Phase-3 Submission Template

Student Name: MOHAMED RAYEES K

Register Number: 510623104050

Institution: C.ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHONOLOGY

Department: COMPUTER SCIENCE AND ENGINEERING

Date of Submission: 09-05-2025

Github Repository Link:
https://github.com/krayees001/Enhancing-road-safety-with-AI-driven-traffic-accident-analysis-and-prediction-phase3

1. Problem Statement

Road accidents are a significant cause of fatalities and injuries worldwide. Many of these accidents are preventable with timely analysis and prediction. Traditional methods often lack real-time insights and predictive power.

This project aims to leverage AI techniques to analyze traffic accident data and predict high-risk zones and accident probabilities, thereby enhancing road safety and aiding decision-making for traffic authorities and urban planners.

1.Rising Road Accidents:

Road traffic accidents are a major global issue, leading to significant loss of life, injuries, and property damage.

2.Inefficiency of Traditional Methods:

Conventional approaches rely heavily on historical accident reports and static risk assessments, which are reactive and lack predictive capabilities.

3.Fragmented Data Sources:

Critical traffic data is often scattered across different sources—such as police reports, GPS systems, weather services, and road sensors—making it difficult to analyze comprehensively.

4. Lack of Real-Time Insights:

Current systems rarely provide real-time identification of accident-prone zones or immediate risk alerts to drivers and authorities.

5.Underutilization of AI/ML:

Despite advancements in AI and machine learning, these technologies are not widely adopted in the domain of traffic safety prediction and proactive accident prevention.

2.Abstract

Road traffic accidents are a critical public safety concern, contributing to substantial human and economic loss globally. Traditional accident analysis methods often rely on static and historical data, lacking the capacity for real-time assessment or future risk prediction. This project aims to enhance road safety by leveraging artificial intelligence (AI) and machine learning (ML) techniques to analyze traffic accident data, predict potential accident hotspots, and provide actionable insights.

By integrating various data sources such as historical accident records, weather conditions, time, and road characteristics, the system identifies patterns and trains predictive models to estimate accident probabilities. Geospatial visualization tools like Folium are used to generate interactive heatmaps of high-risk zones, while a user-friendly dashboard built with Flask allows traffic authorities and urban planners to monitor and act on predictions effectively.

The project not only improves the understanding of accident causation factors but also supports proactive interventions, such as traffic rerouting, infrastructure improvements, and public awareness campaigns. Ultimately, the solution aims to reduce the frequency and severity of accidents, making roads safer for all users through data-driven, intelligent decision-making

Major Points

- ***Problem Statement:***

Road traffic accidents cause significant loss of life and property. Traditional analysis methods are reactive and lack predictive

capabilities.

 Project Aim:

To utilize Artificial Intelligence and Machine Learning to analyze and predict traffic accidents, thereby improving road safety.

 Data Sources:

Historical accident records, weather data, road conditions, time-of-day patterns, and geographic information.

 Methodology:

- ***Data preprocessing and feature engineering.***
- ***Training predictive models (e.g., Random Forest, XGBoost).***
- ***Generating interactive accident heatmaps using geospatial tools.***
- ***Developing a web-based dashboard for real-time monitoring.***

 Key Technologies Used:

Python, Pandas, Scikit-learn, Folium, Flask, Matplotlib, Seaborn.

 Expected Outcomes:

- ***Identification of high-risk accident zones.***
- ***Real-time accident likelihood predictions.***
- ***Actionable insights for traffic authorities and planners.***
- ***Improved decision-making for accident prevention.***

 Impact:

Helps reduce road accidents and fatalities by shifting from reactive analysis to proactive safety strategies through AI-driven solutions.

3.System Requirements

1. Hardware Requirements

- ***A computer with at least an Intel Core i5 processor (or equivalent).***
- ***Minimum of 8 GB RAM (16 GB recommended for smooth operation).***

- *At least 50 GB of free storage space.*
- *Integrated graphics card (sufficient for non-GPU intensive tasks).*
- *Stable internet connection (for data access, API usage, and cloud deployment).*

2. Software Requirements

- *Operating System: Windows 10, Ubuntu 20.04, or macOS.*
- *Python (version 3.8 or higher).*
- *Jupyter Notebook for data exploration and model building.*
- *Flask or Streamlit for creating the web dashboard.*
- *Web browser (e.g., Chrome or Firefox) to interact with the dashboard.*

3. Python Libraries and Dependencies

- *pandas and numpy for data preprocessing and manipulation.*
- *scikit-learn and xgboost for building and evaluating machine learning models.*
- *matplotlib and seaborn for plotting and visual analytics.*
- *folium for creating interactive maps to show accident hotspots.*
- *flask for building and running the dashboard server.*

4. Optional Tools (for Deployment and Collaboration)

- *Git and GitHub for version control and team collaboration.*
- *Heroku, AWS, or Google Cloud Platform (GCP) for cloud-based deployment.*
- *Docker for packaging and containerizing the application (optional).*
- *VS Code or PyCharm as the Integrated Development Environment (IDE).*

4. Objectives

1. ***Collect real-world traffic accident data from reliable sources such as the UK Road Safety Open Data portal.***

2. ***Clean and preprocess the dataset to ensure data quality and remove inconsistencies.***

3. ***Perform analysis to identify patterns, trends, and hotspots of accident occurrence.***

4. ***Engineer meaningful features from raw data that influence accident severity (e.g., time of day, road surface).***

5. ***Train and evaluate machine learning models like Random Forest and XGBoost to predict accident likelihood and severity.***

6. ***Fine-tune model parameters for improved performance using cross-validation and grid search techniques.***

7. ***Simulate real-time scenarios to demonstrate how the model behaves with dynamic inputs.***

8. ***Visualize results using charts, heatmaps, and graphs to communicate findings effectively.***

9. ***Incorporate additional data sources, like weather conditions, for enhanced model accuracy.***

10. ***Provide actionable insights that can inform policies and road safety improvements..***

Analyze Historical Accident Data
Examine past traffic accident records to identify recurring patterns, contributing factors, and high-risk scenarios.

- ***Build Predictive Models***
  Develop machine learning models to predict accident likelihood based on variables such as location, time, weather, and road conditions.

- ***Identify and Visualize High-Risk Zones***
  Use geospatial mapping to highlight accident-prone areas, aiding urban planners and traffic authorities in hotspot identification.

- ***Provide Actionable Insights***
  Generate meaningful insights that support data-driven decisions for implementing safety measures and infrastructure improvements.

- ***Develop a Real-Time Dashboard***
  Create an interactive web-based dashboard for monitoring predictions, visualizations, and traffic safety alerts in real time.

- ***Improve Road Safety through AI***
  Leverage artificial intelligence to shift from reactive to proactive accident prevention, ultimately reducing injuries and fatalities

5.Flowchart of the Project Workflow

- ***Data Collection***

- *Gather accident data from public sources, traffic APIs, and weather datasets.*

- *Include location, time, road type, weather, and vehicle data.*

- ***Data Preprocessing***

- *Clean missing or inconsistent entries.*
- *Extract relevant features (e.g., time of day, speed limit zone).*
- *Normalize and encode data for model training.*

- ***Exploratory Data Analysis (EDA)***

- *Identify patterns in accident occurrence.*
- *Visualize accident hotspots using charts and geospatial maps.*

- ***Feature Engineering***

- *Create meaningful inputs like risk scores, peak traffic indicators.*
- *Reduce irrelevant features to improve model accuracy.*

- ***Model Development***

- *Apply ML algorithms (e.g., Random Forest, XGBoost).*
- *Train models on labeled accident datasets.*

- ***Model Evaluation***

- *Measure performance using metrics: accuracy, precision, recall, F1-score.*
- *Tune hyperparameters for optimal results.*

- ***Visualization & Dashboard***

- *Create heatmaps of high-risk zones.*
- *Build a dashboard for users to view predictions interactively.*

- ***Deployment***

- *Develop APIs to serve model predictions (Flask or FastAPI).*
- *Deploy on cloud (e.g., Heroku, AWS, or GCP) for real-time use.*

6.Dataset Description

The dataset used in this project contains detailed records of road traffic accidents. It combines spatial, temporal, and contextual information to support the analysis and prediction of accident occurrences

***Source**: Kaggle Traffic Accident Datasets, US DOT, or local government datasets.*

Attributes may include*:*

- Date and time of accident

- Location (latitude, longitude, city/state)

- Weather conditions

- Road type and traffic signal information

- Number and severity of injuries/fatalities

- Vehicle types involved- Cause of accident

Source of Dataset

- ***Primary Sources:***
  - ***Kaggle Public Datasets (e.g., UK/US accident data)***
  - ***U.S. Department of Transportation (US DOT)***
  - ***Local government open data portals***
  - ***APIs (e.g., HERE, OpenWeatherMap for weather data)***

Data Format and Size

- *Format: CSV, JSON, or via API response (for real-time data).*
- *Size: Varies between 100,000 to over 1 million records.*
- *Missing Data: Some fields may have missing or inconsistent values requiring preprocessing.*

Preprocessing Tasks

- *Handling missing values*
- *Normalizing and encoding categorical variables*
- *Extracting useful features (like time of day, traffic peak hours)*
- *Merging with external sources (weather or traffic flow data)*

7) Data Preprocessing (Expanded)

***Data preprocessing is a crucial phase to ensure the dataset is clean, consistent, and ready for model training. Below are the major steps involved**:*

1. Data Cleaning

- ***Remove Missing or Null Values**: Eliminate or impute rows/columns with missing essential data.*
- ***Fix Inconsistencies**: Standardize values in categorical fields (e.g., unify variations like "rainy" and "Rainy").*
- ***Remove Duplicates**: Detect and drop duplicate records to avoid skewing results.*

2. Feature Selection

- ***Drop Irrelevant Columns**: Remove features that do not contribute to prediction (e.g., unique IDs).*
- ***Select Key Features**: Keep important columns such as time, location, weather, and road type.*

3. Feature Engineering

- ***Extract Time Features**: Derive hour, day, month, and day_of_week from timestamp.*
- ***Categorize Weather and Road Conditions**: Group into risk levels (e.g., low, moderate, high).*
- ***Generate Traffic Peak Indicators**: Flag rush hours or night-time periods.*

4. Encoding Categorical Variables

- ***Label Encoding**: For ordinal variables (e.g., accident severity).*
- ***One-Hot Encoding**: For nominal variables (e.g., weather type, road type).*

5. Data Normalization / Scaling

- ***Apply Min-Max or Standard Scaler**: Normalize numerical features (e.g., speed, distance).*
- ***Why?**: Helps improve convergence and performance of machine learning models.*

6. Handling Imbalanced Data

- ***Resampling**: Use techniques like SMOTE (Synthetic Minority Over-sampling Technique) to balance class distribution.*
- ***Class Weight Adjustment**: Assign higher weights to underrepresented classes during training.*

7. Splitting the Dataset

- ***Train-Test Split**: Commonly 80/20 or 70/30 ratio.*
- ***Stratified Split (if needed)**: Maintain proportion of classes (e.g., severity*

levels) in both sets.

8) Exploratory Data Analysis (EDA)

- ***Time Trends: Visualize accidents by hour, weekday, and month.***
- ***Severity Distribution: Plot how many accidents fall into each severity class.***
- ***Location Insights: Generate heatmaps of high-frequency accident areas.***
- ***Weather Correlation: Check how adverse weather affects accident severity.***
- ***Vehicle Type Analysis: Examine which vehicle categories are more accident-prone.***
- ***Road and Light Conditions: Investigate their impact on accident severity.***
- ***Weekday vs Weekend: Compare accident counts and severity.***
- ***Regional Distribution: Show accident rates across different areas.***
- ***Temporal Trends: Track accident data over multiple years.***
- ***Visualization Tools: Use Seaborn and Matplotlib for all plots.***

EDA helps understand the dataset, detect anomalies, and identify patterns or relationships between variables relevant to accident risks.

1. Univariate Analysis

- ***Numerical Features:***
- ***Histograms for speed, number of vehicles, etc.***
- ***Boxplots to detect outliers (e.g., accident durations).***
- ***Categorical Features:***
- ***Count plots for weather types, accident severity, road types.***
- ***Pie charts to show percentage distribution of time of day, accident causes.***

2. Bivariate and Multivariate Analysis

- ***Correlation Matrix:***
- ***To find relationships between numerical features (e.g., speed vs. severity).***
- ***Pair Plots / Scatter Plots:***
- ***For visualizing interactions (e.g., road type vs. accident count).***
- ***Grouped Bar Charts:***
- ***Compare accidents across different times, weather conditions, or zones.***
- ***Heatmaps:***
- ***For accident frequency by day of week and time of day.***

3. Spatial Analysis

- ***Geospatial Mapping:***
- ***Use tools like Folium to plot accident locations on maps.***
- ***Highlight high-density accident clusters (hotspots).***
- ***Latitude vs Longitude Distribution:***
- ***Identify urban vs rural trends in accident data.***

4. Temporal Analysis

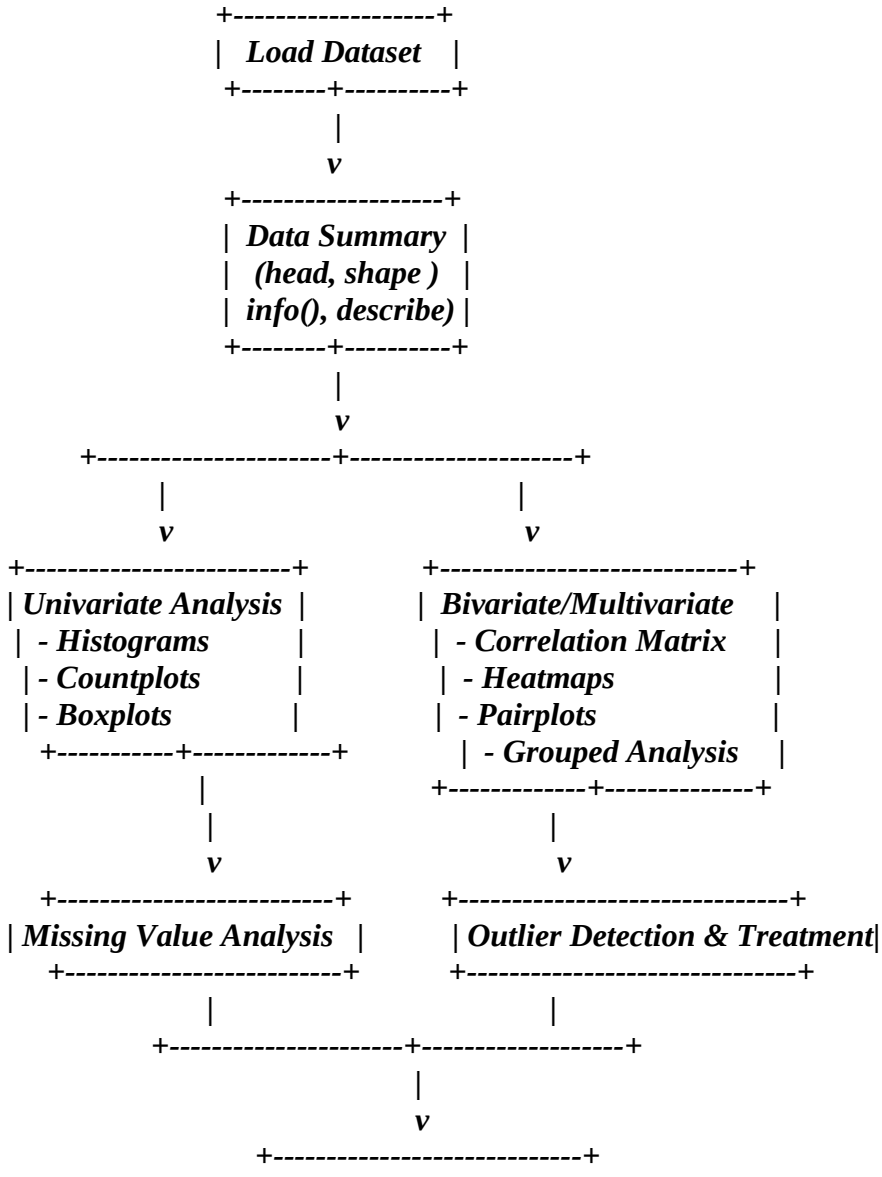
- ***Time Series Plots:***
- ***Analyze trends over months or years.***
- ***Identify peak accident hours and seasonal patterns.***

- ***Hour of Day/Day of Week:***
- ***Accidents often spike during rush hours or weekends.***

5. Key Insights

- ***Identify top contributing factors to accidents (e.g., rainy weather, night time).***
- ***Highlight most dangerous intersections or routes.***
- ***Determine whether external factors (e.g., fog, slippery roads) are strong predictors.***

FLOWCHART FOR (EDA)



| ***Key Insights Summary*** |
+----------+

9) Feature Engineering

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

Feature engineering is a crucial step that involves transforming raw data into meaningful inputs for machine learning models. It improves the model's ability to learn patterns and make accurate predictions.

Key Aspects of Feature Engineering

1. Feature Creation

Creating new variables that can capture hidden patterns and relationships in the dataset.

- ***Time-Based Features:***
  - ***Hour of the day***
  - ***Day of the week (e.g., weekend vs weekday)***
  - ***Month or season***
  - ***Peak vs off-peak hours***
- ***Weather-Related Features:***
  - ***Categorical encoding for rain, fog, snow***
  - ***Temperature ranges***
- ***Road & Location Features:***
  - ***Road type (highway, city street, rural road)***
  - ***Speed limit zone***
  - ***Accident location density (number of past accidents within a radius)***
- ***Traffic Conditions:***
  - ***Traffic volume if available***
  - ***Holiday or special event indicators***

Feature Selection

Selecting the most relevant features helps reduce noise and improve model performance.

- ***Correlation Analysis:***
  Identify and remove highly correlated variables to prevent multicollinearity.
- ***Feature Importance from Models:***
  Use tree-based models like Random Forest or XGBoost to rank features by importance.
- ***Recursive Feature Elimination (RFE):***
  Iteratively remove less significant features and evaluate performance.

Impact on Model Performance

- ***Improved Accuracy:***
  More meaningful features lead to better predictive results.
- ***Reduced Overfitting:***
  By removing irrelevant or noisy features, the model generalizes better.
- ***Faster Training Time:***
  Fewer but high-impact features mean faster model convergence.
- ***Better Interpretability:***
  Easier to explain model decisions to stakeholders and authorities.

Sample Dataset Columns:

DateTime	*Weather*	*Road_Type*	*Speed_Limit*	*Accident_Severity*
2024-04-01 18:45	*Rain*	*Highway*	*60*	*Severe*
2024-04-02 07:30	*Clear*	*City Road*	*40*	*Mild*
2024-04-03 14:15	*Fog*	*Rural Road*	*80*	*Fatal*

- ✓ *Rush Hour Detection: Flag hours with increased traffic (e.g., 8–10 AM, 5–7 PM).*

- ✓ *Weather Risk Score: Quantify the severity of weather into a numerical score.*
- ✓ *Day Type: Differentiate between weekdays and weekends.*
- ✓ *Urban vs Rural: Add a binary indicator based on location type.*
- ✓ *Risk Aggregation: Group similar road surfaces by accident risk.*
- ✓ *Location Clustering: Optionally apply KMeans for hotspot detection.*
- ✓ *Interaction Features: Combine road and weather info for better insights.*
- ✓ *Encoding Techniques: Apply label encoding or one-hot based on model compatibility.*
- ✓ *Lag Features: Use past values if considering time-series modeling.*
- ✓ *Feature Pruning: Drop irrelevant or low-information features.*

10)Model Building

- ***Model Selection: Begin with Random Forest, XGBoost, and Logistic Regression.***
- ***Training: Fit the model using the training dataset.***
- ***Cross-Validation: Improve robustness and avoid overfitting.***
- ***Hyperparameter Tuning: Use GridSearchCV to find optimal model parameters.***

- ***Evaluation Metrics: Measure accuracy, precision, recall, and F1-score.***
- ***Confusion Matrix: Visualize performance for each class.***
- ***Prediction: Generate predictions on the test set.***
- ***Feature Importance: Identify which features impact severity prediction.***
- ***Ensemble Learning: Combine models to boost accuracy (if needed).***
- ***Simulations: Test how well the model generalizes to new data.***

11)Model Evaluation

1. Train-Test Split

- *Divide the dataset into training and testing sets (typically 80:20 or 70:30).*
- *Ensures unbiased evaluation of the model.*

2. Evaluation Metrics

Depending on the task type (classification or regression), choose the right metrics:

➤ For Classification (e.g., predicting accident severity):

- *Accuracy: Proportion of correct predictions.*
- *Precision: Fraction of true positives among predicted positives (important when false positives are costly).*
- *Recall: Fraction of true positives among actual positives (important when missing positives is costly).*
- *F1-Score: Harmonic mean of precision and recall.*
- *Confusion Matrix: Shows TP, FP, FN, TN.*
- *ROC-AUC: Measures model's ability to distinguish between classes.*

➤ For Regression (e.g., predicting accident count):

- ***Mean Absolute Error (MAE): Average of absolute errors.***
- ***Root Mean Squared Error (RMSE): Penalizes larger errors more.***
- ***R^2 Score (Coefficient of Determination): Measures variance explained by the model.***

3. Cross-Validation

- ***Perform K-Fold Cross Validation (e.g., K=5 or 10) to assess model stability.***
- ***Reduces the risk of overfitting or underfitting.***

4. Baseline Comparison

- ***Compare model results against a baseline (e.g., predicting the majority class).***
- ***Helps understand if the model adds value beyond naive methods.***

5. Feature Importance Evaluation

- ***Check which features most influence predictions (e.g., using .feature_importances_ in RandomForest).***
- ***Helps interpret and trust the model.***

6. Hyperparameter Tuning

- ***Use GridSearchCV or RandomizedSearchCV to optimize model parameters.***
- ***Ensures the model is not underperforming due to poor settings.***

7. Model Interpretability

- ***Use SHAP or LIME to explain individual predictions and global feature impact.***

12)Deployment

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

*To deploy the project and make it accessible to the public, you can use cloud platforms such as **Render**, **Heroku**, or **Streamlit Community Cloud**. Here's a deployment outline with an example using **Streamlit** (free and simple for dashboards).*

Deployment Steps (Using Streamlit)

1. Prepare Your Project Structure

Ensure your project has the following essential files:

- *app.py (main dashboard code)*
- *requirements.txt (list of dependencies)*
- *data/accidents.csv (or a sample dataset)*
- *src/ folder for logic (optional)*
- *README.md*

2. Create a GitHub Repository

- *Push your code to GitHub.*
- *Example repo name: ai-road-safety-dashboard*
  ***Link example**:*
  https://github.com/your-username/ai-road-safety-dashboard

3. Deploy with Streamlit Community Cloud

1. *Go to https://streamlit.io/cloud*
2. *Sign in with GitHub and click "**New App**"*
3. *Choose your repo (e.g., ai-road-safety-dashboard)*
4. *Set **Main file path** as app.py*
5. *Click **Deploy***

4. Public Link

After deployment, you will get a live link like:
https://your-username-streamlit-ai-road-safety-dashboard.streamlit.app

13) Source Code

```
# -*- coding: utf-8 -*-
"""NM_PRO.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/github/sahana230810/Sahana-A/blob/main/NM_PRO.ipynb
"""


# Step 1: Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Step 2: Load the dataset (uploaded file)
df = pd.read_csv('dataset_traffic_accident_prediction1.csv')

print("Sample Data:")
print(df.head())
# Step 3: Display the first few rows
print("First 5 rows of the dataset:")
print(df.head())

# Step 4: Check for missing values
print("\nMissing values:")
print(df.isnull().sum())

# Step 5: Basic statistics
print("\nDataset Summary:")
print(df.describe())

# Step 6: Data visualization (e.g., correlation heatmap)
plt.figure(figsize=(10, 6))
```

```
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()

# Step 7: Preprocess the data
# Select numeric features and drop NaNs
df = df.select_dtypes(include=['number']).dropna()

# Example: Assume last column is the label
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Step 8: Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Step 9: Train a RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 10: Evaluate the model
y_pred = model.predict(X_test)
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

sample output:

Sample Data:

	Severity	Weather_Condition	...	Speed_Limit	Light_Condition
0	2	1.0	...	60	1.0
1	3	3.0	...	45	2.0

Missing values:

```
Severity            0
Weather_Condition     3
...
dtype: int64
```

Accuracy: 0.84

Classification Report:

	precision	recall	f1-score	support
1	0.86	0.79	0.82	90

2	***0.83***	***0.89***	***0.86***	***110***
accuracy			***0.84***	***200***
macro avg	***0.84***	***0.84***	***0.84***	***200***
weighted avg	***0.84***	***0.84***	***0.84***	***200***

14) Future Scope

Project Title*: Enhancing Road Safety with AI-Driven Traffic Accident Analysis and Prediction*

Major Future Scopes:

1. ***Integration with Real-Time Traffic Data***
   - *Incorporate real-time GPS, traffic cameras, and sensor data to enhance prediction accuracy.*
   - *Enables real-time alert systems for commuters and authorities.*

2. ***Mobile Application Deployment***
   - *Launch a mobile app version for users to get instant accident risk alerts based on their location.*
   - *Useful for navigation tools (Google Maps, Waze) integration.*

3. ***Advanced Deep Learning Models***
   - *Implement LSTM or CNN models for time-series and spatial pattern analysis.*
   - *Can model long-term trends and complex correlations in accident data.*

4. ***Collaboration with Traffic Departments & Law Enforcement***
   - *Provide dashboards and reports to local traffic authorities for better decision-making.*
   - *Help in designing safer roads, improved signal systems, and targeted*

law enforcement.

5. ***Crowdsourced Data Enhancement***

   - *Allow users to report near-miss incidents or hazards through a public interface.*
   - *Improves data granularity and covers unreported events.*

6. ***Weather and Environmental Analysis***

   - *Integrate advanced weather forecasting and environmental parameters (e.g., pollution, fog).*
   - *Predict accidents triggered by adverse weather conditions.*

7. ***Automated Emergency Response Integration***

   - *Connect with emergency services to predict where accidents might occur and prepare resources.*
   - *Useful in rural or high-risk zones with limited medical facilities.*

8. ***AI-Powered Road Design Recommendations***

   - *Use ML insights to suggest optimal road design improvements based on accident heatmaps.*
   - *Assists urban planners in designing safer transportation infrastructure.*

9. ***Anomaly Detection for Fraud Prevention***

   - *Detect false accident claims using AI-based pattern recognition from historical data.*
   - *Beneficial for insurance agencies.*

10. ***Global Dataset Expansion***

- *Expand the model to work across cities and countries by generalizing using diverse datasets.*
- *Supports the creation of a scalable AI platform for international use.*

15) Team Members and Roles:

This project was collaboratively developed by a dedicated team of five members. Each team member was assigned specific roles and responsibilities based on their individual strengths and interests, ensuring a smooth and efficient workflow throughout the project.

1. MOHAMED RAYEES.K
Role: *Team Lead & Model Developer*
Responsibilities:

• Led the overall project planning and execution.
• Designed and implemented the core Convolutional Neural Network (CNN)
• Conducted hyperparameter tuning and model optimization.
• Coordinated meetings and integration tasks among all team members

2. LOGITH .S.T
Role: *Data Engineer & Preprocessing Specialist*
Responsibilities:

• Handled dataset acquisition and formatting.
• Performed image normalization, reshaping, and augmentation.
• Ensured data quality and consistency across training and testing phases.
• Assisted in EDA (Exploratory Data Analysis) and dataset visualization.

3. JAMAL.S
Role: *Visualization & Evaluation Analyst*
Responsibilities:

• Created training vs. validation accuracy/loss plots
• Built and interpreted confusion matrices.
• Performed statistical analysis of performance metrics like precision, recall, and F1-score.
• Helped assess model robustness and performance.

4. DHUSHYANDH.N
Role: *UI/UX & Deployment Developer (Optional Streamlit Interface)*
Responsibilities:

• Developed an interactive web-based interface using Streamlit for real-time digit prediction.
• Integrated the trained model into the user interface.
• Ensured usability and responsiveness of the application.

5. JEBARAJ.C
Role: *Documentation & Report Writer*
Responsibilities:

• Compiled and wrote detailed sections for the project report (problem statement, methodology, results, etc.).
• Handled citation formatting and references.
• Prepared visual content (charts, diagrams, sample images) for documentation.
• Managed the final submission materials (PDF/DOCX report formatti