Car Accident Severity Prediction in Seattle

Ankit Kapoor

October 2020

1. Introduction

1.1. Background

Car accidents contributes to biggest cause of injuries and deaths. They also impact delivery time ,commuting time &contributes to pollution.Due to accidents massive number of cars are struck waiting for the road to be cleared and lead to inconvenience

1.2. Problem

So this project aims to reduce the collisions in a community and for that an algorithm has to developed. This is done to predict the severity of accidents depending upon the factors such as current weather, road and visibility conditions which are already given to us. An intimation will be given to driver when these conditions are bad by the model .So here the target audience are the drivers in the region(Seattle) which is mentioned in dataset and it is important to solve since it aims to reduce the collisions or accidents

1.3. Interest

As per the described problem and its analysis interested ones will contribute to these two categories of stakeholders:

• Individuals, being them work commuters or professional taxi, truck or bus drivers

• Businesses, like logistic companies, public/private passengers bus companies, taxi

companies, government agencies (urban/suburban mobility managers)

2. Data

2.1. Data sources

Collision data had been fetched from Seattle Department of Transportation Open Data Program in CSV format.

Source :https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

2.2. Data Understanding and Feature Selection

So in this case we are using 'SEVERITYCODE' as predictor or target variable since we can measure or depict the severity of an accident from 0 to 5 within the dataset.

Here attributes used to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.

The codes for severity are as follows:

0 : Little to no Probability (Clear Conditions)

1 : Very Low Probablility - Chance or Property Damage

2 : Low Probability - Chance of Injury

3 : Mild Probability - Chance of Serious Injury

4 : High Probability - Chance of Fatality

Now we will extract the dataset and convert¶

In the raw form data is not as worth to be used directly. So we will drop the non essentials and work on important attributes

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND
0	2	Overcast	Wet	Daylight
1	1	Raining	Wet	Dark - Street Lights On
2	1	Overcast	Dry	Daylight
3	1	Clear	Dry	Daylight
4	2	Raining	Wet	Daylight

So we can see that there is some sort of relation in between the attributes that are discussed above as of now.So we can treat it as an exampe. We can see when lightcond is dadylight ,roadcond is wet,weather is overcast then collision type is angles with severitycode of 2 which means low probability-Chance of injury in this case

We must use label encoding to covert the features to our desired data type.

	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

	SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	WEATHER_cat	ROADCOND_cat	LIGHTCOND_cat
2	1	Overcast	Dry	Daylight	4	0	5
3	1	Clear	Dry	Daylight	1	0	5
4	2	Raining	Wet	Daylight	6	8	5

With the new columns, we can now use this data in our analysis and ML models

Balancing the Dataset

In this dataset our target variable SEVERITYCODE is only 42% balanced. We can notice that severitycode in class 1 is nearly three times the size of class 2.

3. Methodology

Now our data is ready for machine learning models.

We will use the following models:

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

Decision Tree : A decision tree model gives us a layout of all possible outcomes so we can fully analyze the consequences of a decision. It context, the decision tree observes all possible outcomes of different weather conditions.

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

4. Evaluation

Confronting the metrics of the different models, we look for the highest F1 score, the largest

Jaccard score and the smallest log loss.

So I selected the Decsion Tree, and retrained the model over the integral Data Set (without

splitting it).

Initialization

Define X and y

```
import numpy as np
X = np.asarray(colData_balanced[['WEATHER_CAT', 'ROADCOND_CAT', 'LIGHTCOND_CA
X[0:5]
```

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

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

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

Normalize the dataset

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validation.py:595: DataConversionWarning: Data with input dtype int8 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/utils/validation.py:595: DataConversionWarning: Data with input dtype int8 w
as converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

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

Train/Test Split

We will use 30% of our data for testing and 70% for training.

```
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, rand
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (81463, 3) (81463,)
Test set: (34913, 3) (34913,)
```

Here we will begin our modelling and predictions...

K-Nearest Neighbors (KNN)

```
# Building the KNN Model
from sklearn.neighbors import KNeighborsClassifier

k = 25
```

```
#Train Model & Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh

Kyhat = neigh.predict(X_test)
Kyhat[0:5]
```

```
6]: array([2, 2, 1, 1, 2])
```

Decision Tree

```
# Building the Decision Tree
from sklearn.tree import DecisionTreeClassifier
colDataTree = DecisionTreeClassifier(criterion="entropy", max_depth = 7)
colDataTree
colDataTree.fit(X_train,y_train)
```

```
.1]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=
     7,
                 max_features=None, max_leaf_nodes=None,
                 min_impurity_decrease=0.0, min_impurity_split=None,
                 min_samples_leaf=1, min_samples_split=2,
                 min_weight_fraction_leaf=0.0, presort=False, random_state=Non
     e,
                 splitter='best')
```

```
# Train Model & Predict
DTyhat = colDataTree.predict(X_test)
print (predTree [0:5])
print (y_test [0:5])
```

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

Logistic Regression

```
# Building the LR 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
```

```
244]: LogisticRegression(C=6, class_weight=None, dual=False, fit_intercept=True,
                intercept_scaling=1, max_iter=100, multi_class='warn',
                n_jobs=None, penalty='l2', random_state=None, solver='liblinear
       ',
                tol=0.0001, verbose=0, warm_start=False)
```

```
# Train Model & Predicr
LRyhat = LR.predict(X_test)
LRyhat
```

```
245]: array([1, 2, 1, ..., 2, 2, 2])
```

```
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

```
246]: array([[0.57295252, 0.42704748],
             [0.47065071, 0.52934929],
             [0.67630201, 0.32369799],
             ...,
             [0.46929132, 0.53070868],
             [0.47065071, 0.52934929],
             [0.46929132, 0.53070868]])
```

Results & Evaluation

Now we will check the accuracy of our models.

K-Nearest Neighbor

```
]: # Jaccard Similarity Score
   jaccard_similarity_score(y_test, Kyhat)
[197]: 0.564001947698565
```

```
]: # F1-SCORE
   f1_score(y_test, Kyhat, average='macro')
[198]: 0.5401775308974308
```

Model is most accurate when k is 25.

Decision Tree

```
]: # Jaccard Similarity Score
   jaccard_similarity_score(y_test, DTyhat)
[213]: 0.5664365709048206
```

```
]: # F1-SCORE
   f1_score(y_test, DTyhat, average='macro')
[214]: 0.5450597937389444
```

Model is most accurate with a max depth of 7.

Logistic Regression

```
]: # Jaccard Similarity Score
   jaccard_similarity_score(y_test, LRyhat)
[247]: 0.5260218256809784
```

```
]: # F1-SCORE
   f1_score(y_test, LRyhat, average='macro')
[248]: 0.511602093963383
```

```
]: # LOGLOSS
   yhat_prob = LR.predict_proba(X_test)
   log_loss(y_test, yhat_prob)
[249]: 0.6849535383198887
```

Model is most accurate when hyperparameter C is 6.

Steps used

Earlier we had categorical data that was of type 'object'. This is not a data type that could be used to fed to algorithm, so we used label coding & created new classes that were of type int8; a numerical data type.

After solving that issue we were presented with another - imbalanced data. As mentioned earlier, class 1 was nearly three times larger than class 2. The solution to this was downsampling the majority class with sklearn's resample tool. We downsampled to match the minority class exactly with 58188 values each.

After analysing and cleaning 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 the most sense because of its binary nature.

Evaluation metrics used to test the accuracy of our models were jaccard index, f-1 score and logloss for logistic regression. Choosing different k, max

depth and hyperamater C values helped to improve our accuracy to be the best possible.

5. Conclusions

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).