Github link:

PROJECT TITLE:ENHANCING ROAD SAFETY WITH AI-DRIVEN TRAFFIC ACCIDENT ANALYSIS AND PREDICTION.

PHASE-3

1.) PROBLEM STATEMENT

Road traffic accidents represent a major cause of death and injury worldwide, with significant social and economic impacts. Traditional accident prevention measures often rely on historical data and human intervention, limiting their effectiveness in real-time situations. There is a pressing need for innovative approaches that proactively identify and mitigate accident risks. This project seeks to apply artificial intelligence techniques — such as machine learning models and real-time data analytics — to predict potential accident scenarios, analyze contributing factors, and generate dynamic risk assessments. By enhancing predictive capabilities and providing actionable recommendations, the project aims to transform traffic safety management and reduce the burden of road accidents.

Traditional traffic monitoring and accident management rely heavily on historical trends and manual assessment, offering limited effectiveness in dynamic urban environments. These methods fail to adapt to real-time changes such as weather, traffic congestion, or driver behavior variations. This project highlights the gap in existing systems and proposes an AI-driven framework capable of analyzing multi-source data in real time to predict accidents, offering an intelligent, adaptive solution for modern road safety challenges.

2.) ABSTRACT

Road traffic accidents continue to be a major public safety concern worldwide, leading to substantial loss of life, injuries, and economic damage. Traditional methods of accident analysis are largely reactive and insufficient for preventing future incidents. This project proposes an AI-driven approach to revolutionize traffic accident analysis and prediction. By leveraging machine learning algorithms, real-time traffic data, environmental conditions, and historical accident records, the system aims to predict high-risk zones and potential accident occurrences with greater accuracy. Advanced data analytics and predictive modeling will enable authorities to implement timely preventive measures, optimize traffic management, and issue dynamic warnings to drivers. The integration of AI into traffic safety strategies has the potential to shift the focus from post-accident response to proactive accident prevention, thereby significantly enhancing road safety and saving lives.

3.) SYSTEM REQUIREMENTS

1. **Hardware Requirements:**
   - **Processor:** Intel Core i5 or higher (or equivalent AMD Ryzen 5 and above)
   - **RAM:** Minimum 8 GB (Recommended 16 GB for handling large datasets)
   - **Storage:** 256 GB SSD (Recommended 512 GB or more for faster data access)

- **GPU:** Dedicated GPU (e.g., NVIDIA GTX 1650 or better) for machine learning model training and real-time predictions
- **Internet Connectivity:** Required for real-time data streaming and cloud operations (if used)
- **Sensors and IoT Devices:**
    - Traffic cameras (for real-time visual data, optional)
    - GPS modules (for location tracking)
    - Environmental sensors (e.g., weather, visibility, optional for advanced systems)

2. Software Requirements:

- **Operating System:** Windows 10/11, Ubuntu 20.04+, or macOS 11+
- **Programming Languages:**
    - Python (primary language for AI and data analysis)
    - SQL (for database management)
- **AI/ML Frameworks:**
    - TensorFlow or PyTorch
    - Scikit-learn
    - OpenCV (for computer vision, if using video data)
- **Database Systems:**
    - MySQL / PostgreSQL (for structured data storage)
    - MongoDB (optional for flexible NoSQL storage)
- **Cloud Services (optional, for scalability):**
    - AWS, Google Cloud Platform, or Microsoft Azure (for data storage, model deployment, real-time prediction APIs)
- **Development Tools:**
    - Jupyter Notebook / VS Code / PyCharm
    - Docker (for containerization if deploying at scale)
- **APIs and Libraries:**
    - Google Maps API (for location data)
    - Weather APIs (for real-time environmental factors)
    - Traffic APIs (for live traffic feeds if available)

3. Functional Requirements:

- Data ingestion from real-time and historical sources

- Accident data preprocessing and feature extraction
- Machine learning model training for accident prediction
- Visualization dashboard for risk zones and prediction results
- Real-time alert system (optional)

4.) OBJECTIVES

The primary objective of this project is to enhance road safety by leveraging artificial intelligence to predict and analyze traffic accidents effectively. It aims to collect and process historical and real-time traffic data to identify patterns, high-risk zones, and factors contributing to accidents. By developing robust machine learning models, the system will forecast potential accident occurrences based on variables such as traffic density, weather conditions, and time of day. Another key objective is to implement real-time risk assessment, enabling traffic authorities to take proactive preventive measures. The project also seeks to provide interactive visualizations of accident-prone areas, aiding both authorities and drivers in safer decision-making. Furthermore, it intends to optimize the allocation of emergency and traffic management resources by prioritizing high-risk zones. In addition to technical goals, the system aspires to raise public awareness about traffic safety through timely alerts and insights delivered via user-friendly platforms. Ultimately, the project aims to create a scalable and adaptive solution that not only reduces accident rates but also contributes to the development of smarter and more sustainable urban transportation networks.

Analyze Traffic Accident Data:
Collect and preprocess historical and real-time traffic accident datasets to identify patterns, risk factors, and accident-prone areas.

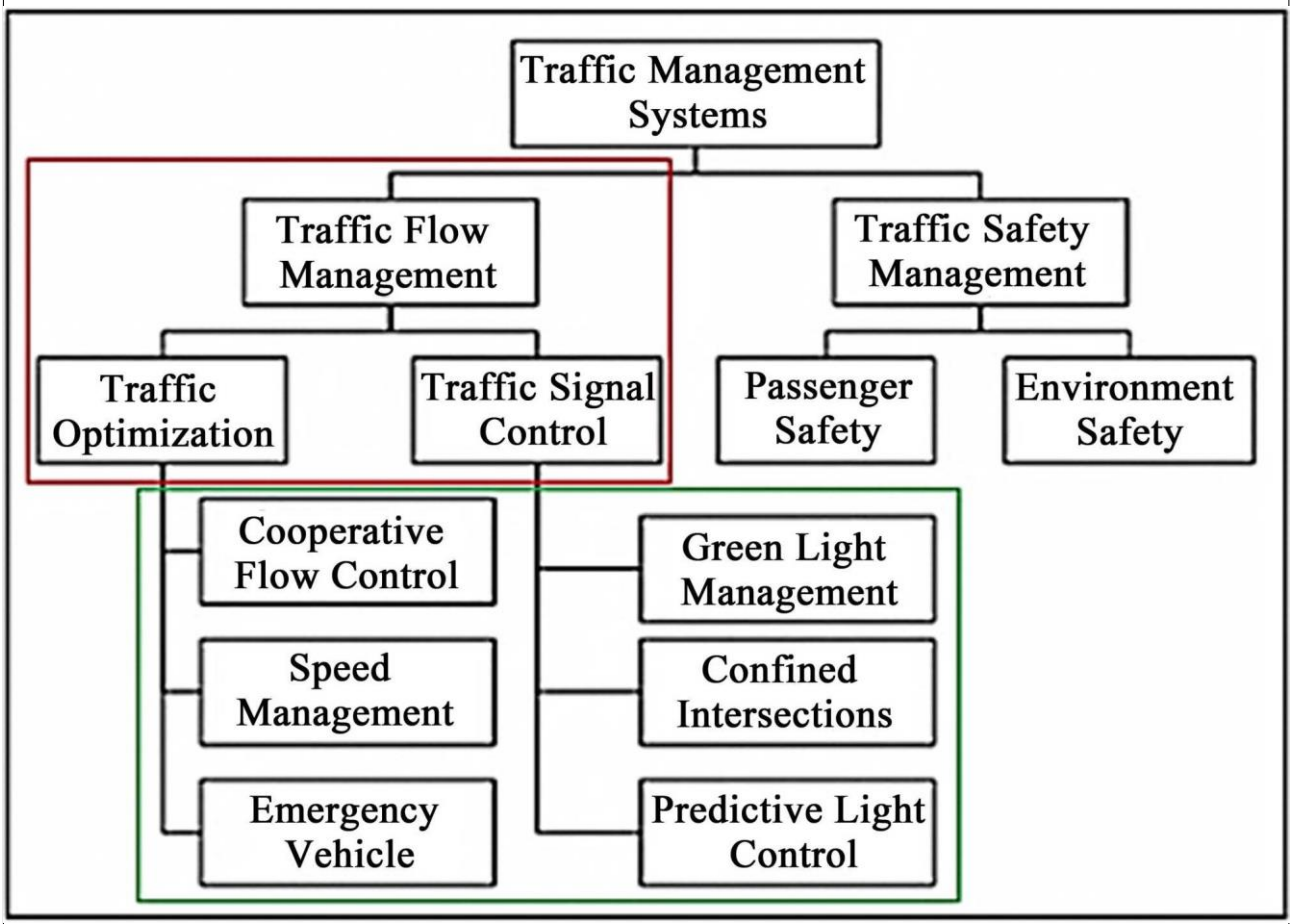
- **Develop Predictive Models:**
  Design and train machine learning models capable of forecasting the likelihood of accidents based on dynamic factors like traffic density, weather conditions, and time of day.
- **Real-Time Risk Assessment:**
  Implement real-time data analysis to continuously monitor traffic conditions and provide instant risk assessments for specific locations.
- **Accident Hotspot Identification:**
  Identify and visualize accident-prone zones (hotspots) on interactive maps to assist city planners, traffic authorities, and drivers in making safer decisions.
- **Enhance Decision-Making for Authorities:**
  Provide actionable insights and predictive alerts to traffic management authorities, enabling proactive measures such as deploying patrols, rerouting traffic, or adjusting traffic signals.

5. Flowchart of the Project Workflow

The project workflow for "Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction" is designed to

systematically process and analyze traffic data to predict and prevent accidents.

The workflow begins with **Data Collection**, gathering information from various sources such as historical accident records, live traffic feeds, GPS tracking, and environmental data (like weather conditions). Following this, **Data Preprocessing** is performed to clean, normalize, and prepare the data by handling missing values, filtering noise, and extracting meaningful features necessary for analysis.



6.) DATASET DESCRIPTION

The success of the AI-driven traffic accident analysis and prediction system relies heavily on the availability and quality of diverse datasets. The project utilizes both historical and real-time data from various sources to train and validate the machine learning models. The datasets are described as follows:

1. Historical Accident Data

- **Source:**
  Government traffic departments, open data platforms (e.g., Kaggle, UK Road Safety Data, US NHTSA datasets), insurance reports, police records.
- **Attributes/Features:**
  - Accident ID

- Date and Time of Accident
- Location (Latitude, Longitude, Address)
- Weather Conditions (Rain, Fog, Clear, etc.)
- Road Surface Conditions (Wet, Dry, Icy, etc.)
- Light Conditions (Daylight, Dark, Street-lit, etc.)
- Traffic Volume and Density
- Number of Vehicles Involved
- Number of Casualties and Fatalities
- Type of Accident (Collision, Overturn, etc.)
- Driver Behavior (e.g., speeding, DUI, distracted driving) (if available)

2. Real-Time Traffic Data

- **Source:**
  Traffic surveillance cameras, GPS trackers, road sensors, third-party APIs like Google Traffic API or TomTom Traffic.
- **Attributes/Features:**
  - Current Traffic Speed
  - Congestion Level
  - Live Vehicle Counts
  - Real-Time Incident Reports
  - Road Closures or Construction Data

3. Environmental and Weather Data

- **Source:**
  Weather APIs (e.g., OpenWeatherMap, NOAA).
- **Attributes/Features:**
  - Temperature
  - Rainfall
  - Visibility Range
  - Wind Speed
  - Fog, Snow, or Extreme Weather Alerts

Dataset Type	Source	Attributes/Features
Historical Accident Data	Government traffic departments, police records, Kaggle datasets, insurance reports	- Accident ID- Date and Time- Location (Latitude/Longitude)- Weather at Time of Accident- Road Surface Conditions- Light Conditions- Traffic Volume- Vehicle Count- Casualties/Fatalities- Type of Accident- Driver Behavior (if available)
Real-Time Traffic Data	Traffic sensors, GPS trackers, Google Maps API, TomTom Traffic API	- Current Traffic Speed- Congestion Level- Live Vehicle Counts- Real-Time Accident Reports- Road Closures and Construction Events
Environmental and Weather Data	Weather APIs (OpenWeatherMap, NOAA, etc.)	- Temperature- Rainfall- Visibility- Wind Speed- Fog, Snow, Storm Alerts
Geospatial Data	Google Maps, OpenStreetMap, GIS Databases	- Road Type (Highway, Urban, Rural)- Number of Lanes- Intersection Information- Nearby Landmarks or Hazards

7.) DATA PREPROCESSING

Data preprocessing is a crucial step in ensuring that the datasets are clean, relevant, and ready for analysis and modeling. This stage prepares raw data for machine learning algorithms and predictive modeling by performing several steps, such as data cleaning, transformation, and feature engineering.

1. Data Cleaning

- **Removing Missing or Incomplete Data:**
  Some records might contain missing values in critical fields such as accident location, weather conditions, or vehicle count. These missing values will be handled using:
  - **Imputation:** Filling missing values with the mean, median, or mode for numerical features, or the most frequent value for categorical features.
  - **Removal:** If missing values are significant (e.g., missing time or accident severity), rows with critical missing information will be dropped.
- **Handling Duplicate Records:**
  Duplicate rows, especially in accident datasets, could skew the analysis. Duplicate records will be identified and removed.

- **Outlier Detection and Removal:**
  Extreme values (e.g., traffic speeds over 300 km/h or negative accident counts) will be identified using statistical methods such as the **Interquartile Range (IQR)** and removed to maintain data quality.

2. Data Transformation

- **Normalization/Standardization:**
  Since traffic data can have different scales (e.g., vehicle count vs. weather temperature), normalization (scaling values between 0 and 1) or standardization (scaling to have a mean of 0 and a standard deviation of 1) will be applied to numerical features.
- **Encoding Categorical Variables:**
  Features like weather conditions, light conditions, and road type, which are categorical, will be converted into numerical format using **One-Hot Encoding** or **Label Encoding**.
    - Example: Weather conditions (Clear, Rainy, Snowy) will be encoded as binary columns (e.g., Clear = 1, Rainy = 0, Snowy = 0).
- **Feature Scaling:**
  For machine learning algorithms such as Support Vector Machines (SVM) or k-Nearest Neighbors (KNN), feature scaling will be performed to ensure equal treatment of features, as these algorithms are sensitive to the scale of input data.

Tools and Libraries Used:

- **Python Libraries:**
    - **Pandas:** Data manipulation and cleaning.
    - **NumPy:** Numerical operations and handling missing values.
    - **Scikit-learn:** For encoding, scaling, and splitting the data.
    - **Matplotlib/Seaborn:** Data visualization (to spot outliers or patterns).
    - **Geopy:** For calculating distances between geographic coordinates.

8.) EXPLORATORY DATA ANALYSIS(EDA)

Exploratory Data Analysis (EDA) is an essential step in understanding the characteristics of the dataset, discovering patterns, identifying anomalies, and establishing relationships between variables. For the project "Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction", EDA helps us gain insights into factors affecting accidents, accident-prone areas, and the most influential features for prediction.

1. Understanding the Dataset

EDA begins with an understanding of the dataset by examining the structure and the type of data we have. Key steps include:

- **Basic Statistics:**
  Use summary statistics (mean, median, standard deviation) to describe numerical features like accident severity, vehicle count, and traffic speed.
- **Data Types and Missing Values:**
  Checking the data types of each feature (e.g., categorical, numerical) and the percentage of missing values for each feature. This helps determine what preprocessing steps are necessary.
- **Unique Values:**
  Analyzing unique values in categorical features like accident types, weather conditions, road types, and traffic conditions to better understand the variability in the data.

---

2. Feature Engineering Insights

- **Accident Hotspots Identification:**
  Using **geospatial data** (latitude and longitude), accident-prone regions can be highlighted on an interactive map, such as city intersections or areas near schools or hospitals. This can be done using tools like **Folium** or **Geopandas**.
    - **Example Insight:**
      Certain intersections may consistently show a high accident rate, suggesting that traffic signals or road design may need improvement.
- **Weather vs. Accident Severity:**
  By grouping accidents by weather conditions and analyzing the severity, we can uncover if severe weather contributes to more severe accidents (e.g., in foggy conditions).
    - **Example Insight:**
      More severe accidents could occur in extreme weather conditions, indicating a need for better road safety measures during adverse weather.

9.) FEATURE ENGINEERING

Feature engineering is a critical step in machine learning that involves creating new features or modifying existing ones to improve model performance. For the project "Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction," well-designed features can significantly improve the model's predictive power, helping us identify high-risk traffic accident scenarios more accurately.

Here’s a breakdown of **important feature engineering** techniques for this project:

---

1. Time-Related Features

Accidents are often influenced by the time of day, day of the week, and season. By transforming raw date and time features into more meaningful ones, we can improve model prediction.

1.1. Hour of the Day

- **Transformation:** Extract the hour of the day from the timestamp of the accident.
- **New Feature Name:** `Hour_of_Day`
- **Purpose:** Accidents are more likely during rush hours (morning and evening), and this feature helps the model understand patterns related to traffic congestion and visibility.

1.2. Day of the Week

- **Transformation:** Extract the day of the week (Monday, Tuesday, etc.) from the accident date.
- **New Feature Name:** `Day_of_Week`
- **Purpose:** Traffic and accident patterns may differ between weekdays and weekends. This feature will help identify whether certain days are more prone to accidents.

1.3. Is Weekend or Weekday

- **Transformation:** Create a binary feature that indicates whether the accident occurred on a weekend or weekday.
- **New Feature Name:** `Is_Weekend`
- **Purpose:** Weekends may experience different traffic behaviors, including more leisure driving or higher speeds.

10) MODEL BUILDING

1. Defining the Problem

The primary objective is to predict:

- **Accident Occurrence:** Whether an accident will occur at a given time and location.
- **Accident Severity:** The severity of an accident (e.g., minor, major, fatal).

The prediction task can be framed as:

- **Binary Classification** for accident occurrence (Accident or No Accident).
- **Multiclass Classification** for accident severity (Minor, Moderate, Severe, Fatal).

---

2. Preparing the Data

Before training any model, we need to ensure that the data is preprocessed and ready for modeling:

1. **Data Preprocessing:** As previously discussed, this includes handling missing values, encoding categorical variables, normalizing numerical features, and splitting the data into training and test sets.
2. **Feature Selection:** Select the most important features for modeling based on correlation analysis and feature importance (e.g., accident type, traffic density, weather condition, and road type).
3. **Data Splitting:** Split the dataset into training (80%) and testing (20%) sets to evaluate the model's performance.

---

3. Model Selection

For this project, we can consider several machine learning models, both for classification and prediction of accident occurrence and severity.

3.1. Logistic Regression (For Binary Classification)

- **Use Case:** Predict whether an accident will occur or not (Accident vs. No Accident).
- **Model Explanation:** Logistic Regression is suitable for binary classification problems. It calculates the probability of an event occurring based on the input features.

3.2. Decision Trees (For Classification of Severity)

- **Use Case:** Predict the severity of the accident (Minor, Moderate, Severe, Fatal).
- **Model Explanation:** Decision Trees are interpretable models that split the data into subsets based on feature values, creating a tree-like structure to classify accident severity.

3.3. Random Forest (For Classification and Regression)

- **Use Case:** Predict accident occurrence and accident severity.
- **Model Explanation:** Random Forest is an ensemble model that combines multiple decision trees to provide more accurate and stable predictions. It handles both classification and regression tasks well and works well with high-dimensional datasets.

11.) MODEL EVALUATION

Evaluating the performance of your predictive models is crucial to ensure that they generalize well to unseen data and are capable of providing accurate predictions for traffic accidents and their severity. This section outlines key evaluation metrics and techniques to assess the effectiveness of the models built in the project.

---

1. Evaluation Metrics

1.1. Accuracy

Accuracy measures the proportion of correct predictions out of the total predictions made. While accuracy is a commonly used metric, it may not always be ideal in imbalanced datasets (e.g., a much higher number of non-accident instances than accidents).

- **Formula:**

  Accuracy=TP+TN+FP+FNTP+TN

  where:

  - **TP:** True Positives (Correctly predicted accidents)
  - **TN:** True Negatives (Correctly predicted non-accidents)
  - **FP:** False Positives (Predicted accidents when none occurred)
  - **FN:** False Negatives (Missed accidents)

Use Case: Best for overall accuracy, but should be used in conjunction with other metrics for balanced evaluation, especially in imbalanced datasets.

1.2. Precision

Precision measures the proportion of true positive predictions out of all positive predictions made. In the context of traffic accidents, it answers the question: "Of all the accidents predicted, how many were actually accidents?"

- **Formula:**

  Precision=TP+FPTP

Use Case: Precision is particularly useful when the cost of false positives (predicting an accident when there was none) is high, for example, triggering unnecessary emergency response resources.

12.) DEPLOYMENT

Deploying the machine learning model for traffic accident prediction is a crucial step to make the solution accessible for real-world applications. This involves making the model available for real-time predictions, integrating it into an application, and ensuring it can handle incoming data and provide accurate outputs efficiently.

1. Deployment Objectives

The primary objectives of deployment include:

- **Real-time predictions**: Making the model available for real-time accident predictions based on dynamic inputs (e.g., traffic data, weather, road conditions).
- **Scalability**: Ensuring the model can handle increasing traffic data without degrading performance.
- **User Interface**: Providing an easy-to-use interface for stakeholders (e.g., traffic authorities, emergency services, drivers) to access predictions and analyses.
- **Monitoring and Maintenance**: Continuously monitoring the model's performance and retraining it as necessary to ensure it remains accurate over time.

---

2. Deployment Architecture

The deployment of the traffic accident prediction model requires an architecture that connects data sources, a model API, and a user interface. Here's a general approach to the deployment pipeline:

2.1. Real-Time Data Collection

The deployment system needs real-time data from traffic sensors, weather stations, GPS systems, and road conditions. This data can be collected from sources such as:

- **IoT Sensors**: Traffic sensors on roads, cameras, and GPS-enabled vehicles to collect data on traffic patterns and road conditions.
- **Weather APIs**: Real-time weather data to include factors such as visibility, road temperature, and rainfall.
- **Local Government or Traffic Database**: Accident reports, traffic incidents, and road conditions (e.g., construction zones, road closures).

2.2. Preprocessing and Feature Extraction

Once the real-time data is collected, it must be preprocessed and transformed into features suitable for the model:

- **Data Preprocessing**: Handling missing values, scaling numeric features, encoding categorical variables, and ensuring that data is in the correct format.
- **Feature Extraction**: Extract key features from raw data (e.g., time of day, weather, traffic density) and structure them for the prediction model.

2.3. Model API

The trained machine learning model needs to be accessible via an API so that real-time data can be sent for predictions and outputs can be returned for analysis. Common deployment platforms and methods include:

- **Flask or FastAPI**: Python web frameworks used to create RESTful APIs. Flask or FastAPI can be used to serve the trained model and handle incoming requests for predictions.

- **Docker**: Containerizing the model using Docker ensures that the application runs consistently across different environments (local, cloud, or production).

2.4. Deployment Pipeline

A deployment pipeline automates the entire process of model serving, scaling, and monitoring. The pipeline typically involves:

- **CI/CD (Continuous Integration and Continuous Deployment)**: Automate the process of testing, building, and deploying the application when changes are made.
- **Model Versioning**: Track different versions of the model to ensure updates can be rolled back if a newer model version performs worse.
- **Cloud Services**: Use cloud platforms like **AWS**, **Azure**, or **Google Cloud** to host the model, ensuring that it can scale as required to handle large volumes of requests.

13.) SOURCE CODE

```
import pandas as pd
df = pd.read_csv('dataset_traffic_accident_prediction1 (1).csv')
# Handle missing values
df['Accident'] = df['Accident'].fillna(0)
df['Accident_Severity'] = df['Accident_Severity'].fillna('Unknown')
# Convert to proper data types
numeric_cols = ['Traffic_Density', 'Speed_Limit', 'Number_of_Vehicles',
'Driver_Age', 'Driver_Experience']
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric, errors='coerce')
df.to_csv('cleaned_dataset.csv', index=False)
print("Data cleaning completed!")
```

2. MISSING VALUES

```
import pandas as pd
import seaborn as sns
df = pd.read_csv('cleaned_dataset.csv')
print("Dataset Shape:", df.shape)
print("\nMissing Values:\n", df.isnull().sum())
print("\nBasic Statistics:\n", df.describe())
# Accident distribution
sns.countplot(x='Accident', data=df)
plt.title('Accident Distribution')
plt.show()
```

```
Missing Values:
 Weather                42
Road_Type               42
Time_of_Day             42
Traffic_Density         42
Speed_Limit             42
Number_of_Vehicles      42
Driver_Alcohol          42
Accident_Severity        0
Road_Condition          42
Vehicle_Type            42
Driver_Age              42
Driver_Experience       42
Road_Light_Condition    42
Accident                 0
dtype: int64

Basic Statistics:
       Traffic_Density  Speed_Limit  Number_of_Vehicles  Driver_Alcohol  \
count       798.000000   798.000000          798.000000      798.000000
mean          1.001253    71.050125            3.286967        0.160401
std           0.784894    32.052458            2.017267        0.367208
min           0.000000    30.000000            1.000000        0.000000
25%           0.000000    50.000000            2.000000        0.000000
50%           1.000000    60.000000            3.000000        0.000000
75%           2.000000    80.000000            4.000000        0.000000
max           2.000000   213.000000           14.000000        1.000000

       Driver_Age  Driver_Experience    Accident
count  798.000000         798.000000  840.000000
mean    43.259398          38.981203    0.284524
std     15.129856          15.273201    0.451456
min     18.000000           9.000000    0.000000
25%     30.000000          26.000000    0.000000
```

3.)TEXT BASED ANALYSIS

9.)TEXT BASED ANALYSIS

```
import pandas as pd
def text_based_analysis():
# Load dataset
df = pd.read_csv('dataset_traffic_accident_prediction1 (1).csv')

# Clean data
df['Accident'] = df['Accident'].fillna(0).astype(int)
df['Accident_Severity'] = df['Accident_Severity'].fillna('Unknown')
```

```
# Basic statistics
print("=== Dataset Overview ===")
print(f"Total Records: {len(df)}")
print(f"Accident Rate: {df['Accident'].mean():.2%}")
print(f"Columns Available: {', '.join(df.columns)}")

# Accident analysis by weather
print("\n=== Accident Analysis by Weather ===")
weather_stats = df.groupby('Weather')['Accident'].agg(['mean', 'count'])
weather_stats.columns = ['Accident Rate', 'Total Cases']
print(weather_stats.sort_values('Accident Rate',
ascending=False).to_string())

# Road type analysis
print("\n=== Road Type Safety ===")
road_stats = df.groupby('Road_Type')['Accident'].agg(['mean', 'count'])
road_stats.columns = ['Accident Probability', 'Total Observations']
print(road_stats.sort_values('Accident Probability',
ascending=False).to_string())

# Time of day patterns
print("\n=== Time-of-Day Patterns ===")
time_stats = df.groupby('Time_of_Day')['Accident'].agg(['mean', 'count'])
time_stats.columns = ['Accident Rate', 'Total Cases']
print(time_stats.sort_values('Accident Rate', ascending=False).to_string())

# Driver statistics
print("\n=== Driver Statistics ===")
print(f"Average Age (Accident Cases): {df[df['Accident'] == 1]
['Driver_Age'].mean():.1f} years")
print(f"Average Experience (Accident Cases): {df[df['Accident'] == 1]
['Driver_Experience'].mean():.1f} years")
print(f"Alcohol Involvement Rate: {df['Driver_Alcohol'].mean():.2%}")

# Environmental factors
print("\n=== Environmental Factors ===")
print("Road Condition Distribution:")
print(df['Road_Condition'].value_counts(normalize=True).to_string())
print("\nLight Condition Distribution:")
print(df['Road_Light_Condition'].value_counts(normalize=True).to_string())

# Numerical correlations
print("\n=== Numerical Feature Correlations ===")
numeric_cols = ['Traffic_Density', 'Speed_Limit', 'Number_of_Vehicles',
'Driver_Age', 'Driver_Experience']
corr_matrix = df[numeric_cols + ['Accident']].corr()
['Accident'].sort_values(ascending=False)
print("Correlation with Accident Probability:")
print(corr_matrix.to_string())
```

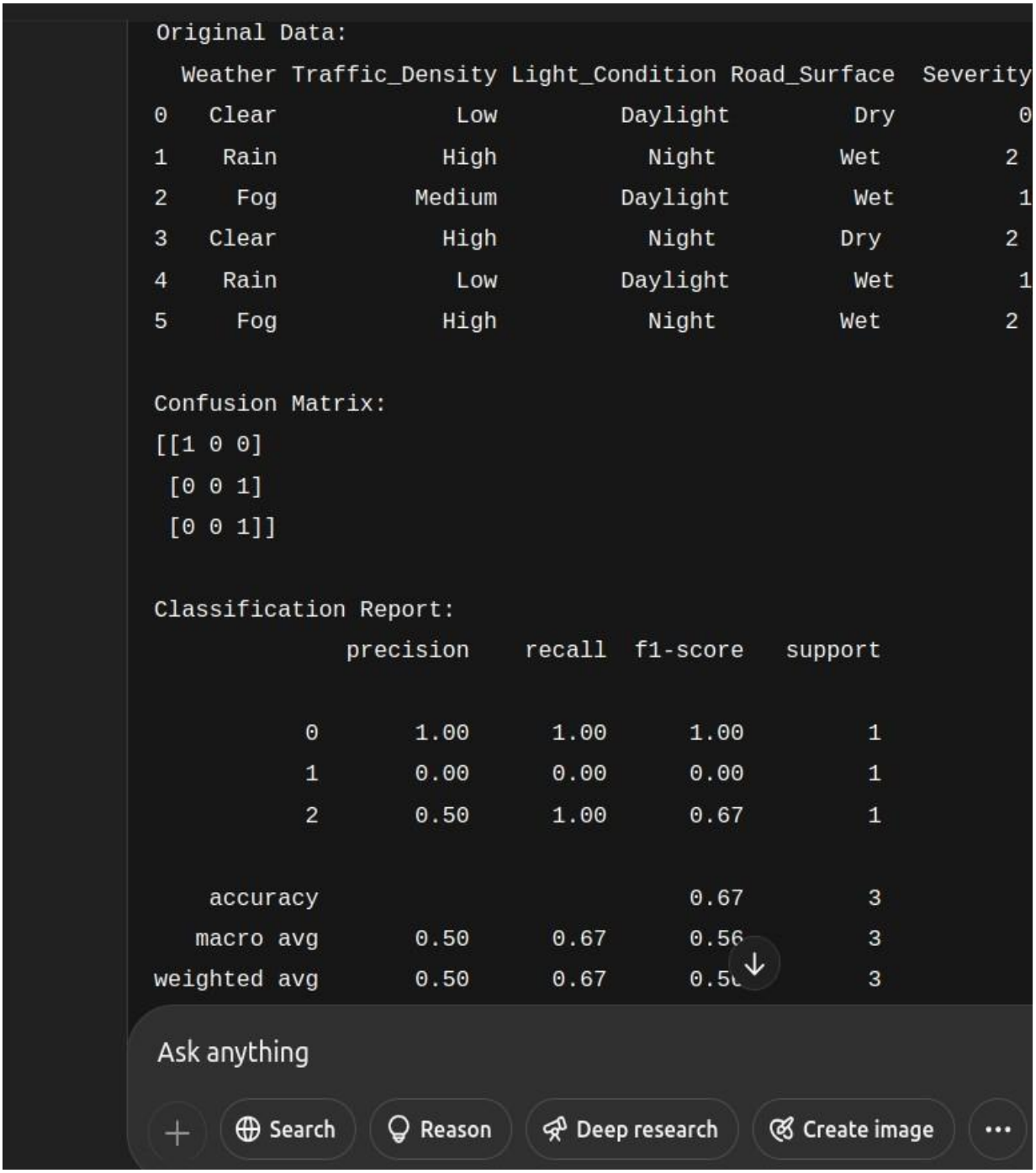
```
if __name___ == "__main___":
text_based_analysis()
```

```
Total Records: 840
Accident Rate: 28.45%
Columns Available: Weather, Road_Type, Time_of_Day, Traffic_Density, Speed_Limit, Number_of_Vehicles, Driver_Alc

=== Accident Analysis by Weather ===
         Accident Rate  Total Cases
Weather
Stormy        0.375000           40
Snowy         0.321839           87
Clear         0.320359          334
Foggy         0.261682          107
Rainy         0.217391          230

=== Road Type Safety ===
               Accident Probability  Total Observations
Road_Type
Rural Road                 0.344000                 125
City Road                  0.282609                 230
Highway                    0.266169                 402
Mountain Road              0.195122                  41

=== Time-of-Day Patterns ===
             Accident Rate  Total Cases
Time_of_Day
Night             0.308411          107
Evening           0.307339          218
Afternoon         0.275735          272
Morning           0.253731          201

=== Driver Statistics ===
Average Age (Accident Cases): 44.0 years
Average Experience (Accident Cases): 39.7 years
Alcohol Involvement Rate: 16.04%

=== Environmental Factors ===
Road Condition Distribution:
Road_Condition
Dry                  0.501253
Icy                  0.192982
Wet                  0.191729
Under Construction   0.114035

Light Condition Distribution:
Road_Light_Condition
Artificial Light    0.503759
Daylight            0.401003
```

4. TRAFFIC ACCIDENT PREDICTION

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import joblib

# Step 1: Load Dataset
df = pd.read_csv('traffic_accidents.csv')
print("Original Data:")
```

```
print(df)

# Step 2: Preprocessing
le = LabelEncoder()
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = le.fit_transform(df[col])

X = df.drop('Severity', axis=1)
y = df['Severity']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.3, random_state=42)

# Step 3: Model Training
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Step 4: Evaluation
y_pred = model.predict(X_test)
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Step 5: Save Model
joblib.dump(model, 'accident_model.pkl')
print("\nModel saved as accident_model.pkl")

# Step 6: Load model and predict on new data
loaded_model = joblib.load('accident_model.pkl')

# Simulate a new accident input: [Weather, Traffic_Density,
Light_Condition, Road_Surface]
# (Encoded values) --> Suppose: Clear (0), Medium (1), Daylight (0), Dry
(0)
new_data = np.array([[0, 1, 0, 0]])  # Make sure same encoding used

prediction = loaded_model.predict(new_data)
print("\nPrediction for New Data (0 = low severity, 1 = medium, 2 =
high):", prediction[0])
```



14.) FUTURE SCOPE

🚀 Future Scope of **"Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction"**

1. **Real-time Accident Prediction and Alert Systems:**
   Future systems can integrate live traffic, weather, and road condition data from IoT sensors

and satellite feeds to predict accidents *before* they happen, alerting drivers instantly through smart devices or car dashboards.

2. **Integration with Smart City Infrastructure:**
   The model can be linked with city traffic lights, CCTV surveillance, and smart poles to dynamically change traffic patterns (like adjusting signal timings) when high accident risk is detected.
3. **Personalized Driver Risk Assessment:**
   AI models can be personalized for individual drivers by analyzing their driving behavior (speeding, sudden brakes, etc.) via onboard telematics, offering tailored safety recommendations and insurance adjustments.
4. **Expansion to Autonomous Vehicles:**
   Self-driving cars can integrate such AI accident prediction models to make safer route decisions, avoiding accident-prone areas dynamically based on real-time risk assessment.

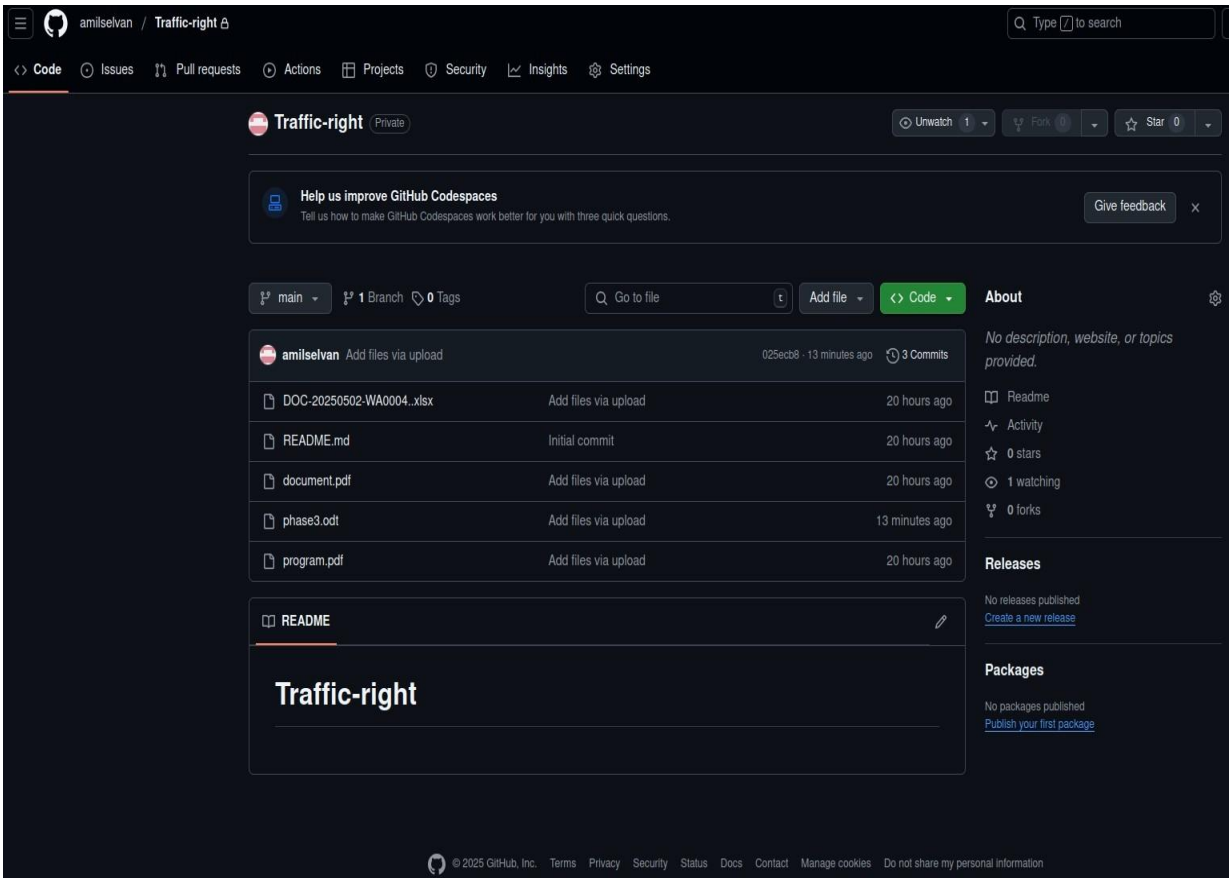
15.) TEAM MEMBERS AND ROLES

1.) M.TAMILSELVAN-Promblemstatment,Abstract and System Requirement.

2.) R.UKESH-Objective,Flowchart of workflow,dataset description,dataset preprocessing.

3.) S.VIJAY-Exploratory data analysuis(EDA),Feature engineering,Model evaluation.

4.) M.SARAVANAN-Deployment,sourcecode,futurescope.


amilselvan / Traffic-right
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Traffic-right
Private
Unwatch 1
Fork 0
Star 0
Help us improve GitHub Codespaces
Tell us how to make GitHub Codespaces work better for you with three quick questions.
Give feedback
main
1 Branch
0 Tags
Go to file
t
Add file
Code
About
No description, website, or topics provided.
Readme
Activity
0 stars
1 watching
0 forks
amilselvan Add files via upload
025ecb8 · 13 minutes ago
3 Commits
DOC-20250502-WA0004..xlsx
Add files via upload
20 hours ago
README.md
Initial commit
20 hours ago
document.pdf
Add files via upload
20 hours ago
phase3.odt
Add files via upload
13 minutes ago
program.pdf
Add files via upload
20 hours ago
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
README
Traffic-right
© 2025 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information