Car Accident Severity Prediction

Introduction/Business Problem

Cars transformed modern society to what it is now and literally sped up economic activities – bringing in financial growth and opportunities to the masses. But all of this came at a cost. Road traffic injuries are estimated to be the eighth leading cause of death globally for all age groups and the leading cause of death for children and young people 5–29 years of age. More people now die in road traffic crashes than from HIV/AIDS [1]. The ability to predict these unfortunate events will help transport authorities to implement policies and laws based on factors that are directly impact the safety of car drivers.

Stakeholders: Metropolitan Transport Authorities and Government Officials

Data Description

The original dataset had irrelevant attributes, so we decided to cut down to the following features:

- SEVERITYCODE - A code that corresponds to the severity of the collision
- ROADCOND - The condition of the road during the collision
- WEATHER - A description of the weather conditions during the time of the collision
- LIGHTCOND - The light conditions during the collision
- ADDRTYPE - Collision address type (Alley, Block, Intersection)

The dataset to be used was then extracted as follows:

```
[10] df = df[['SEVERITYCODE', 'ROADCOND', 'WEATHER', 'LIGHTCOND', 'ADDRTYPE']]
     df
```

	SEVERITYCODE	ROADCOND	WEATHER	LIGHTCOND	ADDRTYPE
0	2	Wet	Overcast	Daylight	Intersection
1	1	Wet	Raining	Dark - Street Lights On	Block
2	1	Dry	Overcast	Daylight	Block
3	1	Dry	Clear	Daylight	Block
4	2	Wet	Raining	Daylight	Intersection
...	...	...	...	...	...
194668	2	Dry	Clear	Daylight	Block
194669	1	Wet	Raining	Daylight	Block
194670	2	Dry	Clear	Daylight	Intersection
194671	2	Dry	Clear	Dusk	Intersection
194672	1	Wet	Clear	Daylight	Block

194673 rows × 5 columns

We then converted the columns into categorical data types

```
[14] df["ROADCOND"] = df["ROADCOND"].astype('category')
     df["WEATHER"] = df["WEATHER"].astype('category')
     df["LIGHTCOND"] = df["LIGHTCOND"].astype('category')
     df["ADDRTYPE"] = df["ADDRTYPE"].astype('category')

     df["ROADCOND_CAT"] = df["ROADCOND"].cat.codes
     df["WEATHER_CAT"] = df["WEATHER"].cat.codes
     df["LIGHTCOND_CAT"] = df["LIGHTCOND"].cat.codes
     df["ADDRTYPE_CAT"] = df["ADDRTYPE"].cat.codes

     df
```

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

One more issue remains – the SEVERITYCODE column is not balanced i.e. the number of entries is not equally distributed across the available classes. This was fixed as shown below

```
[23] from sklearn.utils import resample
     # Seperate majority and minority classes
     df_majority = df[df.SEVERITYCODE==1]
     df_minority = df[df.SEVERITYCODE==2]

     #Downsample majority class
     df_majority_downsampled = resample(df_majority,
                                        replace=False,
                                        n_samples=58188,
                                        random_state=123)

     # Combine minority class with downsampled majority class
     df_balanced = pd.concat([df_majority_downsampled, df_minority])

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

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

Now our dataset is ready for modelling.

Methodology

The training variables:

```
import numpy as np
X = np.asarray(df_balanced[['WEATHER_CAT', 'ROADCOND_CAT', 'LIGHTCOND_CAT', 'ADDRTYPE_CAT']])
X[0:5]
```

```
array([[ 6,  8,  2,  2],
       [ 1,  0,  5,  2],
       [10,  7,  8,  2],
       [ 1,  0,  5,  1],
       [ 1,  0,  5,  1]], dtype=int8)
```

The target variable:

```
[ ] y = np.asarray(df_balanced['SEVERITYCODE'])
    y [0:5]
```

```
array([1, 1, 1, 1, 1])
```

Dataset Normalization

```
[ ] from sklearn import preprocessing
    X = preprocessing.StandardScaler().fit(X).transform(X)
    X[0:5]
```

```
array([[ 1.15236718,  1.52797946, -1.21648407,  1.20776885],
       [-0.67488   , -0.67084969,  0.42978835,  1.20776885],
       [ 2.61416492,  1.25312582,  2.07606076,  1.20776885],
       [-0.67488   , -0.67084969,  0.42978835, -0.66998231],
       [-0.67488   , -0.67084969,  0.42978835, -0.66998231]])
```

Training/Test Split

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4)
    print ('Train set:', X_train.shape,  y_train.shape)
    print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (81463, 4) (81463,)
Test set: (34913, 4) (34913,)
```

Three models were trained, namely:

- K-Nearest Neighbours
- Logistic Regression
- Decision Trees

Results

The following results were obtained after training all three models:

	Jaccard Similarity Score	**F1-SCORE**
K-Nearest Neighbours	0.59	0.58
Logistic Regression	0.599	0.59
Decision Trees	0.60	0.598

Discussion

From the results we can see that the Logistic Regression model performed better than the K-Nearest Neighbours and Decision Tree models. From the results obtained it is clear that the selected training attributes have some correlation with the target variable (Severity) but this correlation is not strong enough as the models in overall were not of very high accuracy. This maybe improved by selecting better training attributes that maybe be stronger predictors of the target variable.

Conclusion

The trained models were of decent accuracy, but for future work this level of accuracy can be improved as it's not near the desired level of accuracy.

References

[1] World Health Organization (WHO). Global Status Report on Road Safety 2018. December 2018. [cited 2019 April 8].Available from URL:https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/external icon