

# **PROJECT**

Accident Severity Prediction

By Saad Mohiuddin Ajaz Mohammed Anwar Ahmed Nahri Syed Hyder Hussaini





## PROJECT OVERVIEW

#### **OBJECTIVE:**

To predict injury severity in traffic accidents using machine learning.

0000

0000

#### **KEY FOCUS AREAS:**

- Data Preprocessing
- Balancing Techniques
- Model Training and Evaluation
- Actionable Recommendations



# PROJECT BENEFITS

0000

### Benefits of Accident Severity Prediction

- Enhanced Traffic Safety: Improve preventive measures and emergency responses.
- Data-Driven Insights: Facilitate data-driven decision-making in traffic management.
- Model Effectiveness: Develop robust predictive models for injury severity.

## DATA OVERVIEW

### **Data Description**

- Dataset: The Motor Vehicle Collisions dataset contains information from all police-reported motor vehicle collisions in NYC. It spans from April 2016, when crash reporting transitioned to an electronic system. The dataset comprises nearly 2,30,000 records, detailing various aspects of each collision.
- **Key Variables:** PERSON\_INJURY, PERSON\_AGE, PERSON\_SEX, SAFETY\_EQUIPMENT, EJECTION, BODILY\_INJURY, and COMPLAINT.
- Challenges: Missing data and imbalanced data set which includes very less instances of minority class in target variable.







#### **Data Preprocessing Steps**

- Cleaning Steps:
- Handling missing values.
- Dropping irrelevant columns.
- Feature Engineering:
- Creation of new features (e.g., 'DATETIME' from 'CRASH\_DATE' and 'CRASH\_TIME').
- <sub>o</sub> Encoding:

0000

0000

0000

0000

0000

- One-hot encoding for categorical variables.
- Label encoding for the target variable.

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

# Display columns with missing values.
print(missing_values[missing_values > 0])
```

```
# Drop irrelevant columns

data_cleaned = data.drop(columns=['UNIQUE_ID', 'COLLISION_ID', 'PERSON_ID', 'VEH:

# Convert categorical variables using one-hot encoding
data_encoded = pd.get_dummies(data_cleaned, columns=['PERSON_TYPE', 'SAFETY_EQUIN

# Create a new 'DATETIME' feature from 'CRASH_DATE' and 'CRASH_TIME'
data_encoded['CRASH_DATETIME'] = pd.to_datetime(data_encoded['CRASH_DATE'] + ''

# Drop the original 'CRASH_DATE' and 'CRASH_TIME' columns
data_encoded = data_encoded.drop(columns=['CRASH_DATE', 'CRASH_TIME'])
```



0000

0000

0000

0000

0000

0000

```
from sklearn.model_selection import train_test_split
# Define features and target
X = data_encoded.drop(columns=['PERSON_INJURY'])
y = data_encoded['PERSON_INJURY']
                                                                                 from sklearn.ensemble import RandomForestClassifier
# Split the data
                                                                                 from sklearn.metrics import classification_report, confusion_matrix, roc_auc_scol
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
                                                                                 # Initialize and train the model
# Verify the split
                                                                                 model = RandomForestClassifier(n_estimators=100, random_state=42)
print(f'Training data: {X_train.shape}, {y_train.shape}')
                                                                                 model.fit(X train, y train)
print(f'Test data: {X_test.shape}, {y_test.shape}')
Training data: (184528, 189), (184528,)
                                                                                 # Predict class probabilities
Test data: (46132, 189), (46132,)
                                                                                 y_pred_proba = model.predict_proba(X_test)
                                                                                 # Evaluate the model
      0000
                                                                                 y_pred = model.predict(X_test)
      0000
                                                                                 print(classification_report(y_test, y_pred))
      0000
                                                                                 print(confusion_matrix(y_test, y_pred))
                                                                                 print(f'ROC AUC Score: {roc_auc_score(y_test, y_pred_proba, multi_class="ovr")}'
```

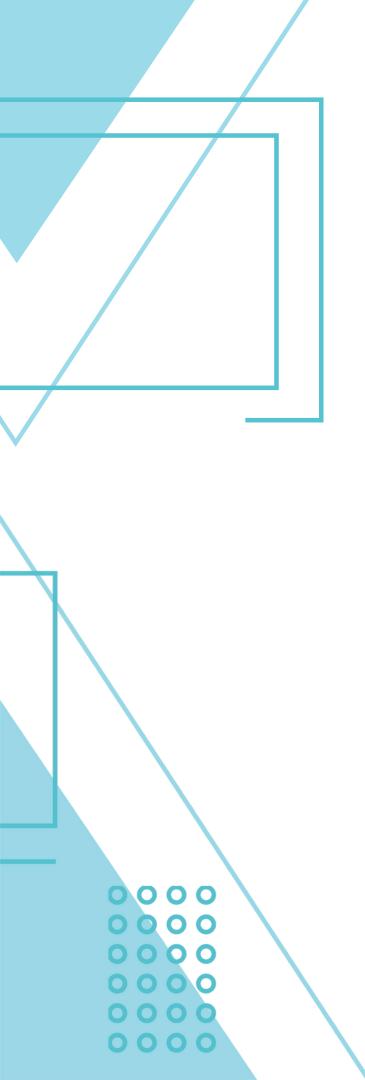
0000

0000



### Intermediate Results

	precision	recall	f1-score	support
0	1.00	1.00	1.00	45916
1	0.89	0.29	0.44	215
2	1.00	1.00	1.00	1
accuracy			1.00	46132
macro avg	0.96	0.76	0.81	46132
weighted avg	• 1.00	1.00	1.00	46132
[[45908 8	0]			
[ 153 62	0]			
[ 0 0	1]]			
ROC AUC Score	0.91464200	2602385		



# THANK YOU

Questions?

Email:saadajaz20@gmail.com

0000