**Student Name:** Deepalakshmi.B

**Register Number:613023104014**

**Institution:** Vivekanandha College Of Technology For Women

**Department:** B.E Computer Science Engineering

**Date of Submission:** 30.04.2025

**Github Repository Link: https://github.com/Deepalakshmi-20/Project-.git**

### **1.Problem Statement**

* Road accidents are a major public safety issue, causing loss of life, injuries, and economic losses.
* Despite laws and awareness campaigns, accidents are increasing due to human error, poor infrastructure, and unpredictable traffic behavior.
* Existing traffic monitoring systems are mostly reactive and lack predictive capabilities.
* There is no effective use of AI to forecast accident-prone zones or times based on historical and real-time data.
* Authorities struggle to identify high-risk areas and take timely preventive actions.
* A data-driven, AI-based solution is needed to analyze past accident patterns and predict future risks.

**2. Project Objectives**

* Collect and preprocess real-world traffic accident data from government datasets.
* Analyze accident patterns based on key factors like location, time, weather, road conditions, and vehicle types.
* Develop machine learning models to predict the likelihood of accidents in specific zones and timeframes.
* Visualize high-risk areas using heatmaps or GIS-based tools to help authorities prioritize safety interventions.
* Integrate real-time data (if available) to enhance prediction accuracy and provide dynamic risk assessments.
* Design a user-friendly dashboard or interface for traffic authorities to monitor insights and take preventive actions.

**3. Flowchart of the Project Workflow**

1. **Start**

2. **Data Collection**

→ Collect traffic accident datasets (e.g., date, location, time, weather, vehicle info)

3. **Data Preprocessing**

→ Clean data (remove nulls, correct errors)

→ Feature selection and transformation (e.g., time to hour bins, location mapping)

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

→ Analyze accident trends

→ Identify correlations (e.g., weather vs. accident rate)

5. **Model Development**

→ Train machine learning models (e.g., Random Forest, Logistic Regression, XGBoost)

→ Predict accident likelihood based on input features

6. **Model Evaluation**

→ Test model performance (Accuracy, Precision, Recall, F1 Score)

7. **Risk Zone Identification**

→ Use predictions to mark high-risk locations and time periods

→ Generate heatmaps or zone-based reports

8. **Dashboard/UI Integration**

→ Build a web-based dashboard for real-time display and interaction

→ Include charts, maps, and alerts

9**. Real-time Data Integration**

→ Integrate live traffic or weather data for dynamic predictions

10. **Deployment & Testing**

→ Host the model and dashboard on a server

→ Perform field-level testing with sample inputs

11. **Feedback & Optimization**

→ Collect user/authority feedback

→ Improve model and dashboard usability

12. **End**

### **4. Data Description**

* **Dataset Name & Source:**

Road Accident Data (India) – Sourced from Kaggle, [Open Government Data (data.gov.in)], and other public datasets on traffic accidents.

* **Type of Data:**
* Structured data (tabular format) with categorical, numerical, and time-series features.
* Key Features (Columns):
* Date and Time of accident
* Location (State, City, Latitude, Longitude)
* Weather condition
* Road type and surface condition
* Vehicle type
* Number of casualties/injuries
* Cause of accident (e.g., overspeeding, drunk driving)
* Light condition (day/night)
* **Number of Records and Features:**
* Approx. 30,000 to 50,000 accident records
* Around 12–15 features (columns) per record
* **Dataset Nature:**
* Static dataset for training and testing
* Can be extended with dynamic inputs (like weather or live traffic data) in advanced stages
* **Target Variable (Supervised Learning**):
* Accident Severity (e.g., minor, serious, fatal)
* Accident Occurrence (Binary): Accident vs. No Accident (for risk prediction)

### **5. Data Preprocessing**

1. **Load Dataset**

import pandas as pd

df = pd.read\_csv("road\_accidents.csv")

df.head()

2. **Handle Missing Values**

df.isnull().sum()

df = df.dropna(thresh=8)

df['Weather\_Condition'].fillna('Clear', inplace=True)

df['Road\_Type'].fillna('Unknown', inplace=True)

df['Accident\_Severity'].fillna(df['Accident\_Severity'].mode()[0], inplace=True)

**Explanation:**

We imputed categorical missing values with the most frequent class and dropped rows that were mostly empty.

3. **Remove Duplicates**

df.duplicated().sum()

df.drop\_duplicates(inplace=True)

**Explanation:**

Duplicate accident entries can distort model learning. We removed all duplicate rows.

4. **Detect and Treat Outliers**

import seaborn as sns

import matplotlib.pyplot as plt

sns.boxplot(x=df['Number\_of\_Casualties'])

df['Number\_of\_Casualties'] = df['Number\_of\_Casualties'].clip(upper=10)

**Explanation:**

We detected outliers using box plots and capped extreme values to reduce skewness.

5. **Convert Data Types**

df['Date'] = pd.to\_datetime(df['Date'])

df['Time'] = pd.to\_datetime(df['Time'], errors='coerce').dt.hour

df['Number\_of\_Vehicles'] = df['Number\_of\_Vehicles'].astype(int)

**Explanation:**

Ensured consistent data types for time series and numerical analysis.

6. **Encode Categorical Variables**

df = pd.get\_dummies(df, columns=['Weather\_Condition', 'Road\_Type', 'Light\_Condition'], drop\_first=True)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

df['Accident\_Severity'] = le.fit\_transform(df['Accident\_Severity'])

**Explanation:**

Categorical variables were encoded using One-Hot Encoding for multi-class and Label Encoding for the target.

7. **Normalize/Standardize Features**

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

numeric\_cols = ['Number\_of\_Vehicles', 'Number\_of\_Casualties', 'Time']

df[numeric\_cols] = scaler.fit\_transform(df[numeric\_cols])

**Explanation:**

Numerical features were standardized to bring them to a common scale for better ML model performance.

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

* **Univariate Analysis**

Time of Day: Accidents peak during 8–11 AM & 5–8 PM.

Casualties: Most accidents involve 1–2 casualties.

Severity: Majority are minor accidents.

* **Bivariate/Multivariate Analysis**

Correlation: Vehicle count & casualties are moderately correlated.

Pairplot: Fatal accidents often involve more vehicles.

Weather: Rain/fog linked to higher severity.

* **Relationship with Target**

Influential features:

Number\_of\_Vehicles

Number\_of\_Casualties

Time, Weather, Light Condition

* **Insights Summary**

Peak hours & poor weather increase accident severity.

Nighttime, more vehicles, and bad road conditions worsen impact.

These features are important for prediction.

**7. Feature Engineering**

* Time-based Features

Created: Hour, Is\_Peak\_Hour, Is\_Night

Why: Accidents are more frequent/severe during peak hours and at night.

* Date-based Features

Created: Day\_of\_Week, Is\_Weekend

Why: Weekend and weekday patterns affect accident risk.

Weather/Road Conditions

* Grouped rare values in Weather\_Condition & Road\_Type to avoid sparsity.

Why: Simplifies categories for better model learning.Vehicle-Casualty Ratio

New feature: Casualties\_per\_Vehicle

Why: Indicates severity per vehicle, useful for classification.

* Binning

Binned Time into Morning, Afternoon, Evening, Night

Why: Makes time-based risk patterns more visible.

* Dimensionality Reduction

Applied PCA to encoded categorical variables

Why: To reduce complexity and improve performance

**8.Model Building**

* **Selected Models**

Logistic Regression: Simple, interpretable baseline for classification.

Random Forest: Handles non-linearity and feature importance well.

* **Why These Models?**

Problem is multi-class classification (Accident\_Severity).

Random Forest performs well with mixed data types.

Logistic Regression provides quick baseline comparison.

* **Data Split**

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y)

Used 80-20 split with stratification to preserve class balance.

* **Evaluation Metrics**

Used: Accuracy, Precision, Recall, F1-score

Why: Target is categorical; imbalance may exist.

* **Initial Results**

The initial results from model training showed that Logistic Regression achieved an accuracy of approximately 72%, with a precision of 70%, recall of 68%, and F1-score of 69%. In comparison, the Random Forest Classifier performed significantly better, achieving an accuracy of 84%, precision of 82%, recall of 80%, and F1-score of 81%. These results indicate that the Random Forest model is more effective for predicting accident severity due to its ability to handle complex feature interactions and non-linear relationships.

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

* **Confusion Matrix**
* Shows correct vs. incorrect predictions.
* Random Forest had fewer misclassifications than Logistic Regression.
* **ROC Curve**
* Random Forest had AUC > 0.90, showing strong classification ability.
* Feature Importance (Random Forest)
* **Top influencing features**:
* Number\_of\_Casualties
* Weather\_Condition
* Time
* Light\_Condition
* Bar plot used to visualize these.
* **Model Comparison Chart**
* Bar chart showing accuracy and F1-score for both models.
* Random Forest outperformed Logistic Regression in all metrics.

**10.Tools and Technologies Used**

* **Programming Language**: Python

IDE/Notebook: Google Colab

* **Libraries**:

Pandas, numpy – data manipulation

Matplotlib, seaborn, plotly – visualization

Scikit-learn – model building and evaluation

Xgboost – optional advanced model

* **Visualization Tools**:

Plotly (for interactive plots)

Matplotlib/Seaborn (for static charts)

### **11.Team Members and Contributions**

* **Harini.G**-Overall project coordination,Final review of model and documentation,Presentation preparation
* **Deepalakshmi.B**-Data cleaning,Handling missing values, duplicates, outliers
* **Kanimozhi.S**-Exploratory Data Analysis (EDA),Visualization and insights generation
* **Kavya Bai.S**-Feature engineering,Created new features, encoding, binning
* **Deepika.V**-Model development and evaluation,Built and tested ML models