```
from google.colab import files
file=files.upload()
```

Choose files No file chosen

```
import pandas as pd
df = pd.read_csv('RTA Dataset.csv')
df.head()
```

	Time	Day_of_week	Age_band_of_driver	Sex_of_driver	Educational_level	Vehicle_driver_relation	Driving_experience	Type_of_
0	17:02:00	Monday	18-30	Male	Above high school	Employee	1-2yr	Au
1	17:02:00	Monday	31-50	Male	Junior high school	Employee	Above 10yr	Pu
2	17:02:00	Monday	18-30	Male	Junior high school	Employee	1-2yr	Lorry (4
3	1:06:00	Sunday	18-30	Male	Junior high school	Employee	5-10yr	Pu
4	1:06:00	Sunday	18-30	Male	Junior high school	Employee	2-5yr	

5 rows × 32 columns

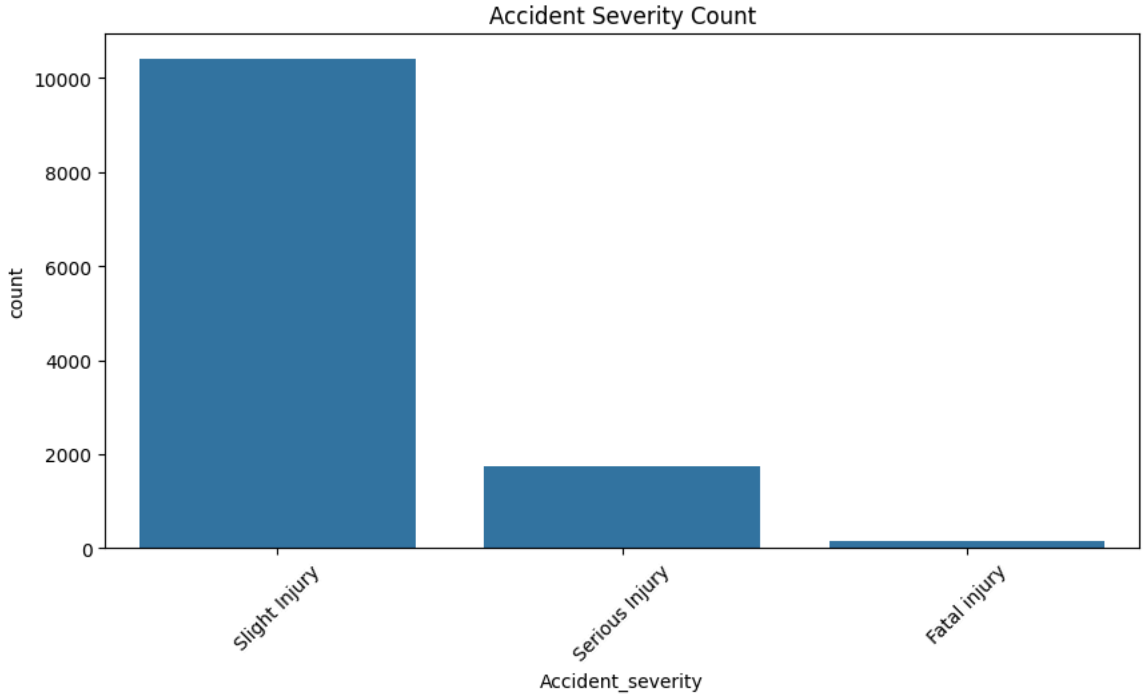
```
print("Shape:", df.shape)
print("Columns:", df.columns)
print(df.describe())
print(df.info())
```

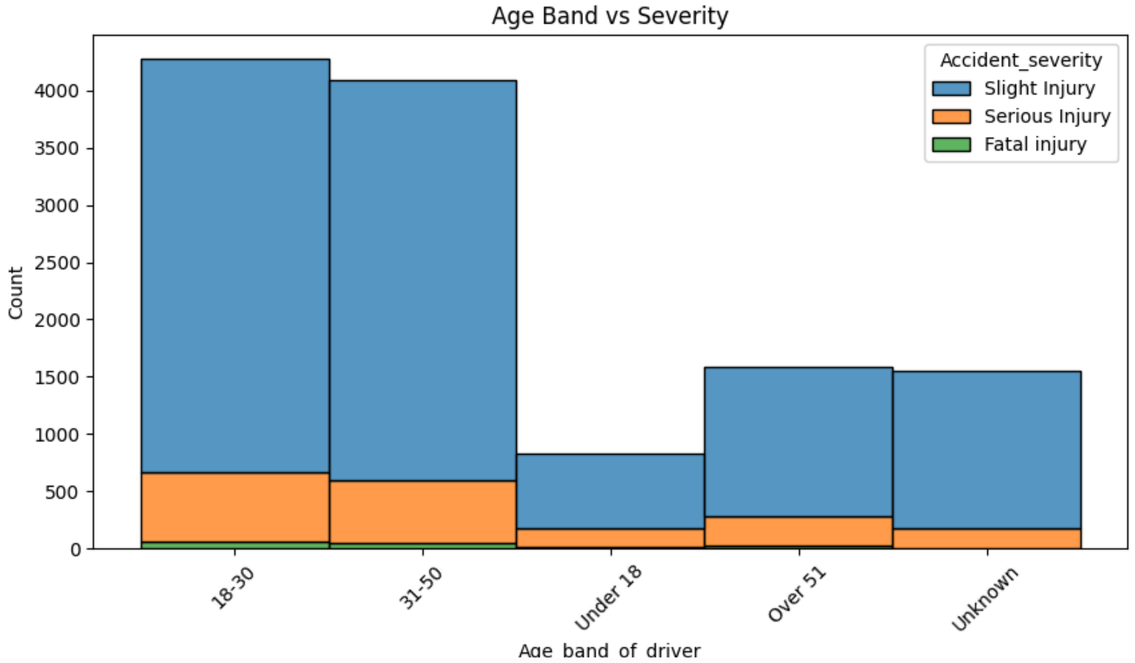
```
Shape: (12316, 32)
Columns: Index(['Time', 'Day_of_week', 'Age_band_of_driver', 'Sex_of_driver',
       'Educational_level', 'Vehicle_driver_relation', 'Driving_experience',
       'Type_of_vehicle', 'Owner_of_vehicle', 'Service_year_of_vehicle',
       'Defect_of_vehicle', 'Area_accident_occured', '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')
       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
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12316 entries, 0 to 12315
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype 
---  ------                       --------------  ----- 
 0   Time                         12316 non-null  object
 1   Day_of_week                  12316 non-null  object
 2   Age_band_of_driver           12316 non-null  object
 3   Sex_of_driver                12316 non-null  object
 4   Educational_level            11575 non-null  object
 5   Vehicle_driver_relation      11737 non-null  object
 6   Driving_experience           11487 non-null  object
 7   Type_of_vehicle              11366 non-null  object
 8   Owner_of_vehicle             11834 non-null  object
 9   Service_year_of_vehicle      8388 non-null   object
 10  Defect_of_vehicle            7889 non-null   object
 11  Area_accident_occured        12077 non-null  object
 12  Lanes_or_Medians             11931 non-null  object
 13  Road_allignment              12174 non-null  object
 14  Types_of_Junction            11429 non-null  object
 15  Road_surface_type            12144 non-null  object
 16  Road_surface_conditions      12316 non-null  object
 17  Light_conditions             12316 non-null  object
 18  Weather_conditions           12316 non-null  object
 19  Type_of_collision            12161 non-null  object
 20  Number_of_vehicles_involved  12316 non-null  int64 
 21  Number_of_casualties         12316 non-null  int64 
 22  Vehicle_movement             12008 non-null  object
 23  Casualty_class               12316 non-null  object
 24  Sex_of_casualty              12316 non-null  object
 25  Age_band_of_casualty         12316 non-null  object
 26  Casualty_severity            12316 non-null  object
 27  Work_of_casuality            9118 non-null   object
 28  Fitness_of_casuality         9681 non-null   object
 29  Pedestrian_movement          12316 non-null  object
 30  Cause_of_accident            12316 non-null  object
```

```
print("Missing values:\n", df.isnull().sum())
print("\nDuplicate entries:", df.duplicated().sum())
```

```
Missing values:
 Time                           0
Day_of_week                    0
Age_band_of_driver             0
Sex_of_driver                  0
Educational_level            741
Vehicle_driver_relation      579
Driving_experience           829
Type_of_vehicle              950
Owner_of_vehicle             482
Service_year_of_vehicle     3928
Defect_of_vehicle           4427
Area_accident_occured        239
Lanes_or_Medians             385
Road_allignment              142
Types_of_Junction            887
Road_surface_type            172
Road_surface_conditions        0
Light_conditions               0
Weather_conditions             0
Type_of_collision            155
Number_of_vehicles_involved    0
Number_of_casualties           0
Vehicle_movement             308
Casualty_class                 0
Sex_of_casualty                0
Age_band_of_casualty           0
Casualty_severity              0
Work_of_casuality           3198
Fitness_of_casuality        2635
Pedestrian_movement            0
Cause_of_accident              0
Accident_severity              0
dtype: int64

Duplicate entries: 0
```

```
# Step 5: Visualize a Few Features
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10,5))
sns.countplot(x='Accident_severity', data=df)
plt.title("Accident Severity Count")
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10,5))
sns.histplot(data=df, x='Age_band_of_driver', hue='Accident_severity', multiple='stack')
plt.title("Age Band vs Severity")
plt.xticks(rotation=45)
plt.show()
```





```
# Step 6: Identify Target and Features
target = 'Accident_severity'
features = df.drop(columns=[target])
labels = df[target]
```

```
# Step 7: Save categorical mappings for later use
original_df = features.copy()
cat_cols = features.select_dtypes(include='object').columns
cat_maps = {}

for col in cat_cols:
    features[col] = features[col].astype('category')
    cat_maps[col] = dict(enumerate(features[col].cat.categories))
    cat_maps[col] = {v: k for k, v in cat_maps[col].items()}  # reverse mapping

    features[col] = features[col].map(cat_maps[col])
```

```
# Step 8: One-Hot Encoding
features = pd.get_dummies(features, drop_first=True)
```

```
# Step 9: Feature Scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)



# Step 10-13: Final Feature & Label Assignment
X = features_scaled
y = labels


# Step 14: Train-Test Split
from sklearn.model_selection import train_test_split

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


# Step 15: Model Building
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier ⓘ ?
RandomForestClassifier()
```

```
# Step 16: Evaluation
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

y_pred = model.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[   2    0   35]
 [   0   19  344]
 [   0    2 2062]]

Classification Report:
                 precision    recall  f1-score   support

  Fatal injury       1.00      0.05      0.10        37
Serious Injury       0.90      0.05      0.10       363
 Slight Injury       0.84      1.00      0.92      2064

      accuracy                           0.85      2464
     macro avg       0.92      0.37      0.37      2464
  weighted avg       0.86      0.85      0.78      2464


Accuracy Score: 0.8453733766233766
```

```
# Step 17-19: Example Prediction
import numpy as np

example_input = np.array([X_test[0]])  # Test example
print("Example Prediction:", model.predict(example_input)[0])
```

```
Example Prediction: Slight Injury
```

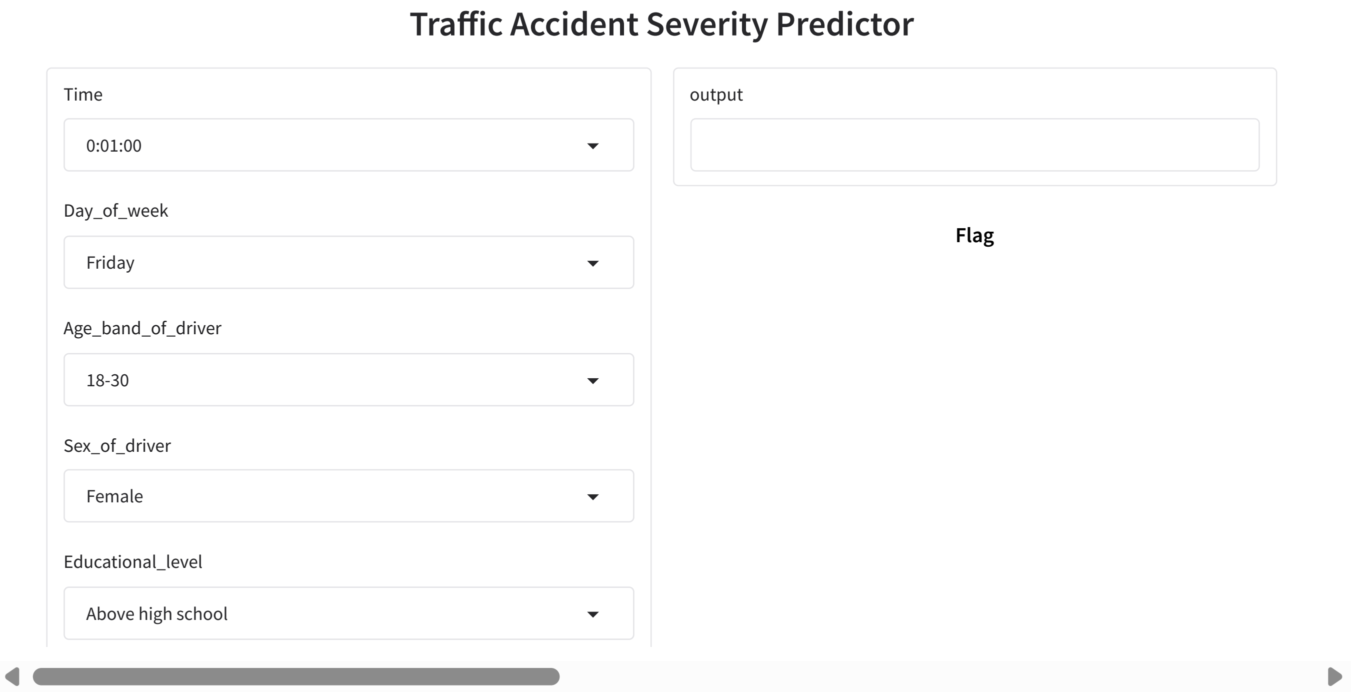
```
# Step 18: Convert to DataFrame and Encode (if taking new raw inputs)
new_df = pd.DataFrame([features.iloc[0]])  # Use new raw input here
new_df_scaled = scaler.transform(new_df)



# Step 19: Predict the Final Grade (Severity)
final_prediction = model.predict(new_df_scaled)
print("Final Predicted Severity:", final_prediction[0])
```

```
Final Predicted Severity: Slight Injury
```

```
# Step 20: Deployment - Install Gradio
!pip install gradio --quiet

# Step 21: Create a Prediction Function with Encoding
```

```
final_columns = features.columns  # store final column order

def predict_severity(*inputs):
    input_dict = {}
    i = 0
    for col in original_df.columns:
        if col in cat_maps:
            input_dict[col] = cat_maps[col].get(inputs[i], 0)
        else:
            input_dict[col] = float(inputs[i])
        i += 1

    # Convert to DataFrame
    input_df = pd.DataFrame([input_dict])
    input_df = pd.get_dummies(input_df)
    input_df = input_df.reindex(columns=final_columns, fill_value=0

    # Scale and predict
    input_scaled = scaler.transform(input_df)
    prediction = model.predict(input_scaled)
    return f"Predicted Severity: {prediction[0]}"

# Step 22: Create the Gradio Interface
import gradio as gr

input_fields = []
for col in original_df.columns:
    if col in cat_maps:
        choices = list(cat_maps[col].keys())
        input_fields.append(gr.Dropdown(choices=choices, label=col)
    else:
        input_fields.append(gr.Number(label=col))

gr.Interface(
    fn=predict_severity,
    inputs=input_fields,
    outputs="text",
    title="Traffic Accident Severity Predictor"
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://75d315de8ddc7850bd.gradio.liv

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```



Traffic Accident Severity Predictor

Time: 0:01:00

Day_of_week: Friday

Age_band_of_driver: 18-30

Sex_of_driver: Female

Educational_level: Above high school

output