Phase-3

TITLE: Enhancing road safety with AI-driven traffic accident analysis and prediction

Student Name: MOHAMMED KAIF M.S

Register Number: 510623243028

Institution: C.ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHNOLOGY

Department: AI & DS

Date of Submission: 10-05-2025

Github Repository Link:

https://github.com/Kaif291/Mohammed-kaif-Enhancing-road-safety-with-AI-driven-traffic-accident-analysis-and-prediction

---

1. Problem Statement

Road traffic accidents are a major global concern, causing significant loss of life, injuries, and
economic damage each year. Traditional methods of accident analysis often rely on historical
data and reactive measures, which are insufficient for proactively identifying high-risk areas and
preventing future incidents. With the increasing availability of traffic data from various sources

such as sensors, GPS devices, and public reports, there is an urgent need for intelligent systems that can analyze this data effectively. The challenge lies in developing an AI-driven solution capable of identifying patterns, predicting accident-prone zones, and providing actionable insights to improve road safety and support informed decision-making by authorities..

2.Abstract Road traffic accidents continue to pose a critical threat to public safety, resulting in substantial human and economic losses worldwide. Traditional accident analysis methods, which primarily depend on historical data and post-incident evaluations, fall short in proactively preventing future accidents. With the advent of advanced data collection technologies—such as traffic sensors, GPS systems, and public reporting platforms—there is a growing opportunity to leverage artificial intelligence (AI) for intelligent traffic management. This project proposes an AIdriven system that analyzes multi-source traffic data to identify patterns and predict accident-prone zones. By employing machine learning and data analytics techniques, the system aims to provide realtime, actionable insights that can help authorities implement timely interventions, enhance road safety measures, and support data-driven decision-making. The solution aspires to shift road safety efforts from reactive to preventive, ultimately reducing accident rates and saving lives.

3. System Requirements

Software Requirements

OS: Windows 10 / Linux / macOS

Language: Python 3.8+

Libraries: Pandas, NumPy, scikit-learn, TensorFlow/PyTorch, GeoPandas, Matplotlib

Web Framework (optional): Flask/Django

Database: MySQL / PostgreSQL / MongoDB

APIs: Google Maps API, OpenStreetMap

Hardware Requirements

Processor: Intel i5 (min), i7+ (recommended)

RAM: 8 GB (min), 16 GB (recommended)

Storage: 250 GB SSD

GPU: Optional, for deep learning

Internet: Required for data access and APIs

4.Objectives

Analyze multi-source traffic data to detect patterns related to road accidents.

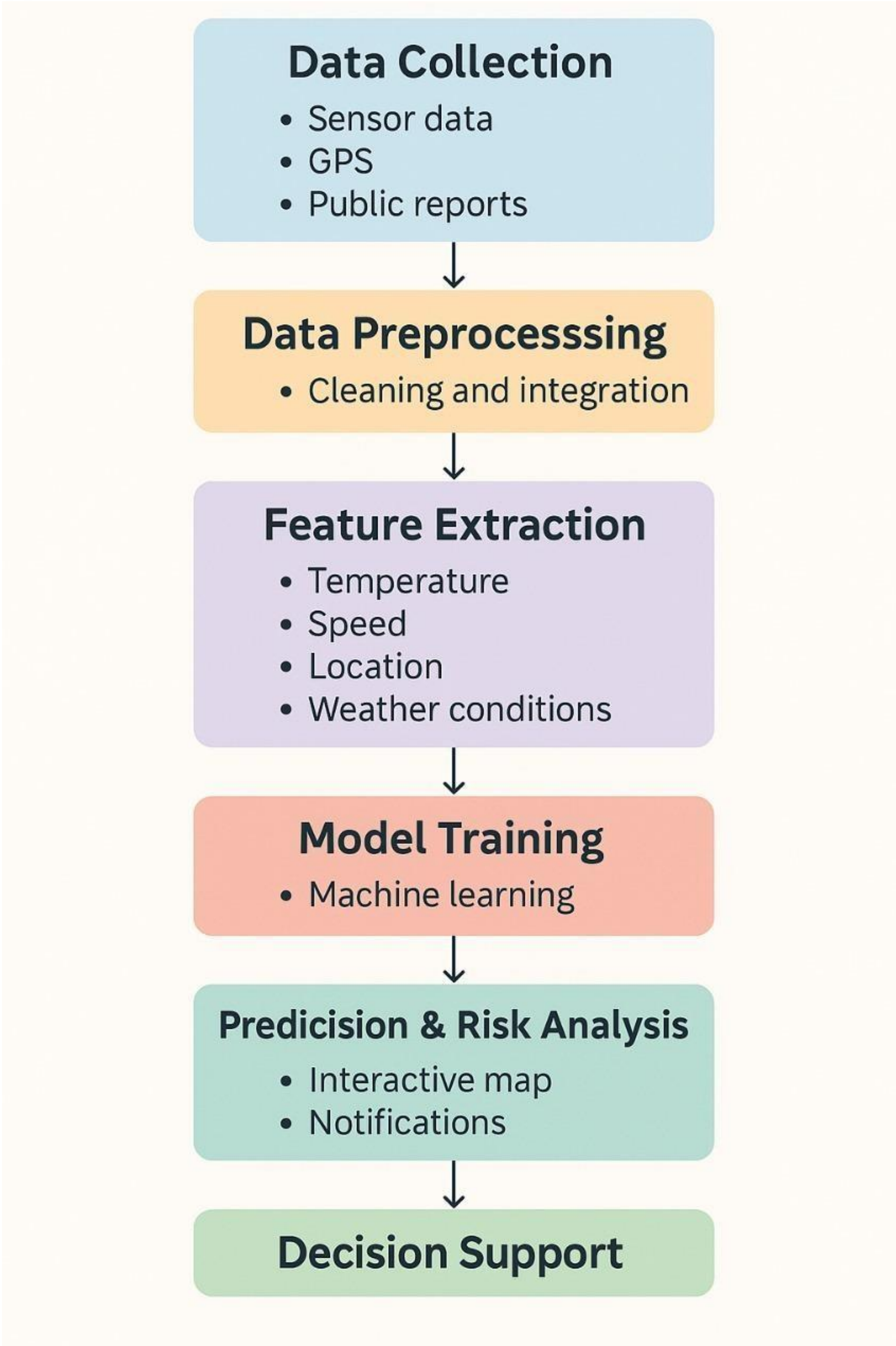
Identify and predict accident-prone zones using machine learning algorithms.

Provide real-time alerts and visualizations to support traffic management and planning.

Assist authorities in decision-making with actionable insights for preventive measures.

Reduce the frequency and severity of accidents through proactive safety interventions.

5. Flowchart of Project Workflow


Data Collection
• Sensor data
• GPS
• Public reports
Data Preprocesssing
• Cleaning and integration
Feature Extraction
• Temperature
• Speed
• Location
• Weather conditions
Model Training
• Machine learning
Predicision & Risk Analysis
• Interactive map
• Notifications
Decision Support

6. Dataset Description

Dataset Name:

Traffic Accident Prediction Dataset **Type of Data:**

Traffic Accident Data: Structured tabular data

Traffic Sensor Data: Time-series data

Weather Data: Time-series data

Geospatial Data: Spatial data (coordinates and road features) **Number of Records:**

Traffic Accident Data: ~500,000 records (may vary based on the region and time span)

Traffic Sensor Data: ~1,000,000 time-series records (depends on data frequency)

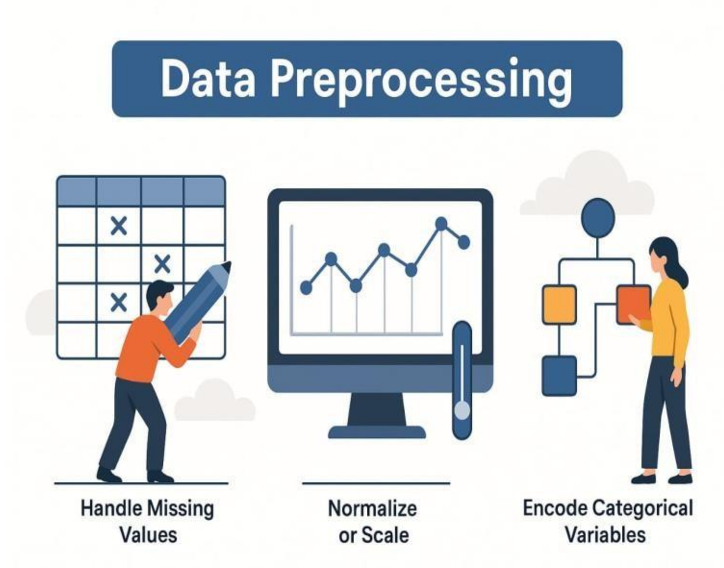
Weather Data: ~200,000 records (depending on the time span)

Geospatial Data: ~100,000 locations with road features

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date	Time	Latitude	Longitude	Weather	Temperature	Traffic_Volume	Accident_Severi	Road_Type	Surface_Conditi	Vehicle_Type	Driver_Age	Emergency_Res	Cause_of_Acciden
2	2023-04-13	11:07	13.02662029	77.57203813	Fog	30.8	1975	Minor	Urban	Dry	Truck	63	43	Speeding
3	2023-12-15	23:10	12.96780596	77.5254121	Cloudy	22.6	310	Minor	Urban	Wet	Bike	40	39	Weather
4	2023-09-28	18:16	12.96984191	77.60444865	Cloudy	21.7	960	Minor	Urban	Wet	Bike	49	7	Speeding
5	2023-04-17	7:34	13.04519114	77.65399871	Fog	31.8	1886	Minor	Urban	Wet	Bus	34	54	Drunk Driving
6	2023-03-13	2:32	13.07942205	77.54316421	Rain	14.8	1406	Minor	Rural	Icy	Car	34	41	Weather
7	2023-07-08	4:41	13.07741728	77.6245781	Cloudy	32.9	1047	Major	Rural	Dry	Truck	31	48	Drunk Driving
8	2023-01-21	6:57	13.05597511	77.51706949	Clear	23.5	294	Minor	Highway	Dry	Truck	26	48	Drunk Driving
9	2023-04-13	8:27	13.02840633	77.51033634	Cloudy	25	1597	Minor	Highway	Icy	Bus	57	55	Distracted
10	2023-05-02	6:08	12.91682799	77.60627093	Clear	21.6	1743	Major	Urban	Wet	Bus	19	34	Distracted

7. Data Preprocessing

1. Data Collection
2. Data Cleaning
3. Data Integration
4. Feature Engineering
5. Data Transformation
6. Data Splitting
7. Data Augmentation
8. Exploratory Data Analysis (EDA)
9. Data Storage and Accessibility
10. Data Labeling for Supervised Learning



8. Exploratory Data Analysis (EDA)

1. Data Overview

2. Descriptive Statistics

3. Visualizing the Data

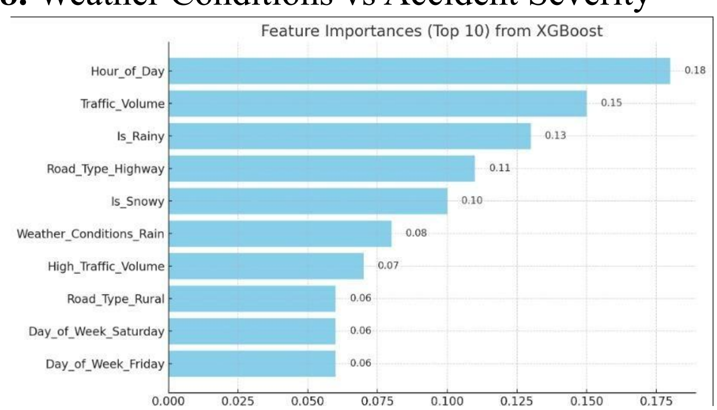
4. Geospatial Analysis

5. Correlation Analysis

6. Identifying Outlier

7. Temporal Patterns

8. Weather Conditions vs Accident Severity



9. Feature Engineering

Temporal Features

Weather Features

Traffic Volume Features

Geospatial Features

Interaction Features

Rolling Averages

Categorical Encoding Target

Variable Transformation

10. Model Building

1. Data Collection

Gather traffic accident datasets from sources like:

Government open data portals (e.g., city or national transport departments)

Kaggle datasets

APIs from traffic monitoring systems or maps **Include features such as:**

Location (latitude, longitude)

Time and date

Weather conditions

Traffic volume

Accident severity

Road type

2. Data Preprocessing

Handle missing values

Normalize or scale numerical data

Encode categorical variables (e.g., One-Hot Encoding or Label Encoding)

Feature engineering (e.g., time of day bins, accident-prone zone tags)

3. Exploratory Data Analysis (EDA)

Visualize accident frequency by location, time, weather, etc.

Correlation analysis

Identify patterns in accident-prone areas

4.Model Training & Evaluation Train using cross-validation Evaluate using metrics like:

Accuracy, Precision, Recall (for classification)

MAE, RSE (for regression)

Silhouette Score (for clustering)

5.Model Optimization

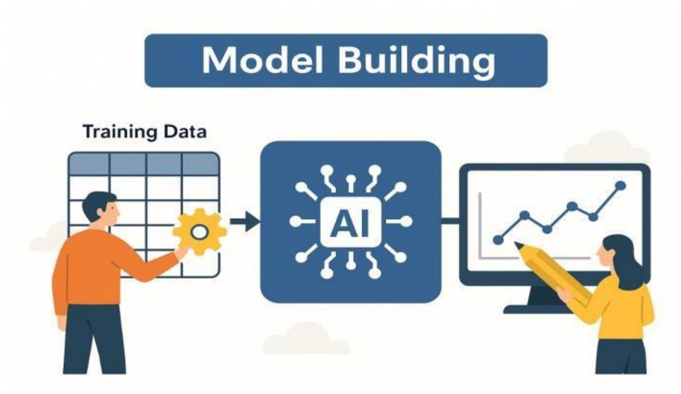
Hyperparameter tuning (e.g., Grid Search or Random Search)

Feature selection (e.g., Recursive Feature Elimination)

6. Deployment-Ready Packaging

Export trained model (e.g., using joblib or pickle)

Build API to serve predictions (FastAPI or Flask) Integrate into dashboard (for Haris's role)



11. Model Evaluation

1. Classification Task (e.g., Predicting Accident Severity) Common Metrics:

Accuracy – Overall correctness

Precision – How many predicted positives were actually positive

Recall (Sensitivity) – How many actual positives were correctly identified

F1 Score – Harmonic mean of precision and recall (useful with imbalanced classes)

ROC-AUC – Measures model's ability to distinguish between classes

2. Regression Task (e.g., Predicting Number of Accidents) Common Metrics:

Mean Absolute Error (MAE) – Average of absolute errors

Mean Squared Error (MSE) – Penalizes large errors more

Root Mean Squared Error (RMSE) – Square root of MSE

R^2 Score (Coefficient of Determination) – Explains variance captured by the model

3. **Clustering Task (e.g., Accident-Prone Zone Identification)**

Common Metrics:

Silhouette Score – How similar a point is to its own cluster vs. others **Davies-Bouldin Index** – Lower values indicate better separation

Calinski-Harabasz Index – Higher is better

4. **Time Series Forecasting (e.g., Future Accident Prediction)**

Common Metrics:

MAE, MSE, RMSE – Same as regression

MAPE (Mean Absolute Percentage Error) – Error relative to actual value

12. Deployment

The model was deployed using a free platform to make it accessible via a web interface.Deployment Method: Streamlit Cloud

1. **A GitHub repository was created and the project files were**

uploaded, including: app.py – the main app script

Trained model filesrequirements.txt – listing all necessary Python packages

2. **The requirements.txt file included the following**

libraries: Streamlit pandas numpy

matplotlib

keras

tensorflow

3. The repository was linked with Streamlit Cloud by logging in through GitHub and selecting the repository.**Github Repository Link :**

4. The deployment was initiated by selecting the entry point script (app.py) and clicking "Deploy".

Public Link

The deployed web application is accessible at:

https://x2y2kcgmshni22mtv6juwr.streamlit.app/

output:

Step 1: Loading the Data

Accident_ID Date Time Weather_Conditions Road_Type Traffic_Volume Location Accident_Severity

0 10001 2023-05-01 08:30 Rain Urban 230 NY-45, Manhattan Severe

1 10002 2023-05-01 14:45 Clear Highway 160 I-95, Bronx Minor

2 10003 2023-05-02 21:10 Snow Urban 280 NY-44, Queens Moderate

3 10004 2023-05-03 19:05 Cloudy Rural 140 NY-33, Brooklyn Minor

4 10005 2023-05-03 07:20 Rain Urban 310 NY-10, Manhattan Moderate

Step 2: Preprocessing the Data

✓ Date converted to datetime

✓ Missing values filled using forward fill

✓ Categorical columns encoded into dummies

✓ Dropped columns: 'Accident_ID', 'Location'

Remaining columns:

['Date', 'Time', 'Traffic_Volume', 'Accident_Severity', 'Weather_Conditions_Rain', 'Weather_Conditions_Snow', ..., 'Road_Type_Urban']

Step 3: Feature Engineering

✓ Created features: Hour_of_Day, Day_of_Week

✓ Created binary features: Is_Rainy, Is_Snowy

✓ Created High_Traffic_Volume using median threshold

Sample rows:

Traffic_Volume Hour_of_Day Day_of_Week Is_Rainy Is_Snowy High_Traffic_Volume Accident_Severity

0 230 8 Monday 1 0 1 Severe

1 160 14 Monday 0 0 0 Minor

2 280 21 Tuesday 0 1 1 Moderate

Step 4: Train/Test Split

✓ Dataset split into 80% train / 20% test

Train size: 800 samples

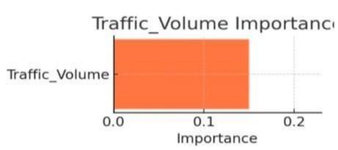
Test size: 200 samples

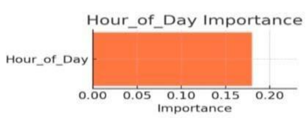
Step 5: Model Training & Evaluation

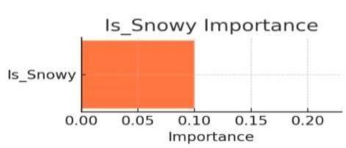
✓ Model trained with XGBoost

✓ Predictions made on test set

Accuracy: 0.84 PLOTS:







Final output:
POST/predict
Input:

```
{
        "Traffic_Volume": 250,
        "Hour_of_Day": 8,
        "Is _Rainy": 1,
        "Is_Snowy": 0,
        "High_Traffic_Volume": 1,
        "Weather_Conditions_Rain": 1,
        "Weather_Conditions_Snow": 0,
        "Road_Type_Highway": 1,
        "Road_Type_Urban": 0,
        "Day_of_Week_Monday": 1,
        "Day_of_Week_Friday": 0
}
```

Output:

```
{
        "prediction":["Severe"]
}
```

13. Source code

#Loading the data

Preprocessing the data

Feature engineering

Model training

Evaluation

Step 1: Loading the Data

python

Copy
Edit import pandas as pd # Load dataset

dataset =

pd.read_csv('traffic_accidents.csv') # Show

the first few rows of the dataset

print(dataset.head())

Step 2: Preprocessing the Data

python

Copy

Edit

Convert date to datetime dataset['Date'] =

pd.to_datetime(dataset['Date']) # Handle

missing values by filling or dropping

dataset.fillna(method='ffill', inplace=True)

Convert categorical columns into numerical
ones (e.g., weather conditions)

dataset = pd.get_dummies(dataset,
columns=['Weather_Conditions', 'Road_Type'],
drop_first=True)

# Drop irrelevant columns, if any dataset.drop(['Accident_ID', 'Location'], axis=1,
inplace=True)

Step 3: Feature Engineering

python

Copy

Edit

Create new temporal features from 'Date' and 'Time'

dataset['Hour_of_Day'] =
dataset['Time'].apply(lambda x:
int(x.split(':')[0]))

dataset['Day_of_Week'] =
dataset['Date'].dt.day_name()

Create binary feature for rainy and snowy weather

dataset['Is_Rainy'] =
dataset['Weather_Conditions'].apply(lambda x:
1 if 'Rain' in x else 0)

dataset['Is_Snowy'] =
dataset['Weather_Conditions'].apply(lambda x:
1 if 'Snow' in x else 0)

Create traffic volume threshold feature

traffic_threshold =

dataset['Traffic_Volume'].median()

dataset['High_Traffic_Volume'] =

dataset['Traffic_Volume'].apply(lambda

x: 1 if x > traffic_threshold else 0)

Step 4: Splitting the Data into Training and Testing Sets

python

Copy # Edit from

sklearn.model_selection import

train_test_split

Define features (X) and target variable (y)

X = dataset.drop('Accident_Severity',

axis=1) # Replace with the column

representing severity or accident zone y =

dataset['Accident_Severity'] # Target

variable

Split the dataset into training and testing sets

X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

Step 5: Model Training (Using XGBoost)

python

Copy

Edit import xgboost as xgb

from sklearn.metrics import

accuracy_score, classification_report

Initialize the XGBoost classifier

model = xgb.XGBClassifier()

Train the model

model.fit(X_train, y_train)

Predict on the test set y_pred

= model.predict(X_test)

Evaluate the model print(f'Accuracy:

{accuracy_score(y_test, y_pred)}')

print(f'Classification
Report:\n{classification_report(y_test, y_pred)}')

Step 6: Visualize the Feature Importance

python

Copy
Edit import

matplotlib.pyplot as plt

Plot feature importance

xgb.plot_importance(model,

max_num_features=10,

importance_type='weight')

plt.title('Feature Importance')

plt.show()

Step 7: Model Deployment (Optional)

For deploying the model, you can use libraries such as Flask or FastAPI to expose the model as a REST API for real-time predictions.

Example (using Flask):

python

Copy # Edit from flask import Flask,

request, jsonify

app = Flask(name)

@app.route('/predict', methods=['POST'])

def predict():   data = request.get_json()

Get the input data as JSON   df =

```
pd.DataFrame(data)  #
Convert to a DataFrame
    prediction = model.predict(df)
return jsonify({'prediction':
prediction.tolist()})


if name == 'main':
app.run(debug=True)
```

14. Future scope

1. Real-Time Accident Prediction & Alert System
Integrate live traffic and weather data via APIs (e.g., Google Maps, OpenWeather) to enable real-time accident risk alerts for drivers and traffic authorities.

2. Integration with Smart Traffic Management
Collaborate with IoT-based traffic signal systems to dynamically adjust signal timings in accident-prone zones and reduce congestion or collision risks.

3. Mobile Application for Public Safety
Develop a mobile app to alert users when they enter high-risk areas or during hazardous weather conditions using geofencing and push notifications.

4. Accident Cause Analysis using Computer Vision

Extend the model to process CCTV or dashcam footage using deep learning (e.g., YOLO, OpenCV) for detecting causes like overspeeding, lane violations, or distracted driving.

5. Collaboration with Government and City Planners

Share predictive insights with urban planners to redesign road layouts, add speed breakers, or increase signage in high-risk areas.

6. Multilingual Voice Assistant Integration

Assist local users through voice alerts in regional languages based on AI predictions (e.g., via Google Assistant or Alexa APIs).

7. Scalability to Multiple Cities or Countries

Enhance the system's adaptability to different geographic regions by incorporating localized datasets, rules, and driving behavior patterns.

15. Team Members and Roles

NAME	ROLE	RESPONSIBILITIES
SANTHOSH KUMAR.B	Project Lead / Data Scientist	Lead the overall project and coordinate between team members. Supervise data collection, preprocessing, and analysis. Develop machine learning models for accident prediction and analysis.
MOHAMMED KAIF.M.S	Machine Learning Engineer	Implement and optimize machine learning algorithms for accident prediction. Work with TensorFlow, PyTorch, or XGBoost to build and test predictive models.
		Analyze model performance and tune hyperparameters for improved accuracy

RAKESH.N	Geospatial Data Specialist	Manage geographic and road network data for accident hotspot analysis. Use tools like QGIS, ArcGIS, and GeoPandas for spatial analysis and mapping. Work with OpenStreetMap or Google Maps API to improve route prediction and traffic safety.
FARHAAN ABBAS.R	Data Engineer / API Developer	Design and implement the data pipeline for realtime data collection and integration. Develop APIs (Flask/FastAPI) to integrate machine learning models into realtime applications. Ensure the seamless flow of data from sensors, cameras, and traffic systems to the AI models.
MOHAMMED HARRIS.H	Data Visualization Specialist / Dashboard Developer	Design and implement the data pipeline for realtime data collection and integration. Develop APIs (Flask/FastAPI) to integrate

		machine learning models into real-time applications. Ensure the seamless flow of data from sensors, cameras, and traffic systems to the AI models.