By: Antina Sharai Arteaga

September 9, 2020

https://github.com/aarteaga1018/Coursera_Capstone/blob/master/README.md

Coursera_Capstone

This repository will contain assignment information on the capstone.

Seattle Car Accident Severity

Introduction:

For the capstone project, we want to analyze the accident "severity" in of how bad an accident impacts in the city of Seattle and how to prevent future accidents. The data was collected by Seattle SPOT Traffic Management Division which dataset is updated weekly and is from 2004 to present. It contains information such as severity code, location, weather, road conditions, collusion types , address type, speeding, many other attributes.

Background:

Death and injuries caused by road collisions leave a long-lasting impact on millions across the world. In Seattle, 20 people's lives are lost each year on average, and another 150 people are hospitalized with severe injuries from traffic crashes. Every one of these deaths and serious injuries affect not just the people directly involved but their family, friends, and communities. These accidents can be can prevented.

The Seattle government is attempting to prevent avoidable car accidents. In most cases, accidents occur due to not paying enough attention while driving, abusing drugs and alcohol or speeding are the main causes of occurring accidents that can be prevented by enacting harsher regulations. Other reasons they play a part are issues with visibility caused by the weather and road conditions.

The target audience of the project is local Seattle government, police, rescue groups, and car insurance companies. 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 of Seattle.

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 large, we may need to remove the missing values and delete any 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.

Severity codes are as follows:

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

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

Furthermore, because of the existence of null values in some records, the data needs to be preprocessed before any further processing.

Preprocessing Data:

In order to prepare the data, the SEVERITYCODE data was balanced and resampled. Non-relevant columns were dropped, missing values were corrected and encoding of the data was used to clean-up the data for attributes "WEATHER", "ROADCOND", and "LIGHTCOND".

```
In [46]: df1["SEVERITYCODE"].value_counts()

Out[46]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64

In [47]: #balance data and downsample class 1
         df1_maj = df1[df1.SEVERITYCODE==1]
         df1_min = df1[df1.SEVERITYCODE==2]

         df1_dsample = resample(df1_maj,
                                replace = False,
                                n_samples=58188,
                                random_state=123)

         balanced_df1 = pd.concat([df1_dsample, df1_min])
         balanced_df1.SEVERITYCODE.value_counts()

Out[47]: 2    58188
         1    58188
         Name: SEVERITYCODE, dtype: int64
```

```
0]: #encoding the data
    #weather
    from sklearn import preprocessing
    weathercond = preprocessing.LabelEncoder()
    weathercond.fit(df['WEATHER'])
    df['WEATHER'] = weathercond.transform(df['WEATHER'])

    #Road
    from sklearn import preprocessing
    roadcond = preprocessing.LabelEncoder()
    roadcond.fit(df['ROADCOND'])
    df['ROADCOND'] = roadcond.transform(df['ROADCOND'])

    #light
    from sklearn import preprocessing
    light = preprocessing.LabelEncoder()
    light.fit(df['LIGHTCOND'])
    df['LIGHTCOND'] = light.transform(df['LIGHTCOND'])
```

Methodology:

The methodology used includes Github respository and running Jupyter notebook from the IBM watson studio. Packages were used such as Pandas, NumPy, and Sklearn to preprocess data and build machine learning models.

After reviewing the data, Exploratory Data Analysis was used to focus efforts on important features to predict the severity of an accident in Seattle. The features includes "WEATHER", "ROADCOND", and "LIGHTCOND" with SEVERITYCODE being the target variable.

Using Predictive Modeling and Evaluation, decided on using three Machine Learning models with test and train data: KNN (K-Nearest Neighbors), Decision Tree and Linear Regression.

K-Nearest Neighbors:

[67]:
```
# Predictive Modeling & Evaluation
X = df[['WEATHER', 'ROADCOND', 'LIGHTCOND']]
y = df['SEVERITYCODE']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

[68]:
```
print("Train set size")
print(X_train.shape)
print(y_train.shape)
print('')
print("Test set size")
print(X_test.shape)
print(y_test.shape)
```

```
Train set size
(136271, 3)
(136271,)

Test set size
(58402, 3)
(58402,)
```

[69]:
```
k=17
knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
knn
```

[69]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=17, p=2,
           weights='uniform')
```

[70]:
```
knn_pred = knn.predict(X_test)
knn_pred
```

[70]:
```
array([1, 1, 1, ..., 1, 1, 1])
```

Decision Tree:

]:
```
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)
```

]:
```
jaccard_similarity_score(y_test, dt_y_pred)
```

]:
```
0.7035204273826239
```

Linear Regression:

]:
```
#Logistic Regression
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)
```

]:
```
lr_pred = lr.predict(X_test)
lr_pred
```

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

]:
```
jaccard_similarity_score(y_test, lr_pred)
```

]:
```
0.7035889181877333
```

Results & Evaluation:

Results:

Results of three models.

```
knn_pred = knn.predict(X_test)
KNNJaccard = jaccard_similarity_score(y_test, knn_pred)
KNNF1 = f1_score(knn_pred, y_test, average='weighted')
KNNACC = accuracy_score(knn_pred, y_test)
print("Accuracy Score: %.2f" % KNNACC )
print("KNN F1-score: %.2f" % KNNF1 )
print("KNN Jaccard Score: %.2f" % KNNJaccard)

dt_y_pred = dt.predict(X_test)
DTJaccard = jaccard_similarity_score(y_test, dt_y_pred)
DTF1 = f1_score(dt_y_pred, y_test, average='weighted')
DTACC = accuracy_score(dt_y_pred, y_test)
print("Accuracy Score: %.2f" % DTACC)
print("Decision Tree F1-score: %.2f" % DTF1 )
print("Decision Tree Jaccard Score: %.2f" % DTJaccard)

lr_pred = lr.predict(X_test)
LogRJaccard = jaccard_similarity_score(y_test, lr_pred)
LogRF1 = f1_score(lr_pred, y_test, average='weighted')
LogACC = accuracy_score(lr_pred, y_test)
print("Accuracy Score: %.2f" % LogACC)
print("LOG F1-score: %.4f" % LogRF1)
print("LOG Jaccard score: %.4f" % LogRJaccard)
```

```
Accuracy Score: 0.69
KNN F1-score: 0.80
KNN Jaccard Score: 0.69
Accuracy Score: 0.70
Decision Tree F1-score: 0.83
Decision Tree Jaccard Score: 0.70
Accuracy Score: 0.70
LOG F1-score: 0.8260
LOG Jaccard score: 0.7036
```

82]:

Algorithm	Jaccard	F1-score	Accuracy
KNN	0.6949248313413924	0.8015036922197663	0.6949248313413924
Decision Tree	0.7035204273826239	0.8259040882568016	0.7035204273826239
LogisticRegression	0.7035889181877333	0.8260078598494366	0.7035889181877333

The models were very close, but the Linear Regression model had the best results.

Conclusion:

Based on historical data from the collision in Seattle, we can conclude that particular weather, road and light conditions have an impact on whether or not the car ride could result in one of the two classes property damage (class 1) or injury (class 2).