import pandas as pd

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

from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.linear\_model import LogisticRegression

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

import matplotlib.pyplot as plt

import seaborn as sns

# Assume you have a CSV file named 'accident\_data.csv'

# with columns like 'location', 'time', 'weather', 'road\_condition', 'severity', etc.

data = pd.read\_csv('accident\_data.csv')

# --- 1. Data Loading and Preprocessing ---

# Handle missing values (example: fill numerical with mean, categorical with mode)

for col in data.select\_dtypes(include=['number']).columns:

data[col].fillna(data[col].mean(), inplace=True)

for col in data.select\_dtypes(include=['object']).columns:

data[col].fillna(data[col].mode()[0], inplace=True)

# Define features (X) and target variable (y) - assuming 'severity' is the target for classification

X = data.drop('severity', axis=1)

y = data['severity']

# Identify categorical and numerical features

categorical\_features = X.select\_dtypes(include=['object']).columns

numerical\_features = X.select\_dtypes(include=['number']).columns

# Create preprocessing pipelines for numerical and categorical features

numerical\_transformer = StandardScaler()

categorical\_transformer = OneHotEncoder(handle\_unknown='ignore')

# Combine transformers using ColumnTransformer

preprocessor = ColumnTransformer(

transformers=[

('num', numerical\_transformer, numerical\_features),

('cat', categorical\_transformer, categorical\_features)])

# Split data 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)

# --- 2. Model Selection and 3. Model Training ---

# Example: Logistic Regression model

model = Pipeline(steps=[('preprocessor', preprocessor),

('classifier', LogisticRegression(solver='liblinear', random\_state=42))])

# Train the model

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

# --- 4. Model Evaluation ---

# Make predictions on the test set

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

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Predicted Severity')

plt.ylabel('Actual Severity')

plt.title('Confusion Matrix')

plt.show()

# --- 5. Prediction (Example) ---

# Create a sample new data point (must have the same columns as the training data)

new\_data = pd.DataFrame({

'location': ['Intersection A'],

'time': ['18:00'],

'weather': ['Rainy'],

'road\_condition': ['Wet'],

'speed\_limit': [50],

'traffic\_volume': [150]

})

# Make a prediction

predicted\_severity = model.predict(new\_data)

print(f"\nPredicted Severity for the new data point: {predicted\_severity}")