Classifying Car Crashes Using Neural Networks

From Raw Data to Severity Prediction

Alazar Gebremehdin, Hannibal Mussie, Feruz Seid, Yassin Bedru, Samir Bahru

2026-01-05

Outline

Introduction 2
Data Understanding (EDA) 4
Data Preparation 8
Model Design 12
Training & Evaluation 16
Conclusion 21

Introduction

Objective & Overview

Goal: Predict the severity of road traffic accidents (**Fatal, Serious, Minor, PDO**) based on accident characteristics.

The Workflow:

1. **Data Understanding:** Handling massive missing data and inconsistencies.
2. **Preparation:** Cleaning, Imputation, and Feature Engineering.
3. **Modeling:** Designing a Multi-Layer Perceptron (MLP).
4. **Training:** Managing class imbalance and overfitting.
5. **Evaluation:** F1-Scores and Confusion Matrices.

Data Understanding (EDA)

Data Quality Challenges

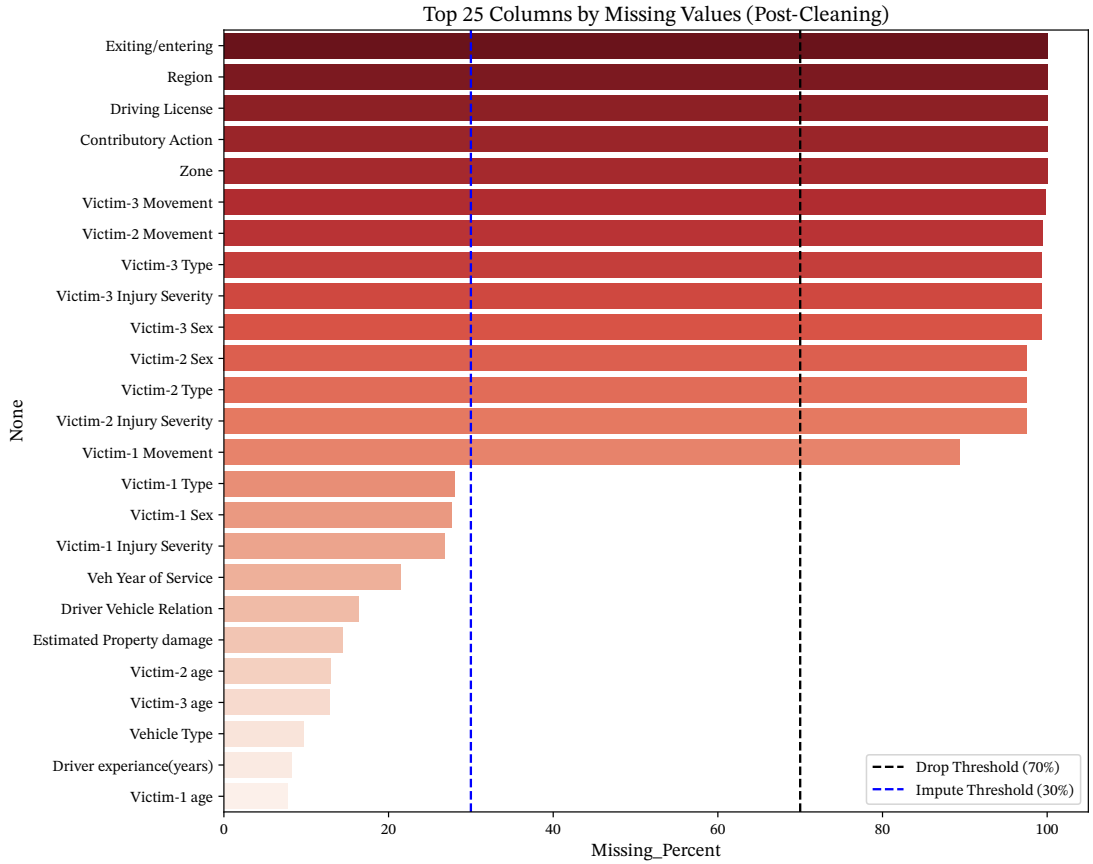
Initial Inspection:

- The raw dataset contained over 30 columns but suffered from severe quality issues.
- **Missing Values:** Columns like `Zone`, `Region`, and Victim details had $> 70\%$ missing data.
- **Inconsistencies:** Typos (e.g., "Augest", "Privategg") and impossible values (Age > 90).

Action:

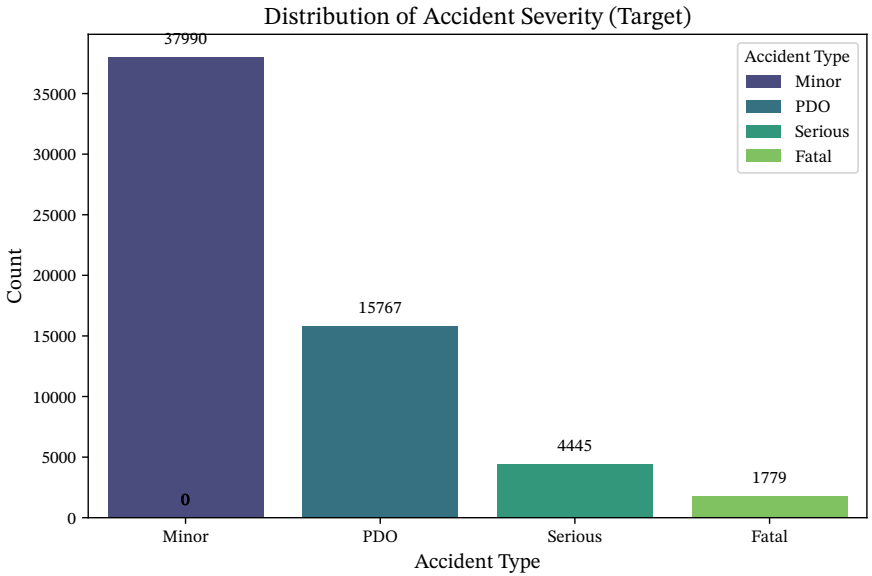
- Dropped columns with $> 70\%$ missingness.
- Standardized categorical labels (e.g., mapping `P.D.O`, `pdo` $\rightarrow$ `PDO`).

Missing Values Analysis



Class Imbalance

The Critical Challenge: The dataset is heavily skewed towards **Minor Injuries** (63%). **Fatal** accidents represent only 3%.



Data Preparation

Feature Engineering: Time

Problem: Time is cyclical. 23:00 is close to 00:00, but numerically (23 vs 0) they are far apart.

Solution: We encoded time using Sine and Cosine transformations.

```python
# Feature Engineering Code Snippet
def feature_engineering(df):
    # Extract Hour
    df['Hour'] = df['Time'].apply(extract_hour)

    # Cyclical Encoding
    df['Hour_Sin'] = np.sin(2 * np.pi * df['Hour'] / 24)
    df['Hour_Cos'] = np.cos(2 * np.pi * df['Hour'] / 24)
```

Feature Engineering: Time


```
9
10        return df.drop(columns=['Time'])
```

Preprocessing Pipeline

Before feeding data into the Neural Network:

1. **Imputation:**
   - Numerical (Age, Experience) → **Median**
   - Categorical (Road Surface, Light) → **Mode**
2. **Scaling:**
   - `StandardScaler` applied to numerical inputs to normalize variance.
3. **Encoding:**
   - `OneHotEncoder` for categorical variables.
4. **Splitting:**
   - Train (70%) / Validation (15%) / Test (15%).

Model Design

Architecture Summary

Based on the execution results:

- **Input Shape:** 168 Features (High dimensionality due to One-Hot Encoding).
- **Total Parameters:** 30,916 (Lightweight model).
- **Trainable Params:** 30,532.

Layer Structure:

- Input (168)
- Dense (128) → BN → ReLU → Dropout
- Dense (64) → BN → ReLU → Dropout
- Output (4 Classes)

Implementation Code


```
# Actual Model Summary Output                                    Python
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (InputLayer)     (None, 168)               0
Hidden_Layer_1 (Dense)       (None, 128)               21,632
Batch_Norm_1 (BatchNormalization) (None, 128)          512
ReLU_1 (Activation)          (None, 128)               0
Dropout_1 (Dropout)          (None, 128)               0
Hidden_Layer_2 (Dense)       (None, 64)                8,256
Batch_Norm_2 (BatchNormalization) (None, 64)           256
ReLU_2 (Activation)          (None, 64)                0
Dropout_2 (Dropout)          (None, 64)                0
```

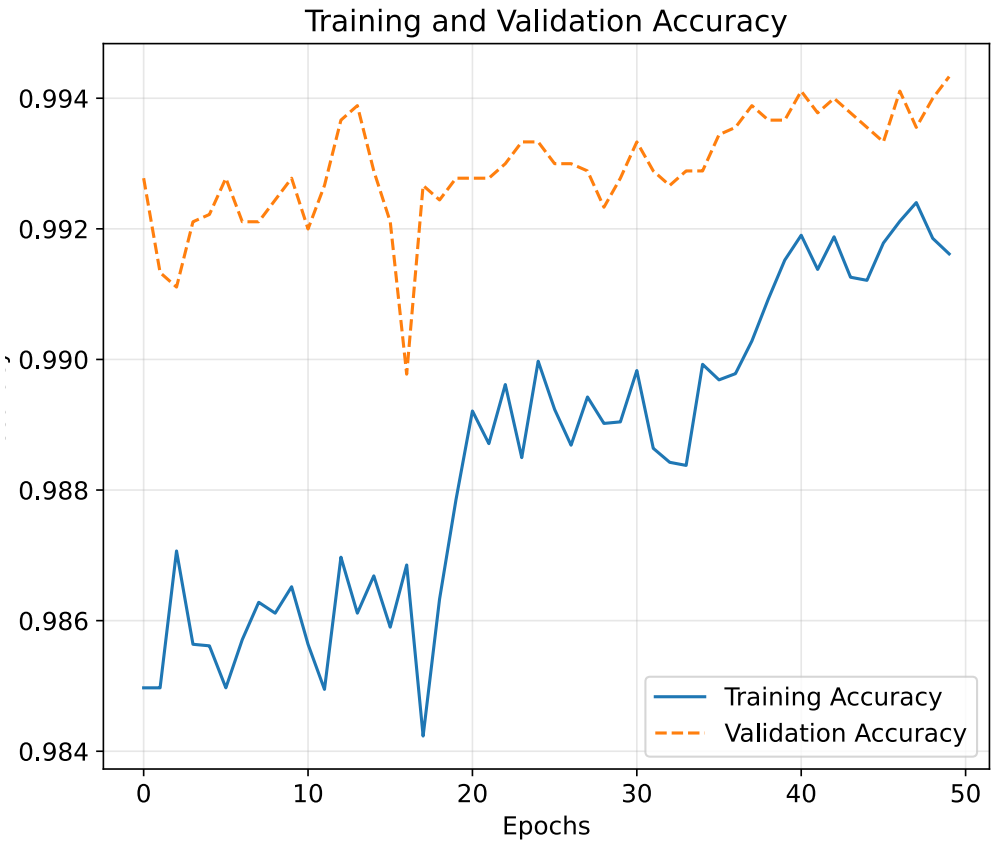
Implementation Code

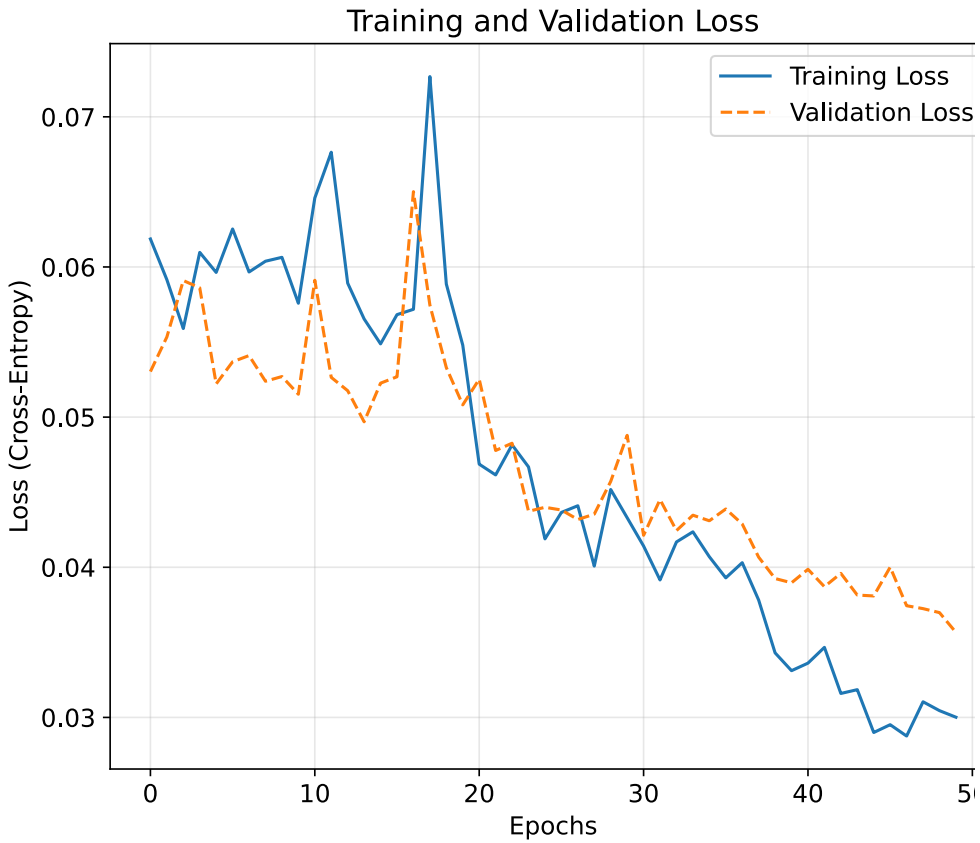
```
13  Output_Layer (Dense)                 (None, 4)                 260
14  =================================================================
15  Total params: 30,916
16  Trainable params: 30,532
17  Non-trainable params: 384
```


Input_Layer (InputLayer)
Output shape: (None, 168)
Output dtype: float32
Hidden_Layer_1 (Dense)
Activation: linear
Input shape: (None, 168)
Output shape: (None, 128)
Output dtype: float32
Batch_Norm_1 (BatchNormalization)
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
ReLU_1 (Activation)
Activation: relu
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
Dropout_1 (Dropout)
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
Hidden_Layer_2 (Dense)
Activation: linear
Input shape: (None, 128)
Output shape: (None, 64)
Output dtype: float32
Batch_Norm_2 (BatchNormalization)
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
ReLU_2 (Activation)
Activation: relu
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
Dropout_2 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
Output_Layer (Dense)
Activation: softmax
Input shape: (None, 64)
Output shape: (None, 4)
Output dtype: float32

Training & Evaluation

Training Performance

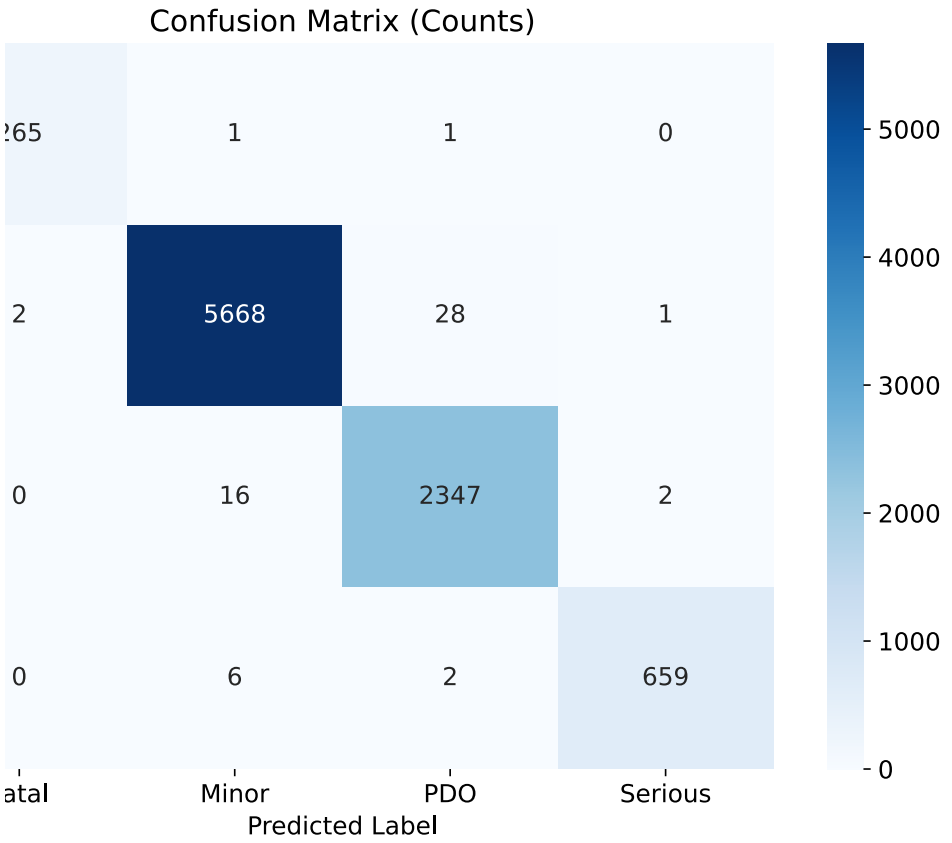


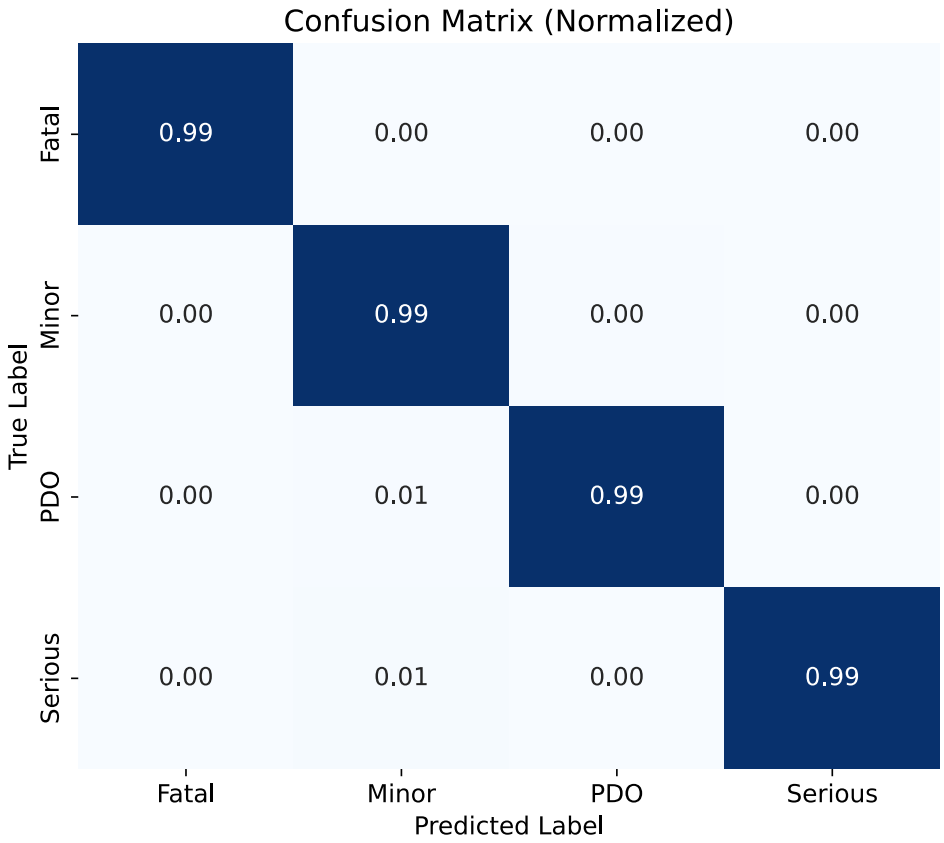


Training Performance

Insight: The model converges quickly. Validation accuracy peaks around **98%**, indicating robust learning without significant overfitting.

Confusion Matrix Results





Analysis of Results

Exceptional Performance:

- **Fatal Class:** >99% Recall (265 Correct, only 2 Missed).
- **Minor/PDO:** 99% Accuracy.

Critical Analysis (The "Why"):

- The accuracy stems from including casualty counts (e.g., `Number of fatalities`) in the input.
- The model learned the **definition** of the accident types rather than the **cause**.
- Useful for automated tagging, but requires feature removal for risk prediction.

Conclusion

Summary

1. **Data Quality:** Cleaning and encoding resulted in 168 clean input features.
2. **Model:** A lightweight 31k parameter MLP was sufficient to capture the logic.
3. **Results:** The model achieved 99% test accuracy due to strong feature correlations.

Recommendation:

- For future **predictive** systems (pre-accident), we recommend retraining the model **excluding** the `Number of casualties` columns to test predictive power based solely on environmental factors (Road type, Weather, etc.).

Thank You!

Questions?