**TOPIC: Regulatory Affairs of Road Accident Data 2020 India**

**Prepared by: Olympia Devi Gurumayum**

**UMID08072549377**

**PROJECT REPORT**

**Tools Used:** Python, SQL, Excel and Power BI

**Executive summary**

* The project analyses road-accident statistics for India in 2020, assesses the regulatory and policy landscape that governs road safety, identifies gaps in data collection and enforcement, and presents evidence-based recommendations for regulatory improvement.
* In 2020, India reported **989k road accidents**, which resulted in **81k deaths** and **908k injuries**. These figures represent a decline compared with 2019 (partly due to COVID-19 mobility restrictions), but the data also reveal structural issues in road safety reporting, enforcement, and protection of vulnerable road users.
* Recommendations focus on improving data quality, strengthening enforcement, targeted infrastructure upgrades, and institutional mechanisms for monitoring and accountability.

**Objectives**

1. Clean and preprocess the dataset for analysis.

2. Conduct exploratory data analysis (EDA).

3. Visualize insights with Python libraries and Power BI.

4. Provide recommendations for content strategy.

1. **Data Description Primary dataset**

The data contains road accident information for 50 cities of India in the year 2020.

**Columns**:

1. Million plus cities: Contains the name of the cities

2. Cause Category: 5 primary categories for classification of accidents (Traffic Control, Junction, Road Features, Impacting Vehicle/Object, Weather Conditions)

3. Cause Subcategory: Further classifying the exact cause for the accident

4. Outcome of Incident: Indicating, injuries, deaths and accidents

5. Count: Count in Millions for each incident

1. **Exploratory Data Analysis (EDA)**

1. Indian cities with max accidents are explored.

2. Out of 989k accidents happened 908k turned out to be non-fatal and the remaining 81k turned out to be fatal.

3. The major causes of the accident are Road Features, Junction, Traffic Control, Traffic Violation, etc.

4. Chennai, Delhi, Jabalpur, Bangalore and Indore are the major cities with max accidents.

1. **Feature Engineering**

**Tools used:** SQL, Power Query Editor and Pandas

* 1. Ensured there are no null values.
  2. Normalized the data into **1NF** where comma separated values in the column **“Cause Category” and “Cause Subcategory”** into multiple rows.
  3. Changed the column “Outcome of Incident” into binary codes where the values- Persons Killed, Grievously Injured and Minor Injury are coded as 0 for non-fatal. Values- Persons Killed are coded as 1 for fatal.
  4. The values Total Injured and Total number of Accidents are removed from the column “Outcome of Incident” as it serves no purpose with the model.
  5. Dropped the column “Count” and categorical values of the column “Outcome of the Incident” as it’s not necessary for the logistic regression model.
  6. To prepare the data for machine learning models, all the columns which contain categorical values were converted into separate binary columns using **one-hot encoding**.

**Code used:**

df\_encoded = pd.get\_dummies(df,columns=cols\_to\_encode)

* + 1. Here **df\_encoded** is the transformed version of the original datasetthat is **df** after applying **one-hot encoding.**
    2. We need to apply the logistic regression here in the transformed dataset that is df\_encoded.

1. **Machine Learning**

**1. Objective**

The aim is to evaluate Logistic Regression as a baseline classifier for predicting the binary target variable “Outcome\_Code”. Multiple configurations were tested, including unbalanced Logistic Regression, weight-balanced Logistic Regression and threshold tuning.

**2. Model Development Steps**

* **Data Preparation**
  + Encoded categorical features and scaled numerical features.
  + Defined features (X) and target (y = Outcome\_Code).
  + Split dataset into training and testing sets (80:20).

**3. Experiments & Results**

**A. Unbalanced Logistic Regression (Default settings)**

• Accuracy: 66.2%

• Confusion Matrix: [[794, 0], [406, 0]]

• Observations: Model predicted all samples as class 0. Recall for class 0 = 1.0, Recall for class 1 = 0.0. Accuracy matched the majority class distribution (~66%). ROC-AUC = 0.36, showing the model failed to distinguish between classes.

**B. Weight-Balanced Logistic Regression (Threshold = 0.5)**

• Accuracy: 41.7%

• Confusion Matrix: [[358, 436], [263, 143]]

• Observations: The model started predicting both classes, but with significant trade-offs. Overall accuracy dropped, ROC-AUC remained poor and performance was weak for both classes.

**C. Weight-Balanced Logistic Regression with Threshold = 0.3**

• Accuracy: 34.0%

• Confusion Matrix: [[0, 794], [0, 406]]

• Observations: Lowering the threshold caused the model to classify all instances as class 1. Recall for class 1 reached 1.0 but class 0 detection dropped to 0.0. Accuracy fell further and ROC-AUC did not improve.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model Type** | **Accuracy** | **Class 0 Recall** | **Class 1 Recall** | **ROC-AUC** | **Remarks** |
| Unbalanced Logistic Regression (0.5) | 66.2% | 1.00 | 0.00 | 0.36 | Predicts only majority class |
| Balanced Logistic Regression (0.5) | 41.7% | 0.45 | 0.35 | 0.36 | Detects both classes but weak separation |
| Logistic Regression (0.3) | 33.8% | 0.00 | 1.00 | 0.5 | Overpredicts class 1 and unstable |

**4. Conclusion**

* Logistic Regression failed to separate the classes effectively.
* ROC-AUC remained consistently low (0.36), indicating the model was performing worse than random guessing.
* The model was highly biased in the baseline run (all predictions = class 0). After balancing and threshold tuning, it becameunstableand produced poor trade-offs between precision and recall.
* Therefore, Logistic Regression was deemed unsuitable for this dataset.

**5. Next Steps that can be considered**

* Exploring more advanced models such as Random Forest, Gradient Boosted Trees (XGBoost/LightGBM) that can handle non-linear patterns and imbalanced data more effectively.
* Applying resampling techniques (e.g., SMOTE) to improve minority class representation during training.
* Evaluate using metrics more appropriate for imbalanced datasets: F1-score (macro/weighted), ROC-AUC, Precision-Recall curves.