Car Accident Severity Report

Applied Data Science Capstone - IBM/Coursera

Table of contents

Introduction | Business Understanding

- Road traffic injuries are currently estimated to be the eighth leading cause of death across all age groups globally, and are predicted to become the seventh leading cause of death by 2030.
- Analysing a significant range of factors, including weather conditions, special events, roadworks, traffic jams among others, an accurate prediction of the severity of the accidents can be performed.
- In an effort to reduce the frequency of car collisions in a community, an algorithm must be developed to predict the severity of an accident given the current weather, road and visibility conditions. In an application, drivers will be alerted of the severity level when conditions are above code 0.

In [10]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Data

In [2]:

```
df =pd.read_csv('Data-Collisions.csv')
df.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3058: DtypeWarning:
Columns (33) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADC
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	

5 rows × 38 columns

Check data using describe()

In [3]:

```
df.describe()
```

Out[3]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	INTKEY	SEVERITYCODE.1
count	194673.000000	189339.000000	189339.000000	194673.000000	194673.000000	194673.000000	65070.000000	194673.000000
mean	1.298901	-122.330518	47.619543	108479.364930	141091.456350	141298.811381	37558.450576	1.298901
std	0.457778	0.029976	0.056157	62649.722558	86634.402737	86986.542110	51745.990273	0.457778
min	1.000000	-122.419091	47.495573	1.000000	1001.000000	1001.000000	23807.000000	1.000000
25%	1.000000	-122.348673	47.575956	54267.000000	70383.000000	70383.000000	28667.000000	1.000000
50%	1.000000	-122.330224	47.615369	106912.000000	123363.000000	123363.000000	29973.000000	1.000000
75%	2.000000	-122.311937	47.663664	162272.000000	203319.000000	203459.000000	33973.000000	2.000000
max	2.000000	-122.238949	47.734142	219547.000000	331454.000000	332954.000000	757580.000000	2.000000

In [4]:

```
df.shape
```

Out[4]:

```
(194673, 38)
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['SEVERITYCODE', 'X', 'Y', 'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPORTNO',
       'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
       'EXCEPTRSNDESC', 'SEVERITYCODE.1', 'SEVERITYDESC', 'COLLISIONTYPE',
       'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT', 'INCDATE',
       'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'SDOT_COLDESC',
       'INATTENTIONIND', 'UNDERINFL', 'WEATHER', 'ROADCOND', 'LIGHTCOND',
       'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'ST_COLCODE', 'ST_COLDESC',
       'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'],
      dtype='object')
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194673 entries, 0 to 194672
Data columns (total 38 columns):
SEVERITYCODE      194673 non-null int64
X                 189339 non-null float64
Y                 189339 non-null float64
OBJECTID          194673 non-null int64
INCKEY            194673 non-null int64
COLDETKEY         194673 non-null int64
REPORTNO          194673 non-null object
STATUS            194673 non-null object
ADDRTYPE          192747 non-null object
INTKEY            65070 non-null float64
LOCATION          191996 non-null object
EXCEPTRSNCODE     84811 non-null object
EXCEPTRSNDESC     5638 non-null object
SEVERITYCODE.1    194673 non-null int64
```

```
SEVERITYDESC      194673 non-null object
COLLISIONTYPE     189769 non-null object
PERSONCOUNT       194673 non-null int64
PEDCOUNT          194673 non-null int64
PEDCYLCOUNT       194673 non-null int64
VEHCOUNT          194673 non-null int64
INCDATE           194673 non-null object
INCDTTM           194673 non-null object
JUNCTIONTYPE      188344 non-null object
SDOT_COLCODE      194673 non-null int64
SDOT_COLDESC      194673 non-null object
INATTENTIONIND    29805 non-null object
UNDERINFL         189789 non-null object
WEATHER           189592 non-null object
ROADCOND          189661 non-null object
LIGHTCOND         189503 non-null object
PEDROWNOTGRNT     4667 non-null object
SDOTCOLNUM        114936 non-null float64
SPEEDING          9333 non-null object
ST_COLCODE        194655 non-null object
ST_COLDESC        189769 non-null object
SEGLANEKEY        194673 non-null int64
CROSSWALKKEY      194673 non-null int64
HITPARKEDCAR      194673 non-null object
dtypes: float64(4), int64(12), object(22)
memory usage: 56.4+ MB
```

Data Understanding

Our predictor or target variable will be 'SEVERITYCODE' because it is used to measure the severity of an accident from 0 to 4 within the dataset. Attributes used to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.

Severity codes are as follows:

- 0 : Little to no Probability (Clear Conditions)
- 1 : Very Low Probability - Chance or Property Damage
- 2 : Low Probability - Chance of Injury
- 3 : Mild Probability - Chance of Serious Injury
- 4 : High Probability - Chance of Fatality

Extract Dataset & Convert

From the summary of the data we see that the data types are coherent with their respective values, with the only exception of the date, and that some features have missing values.

- More than half of the values for the coordenates are missig, as well as roughly a 10% of the data regarding the road_num and more than a 50% of the remaining samples are a 0. Thus, to keep the amount of samples the mentioned features will be dropped.
- Few values are missing in some features such as the atmospheric conditions or road category.

Missing values and outliers will be filled with the label for *Other cases* category if possible. If not the most frequent value of the feature will be applyed.

In [20]:

```
# Drop all columns with no predictive value for the context of this project
colData = df.drop(columns = ['OBJECTID', 'SEVERITYCODE.1', 'REPORTNO', 'INCKEY', 'COLDETKEY',
              'X', 'Y', 'STATUS','ADDRTYPE',
              'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
              'EXCEPTRSNDESC', 'SEVERITYDESC', 'INCDATE',
              'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE',
              'SDOT_COLDESC', 'PEDROWNOTGRNT', 'SDOTCOLNUM',
              'ST_COLCODE', 'ST_COLDESC', 'SEGLANEKEY',
              'CROSSWALKKEY', 'HITPARKEDCAR', 'PEDCOUNT', 'PEDCYLCOUNT',
              'PERSONCOUNT', 'VEHCOUNT', 'COLLISIONTYPE',
              'SPEEDING', 'UNDERINFL', 'INATTENTIONIND'])

# Label Encoding
```

```
# Label Encoding
# Convert column to category
colData["WEATHER"] = colData["WEATHER"].astype('category')
colData["ROADCOND"] = colData["ROADCOND"].astype('category')
colData["LIGHTCOND"] = colData["LIGHTCOND"].astype('category')

# Assign variable to new column for analysis
colData["WEATHER_CAT"] = colData["WEATHER"].cat.codes
colData["ROADCOND_CAT"] = colData["ROADCOND"].cat.codes
colData["LIGHTCOND_CAT"] = colData["LIGHTCOND"].cat.codes

colData.head(5)
```

Out[20]:

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	WEATHER_CAT	ROADCOND_CAT	LIGHTCOND_CAT
0	2	Overcast	Wet	Daylight	4	8	5
1	1	Raining	Wet	Dark - Street Lights On	6	8	2
2	1	Overcast	Dry	Daylight	4	0	5
3	1	Clear	Dry	Daylight	1	0	5
4	2	Raining	Wet	Daylight	6	8	5

In [21]:

```
colData.dtypes
```

Out[21]:

```
SEVERITYCODE        int64
WEATHER          category
ROADCOND         category
LIGHTCOND        category
WEATHER_CAT          int8
ROADCOND_CAT         int8
LIGHTCOND_CAT        int8
dtype: object
```

In [22]:

```
colData["SEVERITYCODE"].value_counts()
```

Out[22]:

```
1    136485
2     58188
Name: SEVERITYCODE, dtype: int64
```

In [28]:

```
colData["WEATHER"].value_counts()
```

Out[28]:

```
Clear                       111135
Raining                      33145
Overcast                     27714
Unknown                      15091
Snowing                        907
Other                          832
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               56
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

In [29]:

```
colData["ROADCOND"].value_counts()
```

Out[29]:

```
Dry               124510
Wet                47474
Unknown            15078
Ice                 1209
Snow/Slush          1004
Other                132
Standing Water       115
Sand/Mud/Dirt         75
Oil                   64
Name: ROADCOND, dtype: int64
```

In [30]:

```
colData["LIGHTCOND"].value_counts()
```

Out[30]:

```
Daylight                    116137
Dark - Street Lights On      48507
Unknown                      13473
Dusk                          5902
Dawn                          2502
Dark - No Street Lights       1537
Dark - Street Lights Off      1199
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

Balancing the Dataset

Our target variable SEVERITYCODE is only 42% balanced. In fact, severitycode in class 1 is nearly three times the size of class 2.

We can fix this by downsampling the majority class.

Down-sample Majority Class

Down-sampling involves randomly removing observations from the majority class to prevent its signal from dominating the learning algorithm.

The most common heuristic for doing so is resampling without replacement.

The process is similar to that of up-sampling. Here are the steps:

- 1.First, we'll separate observations from each class into different DataFrames.
- 2.Next, we'll resample the majority class without replacement, setting the number of samples to match that of the minority class.
- 3.Finally, we'll combine the down-sampled majority class DataFrame with the original minority class DataFrame.

In [33]:

```
from sklearn.utils import resample
```

In [34]:

```
# Seperate majority and minority classes
colData_majority = colData[colData.SEVERITYCODE==1]
colData_minority = colData[colData.SEVERITYCODE==2]

#Downsample majority class
colData_majority_downsampled = resample(colData_majority,
                                        replace=False,
                                        n_samples=58188,
                                        random_state=123)

# Combine minority class with downsampled majority class
colData_balanced = pd.concat([colData_majority_downsampled, colData_minority])
```

```
# Display new class counts
colData_balanced.SEVERITYCODE.value_counts()
```

Out[34]:

```
2    58188
1    58188
Name: SEVERITYCODE, dtype: int64
```

In [74]:

```
colData_balanced.shape
```

Out[74]:

```
(116376, 7)
```