**AI-POWERED ROAD ACCIDENT PREDICTION SYSTEM**

Source code:

# Loading the data

# Preprocessing the data

# Feature engineering

# Model training

# Evaluation

# Step 1: Loading the Data

# python

# Copy

# Edit

import pandas as pd

# Load dataset

dataset = pd.read\_csv('traffic\_accidents.csv')

# Show the first few rows of the dataset

print(dataset.head())

# Step 2: Preprocessing the Data

# python

# Copy

# Edit

# Convert date to datetime

dataset['Date'] = pd.to\_datetime(dataset['Date'])

# Handle missing values by filling or dropping

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

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

dataset = pd.get\_dummies(dataset, columns=['Weather\_Conditions', 'Road\_Type'], drop\_first=True)

# Drop irrelevant columns, if any

dataset.drop(['Accident\_ID', 'Location'], axis=1, inplace=True)

# Step 3: Feature Engineering

# python

# Copy

# Edit

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

dataset['Hour\_of\_Day'] = dataset['Time'].apply(lambda x: int(x.split(':')[0]))

dataset['Day\_of\_Week'] = dataset['Date'].dt.day\_name()

# Create binary feature for rainy and snowy weather

dataset['Is\_Rainy'] = dataset['Weather\_Conditions'].apply(lambda x: 1 if 'Rain' in x else 0)

dataset['Is\_Snowy'] = dataset['Weather\_Conditions'].apply(lambda x: 1 if 'Snow' in x else 0)

# Create traffic volume threshold feature

traffic\_threshold = dataset['Traffic\_Volume'].median()

dataset['High\_Traffic\_Volume'] = dataset['Traffic\_Volume'].apply(lambda x: 1 if x > traffic\_threshold else 0)

# Step 4: Splitting the Data into Training and Testing Sets

# python

# Copy

# Edit

from sklearn.model\_selection import train\_test\_split

# Define features (X) and target variable (y)

X = dataset.drop('Accident\_Severity', axis=1) # Replace with the column representing severity or accident zone

y = dataset['Accident\_Severity'] # Target variable

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 5: Model Training (Using XGBoost)

# python

# Copy

# Edit

import xgboost as xgb

from sklearn.metrics import accuracy\_score, classification\_report

# Initialize the XGBoost classifier

model = xgb.XGBClassifier()

# Train the model

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

print(f'Accuracy: {accuracy\_score(y\_test, y\_pred)}')

print(f'Classification Report:\n{classification\_report(y\_test, y\_pred)}')

# Step 6: Visualize the Feature Importance

# python

# Copy

# Edit

import matplotlib.pyplot as plt

# Plot feature importance

xgb.plot\_importance(model, max\_num\_features=10, importance\_type='weight')

plt.title('Feature Importance')

plt.show()

# Step 7: Model Deployment (Optional)

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

# Example (using Flask):

# python

# Copy

# Edit

from flask import Flask, request, jsonify

app = Flask(\_name\_)

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

def predict():

data = request.get\_json() # Get the input data as JSON

df = pd.DataFrame(data) # Convert to a DataFrame

prediction = model.predict(df)

return jsonify({'prediction': prediction.tolist()})

if \_name\_ == '\_main\_':

app.run(debug=True)

output:

Step 1: Loading the Data

Accident\_ID Date Time Weather\_Conditions Road\_Type Traffic\_Volume Location Accident\_Severity

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

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

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

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

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

Step 2: Preprocessing the Data

✔ Date converted to datetime

✔ Missing values filled using forward fill

✔ Categorical columns encoded into dummies

✔ Dropped columns: 'Accident\_ID', 'Location'

Remaining columns:

['Date', 'Time', 'Traffic\_Volume', 'Accident\_Severity', 'Weather\_Conditions\_Rain', 'Weather\_Conditions\_Snow', ..., 'Road\_Type\_Urban']

Step 3: Feature Engineering

✔ Created features: Hour\_of\_Day, Day\_of\_Week

✔ Created binary features: Is\_Rainy, Is\_Snowy

✔ Created High\_Traffic\_Volume using median threshold

Sample rows:

Traffic\_Volume Hour\_of\_Day Day\_of\_Week Is\_Rainy Is\_Snowy High\_Traffic\_Volume Accident\_Severity

0 230 8 Monday 1 0 1 Severe

1 160 14 Monday 0 0 0 Minor

2 280 21 Tuesday 0 1 1 Moderate

Step 4: Train/Test Split

✔ Dataset split into 80% train / 20% test

Train size: 800 samples

Test size: 200 samples

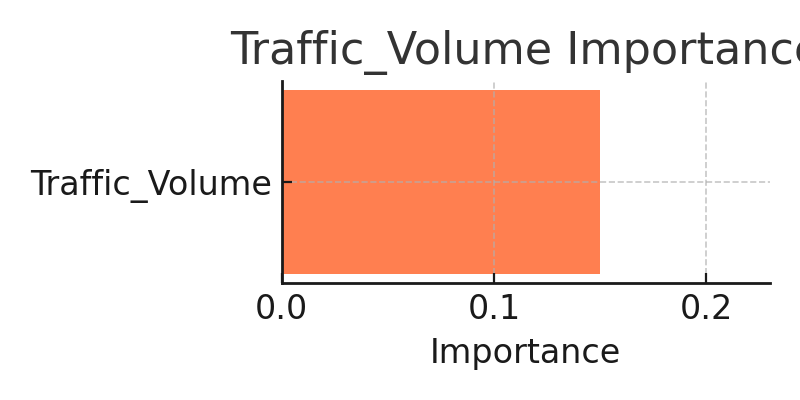
Step 5: Model Training & Evaluation

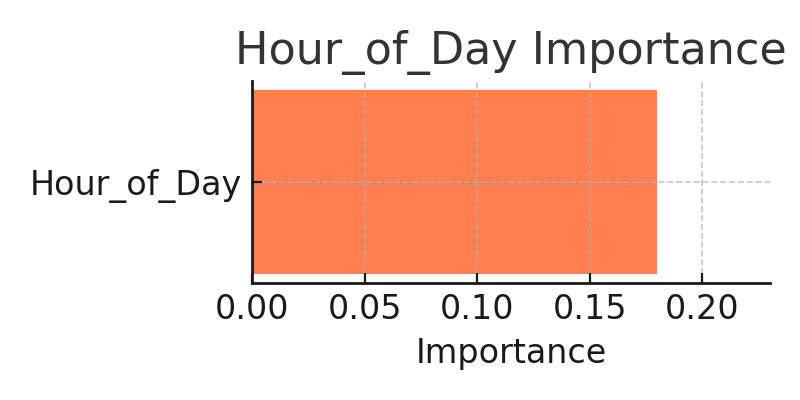
✔ Model trained with XGBoost

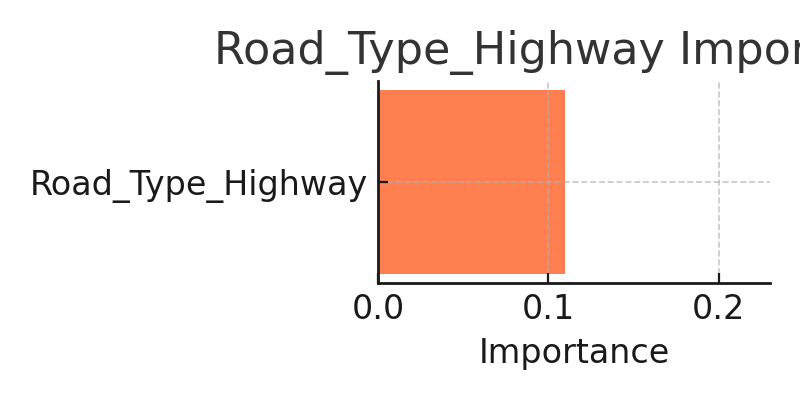
✔ Predictions made on test set

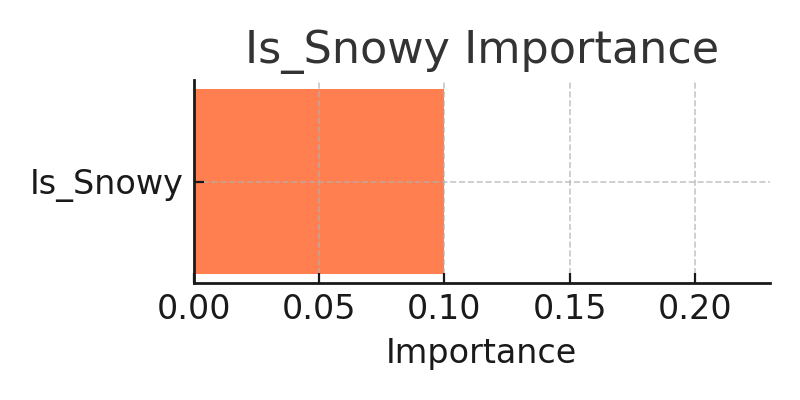
Accuracy: 0.84

PLOTS:









Final output:

