## 1. Introduction:

The Road Accident Dataset is a comprehensive collection of data related to road accidents, capturing key details such as accident causes, driver behaviour, environmental conditions, and severity levels. With **5,001 recorded cases**, this dataset serves as a valuable resource for analyzing traffic safety trends, identifying high-risk factors, and proposing data-driven strategies to minimize accidents and fatalities.

The dataset is structured with **30 attributes**, covering aspects such as accident severity, the number of injuries, vehicle conditions, weather conditions, and emergency response time, etc. The availability of both categorical and numerical data makes it ideal for statistical analysis, machine learning models, and predictive analytics in road safety research.

This dataset is called **secondary data** as we are using to find some insights which can help to increase the surviving rate for the people.

## 2. Problem Definition & Dataset Selection

## 2.1 Problem Definition:

Develop a predictive model to assess the severity of road accidents based on factors such as weather conditions, road type, driver attributes, traffic volume, and emergency response time. This analysis aims to identify key contributors to severe accidents and provide actionable insights for improving road safety and reducing fatalities.

## 2.2 Dataset Selection:

The Road Accident Dataset has been carefully selected to analyze the key factors influencing road accidents. It provides a balanced mix of categorical and numerical data, enabling both statistical and machine learning approaches to uncover patterns, assess risk factors, and improve road safety policies.

# 3. Data Collection & Pre-processing

## 3.1 Data Collection

#### 3.1.1 Source of the Dataset:

The dataset was obtained from a structured CSV file named road\_accident\_dataset.csv. It contains 5,001 records and 30 attributes, covering various factors related to road accidents.

#### 3.1.2 Structure of the Dataset:

The dataset consists of:

- Categorical Variables: Location, Weather Conditions, Road Type, etc.
- Numerical Variables: Number of Vehicles Involved, Speed Limit,
   Economic Loss, etc.

## 3.2 Data Pre-processing

## 3.2.1 Handling Missing Values:

Before analysis, it is crucial to check for missing values and handle them appropriately. **Before handling we have to import two python libraries pandas and numpy.** Pandas library is used for data manipulation and analysis whereas numpy library is used to perform any operations where numerical values are involved.

Here are the steps taken to handle missing values:

### i. Read CSV file:

By using **read\_csv("road\_accident\_dataset.csv) method** we are able to extract the dataset and ready to perform operations on it.

#### ii. Check missing values:

By using **isnull() method**, we can learn if there is any missing values are present or not and with using **sum() method**, we can get the total count of missing values per column.

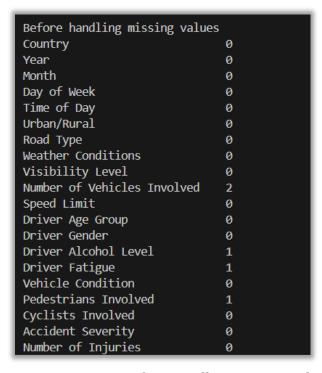


Figure 3.2.1 – ii Before Handling Missing Values

### iii. Applying imputation:

By using **fillna() method**, we can fill the missing values by applying imputation using **mean()**, **median()**, **and mode() with inplace=True** which states modifying original dataframe instead of creating and returning a new one. By **default the value is false**.

### iv. Rounding the values:

By using **round() method**, we can round the values to match with values present inside the columns. For example: If I have single integer values present in a column like "Number of Vehicles Involved" then by using round(0), I can round the values to 0 decimal place if the missing value comes in datatype float.

#### v. Check duplicate values:

By using **duplicated() method**, we can check whether there is any duplicate values present in the dataset.

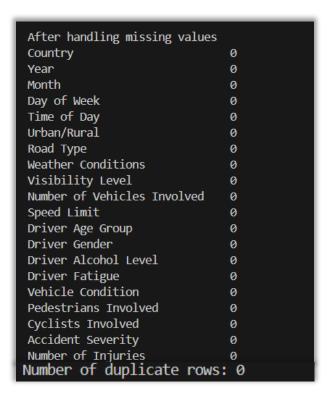


Figure 3.2.1 – v After Handling Missing Values

#### vi. Save the cleaned dataset:

By using to\_csv("cleaned\_road\_accident\_dataset.csv") method, we can save the dataset with all missing values been filled.

## 3.3 Snippet Code Screenshot:

```
import pandas as pd
import numpy as np

# Read the csv file

ff = pd.read_csv("cleaned_road_accident_dataset.csv")

# Task-2

# checking if there is any missing values in dataset
print("Before handling missing values")

print(df.isnull().sum()) # Counts missing values per column
print(df.isnull().sum()) # Counts missing values per column
print(df.info()) # Provides an overview of non-null counts

# Applying imputation methods
df["Number of Vehicles Involved"].fillna(df["Driver Alcohol Level"].median(), inplace=True) # Median
df["Driver Alcohol Level"].fillna(df["Driver Alcohol Level"].median(), inplace=True) # Mode
df["Pedestrians Involved"].fillna(df["Pedestrians Involved"].median(), inplace=True)

# Rounding the values to the desired decimal places
df["Number of Vehicles Involved"] = df["Number of Vehicles Involved"].round(0) # Rounding to 0 decimals
df["Driver Alcohol Level"] = df["Driver Alcohol Level"]
df["Pedestrians Involved"] = df["Driver Alcohol Level"]

# Check again if there is any missing value left after imputation
print("After handling missing values")
print(df.isnull().sum()) # Ensure no missing values remain

# Save cleaned dataset
df.to_csv("cleaned_road_accident_dataset.csv", index=False) # Saves without index
```

# 4. Data Summarization & Descriptive Analysis

## 4.1 Compute Central Tendency

Central Tendencies are the numerical values that are used to represent a large collection of numerical data. These obtained numerical values are called central or average values. A central or average value of any statistical data or series is the variable's value representative of the entire data or its associated frequency distribution.

## 4.1.1 Measures of Central Tendency:

The central tendencies are achieved by using these three measures:

#### Mean:

Mean is called as the average, is calculated by summing up all the values present inside a column.

#### • Median:

Median is the middle value in a dataset when the values are arranged in ascending or descending order. If there is an even number of values, the median is the average of the two middle values. Unlike the mean, the median is less affected by outliers.

#### • Mode:

The mode is the value that occurs most frequently in a dataset. It is particularly useful for categorical data but can also be applied to numerical data.

Here are the steps taken to perform central tendency:

#### i. Identify the numerical columns:

By using **select\_dtypes(include=[np.number]) function**, we can clearly identify the numerical columns inside the dataset.

#### ii. Compute mean, median, and mode:

By using **mean()**, **median() method**, we can achieve and perform computation on the numerical columns. But to calculate mode, we have to use with **mode()**.iloc[0] **method**. As mode() can return multiple values, iloc[0] ensures it takes the first mode value from the column.

## iii. Display the results:

By using simply with **print() method** with mean, median, and mode.

Mean Values:	
Year	2024.000000
Visibility Level	276.383653
Number of Vehicles Involved	2.515897
Speed Limit	74.501500
Driver Alcohol Level	0.123906
Driver Fatigue	0.494901
Pedestrians Involved	1.004799
Cyclists Involved	0.995001
Number of Injuries	9.565287
Number of Fatalities	1.931814
Emergency Response Time	32.519606
Traffic Volume	5109.825612
Insurance Claims	4.507499
Medical Cost	25187.006589
Economic Loss	50431.604469
Population Density	2504.812441
dtype: float64	

Figure 4.1.1 - ii - Mean

Median Values:	
Year	2024.000000
Visibility Level	273.362896
Number of Vehicles Involved	3.000000
Speed Limit	74.000000
Driver Alcohol Level	0.123025
Driver Fatigue	0.000000
Pedestrians Involved	1.000000
Cyclists Involved	1.000000
Number of Injuries	10.000000
Number of Fatalities	2.000000
Emergency Response Time	32.409246
Traffic Volume	5127.396557
Insurance Claims	4.000000
Medical Cost	25127.235433
Economic Loss	50316.086945
Population Density	2524.557334
dtype: float64	

Figure 4.1.1 - ii - Median

Mode Values:	
Year	2024.000000
	50.033725
Visibility Level	
Number of Vehicles Involved	3.000000
Speed Limit	69.000000
Driver Alcohol Level	0.000063
Driver Fatigue	0.000000
Pedestrians Involved	1.000000
Cyclists Involved	0.000000
Number of Injuries	17.000000
Number of Fatalities	0.000000
Emergency Response Time	5.022489
Traffic Volume	103.603462
Insurance Claims	4.000000
Medical Cost	517.965083
Economic Loss	1013.695660
Population Density	10.979547
Name: 0, dtype: float64	

Figure 4.1.1 - ii - Mode

## 4.2 Variation

Measures of variation describe how spread out the data points are in a dataset. They help in understanding data distribution and identifying outliers.

#### **4.2.1** Measure of Variation:

Variation can be achieved by following three approaches:

### • Range:

Range is described as the difference between the maximum and minimum values.

#### • Variance:

Variance is described as how far each data point is from the mean, squared to prevent negative values

#### • Standard Deviation:

Standard deviation is described as the square root of variance, representing the average deviation of data points from the mean.

Here are the steps taken to perform measure of variation/spread:

## i. Computer range, variance, and standard deviation:

We have calculated based on numerical observations. By using max() and min() methods, we have calculated range for the dataset. For variance, we have used var() method to calculate variance. For standard deviation, we have used std() method to get the standard deviation for the dataset.

Range:	
Year	0.000000
Visibility Level	449.926840
Number of Vehicles Involved	3.000000
Speed Limit	89.000000
Driver Alcohol Level	0.249878
Driver Fatigue	1.000000
Pedestrians Involved	2.000000
Cyclists Involved	2.000000
Number of Injuries	19.000000
Number of Fatalities	4.000000
Emergency Response Time	54.961841
Traffic Volume	9890.824045
Insurance Claims	9.000000
Medical Cost	49464.308460
Economic Loss	98972.983326
Population Density	4987.708692
dtype: float64	

Figure 4.2.1 - i - Range

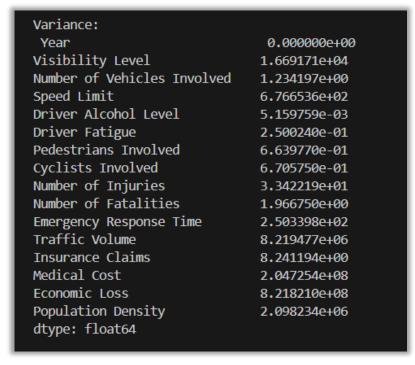


Figure 4.2.1 - i - Variance

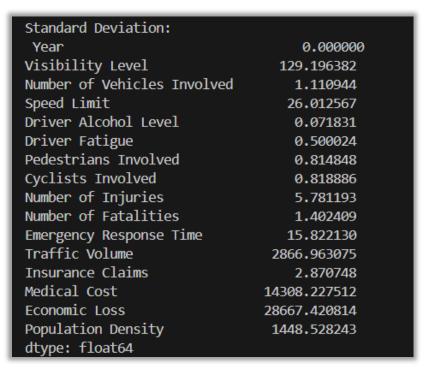


Figure 4.2.1 - i – Standard Deviation

## 4.3 Cross - Tabulation & Frequency Distribution

#### **4.3.1** Cross - Tabulation:

Cross tabulation is a method used to analyze the relationship between two or more categorical variables.

Here are the steps taken to perform cross tabulation:

### i. Identify two categorical observations:

Before executing, we need to identify which categorical variables gives us the learning of relationship between them.

#### ii. Compute cross tabulation:

Here, we have identified the categorical variables as "Accident Severity" and "Weather Conditions". By using **crosstab() method,** we are able to identify the relationship between the variables. It gives us the result in terms of count as numerical values.

Cross-tabulation re						
Weather Conditions	Clear	Foggy	Rainy	Snowy	Windy	
Accident Severity						
Minor	295	317	353	316	349	
Moderate	319	336	316	336	351	
Severe	317	326	362	355	353	

Figure 4.3.1 - ii - Cross - Tabulation

## **4.3.2 Frequency Distribution:**

Frequency distribution is a representation that displays how often any specific values occur in dataset. This method can be performed on both categorical and numerical observations. Frequency distribution can be performed on both categorical and numerical observations. Here, we have chosen categorical observation to perform.

Here are the steps taken to perform frequency distribution:

## i. Select categorical variables:

By using **select\_dtypes(include=['object'])** method, it will make sure you get only the observations which are string or text in datatype.

### ii. Compute frequency distribution:

We have used **for loop** which loops through each column in **categorical\_cols** (a dataframe containing only categorical **columns**). By using **value\_counts()** method, we have calculated frequency distribution.

```
Frequency Distribution for Country:
Country
India
            533
Brazil
            529
China
           517
Germany
           504
USA
            500
          493
Canada
UK
           489
Australia 482
           477
Japan
Russia
           477
Name: count, dtype: int64
 Frequency Distribution for Driver Age Group:
 Driver Age Group
 26-40
        1030
 61+
        1011
         1005
 <18
 41-60 987
 18-25
          968
 Name: count, dtype: int64
 Frequency Distribution for Urban/Rural:
 Urban/Rural
 Rural 2517
 Urban
         2484
 Name: count, dtype: int64
 Frequency Distribution for Road Type:
 Road Type
 Highway
             1668
            1667
 Main Road
 Street
            1666
 Name: count, dtype: int64
 Frequency Distribution for Weather Conditions:
 Weather Conditions
 Windy
         1053
        1031
 Rainy
 Snowy
        1007
        979
 Foggy
          931
 Clear
 Name: count, dtype: int64
```

Figure 4.3.2 - ii - Frequency Distribution

## 4.4 Snippet Code Screenshot

```
#Task-3 (Data Summarization & Descriptive Analysis)
# Compute central tendency (Mean, Median, Mode)

# Select numerical columns
num_columns = df.select_dtypes(include=[np.number])

# Compute Mean
mean = num_columns.mean()

# Compute Median
median = num_columns.median()

# Compute Mode (returns multiple values, so we take the first)
mode = num_columns.mode().iloc[0]

# Display results
print("Nean Values:\n", mean)
print("\nMedian Values:\n", median)
print("\nMode Values:\n", mode)

# Measures of Variation ( Range, Variance, Standard Deviation)

# Compute Range (Max - Min)
# Range is difference between maximum and minimum values in dataset
range = num_columns.max() - num_columns.min()

# Compute Variance
variance = num_columns.var()

# Compute Standard Deviation
standard_deviation = num_columns.std()
```

```
# Display results
print("\nRange:\n", range)
print("\nNariance:\n", variance)
print("\nStandard Deviation:\n", standard_deviation)

# Cross-tabulation
# Cross tabulation is a method used to analyze the relationship between two or more categorical variables

# Perform cross-tabulation: Example (Accident Severity vs. Weather Conditions)
cross_tab = pd.crosstab(df["Accident Severity"], df["Weather Conditions"])

# Display the cross-tabulation table
print["cross-tabulation results:\n"]
print(cross_tab)

# Save the cross-tabulation to a CSV file
cross_tab.to_csv("road_accident_crosstab.csv")

# Frequency distribution is a representation that displays how often any specific values occur in dataset
# Select categorical columns ( Can also be calculated based on numerical values)
categorical_cols = df.select_dtypes(include=['object'])

# Compute and display frequency distribution for each categorical column
for col in categorical_cols.columns:
    print(f"\nFrequency Distribution for {col}:\n")
    print(df[col].value_counts())
```

DATA INSIGHTS INTO GLOBAL ROAD SAFETY		