```
# Import required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# 1. Data Collection
# (Assume you have a CSV file with crash, weather, road data, etc.)
data = pd.read_csv('your_dataset.csv')

# 2. Data Preprocessing
# Handle missing values
data.fillna(method='ffill', inplace=True)

# Encode categorical variables
label_encoders = {}
for col in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# 3. Exploratory Data Analysis (EDA)
print(data.describe())
```

```
sns.heatmap(data.corr(), annot=True)

plt.title("Feature Correlation")

plt.show()


# 4. Feature Engineering

# Example: Time Binning

data['hour_bin'] = pd.cut(data['hour'], bins=[0,6,12,18,24], labels=["Night", "Morning", "Afternoon", "Evening"])

data['hour_bin'] = LabelEncoder().fit_transform(data['hour_bin'])


# Distance or holiday columns assumed to exist

# You can add your custom logic here


# 5. Model Selection and Training

# Define features and target

X = data.drop(columns=['target_column'])

y = data['target_column']


# Train/test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Standardize features

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Train a model (Random Forest as example)

model = RandomForestClassifier()
```

```
model.fit(X_train, y_train)


# 6. Model Evaluation
y_pred = model.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted'))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
```