This dataset includes:

Time: The time at which the accident occurred.

Day_of_week: The day of the week when the accident happened.

Age_band_of_driver: The age range of the driver involved in the accident.

Sex_of_driver: The gender of the driver.

Educational_level: The educational attainment of the driver.

Vehicle_driver_relation: The relationship of the driver to the vehicle (e.g., owner, employee).

Driving_experience: The number of years the driver has been driving.

Type_of_vehicle: The type of vehicle involved in the accident.

Owner_of_vehicle: Indicates if the driver is the owner of the vehicle.

Service_year_of_vehicle: The number of years the vehicle has been in service.

Vehicle_movement: The movement of the vehicle at the time of the accident.

Casualty_class: The class of casualty (e.g., pedestrian, passenger).

Sex_of_casualty: The gender of the casualty.

Age_band_of_casualty: The age range of the casualty.

Casualty_severity: The severity of the casualty's injuries.

Work_of_casuality: The occupation of the casualty.

Fitness_of_casuality: The fitness level of the casualty.

Pedestrian_movement: The movement of the pedestrian involved in the accident.

Cause_of_accident: The cause of the accident.

Accident_severity: The severity of the accident (e.g., slight injury, serious injury).

Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from folium.plugins import HeatMap


from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


df=pd.read_csv("/content/RTA Dataset.csv")


df.head()
```

...vement	Casualty_class	Sex_of_casualty	Age_band_of_casualty	Casualty_severity	Work_of_
...straight	na	na	na	na	
...straight	na	na	na	na	
...straight	Driver or rider	Male	31-50	3	
...straight	Pedestrian	Female	18-30	3	
...straight	na	na	na	na	

```
df.columns
```

```
Index(['Time', 'Day_of_week', 'Age_band_of_driver', 'Sex_of_driver',
       'Educational_level', 'Vehicle_driver_relation', 'Driving_experience',
       'Type_of_vehicle', 'Owner_of_vehicle', 'Service_year_of_vehicle',
       'Defect_of_vehicle', 'Area_accident_occured', 'Lanes_or_Medians',
       'Road_allignment', 'Types_of_Junction', 'Road_surface_type',
       'Road_surface_conditions', 'Light_conditions', 'Weather_conditions',
       'Type_of_collision', 'Number_of_vehicles_involved',
       'Number_of_casualties', 'Vehicle_movement', 'Casualty_class',
       'Sex_of_casualty', 'Age_band_of_casualty', 'Casualty_severity',
       'Work_of_casuality', 'Fitness_of_casuality', 'Pedestrian_movement',
       'Cause_of_accident', 'Accident_severity'],
      dtype='object')
```

```
df.describe()
```

	Number_of_vehicles_involved	Number_of_casualties
count	12316.000000	12316.000000
mean	2.040679	1.548149
std	0.688790	1.007179
min	1.000000	1.000000
25%	2.000000	1.000000
50%	2.000000	1.000000
75%	2.000000	2.000000
max	7.000000	8.000000

```
df.shape
```

```
(12316, 32)
```

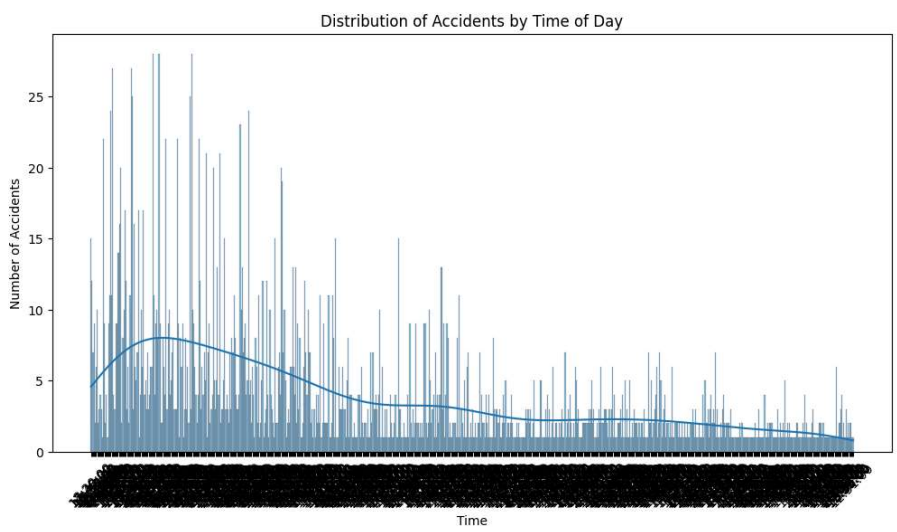
Handle missing values

```
# Check for missing values
missing_values = df.isnull().sum()
```

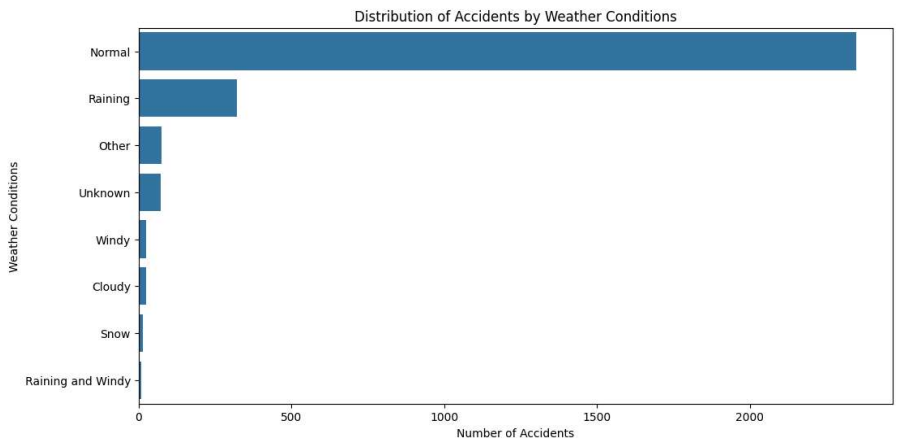
```
df = df.dropna()
df.head(2)
```

ement	Casualty_class	Sex_of_casualty	Age_band_of_casualty	Casualty_severity	Work_of_
straight	Pedestrian	Male	Under 18	3	
U-Turn	Passenger	Male	18-30	3	

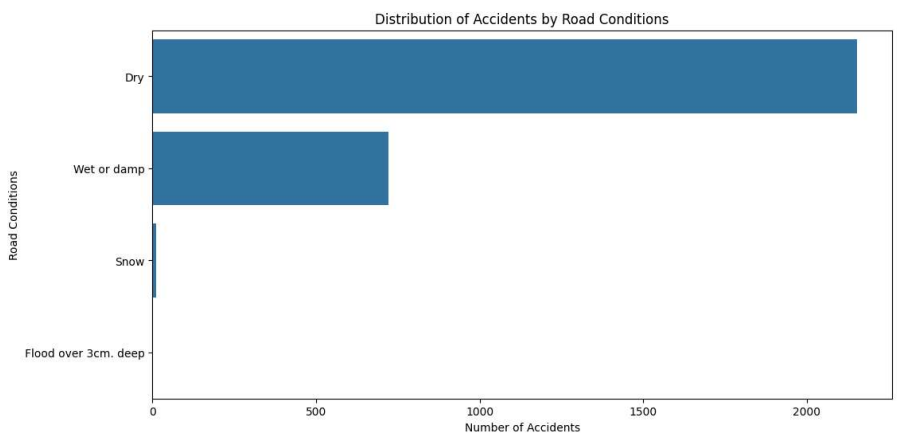
```
plt.figure(figsize=(12, 6))
sns.histplot(df['Time'], bins=24, kde=True)
plt.title('Distribution of Accidents by Time of Day')
plt.xlabel('Time')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(12, 6))
sns.countplot(y='Weather_conditions', data=df, order=df['Weather_conditions'].value_counts().index)
plt.title('Distribution of Accidents by Weather Conditions')
plt.xlabel('Number of Accidents')
plt.ylabel('Weather Conditions')
plt.show()
```

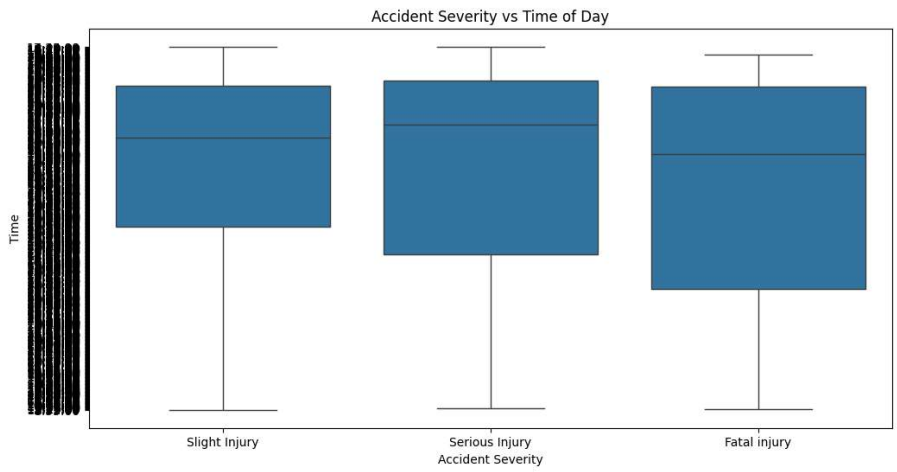



```
plt.figure(figsize=(12, 6))
sns.countplot(y='Road_surface_conditions', data=df, order=df['Road_surface_conditions'].value_counts().index)
plt.title('Distribution of Accidents by Road Conditions')
plt.xlabel('Number of Accidents')
plt.ylabel('Road Conditions')
plt.show()
```

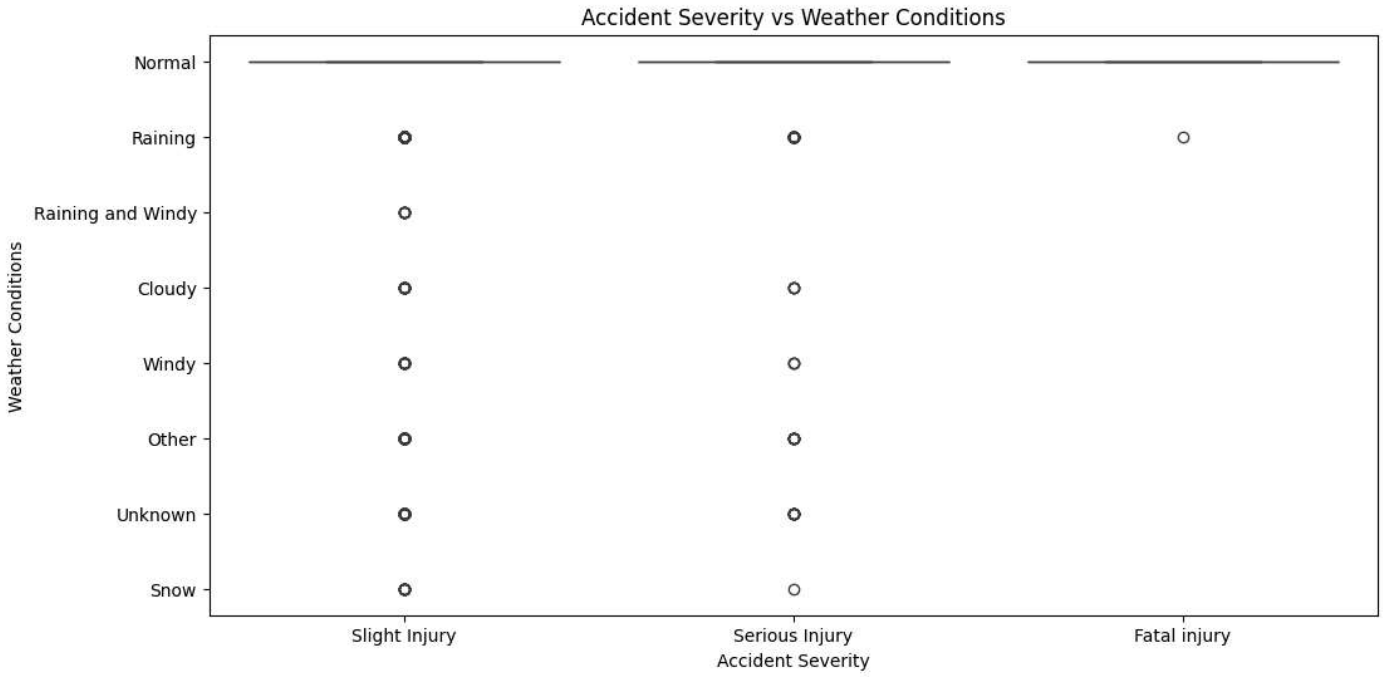


pattern identification

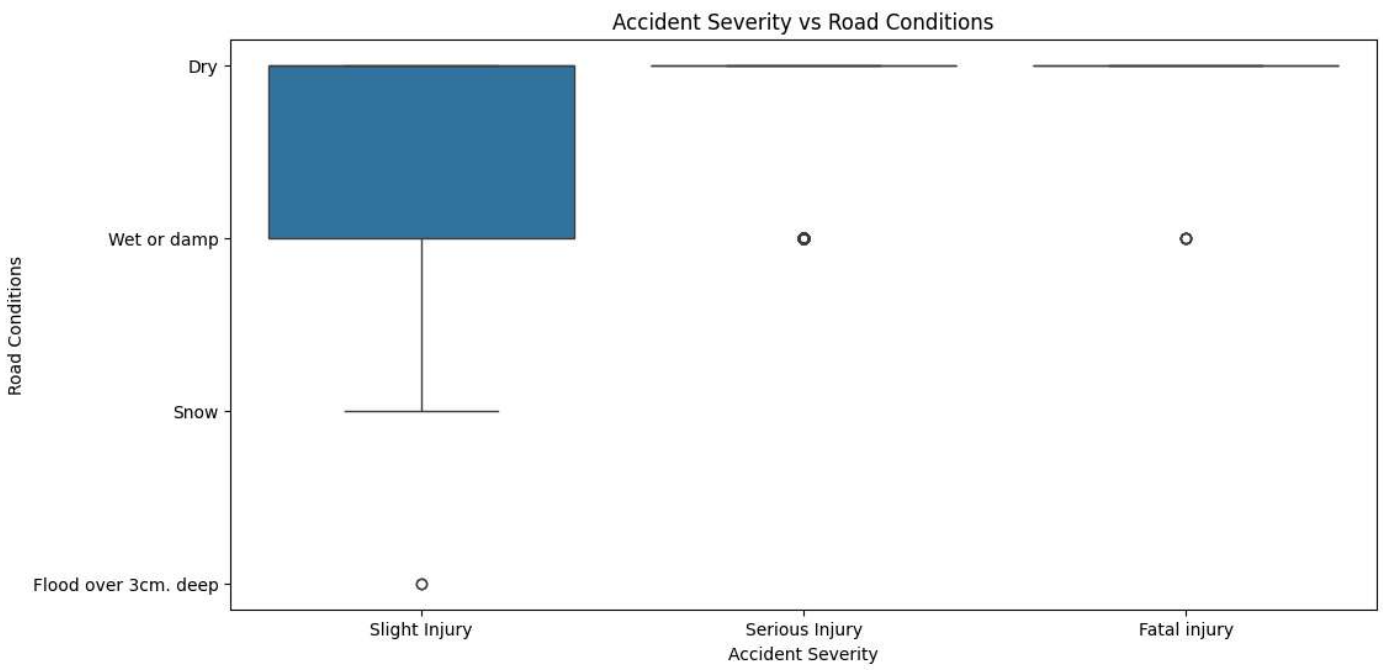
```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Accident_severity', y='Time', data=df)
plt.title('Accident Severity vs Time of Day')
plt.xlabel('Accident Severity')
plt.ylabel('Time')
plt.show()
```



```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Accident_severity', y='Weather_conditions', data=df)
plt.title('Accident Severity vs Weather Conditions')
plt.xlabel('Accident Severity')
plt.ylabel('Weather Conditions')
plt.show()
```

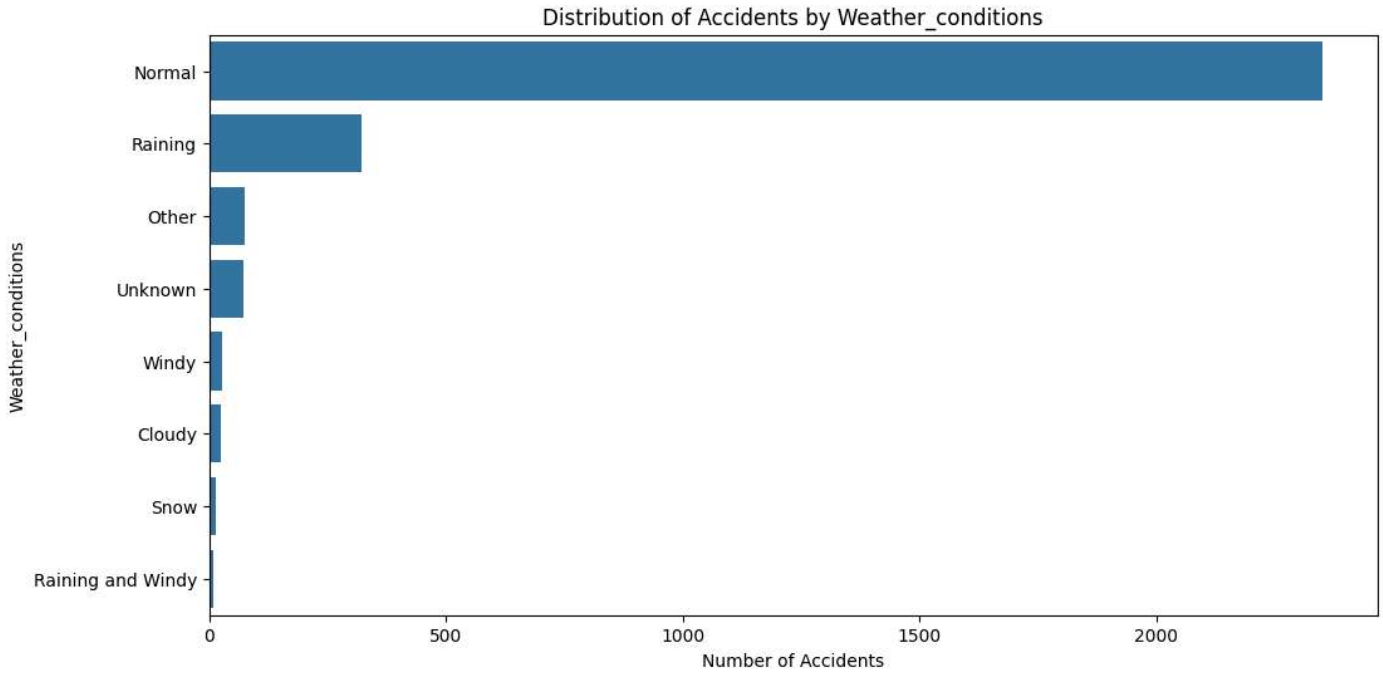


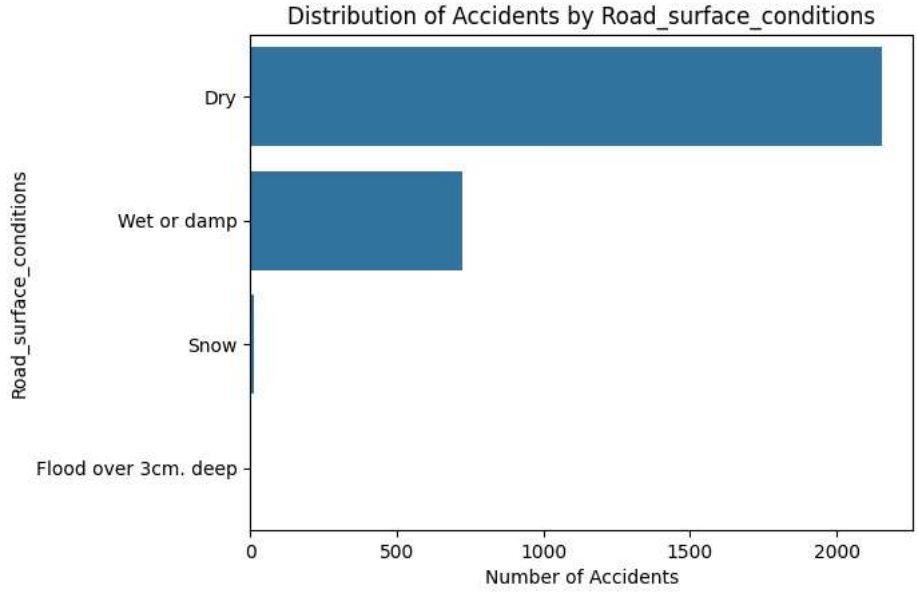

```
plt.figure(figsize=(12, 6))
sns.boxplot(x='Accident_severity', y='Road_surface_conditions', data=df)
plt.title('Accident Severity vs Road Conditions')
plt.xlabel('Accident Severity')
plt.ylabel('Road Conditions')
plt.show()
```

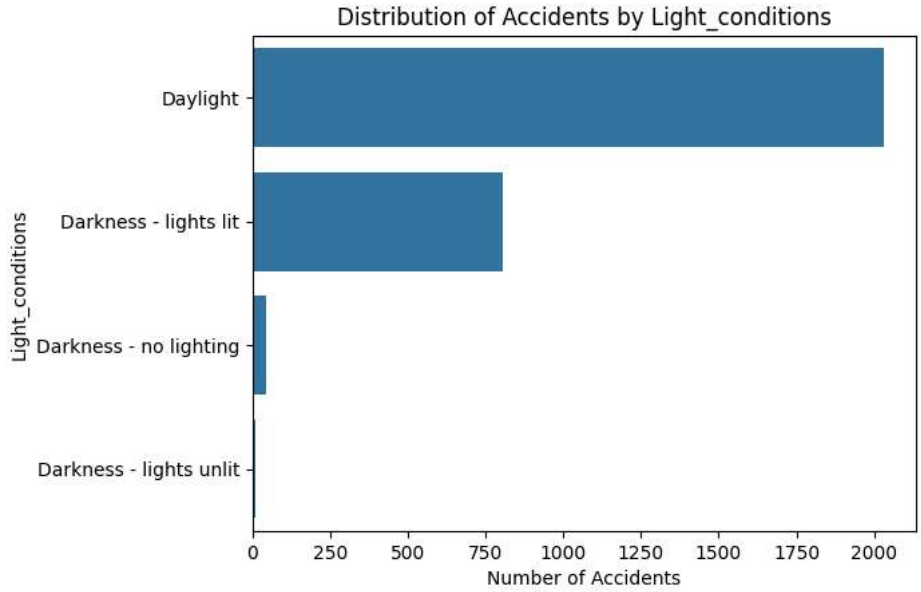



```
factors = ['Weather_conditions', 'Road_surface_conditions', 'Light_conditions', 'Urban_or_Rural_Area']
```

```
    plt.figure(figsize=(12, 6))
    for factor in factors:
        if factor in df.columns:
            sns.countplot(y=factor, data=df, order=df[factor].value_counts().index)
            plt.title(f'Distribution of Accidents by {factor}')
            plt.xlabel('Number of Accidents')
            plt.ylabel(factor)
            plt.show()
        else:
            print(f"Warning: Column '{factor}' not found in DataFrame.")
```








```
Warning: Column 'Urban_or_Rural_Area' not found in DataFrame.
```

Build model

```
X = df.drop('Accident_severity', axis=1)
y = df['Accident_severity']


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

Train a random forest model

```
from sklearn.preprocessing import OneHotEncoder
# Identify categorical columns
categorical_cols = X.select_dtypes(include=['object']).columns


# Apply one-hot encoding to categorical features
encoder = OneHotEncoder(handle_unknown='ignore')
X_encoded = pd.DataFrame(encoder.fit_transform(X[categorical_cols]).toarray())


X_numerical = X.drop(categorical_cols, axis=1)


X_encoded.columns = encoder.get_feature_names_out(categorical_cols)


X_numerical = X_numerical.reset_index(drop=True)
X_encoded = X_encoded.reset_index(drop=True)


X = pd.concat([X_numerical, X_encoded], axis=1)


rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Evaluate

```
y_pred = rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8737024221453287
```

```
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
                   precision    recall  f1-score   support

      Fatal injury       0.00      0.00      0.00         8
     Serious Injury       1.00      0.03      0.06        67
      Slight Injury       0.87      1.00      0.93       503

           accuracy                           0.87       578
          macro avg       0.62      0.34      0.33       578
       weighted avg       0.88      0.87      0.82       578

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```