IBM Data Science Capstone:
Car Accident Severity Report Week 3

By Reham Al.Tamimi, 30th September 2020.

Contents

Introduction

Business Problem

As the number of vehicles and transportation means are varying and increasing the number of road accidents are increased too, this project aimed to use the data science and ML methodologies and algorithms to study the historical circumstances recorded during car accident.

Targeted Audience

The improved model will participate in the effort to reduce the severity of car collisions in a community through alerting drivers and roads riders by police, media and local institutions to be more careful in certain bad circumstances (weather, road and visibility conditions).

.

Data Understanding

A historical data for all types of collisions from (2004) until present recorded by “SDOT Traffic Management Division, Traffic Records Group” and provided by Coursera will be used to train and test the developed model.

Descriptive analysis performed to understand the data more, the irrelevant attributes dropped to decrease the computation cost and increase the models efficiency.

In [10]:
```
df.head()
```

Out[10]:

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND	PEDROWNOTGRNT	SDOTCOLNUM	SPEEDING	ST_COLCODE	ST_COLDESC	SEGLANEKEY	CROSSW
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight	NaN	NaN	NaN	10	Entering at angle	0	
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On	NaN	6354039.0	NaN	11	From same direction - both going straight - bo...	0	
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight	NaN	4323031.0	NaN	32	One parked--one moving	0	
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight	NaN	NaN	NaN	23	From same direction - all others	0	
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight	NaN	4028032.0	NaN	10	Entering at angle	0	

5 rows × 38 columns

In [11]:
```
df.columns
```

```
In [11]: df.columns

Out[11]: 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 [13]: dasta.SEVERITYCODE.value_counts()

Out[13]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

In the developed model the attribute 'SEVERITYCODE' selected as dependent (target) variable, because it is used to measure the severity of an accident and labelled as damage (category 1) or injury (category 2) in the dataset, and the attributes 'WEATHER', 'ROADCOND' and 'LIGHTCOND' selected as the model's independent variables.

```
In [9]: dasta.groupby(["SEVERITYCODE"]).count()

Out[9]:
```

	WEATHER	ROADCOND	LIGHTCOND	WEATHER_CAT	ROADCOND_CAT	LIGHTCOND_CAT
SEVERITYCODE						
1	132488	132533	132405	136485	136485	136485
2	57104	57128	57098	58188	58188	58188

Also in the sake of preparing the dataset for analysis, the unneeded attributes removed, and the independent variables encoded to convert the feature type from object to numerical type to be used in the data analysis process to predict the degree of severity.

```
In [31]: # 1. Data Preparation
         # Drop the columns we do not need in our module to decrease the data analysis cost
         dasta = df.drop(columns={'OBJECTID', 'INCKEY', 'COLDETKEY', 'REPORTNO',
                                 'STATUS', 'ADDRTYPE', 'INTKEY', 'LOCATION', 'EXCEPTRSNCODE',
                                 'X', 'Y','EXCEPTRSNDESC', 'SEVERITYCODE.1', 'SEVERITYDESC',
                                 'COLLISIONTYPE', 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT',
                                 'INCDATE', 'INCDTTM', 'JUNCTIONTYPE', 'SDOT_COLCODE', 'SDOT_COLDESC',
                                 'INATTENTIONIND', 'UNDERINFL',
                                 'PEDROWNOTGRNT', 'SDOTCOLNUM', 'SPEEDING', 'ST_COLCODE', 'ST_COLDESC',
                                 'SEGLANEKEY', 'CROSSWALKKEY', 'HITPARKEDCAR'})
         # convert the independent variables into categories rather than objective
         dasta["WEATHER"] = dasta["WEATHER"].astype('category')
         dasta["ROADCOND"] = dasta["ROADCOND"].astype('category')
         dasta["LIGHTCOND"] = dasta["LIGHTCOND"].astype('category')
         #Coding the independent variables
         dasta["WEATHER_CAT"] = dasta["WEATHER"].cat.codes
         dasta["ROADCOND_CAT"] = dasta["ROADCOND"].cat.codes
         dasta["LIGHTCOND_CAT"] = dasta["LIGHTCOND"].cat.codes
         dasta.dtypes

Out[31]: SEVERITYCODE        int64
         WEATHER          category
         ROADCOND         category
         LIGHTCOND        category
         WEATHER_CAT          int8
         ROADCOND_CAT         int8
         LIGHTCOND_CAT        int8
         dtype: object
```

Also according to the descriptive analysis, number of records where SEVERITYCODE =1 counts (136485), while SEVERITYCODE =2 counts 58188.

```
In [32]: dasta.SEVERITYCODE.value_counts()

Out[32]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

We can fix this by downsampling the majority class (SEVERITYCODE =1).

```
In [34]: #Balance the data set
         from sklearn.utils import resample
         data_more = dasta[dasta.SEVERITYCODE ==1]
         data_less = dasta[dasta.SEVERITYCODE ==2]
         data_more_equal = resample(data_more,
                                    replace=False,
                                    n_samples=58188,
                                    random_state=99)
         data_bal = pd.concat([data_more_equal,data_less])
         data_bal.SEVERITYCODE.value_counts()

Out[34]: 2    58188
         1    58188
         Name: SEVERITYCODE, dtype: int64
```

Methodology

In order to predict the degree of severity, the following models used:

- K-Nearest Neighbour (KNN): to predict the severity code of an outcome by finding the most similar to data point within k distance.

- Decision Tree: this model will gives a layout of all possible outcomes, so we can fully analyze the consequences of a decision. The decision tree observes all possible outcomes of different weather conditions.

- Logistic Regression: As the dataset only provides two severity code outcomes, the model will only predict one of those two classes. This makes our data binary, which is perfect to use with logistic regression.

In order to build and run these models, the independent and dependent variables defined as X and Y and then dataset normalized.

```
In [35]: from sklearn import preprocessing
         # determine x = the independent varaibles and y as the dependent variable
         x = np.asarray(data_bal[['WEATHER_CAT','ROADCOND_CAT','LIGHTCOND_CAT']])
         y = np.asarray(data_bal['SEVERITYCODE'])
         print ('The value of x before preporcessing ',x[0:5])
         print ('The value of x before preporcessing ',y[0:5])

         x=preprocessing.StandardScaler().fit(x).transform(x)
         print ('The value of x after preporcessing ',x[0:5])

         The value of x before preporcessing  [[ 6  8  5]
          [ 1  0  5]
          [ 6  8  5]
          [10  7  8]
          [ 6  8  5]]
         The value of x before preporcessing  [1 1 1 1 1]
         The value of x after preporcessing  [[ 1.15109535  1.53056569  0.42621151]
          [-0.67433413 -0.67001234  0.42621151]
          [ 1.15109535  1.53056569  0.42621151]
          [ 2.61143893  1.25549344  2.07234651]
          [ 1.15109535  1.53056569  0.42621151]]
```

Then the dataset splitted into (80%) used for training the models and (20%) for testing.

```
In [36]: # Split the data set into train (80%) and test (20%)
         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)
```

After that the three models built:

- Logistic Regression model

```
In [37]: # Logistic regression
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix

         LR = LogisticRegression(C=0.01, solver='liblinear').fit(x_train,y_train)
         LR
         yhat_LR = LR.predict(x_test)
         print('Logistic regression = ',yhat_LR)

         yhat_prob_LR = LR.predict_proba(x_test)
         print('Logistic regression probability = ',yhat_prob_LR)

         Logistic regression =  [2 1 2 ... 2 2 2]
         Logistic regression probability =  [[0.47262299 0.52737701]
          [0.68377027 0.31622973]
          [0.4655792  0.5344208 ]
          ...
          [0.47262299 0.52737701]
          [0.47262299 0.52737701]
          [0.47262299 0.52737701]]
```

- KNN model

```
In [38]: # KNN
         from sklearn.neighbors import KNeighborsClassifier
         k = 15
         #Train Model and Predict
         neigh = KNeighborsClassifier(n_neighbors = k).fit(x_train,y_train)
         neigh
         yhat_KNN = neigh.predict(x_test)
         yhat_KNN[0:5]

Out[38]: array([2, 1, 1, 1, 1])
```

- Decision Tree

```
In [39]: # DecisionTreeClassifier
         from sklearn.tree import DecisionTreeClassifier
         DecTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
         DecTree
         DecTree.fit(x_train,y_train)
         pred_DT = DecTree.predict(x_test)
         print (pred_DT [0:5])
         print (y_test [0:5])

         [2 1 2 2 2]
         [2 1 2 1 2]
```

Results

The results gained from the above models evaluated using the logloss, F1_Score and Jaccard_simulirity_score:

```
In [40]: # Evaluate the three models to see which one is more accurate
         from sklearn.metrics import jaccard_similarity_score
         from sklearn.metrics import f1_score
         from sklearn.metrics import log_loss

         #1.Evaluate LR model
         LRjaccard = jaccard_similarity_score(y_test, yhat_LR)
         print ('LR jaccard_similarity_score = ',LRjaccard)
         LRlog_loss = log_loss(y_test, yhat_prob_LR)
         print ('LR log_loss = ',LRlog_loss)
         LR_f1_score= f1_score(y_test, yhat_LR, average='macro')
         print ('LR F1 score = ',LR_f1_score)
         print ('***********************')
         #2. Evaluate KNN model
         KNNjaccard = jaccard_similarity_score(y_test, yhat_KNN)
         print ('KNN jaccard_similarity_score = ',KNNjaccard)
         KNN_f1_score= f1_score(y_test, yhat_KNN, average='macro')
         print ('KNN F1 score = ',KNN_f1_score)
         print ('========================')
         #3.Decision Tree Model
         DTjaccard = jaccard_similarity_score(y_test, pred_DT)
         print ('Decision Tree jaccard_similarity_score = ',DTjaccard)
         DT_f1_score= f1_score(y_test, pred_DT, average='macro')
         print ('Decision Tree F1 score = ',DT_f1_score)

         LR jaccard_similarity_score =  0.5295583433579653
         LR log_loss =  0.6843902239068189
         LR F1 score =  0.5142657138878495
         ***********************
         KNN jaccard_similarity_score =  0.5566248496305207
         KNN F1 score =  0.5449387158254655
         ========================
         Decision Tree jaccard_similarity_score =  0.5648307269290256
         Decision Tree F1 score =  0.48361182945584147
```

Discussion

In the beginning of the project notebook, we had categorical data that need to encoded because they not a data type that we could have fed through an algorithm, so label encoding was used to created new classes that were of type int8; a numerical data type.

After that issue we were presented with another imbalanced of data. As mentioned earlier, class 1 was nearly three times larger than class 2, so the solution to this was downsample the majority class (1) to (58188) record each.

Once we analysed and cleaned the data, it was then fed through three ML models; K-Nearest Neighbor, Decision Tree and Logistic Regression. Although the first two are ideal for this project, logistic regression made most sense because of its binary nature.

Evaluation metrics used to test the accuracy of our models were jaccard index, F1_score and logloss for logistic regression.

Conclusion

Based on historical data from weather conditions pointing to certain classes, we can conclude that particular weather conditions have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2).

Farther studies to select different attributes may result in determining independent attribute that influence strongly the predicting of Serveritycode, as the best evaluation for logistic regression did not exceed 0.68.

```
LR jaccard_similarity_score =  0.5295583433579653
LR log_loss =  0.6843902239068189
LR F1 score =  0.5142657138878495
***********************
KNN jaccard_similarity_score =  0.5566248496305207
KNN F1 score =  0.5449387158254655
========================
Decision Tree jaccard_similarity_score =  0.5648307269290256
Decision Tree F1 score =  0.48361182945584147
```