Aim

1. Make featurized data model ready
2. flatten array features e.g kw_vec (keyword vectors)

In [1]:
```
import os

import numpy as np
import pandas as pd
```

In [2]:
```
DATA_ROOT = f"../data"
```

In [3]:
```
df_train = pd.read_pickle(f"{DATA_ROOT}/train/featurized/data.pkl")
df_test = pd.read_pickle(f"{DATA_ROOT}/test/featurized/data.pkl")
```

In [8]:
```
df_train.iloc[0]
```

Out[8]:
```
ID                                                  A-2478859
Source                                                 [0, 1]
TMC                                                         0
Distance(mi)                                             3.23
Side                                                   [0, 1]
City                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
County                       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
State                                  [0, 0, 0, 0, 0, 1]
Timezone                                         [0, 0, 1]
Airport_Code                 [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
Temperature(F)                                        42.1
Humidity(%)                                             58
Pressure(in)                                         29.76
Visibility(mi)                                          10
Wind_Direction                                 [0, 0, 0, 0, 1]
Wind_Speed(mph)                                       10.4
Weather_Condition                        [0, 0, 0, 0, 0, 0, 1]
Amenity                                              False
Bump                                                 False
Crossing                                             False
Give_Way                                             False
Junction                                             False
No_Exit                                              False
Railway                                              False
Roundabout                                           False
Station                                              False
Stop                                                 False
Traffic_Calming                                      False
Traffic_Signal                                       False
Turning_Loop                                         False
Sunrise_Sunset                                       [0, 1]
Civil_Twilight                                       [0, 1]
Nautical_Twilight                                    [0, 1]
Astronomical_Twilight                                [0, 1]
kw_vec                  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
zip_02                                                 [1]
zip_25                                                 [1]
zip_len                                                  5
zip_is_compound                                      False
Severity                                                 3
Name: 0, dtype: object
```

In [13]:
```
categorical_features_as_vector_columns = [
    # transformed
    "Source",
    "Side",
    "City",
    "County",
    "State",
    "Timezone",
    "Airport_Code",
    "Wind_Direction",
    "Weather_Condition",
    "Sunrise_Sunset",
    "Civil_Twilight",
    "Nautical_Twilight",
    "Astronomical_Twilight",
    # engineered and transformed
    "kw_vec",
    "zip_02",
    "zip_25",
]
```

In [20]:
```
def prepare_vector_columns_for_model(df, vector_column):

    if vector_column == "kw_vec":
        kw_list = [
            "kw_" + i[0] for i in pd.read_pickle(f"{DATA_ROOT}/train/keyword
        ]

        df_vec = pd.DataFrame(df[vector_column].tolist(), columns=kw_list)

    else:
        cols = [f"{vector_column}_{i}" for i in range(len(df[vector_column].
        df_vec = pd.DataFrame(df[vector_column].tolist(), columns=cols)

    df_final = pd.concat([df_vec, df], axis="columns").drop(columns=[vector_
    return df_final


for c in categorical_features_as_vector_columns:
    df_train = prepare_vector_columns_for_model(df_train, vector_column=c)
    df_test = prepare_vector_columns_for_model(df_test, vector_column=c)
```

In [24]:
```
df_train.columns.tolist()
```

Out[24]:
```
['zip_25_0',
 'zip_02_0',
 'kw_Accident',
 'kw_Northbound',
 'kw_Hwy',
 'kw_ramp',
 'kw_slow',
 'kw_Trl',
 'kw_Mopac',
 'kw_Okeechobee',
 'kw_Brookshire',
 'kw_Huntington',
 'kw_NYS',
 'kw_Fuqua',
 'kw_Middlefield',
 'kw_JFK',
 'kw_Cedarhurst',
 'Astronomical_Twilight_0',
 'Astronomical_Twilight_1',
 'Nautical_Twilight_0',
 'Nautical_Twilight_1',
 'Civil_Twilight_0',
 'Civil_Twilight_1',
 'Sunrise_Sunset_0',
 'Sunrise_Sunset_1',
 'Weather_Condition_0',
 'Weather_Condition_1',
 'Weather_Condition_2',
 'Weather_Condition_3',
 'Weather_Condition_4',
 'Weather_Condition_5',
 'Weather_Condition_6',
 'Wind_Direction_0',
 'Wind_Direction_1',
 'Wind_Direction_2',
 'Wind_Direction_3',
 'Wind_Direction_4',
 'Airport_Code_0',
 'Airport_Code_1',
 'Airport_Code_2',
 'Airport_Code_3',
 'Airport_Code_4',
 'Airport_Code_5',
 'Airport_Code_6',
 'Airport_Code_7',
 'Airport_Code_8',
 'Airport_Code_9',
 'Airport_Code_10',
 'Timezone_0',
 'Timezone_1',
 'Timezone_2',
 'State_0',
 'State_1',
 'State_2',
 'State_3',
 'State_4',
 'State_5',
 'County_0',
 'County_1',
 'County_2',
 'County_3',
 'County_4',
 'County_5',
 'County_6',
```

```
'County_7',
'County_8',
'County_9',
'County_10',
'City_0',
'City_1',
'City_2',
'City_3',
'City_4',
'City_5',
'City_6',
'City_7',
'City_8',
'City_9',
'City_10',
'City_11',
'City_12',
'City_13',
'Side_0',
'Side_1',
'Source_0',
'Source_1',
'ID',
'TMC',
'Distance(mi)',
'Temperature(F)',
'Humidity(%)',
'Pressure(in)',
'Visibility(mi)',
'Wind_Speed(mph)',
'Amenity',
'Bump',
'Crossing',
'Give_Way',
'Junction',
'No_Exit',
'Railway',
'Roundabout',
'Station',
'Stop',
'Traffic_Calming',
'Traffic_Signal',
'Turning_Loop',
'zip_len',
'zip_is_compound',
'Severity']
```

In [25]:
```
df_train.drop(columns=["ID"], inplace=True, errors="ignore")
df_test.drop(columns=["ID"], inplace=True, errors="ignore")
```

As we are preparing data for tree based models, there is no need to scale/ standardize the data.

In [26]:
```
os.makedirs(f"{DATA_ROOT}/train/model/", exist_ok=True)
os.makedirs(f"{DATA_ROOT}/test/model/", exist_ok=True)
```

In [30]:
```
df_train.isna().any().sum()  # check nans any
```

Out[30]:
```
0
```

In [31]:
```
df_test.isna().any().sum()  # check nans any
```

Out[31]: 0

In [28]:
```
df_train.to_pickle(f"{DATA_ROOT}/train/model/data.pkl")
df_test.to_pickle(f"{DATA_ROOT}/test/model/data.pkl")
```