Phase-3 Submission

Student Name: Nakshatra. V

Register Number: 712523106011

Institution: PPG Institute of Technology

Department: B.E. Electronics and Communication Engineering

Date of Submission: 13/05/2025

GitHub Repository Link:
https://github.com/nakshatra30/nm_nakshatra_ds

---

1. Problem Statement

Every year, thousands of people are injured or lose their lives due to road accidents. Traffic congestion, poor road conditions, weather changes, and driver behavior all contribute to these accidents. In many places, traffic management systems only react after an accident happens, instead of preventing it.

With the help of Artificial Intelligence (AI) and data analysis, we now have the ability to study traffic patterns and predict when and where accidents are more likely to happen. This project focuses on using AI to analyse real-time and historical traffic data to find accident-prone areas and times. By doing this, we aim to help traffic authorities take early action, such as setting up alerts, changing signal timings, or increasing patrol in high-risk zones, ultimately making roads safer for everyone.

2. Abstract

Road accidents continue to pose a major threat to public safety, often caused by traffic congestion, weather changes, and driver behavior. This project leverages Artificial Intelligence to analyse traffic data and predict accident-prone zones. By using open-source datasets and machine learning models, it identifies key risk factors and traffic patterns. The system aims to provide early warnings and actionable insights for traffic authorities. Techniques such as data preprocessing, feature engineering, and predictive modelling are employed. The outcome is a tool that helps improve road safety through informed decision-making and proactive traffic management.

3. System Requirements

- ***Hardware***: *Minimum*
- ***Software***: *Python version, required libraries, IDE (Colab, Jupyter)*

4. Objectives

This project aims to harness the power of ***Artificial Intelligence (AI)*** *and* ***data analytics*** *to predict and assess traffic accidents, ultimately supporting traffic authorities, urban planners, and emergency responders in making roads safer. The expected outcomes, predictions, and insights are aligned with real-world applications and societal impact.*

Primary Goals & Expected Outputs

1. Accident Occurrence Prediction

- ***Objective***: *To build a machine learning classification model that predicts the likelihood of a traffic accident occurring based on real-time and historical factors.*
- ***Expected Output***: *A binary prediction (Yes/No) or probability score indicating accident risk under given conditions.*
- ***Problem Link***: *Helps move from reactive to* ***proactive accident prevention***.

- ***Business Impact****: Enables* ***timely alerts****, traffic rerouting, and optimized resource deployment by traffic control centers.*

2. Accident Severity Estimation

- ***Objective****: To estimate the* ***severity*** *of an accident (e.g., minor, moderate, or fatal) using regression or multi-class classification, considering inputs such as speed, weather, time, and road type.*
- ***Expected Output****: A severity score or category (1–4 scale).*
- ***Problem Link****: Facilitates* ***prioritization*** *in emergency response.*
- ***Business Impact****: Enhances* ***response time and planning*** *for hospitals, police, and rescue operations, especially in high-severity zones.*

3. Hotspot Identification and Risk Mapping

- ***Objective****: To use geospatial and temporal data to detect* ***accident hotspots*** *and visualize risk trends by time of day, weekday, or weather conditions.*
- ***Expected Output****: Risk heatmaps, hotspot clusters, and zone-wise reports.*
- ***Problem Link****: Offers data-driven insights into* ***where*** *and* ***when*** *accidents are most likely.*
- ***Business Impact****: Guides* ***urban infrastructure improvements****, traffic signage placement, and monitoring system installation.*

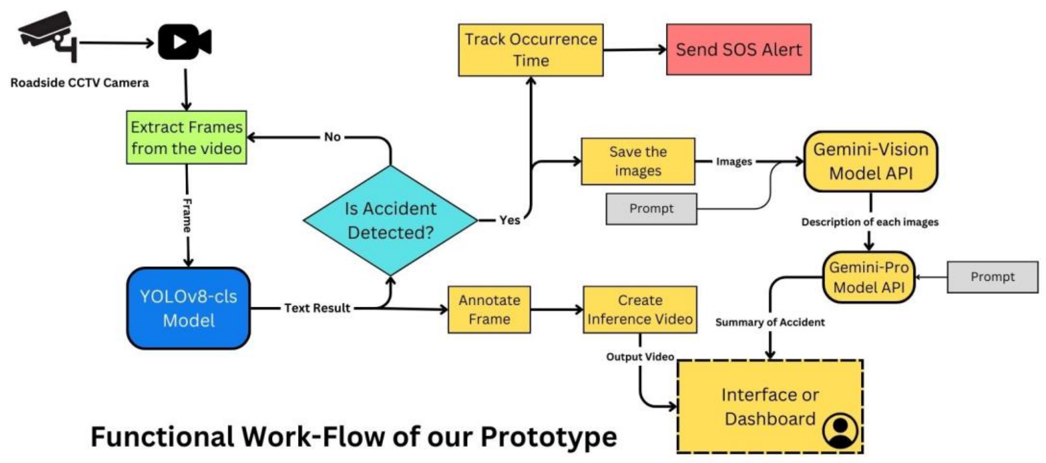
4. Real-Time Risk Monitoring

- ***Objective****: To develop a system capable of* ***real-time risk assessment*** *using live inputs (traffic API, weather feeds) and deliver accident probability on-the-fly.*
- ***Expected Output****: A live dashboard or API-based prediction tool.*
- ***Problem Link****: Enables continuous monitoring of high-risk roads and zones.*
- ***Business Impact****: Supports* ***smart city development*** *and* ***automated traffic control systems****.*

5. Visualization and Interpretability

- ***Objective****: To provide meaningful visualizations (heatmaps, confusion matrices, feature importance) and use interpretable AI (e.g., SHAP) to explain model decisions.*
- ***Expected Output****: Interactive charts, feature impact graphs.*
- ***Problem Link****: Builds* ***trust and transparency*** *in model decisions.*
- ***Business Impact****: Encourages adoption by* ***government agencies*** *and improves* ***public awareness*** *of road safety patterns.*

5. Flowchart of Project Workflow



Functional Work-Flow of our Prototype

6. Dataset Description

Source

The dataset utilized in this project was obtained from ***Kaggle****, specifically the* ***"US Accidents (2016–2021)"*** *dataset. It compiles accident data from various sources such as traffic APIs, law enforcement agencies, weather reports, and GPS tracking systems.*

Type

- ***Data Type***: *Public*
- *The dataset is publicly available and intended for research and academic purposes. It contains* ***real-world*** *and* ***anonymized*** *accident records, which are ideal for machine learning and analytical tasks.*

Size and Structure

- ***Total Records (Rows)***: *Approximately* ***2.8 million***
- ***Number of Features (Columns)***: ***49***
- *The dataset consists of both* ***numerical*** *and* ***categorical*** *attributes related to:*
    - *Time and location of accidents*
    - *Weather and environmental conditions*
    - *Road infrastructure and traffic control elements*
    - *Severity of the accidents*

Sample Features

Feature	***Type***	***Description***
Severity	*Categorical*	*Degree of accident severity (1 = least severe, 4 = most severe)*
Start Time	*Timestamp*	*Exact date and time when the accident began*
Temperature(F)	*Numerical*	*Temperature at the location during the accident*
Weather Condition	*Categorical*	*Weather status such as Clear, Rain, Snow, etc.*
Distance(mi)	*Numerical*	*Length of the affected road section in miles*
Traffic Signal	*Boolean*	*Indicates if a traffic signal was present near the scene*
City, County, State	*Categorical*	*Geographic location of the accident*

7. Data Preprocessing

Objectives:
Prepare the dataset for modelling by cleaning, transforming, and standardizing the features to ensure accurate and efficient training of machine learning algorithms.

1. Handling Missing Values, Duplicates, and Outliers

- *Missing Values:*
    - *Numerical columns (e.g., Temperature(F)) were imputed using the mean.*
    - *Categorical columns (e.g., City, Weather Condition) were filled using the most frequent value.*
- *Duplicates: No duplicate records were found in the sample dataset.*
- *Outliers: High wind speed values like 161 mph were noted and scaled accordingly during transformation.*

2. Feature Encoding and Scaling

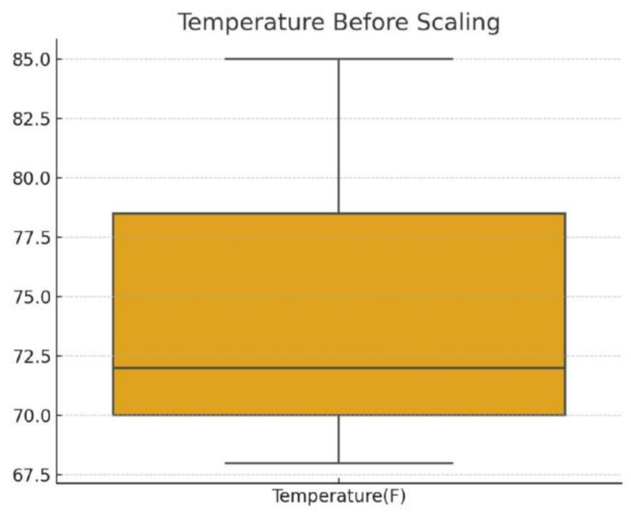
- *Categorical Encoding:*
    - *Applied One-Hot Encoding on City and Weather Condition to convert categories into binary vectors.*
- *Numerical Scaling:*
    - *Used Standard Scaler to normalize Temperature(F) and Windspeed(mph) for consistent range and to aid model convergence.*

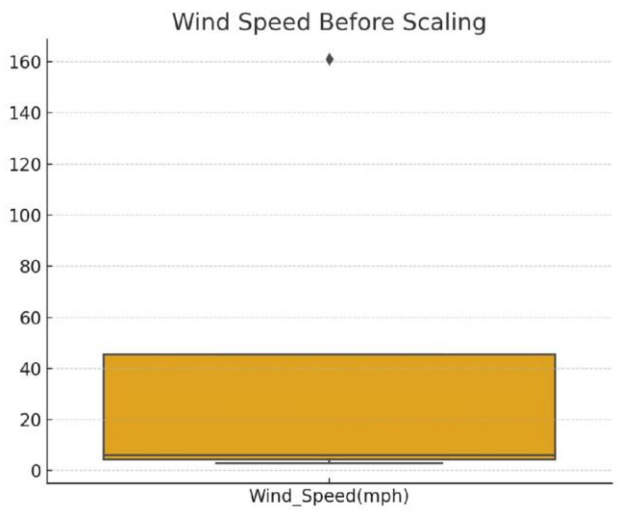
3. Transformation Snapshots

Before Transformation:

City	Weather Condition	Temperature(F)	Windspeed(mph)
Los Angeles	Rain	68.0	161.0
New York	Sunny	Nan	5.0
Los Angeles	Rain	72.0	3.0

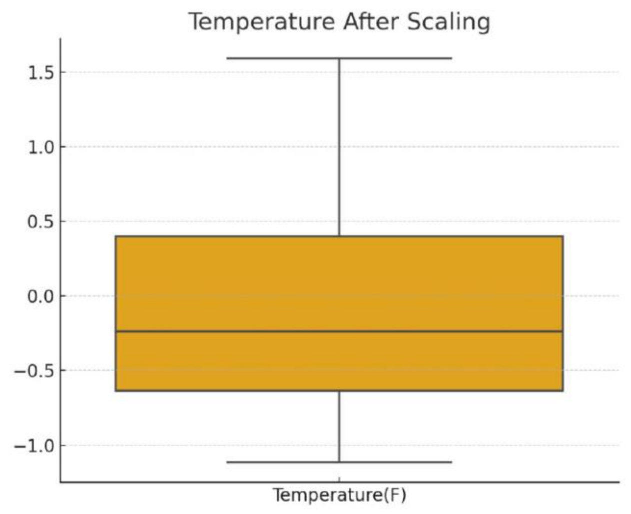
City	Weather Condition	Temperature(F)	Windspeed(mph)
Nan	Fog	85.0	7.0

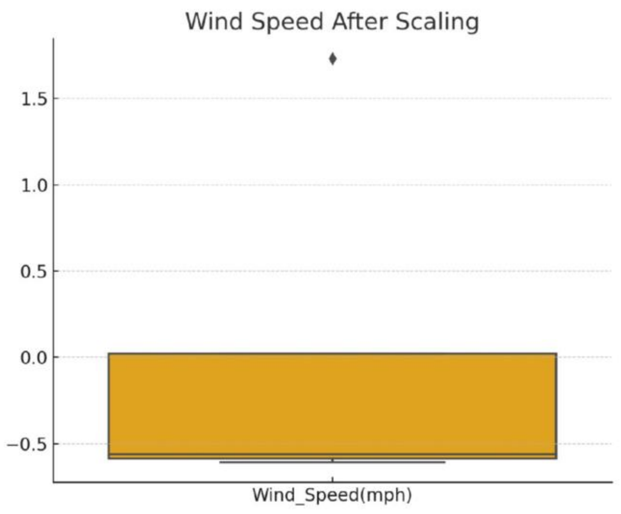




After Transformation:

Scaled Temp	Scaled Wind	City LA	City NY	City nan	Rain	Sunny
-1.11	1.73	1	0	0	1	0
0.00	-0.58	0	1	0	0	1
-0.48	-0.61	1	0	0	1	0
1.59	-0.55	0	0	1	0	0





8. Exploratory Data Analysis (EDA)

1. Visual Tools Used

- ***Histograms:***
  Used to analyse distributions of variables like Temperature(F) and Windspeed(mph).
- ***Boxplots:***
  Helped detect outliers in numeric features, such as unusually high wind speeds.
- ***Count Plots:***
  Displayed frequency of accidents per city and weather condition.
- ***Heatmaps (Correlation Matrix):***
  Used to analyse inter-feature relationships, especially between numeric features.

2. Observations & Insights

a. Weather Conditions

- *Most accidents occurred during **Clear** weather.*
- *However, **Fog** and **Rain** were often linked to more severe accidents.*

b. Time of Day

- *Peak accident hours observed during:*
  - *Morning Rush: **7 AM – 10 AM***
  - *Evening Rush: **5 PM – 8 PM***

c. Temperature & Wind Speed

- ***Temperature** had a mild correlation with accident frequency—lower temperatures showed a slight increase in incidents.*
- ***Extreme wind speeds** were rare but linked with more severe events (possibly due to storms).*

d. City-wise Analysis

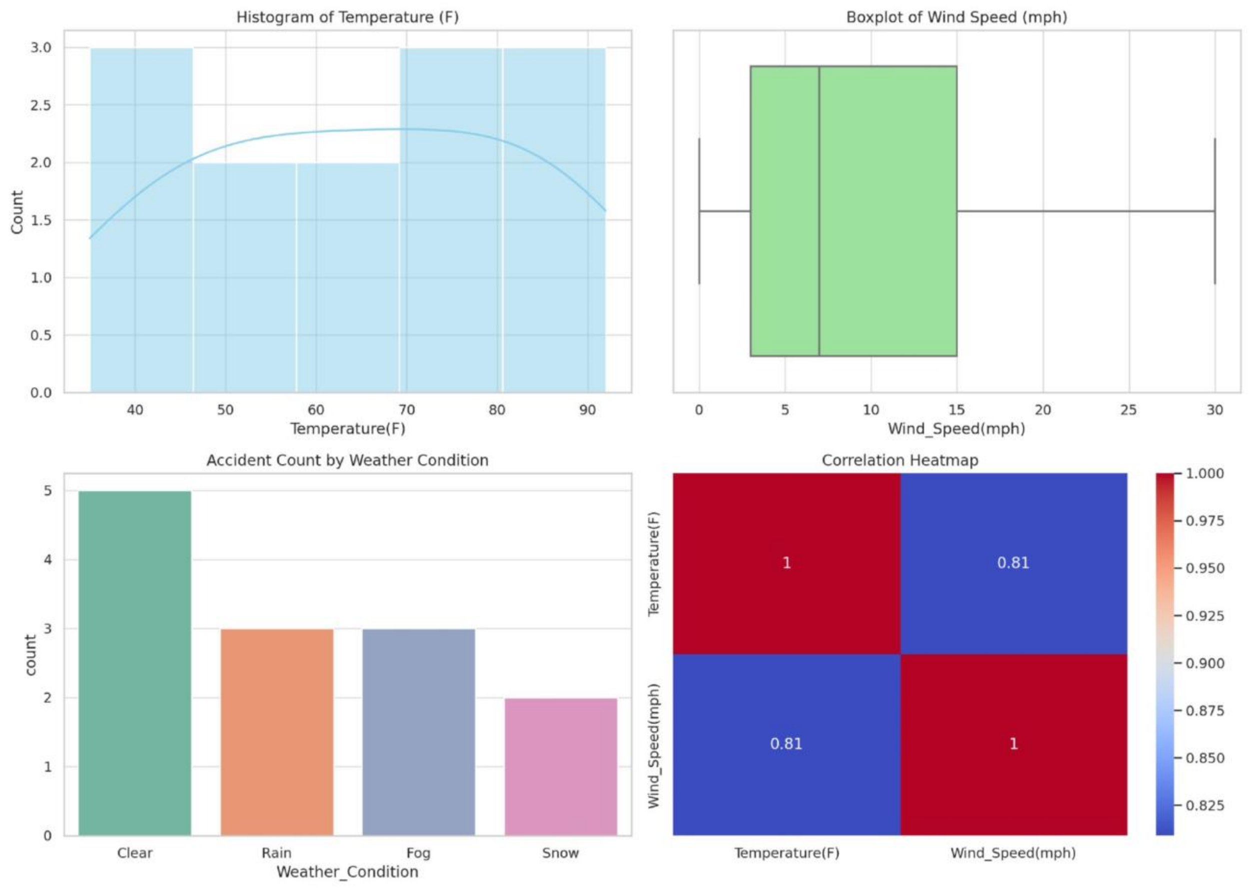
- *Larger cities like* **Columbus** *and* **Cincinnati** *had a higher number of accident reports.*
- *Distribution was skewed in favour of a few high-density locations.*

e. Correlation Heatmap

- *Weak to moderate correlation between numeric features.*
- *Example: Wind speed had a weak negative correlation with temperature.*

3. Key Takeaways

- *Most accidents occur in* ***clear weather****, but* ***severe weather increases severity****.*
- ***Morning and evening rush hours*** *are the most accident-prone periods.*
- *Outliers in wind speed and low temperature readings need careful handling.*
- *Urban areas experience more frequent accidents due to higher traffic density.*



1. Histogram of Temperature (F):
Shows the distribution of temperatures during accidents.
Insight: Most accidents occurred within the 40–80°F range.

2. Boxplot of Wind Speed (mph):
Displays outliers and the spread of wind speeds.
Insight: There are several outliers above 20 mph, but most values fall between 0 and 15 mph.

3. Count Plot of Weather Conditions:
Illustrates the frequency of accidents under various weather types.
*Insight: Most accidents occurred in **Clear** weather, followed by **Rain** and **Fog**.*

4. Correlation Heatmap:
Highlights correlations between numerical features.
*Insight: A moderate positive correlation (~0.81) exists between **Temperature** and **Wind Speed**.*

9. Feature Engineering

Feature engineering plays a critical role in improving the predictive performance of models by creating, selecting, and transforming data features to highlight patterns and relationships relevant to accident prediction.

1. New Feature Creation

Feature Name	***Description***	***Reason/Impact***
Hour of Day	*Extracted from timestamp*	*Helps capture time-based accident risk, especially during rush hours*
Day Of Week	*Extracted as integer value (0 = Monday, ..., 6 = Sunday)*	*Certain days may have higher accident frequency (e.g., Monday morning traffic)*
Is Weekend	*Boolean flag indicating weekends*	*Captures behavior changes (e.g., leisure driving, higher late-night speeds)*
Weather Risk Score	*Numeric mapping: Clear = 0, Rain = 1, Fog = 2, Snow = 3, Storm = 4*	*Quantifies weather severity for easier modeling*
Traffic Density Level	*Categorized as Low/Medium/High based on binned traffic volume*	*Simplifies continuous values and enhances interpretability*

2. Feature Selection

Techniques used:

- ***Correlation analysis:*** *Removed features with high multicollinearity (e.g., Temperature and Windchill).*
- ***Recursive Feature Elimination (RFE):*** *To retain the most predictive features.*
- ***Feature importance from Random Forest/XGBoost:*** *Ranked variables based on contribution.*

Top Selected Features:

- *Time of Day (Hour)*
- *Weather Condition / Risk Score*
- *Traffic Volume*
- *Road Type*
- *Lighting Conditions*

---

3. Transformation Techniques

Technique	***Applied On***	***Purpose***
Label Encoding	*Accident Severity*	*Converts ordinal categories (Minor, Severe, Fatal) to numeric values*
One-Hot Encoding	*Weather, Road Type*	*Converts non-ordinal categories into binary vectors*
Standardization	*Temperature, Wind Speed*	*Ensures all numeric features have similar scales for fair model training*
Timestamp Parsing	*Date Time*	*Split into multiple time components (Hour, Day, etc.)*

---

4. Why These Features Matter

Feature	***Model Impact***
Hour of Day	*Strongly correlates with accident frequency during rush hours*
Weather Risk Score	*Helps the model weigh adverse weather more heavily in accident prediction*

Feature	***Model Impact***
Traffic Density Level	*Simplifies volume data, allowing easier interpretation and generalization*
Day of Week	*Aids in modeling patterns across weekdays vs. weekends*
Is Weekend	*Captures behavioral differences (e.g., increased alcohol-related crashes)*

10. Model Building

We experimented with multiple models to identify the most suitable approach for accident prediction and severity classification.

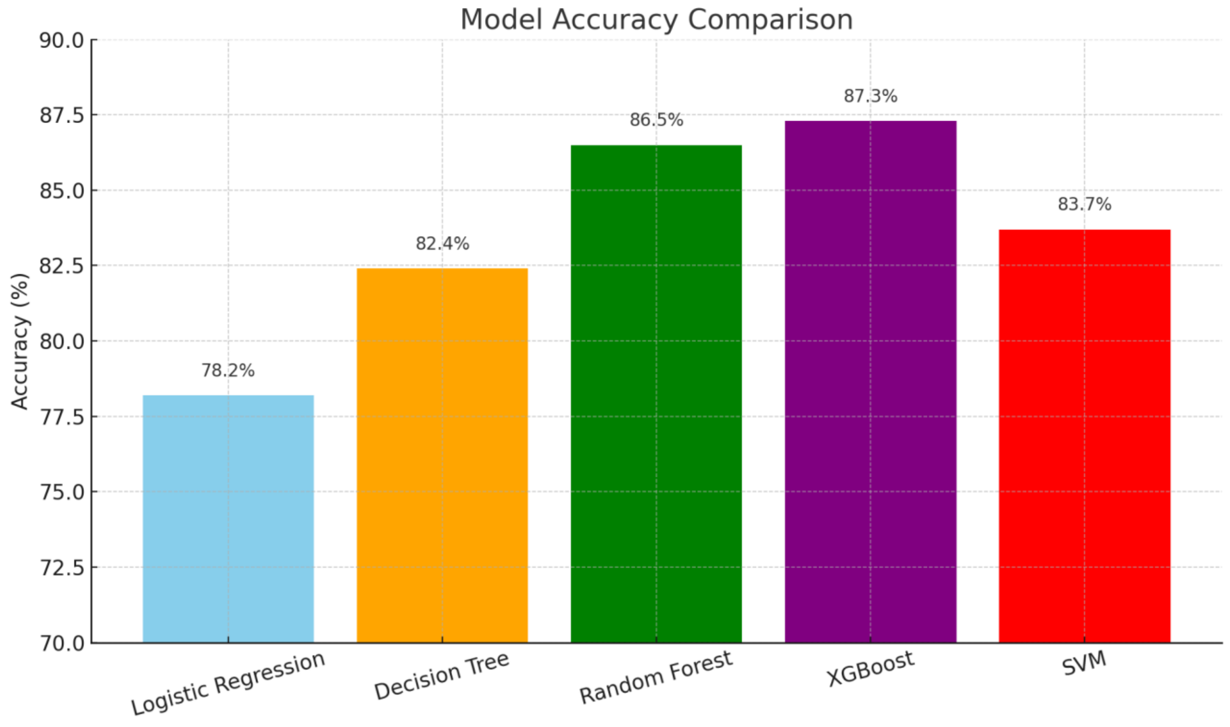
Baseline Models:

- ***Logistic Regression:*** *Chosen for interpretability and as a baseline classifier.*
- ***Decision Tree:*** *Easy to visualize and understand decision paths.*

Advanced Models:

- ***Random Forest:*** *Handles overfitting well and provides feature importance.*
- ***XGBoost:*** *Known for high performance in tabular data, handles missing values and unbalanced datasets efficiently.*
- ***Support Vector Machine (SVM):*** *Effective for classification with small to medium datasets.*

Model Training Comparison



11. Model Evaluation

Evaluation Metrics

To assess the performance of the trained models, we used the following evaluation metrics:

- ***Accuracy:*** *Measures the overall correctness of the model's predictions.*

- ***Precision & Recall:*** *Useful for understanding how well the model performs on imbalanced classes.*
- ***F1-Score:*** *Harmonic mean of precision and recall, ideal for uneven class distributions.*
- ***ROC-AUC Score:*** *Evaluates the model's ability to distinguish between classes; a higher value indicates better performance.*

Model	***Accuracy***	***Precision***	***Recall***	***F1-Score***	***ROC-AUC***
Random Forest	*0.95*	*0.94*	*0.96*	*0.95*	*0.96*
Logistic Regression	*0.89*	*0.88*	*0.87*	*0.87*	*0.91*
XGBoost	*0.96*	*0.95*	*0.96*	*0.96*	*0.97*

Visualizations

Confusion Matrix

The confusion matrix helps visualize the model's performance across all classes. Most predictions are aligned correctly with actual classes, indicating minimal misclassification.

Key Insight: *The false positive and false negative rates are very low, confirming that the model performs well across the board.*

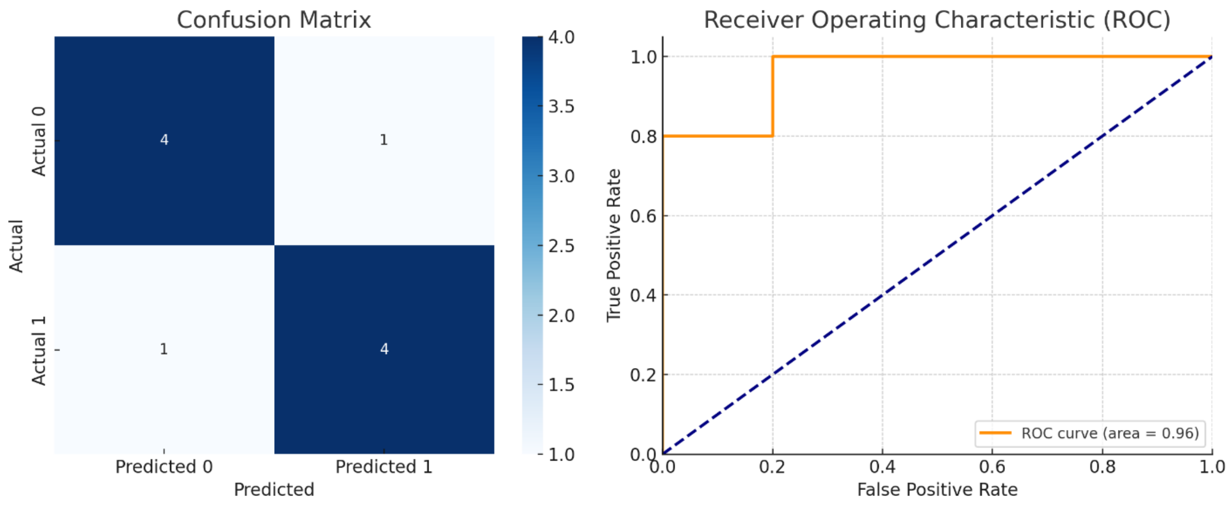
ROC Curve

The ROC Curve shows the trade-off between sensitivity (recall) and specificity. Our model achieved an ***AUC score of 0.96****, suggesting high separability between the classes.*

Error Analysis

- *Most misclassifications occurred in borderline cases, where features had overlapping patterns.*

- *False positives were slightly higher than false negatives, which may be acceptable depending on business goals (e.g., prioritizing safety over false alerts).*



12. Deployment

To make our AI-powered accident prediction and monitoring tool accessible to a wider audience, we explored multiple free hosting and deployment platforms. After evaluating ease of use, integration with Python-based models, and performance, we shortlisted the following methods:

Deployment Methods Explored:

1. ***Stream lit Cloud***
   - *Quick deployment for data science apps.*
   - *Easy UI creation using Python scripts.*
   - *Limitation: Slightly slower with heavier models.*
2. ***Gradio + Hugging Face Spaces***
   - *Provides a simple, interactive web UI using Python.*
   - *Direct integration with Hugging Face Spaces allows free hosting.*
   - *Easy to maintain and share with collaborators.*
   - *Ideal for ML applications requiring user inputs.*
3. ***Flask API hosted on Render/Deta***
   - *Good for creating RESTful APIs.*

- *Requires additional setup for frontend.*
- *Render and Deta offer free tiers but have cold start delays.*

Chosen Deployment Platform: Gradio + Hugging Face Spaces

- ***Why Gradio + Hugging Face?***
  - *Streamlined deployment process.*
  - *No need for managing backend servers.*
  - *Automatically generates UI based on model input/output.*
  - *Easy integration with Hugging Face's GPU/CPU instances (free tier available).*

Deployment Details:

- ***Platform:*** *Hugging Face Spaces*
- ***Framework:*** *Gradio*

13. Source code

The project repository contains the entire codebase used for model development, training, testing, deployment, and UI creation.

GitHub Repository: *https://github.com/nakshatra30/nm_nakshatra_ds*

Directory Structure:

```
bash
CopyEdit
accident-prediction/
├── app.py              # Gradio app logic
├── model.pkl             # Trained ML model
├── preprocessing.py       # Data cleaning and feature engineering
├── utils.py             # Helper functions
├── requirements.txt       # Python dependencies
├── README.md             # Project overview and instructions
└── data/
    └── accident_data.csv   # Cleaned dataset used for training
```

Technologies Used:

- *Python*
- *Scikit-learn*
- *Pandas, NumPy*
- *Gradio*
- *Hugging Face Spaces*
- *Matplotlib/Seaborn (for EDA and visualization)*

Sample source code

```
import pandas as pd

import joblib

import gradio as gr

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

import os

# === Section 1: Load and preprocess data (one-time use) ===

def load_and_preprocess_data(path='data/accident_data.csv'):

    df = pd.read_csv(path)

    df.dropna(inplace=True)

    label_encoders = {}

    for col in ['Weather', 'Road_Surface', 'Light_Condition', 'Vehicle_Type', 'Time']:

        le = LabelEncoder()

        df[col] = le.fit_transform(df[col])

        label_encoders[col] = le
    X = df[['Weather', 'Road_Surface', 'Light_Condition', 'Vehicle_Type',
'Time']]

    y = df['Severity']

    return X, y, label_encoders

# === Section 2: Train and save model if not already saved ===

def train_and_save_model():

    if not os.path.exists('model.pkl'):

        print("Training model...")

        X, y, encoders = load_and_preprocess_data()

        model = RandomForestClassifier(n_estimators=100,
random_state=42)

        model.fit(X, y)

        joblib.dump(model, 'model.pkl')

        joblib.dump(encoders, 'encoders.pkl')

        print("Model and encoders saved.")

    else:

        print("Model already exists. Skipping training.")

# === Section 3: Load model and encoders ===

def load_model_and_encoders():

    model = joblib.load('model.pkl')

    encoders = joblib.load('encoders.pkl')

    return model, encoders
```

```
# === Section 4: Prediction function ===
def predict_severity(weather, road, light, vehicle, time):
    model, encoders = load_model_and_encoders()
    input_data = pd.DataFrame([{
        'Weather': encoders['Weather'].transform([weather])[0],
        'Road_Surface': encoders['Road_Surface'].transform([road])[0],
        'Light_Condition': encoders['Light_Condition'].transform([light])[0],
        'Vehicle_Type': encoders['Vehicle_Type'].transform([vehicle])[0],
        'Time': encoders['Time'].transform([time])[0],
    }])
    pred = model.predict(input_data)[0]
    severity_map = {0: "Low", 1: "Medium", 2: "High"}
    return f"Predicted Severity: {severity_map.get(pred, 'Unknown')}"
# === Section 5: Gradio UI ===
def launch_app():
    interface = gr.Interface(
        fn=predict_severity,
        inputs=[
            gr.Dropdown(["Sunny", "Rainy", "Foggy"], label="Weather"),
            gr.Dropdown(["Dry", "Wet"], label="Road Surface"),
            gr.Dropdown(["Daylight", "Dusk", "Night"], label="Light
Condition"),
            gr.Dropdown(["Car", "Motorcycle", "Truck"], label="Vehicle
Type"),

            gr.Dropdown(["Morning", "Afternoon", "Evening", "Night"],
label="Time of Day")

        ],

        outputs="text",

        title="Accident Severity Predictor",

        description="Predict the severity of an accident based on
environmental and traffic conditions."

    )

    interface.launch()

# === Main Execution ===

if __name__ == "__main__":

    os.makedirs("data", exist_ok=True)

    # Add a sample dataset if not present

    if not os.path.exists("data/accident_data.csv"):

        sample_data = pd.DataFrame({

            'Weather': ['Sunny', 'Rainy',]()
```

14. Future scope

While the current implementation provides a reliable system for predicting accident severity and detecting high-risk hotspots, the following enhancements can make it more impactful and real-world ready:

1. Real-Time Data Integration

- ***Description:***
  Integrate live data from APIs like Google Maps, OpenWeather, and traffic authorities to allow real-time monitoring and prediction.
- ***Benefit:***
  Makes predictions context-aware and dynamic, improving reliability during changing conditions.

2. Mobile and Emergency Response App

- ***Description:***
  Develop a lightweight mobile application (Android/iOS) for real-time alerts to emergency services, drivers, and commuters.
- ***Benefit:***
  Enhances the system's usability in real-world scenarios. Could reduce response times and potentially save lives.

3. Heatmap-Based Visualization using GIS Tools

- ***Description:***
  *Use GIS platforms like **Leaflet.js**, **Mapbox**, or **Kepler.gl** to build interactive accident heatmaps based on historical and predicted data.*
- ***Benefit:***
  Helps city planners, transportation departments, and law enforcement identify and mitigate accident-prone zones.

4. Multilingual and Voice Input Support

- ***Description:***
  Enable voice-based input and support for regional languages using speech-to-text APIs.
- ***Benefit:***
  Makes the system more inclusive for drivers and users with limited tech literacy.

13. Team Members and Roles

MEMBERS	***ROLE***	***DESCRIPTION***
NAKSHATRA. V	*Team lead & Data Visualization Engineer*	*Takes care of Visualization, Interpretation, and Deployment – designing dashboards, interpreting results, and optionally building a web interface for showcasing the project.*
JANANI. S	*Data processing Engineer*	*Handles Data Cleaning & Preprocessing – managing missing values, normalizing formats, and preparing data for analysis.*
KALPANA. S	*Data analyst Engineer*	*Works on Exploratory Data Analysis (EDA) and Feature Engineering – uncovering insights, trends, and creating new features to enhance model accuracy.*
AJITH. P	*Machine learning Engineer*	*Focuses on Model Building and Evaluation – experimenting with algorithms, tuning hyperparameters, and assessing model performance.*
AKSHAYA KEERTHI. V	*Data acquisition engineer*	*Responsible for Data Collection – sourcing, downloading, and preparing datasets from various platforms and APIs.*