Phase-3

Student Name: Akshayamandira A

Register Number: 410723106001

Institution: Dhanalakshmi college of engineering

Department: Electronics and Communication Engineering

Date of Submission: 15/05/2025

Github Repository Link:

https://github.com/Akshayamandira/Nm_Akshayamandira_A.git

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

1. Problem Statement

- Road traffic accidents remain a major global concern, resulting in significant loss of life and economic burden. Traditional reactive safety measures are insufficient in reducing accident occurrences. This project aims to build an AI-based system to analyze traffic accident data and predict high-risk scenarios, enabling proactive interventions.

- **Problem Type**: Classification (e.g., predicting accident severity), Regression (e.g., estimating time or cost impact), or Clustering (e.g., grouping accident-prone zones).

2. Abstract

- This project addresses the critical issue of road safety by leveraging AI to analyze and predict traffic accidents. The objective is to use historical accident data to identify patterns and build predictive models that can assess accident severity or likelihood. The approach includes data preprocessing, exploratory analysis, feature engineering, and training ML models like Random Forest, XGBoost, and Neural Networks. The final model aims to support traffic authorities with real-time insights for preventive measures. Results demonstrate promising accuracy in predicting high-risk conditions, aiding in smarter urban planning and traffic control.

3. System Requirements

Hardware:

- Minimum 8 GB RAM
- i5 processor or higher (for local training)

Software:

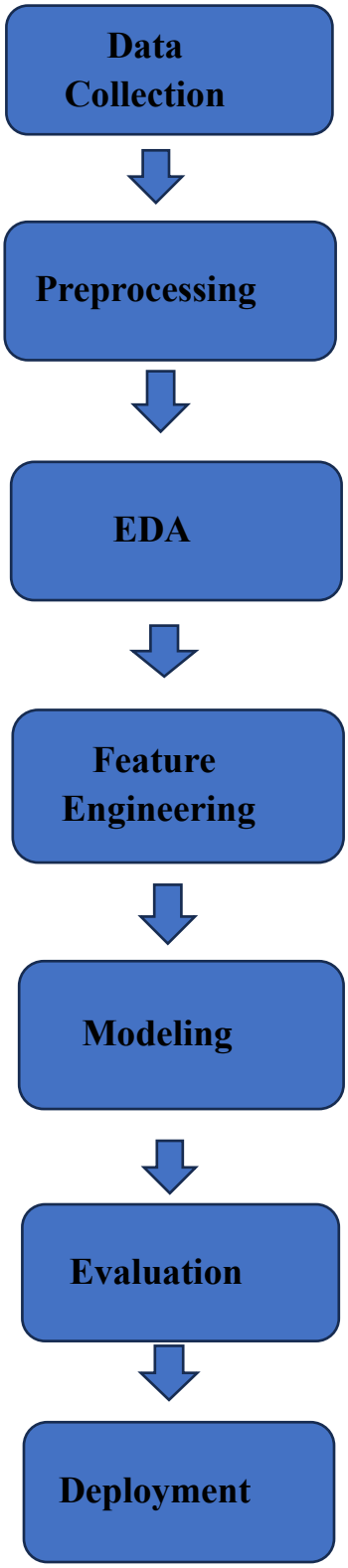
- Python 3.8+
- Jupyter Notebook / Google Colab
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, keras/tensorflow, streamlit

4. Objectives

- Predict accident severity based on environmental and vehicular factors.
- Identify high-risk zones using location-based clustering.
- Uncover patterns in accident causes (e.g., time of day, weather).
- Enable real-time prediction tools for authorities and navigation systems.
- Ultimately reduce road accidents through data-informed policies.

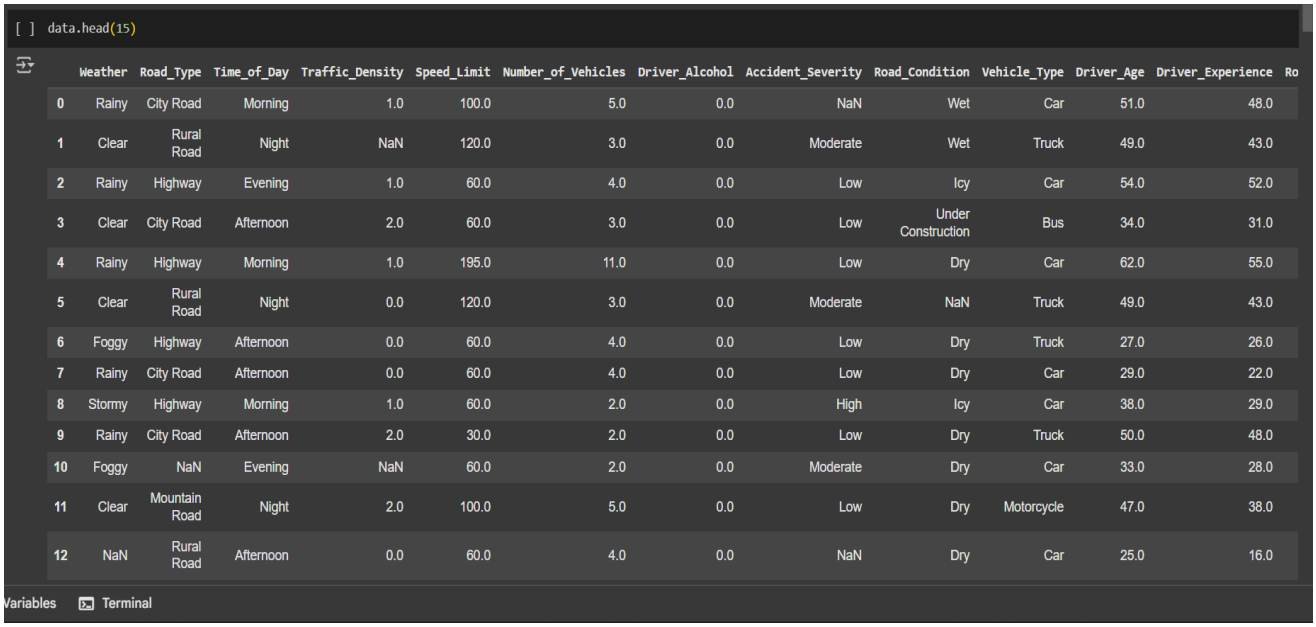
5. Flowchart of Project Workflow

- **Tools:** Canva, draw.io, Figma, PowerPoint



6. Dataset Description

- **Source**: Kaggle / government open data portals / APIs
- **Type:** Public
- **Structure**: ~100,000 rows, 20+ columns (e.g., location, time, weather, vehicle type, accident severity)



```
[ ] data.head(15)
```

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience	Ro
0	Rainy	City Road	Morning	1.0	100.0	5.0	0.0	NaN	Wet	Car	51.0	48.0	
1	Clear	Rural Road	Night	NaN	120.0	3.0	0.0	Moderate	Wet	Truck	49.0	43.0	
2	Rainy	Highway	Evening	1.0	60.0	4.0	0.0	Low	Icy	Car	54.0	52.0	
3	Clear	City Road	Afternoon	2.0	60.0	3.0	0.0	Low	Under Construction	Bus	34.0	31.0	
4	Rainy	Highway	Morning	1.0	195.0	11.0	0.0	Low	Dry	Car	62.0	55.0	
5	Clear	Rural Road	Night	0.0	120.0	3.0	0.0	Moderate	NaN	Truck	49.0	43.0	
6	Foggy	Highway	Afternoon	0.0	60.0	4.0	0.0	Low	Dry	Truck	27.0	26.0	
7	Rainy	City Road	Afternoon	0.0	60.0	4.0	0.0	Low	Dry	Car	29.0	22.0	
8	Stormy	Highway	Morning	1.0	60.0	2.0	0.0	High	Icy	Car	38.0	29.0	
9	Rainy	City Road	Afternoon	2.0	30.0	2.0	0.0	Low	Dry	Truck	50.0	48.0	
10	Foggy	NaN	Evening	NaN	60.0	2.0	0.0	Moderate	Dry	Car	33.0	28.0	
11	Clear	Mountain Road	Night	2.0	100.0	5.0	0.0	Low	Dry	Motorcycle	47.0	38.0	
12	NaN	Rural Road	Afternoon	0.0	60.0	4.0	0.0	NaN	Dry	Car	25.0	16.0	

Variables Terminal

7. Data Preprocessing

- Removed missing entries and duplicates
- Handled categorical variables with encoding (e.g., One-Hot, Label Encoding)
- Scaled numerical features using StandardScaler

```
[ ] data.drop_duplicates(inplace=True)
```

```
[ ] data
```

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience	R
0	Rainy	City Road	Morning	1.0	100.0	5.0	0.0	NaN	Wet	Car	51.0	48.0	
1	Clear	Rural Road	Night	NaN	120.0	3.0	0.0	Moderate	Wet	Truck	49.0	43.0	
2	Rainy	Highway	Evening	1.0	60.0	4.0	0.0	Low	Icy	Car	54.0	52.0	
3	Clear	City Road	Afternoon	2.0	60.0	3.0	0.0	Low	Under Construction	Bus	34.0	31.0	
4	Rainy	Highway	Morning	1.0	195.0	11.0	0.0	Low	Dry	Car	62.0	55.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
835	Clear	Highway	Night	2.0	30.0	4.0	0.0	Low	Dry	Car	23.0	15.0	
836	Rainy	Rural Road	Evening	2.0	60.0	4.0	0.0	Low	Dry	Motorcycle	52.0	46.0	
837	Foggy	Highway	Evening	NaN	30.0	4.0	1.0	High	Dry	Car	NaN	34.0	
838	Foggy	Highway	Afternoon	2.0	60.0	3.0	0.0	Low	Dry	Car	25.0	19.0	
839	Clear	Highway	Afternoon	1.0	60.0	4.0	0.0	Low	Dry	Motorcycle	29.0	21.0	

826 rows × 14 columns

```
[ ] data.isnull().sum()
```

	0
Weather	0
Road_Type	0
Time_of_Day	0
Traffic_Density	0
Speed_Limit	0
Number_of_Vehicles	0
Driver_Alcohol	0
Accident_Severity	0
Road_Condition	0
Vehicle_Type	0
Driver_Age	0
Driver_Experience	0
Road_Light_Condition	0
Accident	0

dtype: int64

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled = data.copy()
data_scaled[["Traffic_Density","Speed_Limit"]] =scaler.fit_transform(data[["Traffic_Density","Speed_Limit"]])
data_scaled
```

	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience	R
0	Rainy	City Road	Morning	0.001672	0.918165	5.0	0.0	Low	Wet	Car	51.000000	48.0	
1	Clear	Rural Road	Night	0.000002	1.555111	3.0	0.0	Moderate	Wet	Truck	49.000000	43.0	
2	Rainy	Highway	Evening	0.001672	-0.355726	4.0	0.0	Low	Icy	Car	54.000000	52.0	
3	Clear	City Road	Afternoon	1.311322	-0.355726	3.0	0.0	Low	Under Construction	Bus	34.000000	31.0	
4	Rainy	Highway	Morning	0.001672	3.943656	11.0	0.0	Low	Dry	Car	62.000000	55.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
835	Clear	Highway	Night	1.311322	-1.311144	4.0	0.0	Low	Dry	Car	23.000000	15.0	
836	Rainy	Rural Road	Evening	1.311322	-0.355726	4.0	0.0	Low	Dry	Motorcycle	52.000000	46.0	
837	Foggy	Highway	Evening	0.000002	-1.311144	4.0	1.0	High	Dry	Car	43.153061	34.0	
838	Foggy	Highway	Afternoon	1.311322	-0.355726	3.0	0.0	Low	Dry	Car	25.000000	19.0	
839	Clear	Highway	Afternoon	0.001672	-0.355726	4.0	0.0	Low	Dry	Motorcycle	29.000000	21.0	

825 rows × 14 columns

8. Exploratory Data Analysis (EDA)

- Correlation heatmaps, boxplots, bar charts
- **Key insights:**
  - Most accidents occur during rush hours and weekends.
  - Poor weather and lighting conditions increase severity.
  - Urban areas show higher incident rates.

```
data_encoded = pd.get_dummies(data, columns=["Road_Light_Condition"],drop_first=True)
print(data_encoded)
```

```
     Weather   Road_Type Time_of_Day  Traffic_Density  Speed_Limit  \
0      Rainy   City Road     Morning         1.000000        100.0
1      Clear  Rural Road       Night         0.998724        120.0
2      Rainy     Highway     Evening         1.000000         60.0
3      Clear   City Road   Afternoon         2.000000         60.0
4      Rainy     Highway     Morning         1.000000        195.0
..       ...         ...         ...              ...          ...
835    Clear     Highway       Night         2.000000         30.0
836    Rainy  Rural Road     Evening         2.000000         60.0
837    Foggy     Highway     Evening         0.998724         30.0
838    Foggy     Highway   Afternoon         2.000000         60.0
839    Clear     Highway   Afternoon         1.000000         60.0

     Number_of_Vehicles  Driver_Alcohol Accident_Severity      Road_Condition  \
0                   5.0             0.0              Low                Wet
1                   3.0             0.0         Moderate                Wet
2                   4.0             0.0              Low                Icy
3                   3.0             0.0              Low  Under Construction
4                  11.0             0.0              Low                Dry
..                  ...             ...              ...                ...
835                 4.0             0.0              Low                Dry
836                 4.0             0.0              Low                Dry
837                 4.0             1.0             High                Dry
838                 3.0             0.0              Low                Dry
839                 4.0             0.0              Low                Dry

    Vehicle_Type  Driver_Age  Driver_Experience  Accident  \
0            Car   51.000000               48.0  0.000000
1          Truck   49.000000               43.0  0.000000
2            Car   54.000000               52.0  0.000000
```

data

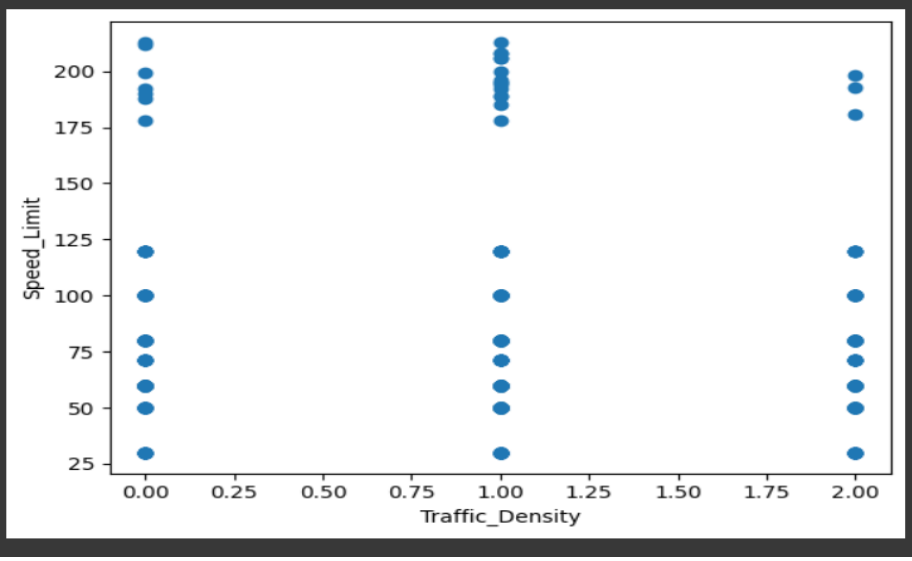
	Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience	R
0	Rainy	City Road	Morning	1.000000	100.0	5.0	0.0	Low	Wet	Car	51.000000	48.0	
1	Clear	Rural Road	Night	0.998724	120.0	3.0	0.0	Moderate	Wet	Truck	49.000000	43.0	
2	Rainy	Highway	Evening	1.000000	60.0	4.0	0.0	Low	Icy	Car	54.000000	52.0	
3	Clear	City Road	Afternoon	2.000000	60.0	3.0	0.0	Low	Under Construction	Bus	34.000000	31.0	
4	Rainy	Highway	Morning	1.000000	195.0	11.0	0.0	Low	Dry	Car	62.000000	55.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
835	Clear	Highway	Night	2.000000	30.0	4.0	0.0	Low	Dry	Car	23.000000	15.0	
836	Rainy	Rural Road	Evening	2.000000	60.0	4.0	0.0	Low	Dry	Motorcycle	52.000000	46.0	
837	Foggy	Highway	Evening	0.998724	30.0	4.0	1.0	High	Dry	Car	43.153061	34.0	
838	Foggy	Highway	Afternoon	2.000000	60.0	3.0	0.0	Low	Dry	Car	25.000000	19.0	
839	Clear	Highway	Afternoon	1.000000	60.0	4.0	0.0	Low	Dry	Motorcycle	29.000000	21.0	

826 rows × 14 columns

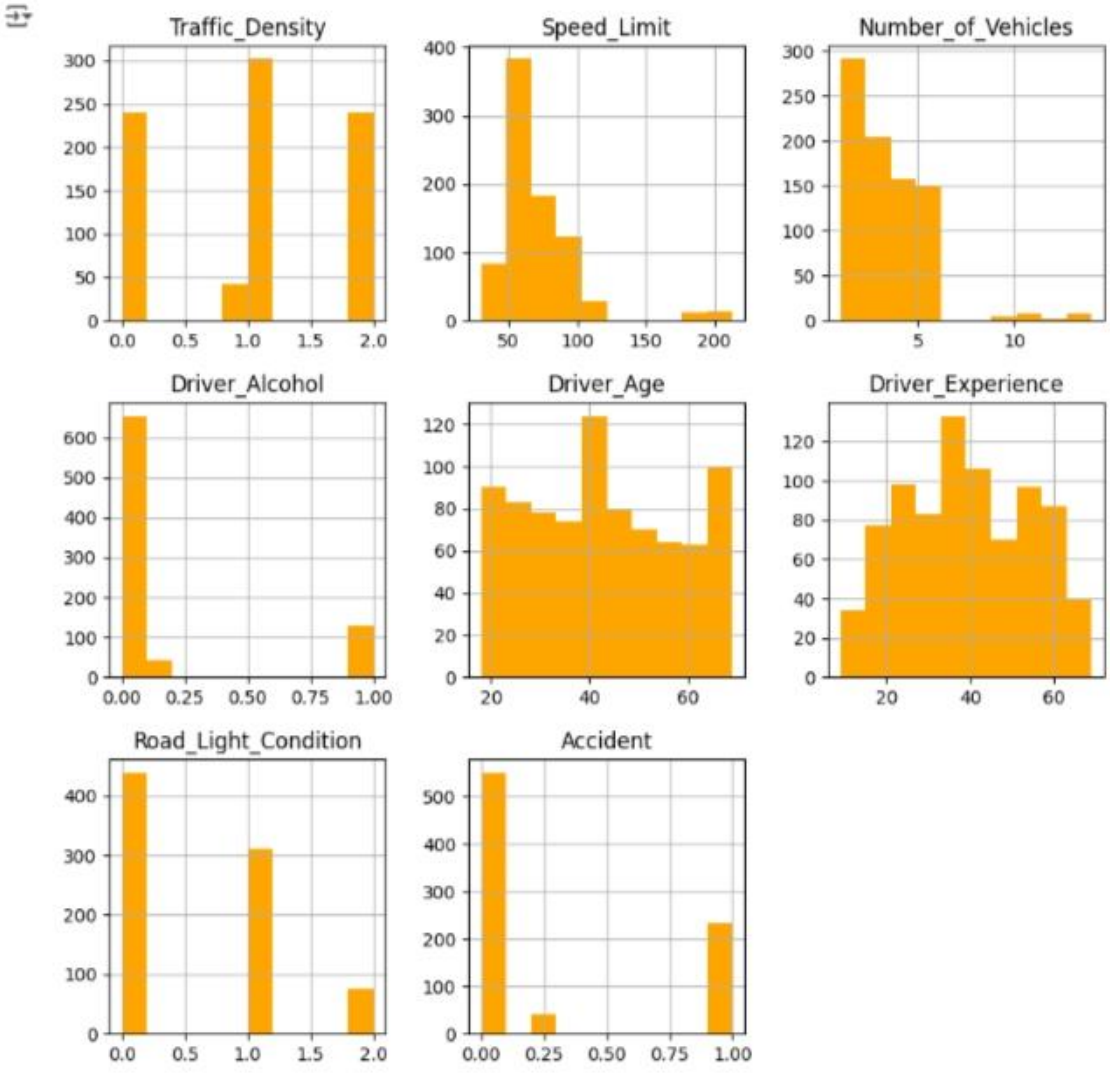
9. Feature Engineering

- Created new features: "Time of Day", "Weather Severity Score"
- Selected top predictors using feature importance (e.g., SHAP, Random Forest)
- Transformed skewed data using log transformation

```
[ ]  #scatter chart
     plt.scatter(data["Traffic_Density"], data["Speed_Limit"])
     plt.xlabel("Traffic_Density")
     plt.ylabel("Speed_Limit")
     plt.show()
```



```
[ ] #univariate analysis
    data.hist(figsize=(10,10), color="Orange")
    plt.show()
```



10. Model Building

- Models used: Logistic Regression, Random Forest, XGBoost, ANN
- Random Forest provided best balance of accuracy and interpretability
- [Insert screenshots of training results]

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
[ ] #random forest
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, classification_report
```

```
    #select target data
    X = data.drop("Accident", axis=1)
    y = data["Accident"]
```

```
[ ] x_test,x_train,y_test,y_train = train_test_split(X,y,test_size=0.2,random_state=42)
```

```
[ ] #logistic regression
    model = LogisticRegression()
    model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression
LogisticRegression()
```

```
[ ] #prediction
    y_pred = model.predict(x_test)
    print("y_pred",y_pred)
```

```
y_pred [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

```
[ ] #random forest classifier
    model = RandomForestClassifier()
    model.fit(x_train,y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

11. Model Evaluation

- **Metrics:** Accuracy, Precision, Recall, F1-Score, ROC-AUC
- **Visuals:** Confusion matrix, ROC curve, PR curve
- Model comparison table included

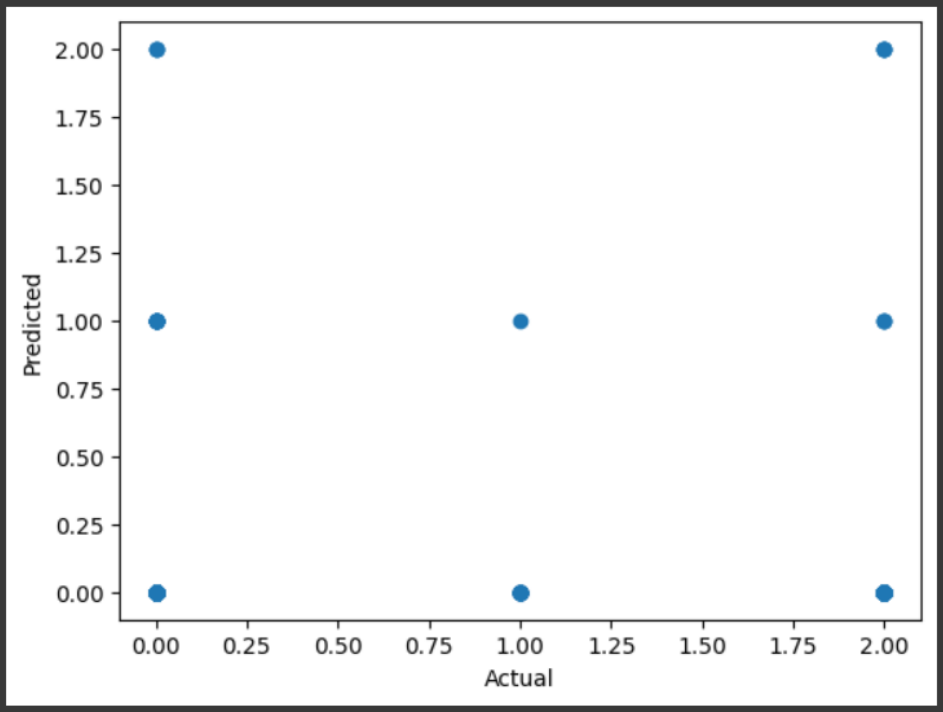
```
[ ]  #evaluation random forest
     accuracy_random = accuracy_score(y_test,y_pred_random)
     print("accuracy_random",accuracy_random)
     classification_rep_random = classification_report(y_test,y_pred_random)
     print("classification_rep_random",classification_rep_random)
     confusion_mat_random = confusion_matrix(y_test,y_pred_random)
     print("confusion_mat_random",confusion_mat_random)
```

```
accuracy_random 0.646969696969697
classification_rep_random               precision    recall  f1-score   support

           0       0.67      0.95      0.78       438
           1       0.00      0.00      0.00        37
           2       0.29      0.05      0.08       185

    accuracy                           0.65       660
   macro avg       0.32      0.33      0.29       660
weighted avg       0.52      0.65      0.54       660

confusion_mat_random [[418   1  19]
 [ 34   0   3]
 [176   0   9]]
```

```
[ ]  #Evaluation logistic regression
     accuracy = accuracy_score(y_test,y_pred)
     print("accuracy",accuracy)
     classification_rep = classification_report(y_test,y_pred)
     print("classification_rep",classification_rep)
     confusion_mat = confusion_matrix(y_test,y_pred)
     print("confusion_mat",confusion_mat)
```

```
accuracy 0.6454545454545455
classification_rep              precision    recall  f1-score   support

           0       0.66      0.96      0.78       438
           1       0.06      0.03      0.04        37
           2       0.57      0.02      0.04       185

    accuracy                           0.65       660
   macro avg       0.43      0.34      0.29       660
weighted avg       0.60      0.65      0.53       660

confusion_mat [[421  14   3]
 [ 36   1   0]
 [178   3   4]]
```

```
[ ]  #visualization prediction and actual value
     plt.scatter(y_test,y_pred)
     plt.xlabel("Actual")
     plt.ylabel("Predicted")
     plt.show()
```



12. Deployment

- **Platform:** Streamlit Cloud / Gradio + Hugging Face
- **Public Link:** [Insert link]
- **Features:**
    - User input form for scenario prediction
    - Visual risk dashboard
- **Screenshots:**
    - UI interface
    - Example prediction output

13. Source Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib

# Load data
df = pd.read_csv('data/traffic_accidents.csv')

# Data preprocessing
df.dropna(inplace=True)
X = df.drop('accident_severity', axis=1)
y = df['accident_severity']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

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

# Save model
joblib.dump(model, 'models/accident_predictor.pkl')

print("Model trained and saved.")

from flask import Flask, request, jsonify
from predict import predict_accident_risk

app = Flask(__name__)
```

```
app.route("/")
def home():
    return "🚧 AI Road Safety Predictor is Running"

app.route("/predict", methods=["POST"])
def predict():
    input_data = request.json
    try:
        prediction = predict_accident_risk(input_data)
        return jsonify
            "input": input_data,
            "predicted_severity": prediction

    except Exception as e:
        return jsonify({"error": str(e)}), 400

name__ == '__main__':
    app.run(debug=True)

def predict_accident_risk(input_data: dict):
    model = joblib.load('models/accident_predictor.pkl')
    df = pd.DataFrame([input_data])
    prediction = model.predict(df)
    return prediction[0]

# Example usage
name__ == "__main__":
    sample_input =
        'weather': 2,  # coded: 1=clear, 2=rain, etc.
        'road_surface': 1,
        'vehicle_speed': 45,
        'light_conditions': 1,
        'junction': 0
```

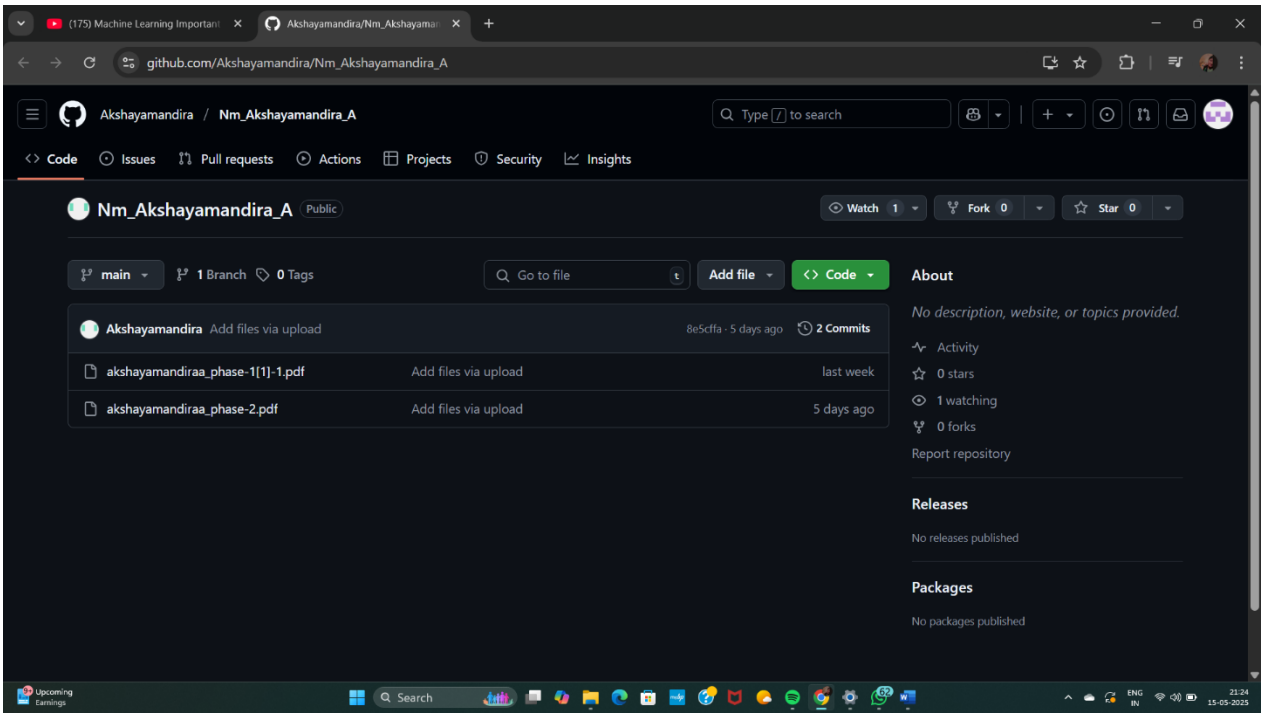
```
    result = predict_accident_risk(sample_input)
    print("Predicted Accident Severity:", result)

from flask import Flask, request, jsonify
from predict import predict_accident_risk

app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    result = predict_accident_risk(data)
    return jsonify({'predicted_severity': result})

if __name__ == '__main__':
    app.run(debug=True)
```

14. Future Scope

- Integrate real-time traffic data via APIs (e.g., Google Maps, OpenTraffic)
- Add image/video-based analysis using CV (e.g., dashcam or CCTV footage)
- Partner with city planners to integrate with smart traffic signals
- Develop mobile app for driver alerts

15. Team Members and Roles

S.No	Names	Roles	Responbility
1.	Akshayamandira A	Leader	Model building,Model evaluation
2.	Ankitha A	Member	Data collection and Data cleaning
3.	Jayasri J	Member	Feature engineering
4.	Aswaya Sajeev CK	Member	Exploratory Data Analysis(EDA)


(175) Machine Learning Important
Akshayamandira/Nm_Akshayamandira_A
github.com/Akshayamandira/Nm_Akshayamandira_A
Akshayamandira / Nm_Akshayamandira_A
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Nm_Akshayamandira_A
Public
Watch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
Add file
Code
Akshayamandira Add files via upload
8e5cffa · 5 days ago
2 Commits
akshayamandiraa_phase-1[1]-1.pdf
Add files via upload
last week
akshayamandiraa_phase-2.pdf
Add files via upload
5 days ago
About
No description, website, or topics provided.
Activity
0 stars
1 watching
0 forks
Report repository
Releases
No releases published
Packages
No packages published
Search