1 Final Project Submission

- Student name: Vinayak Modgil
- Student pace: self paced / part time / full time: Full Time
- Scheduled project review date/time:
- Instructor name: Yish Lim
- Blog post URL:
- Video of 5-min Non-Technical Presentation:

2 Table of Contents

3 Introduction

Crash data shows information about each traffic crash on city streets within the C
under the jurisdiction of Chicago Police Department (CPD). Data are shown as i
reporting system (E-Crash) at CPD, excluding any personally identifiable informa
information can be found here (https://data.cityofchicago.org/Transportation/Traf
t3if).

3.1 Business Statement

It is very crucial for the Vehicle Safety Board to determine the cause of an accid
the city of Chicago has been chosen for the analysis of the accidents occuring ir

3.2 Analysis Methodology

The dataset has information on about 520,000 car crashes in the city of Chicago
known contributory cause. Information on these craashes include many importa
crashes and the aftermath of the crashes.I will clean and explore the data to be
machine learning model to predict the most known contributory cause.

More specifically, I will dive deep into exploring and tuning the models so that th
cause can be known. From there, I will make predictions and conclusions which
prevailing cause of an accident occuring in the city of Chicago.

4 Data Collection

4.1 Importing necessary packages

In [1]:

```
#data wrangling and visualization packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import statsmodels.api as sm
import scipy.stats as stats

#feature engineering packages
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardS

#feature selection packages
from feature_engine.selection import DropDuplicateFeatures
from feature_engine.selection import DropConstantFeatures

#modeling packages
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression, Logis
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

#modeling evaluation packages
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import plot_roc_curve, roc_curve, auc
from sklearn.metrics import get_scorer

#optimization packages
from sklearn.model_selection import GridSearchCV
```

In [2]:

```
#notebook settings
pd.set_option("display.max_columns", 40)
pd.options.display.float_format = '{:,}'.format

import warnings
warnings.filterwarnings('ignore')
```

4.2 Global Functions

Contents

Contents

In [3]:

```
from sklearn.impute import SimpleImputer

impute_mean = SimpleImputer(strategy = "mean")
impute_median = SimpleImputer(strategy = "median")
impute_mode = SimpleImputer(strategy = "most_frequent")
impute_cont_const = SimpleImputer(strategy = "constant", f
impute_cat_const = SimpleImputer(strategy = "constant", fi


def clean_df(df):
    '''
    Takes dataset df as input and returns a clean dataset
    with null values taken care of.

    - df: A dataframe
    '''
    # Dividing datasets in continuous and catergorical var
    cont_features = [col for col in df.columns if df[col].
    cat_features = [col for col in df.columns if df[col].d


    #filling injuries continuous variables with mean
    injuries = ["INJURIES_TOTAL", "INJURIES_FATAL", "INJUR
               , "INJURIES_REPORTED_NOT_EVIDENT", "INJURIE

    df[injuries] = impute_mean.fit_transform(df[injuries])

    # filling latitude and longitude continuous variables
    lat_long = ["LATITUDE", "LONGITUDE"]

    df[lat_long] = impute_cont_const.fit_transform(df[lat_

    df[["BEAT_OF_OCCURRENCE"]] = impute_mode.fit_transform

    #filling num units continuous variable with median
    num_units = ["NUM_UNITS"]
    df[num_units] = impute_cont_const.fit_transform(df[num

    # Filling null categorical values with "missing"
    cat_vars = ["RD_NO", "CRASH_DATE_EST_I", "LANE_CNT", "
               "NOT_RIGHT_OF_WAY_I", "HIT_AND_RUN_I", "PHO
               "WORK_ZONE_TYPE", "WORKERS_PRESENT_I", "MOS

    df[cat_vars] = impute_cat_const.fit_transform(df[cat_v


    return df
```

Contents

In [4]:

```
def rows_to_drop(df, y=None):
    '''
    Cleans rows which are not needed
    '''
    if y!= None:
        df_with_index = df.set_index(y)

        df_with_index.drop(labels=["UNABLE TO DETERMINE",
        df_with_index.reset_index(inplace=True)
    return df_with_index
```

In [5]:

```
def rows_to_drop_unknown(df, y=None):
    '''
    Cleans rows which are not needed
    '''
    if y!= None:
        df_with_index = df.set_index(y)

        df_with_index.drop(labels=["UNKNOWN"], axis=0, inp
        df_with_index.reset_index(inplace=True)
    return df_with_index
```

In [6]:

```
def drop_quasi_const(df):
    '''
    Function taken from Feature Engineering course on Udem
    the constant and quasi-constant features.
    - df: A dataframe
    '''
    #Create an empty list
    quasi_const_feat = []

    #Iterate over every feature
    for feature in df.columns:

        #Find the predominant value, the value that is
        # shared by most observations
        predominant = (df[feature].value_counts() /
                       np.float(len(df))).sort_values(asce

        #Evaluate the predominant feature: do more than 99
        #show 1 value?
        if predominant > 0.998:

            #if yes, append it to the empt list
            quasi_const_feat.append(feature)

    df.drop(labels=quasi_const_feat, axis=1, inplace=True)
    return df
```

Contents

In [7]:

```
def col_summary(df, num_col=None, cat_cols=None, y_col = "
    '''
    this function gives a brief summary of a single col
    in the dataset df. Also, it shows the essential plots
    required for the column w.r.t the dependent variable.

    arguments:
    df - given dataset
    num_col - numerical column in the dataset
    cat_cols - categorical columns in the dataset
    y_col - dependent variable
    label_count - number of labels to draw in bar graph
    '''
    if num_col != None:
        #print the column name
        print(f'Column Name: {num_col}')
        #print the number of unique values
        print(f'Number of unique values: {df[num_col].nuni
        #print the number of duplicate values
        print(f'There are {df[num_col].duplicated().sum()}
        #print the number of null values
        print(f'There are {df[num_col].isna().sum()} null
        #print the number of values equal to 0
        print(f'There are {(df[num_col] == 0).sum()} zeros
        print('\n')
        #print the value counts percentage
        print('Value Counts Percentage', '\n',
              df[num_col].value_counts(normalize=True, dro
        print('\n')
        #print descriptive statistics
        print('Descriptive Metrics:','\n',
              df[num_col].describe())
        #plot boxplot, histogram
        fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(

        histogram = df[num_col].hist(ax=ax[0, 0])
        ax[0, 0].set_title(f'Distribution of {num_col}');

        scatter = df.plot(kind='scatter', x=num_col, y=y_c
        ax[0, 1].set_title(f'{y_col} vs {num_col}');

        boxplot = df.boxplot(column=num_col, ax=ax[1,0]);
        ax[1,0].set_title(f'Boxplot of {num_col}');

        sm.graphics.qqplot(df[num_col], dist=stats.norm, l
        ax[1,1].set_title(f'QQ plot of {num_col}');
        plt.tight_layout()

        plt.show()
        return

    else:

        for col in cat_cols:
            print('============================')
            #print the column name
            print(f'Column Name: {col}')
            print('\n')
            #print the number of unique values
```

Contents


```
            print(f'Number of unique values: {df[col].nuni
            print('\n')
            #print the number of duplicate values
            print(f'There are {df[col].duplicated().sum()}
            print('\n')
            #print the number of null values
            print(f'There are {df[col].isna().sum()} null
            print('\n')
            #print the number of values equal to '0'
            print(f'There are {(df[col] == "0").sum()} zer
            print('\n')
            #print the value counts percentage
            print('Value Counts Percentage', '\n',
                  df[col].value_counts(dropna=False).round
            print('\n')

            #plot barplot, histogram
            fig, ax = plt.subplots(figsize=(15,10))

            bar_graph = df[col].value_counts(normalize=Tru
                                             dropna=False)
            ax.axhline(y=thresh, color='red', linestyle='-
                       label=f'{thresh*100}% Threshold')
            ax.set_title(f'{col} Value Counts')
            ax.set_xlabel(f'{col} Labels')
            ax.set_ylabel('Percentage')
            ax.legend()

            plt.tight_layout()

            plt.show()

        return
```

In [8]:

```
def plot_confusion(y_true, y_pred):
    #Create an instance of confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    #Plot it on a heatmap
    sns.heatmap(cm, annot=True, fmt="0.2g", cmap = sns.col
    print
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()
```

Contents

In [9]:

```
#function to look at plots and stats of column with or wit
def model_eval(model, X_train, y_train, X_test, y_test,
               prev_model=None, prev_X_train=None, prev_y_
               prev_X_test=None, prev_y_test=None):
    '''
    This function takes in a fit model and provides classi
    metrics on that model. Optionally, a previous model ca
    improvement metrics between the current model and the
    Keyword Arguments:
    - model: A fit model
    - X_train, y_train, X_test, y_test: Training and testi
    the "model" stated above was fit on.
    - prev_model: Another fit model
    - prev_X_train, prev_y_train, prev_X_test, prev_y_test
    testing dataframes which the previous model was fit on
    '''
    #current model predictions on testing dataframe
    y_hat_test = model.predict(X_test)
    #get scores of current model on testing dataframe
    recall_model = get_scorer('recall')(model, X_test, y_t
    f1_model = get_scorer('f1')(model, X_test, y_test).rou
    accuracy_model = get_scorer('accuracy')(model, X_test,
    auc_model = get_scorer('roc_auc')(model, X_test, y_tes

    recall_model_train = get_scorer('recall')(model, X_tra
    #if statement to check for availability of a previous
    #and testing dataframes
    if prev_model != None:
        if prev_X_train is None:
            #if prevoius model has a different training an

            #get previous model predictions and scores
            y_hat_test_prev = prev_model.predict(X_test)

            recall_prev = get_scorer('recall')(prev_model,
            f1_prev = get_scorer('f1')(prev_model, X_test,
            accuracy_prev = get_scorer('accuracy')(prev_mo
            auc_prev = get_scorer('roc_auc')(prev_model, X

            print('MODEL EVAL VS PREVIOUS (TEST)')
            print('=====================================

            #create dataframe comparing current and previo
            df = pd.DataFrame(index=['Recall','F1','Accura
                              columns=['Previous Model','C

            df.loc['Recall','Current Model'] = recall_mode
            df.loc['Recall','Previous Model'] = recall_pre
            df.loc['F1','Current Model'] = f1_model
            df.loc['F1','Previous Model'] = f1_prev
            df.loc['Accuracy','Current Model'] = accuracy_
            df.loc['Accuracy','Previous Model'] = accuracy
            df.loc['AUC','Current Model'] = auc_model
            df.loc['AUC','Previous Model'] = auc_prev
            df.loc['Recall','Delta'] = (recall_model - rec
            df.loc['F1','Delta'] = (f1_model - f1_prev).ro
            df.loc['Accuracy','Delta'] = (accuracy_model -
            df.loc['AUC','Delta'] = (auc_model - auc_prev)

```

Contents


```
60          display(df)
61          print('\n')
62
63          #display current model scores, reports and gra
64          print(f"CURRENT MODEL: {'Overfit (Recall)' if
65          print('=======================================
66          print('\n')
67
68          print(f'Recall on Training: {recall_model_trai
69          print(f'Recall on Test: {recall_model}')
70          print('\n')
71
72          print('Classification Reports----------------
73          print(classification_report(y_test, y_hat_test
74
75          print('Test Graphs---------------------------
76          fig, ax = plt.subplots(ncols=2, figsize=(15,8)
77
78          plot_confusion_matrix(model, X_test, y_test, c
79                                        normalize='true',
80                                display_labels=["NO INJU
81                                ax=ax[0]);
82
83          plot_roc_curve(model, X_test, y_test, ax=ax[1]
84          plt.show()
85
86          #display previous model graphs
87          print("PREVIOUS MODEL")
88          print('=======================================
89          fig, ax = plt.subplots(ncols=2, figsize=(15,8)
90
91          plot_confusion_matrix(prev_model, X_test, y_te
92                                        normalize='true',
93                                display_labels=['NO INJU
94                                ax=ax[0]);
95
96          plot_roc_curve(prev_model, X_test, y_test, ax=
97          plt.show()
98
99          plt.tight_layout()
100     else:
101         #if previous model has the same dataframes as
102         y_hat_test_prev = prev_model.predict(prev_X_te
103
104         recall_prev = get_scorer('recall')(prev_model,
105                                                prev_y_test
106         f1_prev = get_scorer('f1')(prev_model, prev_X_
107                                        prev_y_test).round(
108         accuracy_prev = get_scorer('accuracy')(prev_mo
109                                                prev_y_
110         auc_prev = get_scorer('roc_auc')(prev_model, p
111                                              prev_y_test).
112
113         print('MODEL EVAL VS PREVIOUS (TEST)')
114         print('=======================================
115
116         df = pd.DataFrame(index=['Recall','F1','Accura
117                                  columns=['Previous Model','C
118
119         df.loc['Recall','Current Model'] = recall_mode
120         df.loc['Recall','Previous Model'] = recall_pre
```

Contents


```
121             df.loc['F1','Current Model'] = f1_model
122             df.loc['F1','Previous Model'] = f1_prev
123             df.loc['Accuracy','Current Model'] = accuracy_
124             df.loc['Accuracy','Previous Model'] = accuracy
125             df.loc['AUC','Current Model'] = auc_model
126             df.loc['AUC','Previous Model'] = auc_prev
127             df.loc['Recall','Delta'] = (recall_model - rec
128             df.loc['F1','Delta'] = (f1_model - f1_prev).ro
129             df.loc['Accuracy','Delta'] = (accuracy_model -
130             df.loc['AUC','Delta'] = (auc_model - auc_prev)
131
132             display(df)
133             print('\n')
134
135
136             print(f"CURRENT MODEL: {'Overfit (Recall)' if
137             print('=======================================
138             print('\n')
139
140             print(f'Recall on Training: {recall_model_trai
141             print(f'Recall on Test: {recall_model}')
142             print('\n')
143
144             print('Classification Reports----------------
145             print(classification_report(y_test, y_hat_test
146
147             print('Test Graphs---------------------------
148             fig, ax = plt.subplots(ncols=2, figsize=(15,8)
149
150             plot_confusion_matrix(model, X_test, y_test, c
151                                   normalize='true',
152                                   display_labels=['NO INJU
153                                   ax=ax[0]);
154
155             plot_roc_curve(model, X_test, y_test, ax=ax[1]
156             plt.show()
157
158             print("PREVIOUS MODEL")
159             print('=======================================
160             fig, ax = plt.subplots(ncols=2, figsize=(15,8)
161
162             plot_confusion_matrix(prev_model, prev_X_test,
163                                   normalize='true',
164                                   display_labels=['NO INJU
165                                   ax=ax[0]);
166
167             plot_roc_curve(prev_model, prev_X_test, prev_y
168                            ax=ax[1]).ax_.plot([0,1],[0,1])
169             plt.show()
170
171             plt.tight_layout()
172     else:
173         #if there is no previous model, get current model
174         print(f"CURRENT MODEL: {'Overfit (Recall)' if reca
175         print('=======================================
176         print('\n')
177         print('Classification Reports----------------
178         print(classification_report(y_test, y_hat_test))
179
180         print('Test Graphs---------------------------
181         fig, ax = plt.subplots(ncols=2, figsize=(15,8))
```

```
182
183         plot_confusion_matrix(model, X_test, y_test, cmap=
184                                       normalize='true',
185                               display_labels=['NO INJURY',
186                               ax=ax[0]);
187
188         plot_roc_curve(model, X_test, y_test, ax=ax[1]).ax
189         plt.show()
190
191
192     return
```

4.3 Import Data

In [10]:

```
1 #df_cars = pd.read_csv("data/Traffic_Crashes_-_Vehicles.cs
2 df_crashes = pd.read_csv("data/Traffic_Crashes_-_Crashes.c
3 df_crashes.head()
```

Out[10]:

	CRASH_RECORD_ID	RD_NO	CRASH_DAT
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	JC343143	
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	JA329216	
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	JD292400	
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	JD293602	
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	JD290451	

5 rows × 49 columns

4.4 Data Schema

Taken From: [Chicago car crash website (https://data.cityofchicago.org/Transpo Crashes/85ca-t3if)]

- `CRASH_RECORD_ID` This number can be used to link to the same crash in t datasets. This number also serves as a unique ID in this dataset.
- `RD_NO` Chicago Police Department report number. For privacy reasons, thi crashes.

Contents

Contents

- `CRASH_DATE_EST_I`
  Crash date estimated by desk officer or reporting party (only used in cases
  police station days after the crash)
- `CRASH_DATE`
  Date and time of crash as entered by the reporting officer
- `POSTED_SPEED_LIMIT`
  Posted speed limit, as determined by reporting officer
- `TRAFFIC_CONTROL_DEVICE`
  Traffic control device present at crash location, as determined by reporting
- `DEVICE_CONDITION`
  Condition of traffic control device, as determined by reporting officer
- `WEATHER_CONDITION`
  Weather condition at time of crash, as determined by reporting officer
- `LIGHTING_CONDITION`
  Light condition at time of crash, as determined by reporting officer
- `FIRST_CRASH_TYPE`
  Type of first collision in crash
- `TRAFFICWAY_TYPE`
  Trafficway type, as determined by reporting officer
- `LANE_CNT`
  Total number of through lanes in either direction, excluding turn lanes, as de
  (0 = intersection)
- `ALIGNMENT`
  Street alignment at crash location, as determined by reporting officer
- `ROADWAY_SURFACE_COND`
  Road surface condition, as determined by reporting officer
- `ROAD_DEFECT`
  Road defects, as determined by reporting officer
- `REPORT_TYPE`
  Administrative report type (at scene, at desk, amended)
- `CRASH_TYPE`
  A general severity classification for the crash. Can be either Injury and/or To
  Drive Away
- `INTERSECTION_RELATED_I`
  A field observation by the police officer whether an intersection played a rol
  represent whether or not the crash occurred within the intersection.
- `NOT_RIGHT_OF_WAY_I`
  Whether the crash begun or first contact was made outside of the public rig
- `HIT_AND_RUN_I`
  Crash did/did not involve a driver who caused the crash and fled the scene
  information and/or rendering aid
- `DAMAGE`
  A field observation of estimated damage.
- `DATE_POLICE_NOTIFIED`
  Calendar date on which police were notified of the crash
- `PRIM_CONTRIBUTORY_CAUSE`
  The factor which was most significant in causing the crash, as determined b
- `SEC_CONTRIBUTORY_CAUSE`
  The factor which was second most significant in causing the crash, as deter
- `STREET_NO`
  Street address number of crash location, as determined by reporting officer

- `STREET_DIRECTION`
  Street address direction (N,E,S,W) of crash location, as determined by repo
- `STREET_NAME`
  Street address name of crash location, as determined by reporting officer
- `BEAT_OF_OCCURRENCE`
  Chicago Police Department Beat ID. Boundaries available at https://data.cit (https://data.cityofchicago.org/d/aerh-rz74)
- `PHOTOS_TAKEN_I`
  Whether the Chicago Police Department took photos at the location of the c
- `STATEMENTS_TAKEN_I`
  Whether statements were taken from unit(s) involved in crash
- `DOORING_I`
  Whether crash involved a motor vehicle occupant opening a door into the tr causing a crash
- `WORK_ZONE_I`
  Whether the crash occurred in an active work zone
- `WORK_ZONE_TYPE`
  The type of work zone, if any
- `WORKERS_PRESENT_I`
  Whether construction workers were present in an active work zone at crash
- `NUM_UNITS`
  Number of units involved in the crash. A unit can be a motor vehicle, a pede non-passenger roadway user. Each unit represents a mode of traffic with ar
- `MOST_SEVERE_INJURY`
  Most severe injury sustained by any person involved in the crash
- `INJURIES_TOTAL`
  Total persons sustaining fatal, incapacitating, non-incapacitating, and possil the reporting officer
- `INJURIES_FATAL`
  Total persons sustaining fatal injuries in the crash
- `INJURIES_INCAPACITATING`
  Total persons sustaining incapacitating/serious injuries in the crash as deter Any injury other than fatal injury, which prevents the injured person from wa continuing the activities they were capable of performing before the injury o lacerations, broken limbs, skull or chest injuries, and abdominal injuries.
- `INJURIES_NON_INCAPACITATING`
  Total persons sustaining non-incapacitating injuries in the crash as determir Any injury, other than fatal or incapacitating injury, which is evident to obser Includes lump on head, abrasions, bruises, and minor lacerations.
- `INJURIES_REPORTED_NOT_EVIDENT`
  Total persons sustaining possible injuries in the crash as determined by the momentary unconsciousness, claims of injuries not evident, limping, compla hysteria.
- `INJURIES_NO_INDICATION`
  Total persons sustaining no injuries in the crash as determined by the repor
- `INJURIES_UNKNOWN`
  Total persons for whom injuries sustained, if any, are unknown
- `CRASH_HOUR`
  The hour of the day component of CRASH_DATE.
- `CRASH_DAY_OF_WEEK`
  The day of the week component of CRASH_DATE. Sunday=1

- `CRASH_MONTH`
  The month component of CRASH_DATE.
- `LATITUDE`
  The latitude of the crash location, as determined by reporting officer, as deri
  of crash
- `LONGITUDE`
  The longitude of the crash location, as determined by reporting officer, as de
  address of crash
- `LOCATION`
  The crash location, as determined by reporting officer, as derived from the r

4.5 Investigate Data

Contents

Contents

In [11]:

```
1 df_crashes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 533613 entries, 0 to 533612
Data columns (total 49 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   CRASH_RECORD_ID                 533613 non-null  object
 1   RD_NO                           529261 non-null  object
 2   CRASH_DATE_EST_I                40442 non-null   object
 3   CRASH_DATE                      533613 non-null  object
 4   POSTED_SPEED_LIMIT              533613 non-null  int64
 5   TRAFFIC_CONTROL_DEVICE          533613 non-null  object
 6   DEVICE_CONDITION                533613 non-null  object
 7   WEATHER_CONDITION               533613 non-null  object
 8   LIGHTING_CONDITION              533613 non-null  object
 9   FIRST_CRASH_TYPE                533613 non-null  object
 10  TRAFFICWAY_TYPE                 533613 non-null  object
 11  LANE_CNT                        198967 non-null  object
 12  ALIGNMENT                       533613 non-null  object
 13  ROADWAY_SURFACE_COND            533613 non-null  object
 14  ROAD_DEFECT                     533613 non-null  object
 15  REPORT_TYPE                     520300 non-null  object
 16  CRASH_TYPE                      533613 non-null  object
 17  INTERSECTION_RELATED_I          120911 non-null  object
 18  NOT_RIGHT_OF_WAY_I              25243 non-null   object
 19  HIT_AND_RUN_I                   160334 non-null  object
 20  DAMAGE                          533613 non-null  object
 21  DATE_POLICE_NOTIFIED            533613 non-null  object
 22  PRIM_CONTRIBUTORY_CAUSE         533613 non-null  object
 23  SEC_CONTRIBUTORY_CAUSE          533613 non-null  object
 24  STREET_NO                       533613 non-null  int64
 25  STREET_DIRECTION                533610 non-null  object
 26  STREET_NAME                     533612 non-null  object
 27  BEAT_OF_OCCURRENCE              533608 non-null  float64
 28  PHOTOS_TAKEN_I                  6661 non-null    object
 29  STATEMENTS_TAKEN_I              10851 non-null   object
 30  DOORING_I                       1714 non-null    object
 31  WORK_ZONE_I                     3363 non-null    object
 32  WORK_ZONE_TYPE                  2659 non-null    object
 33  WORKERS_PRESENT_I               830 non-null     object
 34  NUM_UNITS                       533608 non-null  float64
 35  MOST_SEVERE_INJURY              532515 non-null  object
 36  INJURIES_TOTAL                  532526 non-null  float64
 37  INJURIES_FATAL                  532526 non-null  float64
 38  INJURIES_INCAPACITATING         532526 non-null  float64
 39  INJURIES_NON_INCAPACITATING     532526 non-null  float64
 40  INJURIES_REPORTED_NOT_EVIDENT   532526 non-null  float64
 41  INJURIES_NO_INDICATION          532526 non-null  float64
 42  INJURIES_UNKNOWN                532526 non-null  float64
 43  CRASH_HOUR                      533613 non-null  int64
 44  CRASH_DAY_OF_WEEK               533613 non-null  int64
 45  CRASH_MONTH                     533613 non-null  int64
 46  LATITUDE                        530564 non-null  float64
 47  LONGITUDE                       530564 non-null  float64
 48  LOCATION                        530564 non-null  object
dtypes: float64(11), int64(5), object(33)
memory usage: 199.5+ MB
```

Observations

- Many columns to explore for null value imputation
- Column names are already standardized
- Data types will require further evaluation during engineering

In [12]:

```
#evaluate numerical data descriptive statistics
df_crashes.describe()
```

Out[12]:

	POSTED_SPEED_LIMIT	STREET_NO	BEAT_OF_OCCURRENCE
count	533,613.0	533,613.0	533,608.0
mean	28.30869750174752	3,663.269457453248	1,238.3603281809867
std	6.424878360535593	2,910.2447183927966	706.9756822999486
min	0.0	0.0	111.0
25%	30.0	1,215.0	712.0
50%	30.0	3,199.0	1,135.0
75%	30.0	5,599.0	1,822.0
max	99.0	451,100.0	6,100.0

Observations

- Few of these numerical features should be transformed into a categ
- `INJURIES_TOTAL`, `INJURIES_FATAL`, `INJURIES_INCAPACITATI` `INJURIES_NON_INCAPACITATING`, `INJURIES_REPORTED_NOT_EV` `INJURIES_NO_INDICATION`, `INJURIES_UNKNOWN`, `CRASH_HOUR` which may be placeholder for unknown.

5 Data Cleaning

In [13]:

```
df_crashes_clean = df_crashes.copy()
```

Contents

Contents

In [14]:

```
1 df_crashes_clean.isnull().sum()
```

Out[14]:

```
CRASH_RECORD_ID                     0
RD_NO                            4352
CRASH_DATE_EST_I               493171
CRASH_DATE                          0
POSTED_SPEED_LIMIT                  0
TRAFFIC_CONTROL_DEVICE              0
DEVICE_CONDITION                    0
WEATHER_CONDITION                   0
LIGHTING_CONDITION                  0
FIRST_CRASH_TYPE                    0
TRAFFICWAY_TYPE                     0
LANE_CNT                       334646
ALIGNMENT                           0
ROADWAY_SURFACE_COND                0
ROAD_DEFECT                         0
REPORT_TYPE                     13313
CRASH_TYPE                          0
INTERSECTION_RELATED_I         412702
NOT_RIGHT_OF_WAY_I             508370
HIT_AND_RUN_I                  373279
DAMAGE                              0
DATE_POLICE_NOTIFIED                0
PRIM_CONTRIBUTORY_CAUSE             0
SEC_CONTRIBUTORY_CAUSE              0
STREET_NO                           0
STREET_DIRECTION                    3
STREET_NAME                         1
BEAT_OF_OCCURRENCE                  5
PHOTOS_TAKEN_I                 526952
STATEMENTS_TAKEN_I             522762
DOORING_I                      531899
WORK_ZONE_I                    530250
WORK_ZONE_TYPE                 530954
WORKERS_PRESENT_I              532783
NUM_UNITS                           5
MOST_SEVERE_INJURY               1098
INJURIES_TOTAL                   1087
INJURIES_FATAL                   1087
INJURIES_INCAPACITATING          1087
INJURIES_NON_INCAPACITATING      1087
INJURIES_REPORTED_NOT_EVIDENT    1087
INJURIES_NO_INDICATION           1087
INJURIES_UNKNOWN                 1087
CRASH_HOUR                          0
CRASH_DAY_OF_WEEK                   0
CRASH_MONTH                         0
LATITUDE                         3049
LONGITUDE                        3049
LOCATION                         3049
dtype: int64
```

In [15]:

```
# Using the global function clean_df to impute null values
df_crashes_clean = clean_df(df_crashes_clean)
df_crashes_clean.isnull().sum()
```

Out[15]:

```
CRASH_RECORD_ID                    0
RD_NO                              0
CRASH_DATE_EST_I                   0
CRASH_DATE                         0
POSTED_SPEED_LIMIT                 0
TRAFFIC_CONTROL_DEVICE             0
DEVICE_CONDITION                   0
WEATHER_CONDITION                  0
LIGHTING_CONDITION                 0
FIRST_CRASH_TYPE                   0
TRAFFICWAY_TYPE                    0
LANE_CNT                           0
ALIGNMENT                          0
ROADWAY_SURFACE_COND               0
ROAD_DEFECT                        0
REPORT_TYPE                        0
CRASH_TYPE                         0
INTERSECTION_RELATED_I             0
NOT_RIGHT_OF_WAY_I                 0
HIT_AND_RUN_I                      0
DAMAGE                             0
DATE_POLICE_NOTIFIED               0
PRIM_CONTRIBUTORY_CAUSE            0
SEC_CONTRIBUTORY_CAUSE             0
STREET_NO                          0
STREET_DIRECTION                   0
STREET_NAME                        0
BEAT_OF_OCCURRENCE                 0
PHOTOS_TAKEN_I                     0
STATEMENTS_TAKEN_I                 0
DOORING_I                          0
WORK_ZONE_I                        0
WORK_ZONE_TYPE                     0
WORKERS_PRESENT_I                  0
NUM_UNITS                          0
MOST_SEVERE_INJURY                 0
INJURIES_TOTAL                     0
INJURIES_FATAL                     0
INJURIES_INCAPACITATING            0
INJURIES_NON_INCAPACITATING        0
INJURIES_REPORTED_NOT_EVIDENT      0
INJURIES_NO_INDICATION             0
INJURIES_UNKNOWN                   0
CRASH_HOUR                         0
CRASH_DAY_OF_WEEK                  0
CRASH_MONTH                        0
LATITUDE                           0
LONGITUDE                          0
LOCATION                           0
dtype: int64
```

5.1 Feature Evaluation

In [16]:

```
#Create a list of all columns
num_cols = [col for col in df_crashes_clean.columns if df_
cat_cols= [col for col in df_crashes_clean.columns if df_c
print(f"There are {len(num_cols)} numerical columns : \n {
print("\n")
print(f"There are {len(cat_cols)} categorical columns : \n
```

```
There are 16 numerical columns :
 ['POSTED_SPEED_LIMIT', 'STREET_NO', 'BEAT_OF_OCCURRENCE', 'NUM_U
JURIES_TOTAL', 'INJURIES_FATAL', 'INJURIES_INCAPACITATING', 'INJU
NCAPACITATING', 'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INI
'INJURIES_UNKNOWN', 'CRASH_HOUR', 'CRASH_DAY_OF_WEEK', 'CRASH_MO
TUDE', 'LONGITUDE']


There are 33 categorical columns :
 ['CRASH_RECORD_ID', 'RD_NO', 'CRASH_DATE_EST_I', 'CRASH_DATE',
NTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CONDITION', 'LIGHTIN
N', 'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE', 'LANE_CNT', 'ALIGNMEN
Y_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE', 'CRASH_TYPE', 'INT
RELATED_I', 'NOT_RIGHT_OF_WAY_I', 'HIT_AND_RUN_I', 'DAMAGE', 'DA
OTIFIED', 'PRIM_CONTRIBUTORY_CAUSE', 'SEC_CONTRIBUTORY_CAUSE', 'S
CTION', 'STREET_NAME', 'PHOTOS_TAKEN_I', 'STATEMENTS_TAKEN_I', 'I
'WORK_ZONE_I', 'WORK_ZONE_TYPE', 'WORKERS_PRESENT_I', 'MOST_SEVE
'LOCATION']
```

In [17]:

```
# Display first 5 rows of numeric columns
df_crashes_clean.head()
```

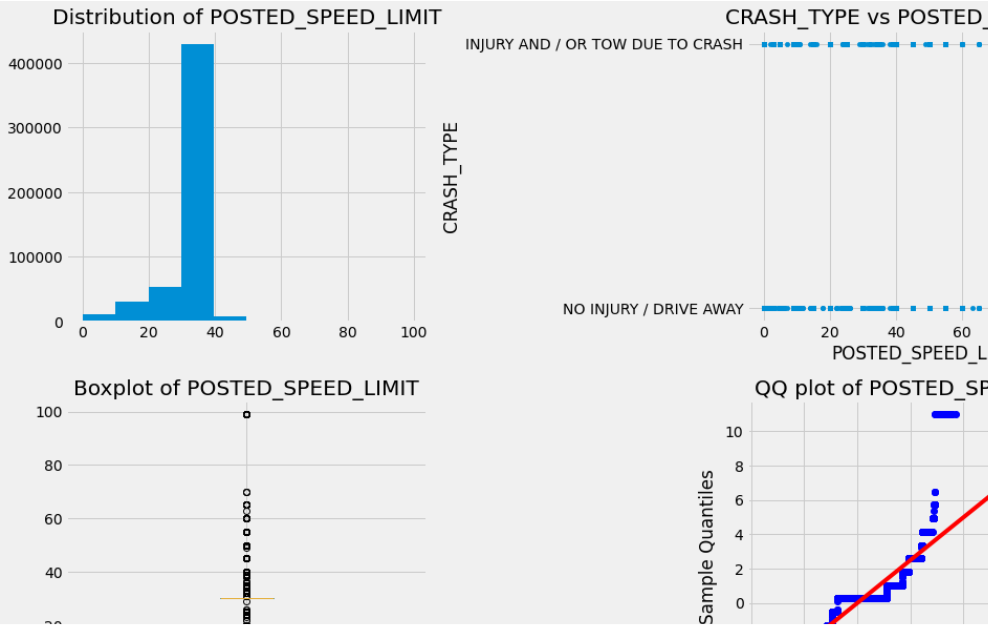
Out[17]:

	CRASH_RECORD_ID	RD_NO	CRASH_DAT
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	JC343143	
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	JA329216	
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	JD292400	
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	JD293602	
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	JD290451	

5 rows × 49 columns

In [18]:

```
1 # posted speed limit summary
2 col_summary(df_crashes_clean, num_col="POSTED_SPEED_LIMIT"
```



Observations

- Does not seem to have extreme outliers

Actions

- Keep all the values in the column

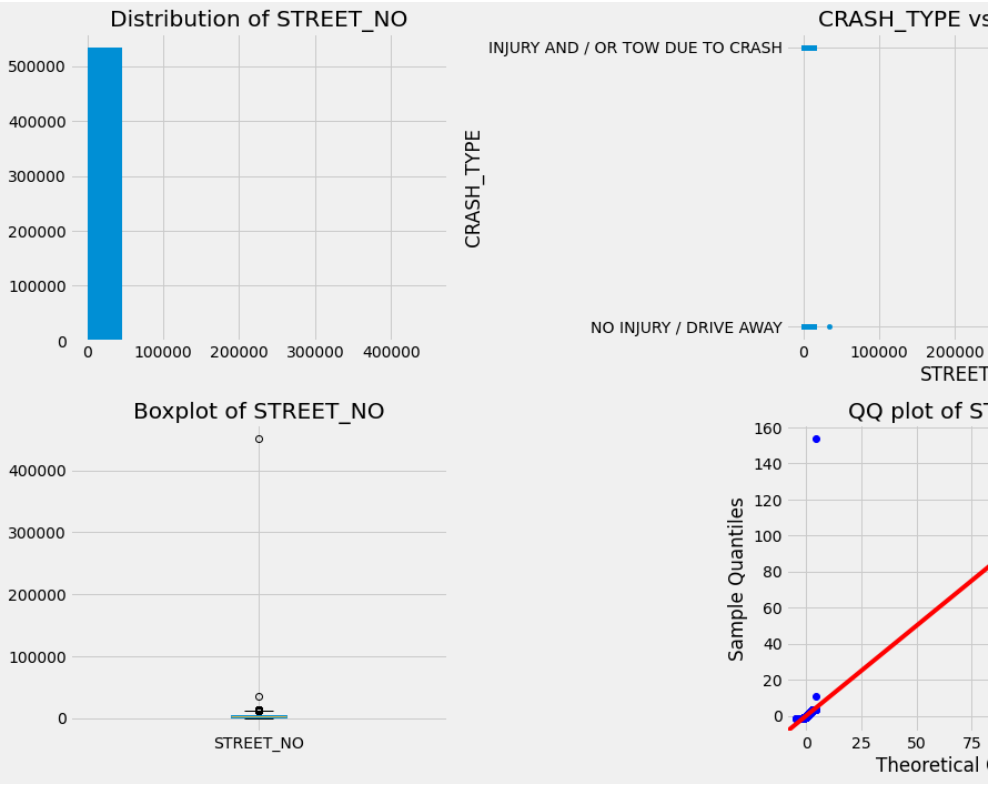
Contents

Contents

In [19]:

```
# Street no summary
col_summary(df_crashes_clean, num_col="STREET_NO")
```

```
Column Name: STREET_NO
Number of unique values: 11217
There are 522396 duplicates
There are 0 null values
There are 2 zeros


Value Counts Percentage
 100     1.0
1600    1.0
800     1.0
200     1.0
300     1.0
        ..
688     0.0
10286   0.0
2283    0.0
6696    0.0
12503   0.0
Name: STREET_NO, Length: 11217, dtype: float64


Descriptive Metrics:
 count             533,613.0
mean     3,663.269457453248
std     2,910.2447183927966
min                     0.0
25%                 1,215.0
50%                 3,199.0
75%                 5,599.0
max               451,100.0
Name: STREET_NO, dtype: float64
```



Observations

- `STREET_NO` should be changed to a categorical variable as it is a u

Actions

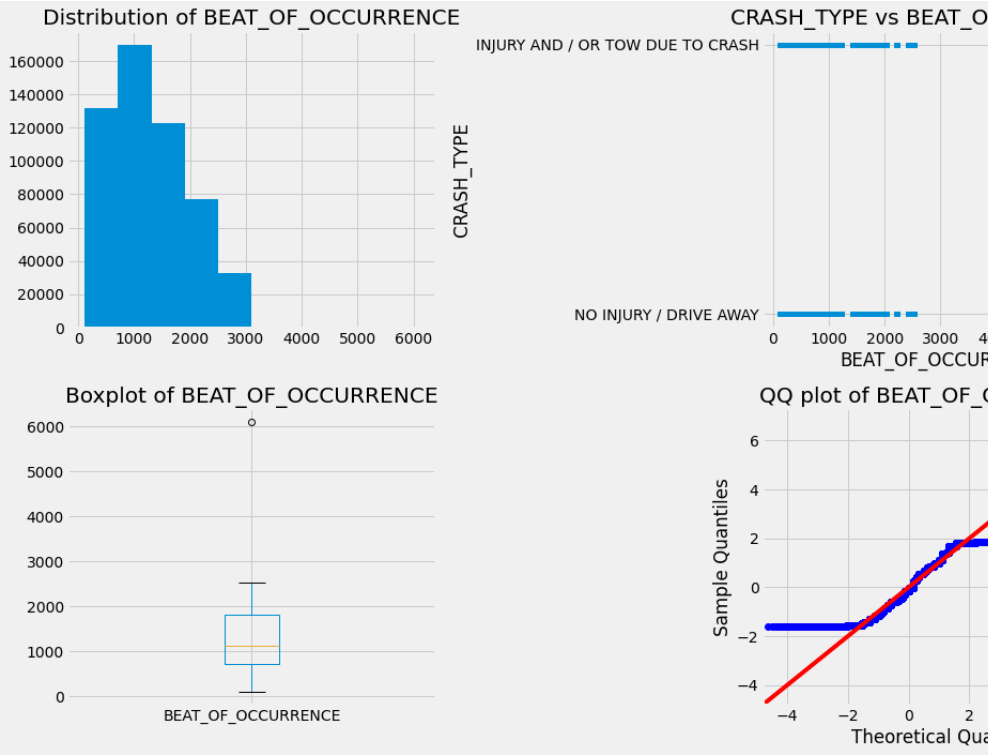
- Recast `STREET_NO` as categorical

Contents

In [20]:

```
1 #Summary of BEAT_OF_OCCURRENCE
2 col_summary(df_crashes_clean, num_col="BEAT_OF_OCCURRENCE"
```

```
Column Name: BEAT_OF_OCCURRENCE
Number of unique values: 275
There are 533338 duplicates
There are 0 null values
There are 0 zeros


Value Counts Percentage
 1,834.0   1.0
114.0     1.0
1,831.0   1.0
122.0     1.0
813.0     1.0
           ..
1,125.0   0.0
1,653.0   0.0
1,652.0   0.0
1,655.0   0.0
6,100.0   0.0
Name: BEAT_OF_OCCURRENCE, Length: 275, dtype: float64


Descriptive Metrics:
 count             533,613.0
mean    1,238.3659093762708
std       706.9747211922776
min                   111.0
25%                   712.0
50%                 1,135.0
75%                 1,822.0
max                 6,100.0
Name: BEAT_OF_OCCURRENCE, dtype: float64
```



Observations

- Needs to be changed to a categorical variable as it is an identifier.
- *Actions**
- Recast `BEAT_OF_OCCURRENCE` as a categorical variable.

Contents

Contents

In [21]:

```
#Summary of NUM_UNITS
col_summary(df_crashes_clean, num_col = "NUM_UNITS")
```

```
Column Name: NUM_UNITS
Number of unique values: 17
There are 533596 duplicates
There are 0 null values
There are 5 zeros


Value Counts Percentage
 2.0    88.0
3.0     5.0
1.0     5.0
4.0     1.0
5.0     0.0
6.0     0.0
7.0     0.0
8.0     0.0
9.0     0.0
10.0    0.0
11.0    0.0
0.0     0.0
12.0    0.0
14.0    0.0
15.0    0.0
18.0    0.0
16.0    0.0
Name: NUM_UNITS, dtype: float64


Descriptive Metrics:
 count            533,613.0
mean     2.033692957255539
std     0.4467122099720147
min                    0.0
25%                    2.0
50%                    2.0
75%                    2.0
max                   18.0
Name: NUM_UNITS, dtype: float64
```

Contents

Observations

- There are outliers present here.

Actions

- Remove outliers from `NUM_UNITS`

In [22]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_TOTAL")
```

```
Column Name: INJURIES_TOTAL
Number of unique values: 19
There are 533594 duplicates
There are 0 null values
There are 462695 zeros


Value Counts Percentage
 0.0                   87.0
1.0                   10.0
2.0                    2.0
3.0                    1.0
4.0                    0.0
0.18022406417714815    0.0
5.0                    0.0
6.0                    0.0
7.0                    0.0
8.0                    0.0
9.0                    0.0
10.0                   0.0
15.0                   0.0
11.0                   0.0
13.0                   0.0
21.0                   0.0
12.0                   0.0
19.0                   0.0
16.0                   0.0
Name: INJURIES_TOTAL, dtype: float64


Descriptive Metrics:
 count             533,613.0
mean    0.18022406417714817
std      0.5518904417749386
min                     0.0
25%                     0.0
50%                     0.0
75%                     0.0
max                    21.0
Name: INJURIES_TOTAL, dtype: float64
```

Contents

Observations

- Maybe useful for modeling by engineering features

Actions

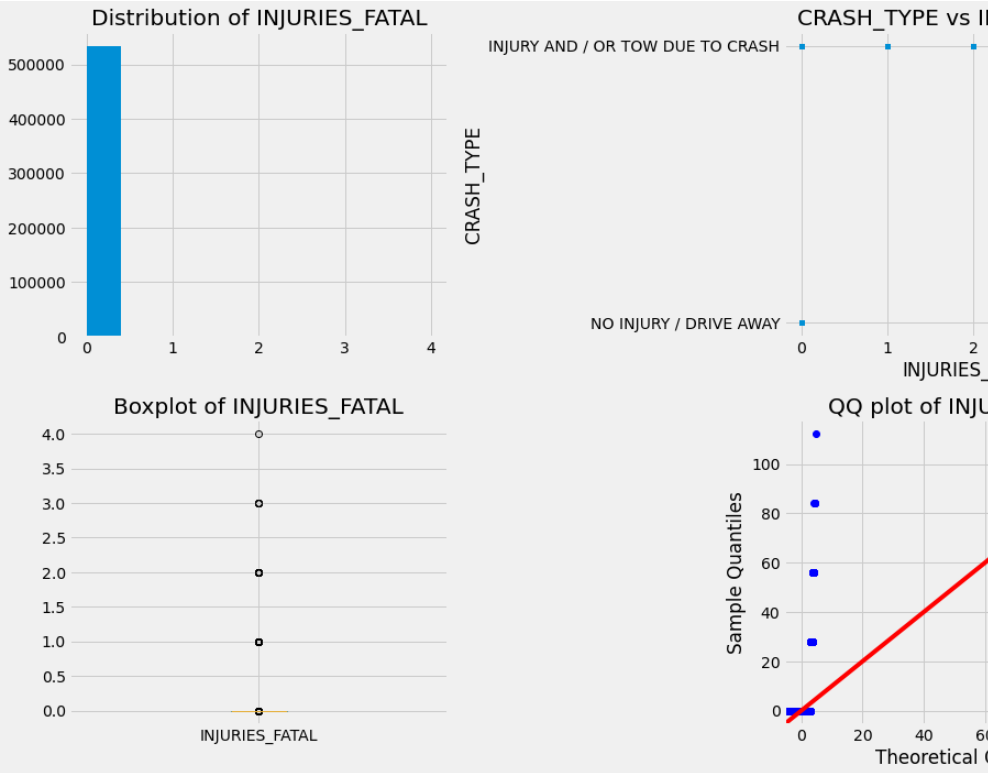
- Keep the column `INJURIES_TOTAL`

Contents

In [23]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_FATAL")
```

```
Column Name: INJURIES_FATAL
Number of unique values: 6
There are 533607 duplicates
There are 0 null values
There are 531991 zeros


Value Counts Percentage
 0.0                    100.0
0.0010835151710902379     0.0
1.0                       0.0
2.0                       0.0
3.0                       0.0
4.0                       0.0
Name: INJURIES_FATAL, dtype: float64


Descriptive Metrics:
 count              533,613.0
mean    0.001083515171090238
std     0.035602534771029544
min                      0.0
25%                      0.0
50%                      0.0
75%                      0.0
max                      4.0
Name: INJURIES_FATAL, dtype: float64
```



Observations

- Seems to be useful for modeling

Actions

- Keep the column `INJURIES_FATAL`

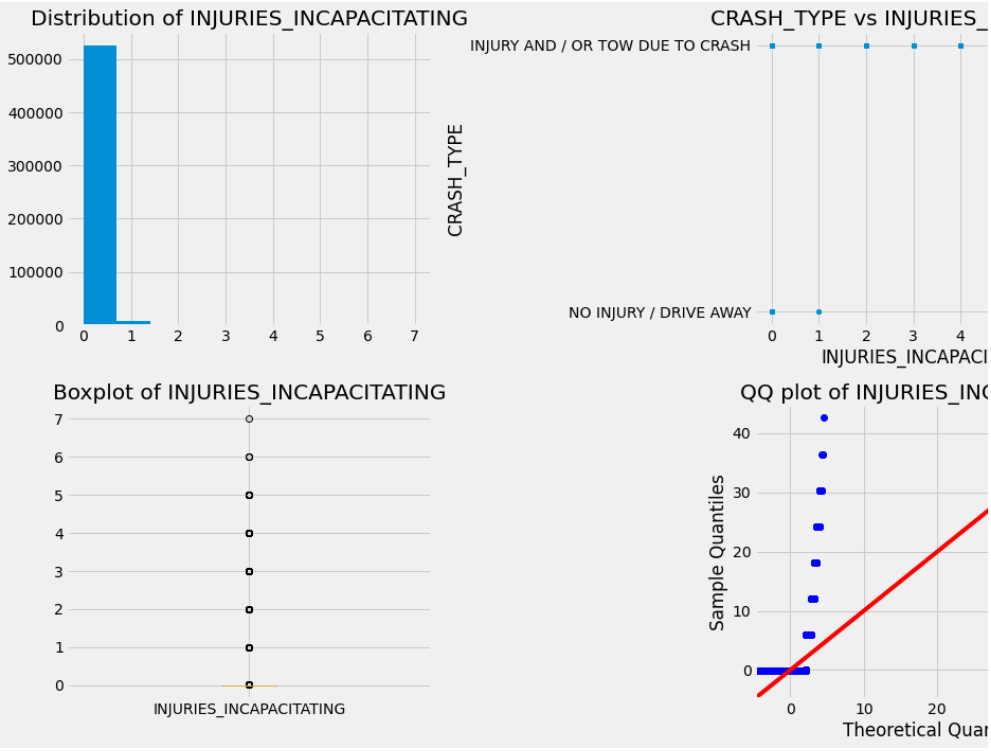
Contents

Contents

In [24]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_INCAPACITA
```

```
Column Name: INJURIES_INCAPACITATING
Number of unique values: 9
There are 533604 duplicates
There are 0 null values
There are 523546 zeros


Value Counts Percentage
 0.0                    98.0
1.0                     1.0
0.019676034597371772    0.0
2.0                     0.0
3.0                     0.0
4.0                     0.0
5.0                     0.0
6.0                     0.0
7.0                     0.0
Name: INJURIES_INCAPACITATING, dtype: float64


Descriptive Metrics:
 count              533,613.0
mean    0.019676034597371776
std      0.16395317797622694
min                      0.0
25%                      0.0
50%                      0.0
75%                      0.0
max                      7.0
Name: INJURIES_INCAPACITATING, dtype: float64
```



Observations

- `INJURIES_INCAPACITATING` has many zeros and needs to evalua

Actions

- Check for outliers in the column.
- Keep the column.

Contents

Contents

In [25]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_NON_INCAPA
```

```
Column Name: INJURIES_NON_INCAPACITATING
Number of unique values: 18
There are 533595 duplicates
There are 0 null values
There are 491937 zeros


Value Counts Percentage
 0.0                   92.0
1.0                    6.0
2.0                    1.0
3.0                    0.0
0.10063170624532887    0.0
4.0                    0.0
5.0                    0.0
6.0                    0.0
7.0                    0.0
8.0                    0.0
10.0                   0.0
11.0                   0.0
9.0                    0.0
21.0                   0.0
12.0                   0.0
18.0                   0.0
16.0                   0.0
14.0                   0.0
Name: INJURIES_NON_INCAPACITATING, dtype: float64


Descriptive Metrics:
 count             533,613.0
mean    0.10063170624532888
std     0.40967550668233826
min                     0.0
25%                     0.0
50%                     0.0
75%                     0.0
max                    21.0
Name: INJURIES_NON_INCAPACITATING, dtype: float64
```

Contents

Observations

- Doesnt seem useful for modeling

Actions

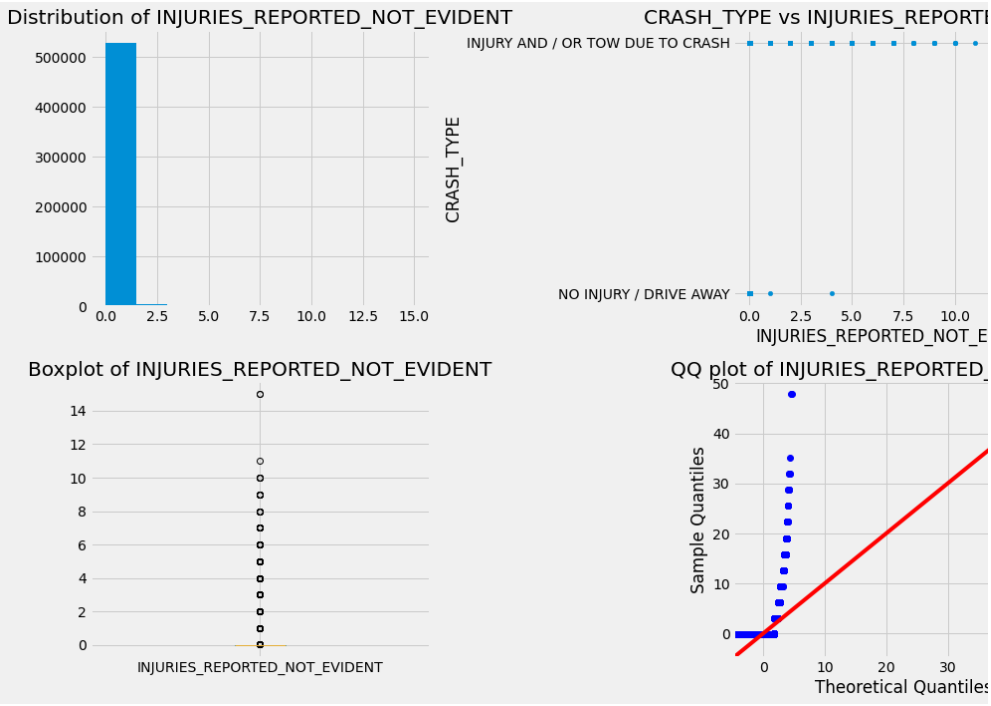
- Drop the column `INJURIES_NON_INCAPACITATING`

Contents

In [26]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_REPORTED_N
```

```
Column Name: INJURIES_REPORTED_NOT_EVIDENT
Number of unique values: 14
There are 533599 duplicates
There are 0 null values
There are 508655 zeros


Value Counts Percentage
 0.0                    95.0
1.0                     3.0
2.0                     1.0
0.05883280816335728     0.0
3.0                     0.0
4.0                     0.0
5.0                     0.0
6.0                     0.0
7.0                     0.0
8.0                     0.0
9.0                     0.0
10.0                    0.0
15.0                    0.0
11.0                    0.0
Name: INJURIES_REPORTED_NOT_EVIDENT, dtype: float64


Descriptive Metrics:
 count              533,613.0
mean    0.05883280816335727
std     0.31134111932104425
min                     0.0
25%                     0.0
50%                     0.0
75%                     0.0
max                    15.0
Name: INJURIES_REPORTED_NOT_EVIDENT, dtype: float64
```



Observations about 95% of the values are 0 and the data schema does
it means, I will drop this column from analysis.

Actions

- Drop `INJURIES_REPORTED_NOT_EVIDENT`

Contents

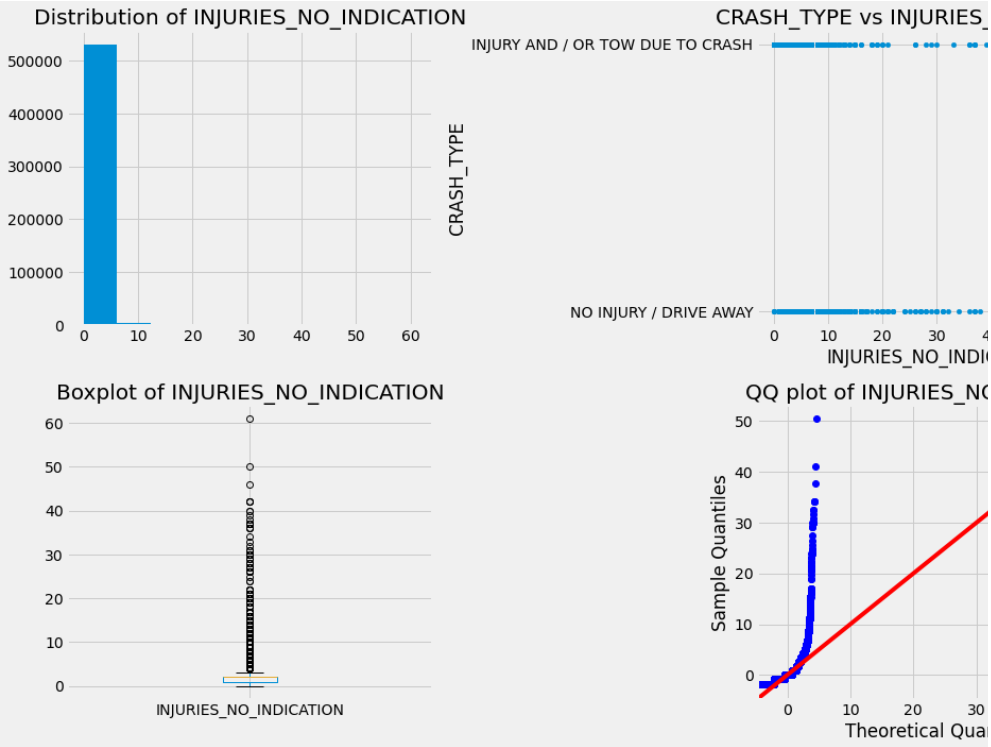
Contents

In [27]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_NO_INDICAT
```

```
Column Name: INJURIES_NO_INDICATION
Number of unique values: 44
There are 533569 duplicates
There are 0 null values
There are 10350 zeros


Value Counts Percentage
 2.0                 46.0
1.0                 30.0
3.0                 13.0
4.0                  5.0
5.0                  2.0
0.0                  2.0
6.0                  1.0
7.0                  0.0
2.019732369874898    0.0
8.0                  0.0
9.0                  0.0
10.0                 0.0
11.0                 0.0
12.0                 0.0
14.0                 0.0
13.0                 0.0
16.0                 0.0
15.0                 0.0
17.0                 0.0
20.0                 0.0
21.0                 0.0
30.0                 0.0
37.0                 0.0
18.0                 0.0
19.0                 0.0
22.0                 0.0
27.0                 0.0
26.0                 0.0
28.0                 0.0
42.0                 0.0
36.0                 0.0
31.0                 0.0
40.0                 0.0
29.0                 0.0
24.0                 0.0
46.0                 0.0
32.0                 0.0
50.0                 0.0
25.0                 0.0
39.0                 0.0
38.0                 0.0
34.0                 0.0
33.0                 0.0
61.0                 0.0
Name: INJURIES_NO_INDICATION, dtype: float64


Descriptive Metrics:
 count            533,613.0
```

```
mean    2.0197323698748977
std     1.1675589724470028
min                    0.0
25%                    1.0
50%                    2.0
75%                    2.0
max                   61.0
Name: INJURIES_NO_INDICATION, dtype: float64
```



Observations

- Column seems to useful for classification

Actions

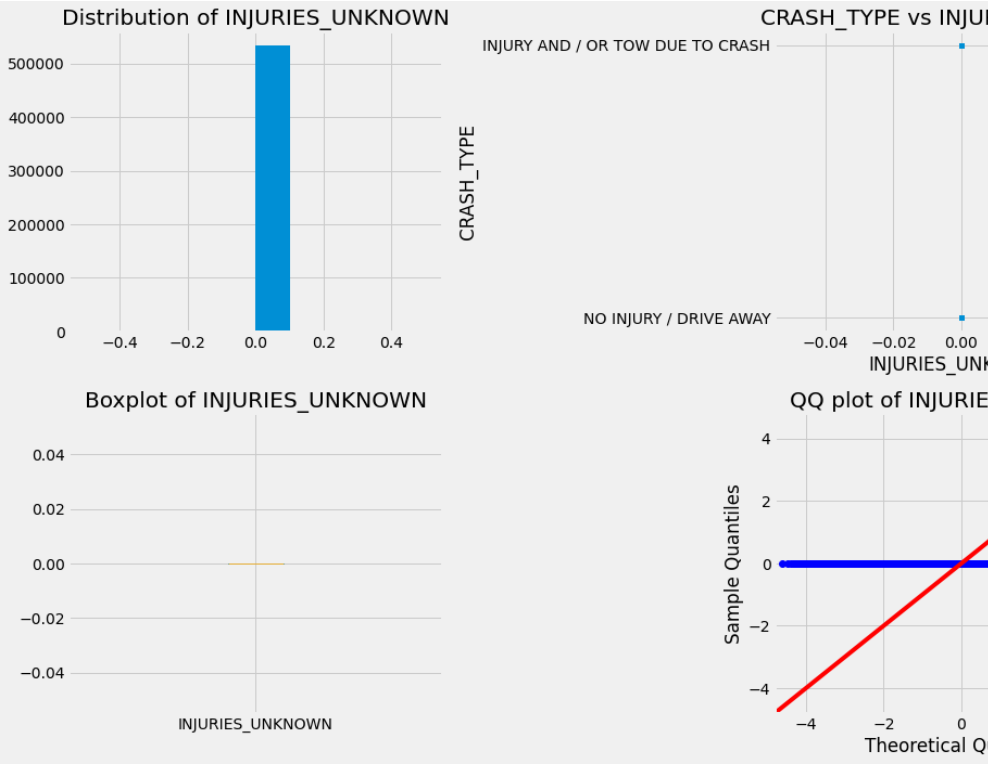
- Keep the column

Contents

In [28]:

```
1 col_summary(df_crashes_clean, num_col="INJURIES_UNKNOWN")
```

```
Column Name: INJURIES_UNKNOWN
Number of unique values: 1
There are 533612 duplicates
There are 0 null values
There are 533613 zeros


Value Counts Percentage
 0.0   100.0
Name: INJURIES_UNKNOWN, dtype: float64


Descriptive Metrics:
 count   533,613.0
mean           0.0
std            0.0
min            0.0
25%            0.0
50%            0.0
75%            0.0
max            0.0
Name: INJURIES_UNKNOWN, dtype: float64
```



Contents

Observations

- Doesnt seem to be a useful column

Actions

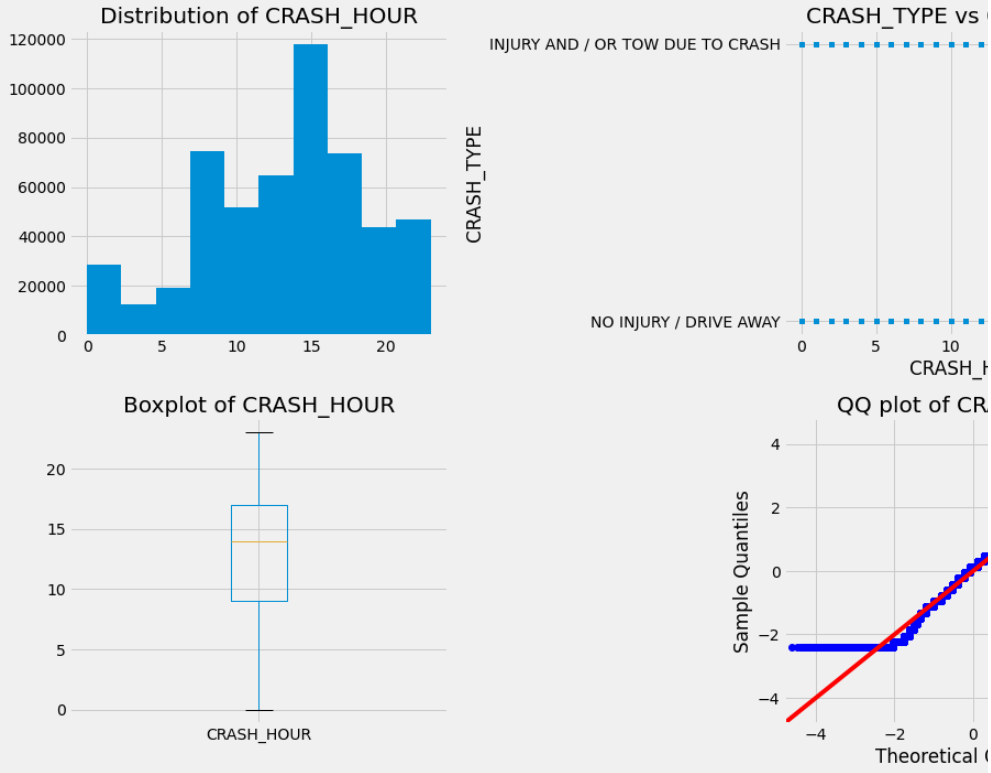
- Drop `INJURIES_UNKNOWN`

Contents

In [29]:

```
1 col_summary(df_crashes_clean, num_col="CRASH_HOUR")
```

```
Column Name: CRASH_HOUR
Number of unique values: 24
There are 533589 duplicates
There are 0 null values
There are 11063 zeros


Value Counts Percentage
 16                 8.0
15                 8.0
17                 8.0
14   7.000000000000001
18                 6.0
13                 6.0
12                 6.0
8                  5.0
11                 5.0
9                  5.0
10                 5.0
19                 5.0
7                  4.0
20                 4.0
21                 3.0
22                 3.0
23                 3.0
6                  2.0
0                  2.0
1                  2.0
2                  2.0
5                  1.0
3                  1.0
4                  1.0
Name: CRASH_HOUR, dtype: float64


Descriptive Metrics:
 count            533,613.0
mean    13.249231184397681
std      5.517232054949403
min                    0.0
25%                    9.0
50%                   14.0
75%                   17.0
max                   23.0
Name: CRASH_HOUR, dtype: float64
```



Observations

- Seems useful for modeling

Actions

- Keep the column

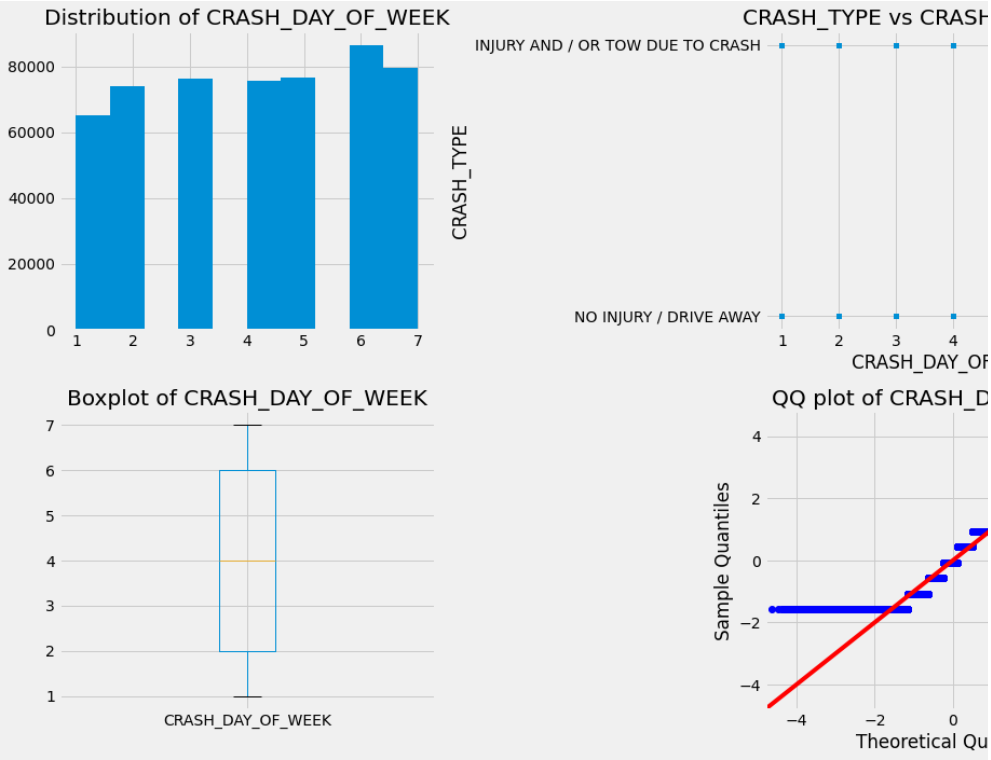
Contents

Contents

In [30]:

```
1 col_summary(df_crashes_clean, num_col="CRASH_DAY_OF_WEEK")
```

```
Column Name: CRASH_DAY_OF_WEEK
Number of unique values: 7
There are 533606 duplicates
There are 0 null values
There are 0 zeros


Value Counts Percentage
 6                 16.0
7                 15.0
5   14.000000000000002
3   14.000000000000002
4   14.000000000000002
2   14.000000000000002
1                 12.0
Name: CRASH_DAY_OF_WEEK, dtype: float64


Descriptive Metrics:
 count           533,613.0
mean    4.125532923673149
std      1.97793398152501
min                   1.0
25%                   2.0
50%                   4.0
75%                   6.0
max                   7.0
Name: CRASH_DAY_OF_WEEK, dtype: float64
```



Observations

- Doesnt seem to be useful

Actions

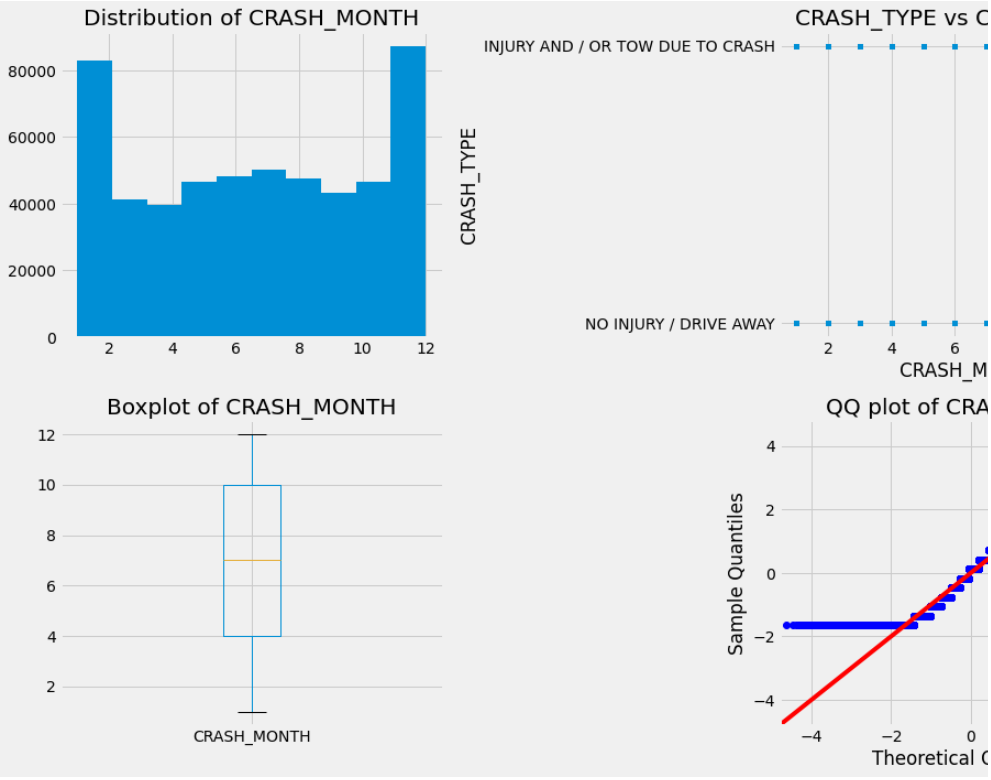
- Drop `CRASH_DAY_OF_WEEK`

Contents

In [31]:

```
1 col_summary(df_crashes_clean, num_col="CRASH_MONTH")
```

```
Column Name: CRASH_MONTH
Number of unique values: 12
There are 533601 duplicates
There are 0 null values
There are 0 zeros


Value Counts Percentage
 7                  9.0
6                   9.0
8                   9.0
5                   9.0
10                  9.0
12                  8.0
9                   8.0
11                  8.0
1                   8.0
3                   8.0
2                   8.0
4    7.000000000000001
Name: CRASH_MONTH, dtype: float64


Descriptive Metrics:
 count           533,613.0
mean    6.595834434318504
std     3.389520115976073
min                   1.0
25%                   4.0
50%                   7.0
75%                  10.0
max                  12.0
Name: CRASH_MONTH, dtype: float64
```



Observations

- Doesnt seem to be useful

Actions

- Drop `CRASH_MONTH`

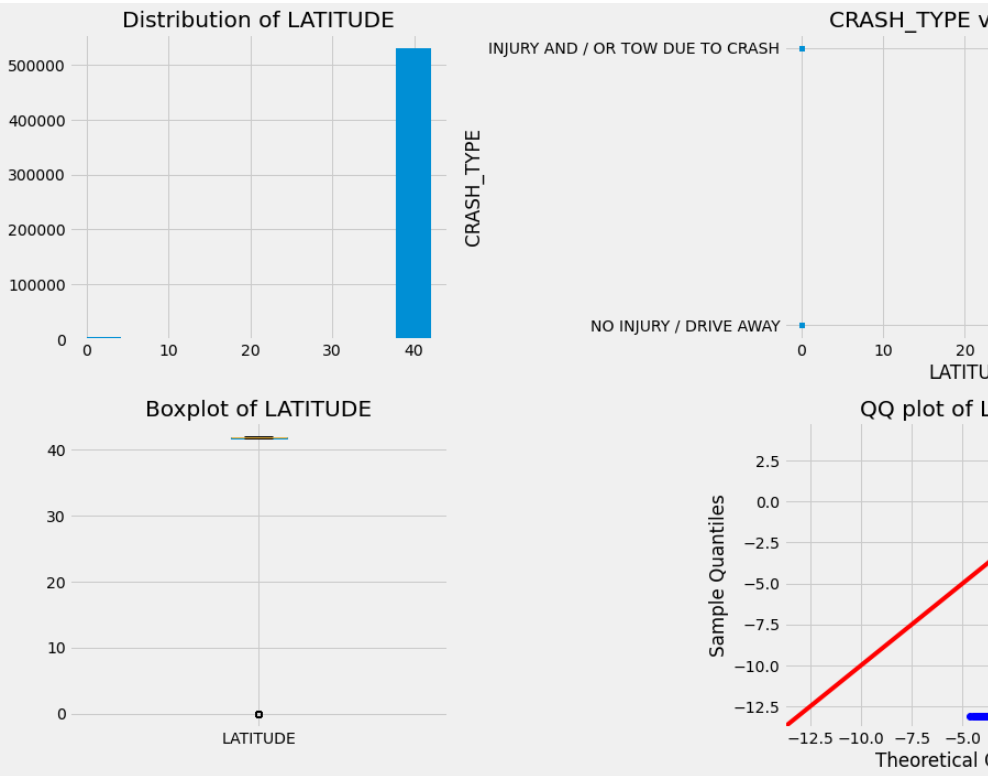
Contents

Contents

In [32]:

```
1 col_summary(df_crashes_clean, num_col="LATITUDE")
```

```
Column Name: LATITUDE
Number of unique values: 226462
There are 307151 duplicates
There are 0 null values
There are 3081 zeros


Value Counts Percentage
 0.0                  1.0
41.976201139000004   0.0
41.751460603000005   0.0
41.791420282         0.0
41.722257273000004   0.0
                      ..
41.713529385         0.0
41.684149414000004   0.0
41.681025124         0.0
41.68694248          0.0
42.000060675         0.0
Name: LATITUDE, Length: 226462, dtype: float64


Descriptive Metrics:
 count            533,613.0
mean     41.61569626170536
std      3.172531407071622
min                    0.0
25%            41.78031747
50%            41.874654272
75%     41.923797791999995
max            42.022779861
Name: LATITUDE, dtype: float64
```



Observations

- Latitude should be a categorical as it is an identifier

Actions

- Recast `LATITUDE` as a categorical feature

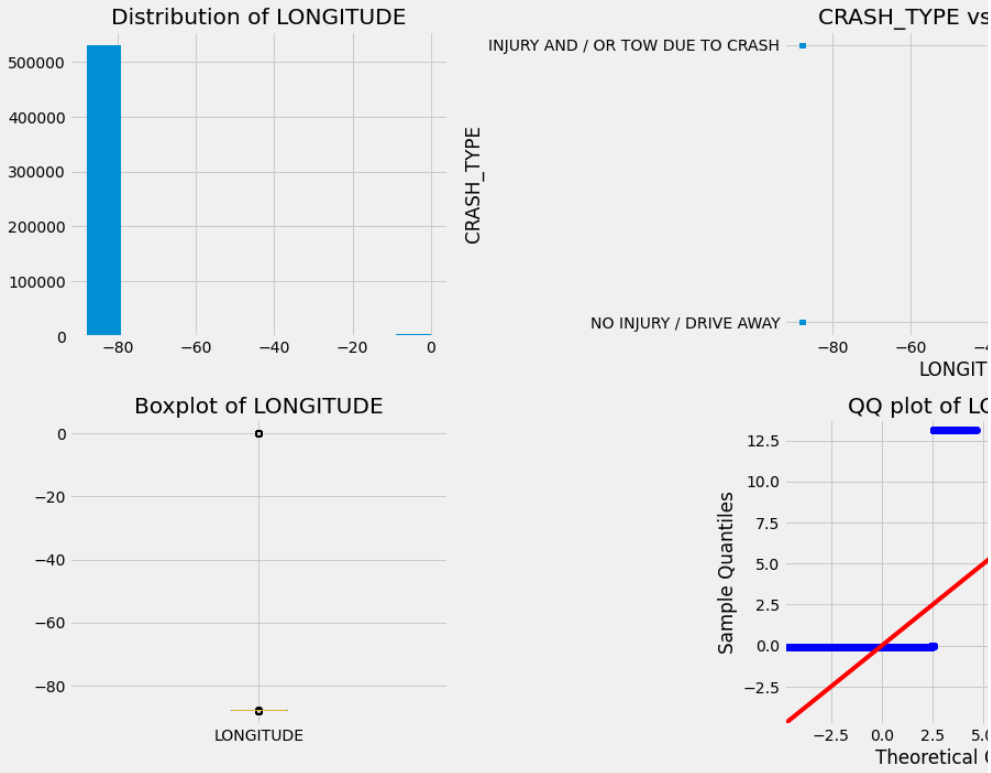
Contents

Contents

In [33]:

```
1 col_summary(df_crashes_clean, num_col="LONGITUDE")
```

```
Column Name: LONGITUDE
Number of unique values: 226440
There are 307173 duplicates
There are 0 null values
There are 3081 zeros


Value Counts Percentage
 0.0                  1.0
-87.905309125         0.0
-87.58597199299999    0.0
-87.58014776899999    0.0
-87.585275565         0.0
                       ..
-87.696262158         0.0
-87.80691731799999    0.0
-87.67134047          0.0
-87.740695279         0.0
-87.551456265         0.0
Name: LONGITUDE, Length: 226440, dtype: float64


Descriptive Metrics:
 count            533,613.0
mean    -87.17201151702875
std      6.643308410037887
min            -87.93587692
25%           -87.720961428
50%     -87.67277405600001
75%           -87.632362725
max                     0.0
Name: LONGITUDE, dtype: float64
```



OBSERVATION

- `LONGITUDE` seems to be a categorical variable as it is an identifier.

Action

- Recast `LONGITUDE` as a categorical variable/

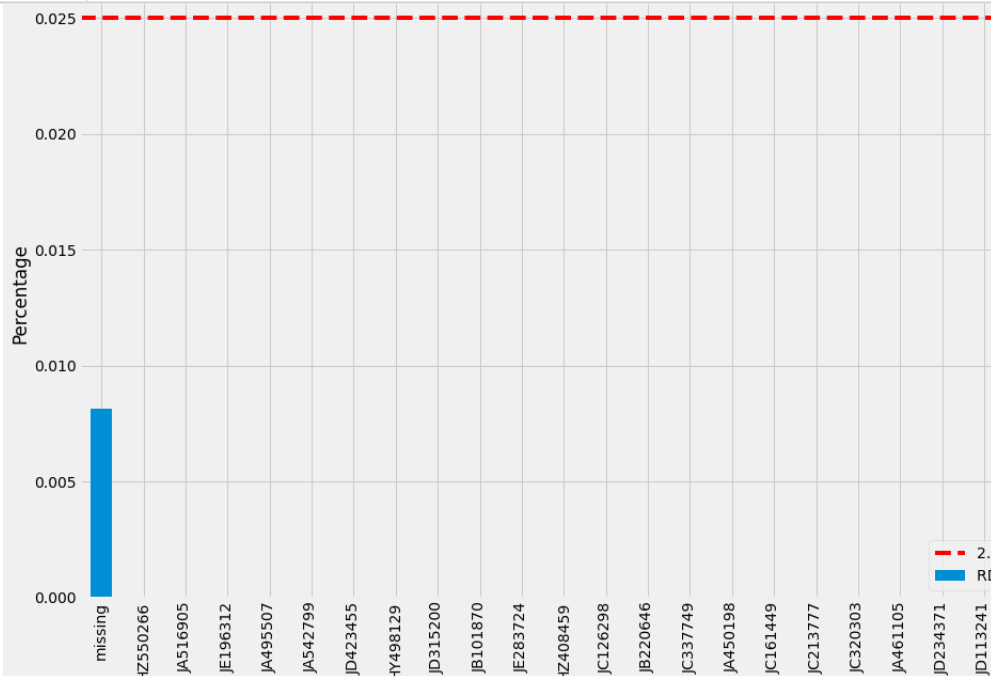
In [34]:

```
1 col_summary(df_crashes_clean, cat_cols = ["CRASH_RECORD_ID
```

```
============================
Column Name: CRASH_RECORD_ID


Number of unique values: 533613


There are 0 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 abadea8b44b10578de88a54a4e40f4c2e5d01ba04d9257e7b53b3ae2ea0343b
c7f253d5773c2c9a3713e018edd266628f53ed1e407b9d22cb1154d1    1
```

In [35]:

```
1 col_summary(df_crashes_clean, cat_cols = ["RD_NO"])
```



Contents

In [36]:

```
1 col_summary(df_crashes_clean, cat_cols = ["CRASH_DATE", "C
```

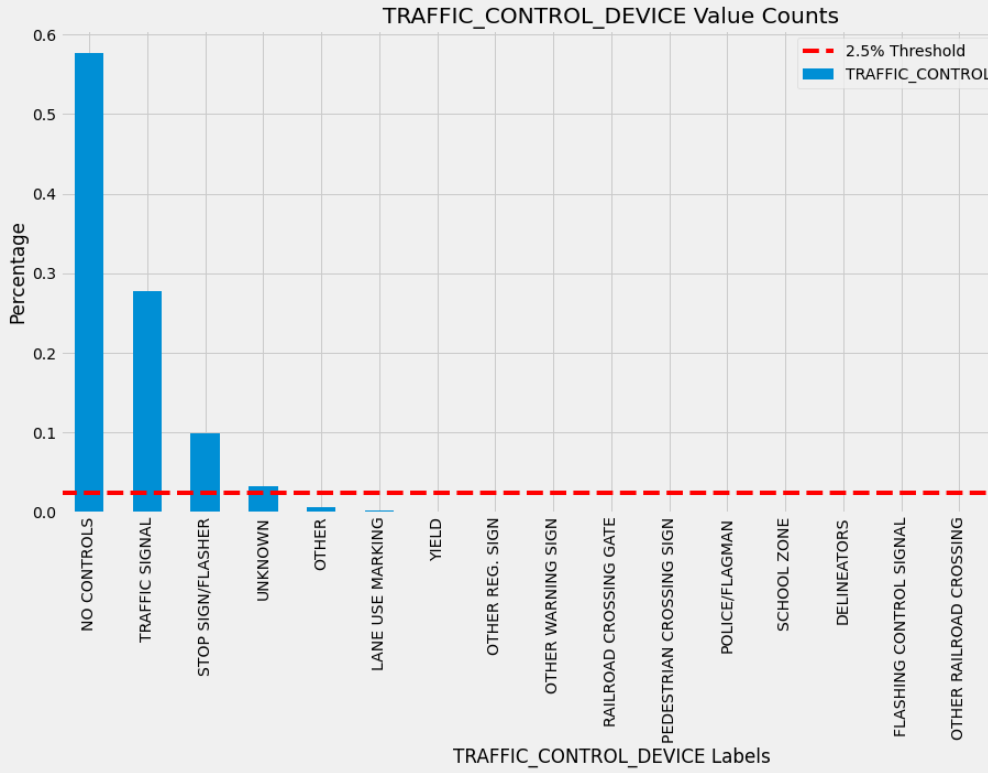
```
============================
Column Name: CRASH_DATE


Number of unique values: 348095


There are 185518 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 12/29/2020 05:00:00 PM    30
11/10/2017 10:30:00 AM    27
```

Contents

In [37]:

```
1 col_summary(df_crashes_clean, cat_cols=["TRAFFIC_CONTROL_D
```

```
============================
Column Name: TRAFFIC_CONTROL_DEVICE


Number of unique values: 19


There are 533594 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 NO CONTROLS                307617
TRAFFIC SIGNAL              148124
STOP SIGN/FLASHER            52848
UNKNOWN                      17261
OTHER                         3257
LANE USE MARKING              1226
YIELD                          753
OTHER REG. SIGN                547
OTHER WARNING SIGN             470
RAILROAD CROSSING GATE         345
PEDESTRIAN CROSSING SIGN       237
POLICE/FLAGMAN                 196
SCHOOL ZONE                    175
DELINEATORS                    169
FLASHING CONTROL SIGNAL        164
OTHER RAILROAD CROSSING        124
RR CROSSING SIGN                58
NO PASSING                      25
BICYCLE CROSSING SIGN           17
Name: TRAFFIC_CONTROL_DEVICE, dtype: int64
```



Contents

In [38]:

```
1 col_summary(df_crashes_clean, cat_cols=["DEVICE_CONDITION"
```

```
============================
Column Name: DEVICE_CONDITION


Number of unique values: 8


There are 533605 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 NO CONTROLS                 310825
FUNCTIONING PROPERLY         184424
```

Contents

In [39]:

```
col_summary(df_crashes_clean, cat_cols=['FIRST_CRASH_TYPE'
 'TRAFFICWAY_TYPE','REPORT_TYPE', 'CRASH_TYPE'])
```

```
============================
Column Name: FIRST_CRASH_TYPE


Number of unique values: 18


There are 533595 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 REAR END                       125539
PARKED MOTOR VEHICLE            124116
```

In [40]:

```
col_summary(df_crashes_clean, cat_cols=['LANE_CNT', 'ALIG
'ROAD_DEFECT'])
```

```
============================
Column Name: LANE_CNT


Number of unique values: 61


There are 533552 duplicates


There are 0 null values


There are 1013 zeros


Value Counts Percentage
 missing      334646
2.0           79498
```

Contents

In [41]:

```
col_summary(df_crashes_clean, cat_cols=['INTERSECTION_RELA
 'NOT_RIGHT_OF_WAY_I', 'HIT_AND_RUN_I'])
```

```
============================
Column Name: INTERSECTION_RELATED_I


Number of unique values: 3


There are 533610 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 missing    412702
Y           115203
```

In [42]:

```
col_summary(df_crashes_clean, cat_cols=['DAMAGE', 'DATE_PO
```

```
============================
Column Name: DAMAGE


Number of unique values: 3


There are 533610 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 OVER $1,500      312779
$501 - $1,500    153690
```

Contents

In [43]:

```
1 col_summary(df_crashes_clean, cat_cols=["PRIM_CONTRIBUTORY
```

```
============================
Column Name: PRIM_CONTRIBUTORY_CAUSE


Number of unique values: 40


There are 533573 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 UNABLE TO DETERMINE
199424
FAILING TO YIELD RIGHT-OF-WAY
```

In [44]:

```
1 col_summary(df_crashes_clean, cat_cols=['STREET_DIRECTION'
```

```
============================
Column Name: STREET_DIRECTION


Number of unique values: 5


There are 533608 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 W          189912
S          178751
N          127501
```

In [45]:

```
col_summary(df_crashes_clean, cat_cols=['PHOTOS_TAKEN_I',
 'WORK_ZONE_TYPE', 'WORKERS_PRESENT_I'])
```

```
============================
Column Name: PHOTOS_TAKEN_I


Number of unique values: 3


There are 533610 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 missing    526952
Y            5159
```

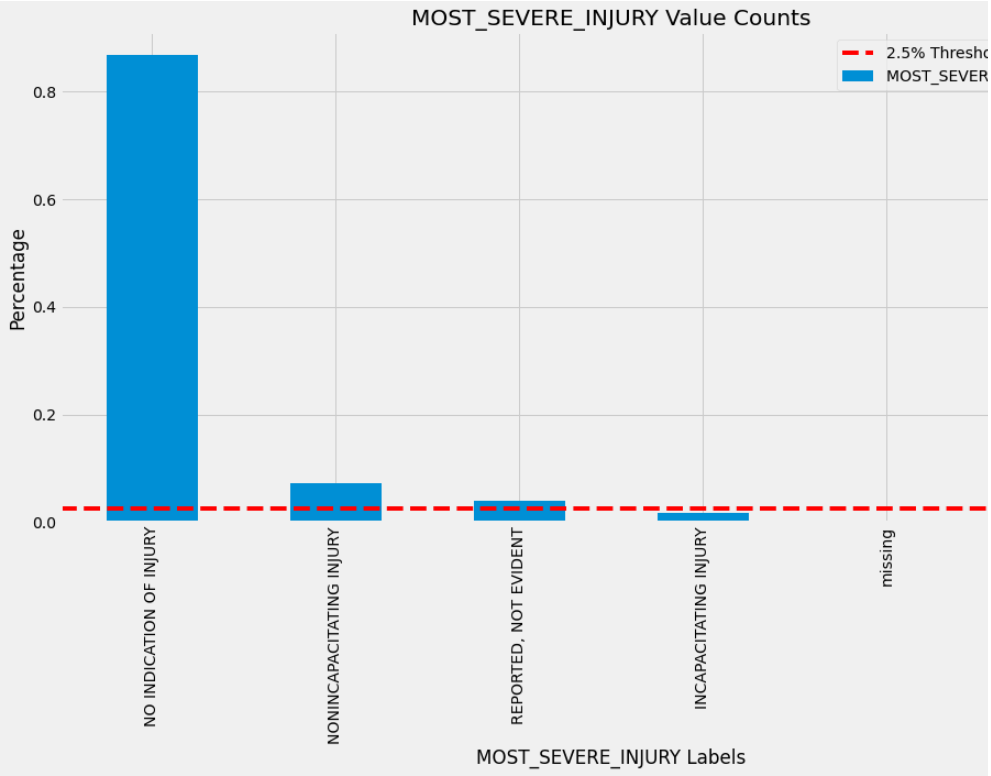
Contents

Contents

In [46]:

```
1 col_summary(df_crashes_clean, cat_cols=["MOST_SEVERE_INJUR
```

```
============================
Column Name: MOST_SEVERE_INJURY


Number of unique values: 6


There are 533607 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 NO INDICATION OF INJURY     462684
NONINCAPACITATING INJURY     38951
REPORTED, NOT EVIDENT        21451
INCAPACITATING INJURY         8894
missing                       1098
FATAL                          535
Name: MOST_SEVERE_INJURY, dtype: int64
```

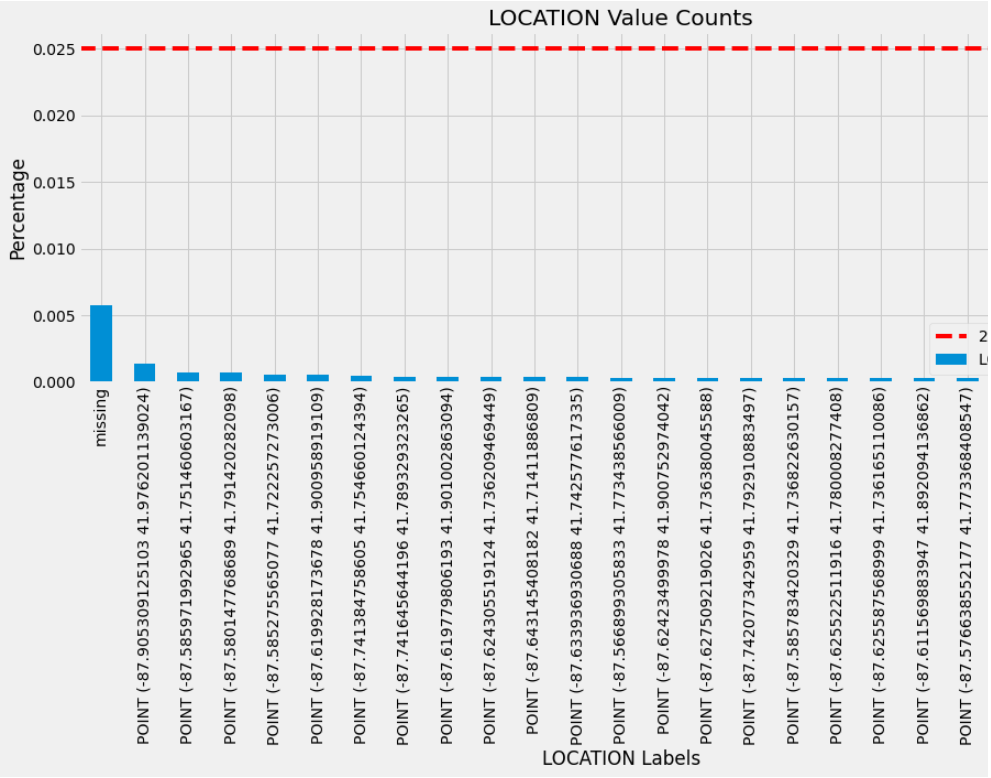


Contents

In [47]:

```
1 col_summary(df_crashes_clean, cat_cols=["LOCATION"])
```

```
============================
Column Name: LOCATION


Number of unique values: 226566


There are 307047 duplicates


There are 0 null values


There are 0 zeros


Value Counts Percentage
 missing                                          3049
POINT (-87.905309125103 41.976201139024)          712
POINT (-87.585971992965 41.751460603167)          383
POINT (-87.580147768689 41.791420282098)          364
POINT (-87.585275565077 41.722257273006)          299
                                                 ...
POINT (-87.762304046473 41.904098719093)            1
POINT (-87.668172579231 41.713989542485)            1
POINT (-87.653231371617 41.984477896755)            1
POINT (-87.715074544297 41.859522313498)            1
POINT (-87.659588397197 41.782801892007)            1
Name: LOCATION, Length: 226566, dtype: int64
```



Feature evaluation done

Summary of actions to take

- recast `STREET_NO` as a string
- recast `BEAT_OF_OCCURRENCE` as a string
- recast `LATITUDE` as a string
- recast `LONGITUDE` as a string
- drop `INJURIES_REPORTED_NOT_EVIDENT` column
- drop `INJURIES_UNKNOWN` column
- drop `CRASH_DAY_OF_WEEK` column
- drop `CRASH_MONTH` column
- drop `CRASH_DATE` column
- drop `CRASH_RECORD_ID` column
- drop `INTERSECTION_RELATED_I` column
- drop `STREET_DIRECTION` column
- drop `STREET_NAME` column
- drop `PHOTOS_TAKEN_I` column
- drop `STATEMENTS_TAKEN_I` column
- drop `WORK_ZONE_I` column
- drop `WORK_ZONE_TYPE` column
- drop `WORKERS_PRESENT_I` column
- drop `LANE_CNT` column
- drop `ALIGNMENT` column

5.2 Data type Recasting

Contents

In [48]:

```
1 df_crashes_clean.dtypes
```

Out[48]:

```
CRASH_RECORD_ID                 object
RD_NO                           object
CRASH_DATE_EST_I                object
CRASH_DATE                      object
POSTED_SPEED_LIMIT               int64
TRAFFIC_CONTROL_DEVICE          object
DEVICE_CONDITION                object
WEATHER_CONDITION               object
LIGHTING_CONDITION              object
FIRST_CRASH_TYPE                object
TRAFFICWAY_TYPE                 object
LANE_CNT                        object
ALIGNMENT                       object
ROADWAY_SURFACE_COND            object
ROAD_DEFECT                     object
REPORT_TYPE                     object
CRASH_TYPE                      object
INTERSECTION_RELATED_I          object
NOT_RIGHT_OF_WAY_I              object
HIT_AND_RUN_I                   object
DAMAGE                          object
DATE_POLICE_NOTIFIED            object
PRIM_CONTRIBUTORY_CAUSE         object
SEC_CONTRIBUTORY_CAUSE          object
STREET_NO                        int64
STREET_DIRECTION                object
STREET_NAME                     object
BEAT_OF_OCCURRENCE             float64
PHOTOS_TAKEN_I                  object
STATEMENTS_TAKEN_I              object
DOORING_I                       object
WORK_ZONE_I                     object
WORK_ZONE_TYPE                  object
WORKERS_PRESENT_I               object
NUM_UNITS                      float64
MOST_SEVERE_INJURY              object
INJURIES_TOTAL                 float64
INJURIES_FATAL                 float64
INJURIES_INCAPACITATING        float64
INJURIES_NON_INCAPACITATING    float64
INJURIES_REPORTED_NOT_EVIDENT  float64
INJURIES_NO_INDICATION         float64
INJURIES_UNKNOWN               float64
CRASH_HOUR                       int64
CRASH_DAY_OF_WEEK                int64
CRASH_MONTH                      int64
LATITUDE                       float64
LONGITUDE                      float64
LOCATION                        object
dtype: object
```

Contents

In [49]:

```
#convert STREET_NO to categorical
df_crashes_clean["STREET_NO"] = df_crashes_clean["STREET_N
```

In [50]:

```
#Convert BEAT_OF_OCCURRENCE to categorical
df_crashes_clean["BEAT_OF_OCCURRENCE"] = df_crashes_clean[
```

In [51]:

```
# Convert LATITUDE to categorical
df_crashes_clean["LATITUDE"] = df_crashes_clean["LATITUDE"
```

In [52]:

```
df_crashes_clean["LONGITUDE"] = df_crashes_clean["LONGITUD
```

In [53]:

```
df_crashes_clean["CRASH_DATE_YR"] = pd.to_datetime(df_cras
```

In [54]:

```
df_crashes_clean["CRASH_TYPE"] = df_crashes_clean["CRASH_T
df_crashes_clean["CRASH_TYPE"] = df_crashes_clean["CRASH_T
```

In [55]:

```
1 df_crashes_clean
```

Out[55]:

	CRASH_RECORD_ID	RD_NO	CRASH
0	4fd0a3e0897b3335b94cd8d5b2d2b350eb691add56c62d...	JC343143	
1	009e9e67203442370272e1a13d6ee51a4155dac65e583d...	JA329216	
2	ee9283eff3a55ac50ee58f3d9528ce1d689b1c4180b4c4...	JD292400	
3	f8960f698e870ebdc60b521b2a141a5395556bc3704191...	JD293602	
4	8eaa2678d1a127804ee9b8c35ddf7d63d913c14eda61d6...	JD290451	
...	...	...	
533608	9a9db62f3334a1fad706f97c5a4ebb8485668447c176e2...	JE299347	
533609	db31327d28803316b8f44f0ec86d6e76a248934f3d1bfc...	JE295652	
533610	d51aae396db49981c7ee26ceb54dfcab3c4b06d0cc5d7d...	JE298826	
533611	6f9abc7e7f54095cef0fa17e16e6f72eb14f8d17d3b572...	JE300245	
533612	bc361965fe506c50f7899493c79b1f7d011376f753acaa...	JE300060	

533613 rows × 50 columns

5.3 Feature/Row Drop

Contents

Contents

In [56]:

```
df_crashes_clean = df_crashes_clean.drop(columns=["RD_NO",
                                                  "STREET_DIRECT
                                                  "WORK_ZONE_I",
                                                       "NOT_RIGH

df_crashes_clean
```

Out[56]:

	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE	DEVICE_CONDITIC
0	35	NO CONTROLS	NO CONTROI
1	35	STOP SIGN/FLASHER	FUNCTIONIN PROPERI
2	30	TRAFFIC SIGNAL	FUNCTIONIN PROPERI
3	30	NO CONTROLS	NO CONTROI
4	20	NO CONTROLS	NO CONTROI
...	...	...	
533608	10	NO CONTROLS	NO CONTROI
533609	30	TRAFFIC SIGNAL	FUNCTIONIN PROPERI
533610	30	NO CONTROLS	NO CONTROI
533611	15	NO CONTROLS	NO CONTROI
533612	30	NO CONTROLS	NO CONTROI

533613 rows × 28 columns

5.4 Feature Selection

Contents

In [57]:

```
df_crashes_clean = drop_quasi_const(df_crashes_clean)
df_crashes_clean
```

Out[57]:

	POSTED_SPEED_LIMIT	TRAFFIC_CONTROL_DEVICE	DEVICE_CONDITIC
0	35	NO CONTROLS	NO CONTROI
1	35	STOP SIGN/FLASHER	FUNCTIONIN PROPERI
2	30	TRAFFIC SIGNAL	FUNCTIONIN PROPERI
3	30	NO CONTROLS	NO CONTROI
4	20	NO CONTROLS	NO CONTROI
...	...	...	
533608	10	NO CONTROLS	NO CONTROI
533609	30	TRAFFIC SIGNAL	FUNCTIONIN PROPERI
533610	30	NO CONTROLS	NO CONTROI
533611	15	NO CONTROLS	NO CONTROI
533612	30	NO CONTROLS	NO CONTROI

533613 rows × 28 columns

In [58]:

```
df_crashes_clean = rows_to_drop(df_crashes_clean,
                                y="PRIM_CONTRIBUTORY_CAUSE
```

In [59]:

```
df_crashes_clean = rows_to_drop(df_crashes_clean,
                                y="SEC_CONTRIBUTORY_CAUSE"
```

In [60]:

```
df_crashes_clean = rows_to_drop_unknown(df_crashes_clean,
```

5.5 Feature Engineering

5.5.1 SEVERELY_INJURED

In [61]:

```
df_crashes_clean["SEVERELY_INJURED"] = df_crashes_clean["I
df_crashes_clean["SEVERELY_INJURED"].value_counts()
```

Out[61]:

```
False    112252
True       6033
Name: SEVERELY_INJURED, dtype: int64
```

5.5.2 DEAD

In [62]:

```
df_crashes_clean["DEAD"] = df_crashes_clean["INJURIES_TOTA
df_crashes_clean["DEAD"].value_counts()
```

Out[62]:

```
False    117949
True        336
Name: DEAD, dtype: int64
```

5.6 Train-test Split

In [63]:

```
# Create train-test split
X = df_crashes_clean.drop(columns="CRASH_TYPE")
y = df_crashes_clean["CRASH_TYPE"]

X_train, X_test, y_train, y_test = train_test_split(X, y,
```

In [64]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[64]:

```
((82799, 29), (35486, 29), (82799,), (35486,))
```

In [65]:

```
X_train_tf = X_train.copy()
X_test_tf = X_test.copy()
```

In this section I will create new features which will improve the ability to gain insi
modeling.

Contents

6 Data Exploration

In [66]:

```
# create data exploration df
df_explore = pd.concat([X_train_tf, X_test_tf], axis=0)
df_explore["CRASH_TYPE"] = pd.concat([y_train, y_test], ax
```

In [67]:

```
df_explore
```

Out[67]:

	LIGHTING_CONDITION	SEC_CONTRIBUTORY_CAUSE	PRIM_CONTRIBU
107521	DAYLIGHT	WEATHER	FAILING TO YIE
83409	DARKNESS, LIGHTED ROAD	DISTRACTION - FROM INSIDE VEHICLE	EXCEEDING
61500	DAYLIGHT	FOLLOWING TOO CLOSELY	DISTRA OUT
10517	DAYLIGHT	FAILING TO REDUCE SPEED TO AVOID CRASH	IMPROPER
82852	DAYLIGHT	OPERATING VEHICLE IN ERRATIC, RECKLESS,	FOLLOWING

In [68]:

```
#Make a copy for linear assumptions check
df_explore_binary = df_explore.copy()
```

In [69]:

```
#Convert DEAD to 0 or 1
# df_explore["DEAD"] = df_explore["DEAD"].map({0:False, 1:
```

In [70]:

```
df_explore["CRASH_TYPE"].value_counts()
```

Out[70]:

```
DRIVE AWAY (NO INJURY)    79501
TOW BY CRASH(INJURY)      38784
Name: CRASH_TYPE, dtype: int64
```

7 Data Modeling

I will take 2 major steps in preprocessing the data for modeling:

1. Scale numerical data

Contents

2. Encode categorical data

7.1 Model Preprocessing

In [71]:

```
#training columns
X_train_tf.columns
```

Out[71]:

```
Index(['LIGHTING_CONDITION', 'SEC_CONTRIBUTORY_CAUSE',
       'PRIM_CONTRIBUTORY_CAUSE', 'POSTED_SPEED_LIMIT',
       'TRAFFIC_CONTROL_DEVICE', 'DEVICE_CONDITION', 'WEATHER_CO
       'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE', 'ROADWAY_SURFACE_C
       'ROAD_DEFECT', 'REPORT_TYPE', 'DAMAGE', 'STREET_NO',
       'BEAT_OF_OCCURRENCE', 'NUM_UNITS', 'MOST_SEVERE_INJURY',
       'INJURIES_TOTAL', 'INJURIES_FATAL', 'INJURIES_INCAPACITAT
       'INJURIES_NON_INCAPACITATING', 'INJURIES_REPORTED_NOT_EVI
       'INJURIES_NO_INDICATION', 'CRASH_HOUR', 'LATITUDE', 'LONG
       'CRASH_DATE_YR', 'SEVERELY_INJURED', 'DEAD'],
      dtype='object')
```

In [72]:

```
X_train_tf.drop(columns=["LATITUDE", "LONGITUDE"], axis=1,
X_test_tf.drop(columns=["LATITUDE", "LONGITUDE"], axis=1,
```

In [73]:

```
X_test_tf.drop("STREET_NO", axis=1, inplace=True)
X_train_tf.drop("STREET_NO", axis=1, inplace=True)
```

In [74]:

```
cat_cols = X_train_tf.select_dtypes(include="object").colu
num_cols = X_train_tf.select_dtypes(exclude="object").colu
num_cols
```

Out[74]:

```
Index(['POSTED_SPEED_LIMIT', 'NUM_UNITS', 'INJURIES_TOTAL', 'INJ
L',
       'INJURIES_INCAPACITATING', 'INJURIES_NON_INCAPACITATING',
       'INJURIES_REPORTED_NOT_EVIDENT', 'INJURIES_NO_INDICATION'
UR',
       'CRASH_DATE_YR', 'SEVERELY_INJURED', 'DEAD'],
      dtype='object')
```

In [75]:

```
1 cat_cols
```

Out[75]:

```
Index(['LIGHTING_CONDITION', 'SEC_CONTRIBUTORY_CAUSE',
       'PRIM_CONTRIBUTORY_CAUSE', 'TRAFFIC_CONTROL_DEVICE', 'DEV
ON',
       'WEATHER_CONDITION', 'FIRST_CRASH_TYPE', 'TRAFFICWAY_TYPE
       'ROADWAY_SURFACE_COND', 'ROAD_DEFECT', 'REPORT_TYPE', 'DAI
       'BEAT_OF_OCCURRENCE', 'MOST_SEVERE_INJURY'],
      dtype='object')
```

In [76]:

```
1  ohe = OneHotEncoder(sparse=False, drop="first")
2  ohe.fit(X_train_tf[cat_cols])
3  train_ohe_df = pd.DataFrame(ohe.transform(X_train_tf[cat_c
4                              columns=ohe.get_feature_names(c
5                               index=X_train_tf.index)
6
7  test_ohe_df = pd.DataFrame(ohe.transform(X_test_tf[cat_col
8                              columns=ohe.get_feature_names(c
9                              index=X_test_tf.index)
10
```

In [77]:

```
1 test_ohe_df
```

Out[77]:

	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	LIGHTING_CONDITION_DAWN	LIGH
28962	1.0	0.0	
35519	0.0	0.0	
100787	1.0	0.0	
71140	0.0	0.0	
114412	0.0	0.0	
...	...	...	
46526	0.0	0.0	
38515	1.0	0.0	
26745	0.0	0.0	
31627	0.0	0.0	
70794	0.0	0.0	

35486 rows × 445 columns

Contents

In [78]:

```
scaler = StandardScaler()
scaler.fit(X_train_tf[num_cols])

train_scale_df = pd.DataFrame(scaler.transform(X_train_tf[
                              columns=num_cols, index=X_tra

test_scale_df = pd.DataFrame(scaler.transform(X_test_tf[nu
                              columns=num_cols, index=X_tes
```

In [79]:

```
test_scale_df
```

Out[79]:

	POSTED_SPEED_LIMIT	NUM_UNITS	INJURIES_TOTAL	
28962	0.15328513149026546	-0.15195031898352457	-0.3774419405364637	-0.
35519	0.15328513149026546	-0.15195031898352457	1.0956572218977136	-0.
100787	0.15328513149026546	-0.15195031898352457	1.0956572218977136	-0.
71140	0.15328513149026546	-0.15195031898352457	-0.3774419405364637	-0.
114412	0.15328513149026546	-0.15195031898352457	-0.3774419405364637	-0.
...	...	...	...	
46526	0.15328513149026546	-2.1531169265096612	-0.3774419405364637	-0.
38515	0.15328513149026546	-0.15195031898352457	-0.3774419405364637	-0.
26745	0.15328513149026546	-0.15195031898352457	1.0956572218977136	-0.
31627	0.15328513149026546	1.849216288542612	-0.3774419405364637	-0.
70794	0.15328513149026546	-0.15195031898352457	-0.3774419405364637	-0.

35486 rows × 12 columns

In [80]:

```
1 X_train_tf = pd.concat([train_ohe_df, train_scale_df], axi
2 X_train_tf
```

Out[80]:

	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	LIGHTING_CONDITION_DAWN	LIGH
107521	0.0	0.0	
83409	1.0	0.0	
61500	0.0	0.0	
10517	0.0	0.0	
82852	0.0	0.0	
...	...	...	
50057	0.0	0.0	
98047	1.0	0.0	
5192	0.0	0.0	
77708	0.0	0.0	
98539	1.0	0.0	

82799 rows × 457 columns

Contents

In [81]:

```
X_test_tf = pd.concat([test_ohe_df, test_scale_df], axis=1
X_test_tf
```

Out[81]:

	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	LIGHTING_CONDITION_DAWN	LIGH
28962	1.0	0.0	
35519	0.0	0.0	
100787	1.0	0.0	
71140	0.0	0.0	
114412	0.0	0.0	
...	...	...	
46526	0.0	0.0	
38515	1.0	0.0	
26745	0.0	0.0	
31627	0.0	0.0	
70794	0.0	0.0	

35486 rows × 457 columns

7.2 Dummy Classifier

7.2.1 Model Creation

In [82]:

```
#Converting target variable to 0s and 1s

# 0 means no injury and 1 means injury
y_train = y_train.map({"DRIVE AWAY (NO INJURY)":0,  "TOW B
y_test = y_test.map({"DRIVE AWAY (NO INJURY)":0, "TOW BY C
```

In [83]:

```
y_train.value_counts()
```

Out[83]:

```
0    55663
1    27136
Name: CRASH_TYPE, dtype: int64
```

Contents

In [84]:

```
#create dummy classifier as a baseline
dummy = DummyClassifier()

#fit the dummy model
dummy.fit(X_train_tf, y_train)
```

Out[84]:

```
DummyClassifier()
```

In [85]:

```
y_train.value_counts(normalize=True)
```

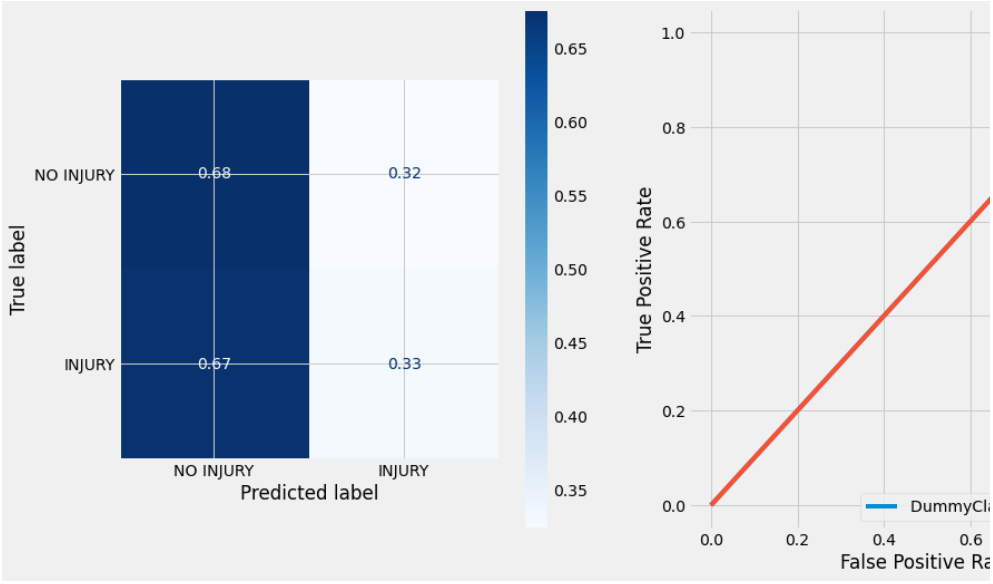
Out[85]:

```
0    0.672266573267793
1   0.3277334267322069
Name: CRASH_TYPE, dtype: float64
```

In [86]:

```
model_eval(dummy, X_train_tf, y_train, X_test_tf, y_test)
```

```
CURRENT MODEL: Not Overfit (Recall)
============================================================


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.67      0.67      0.67     23838
           1       0.33      0.33      0.33     11648

    accuracy                           0.56     35486
   macro avg       0.50      0.50      0.50     35486
weighted avg       0.56      0.56      0.56     35486

Test Graphs---------------------------------------------------
```



7.3 Logistic Regression

First I will create a logistic regression model and check for the scores.|

7.3.1 Linearity with Target

Observation

- The features seem to have a linear relationship with target

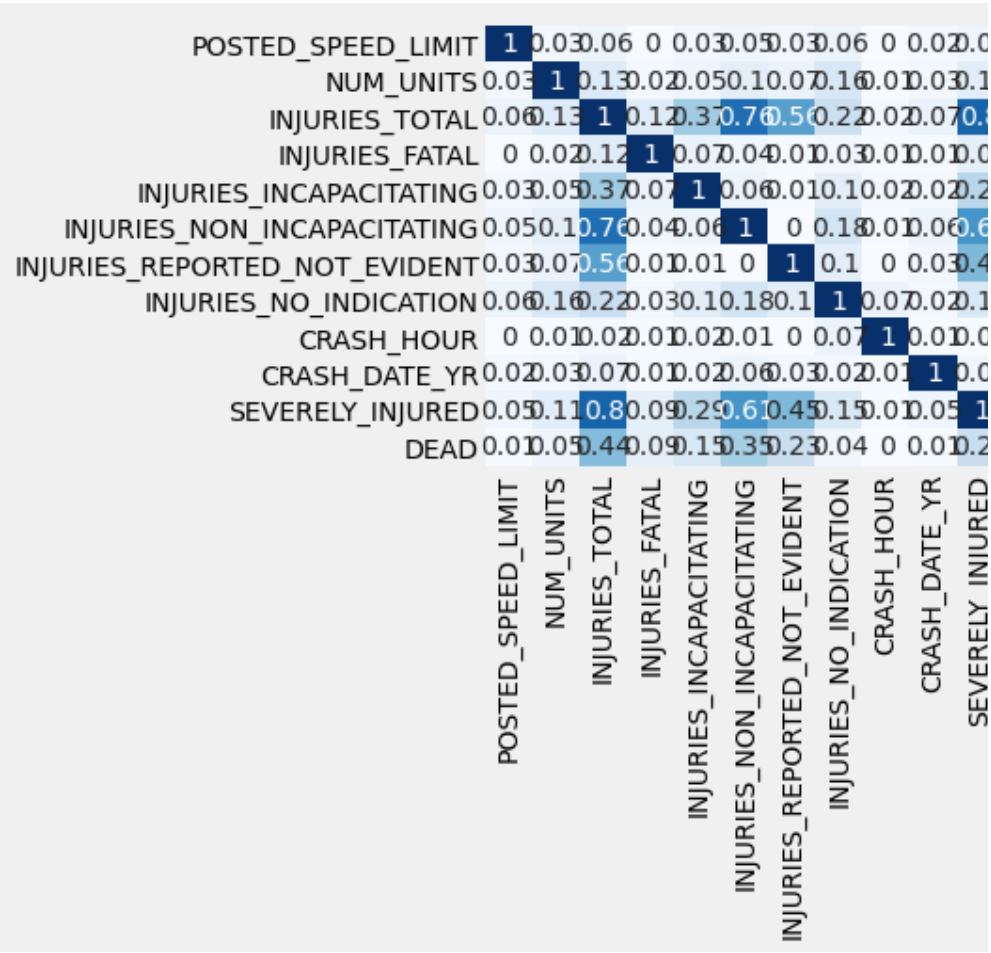
7.3.2 Multicollinearity

In [87]:

```
1 sns.heatmap(df_explore.corr().abs().round(2), annot=True,
```

Out[87]:

```
<AxesSubplot:>
```



Observation

- High correlation between `INJURIES_TOTAL` and `INJURIES_NON_I` observed.

Action

- drop `INJURIES_NON_INCAPACITATING` column.

In [88]:

```
X_train_tf.drop("INJURIES_FATAL", axis=1, inplace=True)
X_test_tf.drop("INJURIES_FATAL", axis=1, inplace=True)
```

In [89]:

```
X_train_tf.drop("INJURIES_TOTAL", axis=1, inplace=True)
X_test_tf.drop("INJURIES_TOTAL", axis=1, inplace=True)
```

7.3.3 Model 1

In [90]:

```
X_train_lr = X_train_tf.copy()
X_test_lr = X_test_tf.copy()
X_train_lr.shape, X_test_lr.shape
```

Out[90]:

```
((82799, 455), (35486, 455))
```

In [91]:

```
lr1 = LogisticRegression()
```

In [92]:

```
lr1.fit(X_train_lr, y_train)
```

Out[92]:

```
LogisticRegression()
```

In [93]:

```
y_test_pred = lr1.predict(X_test_lr)
```

In [94]:

```
lr1_score = accuracy_score(y_test, y_test_pred)
lr1_score
```

Out[94]:

```
0.891450149354675
```

In [95]:

```
1 y_train.value_counts(normalize=True)
```

Out[95]:

```
0    0.672266573267793
1   0.3277334267322069
Name: CRASH_TYPE, dtype: float64
```

7.3.4 Model evaluation

Contents

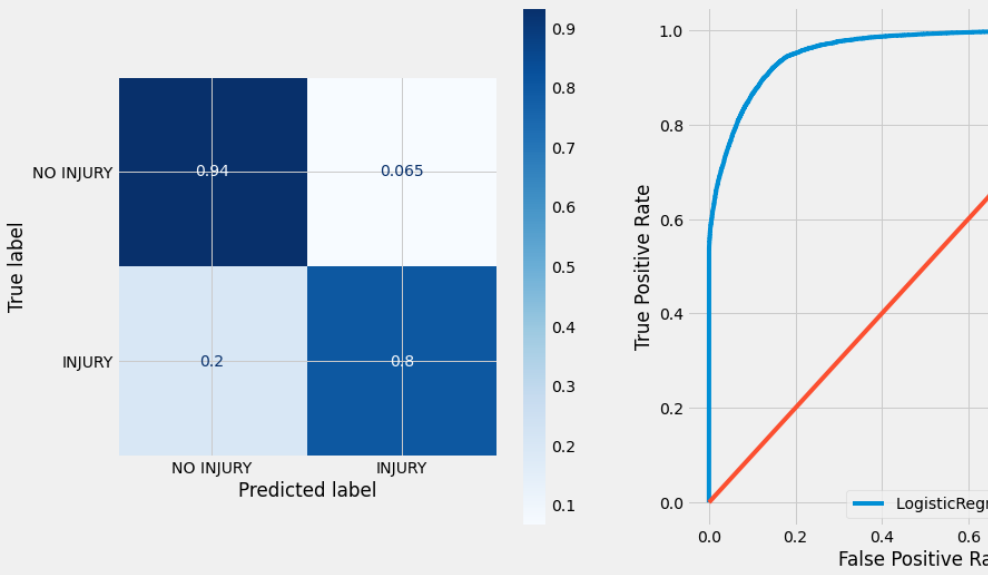
In [96]:

```
1 model_eval(lr1, X_train_lr, y_train, X_test_lr, y_test, pr
```

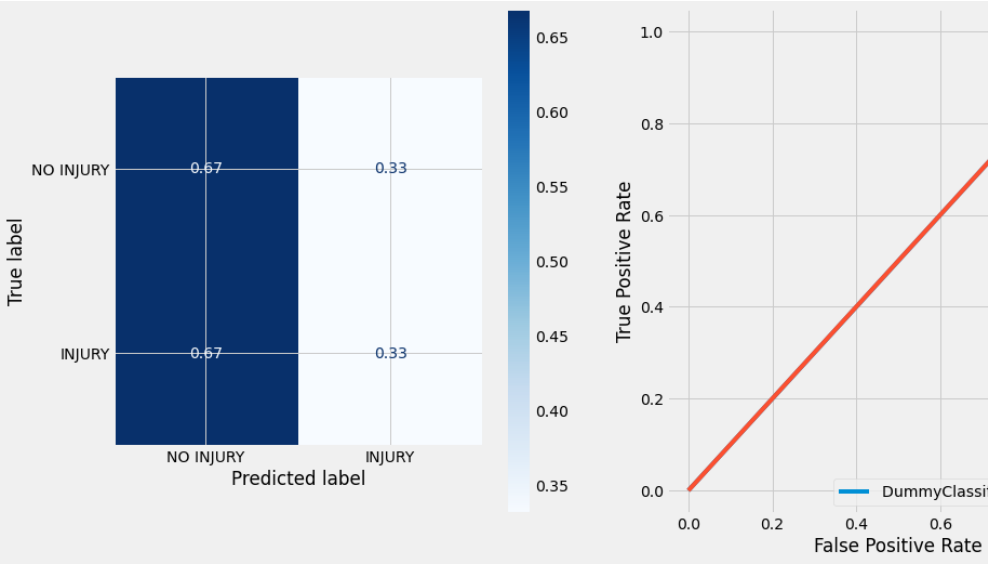
```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

	Previous Model	Current Model	Delta
Recall	0.33	0.8	0.47
F1	0.33	0.83	0.5
Accuracy	0.56	0.89	0.33
AUC	0.5	0.96	0.46

```
CURRENT MODEL: Not Overfit (Recall)
================================================================

Recall on Training: 0.81
Recall on Test: 0.8

Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.94      0.92     23838
           1       0.86      0.80      0.83     11648

    accuracy                           0.89     35486
   macro avg       0.88      0.87      0.87     35486
weighted avg       0.89      0.89      0.89     35486

Test Graphs----------------------------------------------------
```




```
PREVIOUS MODEL
================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

7.4 Feature Selection

In [97]:

```
#create copy of training and test set for logistic regress
X_train_lr_sel = X_train_lr.copy()
X_test_lr_sel = X_test_lr.copy()

X_train_lr_sel.shape, X_test_lr_sel.shape
```

Out[97]:

```
((82799, 455), (35486, 455))
```

In [98]:

```
#remove constant features
sel_const = DropConstantFeatures(tol=1, missing_values='ra
#fit the model
sel_const.fit(X_train_lr_sel)
```

Out[98]:

```
DropConstantFeatures()
```

In [99]:

```
len(sel_const.features_to_drop_)
```

Out[99]:

```
0
```

Contents

In [100]:

```
#remove quasi-constant features
quasi_const = DropConstantFeatures(tol=0.998, missing_valu
#fit the model
quasi_const.fit(X_train_lr_sel)
```

Out[100]:

```
DropConstantFeatures(tol=0.998)
```

In [101]:

```
len(quasi_const.features_to_drop_)
```

Out[101]:

```
95
```

In [102]:

```
X_train_lr_sel.shape, X_test_lr_sel.shape
```

Out[102]:

```
((82799, 455), (35486, 455))
```

In [103]:

```
X_train_lr_sel = quasi_const.transform(X_train_lr)
X_test_lr_sel = quasi_const.transform(X_test_lr)
```

In [104]:

```
X_train_lr_sel.shape, X_test_lr_sel.shape
```

Out[104]:

```
((82799, 360), (35486, 360))
```

In [105]:

```
dup = DropDuplicateFeatures(missing_values="raise")
dup.fit(X_train_lr_sel)
```

Out[105]:

```
DropDuplicateFeatures(missing_values='raise')
```

In [106]:

```
dup.duplicated_feature_sets_
```

Out[106]:

```
[]
```

7.4.1 Model 2

In [107]:

```
lr_2 = LogisticRegression()
lr_2.fit(X_train_lr_sel, y_train)
```

Out[107]:

```
LogisticRegression()
```

7.4.2 Model Evaluation

Contents

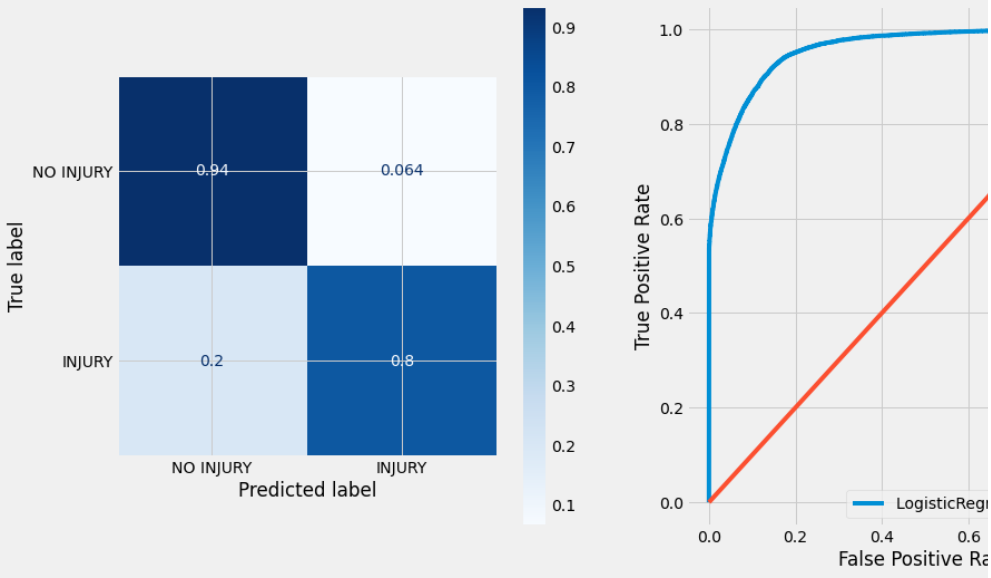
In [108]:

```
model_eval(lr_2, X_train_lr_sel, y_train, X_test_lr_sel, y
               prev_model=lr1, prev_X_train=X_train_lr, pr
               prev_X_test=X_test_lr, prev_y_test=y_test)
```

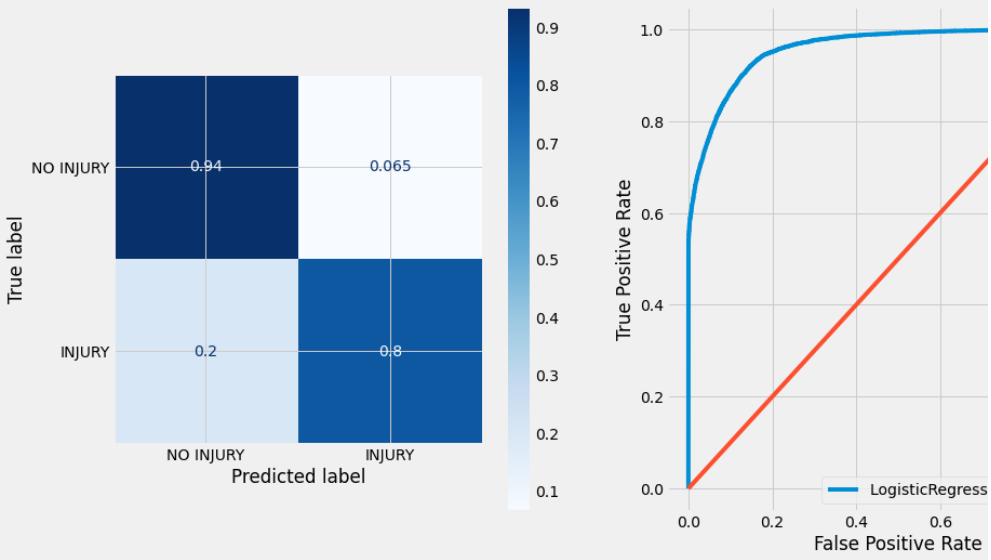
```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

	Previous Model	Current Model	Delta
Recall	0.8	0.8	0.0
F1	0.83	0.83	0.0
Accuracy	0.89	0.89	0.0
AUC	0.96	0.96	0.0

```
CURRENT MODEL: Not Overfit (Recall)
================================================================

Recall on Training: 0.81
Recall on Test: 0.8

Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.94      0.92     23838
           1       0.86      0.80      0.83     11648

    accuracy                           0.89     35486
   macro avg       0.88      0.87      0.87     35486
weighted avg       0.89      0.89      0.89     35486

Test Graphs---------------------------------------------------
```




```
PREVIOUS MODEL
================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

> **Observation**
>
> - Reducing the number of features didnt hurt the model.
> - About 21% of the features were removed.

In [109]:

```
X_train_lr_sel.to_csv("crash_train.csv")
X_test_lr_sel.to_csv("crash_test.csv")
```

In [134]:

```
y_train.to_csv("crash_y_train.csv")
y_test.to_csv("crash_y_test.csv")
```

7.4.3 Model 3

In [110]:

```
# lr_3 = LogisticRegression()
```

In [111]:

```
# lr_3_gridsearch = GridSearchCV(lr_3, params_3,
#                                scoring="recall", n_jobs=
# lr_3_gridsearch.fit(X_train_lr_sel, y_train)
```

7.4.4 Model Evaluation

In [112]:

```
# lr_3_gridsearch.best_params_
```

Contents

In [113]:

```
# model_eval(lr_3_gridsearch.best_estimator_, X_train_lr_s
```

7.4.5 Model 4

In [114]:

```
# lr_4 = LogisticRegression(class_weight="balanced")
```

In [115]:

```
# params_4 = {"C":[.01, 0.1, 1, 1e7, 1e8, 1e9, 1e10],
#             "penalty":["l1", "l2"],
#             "solver":["liblinear"]}
```

In [116]:

```
# lr_4_gridsearch = GridSearchCV(lr_4, params_4, scoring="
# lr_4_gridsearch.fit(X_train_lr_sel, y_train)
```

In [117]:

```
# lr_4_gridsearch.best_params_
```

7.4.6 Model Evaluation

In [118]:

```
# model_eval(lr_4_gridsearch.best_estimator_, X_train_lr_s
#           X_test_lr_sel,y_test,
#            prev_model=lr_3_gridsearch.best_estimator_ )
```

7.5 Random Forest

7.5.1 Model 1

In [119]:

```
# X_train_dt
```

In [120]:

```
# rf_1 = RandomForestClassifier()
# rf_1.fit(X_train_dt, y_train)
```

In [121]:

```
# X_test_dt = X_test_tf.copy()
```

Contents

7.5.2 Model Evaluation

In [122]:

```
# model_eval(rf_1, X_train_dt, y_train, X_test_dt, y_test,
#             prev_model=lr_4_gridsearch.best_estimator_,
#            prev_X_train=X_train_lr_sel,
#            prev_y_train=y_train,
#            prev_X_test=X_test_lr_sel, prev_y_test=y_tes
```

In [123]:

```
# rf_1.estimators_[0].get_n_leaves
```

In [124]:

```
# depths = [m.get_depth() for m in rf_1.estimators_]
# max(depths)
```

7.5.3 Model 2

In [125]:

```
# rf_2 = RandomForestClassifier()
```

In [126]:

```
# params = [{"n_estimators":[100, 500, 1000],
#            "criterion":["entropy", "gini"],
#            "max_features":["auto", "log2"],
#            "class_weight":["balanced"],
#            "max_depth":[20, 35, 65],
#            "min_samples_split":[2, 3],
#            "min_samples_leaf":[1, 2, 3]}]
```

In [127]:

```
# rf_2_gridsearch = GridSearchCV(rf_2, params, scoring="re
# rf_2_gridsearch.fit(X_train_dt, y_train)
```

7.5.4 Model Evaluation

In [128]:

```
# rf_2_gridsearch.best_params_
```

In [129]:

```
# model_eval(rf_2_gridsearch.best_estimator_, X_train_dt,
#            prev_model=rf_1)
```

Contents

In [130]:

```
# rf_2_fi = rf_2_gridsearch.best_estimator_.feature import
```

In [131]:

```
# rf_2_features = pd.DataFrame(data=rf_2_fi, index=X_train_
#                              columns=["Feature Importance
# rf_2_features_sprted = rf_2_features[:10].sort_values("F
```

In [132]:

```
# rf_2_features_sorted
```

In [133]:

```
# fig, ax= plt.subplots(figasize=(10, 10))
# sns.barplot(data=rf_2_features_sorted,
#             x=rf_2_features_sorted["Feature Importances"]
#             y = rf_2_features_sorted.index, orient="h", c
```

Data Interpretation

>This dataset took a lot of cleaning, especially involving u
in the original dataset. After cleaning, a logistic regressi
classification was trained to predict civilians in Chicago b
Classification models were created and optimized for finding
to car crash.

Recommendations and Conclusions

> I have 3 recommendations based on my work:
1.

In [ ]: