Peer-graded Assignment: Capstone Project - Car accident severity (Week 2)

Introduction

The costs of injuries due to traffic accidents have a great impact on the society. In recent years, researchers and data scientists have paid attention to factors that caused the traffic accidents. Applying data mining techniques to model traffic accident data records can help to understand the characteristics of driver's behaviour, roadway condition and weather condition that were causally connected with different injury severity

Local and State Highway and Road Departments as well Traffic Departments would be interested in accurate prediction of car collisions, for the advantage and business values of making the roads safer. Others who are interested in this model would be car manufacturers to see if they can make there vehicles even safer.

Describe Data

The data was collected from Seattle police department, and our predictor will be 'SEVERITYCODE', which contains different levels of 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 following:

0: Little to no Probability (Clear Conditions)
1: Very Low Probability — Chance or Property Damage
2: Low Probability — Chance of Injury
3: Mild Probability — Chance of Serious Injury
4: High Probability — Chance of Fatality

The data set has 38 independent variables and 194673 records. Many of the columns are object types, and some columns and rows have null values, which need to be pre-processed before any further processing.

```
[6]: df.shape

[6]: (194673, 38)

[7]: df.dtypes

[7]: SEVERITYCODE         int64
     X                   float64
     Y                   float64
     OBJECTID              int64
     INCKEY                int64
     COLDETKEY             int64
     REPORTNO             object
     STATUS               object
     ADDRTYPE             object
     INTKEY              float64
     LOCATION             object
     EXCEPTRSNCODE        object
     EXCEPTRSNDESC        object
     SEVERITYCODE.1        int64
     SEVERITYDESC         object
     COLLISIONTYPE        object
     PERSONCOUNT           int64
     PEDCOUNT              int64
     PEDCYLCOUNT           int64
     VEHCOUNT              int64
     INCDATE              object
     INCDTTM              object
```

```
[11]: pre_df.dtypes

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

Data Processing

We can see from below image that the original dataset is not ready for analgising and it is imbalance, so we need to balance it first.

```
[12]: pre_df['SEVERITYCODE'].value_counts()

[12]: 1    136485
      2     58188
      Name: SEVERITYCODE, dtype: int64
```

Let’s balance date set for further analysis.

```
# Balance Dataset for analysis

from sklearn.utils import resample

pre_df_major = pre_df[pre_df.SEVERITYCODE==1]
pre_df_min = pre_df[pre_df.SEVERITYCODE==2]

pre_df_major_dsample = resample(pre_df_major,
                                replace=False,
                                n_samples=58188,
                                random_state=123)
```

```
balanced_df = pd.concat([pre_df_major_dsample, pre_df_min])
balanced_df.SEVERITYCODE.value_counts()
```

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

Methodology

In this report, i am going to use three models for machine learning:

- K Nearest Neighbour(KNN)
- Decision Tree
- Logistic Regression

Define X & y before using the three models

```
# Define X,y

X = np.asarray(balanced_df[['WEATHER_CAT','ROADCOND_CAT','LIGHTCOND_CAT']])
X[0:5]
```

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

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

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

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

```
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

Let's use 30% of our data for testing and 70% for training.

```
import numpy as np
from sklearn.metrics import jaccard_similarity_score
```

```
from sklearn.metrics import f1_score
```

```
from sklearn.metrics import log_loss
```

```
# Train & Test sets

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, random_state=4)

print('Train set rows:', X_train.shape[0])
print('Test set rows', X_test.shape[0])
```

```
Train set rows: 81463
Test set rows 34913
```

let's begin modelling and predictions

KNN -the k-nearest neighbours algorithm is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure, so KNN will help us predict the severity code by finding the most similar points within k distance.

```
# K Nearest Neighbor(KNN)

from sklearn.neighbors import KNeighborsClassifier
k=17
knn = KNeighborsClassifier(n_neighbors =k).fit(X_train, y_train)

knn_y_pred = knn.predict(X_test)
knn_y_pred[0:5]
```

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

```
# KNN Evaluation

jaccard_similarity_score(y_test, knn_y_pred)
```

```
0.5603643342021597
```

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

```
0.5477714681769319
```

Decision Tree belongs to the family of supervised learning algorithms. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from training data. It will give us a layout of all possible outcomes so we can fully analyse the consequences of a decision. so that we could see different weather conditions from all possible outcomes.

```
# Decision Tree

from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth = 7)

dt.fit(X_train,y_train)
```

```
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=None,
            splitter='best')
```

```
dt_y_pred = dt.predict(X_test)
```

```
# Decision Tree Evaluation

jaccard_similarity_score(y_test, dt_y_pred)
```

```
0.5664365709048206
```

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

```
0.5450597937389444
```

Logistic Regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. 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.

```
# 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)
```

```
LR_y_pred = LR.predict(X_test)
LR_y_prob = LR.predict_proba(X_test)
LR_y_prob = LR.predict_proba(X_test)
log_loss(y_test, LR_y_prob)
```

```
0.6849535383198887
```

```
# Linear Regression Evaluation

jaccard_similarity_score(y_test, LR_y_pred)
```

```
0.5260218256809784
```

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

```
0.511602093963383
```

Result

The final results of the model evaluations show as following:

ML Model	JAC Score	F1 Score	Accuracy
KNN	0.3	0.55	0.56
Decision Tree	0.28	0.54	0.57
Logistic Regression	0.27	0.51	0.53

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).