Predicting Car Accident Severity

Mouhamadou Nabi NDOYE

October 2020

1. Introduction

Road accident is an important scourge that lead health damage, and increase insurance fees. By identifying the factors of accident severity life can be saved, public health improved and save insurance fees.

In order to reduce the number of car accident, a model must be developed to anticipate by making prediction of car accident severity.

This prediction will be used to alert the driver to be more careful when conditions are gathered for an accident.

An insurance company could use this model as a service to alert their customer, thus we can reduce the car accident and the company can make more benefit by saving compensation due to car accident.

2. Data

A dataset of 194673 records occurring between 2004 and 2020 and having 37 columns describing the details of each accident including the accident severity, weather conditions, road conditions, light conditions, location, speeding...

Our target is to predict the severity of an accident using dependent variable 'SEVERITYCODE' which has two values (1 or 2) corresponding to the severity of the collision. For the independent variable we will use the weather conditions, road conditions, light conditions, locations and speeding which will be categorized and converted to code during the data preparation.

	SEVERITYCODE	WEATHER_CODE	ROADCOND_CODE	LIGHTCOND_CODE	SPEEDING_CODE	LOCATION_CODE
0	2	4	8	5	-1	8793
1	1	6	8	2	-1	10707
2	1	4	0	5	-1	8049
3	1	1	0	5	-1	4647
4	2	6	8	5	-1	22787

<u>The type of data that will be used for the prediction:</u>

```
Out[23]: SEVERITYCODE          int64
         WEATHER            category
         ROADCOND           category
         LIGHTCOND          category
         LOCATION           category
         SPEEDING           category
         WEATHER_CODE           int8
         ROADCOND_CODE          int8
         LIGHTCOND_CODE         int8
         LOCATION_CODE         int16
         SPEEDING_CODE          int8
         dtype: object
```

3. Methodology

After exploring the data, we are now ready to build the models based on **Logistic Regression** and **Decision Tree**.

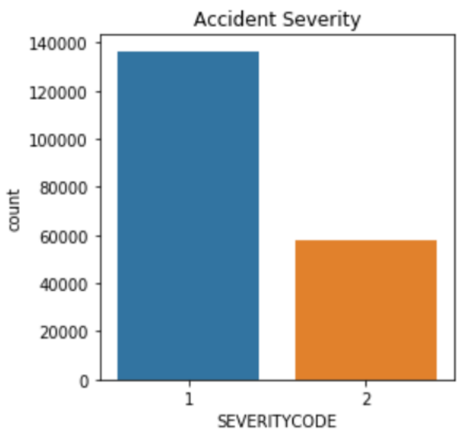
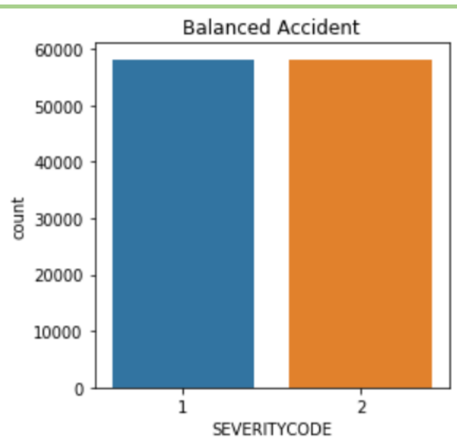
Logistic Regression Model: our dependent variable is binary and this model will only predict one of those two classes.

Decision Tree: it will allow us to observe all possible outcomes.

As methodology I will first **Balance the dataset**, the **define my independent variable (X) and the dependent variable (y)**, **normalize the data** and **split into training set and test set** (30% of the dataset). And at last I will **build the models**.

- **<u>Balancing dataset:</u>**

In order create the least-biased model, we will randomly balance our target variable SEVERITYCODE where records with severity 1 is huge comparing to the record with severity 2.

Original record	Balanced record
	

- **Defining X and y:**

WEATHER_CODE, ROADCOND_CODE, LIGHTCOND_CODE, LOCATION_CODE and SPEEDING_CODE will be used for the prediction.

```
X = np.asarray(study[['WEATHER_CODE','ROADCOND_CODE','LIGHTCOND_CODE', 'LOCATION_CODE', 'SPEEDING_CODE']])
X[0:5]
```

```
array([[ 4.0000e+00,  8.0000e+00,  5.0000e+00,  8.7930e+03, -1.0000e+00],
       [ 6.0000e+00,  8.0000e+00,  2.0000e+00,  1.0707e+04, -1.0000e+00],
       [ 4.0000e+00,  0.0000e+00,  5.0000e+00,  8.0490e+03, -1.0000e+00],
       [ 1.0000e+00,  0.0000e+00,  5.0000e+00,  4.6470e+03, -1.0000e+00],
       [ 6.0000e+00,  8.0000e+00,  5.0000e+00,  2.2787e+04, -1.0000e+00]])
```

SEVERITYCODE will be our predicted value.

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

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

- **Normalizing the dataset:**

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

```
array([[ 0.35364615,  1.50545441,  0.3912104 , -0.45743913, -0.22440165],
       [ 1.04520829,  1.50545441, -1.18714134, -0.17720325, -0.22440165],
       [ 0.35364615, -0.68713674,  0.3912104 , -0.56637095, -0.22440165],
       [-0.68369706, -0.68713674,  0.3912104 , -1.06447047, -0.22440165],
       [ 1.04520829,  1.50545441,  0.3912104 ,  1.59147464, -0.22440165]])
```

- **Splitting into Train/Test set:** We will use 30% of our data for testing and 70% for training

```
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: (136271, 5) (136271,)
Test set: (58402, 5) (58402,)
```

- **Building the model:**

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
```

```
LogisticRegression(C=0.01, 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)
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth=4)
dt.fit(X_train, y_train)
dt
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
            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=None,
            splitter='best')
```

4. Results & Evaluation

I will show in this section the results of the model and the accuracy of the model.

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
```

```
LogisticRegression(C=0.01, 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)
```

```
yhat_lr = LR.predict(X_test)
yhat_lr[0:5]
```

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

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

```
array([[0.73023109, 0.26976891],
       [0.68054332, 0.31945668],
       [0.65708122, 0.34291878],
       ...,
       [0.58469252, 0.41530748],
       [0.8187009 , 0.1812991 ],
       [0.68958664, 0.31041336]])
```

```
from sklearn.metrics import jaccard_similarity_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

```
print("Jaccard: ",jaccard_similarity_score(y_test, yhat_lr))
print("F1 Score: ",f1_score(y_test,yhat_lr,average='weighted'))
print("LogLoss :",log_loss(y_test, yhat_prob))
```

```
Jaccard:  0.7030581144481354
F1 Score:  0.5812429210363583
LogLoss : 0.6003036801500582
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth=4)
dt.fit(X_train, y_train)
dt
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
            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=None,
            splitter='best')
```

```
yhat_dt = dt.predict(X_test)
yhat_dt[0:5]
```

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

```
print("F1 Score: ",f1_score(y_test,yhat_dt,average='weighted'))
print("Jaccard: ",jaccard_similarity_score(y_test, yhat_dt))
```

```
F1 Score:  0.5809904188654375
Jaccard:  0.7034519365775145
```

Performance of models :

	LOGISTIC REGRESSION	DECISION TREE
LOG LOSS	0.60	
F1 SCORE	0.58	0.58
JACCARD	0.70	0.70

5. Discussion

Car accident data for the city of Seattle between 2004 and 2020 has been used to train and evaluate machine learning models (Logistic Regression and Decision Tree) for predicting accident severity based on the location, speeding, weather, road and light conditions.

We had categorical data that was of type 'object', so label encoding was used to created new classes that were of type int8; a numerical data type.

We had unbalance data (class 1 was nearly three times larger than class 2), the solution was here to down sample in order to match the minority class

We chose two machines learning models: Logistic regression (because of the binary characteristic of the predicted value) and Decision Tree (in order to observe all possible outcomes).

And then we evaluate the models by using Jaccard index, F1-Score and Log Loss.

These models can be extended to include new features or applied to accident databases in other cities/regions.

6. Conclusion

In conclusion we build model based on the location, speeding, weather, road and light conditions which can help the decision making of alerting motorists and emergency services call handlers.

This model can be used by and insurance company to alert customer about risk of accident which can save life, improve public health and make benefit for the company by saving compensations.