IBM Data Science Capstone:

Car collision severity Report

Introduction

Car accidents can occur all the time, however there are some conditions were the probabilities of have an accident arise due multiple variables. The Seattle government is going to prevent avoidable car accidents by developing the algorithm to predict the severity of an accident given the current weather, road and visibility conditions. When conditions are bad, this model will alert drivers to remind them to be more careful.

In most cases, not paying enough attention during driving, abusing drugs and alcohol or driving at very high speed are the main causes of occurring accidents that can be prevented by enacting harsher regulations. Besides the aforementioned reasons, weather, visibility, or road conditions are the major uncontrollable factors that can be prevented by revealing hidden patterns in the data and announcing warning to the local government, police and drivers on the targeted roads.

The target audience of the project is local Seattle government, police, rescue groups, and last but not least, car insurance institutes. The model and its results are going to provide some advice for the target audience to make insightful decisions for reducing the number of accidents and injuries for the city. The information was provided by Seattle Police Department from 2004 to 2020.

Libraries Which are Used to Develope the Project:
- Pandas: For creating and manipulating dataframes.
- Numpy: for working with arrays, linear algebra, fourier transform, and matrices.
- Scikit Learn: For importing KNN, decision tree and linear regression.
- Matplotlib: Python Plotting Module.

Data

There are 194,673 observations and 38 variables in this data set. Since we would like to identify the factors that cause the accident and the level of severity, we will use SEVERITYCODE as our dependent variable Y, and try different combinations of independent variables X to get the result. Since the observations are quite large, we may need to filter out the missing value and delete the unrelated columns first. Then we can select the factor which may have more impact on the accidents, such as address type, weather, road condition, and light condition.

The target Data to be predicted under (SEVERITYCODE 1-prop damage 2-injury) label. Other important variables include:

- ADDRTYPE: Collision address type: Alley, Block, Intersection
- LOCATION: Description of the general location of the collision
- PERSONCOUNT: The total number of people involved in the collision helps identify severity involved
- PEDCOUNT: The number of pedestrians involved in the collision helps identify severity involved
- PEDCYLCOUNT: The number of bicycles involved in the collision helps identify severity involved
- VEHCOUNT: The number of vehicles involved in the collision identify severity involved
- JUNCTIONTYPE: Category of junction at which collision took place helps identify where most collisions occur
- WEATHER: A description of the weather conditions during the time of the collision
- ROADCOND: The condition of the road during the collision
- LIGHTCOND: The light conditions during the collision
- SPEEDING: Whether or not speeding was a factor in the collision (Y/N)
- SEGLANEKEY: A key for the lane segment in which the collision occurred
- CROSSWALKKEY: A key for the crosswalk at which the collision occurred
- HITPARKEDCAR: Whether or not the collision involved hitting a parked car

Methodology

After studying the dataset, I choose four columns to analyze and they are: “WEATHER”, “ROADCOND”, “LIGHTCOND” and “SEVERITYCODE” is the target variable. After processing these values into analyzable data, I will employ three machine learning models to make prediction:

K-Nearest Neighbor (KNN)

KNN will help us 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. Its context, the decision tree observes all possible outcomes of different weather conditions.

Logistic Regression

Because 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.

Data processing

I select the four columns out of the original dataset and remove the records that has null value. But this data is still not fit for analysis, 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.

SEVERITYCODE	WEATHER	ROADCOND	LIGHTCOND	WEATHER_CODE	ROADCOND_CODE	LIGHTCOND_CODE
1	Clear	Dry	Daylight	1	0	5
1	Clear	Dry	Daylight	1	0	5
1	Clear	Dry	Daylight	1	0	5
1	Overcast	Dry	Daylight	4	0	5
1	Raining	Wet	Dark - Street Lights On	6	8	2
...	...	...	...	...	...	...
2	Raining	Wet	Daylight	6	8	5
2	Clear	Wet	Daylight	1	8	5
2	Clear	Dry	Daylight	1	0	5
2	Clear	Dry	Daylight	1	0	5
2	Clear	Dry	Dusk	1	0	6

To get a good understanding of the dataset, I have checked different values in the features. The results show, the target feature is imbalance, so we use a simple statistical technique to balance it.

```
project_df['SEVERITYCODE'].value_counts()
```

```
1    132285
2     57052
Name: SEVERITYCODE, dtype: int64
```

As we can see, the number of rows in class 1 is almost three times bigger than the number of rows in class 2. It is possible to solve the issue by down sampling the class 1.

In [9]:
```
project_df_1 = project_df[project_df['SEVERITYCODE'] == 1] = project_df[project_df['SEVERITYCODE'] == 1]
project_df_2 = project_df[project_df['SEVERITYCODE'] == 2]
```

In [10]:
```
project_df_1_dsample = resample(project_df_1,
                                replace=False,
                                n_samples=57052,
                                random_state=123)

balanced_df = pd.concat([project_df_1_dsample, project_df_2])
balanced_df['SEVERITYCODE'].value_counts()
```

Out[10]:
```
2    57052
1    57052
Name: SEVERITYCODE, dtype: int64
```

Now the data is perfectly balanced.

The last step before we start machine learning algorithm, we need to initialize the test and train data. Then we can begin our modeling and predictions.

In [16]:
```
X = np.asarray(balanced_df[['WEATHER_CODE', 'ROADCOND_CODE', 'LIGHTCOND_CODE']])
X[0:5]
```

Out[16]:
```
array([[1, 0, 5],
       [1, 0, 5],
       [1, 0, 5],
       [4, 0, 5],
       [6, 8, 2]], dtype=int8)
```

In [17]:
```
y = np.asarray(balanced_df['SEVERITYCODE'])
y[0:5]
```

Out[17]:
```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [18]:
```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

Out[18]:
```
array([[-0.71907961, -0.69272349,  0.39316776],
       [-0.71907961, -0.69272349,  0.39316776],
       [-0.71907961, -0.69272349,  0.39316776],
       [ 0.39080216, -0.69272349,  0.39316776],
       [ 1.13072334,  1.5045195 , -1.43589428]])
```

In [19]:
```
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=45)
```

K-Nearest Neighbors (KNN)

In [20]:
```
from sklearn.neighbors import KNeighborsClassifier
k = 25
knn = KNeighborsClassifier(n_neighbors=k).fit(X_train, y_train)
knn_y_predict = knn.predict(X_test)
knn_y_predict[0:5]
```

Out[20]:
```
array([2, 2, 2, 2, 1], dtype=int64)
```

In [21]:
```
from sklearn.metrics import jaccard_score
jaccard_score(y_test, knn_y_predict)
```

Out[21]:
```
0.29647156115041806
```

In [22]:
```
from sklearn.metrics import f1_score
f1_score(y_test, knn_y_predict, average='macro')
```

Out[22]:
```
0.538428479262381
```

Decision Tree

In [23]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth=7)
dt.fit(X_train, y_train)
dt_y_predict = dt.predict(X_test)
```

In [24]:
```
jaccard_score(y_test, dt_y_predict)
```

Out[24]:
```
0.2573432060020595
```

In [25]:
```
f1_score(y_test, dt_y_predict, average='macro')
```

Out[25]:
```
0.5278396614127413
```

Linear Regression

```
In [26]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix
         lr = LogisticRegression(C=6, solver="liblinear").fit(X_train,y_train)
         lr_y_predict = lr.predict(X_test)

In [27]: jaccard_score(y_test, lr_y_predict)

Out[27]: 0.2896912283837526

In [28]: f1_score(y_test, lr_y_predict, average='macro')

Out[28]: 0.5239236739141877
```

Results and Evaluations

Evaluation metrics used to test the accuracy of our models were jaccard index and f-1 score, which all can be found in Scikit Learn package. Choosing different k, max depth of decision tree and hyparameter C values helped to improve our accuracy to be the best possible. The final results of the model evaluations are summarized in the following table:

Model	Jaccard score	F1 score
KNN	0.296	0.538
Decision tree	0.257	0.528
Linear regression	0.290	0.524

Based on the above table, KNN is the best model to predict car accident severity.

Conclusion

Based on the dataset provided for this capstone from weather, road, and light conditions pointing to certain classes, we can conclude that particular conditions have a somewhat impact on whether or not travel could result in property damage (class 1) or injury (class 2).