Severity of Accident in Seattle

Imanie Yahya

Introduction/Business Problem

This project objective is to help drivers plan for their trip safely, indirectly will result in reduction on car accident frequency in a community. An algorithim will be developed to predict the severity of an accident given the weather, road and visibility conditions.

Data

- Target variable for this analysis will be 'SEVERITYCODE' because it is used measure the severity of an accident from 0 to 5 within the dataset. Attributes used to weigh the severity of an accident are 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.
- Severity codes are as follows:

CODE	SEVERITY
0	Little to no Probability
1	Very Low Probablility (Chance or Property Damage)
2	Low Probability - Chance of Injury
3	Mild Probability - Chance of Severe Injury
4	High Probability - Chance of Fatality

Extract Dataset & Convert

- In it's original form, this data is not fit for analysis. For one, there are many columns that we will not use for this model. Also, most of the features are of type object, when they should be numerical type.
- We must use label encoding to covert the features to our desired data type.

	WEATHER	ROADCOND	LIGHTCOND	SEVERITYCODE
0	1	2	0	2
1	3	2	1	1
2	1	0	0	1
3	0	0	0	1
4	3	2	0	2

```
WEATHER          int64
ROADCOND         int64
LIGHTCOND        int64
SEVERITYCODE     int64
dtype: object
```

Balancing the Dataset

- Our target variable SEVERITYCODE is only 43% balanced. In fact, SEVERITYCODE in class 1 is nearly three times the size of class 2.

- We can fix this by down sampling the majority class.

```
2    57052
1    57052
Name: SEVERITYCODE, dtype: int64
```

Methodology

1. K-Nearest Neighbor(KNN)
2. Logistic Regression

Initialization

- Define X & Y

```
balanced_df.columns
```

```
Index(['WEATHER', 'ROADCOND', 'LIGHTCOND', 'SEVERITYCODE'], dtype='object')
```

```
#Define X & Y
X = np.asarray(balanced_df[['WEATHER','ROADCOND','LIGHTCOND']])
X[0:5]
```

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0],
       [1, 0, 0],
       [3, 2, 1]])
```

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

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

- Normalize Dataset

```
# Normalize the data sets

from sklearn import preprocessing
from sklearn.model_selection import train_test_split

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

```
/opt/conda/envs/Python36/lib/python3.6/site-packages/skl
was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages/skl
was converted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
array([[-0.64746494, -0.63791567, -0.68713573],
       [-0.64746494, -0.63791567, -0.68713573],
       [-0.64746494, -0.63791567, -0.68713573],
       [ 0.20438442, -0.63791567, -0.68713573],
       [ 1.90808313,  1.5779089 ,  1.28999095]])
```

- Train/Test Split

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
print('Train set: ',X_train.shape,y_train.shape)
print('Test set: ',X_test.shape,y_test.shape)
```

```
Train set:  (79872, 3) (79872,)
Test set:  (34232, 3) (34232,)
```

Modelling & Predictions

- K-Nearest Neighbor (KNN)

```
# KNN

from sklearn.neighbors import KNeighborsClassifier
k = 25

neigh = KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
neigh
Kyhat = neigh.predict(X_test)
Kyhat[0:5]
```

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

Modelling & Predictions (cont.)

- Linear Regression

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=6,solver='liblinear').fit(X_train,y_train)
LR
```

```
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)
```

```
LRyhat = LR.predict(X_test)
LRyhat
```

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

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

```
array([[0.49182137, 0.50817863],
       [0.51565304, 0.48434696],
       [0.50677867, 0.49322133],
       ...,
       [0.49182137, 0.50817863],
       [0.52763992, 0.47236008],
       [0.47096431, 0.52903569]])
```

Result & Evaluation

```
#Check the accuracy of the models

#1. KNN
import numpy as np
from sklearn.metrics import jaccard_similarity_score
#Jaccard Similarity Score
jaccard_similarity_score(y_test,Kyhat)
```

```
0.5065435849497546
```

```
#2. F1 Score
from sklearn.metrics import f1_score
#f1-score
f1_score(y_test,Kyhat,average='macro')
```

```
0.39336498061881875
```

```
from sklearn.metrics import accuracy_score
acc= accuracy_score(y_test,Kyhat)
acc
```

```
0.5065435849497546
```

```
#Logistic Regression
#Jaccard Similarity Score
jaccard_similarity_score(y_test,LRyhat)
```

```
0.5174398223884085
```

```
f1_score(y_test,LRyhat,average = 'macro')
```

```
0.5057025066041737
```

```
# LOGLOSS
from sklearn.metrics import log_loss
yhat_prob = LR.predict_proba(X_test)
log_loss(y_test,yhat_prob)
```

```
0.692135988660187
```

```
from sklearn.metrics import accuracy_score
acc= accuracy_score(y_test,LRyhat)
acc
```

```
0.5174398223884085
```

Discussion

- Dataset provide is categorical data which is type 'object'. In order to carry the analysis using algorithm, label encoding is utilize to convert the data 'object' type to 'numerical' type.
- Datasets provided is also imbalanced. As observed, SEVERITYCOD=1 was more than 2 times larger than SEVERITYCOD=2. Downsampling the majority class with sklearn's resample tool id used to the minority class exactly with 57052 values each.
- After the data is processed, they were analyze using ML models; K-Nearest Neighbor and Logistic Regression. In this case, logistic regression is best used because of its binary nature.
- Evaluation metrics used to test the accuracy of our models were jaccard index, f-1 score and log loss for logistic regression. Choosing different k and C values helped to improve the accuracy of the model.

Conclusion

- Based on historical data from weather conditions is the main factor that contribute to the different category of SEVERITYCODE.