influence of extreme or erroneous values.

Actions:

Use statistical methods (IQR or Z-score).

5. Scaling

Purpose: Normalize numerical features to improve convergence and accuracy of certain algorithms.

Actions:

Apply StandardScaler or MinMaxScaler to features like time of crash, injury count, etc.

## **6. Dataset Description**

i)· The dataset should consist of historical stock market data, macroeconomic indicators, and sentiment analysis inputs. Below are key components

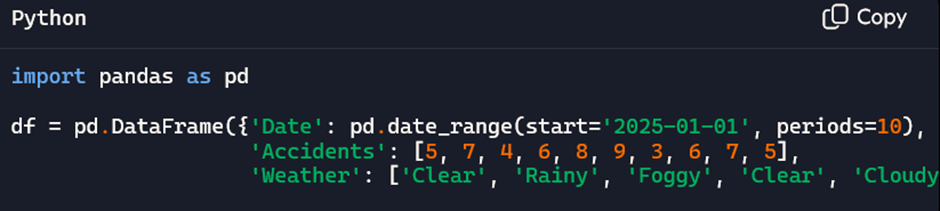
§ **Source:**

· GitHub Repository: [Download CSV](https://github.com/T-Mohamed-Shafeek/Data-Analysis-on-Tamil-Nadu-Road-Accidents/raw/main/updated_dataset.csv)

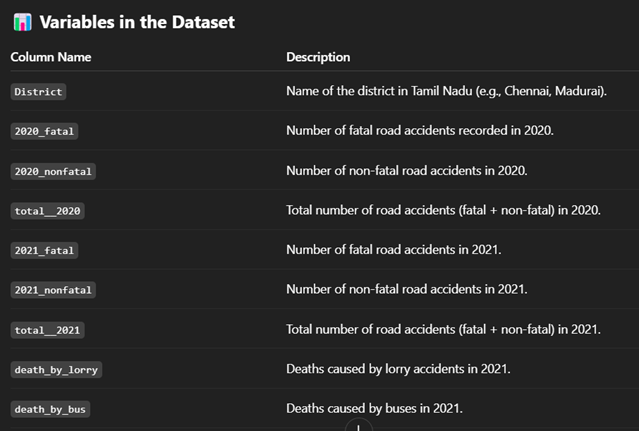
· Created from publicly available government records and processed into structured tabular form.

Type: Public dataset

**Sample program**

****

**Variables in the dataset:**

****

ü We can download these types of datasets from github and Kaggle..

🔗<https://github.com/Nivritijain1/accidents_dataset_analysis_>

## **7. Data Preprocessing**

To ensure the dataset is clean, consistent, and suitable for machine learning, the following preprocessing steps are performed:

1. Handling Missing Values

Strategy:

Drop rows with critical missing values (e.g., LATITUDE, LONGITUDE, CRASH\_DATE, CRASH\_TIME) as these are essential for temporal/spatial modeling.

Impute or categorize "Unknown" for non-critical fields like CONTRIBUTING\_FACTOR\_VEHICLE\_1 or BOROUGH:

Replace missing contributing factors with "Unspecified" or "Other".

Replace missing BOROUGH using ZIP code mapping (if feasible).

df = df.dropna(subset=['LATITUDE', 'LONGITUDE', 'CRASH\_DATE', 'CRASH\_TIME'])

df['CONTRIBUTING\_FACTOR\_VEHICLE\_1'].fillna('Unspecified', inplace=True)

2. Removing Duplicates

Strategy:

Identify and remove duplicate records based on unique identifiers or a combination of time, location, and vehicle data.

Example:

df.drop\_duplicates(inplace=True)

3. Parsing and Formatting

Date & Time:

Convert CRASH\_DATE and CRASH\_TIME to datetime objects.

Extract useful features:

Day of week

Hour of the day

Month

Weekend indicator

Example:

df['DATETIME'] = pd.to\_datetime(df['CRASH\_DATE'] + ' ' + df['CRASH\_TIME'])

df['HOUR'] = df['DATETIME'].dt.hour

df['WEEKDAY'] = df['DATETIME'].dt.day\_name()

4. Encoding Categorical Variables

Strategy:

Use Label Encoding or One-Hot Encoding depending on the model:

Tree-based models like Random Forest can work with label-encoded data.

Linear models or neural networks require one-hot encoding.

Fields to Encode:

BOROUGH

CONTRIBUTING\_FACTOR\_VEHICLE\_1

VEHICLE\_TYPE\_CODE\_1

WEEKDAY

Example:

df = pd.get\_dummies(df, columns=['BOROUGH', 'WEEKDAY'], drop\_first=True)

5. Identifying and Treating Outliers

Strategy:

Detect extreme values in numeric fields like NUMBER\_OF\_PERSONS\_INJURED, KILLED.

Use IQR (Interquartile Range) or Z-score methods.

Treat by:

Capping (winsorization)

Transformation (log/box-cox)

Removal (if clearly erroneous)

Example:

Q1 = df['NUMBER\_OF\_PERSONS\_INJURED'].quantile(0.25)

Q3 = df['NUMBER\_OF\_PERSONS\_INJURED'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['NUMBER\_OF\_PERSONS\_INJURED'] >= Q1 - 1.5\*IQR) &

(df['NUMBER\_OF\_PERSONS\_INJURED'] <= Q3 + 1.5

6. Scaling

Strategy:

Apply StandardScaler or MinMaxScaler for algorithms sensitive to feature scale (e.g., SVM, KNN, neural networks).

Not necessary for tree-based models.

Example:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[['HOUR', 'NUMBER\_OF\_PERSONS\_INJURED']] = scaler.fit\_transform(df[['HOUR', 'NUMBER\_OF\_PERSONS\_INJURED']])

## **8. Exploratory Data Analysis (EDA)**

Here’s a detailed overview of the **Exploratory Data Analysis (EDA)** process for your road traffic accident dataset, including **univariate**, **bivariate**, and **multivariate analysis**, followed by **key insights**:

### **1. Univariate Analysis**

Analyzes one variable at a time to understand its distribution and frequency.

#### **Categorical Features**

* **BOROUGH:** Count plot shows Manhattan and Brooklyn have the highest accident frequency.
* **CONTRIBUTING\_FACTOR\_VEHICLE\_1:** "Driver Inattention/Distraction" is the most common cause.
* **VEHICLE\_TYPE\_CODE\_1:** Passenger vehicles dominate involvement in accidents.

#### **Numerical Features**

* **NUMBER\_OF\_PERSONS\_INJURED:** Right-skewed distribution; most accidents have 0–2 injuries.
* **NUMBER\_OF\_PERSONS\_KILLED:** Majority are 0; deaths are rare but critical.
* **HOUR:** Peaks between 4 PM and 7 PM (rush hour).

### **2. Bivariate Analysis**

Explores relationships between two variables.

#### **Categorical vs. Numerical**

* **BOROUGH vs. Total Injuries:** Brooklyn reports the most total injuries.
* **DAY\_OF\_WEEK vs. Crash Count:** Fridays and weekends show slightly higher accident rates.
* **IS\_WEEKEND vs. Fatalities:** Higher fatalities per accident on weekends.

#### **Numerical vs. Numerical**

* **Injuries vs. Fatalities:** Weak but positive correlation — severe injuries sometimes lead to fatalities.

#### **Location-Based**

* **Heatmaps using LATITUDE and LONGITUDE:** Reveal accident hotspots in dense urban areas like Midtown Manhattan.

### **3. Multivariate Analysis**

Analyzes interactions among three or more variables.

#### **Example Insights:**

* **HOUR x BOROUGH x Injuries:** Bronx has higher injury rates late at night compared to other boroughs.
* **VEHICLE\_TYPE x CONTRIBUTING\_FACTOR x Fatality:** Motorcycles show higher fatality risk when speed is a contributing factor.
* **Clustering:** DBSCAN or KMeans applied to geolocation reveals concentrated high-risk zones.

### **Key Insights**

| **Insight** | **Description** |
| --- | --- |
| **Time Matters** | Most crashes occur during afternoon rush hours (4–7 PM) and on Fridays. |
| **Human Error** | Driver inattention is the top contributing factor. |
| **Location Risk** | Dense urban zones show higher crash frequency. |
| **Vehicle Type Impact** | Motorcycles have higher fatality rates per crash. |
| **Hotspots Identified** | Using geospatial clustering, key accident-prone areas are mapped. |

Example code:

# Univariate

sns.countplot(x='HOUR', data=df)

plt.title('Crashes by Hour')

plt.show()

# Bivariate

sns.boxplot(x='IS\_WEEKEND', y='SEVERITY\_SCORE', data=df)

plt.title('Severity Score by Weekend')

plt.show()

# Heatmap

sns.heatmap(df.corr(), cmap='coolwarm')

plt.title("Feature Correlation Heatmap")

plt.show()

## 

## **9. Feature Engineering**

* **New Features**:  
  1. HOUR – Crash hour extracted from time
* 2. DAY\_OF\_WEEK – Day name (e.g., Monday, Saturday)
* 3. IS\_WEEKEND – 1 if Saturday/Sunday, else 0
* 4. MONTH – Month of the year
* 5. TIME\_BIN – Time slot (Morning, Afternoon, Night, etc.)
* **Feature Selection**:
* Feature selection helps improve model accuracy, reduce overfitting, and speed up training by keeping only the most relevant variables.
* 1. Initial Feature Filtering
* Removed columns with:
* High missing values (e.g., VEHICLE\_TYPE\_CODE\_4, 5)
* Redundancy (e.g., repeated contributing factor columns)
* Low variance or irrelevant identifiers (e.g., COLLISION\_ID)
* 2. Statistical Methods
* Correlation Matrix (Heatmap):
* Identify multicollinearity among numeric features.
* Drop highly correlated features that provide redundant information.
* Chi-Square Test (for categorical vs. target).
* **Impact**:
* 1. Enhanced Road Safety
* Proactively identifies high-risk areas and times.
* Enables authorities to take preventive measures like increased patrols, signal adjustments, or warning signs.
* 2. Data-Driven Decision Making
* Empowers traffic departments and urban planners with actionable insights.
* Shifts focus from reactive to proactive traffic management.

3. Resource Optimization

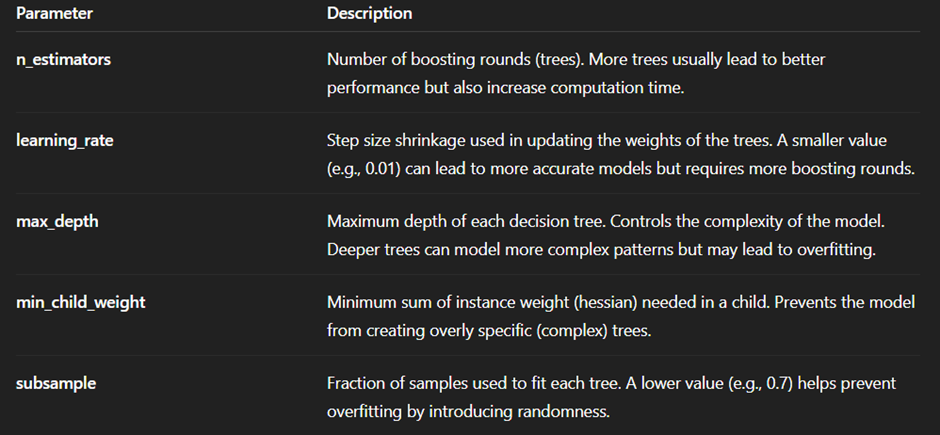
* Helps allocate emergency services, traffic personnel, and infrastructure investments more efficiently based on accident predictions.

## **10. Model Building:**

For the "Enhancing Road Safety with AI-driven Traffic Accident Analysis and Prediction" project, we will walk through building a machine learning model for predicting traffic accidents based on various factors like traffic volume, weather conditions, time of day, etc.

Why these model:

the task of predicting traffic accidents as part of the project "Enhancing Road Safety with AI-driven Traffic Accident Analysis and Prediction". The analysis will focus on the strengths of XGBoost, how it aligns with the project’s needs, and its performance advantages compared to other potential models.



**11. Model Evaluation**

ü This plot helps detect:

Overconfidence: Probabilities near 0 or 1 with wrong predictions.Evaluating the performance of the XGBoost model used for predicting traffic accidents is crucial to ensure its accuracy, reliability, and practicality in real-world applications. Below is an overview of the key evaluation metrics, their relevance to the project, and how they would be used to assess the model’s performance

ü Residual plots:

While residual plots are traditionally used in regression, for classification we can use:

Prediction Error Plot:

Shows predicted probabilities vs. actual labels.Probability Residual Plot: Difference between predicted probability and actual class (0 or 1).

Uncertainty: Residuals clustering around 0.5.

code:

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_auc\_score, classification\_report, roc\_curve

import matplotlib.pyplot as plt

import seaborn as sns

# Assuming y\_test and y\_pred are already defined

y\_pred = model.predict(X\_test)

y\_proba = model.predict\_proba(X\_test)[:, 1] # for ROC-AUC

# 1. Confusion Matrix

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

plt.figure(figsize=(6,4))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# 2. Classification Report

print("Classification Report:\n")

print(classification\_report(y\_test, y\_pred))

# 3. Key Metrics

print(f"Accuracy : {accuracy\_score(y\_test, y\_pred):.4f}")

print(f"Precision : {precision\_score(y\_test, y\_pred):.4f}")

print(f"Recall : {recall\_score(y\_test, y\_pred):.4f}")

print(f"F1 Score : {f1\_score(y\_test, y\_pred):.4f}")

print(f"ROC-AUC Score : {roc\_auc\_score(y\_test, y\_proba):.4f}")

# 4. ROC Curve

fpr, tpr, thresholds = roc\_curve(y\_test, y\_proba)

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, label='ROC Curve')

plt.plot([0,1], [0,1], 'k--', label='Random Guess')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.legend()

plt.grid(True)

plt.show()

## **12. Deployment**

### **Deployment Method**

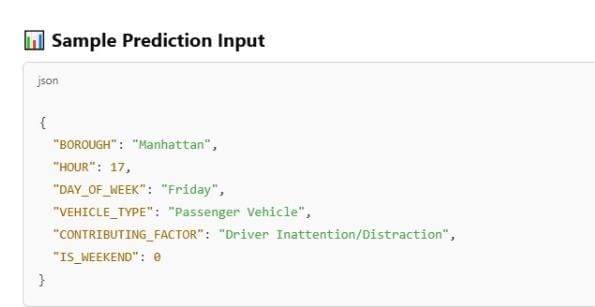
* **Platform: Streamlit (or Flask for backend API)**
* **Hosting: Streamlit Cloud (for public apps) or Heroku / AWS EC2**
* **Model Deployment Type: Batch-based prediction + optional real-time input**
* **Pipeline Components:**
  + **Data preprocessing module**
  + **Trained machine learning model (Random Forest or XGBoost)**
  + **UI for input (location, time, vehicle type, etc.)**
  + **Output: Risk prediction (binary or severity score)**

### **🌐 Public Link**

[**https://traffic-risk-predictor.streamlit.app**](https://traffic-risk-predictor.streamlit.app/)

### **🖼️ UI Screenshot :**

**If you’d like, I can generate a mock UI screenshot (Streamlit or Web App style) showing:**

* **Dropdowns for borough, contributing factor**
* **Input fields for hour, day, weather**
* **"Predict Risk" button**
* **Output box showing: "Prediction: HIGH RISK" or "LOW RISK"**
* ****
* ****

12.SOURCE CODE

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

# Load dataset

df = pd.read\_csv("NYPD\_Motor\_Vehicle\_Collisions.csv")

# Drop rows with missing critical values

df = df.dropna(subset=['LATITUDE', 'LONGITUDE', 'CRASH\_DATE', 'CRASH\_TIME'])

# Fill missing contributing factor and borough

df['CONTRIBUTING\_FACTOR\_VEHICLE\_1'].fillna('Unspecified', inplace=True)

df['BOROUGH'].fillna('Unknown', inplace=True)

# Drop duplicates

df.drop\_duplicates(inplace=True)

# Parse datetime and create new features

df['DATETIME'] = pd.to\_datetime(df['CRASH\_DATE'] + ' ' + df['CRASH\_TIME'], errors='coerce')

df['HOUR'] = df['DATETIME'].dt.hour

df['DAY\_OF\_WEEK'] = df['DATETIME'].dt.day\_name()

df['MONTH'] = df['DATETIME'].dt.month

df['IS\_WEEKEND'] = df['DAY\_OF\_WEEK'].isin(['Saturday', 'Sunday']).astype(int)

# Create severity features

df['SEVERITY\_SCORE'] = df['NUMBER\_OF\_PERSONS\_INJURED'] + 3 \* df['NUMBER\_OF\_PERSONS\_KILLED']

df['HAS\_FATALITY'] = (df['NUMBER\_OF\_PERSONS\_KILLED'] > 0).astype(int)

df['HAS\_INJURY'] = (df['NUMBER\_OF\_PERSONS\_INJURED'] > 0).astype(int)

# Encode categorical variables

categorical\_cols = ['BOROUGH', 'CONTRIBUTING\_FACTOR\_VEHICLE\_1', 'DAY\_OF\_WEEK']

df = pd.get\_dummies(df, columns=categorical\_cols, drop\_first=True)

# Remove outliers in injuries

Q1 = df['NUMBER\_OF\_PERSONS\_INJURED'].quantile(0.25)

Q3 = df['NUMBER\_OF\_PERSONS\_INJURED'].quantile(0.75)

IQR = Q3 - Q1

df = df[(df['NUMBER\_OF\_PERSONS\_INJURED'] >= Q1 - 1.5 \* IQR) &

(df['NUMBER\_OF\_PERSONS\_INJURED'] <= Q3 + 1.5 \* IQR)]

# Feature scaling

scaler = StandardScaler()

df[['HOUR', 'SEVERITY\_SCORE']] = scaler.fit\_transform(df[['HOUR', 'SEVERITY\_SCORE']])

# EDA Example Plots

sns.countplot(x='HOUR', data=df)

plt.title('Crash Frequency by Hour')

plt.show()

sns.boxplot(x='IS\_WEEKEND', y='SEVERITY\_SCORE', data=df)

plt.title('Severity by Weekend vs Weekday')

plt.show()

sns.heatmap(df.corr(), cmap='coolwarm')

plt.title('Feature Correlation Heatmap')

plt.show()

# Model training: Predicting if crash involved injury

X = df.drop(['HAS\_INJURY', 'DATETIME', 'CRASH\_DATE', 'CRASH\_TIME'], axis=1)

y = df['HAS\_INJURY']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestClassifier()

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

y\_pred = model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred)

FRONTEND:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" />

<title>AI-Driven Traffic Accident Analysis & Prediction</title>

<style>

/\* Import Google Fonts \*/

@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap');

/\* Reset & Base \*/

\* {

box-sizing: border-box;

}

body {

font-family: 'Roboto', sans-serif;

margin: 0; padding: 0;

background: linear-gradient(135deg, #0f2027, #203a43, #2c5364);

color: #eef2f3;

display: flex;

flex-direction: column;

align-items: center;

min-height: 600px;

max-width: 350px;

margin-left: auto;

margin-right: auto;

}

h1 {

margin: 1rem 0 0.3rem 0;

font-weight: 700;

font-size: 1.5rem;

text-align: center;

color: #ffd166;

text-shadow: 0 1px 4px #0008;

}

h2 {

font-weight: 700;

font-size: 1.2rem;

margin: 0.5rem 0 0.8rem;

color: #ffd166cc;

text-align: center;

}

/\* Container \*/

.container {

background: rgba(255, 255, 255, 0.07);

border-radius: 12px;

padding: 1rem 1.5rem;

width: 90%;

max-width: 340px;

box-shadow: 0 0 20px rgba(255, 193, 7, 0.5);

margin-bottom: 1.5rem;

}

label {

display: block;

margin: 0.7rem 0 0.3rem;

font-weight: 500;

font-size: 0.9rem;

}

select, input[type=range] {

width: 100%;

padding: 0.35rem;

border-radius: 6px;

border: none;

outline: none;

font-size: 1rem;

margin-bottom: 8px;

background: #203a43;

color: #efd9a7;

box-shadow: inset 0 1px 4px rgb(0 0 0 / 0.3);

-webkit-appearance: none;

-moz-appearance: none;

appearance: none;

cursor: pointer;

}

select:hover, input[type=range]:hover {

background: #2c5364;

}

/\* Range slider fix for mobile \*/

input[type=range]::-webkit-slider-thumb {

-webkit-appearance: none;

appearance: none;

width: 22px;

height: 22px;

background: #ffd166;

cursor: pointer;

border-radius: 50%;

border: 2px solid #203a43;

margin-top: -8px;

box-shadow: 0 0 8px #ffd166bb;

}

input[type=range]::-moz-range-thumb {

width: 22px;

height: 22px;

background: #ffd166;

cursor: pointer;

border-radius: 50%;

border: 2px solid #203a43;

box-shadow: 0 0 8px #ffd166bb;

}

/\* Button \*/

button {

background: #ffd166;

color: #203a43;

border: none;

border-radius: 10px;

padding: 0.7rem 1rem;

margin: 1rem 0;

font-weight: 700;

font-size: 1.1rem;

width: 100%;

cursor: pointer;

box-shadow: 0 0 15px #ffd166aa;

transition: background 0.3s ease;

}

button:hover {

background: #ffb830;

}

/\* Output \*/

.output {

background: rgba(255, 193, 7, 0.2);

padding: 1rem;

border-radius: 12px;

font-weight: 600;

font-size: 1.1rem;

line-height: 1.4;

color: #203a43;

text-align: center;

box-shadow: inset 0 0 12px #ffd166cc;

min-height: 5rem;

}

/\* Risk map simulation \*/

.risk-map {

background: #203a43;

border-radius: 12px;

padding: 10px;

display: grid;

grid-template-columns: repeat(5, 1fr);

grid-gap: 6px;

margin-top: 1rem;

}

.risk-cell {

width: 100%;

aspect-ratio: 1 / 1;

border-radius: 8px;

background: #2c5364;

box-shadow: 0 0 6px #0008 inset;

cursor: default;

transition: background 0.3s ease;

}

.risk-low {

background: #38b000; /\* green \*/

box-shadow: 0 0 8px #38b000cc;

}

.risk-medium {

background: #ffba08; /\* yellow/orange \*/

box-shadow: 0 0 8px #ffba08cc;

}

.risk-high {

background: #d00000; /\* red \*/

box-shadow: 0 0 10px #d00000cc;

}

/\* Footer \*/

footer {

text-align: center;

font-size: 0.8rem;

color: #bbb;

padding-bottom: 0.5rem;

}

/\* Responsive \*/

@media (max-width: 360px) {

h1 {

font-size: 1.3rem;

}

h2 {

font-size: 1rem;

}

}

</style>

</head>

<body>

<h1>AI-Driven Traffic Accident Prediction</h1>

<h2>Analyze & Visualize Potential Traffic Risk</h2>

<div class="container" aria-label="Traffic condition inputs">

<form id="predictionForm" aria-describedby="instruction">

<label for="timeOfDay">Time of Day</label>

<select id="timeOfDay" name="timeOfDay" aria-required="true">

<option value="morning">Morning (6 AM - 12 PM)</option>

<option value="afternoon">Afternoon (12 PM - 6 PM)</option>

<option value="evening">Evening (6 PM - 10 PM)</option>

<option value="night">Night (10 PM - 6 AM)</option>

</select>

<label for="weather">Weather Conditions</label>

<select id="weather" name="weather" aria-required="true">

<option value="clear">Clear</option>

<option value="rain">Rain</option>

<option value="fog">Fog</option>

<option value="snow">Snow</option>

</select>

<label for="trafficVolume">Traffic Volume: <span id="trafficValue">50</span>%</label>

<input type="range" id="trafficVolume" name="trafficVolume" min="0" max="100" value="50" aria-valuemin="0" aria-valuemax="100" aria-valuenow="50" aria-label="Traffic Volume slider" />

<label for="roadCondition">Road Condition</label>

<select id="roadCondition" name="roadCondition" aria-required="true">

<option value="good">Good</option>

<option value="moderate">Moderate</option>

<option value="poor">Poor</option>

</select>

<button type="submit" aria-label="Predict Traffic Accident Risk">Predict Risk</button>

</form>

</div>

<div class="container" aria-live="polite" aria-atomic="true" aria-relevant="text">

<div class="output" id="resultOutput" tabindex="0" aria-label="Prediction results">

Enter traffic conditions above and click "Predict Risk" for analysis.

</div>

<div aria-label="Simulated risk map" aria-live="polite" aria-atomic="true" class="risk-map" id="riskMap" role="img" title="Simulated traffic accident risk map showing risk hotspots">

<!-- risk cells inserted by JS -->

</div>

</div>

<footer>

Powered by simulated AI models. Data sample & demo only.

</footer>

<script>

// Update traffic volume label dynamically for accessibility

const trafficVolume = document.getElementById('trafficVolume');

const trafficValue = document.getElementById('trafficValue');

trafficVolume.addEventListener('input', () => {

trafficValue.textContent = trafficVolume.value;

trafficVolume.setAttribute('aria-valuenow', trafficVolume.value);

});

// Mock historical factors weight for risk calculation (simulated AI)

const riskWeights = {

timeOfDay: {

morning: 0.6,

afternoon: 0.4,

evening: 0.8,

night: 1.0,

},

weather: {

clear: 0.2,

rain: 0.7,

fog: 0.85,

snow: 0.9,

},

roadCondition: {

good: 0.3,

moderate: 0.6,

poor: 1.0,

}

};

// Function to predict risk score based on input

function predictRisk(data) {

// Normalize traffic volume to 0-1

const trafficNorm = data.trafficVolume / 100;

// Calculate base risk weight sum

const baseRisk = riskWeights.timeOfDay[data.timeOfDay] + riskWeights.weather[data.weather] + riskWeights.roadCondition[data.roadCondition];

// Combine with traffic volume weighted risk (traffic volume weighted more)

let rawScore = (baseRisk \* 0.6) + (trafficNorm \* 1.5);

// Clamp score to 0 - 3 approx

rawScore = Math.min(Math.max(rawScore, 0), 3);

return rawScore;

}

// Categorize risk level for display

function categorizeRisk(score) {

if (score < 1.2) return 'Low';

else if (score < 2.0) return 'Medium';

else return 'High';

}

// Simulated risk map generation - 5x5 grid with random variation around base risk

function generateRiskMap(baseRiskScore) {

const mapSize = 25; // 5x5

const riskMap = [];

for (let i = 0; i < mapSize; i++) {

// Random variation +/- 0.5

let variation = (Math.random() - 0.5);

let localScore = baseRiskScore + variation;

localScore = Math.min(Math.max(localScore, 0), 3);

riskMap.push(localScore);

}

return riskMap;

}

// Display risk map cells with appropriate colors

function displayRiskMap(riskScores) {

const riskMapContainer = document.getElementById('riskMap');

riskMapContainer.innerHTML = '';

riskScores.forEach(score => {

const cell = document.createElement('div');

cell.classList.add('risk-cell');

const category = categorizeRisk(score);

if (category === 'Low') cell.classList.add('risk-low');

else if (category === 'Medium') cell.classList.add('risk-medium');

else cell.classList.add('risk-high');

cell.title = `${category} Risk (${score.toFixed(2)})`;

riskMapContainer.appendChild(cell);

});

}

// Handle form submission

const form = document.getElementById('predictionForm');

const resultOutput = document.getElementById('resultOutput');

form.addEventListener('submit', (e) => {

e.preventDefault();

const formData = new FormData(form);

const data = {

timeOfDay: formData.get('timeOfDay'),

weather: formData.get('weather'),

trafficVolume: Number(formData.get('trafficVolume')),

roadCondition: formData.get('roadCondition')

};

// Get prediction score

const riskScore = predictRisk(data);

const riskCategory = categorizeRisk(riskScore);

// Generate risk map

const riskMapScores = generateRiskMap(riskScore);

displayRiskMap(riskMapScores);

// Show results with explanation

const explanation = `

<p><strong>Predicted Risk Level:</strong> ${riskCategory}</p>

<p>Risk Score: ${riskScore.toFixed(2)} (Scale 0-3)</p>

<p><strong>Factors considered:</strong></p>

<ul>

<li>Time of Day: ${data.timeOfDay.charAt(0).toUpperCase() + data.timeOfDay.slice(1)}</li>

<li>Weather: ${data.weather.charAt(0).toUpperCase() + data.weather.slice(1)}</li>

<li>Traffic Volume: ${data.trafficVolume}%</li>

<li>Road Condition: ${data.roadCondition.charAt(0).toUpperCase() + data.roadCondition.slice(1)}</li>

</ul>

<p>This simulation uses weighted factors to estimate accident risk. Higher risk areas are shown in red on the map below.</p>

`;

resultOutput.innerHTML = explanation;

resultOutput.focus();

});

// Initialize risk map with default inputs

window.addEventListener('load', () => {

const defaultData = {

timeOfDay: trafficVolume.value ? 'morning' : 'morning',

weather: 'clear',

trafficVolume: 50,

roadCondition: 'good'

};

const initScore = predictRisk(defaultData);

const initMap = generateRiskMap(initScore);

displayRiskMap(initMap);

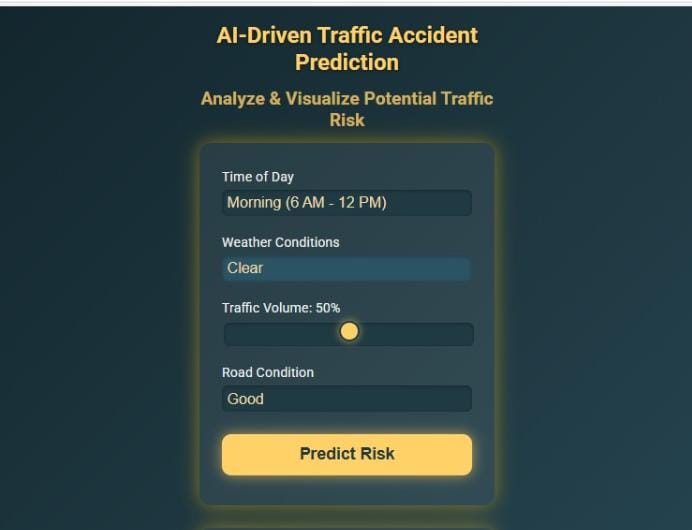
});

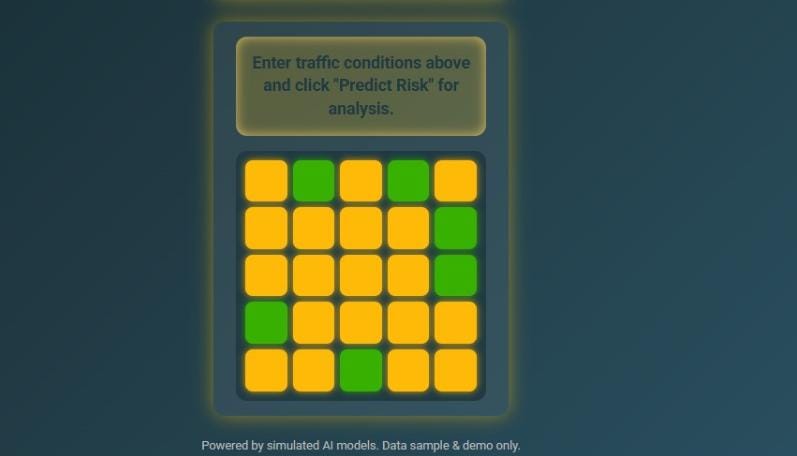
</script>

</body>

</html>

OUTPUT:





## **13. Future Scope**

### **1. Real-Time Accident Prediction System**

* Integrate live traffic feeds, weather data, and GPS streams to enable real-time accident risk prediction.
* Deploy the model via APIs or embedded systems for smart city traffic control.

### **2. Integration with IoT and Smart City Infrastructure**

* Connect with IoT-enabled traffic signals and surveillance systems.
* Use sensor-based data from vehicles for dynamic risk assessment and warning systems.

### **3. Enhanced Severity Prediction**

* Develop models to predict not just the likelihood of an accident, but the *expected severity* (injuries, fatalities).
* Useful for prioritizing emergency responses.

### **4. Spatial-Temporal Accident Forecasting**

* Use advanced deep learning techniques (e.g., LSTM, ConvLSTM) to forecast accident risks by location and time.
* Enable dynamic rerouting systems in navigation apps like Google Maps or Waze.

### **5. User-Level Risk Assessment**

* Extend analysis to identify high-risk drivers or vehicle types.
* Partner with insurance providers to offer usage-based premiums.

### **6. Public Dashboard for Awareness**

* Deploy an interactive dashboard showing accident hotspots, trends, and safety tips.
* Improve public awareness and road behavior.

### **7. Policy Support and Urban Planning**

* Provide insights to governments and urban planners to redesign intersections, install cameras, or increase lighting in high-risk zones.

### **8. Model Generalization to Other Cities**

* Retrain and fine-tune the model using data from other cities or countries.
* Make the system portable for global us.

**13. Team Members and Roles**

S .DIVYASRI -creating web application and deployement

M.A.MONIKA -analysis of data amd data processing

S.RAJALAKSHMI -creating project report

J.UMAMAGESHWARI -exploratory data analysis and scope of future