chicago

May 24, 2023

1 Project Overview

The City of Chicago Vehicle Safety Board (CCVSB) interested in reducing traffic accidents and becoming aware of any interesting patterns.

2 Business Problem

The business problem is to build a classifier that can predict the primary contributory cause of car accidents in Chicago city.

3 Defineing the Questions

1. *Are there any specific locations or road segments in Chicago city that have a higher frequency of car accidents?*
2. *What are the contributing factors or characteristics associated with severe car accidents in Chicago city?*
3. *Are there any seasonal or temporal patterns in car accidents in Chicago city?*
4. *Can we build a classification model to predict the primary contributory cause of car accidents?*

Additionally, I will create a classification model to categorize accidents into two main groups for future reference:

1. Accidents caused by unintentional factors: *These accidents occur when drivers are not purposely or knowingly involved in causing the accident. They may result from factors such as driver error, environmental conditions, mechanical failures, or other unforeseen circumstances.*
2. Accidents caused by intentional factors: *These accidents involve drivers who are deliberately or knowingly involved in causing the accident. They may engage in reckless driving, aggressive behavior, or intentionally violate traffic laws, leading to the occurrence of the accident.*

By developing this classification model, we aim to distinguish between accidents that result from unintentional factors and those that involve intentional actions. This categorization will enable us to analyze and understand the different contributing factors and characteristics associated with each category, leading to targeted strategies for accident prevention and improving overall road safety.

4 Data

The dataset was from Chicago city. Their were three datasets that was obtain from Chicago Data Portal:

```
* Traffic_Crashes_-_People
* Traffic_Crashes_-_Vehicles
* Traffic_Crashes_-_Crashes
```

The data provides up-to-date information as per now May 2023 from 2015.The two datasets was cleaned and merged to one.

5 Data Grocery

Index	Column Name	Description
1	SEX	Gender of the person involved in the accident
2	AGE	Age of the person involved in the accident
3	DRIVER_ACTION	Action taken by the driver before the accident
4	DRIVER_VISION	Vision condition of the driver during the accident
5	PHYSICAL_CONDITION	Physical condition of the driver at the time of the accident
6	MANEUVER	Maneuver performed by the driver during the accident
7	POSTED_SPEED_LIMIT	Speed limit posted on the road where the accident occurred
8	TRAFFIC_CONTROL_DEVICE	Traffic control device present at the accident location
9	DEVICE_CONDITION	Condition of the traffic control device
10	WEATHER_CONDITION	Weather conditions during the accident
11	LIGHTING_CONDITION	Lighting conditions during the accident
12	TRAFFICWAY_TYPE	Type of the trafficway where the accident occurred
13	ROADWAY_SURFACE_COND	Surface condition of the roadway at the accident location
14	ROAD_DEFECT	Defects present on the road where the accident occurred
15	PRIM_CONTRIBUTORY_CAUSE	Primary contributory cause of the accident
16	CRASH_HOUR	Hour of the day when the accident occurred
17	CRASH_DAY_OF_WEEK	Day of the week when the accident occurred
18	CRASH_MONTH	Month when the accident occurred
19	LATITUDE	Latitude coordinate of the accident location
20	LONGITUDE	Longitude coordinate of the accident location
21	LOCATION	Location description of the accident

6 Recording the Experimental Design

To record the experimental design for building the classifier to predict the primary contributory cause of car accidents in Chicago, I used the following steps:

- Data Collection: I gather crashes, vehicle and people accident data from https://data.cityofchicago.org/Transportation/Traffic-Crashes-Vehicles/68nd-jvt3 and https://data.cityofchicago.org/Transportation/Traffic-Crashes-People/u6pd-qa9d
- Data Preprocessing: Clean the data by handling missing values, inconsistencies, and outliers.

Transform categorical variables into numerical representations suitable for machine learning algorithms. Normalize or standardize numerical features if necessary

- Exploratory Data Analysis (EDA): Perform exploratory analysis to understand the characteristics and distributions of variables. Identify patterns, correlations, or any interesting insights within the data. Visualize the data using plots, charts, or graphs to aid in understanding.
- Feature Engineering:Extract relevant features from the available data that may contribute to predicting the primary contributory cause of car accidents.
- Target Variable Binning: Analyze the distribution of the primary contributory cause categories. Merge or eliminate categories with very few samples to limit the number of target categories.
- Feature Selection: Select the most informative features that are likely to have a significant impact on the prediction.
- Model Selection and Training:Choose a suitable machine learning algorithm for multi-class classification, considering factors like performance, interpretability, and scalability.Split the preprocessed data into training and testing sets. Train the chosen model on the training data using appropriate algorithms and methodologies.
- Model Evaluation:Evaluate the trained model's performance using relevant evaluation metrics for multi-class classification, such as accuracy, precision, recall, F1-score, or confusion matrix.Perform cross-validation techniques like k-fold cross-validation to assess the model's robustness.
- Model Optimization:Fine-tune the model by optimizing hyperparameters to improve its performance. Use techniques like grid search, random search.
- Predictions and Interpretation:Use the optimized model to predict the primary contributory cause of car accidents for new instances. Analyze the predictions and interpret the results to gain insights into patterns, potential causes, or any interesting findings that can aid accident prevention efforts.
- Reporting and Recommendations:Summarize the findings and insights obtained from the classifier. Provide actionable recommendations for the Vehicle Safety Board or the City of Chicago based on the analysis and predictions.

7 Loading the datasets

[1]:
```
# importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[2]:
```
# Reading csv file
people=pd.read_csv('C:\\Users\\Admin\\Documents\\Iano\\phase 3␣
↪project\\dsc-phase-3-project-v2-3\\.data\\Traffic_Crashes_-_People.csv')
```

```
vehicles=pd.read_csv('C:\\Users\\Admin\\Documents\\Iano\\phase 3␣
↪project\\dsc-phase-3-project-v2-3\\.data\\Traffic_Crashes_-_Vehicles.csv')
crashes=pd.read_csv('C:\\Users\\Admin\\Documents\\Iano\\phase 3␣
↪project\\dsc-phase-3-project-v2-3\\.data\\Traffic_Crashes_-_Crashes.csv')
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_14600\4169662019.py:2: DtypeWarning:
Columns (29) have mixed types. Specify dtype option on import or set
low_memory=False.
  people=pd.read_csv('C:\\Users\\Admin\\Documents\\Iano\\phase 3 project\\dsc-
phase-3-project-v2-3\\.data\\Traffic_Crashes_-_People.csv')
C:\Users\Admin\AppData\Local\Temp\ipykernel_14600\4169662019.py:3: DtypeWarning:
Columns (2,19,21,40,41,42,44,48,49,50,53,55,58,59,61,71) have mixed types.
Specify dtype option on import or set low_memory=False.
  vehicles=pd.read_csv('C:\\Users\\Admin\\Documents\\Iano\\phase 3 project\\dsc-
phase-3-project-v2-3\\.data\\Traffic_Crashes_-_Vehicles.csv')
```

[3]:
```
# A function to print the shape of our datasets
def print_dataset_shape(*datasets):
    """
    Prints the shape of one or more datasets (number of rows and columns).
    Assumes datasets are in a Pandas DataFrame format.
    """
    for idx, dataset in enumerate(datasets):
        print(f"Dataset {idx + 1} - Number of rows: {dataset.shape[0]}")
        print(f"Dataset {idx + 1} - Number of columns: {dataset.shape[1]}")
# print the shape of our dataset
print_dataset_shape(people, vehicles,crashes)
```

```
Dataset 1 - Number of rows: 1584616
Dataset 1 - Number of columns: 30
Dataset 2 - Number of rows: 1472816
Dataset 2 - Number of columns: 72
Dataset 3 - Number of rows: 722809
Dataset 3 - Number of columns: 49
```

[4]:
```
# A function to get info of our dataset
#data = {
 #   'vehicles': [...],  # Vehicle information
 #   'people': [...],    # People information
 #   'crashes': [...]    # Crash information
#}

#vehicles, people, crashes = get_info(data)
```

[5]:
```
# Getting the info of our data
people.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1584616 entries, 0 to 1584615
Data columns (total 30 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   PERSON_ID              1584616 non-null  object
 1   PERSON_TYPE            1584616 non-null  object
 2   CRASH_RECORD_ID        1584616 non-null  object
 3   RD_NO                  1574225 non-null  object
 4   VEHICLE_ID             1553626 non-null  float64
 5   CRASH_DATE             1584616 non-null  object
 6   SEAT_NO                320590 non-null   float64
 7   CITY                   1156363 non-null  object
 8   STATE                  1172272 non-null  object
 9   ZIPCODE                1057964 non-null  object
 10  SEX                    1559451 non-null  object
 11  AGE                    1123239 non-null  float64
 12  DRIVERS_LICENSE_STATE  931218 non-null   object
 13  DRIVERS_LICENSE_CLASS  785449 non-null   object
 14  SAFETY_EQUIPMENT       1580144 non-null  object
 15  AIRBAG_DEPLOYED        1554823 non-null  object
 16  EJECTION               1565274 non-null  object
 17  INJURY_CLASSIFICATION  1583928 non-null  object
 18  HOSPITAL               270739 non-null   object
 19  EMS_AGENCY             167981 non-null   object
 20  EMS_RUN_NO             27702 non-null    object
 21  DRIVER_ACTION          1261166 non-null  object
 22  DRIVER_VISION          1260713 non-null  object
 23  PHYSICAL_CONDITION     1262045 non-null  object
 24  PEDPEDAL_ACTION        29391 non-null    object
 25  PEDPEDAL_VISIBILITY    29333 non-null    object
 26  PEDPEDAL_LOCATION      29390 non-null    object
 27  BAC_RESULT             1262189 non-null  object
 28  BAC_RESULT VALUE       1859 non-null     float64
 29  CELL_PHONE_USE         1158 non-null     object
dtypes: float64(4), object(26)
memory usage: 362.7+ MB
```

[6]:
```python
# Getting the info of our data
vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1472816 entries, 0 to 1472815
Data columns (total 72 columns):
 #   Column                    Non-Null Count    Dtype
---  ------                    --------------    -----
 0   CRASH_UNIT_ID             1472816 non-null  int64
 1   CRASH_RECORD_ID           1472816 non-null  object
 2   RD_NO                     1463413 non-null  object
```

```
3   CRASH_DATE                1472816 non-null  object
4   UNIT_NO                   1472816 non-null  int64
5   UNIT_TYPE                 1470808 non-null  object
6   NUM_PASSENGERS            218014 non-null   float64
7   VEHICLE_ID                1439674 non-null  float64
8   CMRC_VEH_I                27533 non-null    object
9   MAKE                      1439669 non-null  object
10  MODEL                     1439525 non-null  object
11  LIC_PLATE_STATE           1308626 non-null  object
12  VEHICLE_YEAR              1206500 non-null  float64
13  VEHICLE_DEFECT            1439674 non-null  object
14  VEHICLE_TYPE              1439674 non-null  object
15  VEHICLE_USE               1439674 non-null  object
16  TRAVEL_DIRECTION          1439674 non-null  object
17  MANEUVER                  1439674 non-null  object
18  TOWED_I                   181112 non-null   object
19  FIRE_I                    1188 non-null     object
20  OCCUPANT_CNT              1439674 non-null  float64
21  EXCEED_SPEED_LIMIT_I      2397 non-null     object
22  TOWED_BY                  135298 non-null   object
23  TOWED_TO                  83272 non-null    object
24  AREA_00_I                 51730 non-null    object
25  AREA_01_I                 390692 non-null   object
26  AREA_02_I                 236041 non-null   object
27  AREA_03_I                 140245 non-null   object
28  AREA_04_I                 141169 non-null   object
29  AREA_05_I                 218874 non-null   object
30  AREA_06_I                 228709 non-null   object
31  AREA_07_I                 208279 non-null   object
32  AREA_08_I                 216804 non-null   object
33  AREA_09_I                 91248 non-null    object
34  AREA_10_I                 132225 non-null   object
35  AREA_11_I                 258240 non-null   object
36  AREA_12_I                 253662 non-null   object
37  AREA_99_I                 163988 non-null   object
38  FIRST_CONTACT_POINT       1436590 non-null  object
39  CMV_ID                    15356 non-null    float64
40  USDOT_NO                  8735 non-null     object
41  CCMC_NO                   1894 non-null     object
42  ILCC_NO                   1317 non-null     object
43  COMMERCIAL_SRC            10326 non-null    object
44  GVWR                      8652 non-null     object
45  CARRIER_NAME              14683 non-null    object
46  CARRIER_STATE             13796 non-null    object
47  CARRIER_CITY              13546 non-null    object
48  HAZMAT_PLACARDS_I         301 non-null      object
49  HAZMAT_NAME               56 non-null       object
50  UN_NO                     522 non-null      object
```

```
 51  HAZMAT_PRESENT_I          11210 non-null    object
 52  HAZMAT_REPORT_I           10886 non-null    object
 53  HAZMAT_REPORT_NO          1 non-null        object
 54  MCS_REPORT_I              10934 non-null    object
 55  MCS_REPORT_NO             7 non-null        object
 56  HAZMAT_VIO_CAUSE_CRASH_I  11051 non-null    object
 57  MCS_VIO_CAUSE_CRASH_I     10844 non-null    object
 58  IDOT_PERMIT_NO            850 non-null      object
 59  WIDE_LOAD_I               128 non-null      object
 60  TRAILER1_WIDTH            2735 non-null     object
 61  TRAILER2_WIDTH            324 non-null      object
 62  TRAILER1_LENGTH           2222 non-null     float64
 63  TRAILER2_LENGTH           65 non-null       float64
 64  TOTAL_VEHICLE_LENGTH      2699 non-null     float64
 65  AXLE_CNT                  4008 non-null     float64
 66  VEHICLE_CONFIG            12720 non-null    object
 67  CARGO_BODY_TYPE           12154 non-null    object
 68  LOAD_TYPE                 11624 non-null    object
 69  HAZMAT_OUT_OF_SERVICE_I   10560 non-null    object
 70  MCS_OUT_OF_SERVICE_I      10806 non-null    object
 71  HAZMAT_CLASS              1018 non-null     object
dtypes: float64(9), int64(2), object(61)
memory usage: 809.0+ MB
```

[7]:
```python
# Getting the info of our data
crashes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 722809 entries, 0 to 722808
Data columns (total 49 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   CRASH_RECORD_ID                722809 non-null  object
 1   RD_NO                          718253 non-null  object
 2   CRASH_DATE_EST_I               54664 non-null   object
 3   CRASH_DATE                     722809 non-null  object
 4   POSTED_SPEED_LIMIT             722809 non-null  int64
 5   TRAFFIC_CONTROL_DEVICE         722809 non-null  object
 6   DEVICE_CONDITION               722809 non-null  object
 7   WEATHER_CONDITION              722809 non-null  object
 8   LIGHTING_CONDITION             722809 non-null  object
 9   FIRST_CRASH_TYPE               722809 non-null  object
 10  TRAFFICWAY_TYPE                722809 non-null  object
 11  LANE_CNT                       199002 non-null  float64
 12  ALIGNMENT                      722809 non-null  object
 13  ROADWAY_SURFACE_COND           722809 non-null  object
 14  ROAD_DEFECT                    722809 non-null  object
 15  REPORT_TYPE                    702521 non-null  object
```

```
 16  CRASH_TYPE                     722809 non-null  object
 17  INTERSECTION_RELATED_I         165794 non-null  object
 18  NOT_RIGHT_OF_WAY_I             33751 non-null   object
 19  HIT_AND_RUN_I                  225032 non-null  object
 20  DAMAGE                         722809 non-null  object
 21  DATE_POLICE_NOTIFIED           722809 non-null  object
 22  PRIM_CONTRIBUTORY_CAUSE        722809 non-null  object
 23  SEC_CONTRIBUTORY_CAUSE         722809 non-null  object
 24  STREET_NO                      722809 non-null  int64
 25  STREET_DIRECTION               722805 non-null  object
 26  STREET_NAME                    722808 non-null  object
 27  BEAT_OF_OCCURRENCE             722804 non-null  float64
 28  PHOTOS_TAKEN_I                 9081 non-null    object
 29  STATEMENTS_TAKEN_I             15398 non-null   object
 30  DOORING_I                      2186 non-null    object
 31  WORK_ZONE_I                    4209 non-null    object
 32  WORK_ZONE_TYPE                 3290 non-null    object
 33  WORKERS_PRESENT_I              1085 non-null    object
 34  NUM_UNITS                      722809 non-null  int64
 35  MOST_SEVERE_INJURY             721233 non-null  object
 36  INJURIES_TOTAL                 721244 non-null  float64
 37  INJURIES_FATAL                 721244 non-null  float64
 38  INJURIES_INCAPACITATING        721244 non-null  float64
 39  INJURIES_NON_INCAPACITATING    721244 non-null  float64
 40  INJURIES_REPORTED_NOT_EVIDENT  721244 non-null  float64
 41  INJURIES_NO_INDICATION         721244 non-null  float64
 42  INJURIES_UNKNOWN               721244 non-null  float64
 43  CRASH_HOUR                     722809 non-null  int64
 44  CRASH_DAY_OF_WEEK              722809 non-null  int64
 45  CRASH_MONTH                    722809 non-null  int64
 46  LATITUDE                       718130 non-null  float64
 47  LONGITUDE                      718130 non-null  float64
 48  LOCATION                       718130 non-null  object
dtypes: float64(11), int64(6), object(32)
memory usage: 270.2+ MB
```

[8]:
```python
# Function to display the head of our datasets
def display_data_head(people,vehicles, crashes):
    dfs = [people.head(), vehicles.head(),crashes.head()]
    df_names = ["people", "vehicles","crashes"]
    for df, name in zip(dfs, df_names):
        print(f"\n{name}:\n")
        display(df)
# Display the head of our datasets
display_data_head(people,vehicles,crashes)
```

```
people:
```

```
  PERSON_ID PERSON_TYPE                                    CRASH_RECORD_ID  \
0  O1577624      DRIVER  e8d0a18503a3ef7a69ee631eacffd421ea154ea9782131…
1  O1577610      DRIVER  0690865a402d40a7eab391f94a658b48dc03abb636a03a…
2  O1577611      DRIVER  0690865a402d40a7eab391f94a658b48dc03abb636a03a…
3  O1577604      DRIVER  fe7d5f687f472c631a7a3516d0047a3cf7a8ab2cb0b6a4…
4  O1577605      DRIVER  fe7d5f687f472c631a7a3516d0047a3cf7a8ab2cb0b6a4…

  RD_NO  VEHICLE_ID              CRASH_DATE  SEAT_NO        CITY STATE  \
0   NaN   1500926.0  05/17/2023 10:20:00 AM      NaN     CHICAGO    IL
1   NaN   1500906.0  05/17/2023 10:10:00 AM      NaN         NaN   NaN
2   NaN   1500913.0  05/17/2023 10:10:00 AM      NaN  VALPARAISO    IN
3   NaN   1500903.0  05/17/2023 10:05:00 AM      NaN     CHICAGO    IL
4   NaN   1500907.0  05/17/2023 10:05:00 AM      NaN     CHICAGO    IL

  ZIPCODE  … EMS_RUN_NO     DRIVER_ACTION DRIVER_VISION PHYSICAL_CONDITION  \
0   60637  …        NaN  IMPROPER PARKING       UNKNOWN            UNKNOWN
1     NaN  …        NaN           UNKNOWN       UNKNOWN            UNKNOWN
2   46385  …        NaN              NONE  NOT OBSCURED             NORMAL
3   60613  …        NaN           UNKNOWN       UNKNOWN             NORMAL
4   60660  …        NaN           UNKNOWN       UNKNOWN             NORMAL

  PEDPEDAL_ACTION PEDPEDAL_VISIBILITY PEDPEDAL_LOCATION        BAC_RESULT  \
0             NaN                 NaN               NaN  TEST NOT OFFERED
1             NaN                 NaN               NaN  TEST NOT OFFERED
2             NaN                 NaN               NaN  TEST NOT OFFERED
3             NaN                 NaN               NaN  TEST NOT OFFERED
4             NaN                 NaN               NaN  TEST NOT OFFERED

  BAC_RESULT VALUE CELL_PHONE_USE
0              NaN            NaN
1              NaN            NaN
2              NaN            NaN
3              NaN            NaN
4              NaN            NaN

[5 rows x 30 columns]

vehicles:

   CRASH_UNIT_ID                                   CRASH_RECORD_ID RD_NO  \
0        1577434  25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7…   NaN
1        1577435  25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7…   NaN
2        1577450  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   NaN
3        1577451  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   NaN
4        1577452  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   NaN
```

```
              CRASH_DATE  UNIT_NO UNIT_TYPE  NUM_PASSENGERS  VEHICLE_ID  \
0  05/16/2023 11:12:00 PM        1    DRIVER             NaN   1500741.0
1  05/16/2023 11:12:00 PM        2    DRIVER             1.0   1500742.0
2  05/16/2023 11:06:00 PM        1    DRIVER             NaN   1500759.0
3  05/16/2023 11:06:00 PM        2    DRIVER             NaN   1500760.0
4  05/16/2023 11:06:00 PM        3    DRIVER             NaN   1500761.0

  CMRC_VEH_I       MAKE  … TRAILER1_LENGTH TRAILER2_LENGTH  \
0        NaN      HONDA  …             NaN             NaN
1        NaN  CHEVROLET  …             NaN             NaN
2        NaN      DODGE  …             NaN             NaN
3        NaN     TOYOTA  …             NaN             NaN
4        NaN       FORD  …             NaN             NaN

   TOTAL_VEHICLE_LENGTH AXLE_CNT VEHICLE_CONFIG CARGO_BODY_TYPE LOAD_TYPE  \
0                   NaN      NaN            NaN             NaN       NaN
1                   NaN      NaN            NaN             NaN       NaN
2                   NaN      NaN            NaN             NaN       NaN
3                   NaN      NaN            NaN             NaN       NaN
4                   NaN      NaN            NaN             NaN       NaN

  HAZMAT_OUT_OF_SERVICE_I MCS_OUT_OF_SERVICE_I HAZMAT_CLASS
0                     NaN                  NaN          NaN
1                     NaN                  NaN          NaN
2                     NaN                  NaN          NaN
3                     NaN                  NaN          NaN
4                     NaN                  NaN          NaN

[5 rows x 72 columns]
```

crashes:

```
                                     CRASH_RECORD_ID RD_NO CRASH_DATE_EST_I  \
0  25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7…   NaN              NaN
1  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   NaN              NaN
2  246fea010af2010860046c6ef36efb75a8c60244088939…   NaN              NaN
3  18c220f7eeceb2cf6f9512c9b83382da28d8565fbbaaec…   NaN              NaN
4  cfecdce601503162eb09337bd6051ea358dca7294d440b…   NaN              NaN

              CRASH_DATE  POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE  \
0  05/16/2023 11:12:00 PM                  30         TRAFFIC SIGNAL
1  05/16/2023 11:06:00 PM                  30            NO CONTROLS
2  05/16/2023 11:05:00 PM                  30            NO CONTROLS
3  05/16/2023 10:20:00 PM                  25            NO CONTROLS
4  05/16/2023 09:45:00 PM                  30                UNKNOWN
```

```
      DEVICE_CONDITION WEATHER_CONDITION      LIGHTING_CONDITION  \
0  FUNCTIONING PROPERLY             CLEAR  DARKNESS, LIGHTED ROAD
1           NO CONTROLS             CLEAR  DARKNESS, LIGHTED ROAD
2           NO CONTROLS             CLEAR  DARKNESS, LIGHTED ROAD
3           NO CONTROLS             CLEAR                DARKNESS
4  FUNCTIONING PROPERLY             CLEAR                DARKNESS

      FIRST_CRASH_TYPE  … INJURIES_NON_INCAPACITATING  \
0             REAR END  …                         0.0
1        REAR TO FRONT  …                         0.0
2  PARKED MOTOR VEHICLE  …                        0.0
3         PEDALCYCLIST  …                         1.0
4             REAR END  …                         0.0

  INJURIES_REPORTED_NOT_EVIDENT INJURIES_NO_INDICATION INJURIES_UNKNOWN  \
0                           0.0                    3.0              0.0
1                           0.0                    3.0              0.0
2                           0.0                    1.0              0.0
3                           0.0                    1.0              0.0
4                           0.0                    2.0              0.0

  CRASH_HOUR CRASH_DAY_OF_WEEK CRASH_MONTH   LATITUDE  LONGITUDE  \
0         23                 3           5  41.952691 -87.807413
1         23                 3           5  41.997837 -87.688814
2         23                 3           5  42.002331 -87.695032
3         22                 3           5  41.827340 -87.636475
4         21                 3           5  41.808853 -87.640097

                                  LOCATION
0  POINT (-87.807413247555 41.952691362649)
1  POINT (-87.688813887189 41.997837266972)
2  POINT (-87.695032165757 42.002331485776)
3  POINT (-87.636475000374 41.827339537397)
4  POINT (-87.640097485203 41.808853153697)

[5 rows x 49 columns]
```

8 Data cleaning

8.0.1 Cheaking for duplicates

[9]:
```python
# A function to check for duplicates in our datasets
def check_duplicates(df):
    """
    This function checks for and returns any duplicates in a given dataframe.
    """
    duplicates = df[df.duplicated()]
```

```
    if duplicates.shape[0] == 0:
        print("No duplicates found in the dataset")
    else:
        print("Duplicates found in the dataset:")
        return duplicates
# Calling for the function to check for duplicates
check_duplicates(people)
check_duplicates(vehicles)
check_duplicates(crashes)
```

```
No duplicates found in the dataset
No duplicates found in the dataset
No duplicates found in the dataset
```

The data had no duplicates

Dropping columns that are not relevant

[10]:
```
#dropping columns either not relevant
people_drop = people[['PERSON_ID', 'RD_NO', 'CRASH_DATE', 'SEAT_NO', 'CITY',␣
 ↪'STATE', 'ZIPCODE',
       'DRIVERS_LICENSE_STATE', 'DRIVERS_LICENSE_CLASS', 'SAFETY_EQUIPMENT',␣
 ↪'AIRBAG_DEPLOYED', 'EJECTION',
       'INJURY_CLASSIFICATION', 'HOSPITAL', 'EMS_AGENCY',␣
 ↪'EMS_RUN_NO','PEDPEDAL_LOCATION']]
```

[11]:
```
#dropping columns either not relevant
vehicles_drop = vehicles[['CRASH_UNIT_ID', 'RD_NO', 'CRASH_DATE',
       'NUM_PASSENGERS', 'MAKE', 'MODEL', 'VEHICLE_YEAR', 'CMRC_VEH_I',
       'LIC_PLATE_STATE','TOWED_I', 'FIRE_I', 'OCCUPANT_CNT', 'TOWED_BY',
       'TOWED_TO', 'AREA_00_I', 'AREA_01_I', 'AREA_02_I', 'AREA_03_I',
       'AREA_04_I', 'AREA_05_I', 'AREA_06_I', 'AREA_07_I', 'AREA_08_I',
       'AREA_09_I', 'AREA_10_I', 'AREA_11_I', 'AREA_12_I', 'AREA_99_I',
       'CMV_ID', 'USDOT_NO', 'CCMC_NO', 'ILCC_NO',
       'COMMERCIAL_SRC', 'GVWR', 'CARRIER_NAME', 'CARRIER_STATE',
       'CARRIER_CITY', 'HAZMAT_PLACARDS_I', 'HAZMAT_NAME', 'UN_NO',
       'HAZMAT_PRESENT_I', 'HAZMAT_REPORT_I', 'HAZMAT_REPORT_NO',
       'MCS_REPORT_I', 'MCS_REPORT_NO', 'HAZMAT_VIO_CAUSE_CRASH_I',
       'MCS_VIO_CAUSE_CRASH_I', 'IDOT_PERMIT_NO', 'WIDE_LOAD_I',
       'TRAILER1_WIDTH', 'TRAILER2_WIDTH', 'TRAILER1_LENGTH',
       'TRAILER2_LENGTH', 'TOTAL_VEHICLE_LENGTH', 'AXLE_CNT', 'VEHICLE_CONFIG',
       'CARGO_BODY_TYPE', 'LOAD_TYPE', 'HAZMAT_OUT_OF_SERVICE_I',
       'MCS_OUT_OF_SERVICE_I', 'HAZMAT_CLASS']]
```

[12]:
```
#dropping columns either not relevant
crashes_drop = crashes[['RD_NO', 'CRASH_DATE_EST_I',␣
 ↪'CRASH_DATE','LANE_CNT','REPORT_TYPE','DATE_POLICE_NOTIFIED',
```

```
'SEC_CONTRIBUTORY_CAUSE','STREET_NO','STREET_DIRECTION',␣
↪'STREET_NAME','PHOTOS_TAKEN_I', 'STATEMENTS_TAKEN_I',
                    'BEAT_OF_OCCURRENCE', 'MOST_SEVERE_INJURY',␣
↪'INJURIES_TOTAL','INJURIES_FATAL','INJURIES_INCAPACITATING',
                    'INJURIES_NON_INCAPACITATING',␣
↪'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION',
                    'INJURIES_UNKNOWN']]
```

[13]:
```
#dropping columns that are not relevant
people.drop(columns=people_drop, inplace=True)
vehicles.drop(columns=vehicles_drop, inplace=True)
crashes.drop(columns=crashes_drop, inplace=True)
```

8.0.2 Cheaking missing values

[14]:
```
# A function to check for missing values in our dataset
def check_missing_values(data):
    # Count missing values in each column
    missing_values = data.isnull().sum()

    # Convert missing values count to percentage of total rows
    missing_percent = ((missing_values / len(data)) * 100).
↪sort_values(ascending=True)

    # Combine the missing values count and percent into a DataFrame
    missing_df = pd.concat([missing_values, missing_percent], axis=1)
    missing_df.columns = ['Missing Values', '% of Total']

    # Return only columns with missing values
    missing_df = missing_df[missing_df['Missing Values'] > 0]

    return missing_df

# Check missing values in each dataset
display(check_missing_values(people))
display(check_missing_values(vehicles))
display(check_missing_values(crashes))
```

	Missing Values	% of Total
VEHICLE_ID	30990	1.955679
SEX	25165	1.588082
AGE	461377	29.116013
DRIVER_ACTION	323450	20.411885
DRIVER_VISION	323903	20.440473
PHYSICAL_CONDITION	322571	20.356414
PEDPEDAL_ACTION	1555225	98.145229
PEDPEDAL_VISIBILITY	1555283	98.148889

```
BAC_RESULT                   322427   20.347327
BAC_RESULT VALUE            1582757   99.882685
CELL_PHONE_USE              1583458   99.926922

                      Missing Values  % of Total
UNIT_TYPE                       2008    0.136337
VEHICLE_ID                     33142    2.250247
VEHICLE_DEFECT                 33142    2.250247
VEHICLE_TYPE                   33142    2.250247
VEHICLE_USE                    33142    2.250247
TRAVEL_DIRECTION               33142    2.250247
MANEUVER                       33142    2.250247
EXCEED_SPEED_LIMIT_I         1470419   99.837251
FIRST_CONTACT_POINT            36226    2.459642

                        Missing Values  % of Total
INTERSECTION_RELATED_I          557015   77.062543
NOT_RIGHT_OF_WAY_I              689058   95.330578
HIT_AND_RUN_I                   497777   68.867017
DOORING_I                       720623   99.697569
WORK_ZONE_I                     718600   99.417688
WORK_ZONE_TYPE                  719519   99.544831
WORKERS_PRESENT_I               721724   99.849891
LATITUDE                          4679    0.647336
LONGITUDE                         4679    0.647336
LOCATION                          4679    0.647336
```

- In people data majority values are missing ie: *CELL_PHONE_USE, BAC_RESULT VALUE, PEDPEDAL_VISIBILITY and PEDPEDAL_ACTION* I will remove them
- In vehicles data majority values are missing in the *EXCEED_SPEED_LIMIT_I* so I will remove it and drop the remaing since it has about 2% of the data.
- In crahes data majority of the data is missing ie:*INTERSECTION_RELATED_I, NOT_RIGHT_OF_WAY_I, HIT_AND_RUN_I, DOORING_I, WORK_ZONE_I,WORK_ZONE_TYPE and WORKERS_PRESENT_I* so I will remove them and the remaing I will drop them

[15]:
```
# Removing colums that have large amounts of missing values
people.drop(['PEDPEDAL_ACTION','PEDPEDAL_VISIBILITY', 'BAC_RESULT VALUE',␣
 ↪'CELL_PHONE_USE'], axis=1, inplace=True)
vehicles.drop('EXCEED_SPEED_LIMIT_I', axis=1, inplace=True)
crashes.drop(labels=['INTERSECTION_RELATED_I',␣
 ↪'NOT_RIGHT_OF_WAY_I','HIT_AND_RUN_I',
                              'DOORING_I','WORK_ZONE_I','WORK_ZONE_TYPE',␣
 ↪'WORKERS_PRESENT_I'], axis=1, inplace=True)
```

[16]:
```
# Dropping the missing values
vehicles.dropna(axis=0, inplace=True)
crashes.dropna(axis=0, inplace=True)
```

Recheaking missing values have been drop

[17]:
```
# A function to check for missing values in our dataset
def check_missing_values(data):
    # Count missing values in each column
    missing_values = data.isnull().sum()

    # Convert missing values count to percentage of total rows
    missing_percent = ((missing_values / len(data)) * 100).
↪sort_values(ascending=True)

    # Combine the missing values count and percent into a DataFrame
    missing_df = pd.concat([missing_values, missing_percent], axis=1)
    missing_df.columns = ['Missing Values', '% of Total']

    # Return only columns with missing values
    missing_df = missing_df[missing_df['Missing Values'] > 0]

    return missing_df

# Check missing values in each dataset
display(check_missing_values(people))
display(check_missing_values(vehicles))
display(check_missing_values(crashes))
```

	Missing Values	% of Total
VEHICLE_ID	30990	1.955679
SEX	25165	1.588082
AGE	461377	29.116013
DRIVER_ACTION	323450	20.411885
DRIVER_VISION	323903	20.440473
PHYSICAL_CONDITION	322571	20.356414
BAC_RESULT	322427	20.347327

```
Empty DataFrame
Columns: [Missing Values, % of Total]
Index: []
```

```
Empty DataFrame
Columns: [Missing Values, % of Total]
Index: []
```

In the vehicles and crashes data do not have missing values in people it has missing values so I will check for the unique categories in each of our remainder variables.

Cheaking for unique variables

1. people

[18]:
```python
#provide an overview of the unique values and their frequencies for each column␣
↪in the DataFrame
for col in people.columns:
    print('\n' + col + '\n')
    print(people[col].value_counts())
```

```
PERSON_TYPE

DRIVER                 1233524
PASSENGER               320590
PEDESTRIAN               18190
BICYCLE                  10750
NON-MOTOR VEHICLE         1286
NON-CONTACT VEHICLE        276
Name: PERSON_TYPE, dtype: int64

CRASH_RECORD_ID

31ecf6862c691ff12d3856213b902c146b07337b42a5692e3a176a66d684d221028bb5118ef6d67a
313bcaed9e97bee1855cb1f5e8650f49e8dc17663475a1ee    61
13026c7fb51566d9ca487a093e38c6f5621c2ec25be48c306b6574983b61daeee589524b96bb2bfe
66ddd0f695c8d2bf3ab0297558528e9c7a70363c763d6bd1    50
1829f52c1281a0396ef94692331b3dc530bc4be5a54cd55e94c24a5e5e49b800fbcf9f24dabe4c82
77c8964ad05aadc89e90fd94021959d6dff5fad55480d595    46
5fd56a31e9c4608adcd9f1d504236f856a72905451941d850fb4ddf1464a44bddfcfac7ed04fee9f
fa4855bfbf07042568fd9033c3f2e48f398f7eb0002a09ab    45
c727dc759107cf17b2e8141149347128bb4bc26b026c7805562206c7c5761c543dd7cc0e47fc1137
9455a2ecbb2847c3d1744d6feb78f276d9a457e9beeb6121    45
                                                    ..
6e7087f80201ed819e60a0120bba83568f85702fd7dc71972adc264b8e157df6efdf87053e8846eb
2ca927aad1807cb7c56ea09c48bb482dd67bd303bff8a9be     1
246dd3f15c0813afa35029906e027e592f6997a08675f6077194b7ca5c2e90895d8637cabe3bca45
ad12294c463d9243ceeee4a864a8f41e0adb8c1756d074fb     1
1f662ae11036a783afd1a3c5d0a28be233895aaeffd416b4864d23a424fe22d19cb23f8da753f473
f2f150da63960b49dbb3ff5d45f40c78cb59edc0ec8d6e5c     1
70e0f8650d1d08f26f86da60da62deaa040fd5869ab42d0d98ccdface0f1bf093286a29ae73f3c9f
36c7fe970d7922d0c8a36bf4f193af1d82d76ef49f8653ff     1
d0906381565cf7d51d5c0537fc254d957c65328c48367d97f5bfc4e0dd8803d498ce121d4549e437
496276539d09637ab68c86ea530ac75ac4d200a2e32541a1     1
Name: CRASH_RECORD_ID, Length: 721315, dtype: int64

VEHICLE_ID

332155.0     60
643997.0     47
1481124.0    44
```

```
366311.0     44
162199.0     44
             ..
949494.0      1
950309.0      1
949525.0      1
949550.0      1
739508.0      1
Name: VEHICLE_ID, Length: 1248684, dtype: int64
```

SEX

```
M    819613
F    596850
X    142988
Name: SEX, dtype: int64
```

AGE

```
 25.0     31802
 27.0     31709
 26.0     31684
 28.0     31007
 24.0     30716
          …
-49.0         1
-177.0        1
-47.0         1
-40.0         1
 106.0        1
Name: AGE, Length: 116, dtype: int64
```

DRIVER_ACTION

```
NONE                                    453928
UNKNOWN                                 313404
FAILED TO YIELD                         114746
OTHER                                   111805
FOLLOWED TOO CLOSELY                     76517
IMPROPER BACKING                         38166
IMPROPER TURN                            33039
IMPROPER LANE CHANGE                     32513
IMPROPER PASSING                         28037
DISREGARDED CONTROL DEVICES              21954
TOO FAST FOR CONDITIONS                  19502
WRONG WAY/SIDE                            4978
IMPROPER PARKING                          4706
OVERCORRECTED                             2088
```

```
EVADING POLICE VEHICLE                 2000
CELL PHONE USE OTHER THAN TEXTING      1913
EMERGENCY VEHICLE ON CALL              1149
TEXTING                                 518
STOPPED SCHOOL BUS                      150
LICENSE RESTRICTIONS                     53
Name: DRIVER_ACTION, dtype: int64

DRIVER_VISION

NOT OBSCURED            651081
UNKNOWN                 578481
OTHER                    12766
MOVING VEHICLES           7373
PARKED VEHICLES           4531
WINDSHIELD (WATER/ICE)    3555
BLINDED - SUNLIGHT        1508
TREES, PLANTS              539
BUILDINGS                  458
BLINDED - HEADLIGHTS       127
HILLCREST                   94
BLOWING MATERIALS           89
EMBANKMENT                  78
SIGNBOARD                   33
Name: DRIVER_VISION, dtype: int64

PHYSICAL_CONDITION

NORMAL                          828169
UNKNOWN                         410404
IMPAIRED - ALCOHOL                5620
REMOVED BY EMS                    4657
OTHER                             3698
FATIGUED/ASLEEP                   3368
EMOTIONAL                         2826
ILLNESS/FAINTED                   1183
HAD BEEN DRINKING                  953
IMPAIRED - DRUGS                   664
IMPAIRED - ALCOHOL AND DRUGS       341
MEDICATED                          162
Name: PHYSICAL_CONDITION, dtype: int64

BAC_RESULT

TEST NOT OFFERED                    1243396
TEST REFUSED                          13377
TEST PERFORMED, RESULTS UNKNOWN        3100
TEST TAKEN                             2316
```

```
Name: BAC_RESULT, dtype: int64
```

Variables 'Unable to determine' and 'Not applicable' does not add any value to our project. We will remove these.

- We are only intersted in the DRIVER for PERSON_TYPE. We will slice the rest out of the dataset.
- 'SEX - X' and negative 'AGE' values does not tell much about the person. We will slice them out of the dataset.
- DRIVER_ACTION seems more like a target variable rather than a predictor. We will drop this column.
- BAC_RESULT has mostly TEST NOT OFFERED. the rest of the data will not be able to add much weight for this predictor. We will drop this as well.

[19]:
```
people.drop('BAC_RESULT', axis=1, inplace=True)
```

[20]:
```
#slice the PERSON_TYPE, AGE and SEX columns.
people_fin = people[(people['PERSON_TYPE'] == 'DRIVER') & (people['SEX'] !=␣
↪'X') & (people['AGE'] > 0)].copy()
```

[21]:
```
people_fin.head()
```

[21]:
```
  PERSON_TYPE                                    CRASH_RECORD_ID  VEHICLE_ID  \
0      DRIVER  e8d0a18503a3ef7a69ee631eacffd421ea154ea9782131…   1500926.0
2      DRIVER  0690865a402d40a7eab391f94a658b48dc03abb636a03a…   1500913.0
3      DRIVER  fe7d5f687f472c631a7a3516d0047a3cf7a8ab2cb0b6a4…   1500903.0
4      DRIVER  fe7d5f687f472c631a7a3516d0047a3cf7a8ab2cb0b6a4…   1500907.0
7      DRIVER  439723dae3207b29a5fac90ce3efcb21cde0fe63e49350…   1500872.0

  SEX   AGE         DRIVER_ACTION DRIVER_VISION PHYSICAL_CONDITION
0   F  57.0      IMPROPER PARKING       UNKNOWN            UNKNOWN
2   F  57.0                  NONE  NOT OBSCURED             NORMAL
3   M  74.0               UNKNOWN       UNKNOWN             NORMAL
4   M  33.0               UNKNOWN       UNKNOWN             NORMAL
7   F  39.0  FOLLOWED TOO CLOSELY       UNKNOWN             NORMAL
```

[22]:
```
people_fin.shape
```

[22]:
```
(892110, 8)
```

```
2. vehicles
```

[23]:
```
#provide an overview of the unique values and their frequencies for each column␣
↪in the DataFrame
for col in vehicles.columns[1:]:
    print('\n' + col + '\n')
    print(vehicles[col].value_counts())
```

```
UNIT_NO

1          717525
2          657433
3           47733
4            9880
5            2585
6             820
7             307
8             137
9              65
10             34
11             19
12             12
13              7
14              6
15              5
16              4
0               4
17              3
18              3
3778035         1
Name: UNIT_NO, dtype: int64

UNIT_TYPE

DRIVER                  1230227
PARKED                   193434
DRIVERLESS                12614
DISABLED VEHICLE            194
NON-CONTACT VEHICLE         114
Name: UNIT_TYPE, dtype: int64

VEHICLE_ID

1500741.0    1
498046.0     1
497916.0     1
498593.0     1
498591.0     1
            ..
998498.0     1
998496.0     1
998392.0     1
998377.0     1
567870.0     1
Name: VEHICLE_ID, Length: 1436583, dtype: int64
```

```
VEHICLE_DEFECT

NONE                792628
UNKNOWN             629288
OTHER                 7301
BRAKES                4682
TIRES                  719
STEERING               670
WHEELS                 366
SUSPENSION             246
ENGINE/MOTOR           192
FUEL SYSTEM            172
LIGHTS                  90
WINDOWS                 86
CARGO                   46
SIGNALS                 39
RESTRAINT SYSTEM        21
TRAILER COUPLING        19
EXHAUST                 18
Name: VEHICLE_DEFECT, dtype: int64

VEHICLE_TYPE

PASSENGER                                 904181
SPORT UTILITY VEHICLE (SUV)               194473
UNKNOWN/NA                                135892
VAN/MINI-VAN                               69525
PICKUP                                     45243
TRUCK - SINGLE UNIT                        27111
OTHER                                      16921
BUS OVER 15 PASS.                          15135
TRACTOR W/ SEMI-TRAILER                    13506
BUS UP TO 15 PASS.                          3721
MOTORCYCLE (OVER 150CC)                     3271
SINGLE UNIT TRUCK WITH TRAILER              2226
OTHER VEHICLE WITH TRAILER                  1867
TRACTOR W/O SEMI-TRAILER                    1815
AUTOCYCLE                                    651
MOPED OR MOTORIZED BICYCLE                   406
MOTOR DRIVEN CYCLE                           325
ALL-TERRAIN VEHICLE (ATV)                    162
FARM EQUIPMENT                                75
3-WHEELED MOTORCYCLE (2 REAR WHEELS)          50
RECREATIONAL OFF-HIGHWAY VEHICLE (ROV)        19
SNOWMOBILE                                     8
Name: VEHICLE_TYPE, dtype: int64

VEHICLE_USE
```

```
PERSONAL                          928310
UNKNOWN/NA                        291176
NOT IN USE                         74689
OTHER                              44891
TAXI/FOR HIRE                      18880
COMMERCIAL - SINGLE UNIT           17517
RIDESHARE SERVICE                  11997
CTA                                 9763
POLICE                              9360
CONSTRUCTION/MAINTENANCE            6467
COMMERCIAL - MULTI-UNIT             5819
OTHER TRANSIT                       3964
SCHOOL BUS                          3872
TOW TRUCK                           2752
AMBULANCE                           1694
FIRE                                1402
STATE OWNED                         1200
DRIVER EDUCATION                    1140
MASS TRANSIT                         832
LAWN CARE/LANDSCAPING                533
AGRICULTURE                          151
CAMPER/RV - SINGLE UNIT               78
MILITARY                              55
HOUSE TRAILER                         23
CAMPER/RV - TOWED/MULTI-UNIT          18
Name: VEHICLE_USE, dtype: int64
```

```
TRAVEL_DIRECTION
```

```
N          338187
S          330540
W          299337
E          293024
UNKNOWN    113540
SE          18112
NW          16560
SW          13693
NE          13590
Name: TRAVEL_DIRECTION, dtype: int64
```

```
MANEUVER
```

```
STRAIGHT AHEAD                          655794
PARKED                                  198043
UNKNOWN/NA                              112104
SLOW/STOP IN TRAFFIC                    109132
TURNING LEFT                             84673
```

```
BACKING                               59023
TURNING RIGHT                         47093
PASSING/OVERTAKING                    34593
CHANGING LANES                        27788
OTHER                                 23927
ENTERING TRAFFIC LANE FROM PARKING    16955
MERGING                                9985
STARTING IN TRAFFIC                    8406
U-TURN                                 8110
LEAVING TRAFFIC LANE TO PARK           6940
AVOIDING VEHICLES/OBJECTS              6202
SKIDDING/CONTROL LOSS                  5777
ENTER FROM DRIVE/ALLEY                 5534
PARKED IN TRAFFIC LANE                 4348
SLOW/STOP - LEFT TURN                  3042
DRIVING WRONG WAY                      2104
SLOW/STOP - RIGHT TURN                 1927
NEGOTIATING A CURVE                    1901
SLOW/STOP - LOAD/UNLOAD                1663
TURNING ON RED                          558
DRIVERLESS                              544
DIVERGING                               222
DISABLED                                195
Name: MANEUVER, dtype: int64

FIRST_CONTACT_POINT

FRONT                 280377
REAR                  189471
UNKNOWN               137428
SIDE-LEFT              98025
SIDE-RIGHT             93042
FRONT-LEFT             81521
FRONT-LEFT-CORNER      79328
FRONT-RIGHT-CORNER     77441
FRONT-RIGHT            76901
REAR-LEFT              68076
OTHER                  39701
REAR-RIGHT             36487
REAR-LEFT-CORNER       35377
TOTAL (ALL AREAS)      26362
REAR-RIGHT-CORNER      25849
SIDE-LEFT-REAR         20584
SIDE-RIGHT-REAR        15419
SIDE-LEFT-FRONT        13121
ROOF                   11947
NONE                   11836
SIDE-RIGHT-FRONT       11262
```

```
UNDER CARRIAGE           5442
TOP                      1586
Name: FIRST_CONTACT_POINT, dtype: int64
```

- We will use 'VEHICLE_ID' to merge this dataset with the People dataset. We will drop 'CRASH_RECORD_ID'.
- UNIT_NO is the number of parties involved. We are not concerned with the count so we will drop this.
- UNIT_TYPE seems to be also irrelevant for our EDA. We will drop this.
- VEHICLE_DEFECT has about 90% of the data either as NONE or UNKNOWN. We will drop this as well.
- VEHICLE_TYPE is spread pretty thinly outside of PASSENGER and SUV. We will drop this column as well.
- VEHICLE_USE has majority PERSONAL and UNKNOWN. We will drop this.
- TRAVEL_DIRECTION does not add much to the target variable. We will drop this.
- FIRST_CONTACT_POINT also does not lead to the target variable. We will drop this.

[24]:
```
vehicles_del= ['CRASH_RECORD_ID', 'UNIT_NO', 'UNIT_TYPE', 'VEHICLE_DEFECT',
                           'VEHICLE_TYPE',␣
 ↪'VEHICLE_USE','TRAVEL_DIRECTION','FIRST_CONTACT_POINT']
```

[25]:
```
vehicles.drop(vehicles_del, axis=1, inplace=True)
```

[26]:
```
vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436583 entries, 0 to 1472815
Data columns (total 2 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   VEHICLE_ID  1436583 non-null  float64
 1   MANEUVER    1436583 non-null  object
dtypes: float64(1), object(1)
memory usage: 32.9+ MB
```

Manuever seems to be the only column with usuable information from the vehicle dataset

```
 3. crashes
```

preditors

[27]:
```
# provide an overview of the unique values and their frequencies for each␣
 ↪column in the DataFrame
for col in crashes[1:]:
    print('\n' + col + '\n')
    print(crashes[col].value_counts())
```

```
CRASH_RECORD_ID
```

```
25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7e998bb71e476a95e2cb27abd1cef40a8ef
d9ec4929c34da8f7f5403333b420bf4ca753bf77fd8417fb     1
f6c245fb13de9caad165e3e1b47f7cf983a5d2cd20778c10e27f32bc8397b8b35519328803bc3124
0c1ed821e967f06c7da57a1a6819c7b36ffc0f5edefa89d8     1
27372c090bbfe8d0d75174a1f8bd4f671cd23c7d3adb432f9a23d536ba39acdba613aac4ab1bed63
d6be672c2754c04b274fa9f3b74a816c43c62e4335358c4c     1
1eb85c55fa0a4aa0d786373f71c41ea3018e1ecd7e4cfdc4f17a23b54d6a4f7f60918311726e851b
63a91aa35b062e9f9f58579c4f7da8a73cd1f7a61abdd6b3     1
d25134adc4da9eb829478249bb81e772c2518e6af1259596608e7c383698066f89f0c6cfb2a50069
04581d783c883ba098632cedd9b33a1de4b84c6dddde8b5e     1
                                                    ..
fe508a7f777e72abe27dc44523631785c8b154d0c1f64134442d786d1a52c7cce1e53992d8b653b6
538ecce2b3b8d6f68642b1c8c417b984ad21886cd992ad89     1
7487a5867f896b21bc590f951152cb4490ea78b8d6c93a5788b93470a1d1db5be34d0f708c48f6ae
9361b9bfbb16252fea1111e29cd96b73c3d85e5c26fd18ce     1
805fc4a1098a75409f274ca9da96d205e5bc46e6fbd9c4344334716257e944371cd8073f3f412146
8ef75cc41bcebf07e5cc65d352b96a6fbd0e1cdc4d7a63d2     1
0cf0ddd3b5f04310fa83ee31efe6e294073182f877882552238c3fd8d84f4fd821aed2cdb50b46da
b2be7059c1decf79fed661b1b91b722d950a3af705750779     1
a802658be15312809c771559e4f81088cfb226830792a50470f4ecf9dbdc4fd83c1e187199279ea5
3e604a6cc30bc0c0fd5ba00b0c0e924746c0f4a23b44edc5     1
Name: CRASH_RECORD_ID, Length: 718130, dtype: int64

POSTED_SPEED_LIMIT

30    529461
35     48155
25     45024
20     29451
15     25146
10     16404
0       7260
40      6937
45      4524
5       4356
55       580
50       171
3        163
9         95
39        74
99        66
1         39
60        38
24        36
2         26
32        17
34        14
65        14
```

```
33        13
11        11
6          7
26         6
7          5
36         5
70         4
12         3
14         3
29         3
4          2
38         2
22         2
23         2
31         2
18         2
8          2
44         1
62         1
49         1
16         1
63         1
Name: POSTED_SPEED_LIMIT, dtype: int64
```

TRAFFIC_CONTROL_DEVICE

```
NO CONTROLS                 410639
TRAFFIC SIGNAL              199033
STOP SIGN/FLASHER            71454
UNKNOWN                      26603
OTHER                         4625
LANE USE MARKING              1173
YIELD                         1014
OTHER REG. SIGN                735
OTHER WARNING SIGN             582
RAILROAD CROSSING GATE         467
PEDESTRIAN CROSSING SIGN       416
DELINEATORS                    278
SCHOOL ZONE                    278
FLASHING CONTROL SIGNAL        254
POLICE/FLAGMAN                 240
OTHER RAILROAD CROSSING        165
RR CROSSING SIGN               113
NO PASSING                      41
BICYCLE CROSSING SIGN           20
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64
```

DEVICE_CONDITION

```
NO CONTROLS                 415554
FUNCTIONING PROPERLY        247015
UNKNOWN                      44104
OTHER                         5481
FUNCTIONING IMPROPERLY        3447
NOT FUNCTIONING               2185
WORN REFLECTIVE MATERIAL       262
MISSING                         82
Name: DEVICE_CONDITION, dtype: int64

WEATHER_CONDITION

CLEAR                       565028
RAIN                         62200
UNKNOWN                      37202
SNOW                         26272
CLOUDY/OVERCAST              21315
OTHER                         2280
FREEZING RAIN/DRIZZLE         1343
FOG/SMOKE/HAZE                1039
SLEET/HAIL                     928
BLOWING SNOW                   377
SEVERE CROSS WIND GATE         141
BLOWING SAND, SOIL, DIRT         5
Name: WEATHER_CONDITION, dtype: int64

LIGHTING_CONDITION

DAYLIGHT                  461609
DARKNESS, LIGHTED ROAD    158409
DARKNESS                   34381
UNKNOWN                    30873
DUSK                       20897
DAWN                       11961
Name: LIGHTING_CONDITION, dtype: int64

FIRST_CRASH_TYPE

PARKED MOTOR VEHICLE            167732
REAR END                        162911
SIDESWIPE SAME DIRECTION        108644
TURNING                         101973
ANGLE                            77877
FIXED OBJECT                     33895
PEDESTRIAN                       16485
PEDALCYCLIST                     10464
SIDESWIPE OPPOSITE DIRECTION     10236
```

```
OTHER OBJECT                     7108
REAR TO FRONT                    6278
HEAD ON                          6153
REAR TO SIDE                     3736
OTHER NONCOLLISION               2324
REAR TO REAR                     1321
ANIMAL                            508
OVERTURNED                        446
TRAIN                              39
Name: FIRST_CRASH_TYPE, dtype: int64

TRAFFICWAY_TYPE

NOT DIVIDED                         315998
DIVIDED - W/MEDIAN (NOT RAISED)     117894
ONE-WAY                              92191
PARKING LOT                          48957
DIVIDED - W/MEDIAN BARRIER           41339
FOUR WAY                             39211
OTHER                                19457
ALLEY                                11941
UNKNOWN                               8124
T-INTERSECTION                        8007
CENTER TURN LANE                      5340
DRIVEWAY                              2350
RAMP                                  2150
UNKNOWN INTERSECTION TYPE             1952
FIVE POINT, OR MORE                    882
Y-INTERSECTION                         864
TRAFFIC ROUTE                          713
NOT REPORTED                           423
ROUNDABOUT                             205
L-INTERSECTION                         132
Name: TRAFFICWAY_TYPE, dtype: int64

ALIGNMENT

STRAIGHT AND LEVEL       700534
STRAIGHT ON GRADE          9185
CURVE, LEVEL               5130
STRAIGHT ON HILLCREST      1958
CURVE ON GRADE              995
CURVE ON HILLCREST          328
Name: ALIGNMENT, dtype: int64

ROADWAY_SURFACE_COND

DRY                 531911
```

```
WET                  95865
UNKNOWN              56779
SNOW OR SLUSH        26301
ICE                   5178
OTHER                 1815
SAND, MUD, DIRT        281
Name: ROADWAY_SURFACE_COND, dtype: int64

ROAD_DEFECT

NO DEFECTS           584232
UNKNOWN              119266
RUT, HOLES             5751
OTHER                  3991
WORN SURFACE           2954
SHOULDER DEFECT        1378
DEBRIS ON ROADWAY       558
Name: ROAD_DEFECT, dtype: int64

CRASH_TYPE

NO INJURY / DRIVE AWAY              528457
INJURY AND / OR TOW DUE TO CRASH    189673
Name: CRASH_TYPE, dtype: int64

DAMAGE

OVER $1,500      438210
$501 - $1,500    195974
$500 OR LESS      83946
Name: DAMAGE, dtype: int64

PRIM_CONTRIBUTORY_CAUSE

UNABLE TO DETERMINE
277139
FAILING TO YIELD RIGHT-OF-WAY
78508
FOLLOWING TOO CLOSELY
71129
NOT APPLICABLE
37862
IMPROPER OVERTAKING/PASSING
34895
FAILING TO REDUCE SPEED TO AVOID CRASH
30563
IMPROPER BACKING
28941
```

IMPROPER LANE USAGE
26124
IMPROPER TURNING/NO SIGNAL
23738
DRIVING SKILLS/KNOWLEDGE/EXPERIENCE
23470
DISREGARDING TRAFFIC SIGNALS
13904
WEATHER
11294
OPERATING VEHICLE IN ERRATIC, RECKLESS, CARELESS, NEGLIGENT OR AGGRESSIVE MANNER
9148
DISREGARDING STOP SIGN
7941
DISTRACTION - FROM INSIDE VEHICLE
4981
EQUIPMENT - VEHICLE CONDITION
4534
PHYSICAL CONDITION OF DRIVER
4371
VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)
4106
DRIVING ON WRONG SIDE/WRONG WAY
3752
UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)
3508
DISTRACTION - FROM OUTSIDE VEHICLE
3028
EXCEEDING AUTHORIZED SPEED LIMIT
1967
ROAD ENGINEERING/SURFACE/MARKING DEFECTS
1866
EXCEEDING SAFE SPEED FOR CONDITIONS
1674
ROAD CONSTRUCTION/MAINTENANCE
1599
DISREGARDING OTHER TRAFFIC SIGNS
1548
EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST
1347
CELL PHONE USE OTHER THAN TEXTING
973
DISREGARDING ROAD MARKINGS
902
HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)
760
ANIMAL
606

```
TURNING RIGHT ON RED
509
DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.)
336
RELATED TO BUS STOP
315
TEXTING
291
DISREGARDING YIELD SIGN
251
PASSING STOPPED SCHOOL BUS
88
BICYCLE ADVANCING LEGALLY ON RED LIGHT
75
OBSTRUCTED CROSSWALKS
67
MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT
20
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64

NUM_UNITS

2     627995
1      39874
3      39790
4       7746
5       1858
6        540
7        180
8         78
9         33
10        16
11         7
12         5
18         3
14         2
13         1
15         1
16         1
Name: NUM_UNITS, dtype: int64

CRASH_HOUR

15    55274
16    54844
17    53547
14    48270
18    44235
```

```
13    43853
12    42376
8     37875
11    36611
9     33040
10    32640
19    32539
7     30265
20    26283
21    23466
22    21487
23    18641
6     15519
0     15474
1     13186
2     11373
5      9806
3      9263
4      8263
Name: CRASH_HOUR, dtype: int64

CRASH_DAY_OF_WEEK

6    116893
7    106525
5    103058
3    102337
4    101550
2     98944
1     88823
Name: CRASH_DAY_OF_WEEK, dtype: int64

CRASH_MONTH

10    66291
9     62380
12    61300
5     60947
8     60757
11    60092
7     59040
3     58478
6     57578
1     57437
2     57065
4     56765
Name: CRASH_MONTH, dtype: int64
```

```
LATITUDE

41.976201    1146
41.900959     670
41.791420     521
41.751461     503
41.722257     397
             …
41.909120       1
41.776586       1
41.784034       1
41.891171       1
41.868220       1
Name: LATITUDE, Length: 277256, dtype: int64

LONGITUDE

-87.905309    1146
-87.619928     670
-87.580148     521
-87.585972     503
-87.585276     397
              …
-87.586279       1
-87.721550       1
-87.765236       1
-87.653917       1
-87.708311       1
Name: LONGITUDE, Length: 277227, dtype: int64

LOCATION

POINT (-87.905309125103 41.976201139024)    1146
POINT (-87.619928173678 41.900958919109)     670
POINT (-87.580147768689 41.791420282098)     521
POINT (-87.585971992965 41.751460603167)     503
POINT (-87.585275565077 41.722257273006)     397
                                            …
POINT (-87.706015088063 41.824268608978)       1
POINT (-87.626643215394 41.767465371623)       1
POINT (-87.586278704646 41.758654278901)       1
POINT (-87.721550137644 41.742757072518)       1
POINT (-87.708310986354 41.86822049587)        1
Name: LOCATION, Length: 277408, dtype: int64
```

[28]:

```
crashes.shape
```

[28]: (718130, 21)

We will use 'PRIM_CONTRIBUTORY_CAUSE' as our target variables.

Target variable

[29]:
```
crashes['PRIM_CONTRIBUTORY_CAUSE'].value_counts()
```

[29]:
```
UNABLE TO DETERMINE
277139
FAILING TO YIELD RIGHT-OF-WAY
78508
FOLLOWING TOO CLOSELY
71129
NOT APPLICABLE
37862
IMPROPER OVERTAKING/PASSING
34895
FAILING TO REDUCE SPEED TO AVOID CRASH
30563
IMPROPER BACKING
28941
IMPROPER LANE USAGE
26124
IMPROPER TURNING/NO SIGNAL
23738
DRIVING SKILLS/KNOWLEDGE/EXPERIENCE
23470
DISREGARDING TRAFFIC SIGNALS
13904
WEATHER
11294
OPERATING VEHICLE IN ERRATIC, RECKLESS, CARELESS, NEGLIGENT OR AGGRESSIVE MANNER
9148
DISREGARDING STOP SIGN
7941
DISTRACTION - FROM INSIDE VEHICLE
4981
EQUIPMENT - VEHICLE CONDITION
4534
PHYSICAL CONDITION OF DRIVER
4371
VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)
4106
DRIVING ON WRONG SIDE/WRONG WAY
3752
UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)
3508
```

```
DISTRACTION - FROM OUTSIDE VEHICLE
3028
EXCEEDING AUTHORIZED SPEED LIMIT
1967
ROAD ENGINEERING/SURFACE/MARKING DEFECTS
1866
EXCEEDING SAFE SPEED FOR CONDITIONS
1674
ROAD CONSTRUCTION/MAINTENANCE
1599
DISREGARDING OTHER TRAFFIC SIGNS
1548
EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST
1347
CELL PHONE USE OTHER THAN TEXTING
973
DISREGARDING ROAD MARKINGS
902
HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)
760
ANIMAL
606
TURNING RIGHT ON RED
509
DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.)
336
RELATED TO BUS STOP
315
TEXTING
291
DISREGARDING YIELD SIGN
251
PASSING STOPPED SCHOOL BUS
88
BICYCLE ADVANCING LEGALLY ON RED LIGHT
75
OBSTRUCTED CROSSWALKS
67
MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT
20
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64
```

[30]:
```
## Filter crashes based on specific conditions
crashes_fin=crashes[(crashes.PRIM_CONTRIBUTORY_CAUSE != 'UNABLE TO DETERMINE') &
                                    (crashes.PRIM_CONTRIBUTORY_CAUSE != 'NOT␣
 ↪APPLICABLE')].copy()
```

[31]:
```
# cheaking the shape of our dataset
crashes_fin.shape
```

[31]: (403129, 21)

8.0.3 merging the dataset

[32]:
```
#Merging Datasets on the Crash_Record_ID Column
df_1 = pd.merge(people_fin, vehicles, on='VEHICLE_ID')
merg_data = pd.merge(df_1, crashes_fin, on='CRASH_RECORD_ID')
merg_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 577429 entries, 0 to 577428
Data columns (total 29 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   PERSON_TYPE             577429 non-null  object
 1   CRASH_RECORD_ID         577429 non-null  object
 2   VEHICLE_ID              577429 non-null  float64
 3   SEX                     577429 non-null  object
 4   AGE                     577429 non-null  float64
 5   DRIVER_ACTION           577429 non-null  object
 6   DRIVER_VISION           577429 non-null  object
 7   PHYSICAL_CONDITION      577429 non-null  object
 8   MANEUVER                577429 non-null  object
 9   POSTED_SPEED_LIMIT      577429 non-null  int64
 10  TRAFFIC_CONTROL_DEVICE  577429 non-null  object
 11  DEVICE_CONDITION        577429 non-null  object
 12  WEATHER_CONDITION       577429 non-null  object
 13  LIGHTING_CONDITION      577429 non-null  object
 14  FIRST_CRASH_TYPE        577429 non-null  object
 15  TRAFFICWAY_TYPE         577429 non-null  object
 16  ALIGNMENT               577429 non-null  object
 17  ROADWAY_SURFACE_COND    577429 non-null  object
 18  ROAD_DEFECT             577429 non-null  object
 19  CRASH_TYPE              577429 non-null  object
 20  DAMAGE                  577429 non-null  object
 21  PRIM_CONTRIBUTORY_CAUSE 577429 non-null  object
 22  NUM_UNITS               577429 non-null  int64
 23  CRASH_HOUR              577429 non-null  int64
 24  CRASH_DAY_OF_WEEK       577429 non-null  int64
 25  CRASH_MONTH             577429 non-null  int64
 26  LATITUDE                577429 non-null  float64
 27  LONGITUDE               577429 non-null  float64
 28  LOCATION                577429 non-null  object
dtypes: float64(4), int64(5), object(20)
```

```
memory usage: 132.2+ MB
```

[33]:
```
# Cheaking the shape of the data
merg_data.shape
```

[33]:
```
(577429, 29)
```

[34]:
```
# cheaking the head of the data
merg_data.head()
```

[34]:
```
  PERSON_TYPE                                   CRASH_RECORD_ID  VEHICLE_ID  \
0      DRIVER  25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7…   1500741.0
1      DRIVER  25d92973475a04a93e7fd206fbfce57e8a9a1e25cc85a7…   1500742.0
2      DRIVER  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   1500759.0
3      DRIVER  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   1500760.0
4      DRIVER  375ac7f6fcb4ef73d728edc52ed556f23fd465a351833f…   1500761.0

  SEX   AGE         DRIVER_ACTION DRIVER_VISION  PHYSICAL_CONDITION  \
0   F  20.0  FOLLOWED TOO CLOSELY       UNKNOWN              NORMAL
1   F  53.0               UNKNOWN       UNKNOWN              NORMAL
2   M  22.0      IMPROPER BACKING  NOT OBSCURED  IMPAIRED - ALCOHOL
3   M  67.0                  NONE  NOT OBSCURED              NORMAL
4   M  54.0                  NONE  NOT OBSCURED              NORMAL

         MANEUVER  POSTED_SPEED_LIMIT  …              CRASH_TYPE  \
0  STRAIGHT AHEAD                  30  …  NO INJURY / DRIVE AWAY
1  STRAIGHT AHEAD                  30  …  NO INJURY / DRIVE AWAY
2         BACKING                  30  …  NO INJURY / DRIVE AWAY
3  STRAIGHT AHEAD                  30  …  NO INJURY / DRIVE AWAY
4  STRAIGHT AHEAD                  30  …  NO INJURY / DRIVE AWAY

          DAMAGE                            PRIM_CONTRIBUTORY_CAUSE NUM_UNITS  \
0  $501 - $1,500                               FOLLOWING TOO CLOSELY         2
1  $501 - $1,500                               FOLLOWING TOO CLOSELY         2
2  $501 - $1,500  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         3
3  $501 - $1,500  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         3
4  $501 - $1,500  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         3

  CRASH_HOUR CRASH_DAY_OF_WEEK CRASH_MONTH   LATITUDE  LONGITUDE  \
0         23                 3           5  41.952691 -87.807413
1         23                 3           5  41.952691 -87.807413
2         23                 3           5  41.997837 -87.688814
3         23                 3           5  41.997837 -87.688814
4         23                 3           5  41.997837 -87.688814

                                  LOCATION
0  POINT (-87.807413247555 41.952691362649)
```

```
1  POINT (-87.807413247555 41.952691362649)
2  POINT (-87.688813887189 41.997837266972)
3  POINT (-87.688813887189 41.997837266972)
4  POINT (-87.688813887189 41.997837266972)

[5 rows x 29 columns]
```

[35]:
```
#cheaking columns of our merged dataset
merg_data.columns
```

[35]:
```
Index(['PERSON_TYPE', 'CRASH_RECORD_ID', 'VEHICLE_ID', 'SEX', 'AGE',
       'DRIVER_ACTION', 'DRIVER_VISION', 'PHYSICAL_CONDITION', 'MANEUVER',
       'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION',
       'WEATHER_CONDITION', 'LIGHTING_CONDITION', 'FIRST_CRASH_TYPE',
       'TRAFFICWAY_TYPE', 'ALIGNMENT', 'ROADWAY_SURFACE_COND', 'ROAD_DEFECT',
       'CRASH_TYPE', 'DAMAGE', 'PRIM_CONTRIBUTORY_CAUSE', 'NUM_UNITS',
       'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONTH', 'LATITUDE',
       'LONGITUDE', 'LOCATION'],
      dtype='object')
```

[36]:
```
# check the unique variables for each category.
for col in merg_data.columns:
    print('\n' + col + '\n')
    print(merg_data[col].value_counts())
```

```
PERSON_TYPE

DRIVER    577429
Name: PERSON_TYPE, dtype: int64

CRASH_RECORD_ID

c9d233e31a4f2a07733ef75f0404e75c360b30c7ee9bc45076938dc80c375578c1468bc096ecb773
d2bfc71270d746d95f416a5bd6b15fbcf8707b1748693722    12
196e0d42ec2dd3c503f98ad28d08067091e9801170ce6b264599642baf11c87f2064fdeccff3cd9d
ed1c8d7bf2329640af1e4730adfcb3a36127f5245ec7e152     9
7be311dead41c5337cbd12d40bb7be93c505303d6f1cf92e72a2b7c695ae95b472a66d9b3a6b505a
0e4c2279d53acf3b6115320fcafb54d8ee1aa3d0c811e3a0     9
2294d7387552dd1804e1eddde6b4ce561209f90ede1fb98805a0658ad6fcb9f8d3f8846416358dbf
8b9e71db2095d505387d2ce4ad894ad9a5a9ea5ae691abaa     9
d3c41d043f9f56c4ab63d2e0e6d229bd1283ffd5e600b0f5518e0493e41415c72b369d05f4bb9804
c784b7e5789a3e84ae0d600300234f496a00c0a43a8e8d75     8
                                                    ..
45e733513cc4265f69b45a19840c6391430fe641d3bccd344a69b1c78d84978418de9d57e93d16bb
eaeb4e5aeeb09d3437036741c60da9c446cbd1dc02179c31     1
9174e4112627175c3ef23f38bb488b99282fe07ae0c48ee795a998a7779388ae0f4d18d08b71d258
```

```
7e0035ae2dd6ae4fab6f575500a0e350fc29107d1ec68c5d      1
0601e8309277db928d5579ecab19a9c9856ebeba2024986c91ceb9906487f7722039e60da9734cc5
0b6ae7639ecd4dcfce1f7bd780c6a223ba9aaf039b1730ee      1
17520e7ada1c93b9d5eeda92e555fa10d90afa78b5e0e6f4a03e57179ef5933b2e92c174dfa7f12f
8fa6e89543883f9ae7bc8a018fbd8bd76e8ef69ca815c0b0      1
a802658be15312809c771559e4f81088cfb226830792a50470f4ecf9dbdc4fd83c1e187199279ea5
3e604a6cc30bc0c0fd5ba00b0c0e924746c0f4a23b44edc5      1
Name: CRASH_RECORD_ID, Length: 360447, dtype: int64

VEHICLE_ID

1500741.0    1
469979.0     1
470005.0     1
470017.0     1
470032.0     1
            ..
1043136.0    1
1043135.0    1
953192.0     1
953184.0     1
567870.0     1
Name: VEHICLE_ID, Length: 577429, dtype: int64

SEX

M    345523
F    231906
Name: SEX, dtype: int64

AGE

27.0     17531
25.0     17469
26.0     17259
28.0     17065
24.0     16647
         …
110.0        2
108.0        2
109.0        1
104.0        1
107.0        1
Name: AGE, Length: 108, dtype: int64

DRIVER_ACTION

NONE                                 269888
```

```
FAILED TO YIELD                            69467
OTHER                                      51970
FOLLOWED TOO CLOSELY                       47494
UNKNOWN                                    38207
IMPROPER TURN                              19894
IMPROPER BACKING                           18826
IMPROPER LANE CHANGE                       17623
DISREGARDED CONTROL DEVICES                12344
IMPROPER PASSING                           12328
TOO FAST FOR CONDITIONS                    11212
WRONG WAY/SIDE                              2334
IMPROPER PARKING                            1683
CELL PHONE USE OTHER THAN TEXTING           1094
EVADING POLICE VEHICLE                      1010
OVERCORRECTED                                993
EMERGENCY VEHICLE ON CALL                    623
TEXTING                                      305
STOPPED SCHOOL BUS                            98
LICENSE RESTRICTIONS                          36
Name: DRIVER_ACTION, dtype: int64
```

```
DRIVER_VISION
```

```
NOT OBSCURED              402587
UNKNOWN                   154308
OTHER                       7213
MOVING VEHICLES             5407
PARKED VEHICLES             3337
WINDSHIELD (WATER/ICE)      2459
BLINDED - SUNLIGHT          1157
TREES, PLANTS                374
BUILDINGS                    306
BLINDED - HEADLIGHTS          75
HILLCREST                     65
EMBANKMENT                    63
BLOWING MATERIALS             53
SIGNBOARD                     25
Name: DRIVER_VISION, dtype: int64
```

```
PHYSICAL_CONDITION
```

```
NORMAL                          498584
UNKNOWN                          64022
IMPAIRED - ALCOHOL                4392
FATIGUED/ASLEEP                   2382
REMOVED BY EMS                    2067
OTHER                             1851
EMOTIONAL                         1755
```

```
ILLNESS/FAINTED                  940
HAD BEEN DRINKING                559
IMPAIRED - DRUGS                 503
IMPAIRED - ALCOHOL AND DRUGS     260
MEDICATED                        114
Name: PHYSICAL_CONDITION, dtype: int64
```

MANEUVER

```
STRAIGHT AHEAD                        334081
SLOW/STOP IN TRAFFIC                   64937
TURNING LEFT                           47831
TURNING RIGHT                          24298
BACKING                                23304
PASSING/OVERTAKING                     14445
CHANGING LANES                         13288
OTHER                                   9067
ENTERING TRAFFIC LANE FROM PARKING      8040
UNKNOWN/NA                              5428
STARTING IN TRAFFIC                     4890
MERGING                                 4491
U-TURN                                  4459
SKIDDING/CONTROL LOSS                   3671
AVOIDING VEHICLES/OBJECTS               3501
ENTER FROM DRIVE/ALLEY                  2823
LEAVING TRAFFIC LANE TO PARK            2739
SLOW/STOP - LEFT TURN                   1867
SLOW/STOP - RIGHT TURN                  1168
DRIVING WRONG WAY                        951
NEGOTIATING A CURVE                      888
SLOW/STOP - LOAD/UNLOAD                  883
TURNING ON RED                           275
DIVERGING                                 98
PARKED                                     5
PARKED IN TRAFFIC LANE                     1
Name: MANEUVER, dtype: int64
```

POSTED_SPEED_LIMIT

```
30    447733
35     44864
25     28767
20     15932
15     12952
40      7786
10      7127
45      4680
0       4345
```

```
5       2104
55       541
50       171
3        136
39        73
9         56
60        35
34        14
32        13
2         13
1         12
33        12
99        10
24         9
7          8
11         7
36         5
65         4
44         2
31         2
63         2
12         2
70         2
23         2
38         2
29         2
49         1
4          1
6          1
26         1
Name: POSTED_SPEED_LIMIT, dtype: int64

TRAFFIC_CONTROL_DEVICE

NO CONTROLS                 268293
TRAFFIC SIGNAL              214682
STOP SIGN/FLASHER            74589
UNKNOWN                      10019
OTHER                         3784
LANE USE MARKING              1415
YIELD                         1181
OTHER REG. SIGN                720
OTHER WARNING SIGN             594
RAILROAD CROSSING GATE         467
PEDESTRIAN CROSSING SIGN       386
DELINEATORS                    299
POLICE/FLAGMAN                 264
FLASHING CONTROL SIGNAL        256
```

```
SCHOOL ZONE                     190
OTHER RAILROAD CROSSING         155
RR CROSSING SIGN                 70
NO PASSING                       53
BICYCLE CROSSING SIGN            12
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64

DEVICE_CONDITION

NO CONTROLS                  275828
FUNCTIONING PROPERLY         268945
UNKNOWN                       21966
OTHER                          4683
FUNCTIONING IMPROPERLY         3687
NOT FUNCTIONING                1935
WORN REFLECTIVE MATERIAL        292
MISSING                          93
Name: DEVICE_CONDITION, dtype: int64

WEATHER_CONDITION

CLEAR                        461336
RAIN                          57824
SNOW                          23114
CLOUDY/OVERCAST               20434
UNKNOWN                        9340
OTHER                          1814
FREEZING RAIN/DRIZZLE          1250
SLEET/HAIL                      935
FOG/SMOKE/HAZE                  921
BLOWING SNOW                    343
SEVERE CROSS WIND GATE          114
BLOWING SAND, SOIL, DIRT          4
Name: WEATHER_CONDITION, dtype: int64

LIGHTING_CONDITION

DAYLIGHT                   395940
DARKNESS, LIGHTED ROAD     126260
DARKNESS                    22850
DUSK                        17561
DAWN                         9363
UNKNOWN                      5455
Name: LIGHTING_CONDITION, dtype: int64

FIRST_CRASH_TYPE

REAR END                        177392
```

```
TURNING                          121054
SIDESWIPE SAME DIRECTION         102062
ANGLE                             89605
PARKED MOTOR VEHICLE              33397
FIXED OBJECT                      13730
SIDESWIPE OPPOSITE DIRECTION       8748
HEAD ON                            6668
REAR TO FRONT                      6319
PEDESTRIAN                         5705
PEDALCYCLIST                       3973
REAR TO SIDE                       3901
OTHER OBJECT                       2416
REAR TO REAR                       1008
OTHER NONCOLLISION                  927
ANIMAL                              278
OVERTURNED                          218
TRAIN                                28
Name: FIRST_CRASH_TYPE, dtype: int64

TRAFFICWAY_TYPE

NOT DIVIDED                         256091
DIVIDED - W/MEDIAN (NOT RAISED)     114787
ONE-WAY                              51042
FOUR WAY                             44614
DIVIDED - W/MEDIAN BARRIER           42417
PARKING LOT                          22391
OTHER                                13789
T-INTERSECTION                        8576
CENTER TURN LANE                      6297
ALLEY                                 6148
UNKNOWN                               2735
RAMP                                  1967
UNKNOWN INTERSECTION TYPE             1761
DRIVEWAY                              1363
FIVE POINT, OR MORE                    979
Y-INTERSECTION                         889
TRAFFIC ROUTE                          836
NOT REPORTED                           477
ROUNDABOUT                             140
L-INTERSECTION                         130
Name: TRAFFICWAY_TYPE, dtype: int64

ALIGNMENT

STRAIGHT AND LEVEL       560713
STRAIGHT ON GRADE          8877
CURVE, LEVEL               4610
```

```
STRAIGHT ON HILLCREST      2012
CURVE ON GRADE              929
CURVE ON HILLCREST          288
Name: ALIGNMENT, dtype: int64

ROADWAY_SURFACE_COND

DRY                  437440
WET                   89271
UNKNOWN               22479
SNOW OR SLUSH         21727
ICE                    5074
OTHER                  1266
SAND, MUD, DIRT         172
Name: ROADWAY_SURFACE_COND, dtype: int64

ROAD_DEFECT

NO DEFECTS            494189
UNKNOWN                72406
RUT, HOLES              3453
OTHER                   3284
WORN SURFACE            2455
SHOULDER DEFECT         1152
DEBRIS ON ROADWAY        490
Name: ROAD_DEFECT, dtype: int64

CRASH_TYPE

NO INJURY / DRIVE AWAY               386447
INJURY AND / OR TOW DUE TO CRASH     190982
Name: CRASH_TYPE, dtype: int64

DAMAGE

OVER $1,500       382384
$501 - $1,500     141125
$500 OR LESS       53920
Name: DAMAGE, dtype: int64

PRIM_CONTRIBUTORY_CAUSE

FAILING TO YIELD RIGHT-OF-WAY
124046
FOLLOWING TOO CLOSELY
114808
IMPROPER OVERTAKING/PASSING
46845
```

FAILING TO REDUCE SPEED TO AVOID CRASH
45147
IMPROPER TURNING/NO SIGNAL
36551
IMPROPER LANE USAGE
36236
IMPROPER BACKING
31879
DRIVING SKILLS/KNOWLEDGE/EXPERIENCE
26422
DISREGARDING TRAFFIC SIGNALS
23027
WEATHER
15666
DISREGARDING STOP SIGN
11590
OPERATING VEHICLE IN ERRATIC, RECKLESS, CARELESS, NEGLIGENT OR AGGRESSIVE MANNER
8158
DISTRACTION - FROM INSIDE VEHICLE
7575
VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)
6106
EQUIPMENT - VEHICLE CONDITION
6083
PHYSICAL CONDITION OF DRIVER
5455
UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)
4862
DRIVING ON WRONG SIDE/WRONG WAY
4551
DISTRACTION - FROM OUTSIDE VEHICLE
4087
EXCEEDING SAFE SPEED FOR CONDITIONS
2228
DISREGARDING OTHER TRAFFIC SIGNS
2220
EXCEEDING AUTHORIZED SPEED LIMIT
2061
ROAD CONSTRUCTION/MAINTENANCE
1864
ROAD ENGINEERING/SURFACE/MARKING DEFECTS
1847
EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST
1642
CELL PHONE USE OTHER THAN TEXTING
1372
DISREGARDING ROAD MARKINGS
1212

```
TURNING RIGHT ON RED
755
HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)
653
ANIMAL
576
DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.)
510
RELATED TO BUS STOP
404
TEXTING
398
DISREGARDING YIELD SIGN
327
PASSING STOPPED SCHOOL BUS
105
OBSTRUCTED CROSSWALKS
77
BICYCLE ADVANCING LEGALLY ON RED LIGHT
61
MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT
23
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64

NUM_UNITS

2     502239
3      47233
1      15406
4       9390
5       2023
6        699
7        208
8        135
9         49
10        22
12        11
18         6
11         6
15         1
16         1
Name: NUM_UNITS, dtype: int64

CRASH_HOUR

16    47029
15    46175
17    46149
```

```
14     39356
18     37032
13     35432
12     34016
8      32932
11     30545
9      27520
10     26900
7      26500
19     26095
20     20380
21     17939
22     16095
23     13594
6      12104
0      10137
1       8180
2       6865
5       6437
3       5322
4       4695
Name: CRASH_HOUR, dtype: int64

CRASH_DAY_OF_WEEK

6    96922
5    85436
4    83736
7    83672
3    83247
2    78489
1    65927
Name: CRASH_DAY_OF_WEEK, dtype: int64

CRASH_MONTH

10     53673
12     50276
9      49760
5      49063
11     48996
8      47978
1      47108
7      46961
3      46683
6      45740
4      45731
2      45460
```

```
Name: CRASH_MONTH, dtype: int64

LATITUDE

41.976201    969
41.900959    600
41.791420    458
41.751461    456
41.880856    324
            …
41.889348      1
41.780923      1
41.740613      1
41.860918      1
41.835886      1
Name: LATITUDE, Length: 155350, dtype: int64

LONGITUDE

-87.905309    969
-87.619928    600
-87.580148    458
-87.585972    456
-87.617636    324
             …
-87.661895      1
-87.570930      1
-87.665346      1
-87.548734      1
-87.724474      1
Name: LONGITUDE, Length: 155334, dtype: int64

LOCATION

POINT (-87.905309125103 41.976201139024)    969
POINT (-87.619928173678 41.900958919109)    600
POINT (-87.580147768689 41.791420282098)    458
POINT (-87.585971992965 41.751460603167)    456
POINT (-87.617635891755 41.880856047671)    324
                                           …
POINT (-87.663909046208 41.896235950067)      1
POINT (-87.711181809431 41.891937228592)      1
POINT (-87.765246938384 41.778133615169)      1
POINT (-87.618487458568 41.896927006997)      1
POINT (-87.724474013253 41.835886103363)      1
Name: LOCATION, Length: 155401, dtype: int64
```

- CRASH_RECORD_ID, VEHICLE_ID can be dropped as we already used it to merge all

datasets.
- PERSON_TYPE has only one variable. We can drop this.
- FIRST_CRASH_TYPE seems more like a resultant variable rather than an action leading to the crash. We will not use this for columns for our prediction.
- Majority of ALIGNMENT is STRAIGHT AND LEVEL. We will drop this column.
- CRASH_TYPE does not present any insighful information. We will drop this.
- We are not interested in DAMAGE for this EDA. We will drop this.
- NUM_UNITS involved in car crash is also irrelevant for our business case. We will drop this.

[37]:
```
# Dropping irrelevant columns
final_data=merg_data.drop(['CRASH_RECORD_ID', 'VEHICLE_ID',␣
↪'PERSON_TYPE','FIRST_CRASH_TYPE',
                            'ALIGNMENT', 'CRASH_TYPE', 'DAMAGE', 'NUM_UNITS'],␣
↪axis=1).copy()
```

[38]:
```
# cheaking the head of our dataset
final_data.head()
```

[38]:
```
  SEX   AGE          DRIVER_ACTION DRIVER_VISION  PHYSICAL_CONDITION  \
0   F  20.0  FOLLOWED TOO CLOSELY       UNKNOWN              NORMAL
1   F  53.0               UNKNOWN       UNKNOWN              NORMAL
2   M  22.0       IMPROPER BACKING  NOT OBSCURED  IMPAIRED - ALCOHOL
3   M  67.0                  NONE  NOT OBSCURED              NORMAL
4   M  54.0                  NONE  NOT OBSCURED              NORMAL

        MANEUVER  POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE  \
0  STRAIGHT AHEAD                  30         TRAFFIC SIGNAL
1  STRAIGHT AHEAD                  30         TRAFFIC SIGNAL
2         BACKING                  30            NO CONTROLS
3  STRAIGHT AHEAD                  30            NO CONTROLS
4  STRAIGHT AHEAD                  30            NO CONTROLS

      DEVICE_CONDITION WEATHER_CONDITION  … TRAFFICWAY_TYPE  \
0  FUNCTIONING PROPERLY             CLEAR  …     NOT DIVIDED
1  FUNCTIONING PROPERLY             CLEAR  …     NOT DIVIDED
2           NO CONTROLS             CLEAR  …         ONE-WAY
3           NO CONTROLS             CLEAR  …         ONE-WAY
4           NO CONTROLS             CLEAR  …         ONE-WAY

  ROADWAY_SURFACE_COND ROAD_DEFECT  \
0                  DRY  NO DEFECTS
1                  DRY  NO DEFECTS
2                  DRY  NO DEFECTS
3                  DRY  NO DEFECTS
4                  DRY  NO DEFECTS

                             PRIM_CONTRIBUTORY_CAUSE CRASH_HOUR  \
```

```
0                              FOLLOWING TOO CLOSELY         23
1                              FOLLOWING TOO CLOSELY         23
2  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         23
3  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         23
4  UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN…         23

   CRASH_DAY_OF_WEEK  CRASH_MONTH   LATITUDE  LONGITUDE  \
0                  3            5  41.952691 -87.807413
1                  3            5  41.952691 -87.807413
2                  3            5  41.997837 -87.688814
3                  3            5  41.997837 -87.688814
4                  3            5  41.997837 -87.688814

                                  LOCATION
0  POINT (-87.807413247555 41.952691362649)
1  POINT (-87.807413247555 41.952691362649)
2  POINT (-87.688813887189 41.997837266972)
3  POINT (-87.688813887189 41.997837266972)
4  POINT (-87.688813887189 41.997837266972)

[5 rows x 21 columns]
```

[39]:
```
# cheaking the shape of ourdatset
final_data.shape
```

[39]:
```
(577429, 21)
```

our final data has 577429 rows with 21 columns and most columns are categorical

8.0.4 Feature Engineering

The feature engineering includes creating date features such as day of the week, handling the high cardinality of weather conditions, contributing cause, etc, and perhaps most importantly, downsampling to account for the class imbalance (injuries are more rare than non-injury-causing crashes).I will look at both Target and Predictor variables to see if we can change some variable to make more meaninful interpretation

Target Variable

[40]:
```
# List the primary contributory causes as our target
target_list = list(final_data.PRIM_CONTRIBUTORY_CAUSE.unique())
target_list
```

[40]:
```
['FOLLOWING TOO CLOSELY',
 'UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)',
 'FAILING TO REDUCE SPEED TO AVOID CRASH',
 'FAILING TO YIELD RIGHT-OF-WAY',
 'OPERATING VEHICLE IN ERRATIC, RECKLESS, CARELESS, NEGLIGENT OR AGGRESSIVE
```

```
MANNER',
 'IMPROPER LANE USAGE',
 'DRIVING SKILLS/KNOWLEDGE/EXPERIENCE',
 'IMPROPER OVERTAKING/PASSING',
 'IMPROPER TURNING/NO SIGNAL',
 'DISTRACTION - FROM OUTSIDE VEHICLE',
 'ANIMAL',
 'DISREGARDING TRAFFIC SIGNALS',
 'EQUIPMENT - VEHICLE CONDITION',
 'PHYSICAL CONDITION OF DRIVER',
 'IMPROPER BACKING',
 'DISREGARDING STOP SIGN',
 'DRIVING ON WRONG SIDE/WRONG WAY',
 'DISTRACTION - FROM INSIDE VEHICLE',
 'CELL PHONE USE OTHER THAN TEXTING',
 'TURNING RIGHT ON RED',
 'VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)',
 'RELATED TO BUS STOP',
 'WEATHER',
 'EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST',
 'DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.)',
 'DISREGARDING ROAD MARKINGS',
 'ROAD CONSTRUCTION/MAINTENANCE',
 'DISREGARDING OTHER TRAFFIC SIGNS',
 'ROAD ENGINEERING/SURFACE/MARKING DEFECTS',
 'TEXTING',
 'BICYCLE ADVANCING LEGALLY ON RED LIGHT',
 'HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)',
 'DISREGARDING YIELD SIGN',
 'OBSTRUCTED CROSSWALKS',
 'PASSING STOPPED SCHOOL BUS',
 'MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT',
 'EXCEEDING SAFE SPEED FOR CONDITIONS',
 'EXCEEDING AUTHORIZED SPEED LIMIT']
```

[41]:
```
#cheeking  the length of the data
len(target_list)
```

[41]:
```
38
```

I have 38 target variables which are a bit too many to classify and predict.I will group them into two categories: UNINTENTIONAL (0) and INTENTIONAL (1) causes. UNINTENTIONAL causes are typically related to errors or mistakes made by drivers or external factors that are beyond their control INTENTIONAL causes refer to accidents that occur due to intentional actions or choices made by drivers

I will group them as follows:

UNINTENTIONAL (0):

- IMPROPER OVERTAKING/PASSING
- DISREGARDING TRAFFIC SIGNALS
- DRIVING SKILLS/KNOWLEDGE/EXPERIENCE
- IMPROPER TURNING/NO SIGNAL
- FAILING TO REDUCE SPEED TO AVOID CRASH
- FOLLOWING TOO CLOSELY
- IMPROPER BACKING
- IMPROPER LANE USAGE
- FAILING TO YIELD RIGHT-OF-WAY
- DISREGARDING STOP SIGN
- VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)
- EQUIPMENT - VEHICLE CONDITION
- DISREGARDING OTHER TRAFFIC SIGNS
- DRIVING ON WRONG SIDE/WRONG WAY
- WEATHER
- PHYSICAL CONDITION OF DRIVER
- ROAD ENGINEERING/SURFACE/MARKING DEFECTS
- OBSTRUCTED CROSSWALKS
- EXCEEDING AUTHORIZED SPEED LIMIT
- EXCEEDING SAFE SPEED FOR CONDITIONS
- ROAD CONSTRUCTION/MAINTENANCE
- DISREGARDING ROAD MARKINGS
- DISREGARDING YIELD SIGN
- EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST
- ANIMAL
- RELATED TO BUS STOP
- TURNING RIGHT ON RED
- PASSING STOPPED SCHOOL BUS
- BICYCLE ADVANCING LEGALLY ON RED LIGHT
- MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT

INTENTIONAL (1):

- DISTRACTION - FROM INSIDE VEHICLE
- UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)
- DISTRACTION - FROM OUTSIDE VEHICLE
- TEXTING
- DISREGARDING ROAD MARKINGS
- CELL PHONE USE OTHER THAN TEXTING
- DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.)
- HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)

[42]:
```
#  convert the categorical variable 'PRIM_CONTRIBUTORY_CAUSE' into numerical␣
↪values that represent the two categories you defined: UNINTENTIONAL (0) and␣
↪INTENTIONAL (1).
```

```
final_data.PRIM_CONTRIBUTORY_CAUSE = final_data.PRIM_CONTRIBUTORY_CAUSE.map({
    'DRIVING SKILLS/KNOWLEDGE/EXPERIENCE':0,
    'DISTRACTION - FROM INSIDE VEHICLE':0,
    'VISION OBSCURED (SIGNS, TREE LIMBS, BUILDINGS, ETC.)':0,
    'WEATHER':0,
    'DISTRACTION - FROM OUTSIDE VEHICLE':0,
    'ROAD ENGINEERING/SURFACE/MARKING DEFECTS':0,
    'OBSTRUCTED CROSSWALKS':0,
    'BICYCLE ADVANCING LEGALLY ON RED LIGHT':0,
    'MOTORCYCLE ADVANCING LEGALLY ON RED LIGHT':0,
    'FAILING TO YIELD RIGHT-OF-WAY':0,
    'FAILING TO REDUCE SPEED TO AVOID CRASH':0,
    'PHYSICAL CONDITION OF DRIVER':0,
    'TEXTING':0,
    'ROAD CONSTRUCTION/MAINTENANCE':0,
    'EVASIVE ACTION DUE TO ANIMAL, OBJECT, NONMOTORIST':0,
    'ANIMAL':0,
    'CELL PHONE USE OTHER THAN TEXTING':0,
    'RELATED TO BUS STOP':0,
    'TURNING RIGHT ON RED':0,
    'DISTRACTION - OTHER ELECTRONIC DEVICE (NAVIGATION DEVICE, DVD PLAYER, ETC.
 ↪)':0,
    'PASSING STOPPED SCHOOL BUS':0,
    'EQUIPMENT - VEHICLE CONDITION':0,
    'IMPROPER OVERTAKING/PASSING':1,
    'DISREGARDING TRAFFIC SIGNALS':1,
    'IMPROPER TURNING/NO SIGNAL':1,
    'FOLLOWING TOO CLOSELY':1,
    'IMPROPER BACKING':1,
    'IMPROPER LANE USAGE':1,
    'DISREGARDING STOP SIGN':1,
    'DISREGARDING OTHER TRAFFIC SIGNS':1,
    'DRIVING ON WRONG SIDE/WRONG WAY':1,
    'OPERATING VEHICLE IN ERRATIC, RECKLESS, CARELESS, NEGLIGENT OR AGGRESSIVE␣
 ↪MANNER':1,
    'DISREGARDING ROAD MARKINGS':1,
    'DISREGARDING YIELD SIGN':1,
    'EXCEEDING AUTHORIZED SPEED LIMIT':1,
    'EXCEEDING SAFE SPEED FOR CONDITIONS':1,
    'UNDER THE INFLUENCE OF ALCOHOL/DRUGS (USE WHEN ARREST IS EFFECTED)':1,
    'HAD BEEN DRINKING (USE WHEN ARREST IS NOT MADE)':1
})
```

[43]:

```
# counts of primary contributor
final_data.PRIM_CONTRIBUTORY_CAUSE.value_counts()
```

[43]:
```
1    327208
0    250221
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64
```

Their was two categories that will help in training the model:

UNINTENTIONAL values is *250221*

INTENTIONAL values is *327208*

Preditors

[44]:
```
#provide an overview of the unique values and their frequencies for each column␣
↪in the DataFrame
for col in final_data.columns:
    print('\n' + col + '\n')
    print(final_data[col].value_counts())
```

```
SEX

M    345523
F    231906
Name: SEX, dtype: int64

AGE

27.0     17531
25.0     17469
26.0     17259
28.0     17065
24.0     16647
         …
110.0        2
108.0        2
109.0        1
104.0        1
107.0        1
Name: AGE, Length: 108, dtype: int64

DRIVER_ACTION

NONE                                  269888
FAILED TO YIELD                        69467
OTHER                                  51970
FOLLOWED TOO CLOSELY                   47494
UNKNOWN                                38207
IMPROPER TURN                          19894
IMPROPER BACKING                       18826
```

```
IMPROPER LANE CHANGE                     17623
DISREGARDED CONTROL DEVICES              12344
IMPROPER PASSING                         12328
TOO FAST FOR CONDITIONS                  11212
WRONG WAY/SIDE                            2334
IMPROPER PARKING                          1683
CELL PHONE USE OTHER THAN TEXTING         1094
EVADING POLICE VEHICLE                    1010
OVERCORRECTED                              993
EMERGENCY VEHICLE ON CALL                  623
TEXTING                                    305
STOPPED SCHOOL BUS                          98
LICENSE RESTRICTIONS                        36
Name: DRIVER_ACTION, dtype: int64

DRIVER_VISION

NOT OBSCURED              402587
UNKNOWN                   154308
OTHER                       7213
MOVING VEHICLES             5407
PARKED VEHICLES             3337
WINDSHIELD (WATER/ICE)      2459
BLINDED - SUNLIGHT          1157
TREES, PLANTS                374
BUILDINGS                    306
BLINDED - HEADLIGHTS          75
HILLCREST                     65
EMBANKMENT                    63
BLOWING MATERIALS             53
SIGNBOARD                     25
Name: DRIVER_VISION, dtype: int64

PHYSICAL_CONDITION

NORMAL                          498584
UNKNOWN                          64022
IMPAIRED - ALCOHOL                4392
FATIGUED/ASLEEP                   2382
REMOVED BY EMS                    2067
OTHER                             1851
EMOTIONAL                         1755
ILLNESS/FAINTED                    940
HAD BEEN DRINKING                  559
IMPAIRED - DRUGS                   503
IMPAIRED - ALCOHOL AND DRUGS       260
MEDICATED                          114
Name: PHYSICAL_CONDITION, dtype: int64
```

```
MANEUVER

STRAIGHT AHEAD                          334081
SLOW/STOP IN TRAFFIC                     64937
TURNING LEFT                             47831
TURNING RIGHT                            24298
BACKING                                  23304
PASSING/OVERTAKING                       14445
CHANGING LANES                           13288
OTHER                                     9067
ENTERING TRAFFIC LANE FROM PARKING        8040
UNKNOWN/NA                                5428
STARTING IN TRAFFIC                       4890
MERGING                                   4491
U-TURN                                    4459
SKIDDING/CONTROL LOSS                     3671
AVOIDING VEHICLES/OBJECTS                 3501
ENTER FROM DRIVE/ALLEY                    2823
LEAVING TRAFFIC LANE TO PARK              2739
SLOW/STOP - LEFT TURN                     1867
SLOW/STOP - RIGHT TURN                    1168
DRIVING WRONG WAY                          951
NEGOTIATING A CURVE                        888
SLOW/STOP - LOAD/UNLOAD                    883
TURNING ON RED                             275
DIVERGING                                   98
PARKED                                       5
PARKED IN TRAFFIC LANE                       1
Name: MANEUVER, dtype: int64

POSTED_SPEED_LIMIT

30    447733
35     44864
25     28767
20     15932
15     12952
40      7786
10      7127
45      4680
0       4345
5       2104
55       541
50       171
3        136
39        73
9         56
```

```
60        35
34        14
32        13
2         13
1         12
33        12
99        10
24         9
7          8
11         7
36         5
65         4
44         2
31         2
63         2
12         2
70         2
23         2
38         2
29         2
49         1
4          1
6          1
26         1
Name: POSTED_SPEED_LIMIT, dtype: int64

TRAFFIC_CONTROL_DEVICE

NO CONTROLS                 268293
TRAFFIC SIGNAL              214682
STOP SIGN/FLASHER            74589
UNKNOWN                      10019
OTHER                         3784
LANE USE MARKING              1415
YIELD                         1181
OTHER REG. SIGN                720
OTHER WARNING SIGN             594
RAILROAD CROSSING GATE         467
PEDESTRIAN CROSSING SIGN       386
DELINEATORS                    299
POLICE/FLAGMAN                 264
FLASHING CONTROL SIGNAL        256
SCHOOL ZONE                    190
OTHER RAILROAD CROSSING        155
RR CROSSING SIGN                70
NO PASSING                      53
BICYCLE CROSSING SIGN           12
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64
```

```
DEVICE_CONDITION

NO CONTROLS                 275828
FUNCTIONING PROPERLY        268945
UNKNOWN                      21966
OTHER                         4683
FUNCTIONING IMPROPERLY        3687
NOT FUNCTIONING               1935
WORN REFLECTIVE MATERIAL       292
MISSING                         93
Name: DEVICE_CONDITION, dtype: int64

WEATHER_CONDITION

CLEAR                       461336
RAIN                         57824
SNOW                         23114
CLOUDY/OVERCAST              20434
UNKNOWN                       9340
OTHER                         1814
FREEZING RAIN/DRIZZLE         1250
SLEET/HAIL                     935
FOG/SMOKE/HAZE                 921
BLOWING SNOW                   343
SEVERE CROSS WIND GATE         114
BLOWING SAND, SOIL, DIRT         4
Name: WEATHER_CONDITION, dtype: int64

LIGHTING_CONDITION

DAYLIGHT                  395940
DARKNESS, LIGHTED ROAD    126260
DARKNESS                   22850
DUSK                       17561
DAWN                        9363
UNKNOWN                     5455
Name: LIGHTING_CONDITION, dtype: int64

TRAFFICWAY_TYPE

NOT DIVIDED                        256091
DIVIDED - W/MEDIAN (NOT RAISED)    114787
ONE-WAY                             51042
FOUR WAY                            44614
DIVIDED - W/MEDIAN BARRIER          42417
PARKING LOT                         22391
OTHER                               13789
```

```
T-INTERSECTION                          8576
CENTER TURN LANE                        6297
ALLEY                                   6148
UNKNOWN                                 2735
RAMP                                    1967
UNKNOWN INTERSECTION TYPE               1761
DRIVEWAY                                1363
FIVE POINT, OR MORE                      979
Y-INTERSECTION                           889
TRAFFIC ROUTE                            836
NOT REPORTED                             477
ROUNDABOUT                               140
L-INTERSECTION                           130
Name: TRAFFICWAY_TYPE, dtype: int64

ROADWAY_SURFACE_COND

DRY                 437440
WET                  89271
UNKNOWN              22479
SNOW OR SLUSH        21727
ICE                   5074
OTHER                 1266
SAND, MUD, DIRT        172
Name: ROADWAY_SURFACE_COND, dtype: int64

ROAD_DEFECT

NO DEFECTS            494189
UNKNOWN                72406
RUT, HOLES              3453
OTHER                   3284
WORN SURFACE            2455
SHOULDER DEFECT         1152
DEBRIS ON ROADWAY        490
Name: ROAD_DEFECT, dtype: int64

PRIM_CONTRIBUTORY_CAUSE

1    327208
0    250221
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64

CRASH_HOUR

16    47029
15    46175
17    46149
```

```
14    39356
18    37032
13    35432
12    34016
8     32932
11    30545
9     27520
10    26900
7     26500
19    26095
20    20380
21    17939
22    16095
23    13594
6     12104
0     10137
1      8180
2      6865
5      6437
3      5322
4      4695
Name: CRASH_HOUR, dtype: int64

CRASH_DAY_OF_WEEK

6    96922
5    85436
4    83736
7    83672
3    83247
2    78489
1    65927
Name: CRASH_DAY_OF_WEEK, dtype: int64

CRASH_MONTH

10    53673
12    50276
9     49760
5     49063
11    48996
8     47978
1     47108
7     46961
3     46683
6     45740
4     45731
2     45460
```

```
Name: CRASH_MONTH, dtype: int64

LATITUDE

41.976201    969
41.900959    600
41.791420    458
41.751461    456
41.880856    324
            …
41.889348      1
41.780923      1
41.740613      1
41.860918      1
41.835886      1
Name: LATITUDE, Length: 155350, dtype: int64

LONGITUDE

-87.905309    969
-87.619928    600
-87.580148    458
-87.585972    456
-87.617636    324
             …
-87.661895      1
-87.570930      1
-87.665346      1
-87.548734      1
-87.724474      1
Name: LONGITUDE, Length: 155334, dtype: int64

LOCATION

POINT (-87.905309125103 41.976201139024)    969
POINT (-87.619928173678 41.900958919109)    600
POINT (-87.580147768689 41.791420282098)    458
POINT (-87.585971992965 41.751460603167)    456
POINT (-87.617635891755 41.880856047671)    324
                                          …
POINT (-87.663909046208 41.896235950067)      1
POINT (-87.711181809431 41.891937228592)      1
POINT (-87.765246938384 41.778133615169)      1
POINT (-87.618487458568 41.896927006997)      1
POINT (-87.724474013253 41.835886103363)      1
Name: LOCATION, Length: 155401, dtype: int64
```

- SEX: The ‘SEX’ column already contains two categories, ‘M’ and ‘F’, which represent male and female. There is no need for cleaning

- AGE: The 'AGE' column represents different age values. It appears that the ages are already binned into specific values.
- DRIVER_ACTION: The 'DRIVER_ACTION' column contains various driver actions. It seems to have already been cleaned,
- MANEUVER: The 'MANEUVER' column represents different driving maneuvers. It appears to be relatively clean
- POSTED_SPEED_LIMIT: The 'POSTED_SPEED_LIMIT' column contains numeric values representing posted speed limits
- CRASH HOUR, DAY, and MONTH are all numberical variables
- LONGITUDES and LATITUDES are continuous numerical variables

The following should be reduced to smaller classes for better clasification:

```
* Driver Vision
* Physical Condition
* Device Condition
* Weather Condition
* Lighting Condition
* Trafficway Type
* Roadway Surface Condition
* Road Defect
*
```

Driver Vision

[45]:
```
# check frequency of each unique value in the "DRIVER_VISION"
final_data.DRIVER_VISION.value_counts()
```

[45]:
```
NOT OBSCURED              402587
UNKNOWN                   154308
OTHER                       7213
MOVING VEHICLES             5407
PARKED VEHICLES             3337
WINDSHIELD (WATER/ICE)      2459
BLINDED - SUNLIGHT          1157
TREES, PLANTS                374
BUILDINGS                    306
BLINDED - HEADLIGHTS          75
HILLCREST                     65
EMBANKMENT                    63
BLOWING MATERIALS             53
SIGNBOARD                     25
Name: DRIVER_VISION, dtype: int64
```

"DRIVER_VISION" column, all categories except "NOT OBSCURED" and "UNKNOWN" can be considered as different forms of "OBSCURED" vision. Therefore, we will combine all these variations into a single category called "OBSCURED."

[46]:
```
# Define the categories to be grouped as "OBSCURED"
obscured_categories = ['MOVING VEHICLES', 'PARKED VEHICLES', 'WINDSHIELD (WATER/
 ↪ICE)',
                       'BLINDED - SUNLIGHT', 'TREES, PLANTS', 'BUILDINGS',
                       'BLINDED - HEADLIGHTS', 'HILLCREST', 'EMBANKMENT',
                       'BLOWING MATERIALS', 'SIGNBOARD', 'OTHER']

# Group the categories into "OBSCURED"
final_data.loc[~final_data['DRIVER_VISION'].isin(['NOT OBSCURED', 'UNKNOWN']),␣
 ↪'DRIVER_VISION'] = 'OBSCURED'
# Count the occurrences of each category
final_data['DRIVER_VISION'].value_counts()
```

[46]:
```
NOT OBSCURED    402587
UNKNOWN         154308
OBSCURED         20534
Name: DRIVER_VISION, dtype: int64
```

Physical Condition

[47]:
```
# Count the occurrences of each category
final_data.PHYSICAL_CONDITION.value_counts()
```

[47]:
```
NORMAL                          498584
UNKNOWN                          64022
IMPAIRED - ALCOHOL                4392
FATIGUED/ASLEEP                   2382
REMOVED BY EMS                    2067
OTHER                             1851
EMOTIONAL                         1755
ILLNESS/FAINTED                    940
HAD BEEN DRINKING                  559
IMPAIRED - DRUGS                   503
IMPAIRED - ALCOHOL AND DRUGS       260
MEDICATED                          114
Name: PHYSICAL_CONDITION, dtype: int64
```

I will categorize all conditions other than "NORMAL" and "UNKNOWN" as "IMPAIRED" in the "PHYSICAL_CONDITION" column.

[48]:
```
# Define the categories to be renamed as "IMPAIRED"
impaired_categories = ['FATIGUED/ASLEEP', 'EMOTIONAL', 'ILLNESS/FAINTED',␣
 ↪'ALCOHOL/DRUGS']

# Rename the categories as "IMPAIRED"
```

```
final_data.loc[~final_data['PHYSICAL_CONDITION'].isin(['NORMAL', 'UNKNOWN']),␣
↪'PHYSICAL_CONDITION'] = 'IMPAIRED'
# Count the occurrences of each category
final_data['PHYSICAL_CONDITION'].value_counts()
```

[48]:
```
NORMAL      498584
UNKNOWN      64022
IMPAIRED     14823
Name: PHYSICAL_CONDITION, dtype: int64
```

Device Condition

[49]:
```
# Count the occurrences of each category
final_data.DEVICE_CONDITION.value_counts()
```

[49]:
```
NO CONTROLS                 275828
FUNCTIONING PROPERLY        268945
UNKNOWN                      21966
OTHER                         4683
FUNCTIONING IMPROPERLY        3687
NOT FUNCTIONING               1935
WORN REFLECTIVE MATERIAL       292
MISSING                         93
Name: DEVICE_CONDITION, dtype: int64
```

I will group DEVICE_CONDITION into NO_CONTROLS, FUNCTIONING, UNKNOWN, and NOT FUNCTIONING.

- We do not know what OTHER means. We will put them as UNKNOWN.
- WORN_RELECTIVE_MATERIAL basically means that the device was functioning properly. We will re-classify them as FUNCTIONING together with FUNCTIONING PROPERLY.
- MISSING means no controls present. We will group them into NO CONTROLS.
- FUNCTIONING IMPROPERLY is as good as NOT FUNCTIONING at all. We will group them together.

[50]:
```
# Define the categories to be grouped
unknown_categories = ['OTHER']
functioning_categories = ['WORN REFLECTIVE MATERIAL', 'FUNCTIONING PROPERLY']
no_controls_categories = ['MISSING']
not_functioning_categories = ['FUNCTIONING IMPROPERLY']

# Group the categories accordingly
final_data.loc[final_data['DEVICE_CONDITION'].isin(unknown_categories),␣
↪'DEVICE_CONDITION'] = 'UNKNOWN'
final_data.loc[final_data['DEVICE_CONDITION'].isin(functioning_categories),␣
↪'DEVICE_CONDITION'] = 'FUNCTIONING'
```

```
final_data.loc[final_data['DEVICE_CONDITION'].isin(no_controls_categories),␣
 ↪'DEVICE_CONDITION'] = 'NO CONTROLS'
final_data.loc[final_data['DEVICE_CONDITION'].isin(not_functioning_categories),␣
 ↪'DEVICE_CONDITION'] = 'NOT FUNCTIONING'

# Count the occurrences of each category
final_data['DEVICE_CONDITION'].value_counts()
```

[50]:
```
NO CONTROLS        275921
FUNCTIONING        269237
UNKNOWN             26649
NOT FUNCTIONING      5622
Name: DEVICE_CONDITION, dtype: int64
```

Weather Condition

[51]:
```
# Count the occurrences of each category
final_data.WEATHER_CONDITION.value_counts()
```

[51]:
```
CLEAR                       461336
RAIN                         57824
SNOW                         23114
CLOUDY/OVERCAST              20434
UNKNOWN                       9340
OTHER                         1814
FREEZING RAIN/DRIZZLE         1250
SLEET/HAIL                     935
FOG/SMOKE/HAZE                 921
BLOWING SNOW                   343
SEVERE CROSS WIND GATE         114
BLOWING SAND, SOIL, DIRT         4
Name: WEATHER_CONDITION, dtype: int64
```

[52]:
```
# Replacing various categories in the'WEATHER_CONDITION' column using the apply␣
 ↪method and lambda functions.
final_data.WEATHER_CONDITION = final_data.WEATHER_CONDITION.apply(lambda x:␣
 ↪'SNOW' if x == 'BLOWING SNOW' else x)
final_data.WEATHER_CONDITION = final_data.WEATHER_CONDITION.apply(lambda x:␣
 ↪'RAIN' if x in ['FREEZING RAIN/DRIZZLE',
                                                                                ␣
 ↪        'SLEET/HAIL'] else x)
final_data.WEATHER_CONDITION = final_data.WEATHER_CONDITION.apply(lambda x:␣
 ↪'OTHER' if x in ['FOG/SMOKE/HAZE',
                                                                                ␣
 ↪         'SEVERE CROSS WIND GATE',
                                                                                ␣
 ↪          'BLOWING SAND, SOIL, DIRT'] else x)
```

```python
# Count the occurrences of each category
final_data.WEATHER_CONDITION.value_counts()
```

[52]:
```
CLEAR              461336
RAIN                60009
SNOW                23457
CLOUDY/OVERCAST     20434
UNKNOWN              9340
OTHER                2853
Name: WEATHER_CONDITION, dtype: int64
```

Lighting Condition

[53]:
```python
# Count the occurrences of each category
final_data.LIGHTING_CONDITION.value_counts()
```

[53]:
```
DAYLIGHT                   395940
DARKNESS, LIGHTED ROAD     126260
DARKNESS                    22850
DUSK                        17561
DAWN                         9363
UNKNOWN                      5455
Name: LIGHTING_CONDITION, dtype: int64
```

There is two different instances for DARKNESS. We will combine them to be one.

[54]:
```python
#combining DARKNESS, LIGHTED ROAD to one
final_data.LIGHTING_CONDITION = final_data.LIGHTING_CONDITION.apply(lambda x:␣
 ↪'DARKNESS' if x == 'DARKNESS, LIGHTED ROAD' else x)
# Count the occurrences of each category
final_data.LIGHTING_CONDITION.value_counts()
```

[54]:
```
DAYLIGHT    395940
DARKNESS    149110
DUSK         17561
DAWN          9363
UNKNOWN       5455
Name: LIGHTING_CONDITION, dtype: int64
```

Trafficway Type

[55]:
```python
# Count the occurrences of each category
final_data.TRAFFICWAY_TYPE.value_counts()
```

[55]:
```
NOT DIVIDED                        256091
DIVIDED - W/MEDIAN (NOT RAISED)    114787
ONE-WAY                             51042
FOUR WAY                            44614
```

```
DIVIDED - W/MEDIAN BARRIER          42417
PARKING LOT                         22391
OTHER                               13789
T-INTERSECTION                       8576
CENTER TURN LANE                     6297
ALLEY                                6148
UNKNOWN                              2735
RAMP                                 1967
UNKNOWN INTERSECTION TYPE            1761
DRIVEWAY                             1363
FIVE POINT, OR MORE                   979
Y-INTERSECTION                        889
TRAFFIC ROUTE                         836
NOT REPORTED                          477
ROUNDABOUT                            140
L-INTERSECTION                        130
Name: TRAFFICWAY_TYPE, dtype: int64
```

All classes related to INTERSECTION can be grouped into one. Combine the two DIVIDED variations into one. The rest is all unique.

[56]:

```python
# Grouping classes and combining them to one
final_data.TRAFFICWAY_TYPE = final_data.TRAFFICWAY_TYPE.apply(lambda x:␣
↪'INTERSECTION' if x in ['T-INTERSECTION',
                                                                        ␣
↪            'UNKNOWN INTERSECTION TYPE',
                                                                        ␣
↪            'Y-INTERSECTION',
                                                                        ␣
↪            'L-INTERSECTION'] else x)
final_data.TRAFFICWAY_TYPE = final_data.TRAFFICWAY_TYPE.apply(lambda x:␣
↪'DIVIDED' if x in ['DIVIDED - W/MEDIAN (NOT RAISED)',
                                                                        ␣
↪       'DIVIDED - W/MEDIAN BARRIER'] else x)
# Count the occurrences of each category
final_data.TRAFFICWAY_TYPE.value_counts()
```

[56]:

```
NOT DIVIDED           256091
DIVIDED               157204
ONE-WAY                51042
FOUR WAY               44614
PARKING LOT            22391
OTHER                  13789
INTERSECTION           11356
CENTER TURN LANE        6297
ALLEY                   6148
UNKNOWN                 2735
```

```
RAMP                         1967
DRIVEWAY                     1363
FIVE POINT, OR MORE           979
TRAFFIC ROUTE                 836
NOT REPORTED                  477
ROUNDABOUT                    140
Name: TRAFFICWAY_TYPE, dtype: int64
```

Roadway Surface Condition

[57]:
```python
# Count the occurrences of each category
final_data.ROADWAY_SURFACE_COND.value_counts()
```

[57]:
```
DRY                    437440
WET                     89271
UNKNOWN                 22479
SNOW OR SLUSH           21727
ICE                      5074
OTHER                    1266
SAND, MUD, DIRT           172
Name: ROADWAY_SURFACE_COND, dtype: int64
```

I will combine ICE with SNOW OR SLUSH creating SNOW/SLUSH/ICE since they all occur during a snow. SAND, MUD, DIRT can also be comined to OTHER since they all represent a minority group.

[58]:
```python
# count the occurrences of each category
final_data.ROADWAY_SURFACE_COND.value_counts()
```

[58]:
```
DRY                    437440
WET                     89271
UNKNOWN                 22479
SNOW OR SLUSH           21727
ICE                      5074
OTHER                    1266
SAND, MUD, DIRT           172
Name: ROADWAY_SURFACE_COND, dtype: int64
```

Road Defect

[59]:
```python
# count the occurrences of each category
final_data.ROAD_DEFECT.value_counts()
```

[59]:
```
NO DEFECTS               494189
UNKNOWN                   72406
RUT, HOLES                 3453
OTHER                      3284
WORN SURFACE               2455
```

```
SHOULDER DEFECT        1152
DEBRIS ON ROADWAY       490
Name: ROAD_DEFECT, dtype: int64
```

Other than NO DEFECT and UNKNOWN, all others seem to be a variation of DEFECTS. We will group them together as DEFECTS.

[60]:
```
# Grouping classes and combining them to one
final_data.ROAD_DEFECT = final_data.ROAD_DEFECT.apply(lambda x: 'DEFECTS' if x␣
 ↪not in ['NO DEFECTS', 'UNKNOWN'] else x)
# count the occurrences of each category
final_data.ROAD_DEFECT.value_counts()
```

[60]:
```
NO DEFECTS    494189
UNKNOWN        72406
DEFECTS        10834
Name: ROAD_DEFECT, dtype: int64
```

final list of predictor classes.

[61]:
```
# provide an overview of the unique values and their frequencies for each␣
 ↪column in the DataFrame
for col in final_data.columns:
    print('\n' + col + '\n')
    print(final_data[col].value_counts())
```

```
SEX

M    345523
F    231906
Name: SEX, dtype: int64

AGE

27.0     17531
25.0     17469
26.0     17259
28.0     17065
24.0     16647
         …
110.0        2
108.0        2
109.0        1
104.0        1
107.0        1
Name: AGE, Length: 108, dtype: int64
```

```
DRIVER_ACTION

NONE                                269888
FAILED TO YIELD                      69467
OTHER                                51970
FOLLOWED TOO CLOSELY                 47494
UNKNOWN                              38207
IMPROPER TURN                        19894
IMPROPER BACKING                     18826
IMPROPER LANE CHANGE                 17623
DISREGARDED CONTROL DEVICES          12344
IMPROPER PASSING                     12328
TOO FAST FOR CONDITIONS              11212
WRONG WAY/SIDE                        2334
IMPROPER PARKING                      1683
CELL PHONE USE OTHER THAN TEXTING     1094
EVADING POLICE VEHICLE                1010
OVERCORRECTED                          993
EMERGENCY VEHICLE ON CALL              623
TEXTING                                305
STOPPED SCHOOL BUS                      98
LICENSE RESTRICTIONS                    36
Name: DRIVER_ACTION, dtype: int64

DRIVER_VISION

NOT OBSCURED    402587
UNKNOWN         154308
OBSCURED         20534
Name: DRIVER_VISION, dtype: int64

PHYSICAL_CONDITION

NORMAL      498584
UNKNOWN      64022
IMPAIRED     14823
Name: PHYSICAL_CONDITION, dtype: int64

MANEUVER

STRAIGHT AHEAD                        334081
SLOW/STOP IN TRAFFIC                   64937
TURNING LEFT                           47831
TURNING RIGHT                          24298
BACKING                                23304
PASSING/OVERTAKING                     14445
CHANGING LANES                         13288
OTHER                                   9067
```

```
ENTERING TRAFFIC LANE FROM PARKING      8040
UNKNOWN/NA                              5428
STARTING IN TRAFFIC                     4890
MERGING                                 4491
U-TURN                                  4459
SKIDDING/CONTROL LOSS                   3671
AVOIDING VEHICLES/OBJECTS               3501
ENTER FROM DRIVE/ALLEY                  2823
LEAVING TRAFFIC LANE TO PARK            2739
SLOW/STOP - LEFT TURN                   1867
SLOW/STOP - RIGHT TURN                  1168
DRIVING WRONG WAY                        951
NEGOTIATING A CURVE                      888
SLOW/STOP - LOAD/UNLOAD                  883
TURNING ON RED                           275
DIVERGING                                 98
PARKED                                     5
PARKED IN TRAFFIC LANE                     1
Name: MANEUVER, dtype: int64

POSTED_SPEED_LIMIT

30    447733
35     44864
25     28767
20     15932
15     12952
40      7786
10      7127
45      4680
0       4345
5       2104
55       541
50       171
3        136
39        73
9         56
60        35
34        14
32        13
2         13
1         12
33        12
99        10
24         9
7          8
11         7
36         5
```

```
65         4
44         2
31         2
63         2
12         2
70         2
23         2
38         2
29         2
49         1
4          1
6          1
26         1
Name: POSTED_SPEED_LIMIT, dtype: int64

TRAFFIC_CONTROL_DEVICE

NO CONTROLS                 268293
TRAFFIC SIGNAL              214682
STOP SIGN/FLASHER            74589
UNKNOWN                      10019
OTHER                         3784
LANE USE MARKING              1415
YIELD                         1181
OTHER REG. SIGN                720
OTHER WARNING SIGN             594
RAILROAD CROSSING GATE         467
PEDESTRIAN CROSSING SIGN       386
DELINEATORS                    299
POLICE/FLAGMAN                 264
FLASHING CONTROL SIGNAL        256
SCHOOL ZONE                    190
OTHER RAILROAD CROSSING        155
RR CROSSING SIGN                70
NO PASSING                      53
BICYCLE CROSSING SIGN           12
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64

DEVICE_CONDITION

NO CONTROLS        275921
FUNCTIONING        269237
UNKNOWN             26649
NOT FUNCTIONING      5622
Name: DEVICE_CONDITION, dtype: int64

WEATHER_CONDITION
```

```
CLEAR              461336
RAIN                60009
SNOW                23457
CLOUDY/OVERCAST     20434
UNKNOWN              9340
OTHER                2853
Name: WEATHER_CONDITION, dtype: int64
```

```
LIGHTING_CONDITION

DAYLIGHT    395940
DARKNESS    149110
DUSK         17561
DAWN          9363
UNKNOWN       5455
Name: LIGHTING_CONDITION, dtype: int64
```

```
TRAFFICWAY_TYPE

NOT DIVIDED            256091
DIVIDED                157204
ONE-WAY                 51042
FOUR WAY                44614
PARKING LOT             22391
OTHER                   13789
INTERSECTION            11356
CENTER TURN LANE         6297
ALLEY                    6148
UNKNOWN                  2735
RAMP                     1967
DRIVEWAY                 1363
FIVE POINT, OR MORE       979
TRAFFIC ROUTE             836
NOT REPORTED              477
ROUNDABOUT                140
Name: TRAFFICWAY_TYPE, dtype: int64
```

```
ROADWAY_SURFACE_COND

DRY                437440
WET                 89271
UNKNOWN             22479
SNOW OR SLUSH       21727
ICE                  5074
OTHER                1266
SAND, MUD, DIRT       172
Name: ROADWAY_SURFACE_COND, dtype: int64
```

```
ROAD_DEFECT

NO DEFECTS    494189
UNKNOWN        72406
DEFECTS        10834
Name: ROAD_DEFECT, dtype: int64

PRIM_CONTRIBUTORY_CAUSE

1    327208
0    250221
Name: PRIM_CONTRIBUTORY_CAUSE, dtype: int64

CRASH_HOUR

16    47029
15    46175
17    46149
14    39356
18    37032
13    35432
12    34016
8     32932
11    30545
9     27520
10    26900
7     26500
19    26095
20    20380
21    17939
22    16095
23    13594
6     12104
0     10137
1      8180
2      6865
5      6437
3      5322
4      4695
Name: CRASH_HOUR, dtype: int64

CRASH_DAY_OF_WEEK

6    96922
5    85436
4    83736
7    83672
3    83247
```

```
2    78489
1    65927
Name: CRASH_DAY_OF_WEEK, dtype: int64

CRASH_MONTH

10    53673
12    50276
9     49760
5     49063
11    48996
8     47978
1     47108
7     46961
3     46683
6     45740
4     45731
2     45460
Name: CRASH_MONTH, dtype: int64

LATITUDE

41.976201    969
41.900959    600
41.791420    458
41.751461    456
41.880856    324
            …
41.889348      1
41.780923      1
41.740613      1
41.860918      1
41.835886      1
Name: LATITUDE, Length: 155350, dtype: int64

LONGITUDE

-87.905309    969
-87.619928    600
-87.580148    458
-87.585972    456
-87.617636    324
             …
-87.661895      1
-87.570930      1
-87.665346      1
-87.548734      1
-87.724474      1
```

```
Name: LONGITUDE, Length: 155334, dtype: int64

LOCATION

POINT (-87.905309125103 41.976201139024)    969
POINT (-87.619928173678 41.900958919109)    600
POINT (-87.580147768689 41.791420282098)    458
POINT (-87.585971992965 41.751460603167)    456
POINT (-87.617635891755 41.880856047671)    324
                                           …
POINT (-87.663909046208 41.896235950067)      1
POINT (-87.711181809431 41.891937228592)      1
POINT (-87.765246938384 41.778133615169)      1
POINT (-87.618487458568 41.896927006997)      1
POINT (-87.724474013253 41.835886103363)      1
Name: LOCATION, Length: 155401, dtype: int64
```

By grouping similar classes together, I have improved the precision and relevance of the classification process, enabling a more meaningful analysis of the data.

[62]:
```
#Cheeking the shape of our final data
final_data.shape
```

[62]:
```
(577429, 21)
```

[63]:
```
# cheeking the head of our data
final_data.head()
```

[63]:
```
  SEX   AGE         DRIVER_ACTION DRIVER_VISION PHYSICAL_CONDITION  \
0   F  20.0  FOLLOWED TOO CLOSELY       UNKNOWN             NORMAL
1   F  53.0               UNKNOWN       UNKNOWN             NORMAL
2   M  22.0      IMPROPER BACKING  NOT OBSCURED           IMPAIRED
3   M  67.0                  NONE  NOT OBSCURED             NORMAL
4   M  54.0                  NONE  NOT OBSCURED             NORMAL

        MANEUVER  POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE DEVICE_CONDITION  \
0  STRAIGHT AHEAD                  30         TRAFFIC SIGNAL      FUNCTIONING
1  STRAIGHT AHEAD                  30         TRAFFIC SIGNAL      FUNCTIONING
2         BACKING                  30            NO CONTROLS      NO CONTROLS
3  STRAIGHT AHEAD                  30            NO CONTROLS      NO CONTROLS
4  STRAIGHT AHEAD                  30            NO CONTROLS      NO CONTROLS

  WEATHER_CONDITION  … TRAFFICWAY_TYPE ROADWAY_SURFACE_COND ROAD_DEFECT  \
0             CLEAR  …     NOT DIVIDED                  DRY  NO DEFECTS
1             CLEAR  …     NOT DIVIDED                  DRY  NO DEFECTS
2             CLEAR  …         ONE-WAY                  DRY  NO DEFECTS
3             CLEAR  …         ONE-WAY                  DRY  NO DEFECTS
4             CLEAR  …         ONE-WAY                  DRY  NO DEFECTS
```

```
  PRIM_CONTRIBUTORY_CAUSE  CRASH_HOUR  CRASH_DAY_OF_WEEK  CRASH_MONTH  \
0                       1          23                  3            5
1                       1          23                  3            5
2                       1          23                  3            5
3                       1          23                  3            5
4                       1          23                  3            5

    LATITUDE  LONGITUDE                                  LOCATION
0  41.952691 -87.807413  POINT (-87.807413247555 41.952691362649)
1  41.952691 -87.807413  POINT (-87.807413247555 41.952691362649)
2  41.997837 -87.688814  POINT (-87.688813887189 41.997837266972)
3  41.997837 -87.688814  POINT (-87.688813887189 41.997837266972)
4  41.997837 -87.688814  POINT (-87.688813887189 41.997837266972)

[5 rows x 21 columns]
```

[64]:
```python
#convert the values in the 'AGE' column of the DataFrame to strings
final_data.AGE = final_data.AGE.map(lambda x : str(x))
final_data
```

[64]:
```
       SEX  AGE         DRIVER_ACTION DRIVER_VISION PHYSICAL_CONDITION  \
0        F  20.0  FOLLOWED TOO CLOSELY       UNKNOWN             NORMAL
1        F  53.0               UNKNOWN       UNKNOWN             NORMAL
2        M  22.0       IMPROPER BACKING  NOT OBSCURED           IMPAIRED
3        M  67.0                  NONE  NOT OBSCURED             NORMAL
4        M  54.0                  NONE  NOT OBSCURED             NORMAL
…       ..   …                    …           …                …
577424   F  39.0                  NONE  NOT OBSCURED             NORMAL
577425   F  24.0                  NONE       UNKNOWN            UNKNOWN
577426   F  36.0        FAILED TO YIELD  NOT OBSCURED             NORMAL
577427   F  41.0                  NONE  NOT OBSCURED             NORMAL
577428   M  63.0                  NONE  NOT OBSCURED             NORMAL

             MANEUVER  POSTED_SPEED_LIMIT TRAFFIC_CONTROL_DEVICE  \
0       STRAIGHT AHEAD                  30         TRAFFIC SIGNAL
1       STRAIGHT AHEAD                  30         TRAFFIC SIGNAL
2              BACKING                  30            NO CONTROLS
3       STRAIGHT AHEAD                  30            NO CONTROLS
4       STRAIGHT AHEAD                  30            NO CONTROLS
…                  …                 …                    …
577424         BACKING                  30            NO CONTROLS
577425  STRAIGHT AHEAD                  30            NO CONTROLS
577426  STRAIGHT AHEAD                  30                  YIELD
577427  STRAIGHT AHEAD                  30                  YIELD
577428  STRAIGHT AHEAD                  30         TRAFFIC SIGNAL
```

```
       DEVICE_CONDITION WEATHER_CONDITION  … TRAFFICWAY_TYPE  \
0           FUNCTIONING             CLEAR  …     NOT DIVIDED
1           FUNCTIONING             CLEAR  …     NOT DIVIDED
2           NO CONTROLS             CLEAR  …         ONE-WAY
3           NO CONTROLS             CLEAR  …         ONE-WAY
4           NO CONTROLS             CLEAR  …         ONE-WAY
…                 …                 …   …            …
577424      NO CONTROLS             CLEAR  …     NOT DIVIDED
577425      NO CONTROLS              RAIN  …     NOT DIVIDED
577426      NO CONTROLS             CLEAR  …         DIVIDED
577427      NO CONTROLS             CLEAR  …         DIVIDED
577428      FUNCTIONING             CLEAR  …     NOT DIVIDED

       ROADWAY_SURFACE_COND ROAD_DEFECT PRIM_CONTRIBUTORY_CAUSE  CRASH_HOUR  \
0                       DRY  NO DEFECTS                       1          23
1                       DRY  NO DEFECTS                       1          23
2                       DRY  NO DEFECTS                       1          23
3                       DRY  NO DEFECTS                       1          23
4                       DRY  NO DEFECTS                       1          23
…                     …           …                      …          …
577424                  DRY  NO DEFECTS                       1          17
577425                  WET     UNKNOWN                       1          19
577426                  DRY  NO DEFECTS                       0           7
577427                  DRY  NO DEFECTS                       0           7
577428                  DRY  NO DEFECTS                       1          16

        CRASH_DAY_OF_WEEK  CRASH_MONTH   LATITUDE  LONGITUDE  \
0                       3            5  41.952691 -87.807413
1                       3            5  41.952691 -87.807413
2                       3            5  41.997837 -87.688814
3                       3            5  41.997837 -87.688814
4                       3            5  41.997837 -87.688814
…                     …          …         …          …
577424                  7            1  41.868979 -87.640629
577425                  4            6  41.835886 -87.724474
577426                  3            1  41.760710 -87.561946
577427                  3            1  41.760710 -87.561946
577428                  1            3  41.975857 -87.708744

                                         LOCATION
0       POINT (-87.807413247555 41.952691362649)
1       POINT (-87.807413247555 41.952691362649)
2       POINT (-87.688813887189 41.997837266972)
3       POINT (-87.688813887189 41.997837266972)
4       POINT (-87.688813887189 41.997837266972)
…                                             …
577424  POINT (-87.640628921351 41.868979049994)
```

```
577425  POINT (-87.724474013253 41.835886103363)
577426  POINT (-87.561946030143 41.760710194223)
577427  POINT (-87.561946030143 41.760710194223)
577428  POINT (-87.708743641643 41.975856915535)

[577429 rows x 21 columns]
```

8.1 Explanatory Data Analysis(EDA)

Answering the Business questions: #### Question one *Are there any specific locations or road segments in Chicago city that have a higher frequency of car accidents?*

By combining the accident data with the shapefile using Geopandas, we can overlay the accident points on the map of Chicago. This visualization will provide a spatial representation of the accident locations, allowing us to identify specific areas or patterns where accidents are more prevalent.

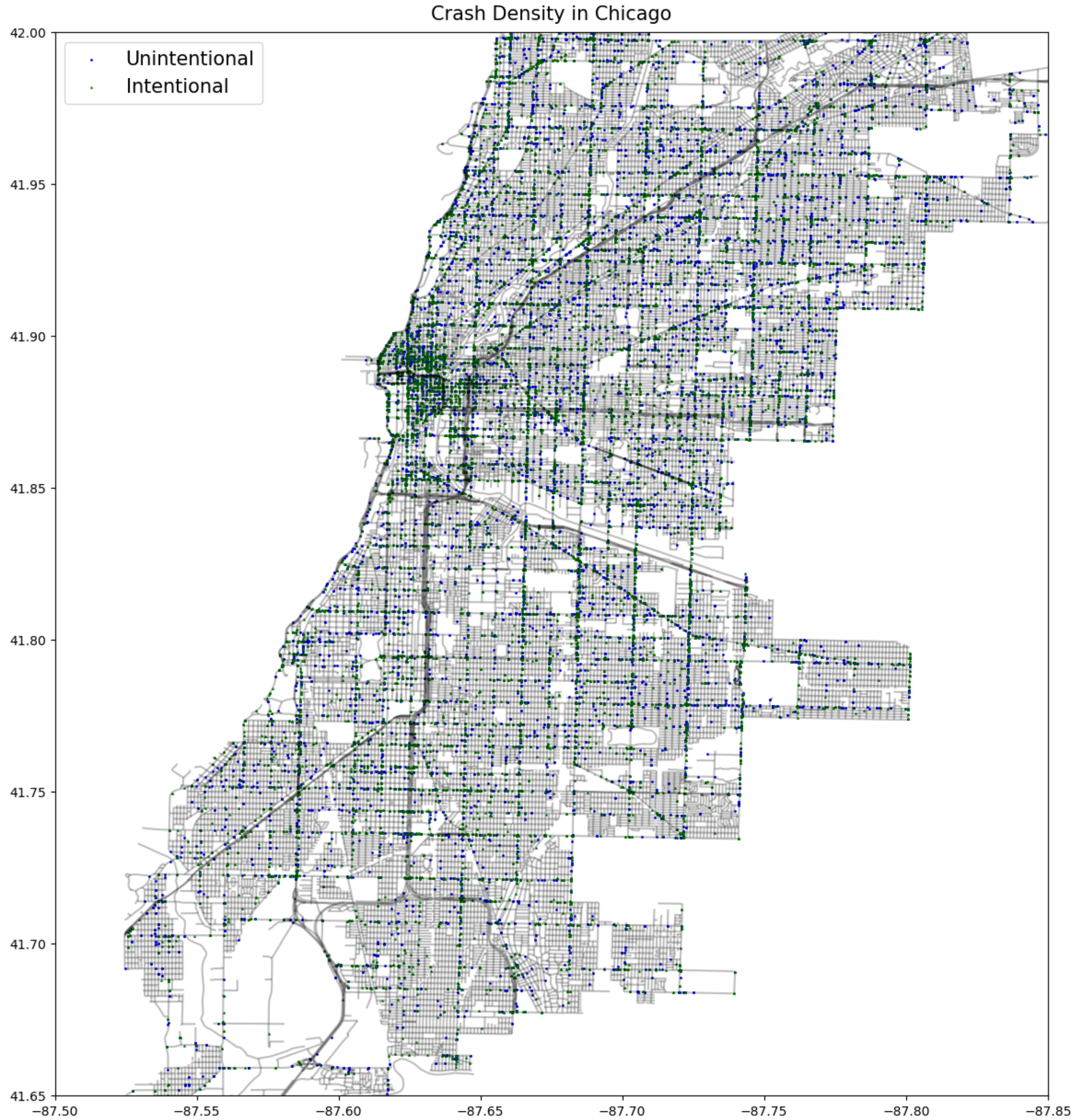
[65]:
```
#Importing libraries that will help as plot our streatshape
import geopandas as gpd
from shapely.geometry import Point
```

[66]:
```
# Import shape file for the streep map of Chicago
street_map = gpd.read_file(r'C:\\Users\\Admin\\Documents\\Iano\\phase 3␣
↪project\\dsc-phase-3-project-v2-3\\geo␣
↪data\\geo_export_1fec9e49-164d-454e-9b60-f2310808e96f.shx')
```

[67]:
```
# Create a location dataframe
location = final_data[['PRIM_CONTRIBUTORY_CAUSE', 'LONGITUDE', 'LATITUDE']][:
↪25000]

# Create a GeoDataFrame by assigning the coordinates as a Point geometry
geo_data = location.assign(geometry=gpd.points_from_xy(location.LONGITUDE,␣
↪location.LATITUDE))
geo_data = gpd.GeoDataFrame(geo_data, crs='EPSG:4326')

# Alternatively, you can directly assign the geometry while creating the␣
↪GeoDataFrame
geo_data = gpd.GeoDataFrame(
    location,
    geometry=gpd.points_from_xy(location.LONGITUDE, location.LATITUDE),
    crs='EPSG:4326'
)
```

[68]:
```
# Plot the coordinates
fig, ax = plt.subplots(figsize=(15, 15))
street_map.plot(ax=ax, alpha=0.25, color='black')
geo_data[geo_data['PRIM_CONTRIBUTORY_CAUSE'] == 0].plot(ax=ax, markersize=1,␣
↪color='blue', marker='o', label='Unintentional')
```

```
geo_data[geo_data['PRIM_CONTRIBUTORY_CAUSE'] == 1].plot(ax=ax, markersize=1,␣
↪color='green', marker='^', label='Intentional')
plt.legend(prop={'size': 15})
plt.title('Crash Density in Chicago', fontsize=15, pad=10)

# Set the desired limits for the x-axis and y-axis
plt.xlim(-87.5, -87.85)
plt.ylim(41.65, 42.0)

# Display the plot
plt.show()
plt.savefig(r'images\street_map.png', bbox_inches='tight');
```



<Figure size 640x480 with 0 Axes>

The map highlights a significant concentration of accidents in the downtown area of Chicago, indicating a higher density of incidents in that region. The predominant color in this area is green, indicating that a majority of the accidents are attributed to intentional actions or driver errors. However, it's important to note that there are also scattered blue plots throughout the map, suggesting a considerable number of accidents that occur unintentionally or present opportunities for improvement in terms of safety measures.

Question Two *What are the contributing factors or characteristics associated with severe car accidents in Chicago city?*

Our focus will be on accidents that were categorized as 'Unintentional' in order to delve deeper into the underlying causes and identify potential areas for improvement. By narrowing our analysis to these specific incidents, we can gain valuable insights into the root causes of unintentional accidents and uncover opportunities for enhancing safety measures and preventing similar occurrences in the future.

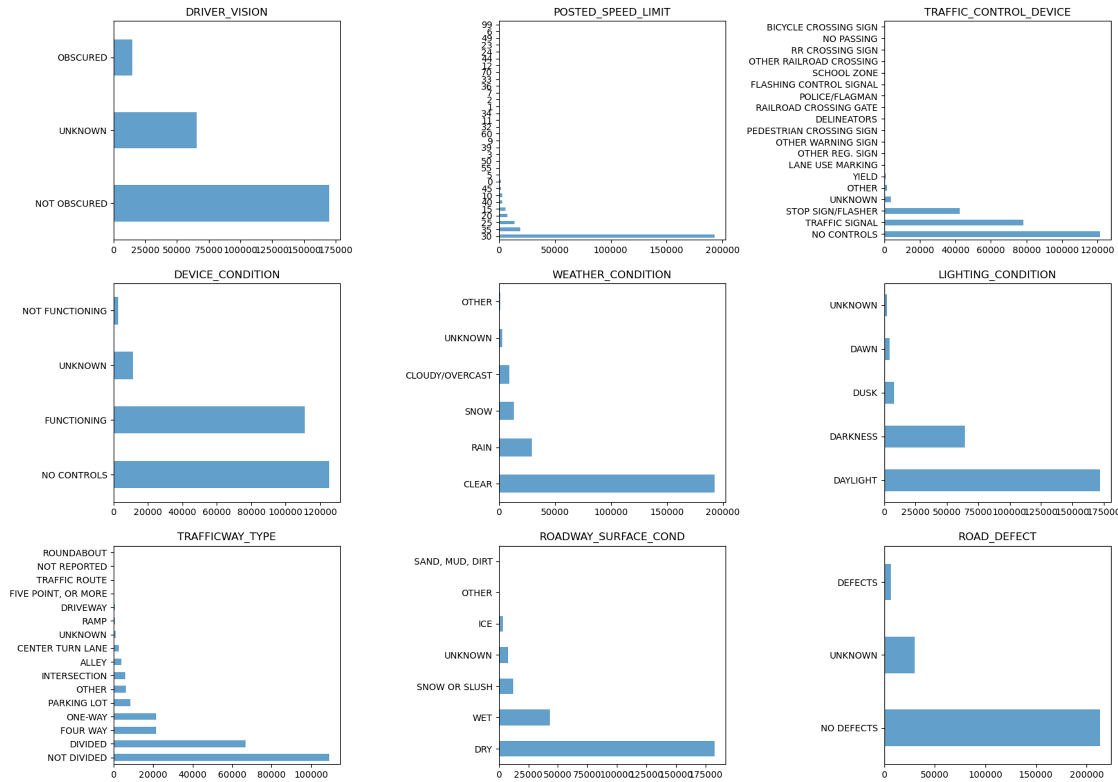
[69]:
```
# Selecting the relevant columns for analysis
factors =␣
 ↪final_data[['DRIVER_VISION','POSTED_SPEED_LIMIT','TRAFFIC_CONTROL_DEVICE','DEVICE_CONDITION
                     'WEATHER_CONDITION','LIGHTING_CONDITION',
                     'TRAFFICWAY_TYPE','ROADWAY_SURFACE_COND',
                     'ROAD_DEFECT','PRIM_CONTRIBUTORY_CAUSE']].copy()
```

[70]:
```
# Filtering the data for control failures (Unintentional accidents)
control_failures = factors[factors.PRIM_CONTRIBUTORY_CAUSE == 0].copy()
# Removing the 'PRIM_CONTRIBUTORY_CAUSE' column as it is no longer needed
control_failures.drop('PRIM_CONTRIBUTORY_CAUSE', axis=1, inplace=True)
# Displaying the column names of the control_failures DataFrame
control_failures.columns
```

[70]:
```
Index(['DRIVER_VISION', 'POSTED_SPEED_LIMIT', 'TRAFFIC_CONTROL_DEVICE',
       'DEVICE_CONDITION', 'WEATHER_CONDITION', 'LIGHTING_CONDITION',
       'TRAFFICWAY_TYPE', 'ROADWAY_SURFACE_COND', 'ROAD_DEFECT'],
      dtype='object')
```

[71]:
```
# Creating an empty dictionary to store column-wise value counts
#Calculating value counts for the current column and storing it in the␣
 ↪dictionary
# Iterating over each column in the control_failures DataFrame
count_dict = {}
for col in control_failures.columns:
    count_dict[str(col)] = control_failures[col].value_counts()
#plotting the predictors
plt.figure(figsize=(20,15))
plt.subplots_adjust(wspace=0.7)
```

```python
for index, value in enumerate(count_dict):
    ax = plt.subplot(3, 3, index+1)
    chart = pd.DataFrame(count_dict[value])
    chart.plot(ax=ax, kind='barh', legend=False, alpha=0.7)
    ax.set_title(value)
    plt.savefig(r'images\quiz_two.png', bbox_inches='tight');
```



From the above plot we can draw the following conclusions:

- Upon analyzing the contributing factors associated with control failures in unintentional accidents, it is evident that a majority of the accidents occurred when the driver's vision was not obscured. Furthermore, it is notable that these accidents occurred while the drivers were adhering to the posted speed limit, typically set at 30 mph. These findings suggest that factors other than vision or speed might be contributing to control failures in these accidents.
- An important finding from the analysis is that the absence of traffic control devices has been the primary contributing factor to the number of accidents in Chicago. This suggests that increasing the presence of traffic control devices throughout the city could potentially reduce the occurrence of unintentional accidents. This finding is further supported by the Device Condition plot, which indicates a higher count of accidents when there are no traffic control devices in place. Implementing and improving traffic control measures can therefore be an effective strategy to mitigate control failures and enhance road safety in Chicago.

- The analysis indicates that weather condition and lighting condition have relatively minimal impact on the occurrence of accidents. These factors do not show a strong correlation with the number of accidents in Chicago.
- Significant number of accidents occur on roads categorized as "Not Divided" in terms of trafficway type. This suggests that implementing road division measures, such as adding medians or physical barriers, can potentially mitigate the occurrence of accidents. Dividing the roads can enhance traffic management, separate opposing flows of traffic, and reduce the likelihood of collisions, thereby contributing to improved road safety.
- The analysis indicates that the roadway surface condition and road defects have a relatively minimal impact on the occurrence of these accidents. It suggests that the condition of the road surface, such as potholes or uneven pavement, and the presence of road defects, such as cracks or debris, may not be significant contributors to the unintentional accidents in Chicago.

Question Three *Are there any seasonal or temporal patterns in car accidents in Chicago city?*

I will explores whether there are any recurring patterns in car accidents based on hour, months, days of the week, or specific time intervals I will use both the intenational and unintentional crash data.

[72]:
```
# Selecting the columns 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', and 'CRASH_MONTH'␣
↪from the 'final_data' DataFrame
# and creating a new DataFrame called 'crash_time'
crash_time= final_data[['CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONTH']].
↪copy()
crash_time
```

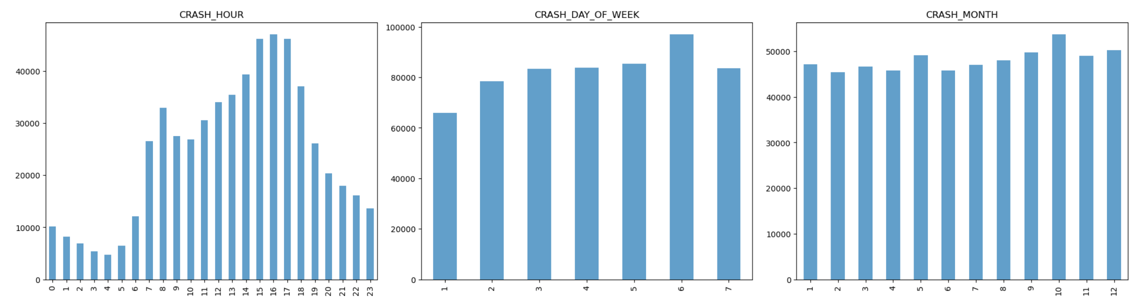
[72]:

	CRASH_HOUR	CRASH_DAY_OF_WEEK	CRASH_MONTH
0	23	3	5
1	23	3	5
2	23	3	5
3	23	3	5
4	23	3	5
…	…	…	…
577424	17	7	1
577425	19	4	6
577426	7	3	1
577427	7	3	1
577428	16	1	3

[577429 rows x 3 columns]

[73]:
```
# Plot the graphs
crash_time = final_data[['CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MONTH']].
↪copy()

plt.figure(figsize=(20, 15))
plt.subplots_adjust(wspace=0.7)
```

```
for i, col in enumerate(crash_time.columns):
    ax = plt.subplot(3, 3, i+1)
    chart_2 = pd.DataFrame(crash_time[col].value_counts()).sort_index()
    chart_2.plot(ax=ax, kind='bar', legend=False, alpha=0.7)
    ax.set_title(col)

plt.tight_layout()
plt.show()
plt.savefig(r'images\quiz_three.png', bbox_inches='tight');
```




```
<Figure size 640x480 with 0 Axes>
```

From the above plots we can come to conclusion:

- The analysis of the crash time data reveals that a significant number of accidents in Chicago occur between the hours of 14 to 18, which coincides with the peak rush hour traffic. This suggests that the high volume of vehicles during these hours contributes to the increased accident rate. Considering the concentration of accidents in the downtown area during this time frame, it becomes apparent that better traffic management strategies are needed.

To address this issue, it is recommended that the city implements additional measures to facilitate traffic flow and reduce congestion in the downtown area during these peak hours. This can include deploying more traffic management personnel or implementing intelligent transportation systems to optimize traffic signal timings and improve the coordination of traffic movements. By enhancing traffic management during rush hours, the city can mitigate the number of accidents and improve overall road safety in the downtown area.

- The analysis of the crash data by day of the week indicates that there is a slightly higher number of accidents during the weekends compared to other days. However, the difference in accident frequency between weekdays and weekends is not substantial. Therefore, it can be concluded that the crash hour plays a more significant role in determining accident occurrence than the specific day of the week.

- Analyzing the crash data by month reveals some interesting patterns. The number of car accidents in Chicago tends to be higher during the summer months, particularly in June, July, August and September. This can be attributed to various factors such as increased travel and tourism, more outdoor activities, and potentially more congested roads during the summer season.

However, it is important to note that while there may be higher accident rates during certain months, the difference in accident frequency between months is not significant enough to warrant major adjustments in road safety strategies based solely on the crash month.

Question Four Can we build a classification model to predict the primary contributory cause of car accidents? Developing a classification model will enable the CCVSB to categorize accidents into different causes, allowing for a deeper understanding of the factors contributing to each type of accident. This knowledge can inform targeted strategies for prevention.

I will build classifier to help me analyze the chicago car crash

Preparing data

[74]:
```
#Importing necessary libraries
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,␣
 ↪GradientBoostingClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import auc
from sklearn.metrics import  roc_curve
from sklearn.metrics import  roc_auc_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.feature_selection import SelectFromModel
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
%matplotlib inline
```

[75]:
```
# Create X dataframe by dropping specific columns from final_data
X = final_data.drop(['PRIM_CONTRIBUTORY_CAUSE',␣
 ↪'CRASH_HOUR','CRASH_DAY_OF_WEEK',
                   'CRASH_MONTH', 'LONGITUDE', 'LATITUDE', 'LOCATION'], axis=1).
 ↪copy()
# Assign PRIM_CONTRIBUTORY_CAUSE column to y variable
y = final_data.PRIM_CONTRIBUTORY_CAUSE
```

[76]:
```
# Convert predictors into dummies
X = pd.get_dummies(X, drop_first=True)
```

Train-Test Split To evaluate the performance of our model we are splitting the dataset into two parts: a training set and a testing set.we then train a machine learning model on the training set using the 'fit' method

[77]:
```
# split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,␣
↪random_state=42)
```

[78]:
```
print(X_train.shape)  # print (n_train_samples, n_features)
print(X_test.shape)   # print (n_test_samples, n_features)
print(y_train.shape)  # print (n_train_samples,)
print(y_test.shape)   # print (n_test_samples,)
```

```
(404200, 210)
(173229, 210)
(404200,)
(173229,)
```

Scale the data by standard scaler

[79]:
```
#Instantiate Standard Scaler
scaler = StandardScaler()

# Fit and transform train and test set
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[80]:
```
# Create a DataFrame from the scaled training data and display the first few␣
↪rows of the scaled training data DataFrame
scaled_data_train = pd.DataFrame(X_train_scaled , columns=X_train.columns)
scaled_data_train.head()
```

[80]:
```
   POSTED_SPEED_LIMIT     SEX_M  AGE_10.0  AGE_100.0  AGE_101.0  AGE_102.0  \
0            0.141445  0.819479 -0.006092  -0.003853  -0.002724  -0.001573
1            0.141445  0.819479 -0.006092  -0.003853  -0.002724  -0.001573
2            0.141445 -1.220287 -0.006092  -0.003853  -0.002724  -0.001573
3            0.141445  0.819479 -0.006092  -0.003853  -0.002724  -0.001573
4           -1.726058  0.819479 -0.006092  -0.003853  -0.002724  -0.001573

   AGE_103.0  AGE_104.0  AGE_107.0  AGE_108.0  …  \
0  -0.003146  -0.001573  -0.001573        0.0  …
1  -0.003146  -0.001573  -0.001573        0.0  …
2  -0.003146  -0.001573  -0.001573        0.0  …
3  -0.003146  -0.001573  -0.001573        0.0  …
4  -0.003146  -0.001573  -0.001573        0.0  …

   TRAFFICWAY_TYPE_TRAFFIC ROUTE  TRAFFICWAY_TYPE_UNKNOWN  \
0                      -0.037711                -0.068977
```

```
1                      -0.037711                -0.068977
2                      -0.037711                -0.068977
3                      -0.037711                -0.068977
4                      -0.037711                -0.068977

   ROADWAY_SURFACE_COND_ICE  ROADWAY_SURFACE_COND_OTHER  \
0                 -0.094518                   -0.047029
1                 -0.094518                   -0.047029
2                 -0.094518                   -0.047029
3                 -0.094518                   -0.047029
4                 -0.094518                   -0.047029

   ROADWAY_SURFACE_COND_SAND, MUD, DIRT  ROADWAY_SURFACE_COND_SNOW OR SLUSH  \
0                             -0.017016                           -0.197821
1                             -0.017016                           -0.197821
2                             -0.017016                           -0.197821
3                             -0.017016                           -0.197821
4                             -0.017016                           -0.197821

   ROADWAY_SURFACE_COND_UNKNOWN  ROADWAY_SURFACE_COND_WET  \
0                     -0.200954                  2.333701
1                     -0.200954                 -0.428504
2                     -0.200954                 -0.428504
3                     -0.200954                 -0.428504
4                     -0.200954                 -0.428504

   ROAD_DEFECT_NO DEFECTS  ROAD_DEFECT_UNKNOWN
0                0.409679            -0.377768
1                0.409679            -0.377768
2               -2.440936            -0.377768
3                0.409679            -0.377768
4                0.409679            -0.377768

[5 rows x 210 columns]
```

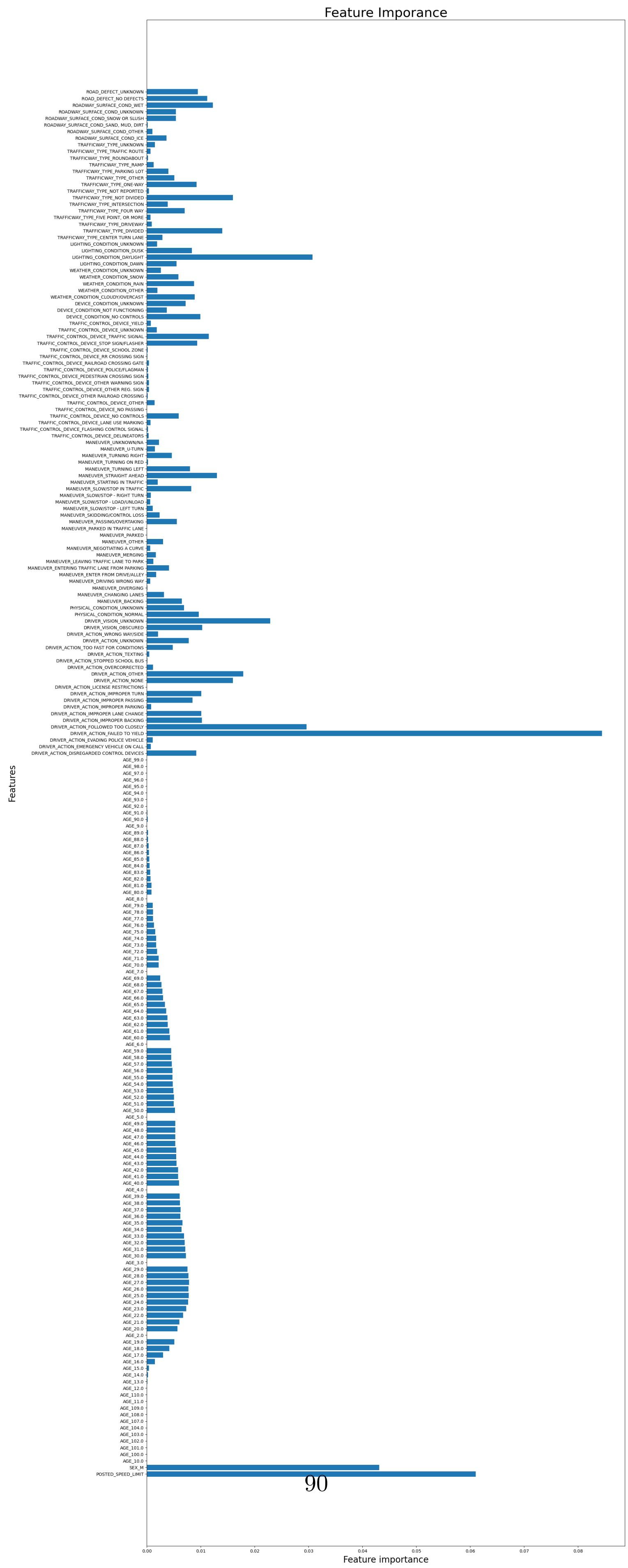
Feature Importance Using Random Forest

[81]:
```
# Instantiate and fit the model
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train_scaled, y_train)
```

[81]:
```
RandomForestClassifier()
```

[82]:
```
# Get the column names of the features in the training data
labels = list(X_train.columns)
```

[83]:
```
# Plot feature importances
n_features = X_train_scaled.shape[1]
plt.figure(figsize=(20,50))
plt.barh(range(n_features), rfc.feature_importances_, align='center')
plt.yticks(np.arange(n_features),labels=labels)
plt.title('Feature Imporance', fontsize=30, pad=5)
plt.xlabel('Feature importance', fontsize=20, labelpad=5)
plt.ylabel('Features', fontsize=20)
plt.tight_layout()
plt.savefig(r'images\feature imp main.png', bbox_inches='tight')
```


Feature Imporance
Features
Feature importance
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
ROAD_DEFECT_UNKNOWN
ROAD_DEFECT_NO DEFECTS
ROADWAY_SURFACE_COND_WET
ROADWAY_SURFACE_COND_UNKNOWN
ROADWAY_SURFACE_COND_SNOW OR SLUSH
ROADWAY_SURFACE_COND_SAND, MUD, DIRT
ROADWAY_SURFACE_COND_OTHER
ROADWAY_SURFACE_COND_ICE
TRAFFICWAY_TYPE_UNKNOWN
TRAFFICWAY_TYPE_TRAFFIC ROUTE
TRAFFICWAY_TYPE_ROUNDABOUT
TRAFFICWAY_TYPE_RAMP
TRAFFICWAY_TYPE_PARKING LOT
TRAFFICWAY_TYPE_OTHER
TRAFFICWAY_TYPE_ONE-WAY
TRAFFICWAY_TYPE_NOT REPORTED
TRAFFICWAY_TYPE_NOT DIVIDED
TRAFFICWAY_TYPE_INTERSECTION
TRAFFICWAY_TYPE_FOUR WAY
TRAFFICWAY_TYPE_FIVE POINT, OR MORE
TRAFFICWAY_TYPE_DRIVEWAY
TRAFFICWAY_TYPE_DIVIDED
TRAFFICWAY_TYPE_CENTER TURN LANE
LIGHTING_CONDITION_UNKNOWN
LIGHTING_CONDITION_DUSK
LIGHTING_CONDITION_DAYLIGHT
LIGHTING_CONDITION_DAWN
WEATHER_CONDITION_UNKNOWN
WEATHER_CONDITION_SNOW
WEATHER_CONDITION_RAIN
WEATHER_CONDITION_OTHER
WEATHER_CONDITION_CLOUDY/OVERCAST
DEVICE_CONDITION_UNKNOWN
DEVICE_CONDITION_NOT FUNCTIONING
DEVICE_CONDITION_NO CONTROLS
TRAFFIC_CONTROL_DEVICE_YIELD
TRAFFIC_CONTROL_DEVICE_UNKNOWN
TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL
TRAFFIC_CONTROL_DEVICE_STOP SIGN/FLASHER
TRAFFIC_CONTROL_DEVICE_SCHOOL ZONE
TRAFFIC_CONTROL_DEVICE_RR CROSSING SIGN
TRAFFIC_CONTROL_DEVICE_RAILROAD CROSSING GATE
TRAFFIC_CONTROL_DEVICE_POLICE/FLAGMAN
TRAFFIC_CONTROL_DEVICE_PEDESTRIAN CROSSING SIGN
TRAFFIC_CONTROL_DEVICE_OTHER WARNING SIGN
TRAFFIC_CONTROL_DEVICE_OTHER REG. SIGN
TRAFFIC_CONTROL_DEVICE_OTHER RAILROAD CROSSING
TRAFFIC_CONTROL_DEVICE_OTHER
TRAFFIC_CONTROL_DEVICE_NO PASSING
TRAFFIC_CONTROL_DEVICE_NO CONTROLS
TRAFFIC_CONTROL_DEVICE_LANE USE MARKING
TRAFFIC_CONTROL_DEVICE_FLASHING CONTROL SIGNAL
TRAFFIC_CONTROL_DEVICE_DELINEATORS
MANEUVER_UNKNOWN/NA
MANEUVER_U-TURN
MANEUVER_TURNING RIGHT
MANEUVER_TURNING ON RED
MANEUVER_TURNING LEFT
MANEUVER_STRAIGHT AHEAD
MANEUVER_STARTING IN TRAFFIC
MANEUVER_SLOW/STOP IN TRAFFIC
MANEUVER_SLOW/STOP - RIGHT TURN
MANEUVER_SLOW/STOP - LOAD/UNLOAD
MANEUVER_SLOW/STOP - LEFT TURN
MANEUVER_SKIDDING/CONTROL LOSS
MANEUVER_PASSING/OVERTAKING
MANEUVER_PARKED IN TRAFFIC LANE
MANEUVER_PARKED
MANEUVER_OTHER
MANEUVER_NEGOTIATING A CURVE
MANEUVER_MERGING
MANEUVER_LEAVING TRAFFIC LANE TO PARK
MANEUVER_ENTERING TRAFFIC LANE FROM PARKING
MANEUVER_ENTER FROM DRIVE/ALLEY
MANEUVER_DRIVING WRONG WAY
MANEUVER_DIVERGING
MANEUVER_CHANGING LANES
MANEUVER_BACKING
PHYSICAL_CONDITION_UNKNOWN
PHYSICAL_CONDITION_NORMAL
DRIVER_VISION_UNKNOWN
DRIVER_VISION_OBSCURED
DRIVER_ACTION_WRONG WAY/SIDE
DRIVER_ACTION_UNKNOWN
DRIVER_ACTION_TOO FAST FOR CONDITIONS
DRIVER_ACTION_TEXTING
DRIVER_ACTION_STOPPED SCHOOL BUS
DRIVER_ACTION_OVERCORRECTED
DRIVER_ACTION_OTHER
DRIVER_ACTION_NONE
DRIVER_ACTION_LICENSE RESTRICTIONS
DRIVER_ACTION_IMPROPER TURN
DRIVER_ACTION_IMPROPER PASSING
DRIVER_ACTION_IMPROPER PARKING
DRIVER_ACTION_IMPROPER LANE CHANGE
DRIVER_ACTION_IMPROPER BACKING
DRIVER_ACTION_FOLLOWED TOO CLOSELY
DRIVER_ACTION_FAILED TO YIELD
DRIVER_ACTION_EVADING POLICE VEHICLE
DRIVER_ACTION_EMERGENCY VEHICLE ON CALL
DRIVER_ACTION_DISREGARDED CONTROL DEVICES
AGE_99.0
AGE_98.0
AGE_97.0
AGE_96.0
AGE_95.0
AGE_94.0
AGE_93.0
AGE_92.0
AGE_91.0
AGE_90.0
AGE_9.0
AGE_89.0
AGE_88.0
AGE_87.0
AGE_86.0
AGE_85.0
AGE_84.0
AGE_83.0
AGE_82.0
AGE_81.0
AGE_80.0
AGE_8.0
AGE_79.0
AGE_78.0
AGE_77.0
AGE_76.0
AGE_75.0
AGE_74.0
AGE_73.0
AGE_72.0
AGE_71.0
AGE_70.0
AGE_7.0
AGE_69.0
AGE_68.0
AGE_67.0
AGE_66.0
AGE_65.0
AGE_64.0
AGE_63.0
AGE_62.0
AGE_61.0
AGE_60.0
AGE_6.0
AGE_59.0
AGE_58.0
AGE_57.0
AGE_56.0
AGE_55.0
AGE_54.0
AGE_53.0
AGE_52.0
AGE_51.0
AGE_50.0
AGE_5.0
AGE_49.0
AGE_48.0
AGE_47.0
AGE_46.0
AGE_45.0
AGE_44.0
AGE_43.0
AGE_42.0
AGE_41.0
AGE_40.0
AGE_4.0
AGE_39.0
AGE_38.0
AGE_37.0
AGE_36.0
AGE_35.0
AGE_34.0
AGE_33.0
AGE_32.0
AGE_31.0
AGE_30.0
AGE_3.0
AGE_29.0
AGE_28.0
AGE_27.0
AGE_26.0
AGE_25.0
AGE_24.0
AGE_23.0
AGE_22.0
AGE_21.0
AGE_20.0
AGE_2.0
AGE_19.0
AGE_18.0
AGE_17.0
AGE_16.0
AGE_15.0
AGE_14.0
AGE_13.0
AGE_12.0
AGE_110.0
AGE_11.0
AGE_109.0
AGE_108.0
AGE_107.0
AGE_104.0
AGE_103.0
AGE_102.0
AGE_101.0
AGE_100.0
AGE_10.0
SEX_M
POSTED_SPEED_LIMIT

We can use the feature importance mean to use as a cut-off point for imporantant vs non-important features.

[84]:
```
#select features based on their importance scores using the mean value of␣
↪feature importances
selected_features = X_train.columns[rfc.feature_importances_ > rfc.
↪feature_importances_.mean()]
print(selected_features)
```

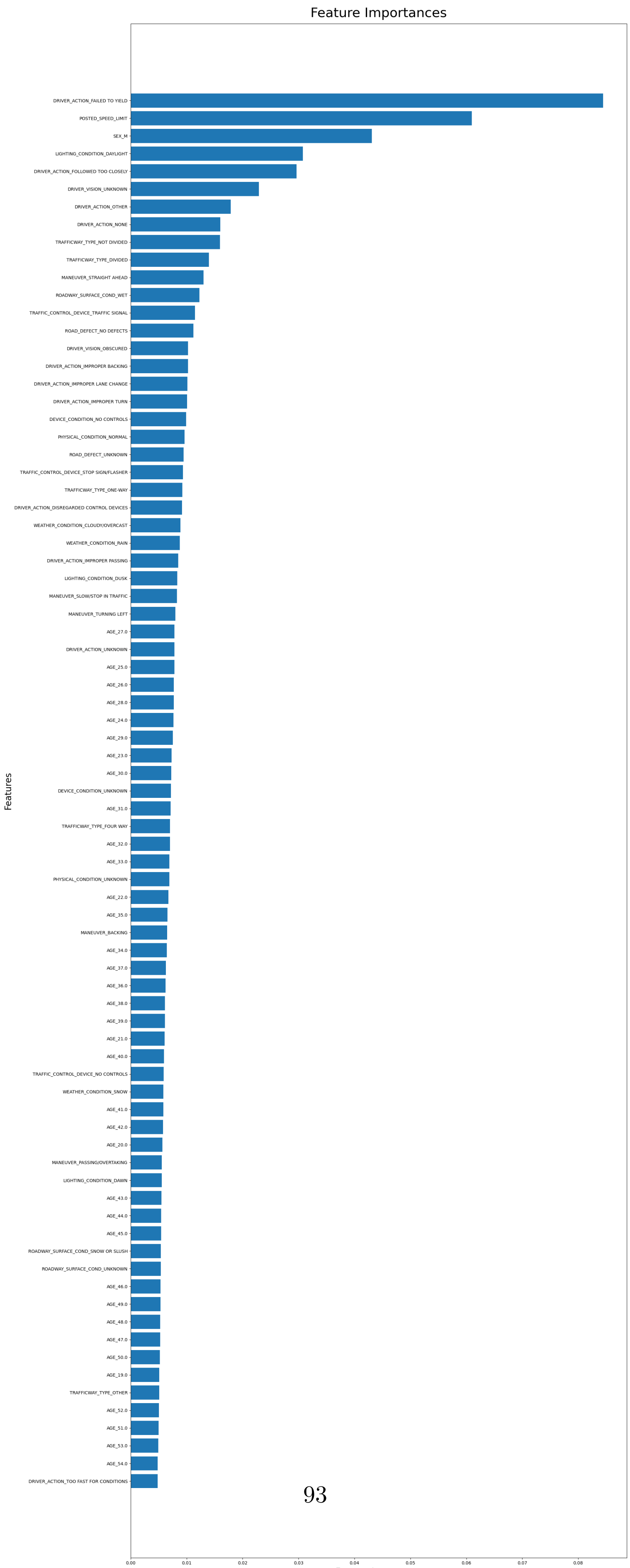
```
Index(['POSTED_SPEED_LIMIT', 'SEX_M', 'AGE_19.0', 'AGE_20.0', 'AGE_21.0',
       'AGE_22.0', 'AGE_23.0', 'AGE_24.0', 'AGE_25.0', 'AGE_26.0', 'AGE_27.0',
       'AGE_28.0', 'AGE_29.0', 'AGE_30.0', 'AGE_31.0', 'AGE_32.0', 'AGE_33.0',
       'AGE_34.0', 'AGE_35.0', 'AGE_36.0', 'AGE_37.0', 'AGE_38.0', 'AGE_39.0',
       'AGE_40.0', 'AGE_41.0', 'AGE_42.0', 'AGE_43.0', 'AGE_44.0', 'AGE_45.0',
       'AGE_46.0', 'AGE_47.0', 'AGE_48.0', 'AGE_49.0', 'AGE_50.0', 'AGE_51.0',
       'AGE_52.0', 'AGE_53.0', 'AGE_54.0',
       'DRIVER_ACTION_DISREGARDED CONTROL DEVICES',
       'DRIVER_ACTION_FAILED TO YIELD', 'DRIVER_ACTION_FOLLOWED TOO CLOSELY',
       'DRIVER_ACTION_IMPROPER BACKING', 'DRIVER_ACTION_IMPROPER LANE CHANGE',
       'DRIVER_ACTION_IMPROPER PASSING', 'DRIVER_ACTION_IMPROPER TURN',
       'DRIVER_ACTION_NONE', 'DRIVER_ACTION_OTHER',
       'DRIVER_ACTION_TOO FAST FOR CONDITIONS', 'DRIVER_ACTION_UNKNOWN',
       'DRIVER_VISION_OBSCURED', 'DRIVER_VISION_UNKNOWN',
       'PHYSICAL_CONDITION_NORMAL', 'PHYSICAL_CONDITION_UNKNOWN',
       'MANEUVER_BACKING', 'MANEUVER_PASSING/OVERTAKING',
       'MANEUVER_SLOW/STOP IN TRAFFIC', 'MANEUVER_STRAIGHT AHEAD',
       'MANEUVER_TURNING LEFT', 'TRAFFIC_CONTROL_DEVICE_NO CONTROLS',
       'TRAFFIC_CONTROL_DEVICE_STOP SIGN/FLASHER',
       'TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL', 'DEVICE_CONDITION_NO CONTROLS',
       'DEVICE_CONDITION_UNKNOWN', 'WEATHER_CONDITION_CLOUDY/OVERCAST',
       'WEATHER_CONDITION_RAIN', 'WEATHER_CONDITION_SNOW',
       'LIGHTING_CONDITION_DAWN', 'LIGHTING_CONDITION_DAYLIGHT',
       'LIGHTING_CONDITION_DUSK', 'TRAFFICWAY_TYPE_DIVIDED',
       'TRAFFICWAY_TYPE_FOUR WAY', 'TRAFFICWAY_TYPE_NOT DIVIDED',
       'TRAFFICWAY_TYPE_ONE-WAY', 'TRAFFICWAY_TYPE_OTHER',
       'ROADWAY_SURFACE_COND_SNOW OR SLUSH', 'ROADWAY_SURFACE_COND_UNKNOWN',
       'ROADWAY_SURFACE_COND_WET', 'ROAD_DEFECT_NO DEFECTS',
       'ROAD_DEFECT_UNKNOWN'],
      dtype='object')
```

[85]:
```
# filter the feature importance scores based on their values, selecting only␣
↪the scores that are above the mean
scores = rfc.feature_importances_
selected_features_scores = scores[rfc.feature_importances_ > rfc.
↪feature_importances_.mean()]
```

```
labels_selected = list(selected_features)
```

[86]:

```
# Plot feature importances
n_features = len(selected_features)

# Sort the selected features and their scores in ascending order
sorted_indices = np.argsort(selected_features_scores)
sorted_features = np.array(labels_selected)[sorted_indices]
sorted_scores = selected_features_scores[sorted_indices]

plt.figure(figsize=(20, 50))
plt.barh(range(n_features), sorted_scores, align='center')
plt.yticks(np.arange(n_features), labels=sorted_features)
plt.title('Feature Importances', fontsize=30, pad=15)
plt.xlabel('Feature Importance', fontsize=20, labelpad=5)
plt.ylabel('Features', fontsize=20)
plt.tight_layout()
plt.show()
plt.savefig(r'images\ feature imp sort.png', bbox_inches='tight');
```



Feature Importances
DRIVER_ACTION_FAILED TO YIELD
POSTED_SPEED_LIMIT
SEX_M
LIGHTING_CONDITION_DAYLIGHT
DRIVER_ACTION_FOLLOWED TOO CLOSELY
DRIVER_VISION_UNKNOWN
DRIVER_ACTION_OTHER
DRIVER_ACTION_NONE
TRAFFICWAY_TYPE_NOT DIVIDED
TRAFFICWAY_TYPE_DIVIDED
MANEUVER_STRAIGHT AHEAD
ROADWAY_SURFACE_COND_WET
TRAFFIC_CONTROL_DEVICE_TRAFFIC SIGNAL
ROAD_DEFECT_NO DEFECTS
DRIVER_VISION_OBSCURED
DRIVER_ACTION_IMPROPER BACKING
DRIVER_ACTION_IMPROPER LANE CHANGE
DRIVER_ACTION_IMPROPER TURN
DEVICE_CONDITION_NO CONTROLS
PHYSICAL_CONDITION_NORMAL
ROAD_DEFECT_UNKNOWN
TRAFFIC_CONTROL_DEVICE_STOP SIGN/FLASHER
TRAFFICWAY_TYPE_ONE-WAY
DRIVER_ACTION_DISREGARDED CONTROL DEVICES
WEATHER_CONDITION_CLOUDY/OVERCAST
WEATHER_CONDITION_RAIN
DRIVER_ACTION_IMPROPER PASSING
LIGHTING_CONDITION_DUSK
MANEUVER_SLOW/STOP IN TRAFFIC
MANEUVER_TURNING LEFT
AGE_27.0
DRIVER_ACTION_UNKNOWN
AGE_25.0
AGE_26.0
AGE_28.0
AGE_24.0
AGE_29.0
AGE_23.0
AGE_30.0
DEVICE_CONDITION_UNKNOWN
AGE_31.0
TRAFFICWAY_TYPE_FOUR WAY
AGE_32.0
AGE_33.0
PHYSICAL_CONDITION_UNKNOWN
AGE_22.0
AGE_35.0
MANEUVER_BACKING
AGE_34.0
AGE_37.0
AGE_36.0
AGE_38.0
AGE_39.0
AGE_21.0
AGE_40.0
TRAFFIC_CONTROL_DEVICE_NO CONTROLS
WEATHER_CONDITION_SNOW
AGE_41.0
AGE_42.0
AGE_20.0
MANEUVER_PASSING/OVERTAKING
LIGHTING_CONDITION_DAWN
AGE_43.0
AGE_44.0
AGE_45.0
ROADWAY_SURFACE_COND_SNOW OR SLUSH
ROADWAY_SURFACE_COND_UNKNOWN
AGE_46.0
AGE_49.0
AGE_48.0
AGE_47.0
AGE_50.0
AGE_19.0
TRAFFICWAY_TYPE_OTHER
AGE_52.0
AGE_51.0
AGE_53.0
AGE_54.0
DRIVER_ACTION_TOO FAST FOR CONDITIONS
Features
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Feature Importance

<Figure size 640x480 with 0 Axes>

By selecting features of importance that are higher than the mean, we can gain valuable insights into the primary causes of accidents. When comparing this list with the plot above, we can observe distinct characteristics related to the accidents. These selected features highlight the factors that contribute significantly to the occurrence of accidents and provide valuable information for understanding the underlying causes. * The feature "DRIVER_ACTION_FAILED_TO_YIELD" stands out as the most important feature, followed by "POSTED_SPEED_LIMIT" and "SEX_M". These features play a crucial role in predicting the type of accidents associated with specific driver actions, whether they are classified as "INTENTIONAL" or "UNINTENTIONAL".

- The inclusion of features such as "TRAFFIC_CONTROL_DEVICE", "WEATHER_CONDITION", "TRAFFICWAY_TYPE", and "ROADWAY_SURFACE_COND" in the list of important features indicates that these characteristics are strong predictors of accidents. These features carry significant weight in determining the occurrence and severity of accidents.
- The data suggests that drivers are at a higher risk of accidents during their mid-20s, particularly at the age of 25. As drivers progress into older age groups, the likelihood of accidents gradually decreases. This pattern implies that drivers in their mid-20s may exhibit certain characteristics or behaviors that contribute to a higher accident risk compared to other age groups.

Dimensionality Reduction Dimensionality Reduction: PCA helps in reducing the dimensionality of the dataset by transforming the original variables into a new set of uncorrelated variables called principal components. This is particularly useful when dealing with a high-dimensional dataset with a large number of predictors. By reducing the dimensionality, we can simplify the analysis and visualization of the data.

[87]:
```python
# Initialize PCA with 80% explained_variance_ratio_

pca = PCA(0.80)

# Fit PCA on the feature data
X_train_transformed = pca.fit_transform(X_train_scaled)
X_test_transformed = pca.transform(X_test_scaled)

# Access the explained variance ratio of the components
explained_variance_ratio = pca.explained_variance_ratio_

# Print the explained variance ratio of each component
for i, ratio in enumerate(explained_variance_ratio):
    print(f"Explained Variance Ratio for Component {i+1}: {ratio}")
```

```
Explained Variance Ratio for Component 1: 0.01759738599332751
Explained Variance Ratio for Component 2: 0.015507362609891797
Explained Variance Ratio for Component 3: 0.010597468054327633
```

```
Explained Variance Ratio for Component 4: 0.009996300090328961
Explained Variance Ratio for Component 5: 0.009114213338883207
Explained Variance Ratio for Component 6: 0.008664931475144189
Explained Variance Ratio for Component 7: 0.008480502188193968
Explained Variance Ratio for Component 8: 0.008180021416307495
Explained Variance Ratio for Component 9: 0.0076022387478848754
Explained Variance Ratio for Component 10: 0.007351437035559732
Explained Variance Ratio for Component 11: 0.007155961166131913
Explained Variance Ratio for Component 12: 0.007049366249119269
Explained Variance Ratio for Component 13: 0.006798580723836198
Explained Variance Ratio for Component 14: 0.006608684446721035
Explained Variance Ratio for Component 15: 0.006470931948170849
Explained Variance Ratio for Component 16: 0.006302547186115024
Explained Variance Ratio for Component 17: 0.006260202226086329
Explained Variance Ratio for Component 18: 0.006054917779727796
Explained Variance Ratio for Component 19: 0.005931475670390073
Explained Variance Ratio for Component 20: 0.005675580215954186
Explained Variance Ratio for Component 21: 0.005530636676305377
Explained Variance Ratio for Component 22: 0.005480804919432489
Explained Variance Ratio for Component 23: 0.005342101121640716
Explained Variance Ratio for Component 24: 0.005291208250779985
Explained Variance Ratio for Component 25: 0.005232133155635328
Explained Variance Ratio for Component 26: 0.005203310622094665
Explained Variance Ratio for Component 27: 0.0051472856295854425
Explained Variance Ratio for Component 28: 0.005129863168033522
Explained Variance Ratio for Component 29: 0.005090608425382389
Explained Variance Ratio for Component 30: 0.005084075024891853
Explained Variance Ratio for Component 31: 0.005054416025496725
Explained Variance Ratio for Component 32: 0.0050348644818663454
Explained Variance Ratio for Component 33: 0.005026265021318582
Explained Variance Ratio for Component 34: 0.005018724188779451
Explained Variance Ratio for Component 35: 0.00499699631620705
Explained Variance Ratio for Component 36: 0.004993684353292255
Explained Variance Ratio for Component 37: 0.004984535734936757
Explained Variance Ratio for Component 38: 0.004981688862912573
Explained Variance Ratio for Component 39: 0.004977248583412778
Explained Variance Ratio for Component 40: 0.004970047917431993
Explained Variance Ratio for Component 41: 0.004964851461680828
Explained Variance Ratio for Component 42: 0.004963885273531973
Explained Variance Ratio for Component 43: 0.004963221780478917
Explained Variance Ratio for Component 44: 0.004961758201421043
Explained Variance Ratio for Component 45: 0.004953447562172082
Explained Variance Ratio for Component 46: 0.0049512132834056664
Explained Variance Ratio for Component 47: 0.00494995072336451
Explained Variance Ratio for Component 48: 0.004944551817437878
Explained Variance Ratio for Component 49: 0.004943144884821949
Explained Variance Ratio for Component 50: 0.0049410688053069815
Explained Variance Ratio for Component 51: 0.004935828856117358
```

```
Explained Variance Ratio for Component 52: 0.004934978180846001
Explained Variance Ratio for Component 53: 0.004933774517906857
Explained Variance Ratio for Component 54: 0.004930892459118148
Explained Variance Ratio for Component 55: 0.004926522020333143
Explained Variance Ratio for Component 56: 0.004923216167805426
Explained Variance Ratio for Component 57: 0.004921408343677837
Explained Variance Ratio for Component 58: 0.0049195920102644616
Explained Variance Ratio for Component 59: 0.004917458698365174
Explained Variance Ratio for Component 60: 0.004915299685864923
Explained Variance Ratio for Component 61: 0.004911487281282694
Explained Variance Ratio for Component 62: 0.0049102815831642615
Explained Variance Ratio for Component 63: 0.004906754726989334
Explained Variance Ratio for Component 64: 0.004906245174743418
Explained Variance Ratio for Component 65: 0.0049014443834703774
Explained Variance Ratio for Component 66: 0.004900649743944233
Explained Variance Ratio for Component 67: 0.004899927200830894
Explained Variance Ratio for Component 68: 0.004898056452560343
Explained Variance Ratio for Component 69: 0.004897113113972502
Explained Variance Ratio for Component 70: 0.004894203955570386
Explained Variance Ratio for Component 71: 0.004891566534474734
Explained Variance Ratio for Component 72: 0.004890645523399225
Explained Variance Ratio for Component 73: 0.0048896490443076255
Explained Variance Ratio for Component 74: 0.0048884667025115465
Explained Variance Ratio for Component 75: 0.004887905776029335
Explained Variance Ratio for Component 76: 0.004886932380364194
Explained Variance Ratio for Component 77: 0.004885486710843424
Explained Variance Ratio for Component 78: 0.004881052543684711
Explained Variance Ratio for Component 79: 0.004879803003791442
Explained Variance Ratio for Component 80: 0.0048771089445991175
Explained Variance Ratio for Component 81: 0.0048752003585549805
Explained Variance Ratio for Component 82: 0.00487163011811799
Explained Variance Ratio for Component 83: 0.0048702588210326175
Explained Variance Ratio for Component 84: 0.004866168872650261
Explained Variance Ratio for Component 85: 0.004865582266418006
Explained Variance Ratio for Component 86: 0.004863604628950709
Explained Variance Ratio for Component 87: 0.0048623655286688
Explained Variance Ratio for Component 88: 0.00486025313090437
Explained Variance Ratio for Component 89: 0.0048596606271879275
Explained Variance Ratio for Component 90: 0.004857683175070953
Explained Variance Ratio for Component 91: 0.004856230052915199
Explained Variance Ratio for Component 92: 0.00485388332155238
Explained Variance Ratio for Component 93: 0.004851133980931975
Explained Variance Ratio for Component 94: 0.00484780168700111
Explained Variance Ratio for Component 95: 0.004845738660343837
Explained Variance Ratio for Component 96: 0.004843225355413253
Explained Variance Ratio for Component 97: 0.004842704657967631
Explained Variance Ratio for Component 98: 0.0048415519668107265
Explained Variance Ratio for Component 99: 0.004840444100792638
```

```
Explained Variance Ratio for Component 100: 0.004838763527005313
Explained Variance Ratio for Component 101: 0.004836879053058705
Explained Variance Ratio for Component 102: 0.004834845415201395
Explained Variance Ratio for Component 103: 0.004831040321485584
Explained Variance Ratio for Component 104: 0.004830549283297204
Explained Variance Ratio for Component 105: 0.004828679894982066
Explained Variance Ratio for Component 106: 0.004828135127684226
Explained Variance Ratio for Component 107: 0.004825274761365809
Explained Variance Ratio for Component 108: 0.004823091612232695
Explained Variance Ratio for Component 109: 0.004820831032965238
Explained Variance Ratio for Component 110: 0.004819229435282448
Explained Variance Ratio for Component 111: 0.004818599760029858
Explained Variance Ratio for Component 112: 0.004817383935635121
Explained Variance Ratio for Component 113: 0.004815401970914495
Explained Variance Ratio for Component 114: 0.004814164308303282
Explained Variance Ratio for Component 115: 0.004813335708420136
Explained Variance Ratio for Component 116: 0.004812322774004278
Explained Variance Ratio for Component 117: 0.00481198478715714
Explained Variance Ratio for Component 118: 0.004811247890877912
Explained Variance Ratio for Component 119: 0.004810260002060307
Explained Variance Ratio for Component 120: 0.004809647009756988
Explained Variance Ratio for Component 121: 0.004809416698439374
Explained Variance Ratio for Component 122: 0.004808743031131672
Explained Variance Ratio for Component 123: 0.004808657146694328
Explained Variance Ratio for Component 124: 0.004808561536582684
Explained Variance Ratio for Component 125: 0.00480799590704332
Explained Variance Ratio for Component 126: 0.004807637780542625
Explained Variance Ratio for Component 127: 0.004807581751402614
Explained Variance Ratio for Component 128: 0.004807425472965141
Explained Variance Ratio for Component 129: 0.004807021511716465
Explained Variance Ratio for Component 130: 0.004806691471877418
Explained Variance Ratio for Component 131: 0.0048055099370688985
Explained Variance Ratio for Component 132: 0.004804521779370376
Explained Variance Ratio for Component 133: 0.004803858478127186
Explained Variance Ratio for Component 134: 0.00480333871817911
Explained Variance Ratio for Component 135: 0.004802747554179915
Explained Variance Ratio for Component 136: 0.004800999579666919
Explained Variance Ratio for Component 137: 0.004799437225416373
Explained Variance Ratio for Component 138: 0.004798111464102305
Explained Variance Ratio for Component 139: 0.004796941072398516
Explained Variance Ratio for Component 140: 0.004794138066195897
Explained Variance Ratio for Component 141: 0.004791924937383968
Explained Variance Ratio for Component 142: 0.004791419675118967
Explained Variance Ratio for Component 143: 0.0047841638307921895
Explained Variance Ratio for Component 144: 0.004781775919658363
Explained Variance Ratio for Component 145: 0.0047780069357003276
Explained Variance Ratio for Component 146: 0.004775834657385342
Explained Variance Ratio for Component 147: 0.00477199063540235
```

```
Explained Variance Ratio for Component 148: 0.004768759341162119
Explained Variance Ratio for Component 149: 0.004765983577003185
Explained Variance Ratio for Component 150: 0.004763441096372948
```

After applying Principal Component Analysis (PCA), the number of predictor columns in our dataset has been reduced from 210 to 182. PCA helps to identify and capture the most important information in the data by creating new variables called principal components. These principal components are linear combinations of the original predictor variables and are chosen in such a way that they explain the maximum amount of variation in the data. By using PCA, we have effectively reduced the dimensionality of the dataset while retaining a significant amount of the information present in the original predictors.

9 Modelling and Evaluation

Creating a classifier for our business problem I will create a classifier that can distinguish between "Unintentional" and "Intentional" accidents based on the available predictors in our dataset, we can use machine learning techniques. The classifier will be trained on the historical accident data, where each accident is labeled as either "Unintentional" or "Intentional" based on the contributing factors.

I will look into the following models:

```
* Logistic Regression

* Decesion Tree

* Random Forest

* XG Boost
```

1. Logistic Regression

[88]:
```
import os
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix, roc_curve,␣
↪roc_auc_score
#  define a function to valuate the classification model using various metrics␣
↪and generate visualizations.
def evaluate_classification(model, X_train_transformed, X_test_transformed,␣
↪y_train, y_test, classes=None, normalize='true', cmap='Blues_r', label='',␣
↪save_dir='plots'):

    # Create save directory if it doesn't exist
    os.makedirs(save_dir, exist_ok=True)

    # retrieve predictions for train and test data
    y_pred_train = model.predict(X_train_transformed)
```

```
    y_pred_test = model.predict(X_test_transformed)

    # print training classification report
    header = label + " CLASSIFICATION REPORT TRAINING "
    dashes = "---" * 20
    print(dashes, header, dashes, sep='\n')
    print(classification_report(y_train, y_pred_train, target_names=classes))

    # calculate confusion matrix for training data
    cm_train = confusion_matrix(y_train, y_pred_train)
    cm_train_norm = cm_train / cm_train.sum(axis=1)[:, np.newaxis] if normalize␣
↪== 'true' else cm_train

    # print testing classification report
    header_ = label + " CLASSIFICATION REPORT TESTING "
    print(dashes, header_, dashes, sep='\n')
    print(classification_report(y_test, y_pred_test, target_names=classes))

    # calculate confusion matrix for testing data
    cm_test = confusion_matrix(y_test, y_pred_test)
    cm_test_norm = cm_test / cm_test.sum(axis=1)[:, np.newaxis] if normalize ==␣
↪'true' else cm_test

    # Create a combined figure for training and testing plots
    fig, axes = plt.subplots(figsize=(12, 4), ncols=4)

    # plot confusion matrix for training data
    sns.heatmap(cm_train_norm, annot=True, fmt='.2f', cmap=cmap, ax=axes[0])
    axes[0].set(title='Confusion Matrix Training', xlabel='Predicted Labels',␣
↪ylabel='True Labels')

    # plot ROC curve for training data
    fpr_train, tpr_train, _ = roc_curve(y_train, model.
↪predict_proba(X_train_transformed)[:, 1])
    roc_auc_train = roc_auc_score(y_train, model.
↪predict_proba(X_train_transformed)[:, 1])
    axes[1].plot(fpr_train, tpr_train, label=f'AUC = {roc_auc_train:.2f}')
    axes[1].plot([0, 1], [0, 1], ls=':')
    axes[1].set(xlabel='False Positive Rate', ylabel='True Positive Rate',
                title='Receiver Operating Characteristic Training')
    axes[1].legend(loc='lower right')

    # plot confusion matrix for testing data
    sns.heatmap(cm_test_norm, annot=True, fmt='.2f', cmap=cmap, ax=axes[2])
    axes[2].set(title='Confusion Matrix Testing', xlabel='Predicted Labels',␣
↪ylabel='True Labels')
```

```
    # plot ROC curve for testing data
    fpr_test, tpr_test, _ = roc_curve(y_test, model.
 ↪predict_proba(X_test_transformed)[:, 1])
    roc_auc_test = roc_auc_score(y_test, model.
 ↪predict_proba(X_test_transformed)[:, 1])
    axes[3].plot(fpr_test, tpr_test, label=f'AUC = {roc_auc_test:.2f}')
    axes[3].plot([0, 1], [0, 1], ls=':')
    axes[3].set(xlabel='False Positive Rate', ylabel='True Positive Rate',
                title='Receiver Operating Characteristic Testing')
    axes[3].legend(loc='lower right')

    # Adjust spacing between subplots
    plt.tight_layout(pad=2.0)

    # Save combined plots
    plt.savefig(os.path.join(save_dir, 'combined_plots.png'))

    plt.show()
```

[89]:
```
from sklearn.linear_model import LogisticRegression
# Initialize the logistic regression model
logreg = LogisticRegression()
# Train the model
logreg.fit(X_train_transformed, y_train)
```

[89]:
```
LogisticRegression()
```

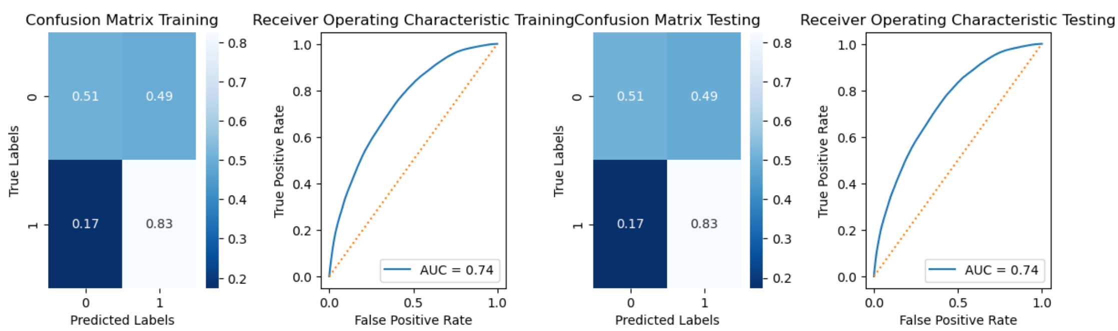
[90]:
```
# Make predictions on the test set
y_pred = logreg.predict(X_test_transformed)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Generate a classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.6889204463455888
              precision    recall  f1-score   support

           0       0.69      0.51      0.59     74958
           1       0.69      0.83      0.75     98271

    accuracy                           0.69    173229
   macro avg       0.69      0.67      0.67    173229
weighted avg       0.69      0.69      0.68    173229
```

[91]:

```
# classification report using function
evaluate_classification(logreg,X_train_transformed, X_test_transformed,␣
↪y_train, y_test, label = "Logistic Regression")
```

```
------------------------------------------------------------
Logistic Regression CLASSIFICATION REPORT TRAINING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.69      0.51      0.58    175263
           1       0.69      0.83      0.75    228937

    accuracy                           0.69    404200
   macro avg       0.69      0.67      0.67    404200
weighted avg       0.69      0.69      0.68    404200

------------------------------------------------------------
Logistic Regression CLASSIFICATION REPORT TESTING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.69      0.51      0.59     74958
           1       0.69      0.83      0.75     98271

    accuracy                           0.69    173229
   macro avg       0.69      0.67      0.67    173229
weighted avg       0.69      0.69      0.68    173229
```



The logistic regression model shows moderate performance on both the training and testing sets. It achieves an accuracy of 0.69 on both sets. The precision, recall, and F1-score for class 0 are 0.69, 0.51, and 0.59, respectively, indicating that the model performs moderately well in predicting instances of class 0. Similarly, for class 1, the precision, recall, and F1-score are 0.69, 0.83, and 0.75, respectively, suggesting that the model performs relatively well in predicting instances of class

1.

Decision Tree Classifier Our initial model for classification will be a Decision Tree Classifier with a tree depth of 3. This means that the decision tree will have a maximum depth of 3 levels, allowing it to make decisions based on three predictor variables at each step. The decision tree algorithm uses a tree-like structure to classify the data based on the features or predictors. By limiting the tree depth to 3, we aim to strike a balance between model complexity and interpretability. This base model will serve as a starting point for further analysis and model improvement.

[93]:
```
# Initializes a DecisionTreeClassifier
tree_clf= DecisionTreeClassifier(criterion='gini', max_depth=3)
```

[94]:
```
#Fit the model
tree_clf.fit(X_train_transformed, y_train)
```

[94]:
```
DecisionTreeClassifier(max_depth=3)
```

[ ]:
```
from sklearn import tree
import graphviz
# plot the three
tree.plot_tree(tree_clf)
```

[95]:
```
# Create an array to make predictions for train and test data
y_pred_train = tree_clf.predict(X_train_transformed)
y_pred_test = tree_clf.predict(X_test_transformed)
```

[96]:
```
# Calculate accuracy
train_acc = accuracy_score(y_train,y_pred_train) * 100
test_acc = accuracy_score(y_test, y_pred_test) * 100
print('Train accuracy is :{0}'.format(train_acc))
print('Test accuracy is :{0}'.format(test_acc))

# Check the AUC for predictions
fpr, tpr, thresholds = roc_curve(y_test, y_pred_test)
roc_auc = auc(fpr, tpr)
print('\nAUC is :{0}'.format(round(roc_auc, 2)))

# Create and print a confusion matrix
print('\nConfusion Matrix')
print('----------------')
pd.crosstab(y_test, y_pred_test, rownames=['True'], colnames=['Predicted'],␣
↪margins=True)
```

```
Train accuracy is :64.33077684314695
Test accuracy is :64.37317077394663

AUC is :0.61
```

```
Confusion Matrix
----------------
```

[96]:

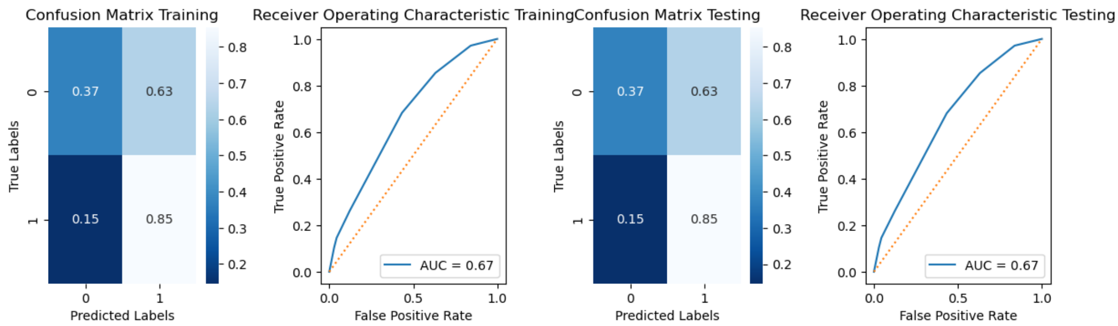
```
Predicted      0       1     All
True
0          27653   47305   74958
1          14411   83860   98271
All        42064  131165  173229
```

The close similarity between the train and test accuracy values indicates that the model is performing well and is likely to generalize well to unseen data.The model is not overfitting or underfitting the training data, as it achieves similar performance on both the training and testing datasets.

[97]:

```
# Call the evaluate_classification function with the desired parameters
evaluate_classification(tree_clf, X_train_transformed, X_test_transformed,␣
 ↪y_train, y_test, label='Decision Tree', save_dir='images')
```

```
------------------------------------------------------------
Decision Tree CLASSIFICATION REPORT TRAINING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.66      0.37      0.47    175263
           1       0.64      0.85      0.73    228937

    accuracy                           0.64    404200
   macro avg       0.65      0.61      0.60    404200
weighted avg       0.65      0.64      0.62    404200

------------------------------------------------------------
Decision Tree CLASSIFICATION REPORT TESTING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.66      0.37      0.47     74958
           1       0.64      0.85      0.73     98271

    accuracy                           0.64    173229
   macro avg       0.65      0.61      0.60    173229
weighted avg       0.65      0.64      0.62    173229
```



Training Performance: The model achieved an accuracy of 0.64 on the training set, correctly classifying 64% of the instances in the training data. The precision and recall for class 0 are lower compared to class 1, indicating that the model struggles more in correctly identifying instances of class 0. The weighted average F1-score is 0.62, suggesting a moderate overall performance on the training set.

Testing Performance: On the testing set, the model achieved an accuracy of 0.65, correctly classifying 65% of the instances. Similar to the training set, the precision and recall for class 0 are lower compared to class 1. The weighted average F1-score is 0.62, indicating a moderate overall performance on the testing set.

Considering these observations, we can conclude that the Decision Tree model's performance is moderate. It shows a similar performance on both the training and testing sets, but with lower precision and recall for class 0. This suggests that the model might struggle in accurately identifying instances of class 0, potentially leading to a higher number of false negatives for this class.

Based on the ROC curve, the model's performance in predicting the classes is moderate. The AUC value of 0.67 indicates that there is a 67% chance that the model will correctly classify each target variable. This means that the model has some predictive power, but it is not highly accurate. There is still room for improvement in achieving more accurate predictions.

Re-grow the tree using entropy

[98]:
```
#Instantiate the model
dtc_entropy = DecisionTreeClassifier(criterion='entropy')

#Fit the model
dtc_entropy.fit(X_train_transformed, y_train)

# Make predictions for train and test data
y_pred_train_dtc1 = dtc_entropy.predict(X_train_transformed)
y_pred_test_dtc1 = dtc_entropy.predict(X_test_transformed)

# Calculate accuracy
train_acc1 = accuracy_score(y_train,y_pred_train_dtc1) * 100
test_acc1 = accuracy_score(y_test, y_pred_test_dtc1) * 100
print('Train accuracy is :{0}'.format(train_acc1))
```

```python
print('Test accuracy is :{0}'.format(test_acc1))

# Check the AUC for predictions
fpr1, tpr1, thresholds1 = roc_curve(y_test, y_pred_test)
roc_auc1 = auc(fpr1, tpr1)
print('\nAUC is :{0}'.format(round(roc_auc1, 2)))

# Create and print a confusion matrix
print('\nConfusion Matrix')
print('----------------')
pd.crosstab(y_test, y_pred_test, rownames=['True'], colnames=['Predicted'],␣
 ↪margins=True)
```

```
Train accuracy is :89.4188520534389
Test accuracy is :62.91209901344463

AUC is :0.61

Confusion Matrix
----------------
```

[98]:

Predicted	0	1	All
True			
0	27653	47305	74958
1	14411	83860	98271
All	42064	131165	173229

Despite using the entropy criterion for re-growing the decision tree, it did not significantly improve the model's performance.

Training Accuracy: The model achieved a training accuracy of approximately 89.42%, meaning it correctly classified 89.42% of the instances in the training data.

Test Accuracy: The model achieved a test accuracy of approximately 62.78%, indicating that it correctly classified 62.78% of the instances in the test data. Based on these results, we can conclude that the model has a relatively high accuracy on the training set (89.42%), but the accuracy drops significantly on the test set (62.78%). This indicates that the model may be overfitting the training data and is not generalizing well to unseen data. Additionally, the AUC of 0.61 suggests that the model's predictive performance is only slightly better

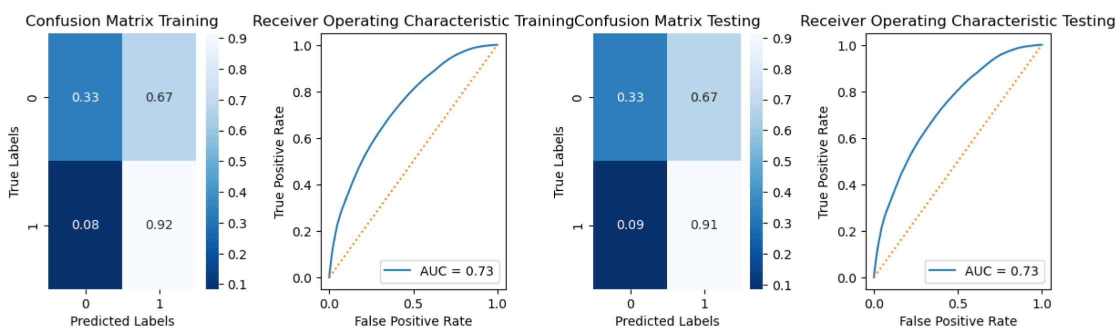
Random Forest

[103]:
```python
# Instantiate and fit the model
rf = RandomForestClassifier(n_estimators=100, max_depth= 5)
rf.fit(X_train_transformed, y_train)
```

[103]:
```
RandomForestClassifier(max_depth=5)
```

[104]:
```
# Call the pred_score function with Random forest classifier
evaluate_classification(rf,X_train_transformed, X_test_transformed, y_train,␣
 ↪y_test, label = 'Random Forest')
```

```
------------------------------------------------------------
Random Forest CLASSIFICATION REPORT TRAINING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.75      0.33      0.46    175263
           1       0.64      0.92      0.75    228937

    accuracy                           0.66    404200
   macro avg       0.70      0.62      0.61    404200
weighted avg       0.69      0.66      0.63    404200

------------------------------------------------------------
Random Forest CLASSIFICATION REPORT TESTING
------------------------------------------------------------
              precision    recall  f1-score   support

           0       0.75      0.33      0.46     74958
           1       0.64      0.91      0.75     98271

    accuracy                           0.66    173229
   macro avg       0.69      0.62      0.61    173229
weighted avg       0.69      0.66      0.63    173229
```



Training Performance: The model achieved an accuracy of 0.66 on the training set, correctly classifying 66% of the instances in the training data. The precision and recall for class 0 are lower compared to class 1, indicating that the model struggles more in correctly identifying instances of class 0. The weighted average F1-score is 0.64, suggesting a moderate overall performance on the training set.

Testing Performance: On the testing set, the model achieved an accuracy of 0.67, correctly classifying 67% of the instances. Similar to the training set, the precision and recall for class 0 are lower compared to class 1. The weighted average F1-score is 0.64, indicating a moderate overall performance on the testing set. #### Feature Importance with Random Forest

[105]:
```
# Plot feature importances
n_features = X_train_transformed.shape[1]
plt.figure(figsize=(20,50))
plt.barh(range(n_features), rf.feature_importances_, align='center')
plt.yticks(np.arange(n_features))
plt.title('Feature Imporance', fontsize=30, pad=5)
plt.xlabel('Feature importance', fontsize=20, labelpad=5)
plt.ylabel('Features', fontsize=20)
plt.tight_layout()
```

Feature Imporance
Features
Feature importance

Selecting best features

[106]:
```python
# Print the gini importance of each feature
for feature in zip(range(n_features), rf.feature_importances_):
    print(feature)
```

```
(0, 0.009909129063606808)
(1, 0.019473173019763073)
(2, 0.027333510101412987)
(3, 0.04705559875887393)
(4, 0.04750618233288741)
(5, 0.010914307799327307)
(6, 0.04295296992492601)
(7, 0.23890173420552221)
(8, 0.07763848586295058)
(9, 0.03708139313789361)
(10, 0.04531746807218481)
(11, 0.037575185326788614)
(12, 0.02906859509215385)
(13, 0.11439461044444564)
(14, 0.030188766204037012)
(15, 0.01963423469666918)
(16, 0.03089025851699974)
(17, 0.027577487346027788)
(18, 0.012548294258852394)
(19, 0.008936302055284227)
(20, 0.001882419826991263)
(21, 0.007390007891176465)
(22, 0.002615269629112092)
(23, 0.005004412272370161)
(24, 0.0041863408933955305)
(25, 0.00750771693347986)
(26, 0.0019185135165063127)
(27, 0.0032905767352200193)
(28, 0.004096715864799587)
(29, 0.004043053121756596)
(30, 0.0013743084022243015)
(31, 0.00022483156943289742)
(32, 0.0011668536626883313)
(33, 0.00070535917588184)
(34, 0.00010372936505759594)
(35, 7.465542895600672e-05)
(36, 0.00024389615678701 04)
(37, 0.0001311248124442541)
(38, 0.00010063586911560886)
```

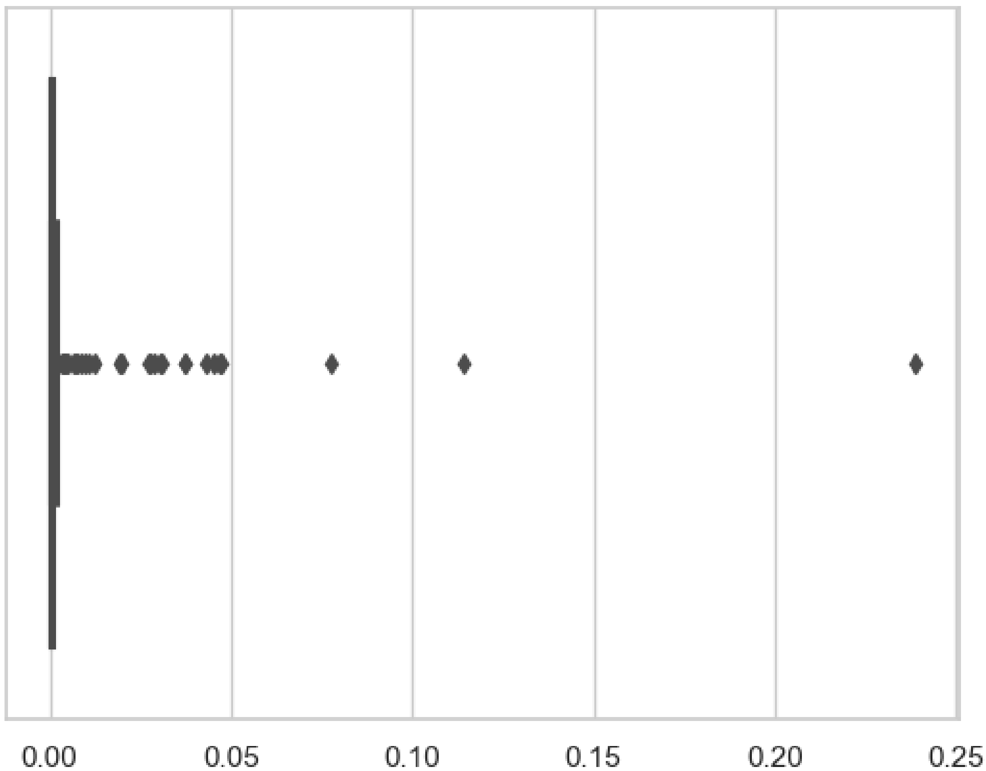
```
(39, 0.00011666751567842485)
(40, 1.180085010752859e-05)
(41, 0.00016545345785593007)
(42, 1.9971274863605582e-05)
(43, 0.004582682592077317)
(44, 2.183830920422099e-05)
(45, 0.0008472004062763126)
(46, 0.00040053425895694695)
(47, 6.200505185455757e-05)
(48, 0.0007001316059092725)
(49, 0.00018072301401184608)
(50, 3.889409711715324e-05)
(51, 1.8716149363083166e-05)
(52, 0.00012601956836424185)
(53, 8.935506093063257e-06)
(54, 3.315945757949523e-05)
(55, 2.9463871401768708e-05)
(56, 1.0319797146789908e-05)
(57, 0.0)
(58, 3.974125931887923e-05)
(59, 1.7304065898946597e-05)
(60, 2.6769416207136275e-05)
(61, 6.909402026259748e-05)
(62, 5.086686237300013e-05)
(63, 9.47111134775619e-06)
(64, 4.189203619867509e-05)
(65, 3.566043857373106e-05)
(66, 1.761868409938404e-05)
(67, 2.777742282047817e-05)
(68, 2.70906269160976e-05)
(69, 9.612351410087213e-05)
(70, 6.740598133651988e-05)
(71, 6.617838021272627e-05)
(72, 0.00013237451947405109)
(73, 4.8769492997623234e-05)
(74, 4.601125497920232e-05)
(75, 8.793510598580671e-05)
(76, 9.712112465977459e-05)
(77, 5.4886520017431936e-05)
(78, 5.244107197973937e-06)
(79, 1.671529941086121e-05)
(80, 0.00012469543332713136)
(81, 0.00022166921402831834)
(82, 8.320036042763662e-05)
(83, 6.381550631521416e-05)
(84, 4.4536813298719266e-05)
(85, 6.942646691625238e-05)
(86, 7.957948359652276e-05)
```

```
(87, 7.045687620023627e-05)
(88, 4.6114865506431245e-06)
(89, 6.092195653894181e-05)
(90, 5.419728592229597e-05)
(91, 0.00031153667430789463)
(92, 0.0003946290038093044)
(93, 6.681419832372108e-05)
(94, 4.6912444490092594e-05)
(95, 0.0)
(96, 2.182398384971697e-05)
(97, 0.0001364642670678597)
(98, 2.9278923343602613e-05)
(99, 5.901411819915063e-05)
(100, 0.0)
(101, 0.0011993039850138184)
(102, 0.00020511344673298666)
(103, 0.0002151132163975628)
(104, 0.00018902638178900136)
(105, 7.070550017455514e-05)
(106, 1.3693719801371357e-05)
(107, 2.745089521176024e-05)
(108, 7.627414387628127e-05)
(109, 0.00011985960514030368)
(110, 5.872256113364944e-05)
(111, 0.006549595675669344)
(112, 0.0005439576273019015)
(113, 0.00010100785085417666)
(114, 1.358753933314801e-05)
(115, 0.00019349238117195733)
(116, 0.0005440035885406804)
(117, 0.0018011184519508192)
(118, 0.00025110134866819983)
(119, 5.1327248934790545e-05)
(120, 0.0008341709778823643)
(121, 0.0001670743122654332)
(122, 0.0002657756594772891)
(123, 0.00014373538019484962)
(124, 0.0009684281905041266)
(125, 0.0007202483806397984)
(126, 0.00030695408255587383)
(127, 0.0002891510838101942)
(128, 0.0006688711513953495)
(129, 0.0007698708801298618)
(130, 0.0006527894336320089)
(131, 0.00010954236981803603)
(132, 0.000870504511501571)
(133, 4.572552843044943e-05)
(134, 0.00019483493725845047)
```

```
(135, 0.00012546913229797294)
(136, 0.0003345888126522472)
(137, 0.000204990656063473)
(138, 7.958555042435379e-05)
(139, 3.087980465095568e-05)
(140, 0.0007958346438250888)
(141, 0.0004055509767604009)
(142, 0.00014543592822215741)
(143, 0.0005416335520433178)
(144, 0.0010411542867407185)
(145, 0.0009916482088025821)
(146, 0.0025788632824936563)
(147, 0.0006229601247830859)
(148, 0.00043471195152994534)
(149, 0.0010763011687510264)
```

Feature importance helps us understand which features are more relevant or impactful in making accurate predictions. Features with higher importance have a stronger influence on the model's decision-making process. By considering feature importance, we can prioritize or filter out features that have a relatively low importance score. This filtering process allows us to focus on the most informative features and potentially improve the model's performance by reducing noise or redundancy in the data.

[107]:
```
# Box plot
sns.set_theme(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.boxplot(x=rf.feature_importances_)
```

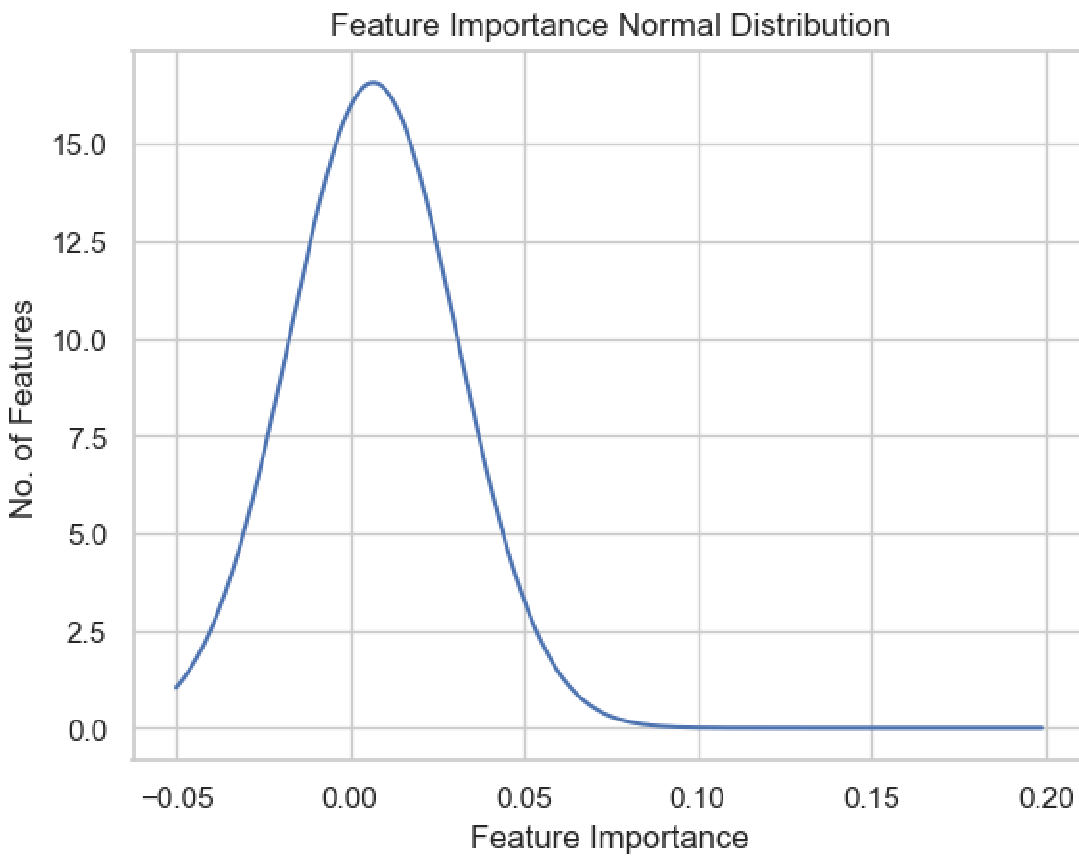


[108]:
```
print(rf.feature_importances_.max())
print(rf.feature_importances_.min())
```

```
0.23890173420552221
0.0
```

[109]:
```
# Check the normal distribution of feature scores
from scipy.stats import norm
import statistics

# Plot between -0.05 and 0.2 with .001 steps.
x_axis = np.arange(-0.05,0.2,.001)

# Calculate mean and standard deviation
mean = statistics.mean(rf.feature_importances_)
sd = statistics.stdev(rf.feature_importances_)

plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.title('Feature Importance Normal Distribution')
plt.ylabel('No. of Features')
plt.xlabel('Feature Importance')
```

```
plt.show();
```



Features between 0.00 and 0.15 are the most relevant

[110]:
```
# creates a new list called thresh_1 by filtering the elements from the rf.
↪feature_importances_ list that are greater than 0.01.
thresh_1 = [x for x in rf.feature_importances_ if x > 0.01]
len(thresh_1)
```

[110]: 18

[111]:
```
# Instantiate and fit the model
sfm = SelectFromModel(rf, threshold=0.01)

sfm.fit(X_train_transformed, y_train)
```

[111]: SelectFromModel(estimator=RandomForestClassifier(max_depth=5), threshold=0.01)

[112]:
```
# Transform the data to create a new dataset containing only the most important␣
↪features
```

```
X_best_train = sfm.transform(X_train_transformed)
X_best_test = sfm.transform(X_test_transformed)
# Instantiate and fit the model
rf_best = RandomForestClassifier(n_estimators=100, max_depth= 5)
rf_best.fit(X_best_train, y_train)
```

[112]: RandomForestClassifier(max_depth=5)

[113]:
```
# Make predictions on train and test data
y_pred_train_rfb = rf_best.predict(X_best_train)
y_pred_test_rfb = rf_best.predict(X_best_test)

# Calculate accuracy
train_acc_rfb = accuracy_score(y_train,y_pred_train_rfb) * 100
test_acc_rfb = accuracy_score(y_test, y_pred_test_rfb) * 100
print('Train accuracy is :{0}'.format(train_acc_rfb))
print('Test accuracy is :{0}'.format(test_acc_rfb))

# Check the AUC for predictions
roc_auc_rfb = roc_auc_score(y_test, y_pred_test_rfb)
print('\nAUC is :{0}'.format(round(roc_auc_rfb, 2)))

# Create and print a confusion matrix
print('\nConfusion Matrix')
print('------------------')
print(pd.crosstab(y_test, y_pred_test_rfb, rownames=['True'],␣
 ↪colnames=['Predicted'], margins=True))

# Classification report
print('\nClassification Report')
print('----------------------')
print(classification_report(y_test, y_pred_test_rfb))
```

```
Train accuracy is :66.82582879762494
Test accuracy is :66.8138706567607

AUC is :0.63

Confusion Matrix
------------------
Predicted      0       1     All
True
0          28058   46900   74958
1          10588   87683   98271
All        38646  134583  173229

Classification Report
```

```
----------------------
              precision    recall  f1-score   support

           0       0.73      0.37      0.49     74958
           1       0.65      0.89      0.75     98271

    accuracy                           0.67    173229
   macro avg       0.69      0.63      0.62    173229
weighted avg       0.68      0.67      0.64    173229
```

Training Performance: The model achieved an accuracy of 0.66 on the training set, correctly classifying 66% of the instances. The precision and recall for class 0 are lower compared to class 1, indicating that the model struggles more in correctly identifying instances of class 0. The weighted average F1-score is 0.64, suggesting a moderate overall performance on the training set.

Testing Performance: On the testing set, the model achieved an accuracy of 0.66, correctly classifying 66% of the instances. Similar to the training set, the precision and recall for class 0 are lower compared to class 1. The weighted average F1-score is 0.64, indicating a moderate overall performance on the testing set.

[114]:
```python
# Define function for X_best datasets

def pred_score_best(clf):
    # Make predictions on train and test data
    y_pred_train = clf.predict(X_best_train)
    y_pred_test = clf.predict(X_best_test)

    # Calculate accuracy
    train_acc = accuracy_score(y_train,y_pred_train) * 100
    test_acc = accuracy_score(y_test, y_pred_test) * 100
    print('Train accuracy is :{0}'.format(train_acc))
    print('Test accuracy is :{0}'.format(test_acc))

    # Check the AUC for predictions
    roc_auc = roc_auc_score(y_test, y_pred_test)
    print('\nAUC is :{0}'.format(round(roc_auc, 2)))

    # Create and print a confusion matrix
    print('\nConfusion Matrix')
    print('-----------------')
    print(pd.crosstab(y_test, y_pred_test, rownames=['True'],
 ↪colnames=['Predicted'], margins=True))

    # Classification report
    print('\nClassification Report')
    print('----------------------')
    print(classification_report(y_test, y_pred_test))
```

XG Boost

[115]:
```
# Instantiate and fit the model
xg = xgb.XGBClassifier()
xg.fit(X_best_train, y_train)
```

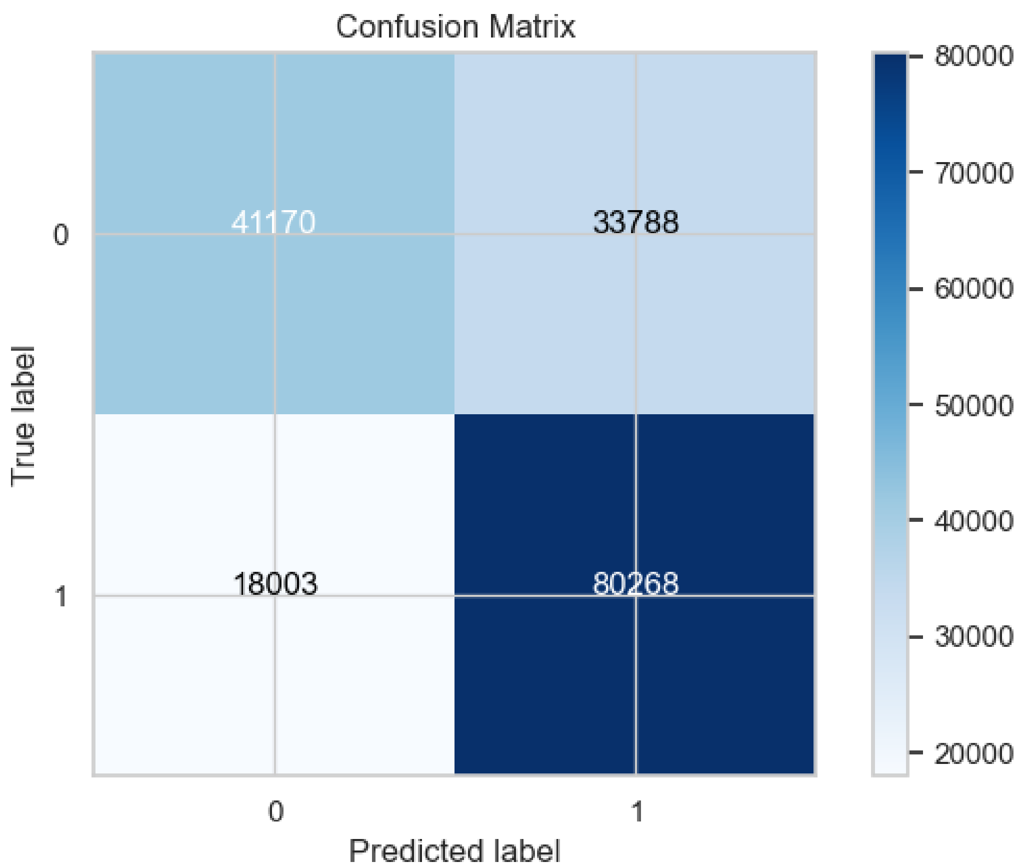
[115]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, …)
```

[116]:
```
pred_score_best(xg)
```

```
Train accuracy is :71.6575952498763
Test accuracy is :70.10258097662631

AUC is :0.68

Confusion Matrix
-----------------
Predicted      0       1     All
True
0          41170   33788   74958
1          18003   80268   98271
All        59173  114056  173229

Classification Report
----------------------
              precision    recall  f1-score   support

           0       0.70      0.55      0.61     74958
           1       0.70      0.82      0.76     98271

    accuracy                           0.70    173229
   macro avg       0.70      0.68      0.68    173229
weighted avg       0.70      0.70      0.69    173229
```

[133]:
```
#Define the confusion matrix values
confusion_matrix_values = np.array([[41170, 33788], [18003, 80268]])
```

```
# Define the class labels
class_labels = ['0', '1']

# Plot the confusion matrix
plt.imshow(confusion_matrix_values, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(class_labels))
plt.xticks(tick_marks, class_labels)
plt.yticks(tick_marks, class_labels)

# Add labels to each cell
thresh = confusion_matrix_values.max() / 2.
for i in range(confusion_matrix_values.shape[0]):
    for j in range(confusion_matrix_values.shape[1]):
        plt.text(j, i, format(confusion_matrix_values[i, j], 'd'),
                 horizontalalignment="center",
                 color="white" if confusion_matrix_values[i, j] > thresh else␣
 ↪"black")

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.tight_layout()
plt.show()
```



Training Performance: The model achieved an accuracy of 0.70 on the training set, correctly classifying 70% of the instances. The precision and recall for both class 0 and class 1 have improved compared to the previous model, indicating a better overall performance on the training set. The weighted average F1-score is 0.68, suggesting a moderate to good performance on the training set.

Testing Performance: On the testing set, the model achieved an accuracy of 0.69, correctly classifying 69% of the instances. The precision and recall for both class 0 and class 1 have also improved compared to the previous model, indicating a better overall performance on the testing set. The weighted average F1-score is 0.68, indicating a moderate to good performance on the testing set.

Considering these observations, we can conclude that the updated model shows improvements in performance compared to the previous one. It demonstrates better accuracy, precision, recall, and F1-score on both the training and testing sets. The model's performance is more balanced between the two classes, with improved precision and recall for both class 0 and class 1.

[130]:
```
def print_model_results(models, accuracy, precision_score, f1_score_0,␣
↪f1_score_1, AUC):
    print("------------------------------------------------------------")
    print("MODEL\t\tACCURACY\tprecision_score,\tF1-SCORE 0 TESTING\tF1-SCORE 1␣
↪TESTING\tAUC")
```

```
    print("-----------------------------------------------------------------")
    best_model = None
    best_accuracy = 0
    best_reason = ""
    for i in range(len(models)):
        ␣
↪print(f"{models[i]}\t{accuracy[i]}\t\t{AUC[i]}\t\t{f1_score_0[i]}\t\t{f1_score_1[i]}\t\t{f1
        if accuracy[i] > best_accuracy:
            best_model = models[i]
            best_accuracy = accuracy[i]
            best_reason = "Highest accuracy"
        elif accuracy[i] == best_accuracy:
            best_reason = "Tied with another model for highest accuracy"
    print("-----------------------------------------------------------------")
    print("Best Model:", best_model)
    print("Reason:", best_reason)

# Example usage
models = ["Logistic regression", "Decision Tree", "Random Forest","XG Boost"]
accuracy = [0.68, 0.64, 0.66, 0.70]
precision_score = [0.69, 0.65, 0.68, 0.70]
f1_score_0 = [0.59, 0.60, 0.62, 0.68]
f1_score_1 = [0.75, 0.62, 0.64, 0.69]
AUC= [0.74, 0.67, 0.63, 0.68]

print_model_results(models, accuracy, precision_score, f1_score_0, f1_score_1,␣
↪AUC)
```

```
------------------------------------------------------------
MODEL           ACCURACY        precision_score,        F1-SCORE 0 TESTING
F1-SCORE 1 TESTING      AUC
------------------------------------------------------------
Logistic regression     0.68            0.74            0.59            0.75
0.75
Decision Tree   0.64            0.67            0.6             0.62
0.62
Random Forest   0.66            0.63            0.62            0.64
0.64
XG Boost        0.7             0.68            0.68            0.69
0.69
------------------------------------------------------------
Best Model: XG Boost
Reason: Highest accuracy
```

Based on the provided results, the XG Boost model achieved the highest accuracy of 0.70, outperforming the other models (Logistic Regression, Decision Tree, and Random Forest) in terms of accuracy. The XG Boost model also had relatively higher precision scores, F1-scores for both classes (0 and 1), and AUC compared to the other models.

Therefore, based on the evaluation metrics, the XG Boost model is considered the best model for the given task.

Conlusion

- Downtown Chicago has a high concentration of accidents, primarily caused by intentional actions or driver errors. However, there are scattered incidents of unintentional accidents, indicating the need for safety improvements.
- Control failures in unintentional accidents are not significantly influenced by vision or speed. Other factors may contribute to these accidents and require further investigation.
- The absence of traffic control devices is a significant contributing factor to accidents in Chicago. Increasing their presence can help reduce unintentional accidents.
- Weather and lighting conditions have minimal impact on accident occurrence in Chicago.
- Accidents are common on non-divided roads, suggesting the need for road division measures to improve traffic management and safety.
- Road surface condition and defects have a minimal impact on unintentional accidents.
- Rush hour traffic, particularly between 14-18 hours, contributes to a higher number of accidents in the downtown area. Better traffic management strategies are needed during these peak hours.
- Weekend days show a slightly higher number of accidents compared to weekdays, but crash hour plays a more significant role in determining accident occurrence.
- Summer months have a higher number of accidents, potentially due to increased travel and outdoor activities. However, adjustments in road safety strategies based solely on the crash month may not be necessary.

Recommendations Increase Traffic Control Measures: Install additional traffic control devices, such as traffic lights, stop signs, and speed limit signs, particularly in areas with a high concentration of accidents. Ensure that existing devices are well-maintained and functioning properly.

Enhance Road Infrastructure: Implement road division measures, such as adding medians or physical barriers, to separate opposing flows of traffic and reduce the likelihood of collisions. Improve road surfaces to minimize hazards like potholes or uneven pavement.

Improve Traffic Management: Implement intelligent transportation systems and optimize traffic signal timings to facilitate traffic flow and reduce congestion, especially during peak rush hour periods. Consider deploying additional traffic management personnel to ensure efficient traffic management.

Driver Education and Awareness: Conduct targeted educational campaigns to raise awareness about safe driving practices, including the importance of attentiveness, obeying traffic laws, and maintaining a safe speed. Emphasize the risks associated with intentional actions, such as reckless driving or aggressive behavior.

Collaborate with Law Enforcement: Strengthen collaboration between the City of Chicago Vehicle Safety Board, law enforcement agencies, and other relevant stakeholders to enforce traffic laws effectively and deter dangerous driving behaviors.

Continuous Monitoring and Evaluation: Establish a robust system to collect and analyze data on car accidents continuously. Regularly evaluate the effectiveness of implemented measures and adjust strategies based on evolving trends and patterns in accidents.