Phase-3

Student Name: SANJAY.U

Register Number: 421223104073

Institution:Karpaga Vinayaga College of Engineering and Technology

Department: BE - Computer Science and Engineering

Date of Submission: 26.04.2025

Github Repository Link:

https://github.com/Sanjay4413886/Enhancing-Road-Safety-with-AI-Driven-Traffic-Accident-Analysis-and-Prediction

---

PROJECT TITLE:

- Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction

1. Problem Statement

Road accidents pose a serious threat to public safety, resulting in significant loss of life and economic damage. Traditional methods of traffic monitoring and accident prevention are often reactive rather than proactive. There is a critical need for intelligent systems capable of analyzing traffic patterns, predicting accident risks, and providing actionable insights to prevent accidents before they occur. This project aims to leverage AI-driven data analysis and machine learning models to forecast potential accident hotspots and contribute to safer transportation environments.

2. Objectives of the Project

- Analyze historical traffic accident data to identify patterns and risk factors.
- Build predictive models capable of forecasting accident-prone zones.
- Develop a real-time alert system to inform authorities and drivers about potential risks.
- Provide actionable recommendations to improve traffic safety measures.
- Build a user-friendly dashboard for visualizing accident trends and predictions.

3. Scope of the Project

Features:

- In-depth analysis of traffic-related datasets (weather, time, road conditions, vehicle types).

- Application of AI techniques like classification, clustering, and time-series prediction.
- Identification of high-risk areas and accident hotspots.

Limitations:

- Predictions rely heavily on the availability and quality of traffic and accident datasets.
- The model's performance may vary across different geographic regions.

Constraints:

- Only publicly available or government-published traffic accident datasets will be used.
- Focus will be on prediction and analysis; implementation of physical interventions (e.g., road repairs) is outside the project's scope.

4. Data Sources

- **Dataset**: Road Accident Data (e.g., National Highway Traffic Safety Administration, Kaggle public datasets)
- **Sources**:
  - Kaggle - US Accidents (3.0 million records)
  - Government traffic accident reports and open datasets.
- **Type**: Public, time-series, and geo-spatial data.

5. High-Level Methodology

Data Collection:

- Download accident datasets from public sources.

Data Cleaning:

- Handle missing or inconsistent data entries.
- Normalize weather and location features.

Exploratory Data Analysis (EDA):

- Visualize accident frequency based on time, location, weather, and road conditions.
- Identify correlations between factors and accident occurrences.

Feature Engineering:

- Create new features like "peak traffic hours", "adverse weather indicator", etc.
- Use geospatial features (latitude, longitude clustering).

Model Building:

- Models: Random Forest, XGBoost, Decision Trees, LSTM for time-series accident prediction.

- Justification: Ensemble models and sequence models help capture complex patterns and trends.

Model Evaluation:

- Metrics: Accuracy, Precision, Recall, F1-Score, AUC-ROC for classification tasks.
- Validation Strategy: Stratified K-Fold Cross Validation.

Visualization & Interpretation:

- Accident heatmaps.
- Risk-level classification maps.

Deployment:

- Build a dashboard using Streamlit to visualize accident hotspots and risk predictions in real time.

7. Source Code

import cv2 import torch import datetime import os import random import numpy as np import json import time

print("Loading YOLOv5 model...") model = torch.hub.load('ultralytics/yolov5', 'yolov5s') model.conf = 0.4

DANGER_LINE_Y = 300 FRAME_WIDTH = 640 FRAME_HEIGHT = 480

MONITOR_CLASSES = ['person', 'car', 'motorcycle', 'bicycle', 'truck', 'bus']

LOG_DIR = "road_safety_logs" os.makedirs(LOG_DIR, exist_ok=True)

VIOLATION_IMAGE_DIR = os.path.join(LOG_DIR, "violation_images") os.makedirs(VIOLATION_IMAGE_DIR, exist_ok=True)

SESSION_LOG_FILE = os.path.join(LOG_DIR, "session_summary.json")

session_data = { "start_time": str(datetime.datetime.now()), "violations": [] }

def log_violation(class_name, frame, speed): timestamp = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S') log_entry = f"{timestamp}: {class_name} crossed the danger line at {speed} km/h.\n" with open(os.path.join(LOG_DIR, "violations.txt"), "a") as f: f.write(log_entry) image_path = os.path.join(VIOLATION_IMAGE_DIR, f"{class_name}_{timestamp}.jpg") cv2.imwrite(image_path, frame) print(log_entry.strip()) session_data["violations"].append({ "time": timestamp, "class": class_name, "speed": speed, "image_path": image_path }) def draw_info_panel(frame, fps, count): cv2.rectangle(frame, (0, 0), (FRAME_WIDTH, 50), (50, 50, 50), -1) cv2.putText(frame, f"FPS: {fps:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 255, 255), 2) cv2.putText(frame, f"Violations: {count}", (150, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255), 2)

def simulate_speed(): return random.randint(30, 100)

def save_session_summary(): session_data["end_time"] = str(datetime.datetime.now()) with open(SESSION_LOG_FILE, "w") as json_file: json.dump(session_data, json_file, indent=4)

def display_warning_banner(): print("==============================") print(" ROAD SAFETY MONITORING ") print("==============================") print("Live detection in progress...") print("Press 'q' to quit and generate report.") print("==============================")

def main(): cap = cv2.VideoCapture(0) cap.set(cv2.CAP_PROP_FRAME_WIDTH, FRAME_WIDTH) cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT)

```
display_warning_banner()
```

```
violation_count = 0

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    start = datetime.datetime.now()

    results = model(frame)
    labels, cords = results.xyxyn[0][:, -1],
results.xyxyn[0][:, :-1]

    for i in range(len(labels)):
        row = cords[i]
        if row[4] >= 0.4:
            x1, y1, x2, y2 =
int(row[0]*FRAME_WIDTH),
int(row[1]*FRAME_HEIGHT),
int(row[2]*FRAME_WIDTH),
int(row[3]*FRAME_HEIGHT)
            class_id = int(labels[i])
            class_name = model.names[class_id]

            cv2.rectangle(frame, (x1, y1), (x2,
y2), (0, 255, 0), 2)
            cy = (y1 + y2) // 2

            cv2.putText(frame, class_name, (x1,
```

```
y1 - 10),

cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 255),
2)
            if class_name in MONITOR_CLASSES
and cy < DANGER_LINE_Y:
                speed = simulate_speed()
                cv2.putText(frame,
"Violation!", (x1, y1 - 30),

cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)
                cv2.putText(frame, f"Speed:
{speed} km/h", (x1, y2 + 20),

cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 255),
2)
                log_violation(class_name,
frame, speed)
                violation_count += 1

    end = datetime.datetime.now()
    fps = 1 / (end - start).total_seconds()

    cv2.line(frame, (0, DANGER_LINE_Y),
(FRAME_WIDTH, DANGER_LINE_Y), (255, 0, 0), 2)
    draw_info_panel(frame, fps,
violation_count)
```

```
    cv2.imshow("AI Road Safety Monitor", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
save_session_summary()
print("Monitoring session ended. Summary saved
to:", SESSION_LOG_FILE)
```

if **name** == "**main**": main()

6. Tools and Technologies

- **Programming Language**: Python
- **Notebook/IDE**: Jupyter Notebook, Google Colab
- **Libraries**: pandas, numpy, scikit-learn, matplotlib, seaborn, xgboost, folium (for maps), streamlit
- **Optional Deployment Tools**: Streamlit or Flask for web deployment

7. Team Members and Roles

1. SANJAY.K – DATA ENGINEER
2. SANJAY.U – MACHINE LEARNING ENGINEER

3.SANTHOSH BABU.S – FULL STACK DEVELOPER