Capstone Project : Car Accident Severity

Applied Data Science Capstone by IBM/Coursera

Introduction: Business Problem

Average number of car accidents in the U.S. every year is 6 million.

3 million people in the U.S. are injured every year in car accidents.

Analysing the conditions that contribute to these accidents would lead to the prevention of significant loss of life and financial resources.

The project is aimed at predicting the severity of a car accident given the location, weather, road and visibility conditions in order to reduce the frequency of car collisions in a community based on a dataset provided by Seattle PD.

Consequently, this analysis would aid drivers to exercise more caution while driving or even choose an alternative route or time for their travel if possible.

It could also potentially help the local government, the police department and car insurance providers to gain deeper insight into road accidents.

Data

The detailed dataset of all road collisions (since 2004 to present) can be found here. This data was provided by the Seattle Police Department and recorded by the Traffic Records Department. The dataset consists of 37 independent fields and 194673 records, which includes both numerical and categorical data. The dependent field or label for the data set is SEVERITY CODE, which describes the fatality of an accident. The values under this label are categorised into fatality (3), serious injury (2b), injury (2), prop damage (1) and unknown (0).

Methodology

First, the necessary packages are imported.

```
#import necessary packages

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy import stats
from pylab import rcParams
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import jaccard_score, log_loss
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
```

Then the dataset is read from the .csv file and assigned to a DataFrame df.

DATA WRANGLING

The fields necessary for the analysis are identified and a new dataset df_acc is created with these selected fields. These fields are -
SEVERITYCODE,
ADDRTYPE,
LOCATION,
JUNCTIONTYPE,
WEATHER,
ROADCOND,
LIGHTCOND,
SPEEDING

The datatype of each of the included columns is examined and the following output is observed.

```
df_acc.dtypes
```

```
SEVERITYCODE      int64
ADDRTYPE         object
LOCATION         object
JUNCTIONTYPE     object
WEATHER          object
ROADCOND         object
LIGHTCOND        object
SPEEDING         object
dtype: object
```

Null values in the DataFrame are then identified using the following code.

```
missing_data = df_acc.isnull()
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

Subsequently, null values in the fields 'JUNCTIONTYPE', 'WEATHER', 'ROADCOND', 'LIGHTCOND', and 'SPEEDING' are replaced with suitable values.

UNDERSAMPLING

On performing a count of values on the DataFrame, it is observed that the dependent variable is an unbalanced label.

```
df_acc.value_counts("SEVERITYCODE")
```

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

Hence, undersampling is performed on df_acc to ensure that the prediction data obtained after passing the sample data through the various Machine Learning models is not biased.

```
#balance label SEVERITYCODE by undersampling

target="SEVERITYCODE"
minority_class_len = len(df_acc[df_acc[target] ==2])
majority_class_indices = df_acc[df_acc[target] ==1].index
random_majority_indices = np.random.choice(majority_class_indices,minority_class_len, replace = False)
minority_class_indices = df_acc[df_acc[target] ==2].index

under_sample_indices = np.concatenate([minority_class_indices, random_majority_indices])
df_acc = df_acc.loc[under_sample_indices]
df_acc.value_counts("SEVERITYCODE")
```

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

EXPLORATORY DATA ANALYSIS

The DataFrame is examined to obtain the following output.

```
df_acc.describe(include="all")
```

	SEVERITYCODE	ADDRTYPE	LOCATION	JUNCTIONTYPE	WEATHER	ROADCOND	LIGHTCOND	SPEEDING
count	116376.000000	115473	115136	116376	116376	116376	116376	116376
unique	NaN	3	19829	7	11	9	9	2
top	NaN	Block	AURORA AVE N BETWEEN N 117TH PL AND N 125TH ST	Mid-Block (not related to intersection)	Clear	Dry	Daylight	N
freq	NaN	71587	177	49590	68073	76182	71623	110371
mean	1.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	0.500002	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	1.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Further, the distribution of SEVERITY CODE by different columns is explored through the *value_counts*() as shown below.

SEVERITYCODE	1	2	% of Total
JUNCTIONTYPE			
Mid-Block (not related to intersection)	30186	19404	42.61
At Intersection (intersection related)	15036	27174	36.27
Mid-Block (but intersection related)	6613	7297	11.95
Driveway Junction	3126	3234	5.47
Unknown	2543	402	2.53
At Intersection (but not related to intersection)	637	623	1.08
Ramp Junction	47	54	0.09

SEVERITYCODE LOCATION	1	2	Total
AURORA AVE N BETWEEN N 117TH PL AND N 125TH ST	57	120	177
N NORTHGATE WAY BETWEEN MERIDIAN AVE N AND CORLISS AVE N	72	94	166
BATTERY ST TUNNEL SB BETWEEN AURORA AVE N AND ALASKAN WY VI SB	80	86	166
6TH AVE AND JAMES ST	57	107	164
BATTERY ST TUNNEL NB BETWEEN ALASKAN WY VI NB AND AURORA AVE N	86	78	164
AURORA AVE N BETWEEN N 130TH ST AND N 135TH ST	69	88	157
RAINIER AVE S BETWEEN S BAYVIEW ST AND S MCCLELLAN ST	62	94	156
WEST SEATTLE BR EB BETWEEN ALASKAN WY VI NB ON RP AND DELRIDGE-W SEATTLE BR EB ON RP	56	78	134
ALASKAN WY VI NB BETWEEN S ROYAL BROUGHAM WAY ON RP AND SENECA ST OFF RP	69	63	132
ALASKAN WY VI SB BETWEEN COLUMBIA ST ON RP AND ALASKAN WY VI SB EFR OFF RP	54	68	122

SEVERITYCODE WEATHER	1	2	% of Total
Clear	32233	35840	58.49
Raining	9353	11176	17.64
Overcast	8006	8745	14.39
Unknown	7780	1900	8.32
Snowing	322	171	0.42
Other	270	116	0.33
Fog/Smog/Smoke	172	187	0.31
Sleet/Hail/Freezing Rain	32	28	0.05
Blowing Sand/Dirt	15	15	0.03
Severe Crosswind	4	7	0.01
Partly Cloudy	1	3	0.00

SEVERITYCODE ROADCOND	1	2	% of Total
Dry	36118	40064	65.46
Wet	13463	15755	25.11
Unknown	7742	1809	8.21
Ice	414	273	0.59
Snow/Slush	340	167	0.44

SEVERITYCODE LIGHTCOND	1	2	% of Total
Daylight	33079	38544	61.54
Dark - Street Lights On	14549	14475	24.94
Unknown	7270	1695	7.70
Dusk	1624	1944	3.07
Dawn	696	824	1.31

SEVERITYCODE SPEEDING	1	2	% of Total
N	55714	54657	94.84
Y	2474	3531	5.16

MODEL EVALUATION

The values in the current dataset are not in a suitable format to be used as testing and training data for the prediction models. Thus, label encoding is carried out to convert the categorical data to numerical data ranging from digits 0 to 9 .
The data is then normalized using the following code.

```
#initialization
x= df_acc[['WEATHER_cat','ROADCOND_cat','LIGHTCOND_cat','SPEEDING_cat','JUNCTIONTYPE_cat']]
y=df_acc['SEVERITYCODE']
```

```
#Normalizing the dataset
x=preprocessing.StandardScaler().fit(x).transform(x)
x
```

```
array([[ 0.30875114,  1.47118375,  0.32434045, -0.23325396, -1.14943182],
       [ 1.0026519 ,  1.47118375,  0.32434045, -0.23325396, -1.14943182],
       [-0.73210001, -0.71572314,  0.32434045, -0.23325396, -1.14943182],
       ...,
       [-0.73210001, -0.71572314,  0.32434045, -0.23325396, -1.14943182],
       [ 2.39045343,  1.19782039,  2.07035973, -0.23325396, -1.14943182],
       [-0.73210001, -0.71572314,  0.32434045, -0.23325396, -1.14943182]])
```

The normalized data is then split into 70% training and 30% testing data.

```
#Splitting the data as 70 % for training and 30 % for testing

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
print("Train set:", x_train.shape, y_train.shape)
print("Test set:", x_test.shape, y_test.shape)
```

```
Train set: (81463, 5) (81463,)
Test set: (34913, 5) (34913,)
```

MODELING AND PREDICTIONS

To determine the best fit for predicting the SEVERITY CODE, three models are considered.

K-Nearest Neighbors

```
# Training the Model
from sklearn.neighbors import KNeighborsClassifier
k=25

kneigh = KNeighborsClassifier(n_neighbors = k).fit(x_train, y_train)
k_y_pred = kneigh.predict(x_test)
k_y_pred[0:5]
```

```
array([2, 1, 1, 1, 1], dtype=int64)
```

```
#Model Evaluation
j1=jaccard_score(y_test,k_y_pred)
f1=f1_score(y_test,k_y_pred, average = 'macro')
print("Jaccard Score: ",j1)
print("F1 Score: ",f1)
```

```
Jaccard Score:  0.4510596594389318
F1 Score:  0.6148349280182468
```

Decision Tree

```
# Training the Model
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier (criterion = 'entropy', max_depth = 7)

dt.fit(x_train, y_train)
dt_y_pred = dt.predict(x_test)
dt_y_pred[0:5]
```

```
array([2, 1, 1, 1, 1], dtype=int64)
```

```
#Model Evaluation
j2=jaccard_score(y_test,dt_y_pred)
f2=f1_score(y_test,dt_y_pred, average = 'macro')
print("Jaccard Score: ",j2)
print("F1 Score: ",f2)
```

```
Jaccard Score:  0.43849031227672003
F1 Score:  0.6171862904671328
```

Linear Regression

```
# Training the Model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
lr = LogisticRegression(C = 6, solver = 'liblinear').fit(x_train, y_train)

lr_y_pred = lr.predict(x_test)
lr_y_prob = lr.predict_proba(x_test)
lr_y_prob
```

```
array([[0.39458097, 0.60541903],
       [0.66094284, 0.33905716],
       [0.64601659, 0.35398341],
       ...,
       [0.53628072, 0.46371928],
       [0.58710319, 0.41289681],
       [0.51892818, 0.48107182]])
```

```
#Model Evaluation
j3=jaccard_score(y_test,lr_y_pred)
f3=f1_score(y_test,lr_y_pred, average = 'macro')
print("Jaccard Score: ",j3)
print("F1 Score: ",f3)
print("Log Loss: ",log_loss(y_test,lr_y_prob))
```

```
Jaccard Score:  0.45384
F1 Score:  0.6082536169528481
Log Loss:  0.6561502998049485
```

Results and Discussion

-Evaluation metrics used to test the accuracy of our models were the Jaccard index and the f-1 score.
-Choosing different k, max depth, and hyperparameter C values helped to improve the accuracy of our models.
-Through our exploratory data analysis, we have also observed that approximately 43% of all recorded accidents occur at the Blocks but the severity of the accident tends to be higher (SEVERITY CODE= 2 - Injury) at the intersection as compared to the blocks.
-From this exercise, we can draw the result that our model can be trained to determine the severity of an accident to a certain extent using the given dataset and specific parameters such as WEATHER, ROADCOND, LIGHTCOND, SPEEDING, and JUNCTION TYPE.

Conclusion

Based on the evaluation of the above models, it can be concluded that Linear Regression is the most ideal model for this case.