IBM Coursera Capstone: Car Accident Severity Report

Introduction: Business Problem

According to the National Highway Traffic Safety Administration (NHTSA), 36,096 people died in motor vehicle crashes in 2019. In an effort to reduce car accident fatalities, an algorithm must be created to predict the probability of an accident given several parameters: weather, road and visibility conditions. In the future, when these conditions are poor, a service will alert drivers and remind them to drive more carefully.

Data:

- The data was collected by the Seattle Police Department and Accident Traffic Records Department from 2004 to present.

- The data consists of 37 independent variables and 194,673 rows.

- Our predictor variable will be 'SEVERITYCODE' because it is used measure the severity of an accident on a scale from 0 to 5 within the dataset (outlined below). Attributes used to weigh the severity of an accident are 'WEATHER,' the weather conditions, 'ROADCOND,' the road conditions, and 'LIGHTCOND,' the visibility.

The severity code meanings:
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

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

Data Preprocessing

In its current form, the data is not ready for analysis. First, the non-relevant columns must be dropped, as well as the features of the relevant columns need to be converted from object data types to numerical data types.

After my initial overview of the data set, I have decided to focus on only four features: accident severity, weather conditions, road conditions, and light conditions.
In order 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 in order to balance it.

```
[132]: pre_df["SEVERITYCODE"].value_counts()

[132]: 1    136485
       2     58188
       Name: SEVERITYCODE, dtype: int64
```

As displayed above, the number of samples with class 1 severity are nearly 3x larger than the samples with class 2 severity. In order to have a more accurate model, we must reduce the sample size of class 1 severity cases.

```
[134]: from sklearn.utils import resample
       pre_df_maj = pre_df[pre_df.SEVERITYCODE==1]
       pre_df_min = pre_df[pre_df.SEVERITYCODE==2]

       pre_df_maj_dsample = resample(pre_df_maj,
                                     replace=False,
                                     n_samples=58188,
                                     random_state=123)

       balanced_df = pd.concat([pre_df_maj_dsample, pre_df_min])

       balanced_df.SEVERITYCODE.value_counts()

[134]: 2    58188
       1    58188
       Name: SEVERITYCODE, dtype: int64
```

By using the “resample” tool, we are able to equalize the two data sets.

Methodology:

To explore the data and postulate a solution, I will use preprocessing and machine learning models in Jupyter Notebooks, uploaded to a GitHub repository. The coding language I use is Python, in combination with Pandas, Sklearn, and NumPy packages.

After loading the dataset into the Pandas dataframe, I looked at the feature names and data types of the dataset. After an initial evauation I determined that three features would help predict the sevarity of an accident: Weather, Road Conditions, and Light Conditions. As mentioned earlier, “Severity Code” Will be the target variable.

Initially I ran a value count on the three independent variables below,

```
pre_df["LIGHTCOND"].value_counts()
```

```
Daylight                    116137
Dark - Street Lights On      48507
Unknown                      13473
Dusk                          5902
Dawn                          2502
Dark - No Street Lights       1537
Dark - Street Lights Off      1199
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```

```
pre_df["ROADCOND"].value_counts()
```

```
Dry               124510
Wet                47474
Unknown            15078
Ice                 1209
Snow/Slush          1004
Other                132
Standing Water       115
Sand/Mud/Dirt         75
Oil                   64
Name: ROADCOND, dtype: int64
```

```
pre_df["WEATHER"].value_counts()
```

```
Clear                       111135
Raining                      33145
Overcast                     27714
Unknown                      15091
Snowing                        907
Other                          832
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               56
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```

Once the Severity Code features are balanced across the dataset as outlined above, and standardizing the input feature, I utilized three machine learning models:

1. K Nearest Neighbor (KNN)
2. Decision Tree
3. Linear Regression

After importing the necessary packages and splitting the data into train and testing sets, I built and evaluated each model as seen below:

KNN

[*]:
```
## K Nearest Neighbor

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]
```

[110]:
```
# K Nearest Neighbor Evaluation
jaccard_similarity_score(y_test, knn_y_pred)
```

[110]:
```
0.5603643342021597
```

[111]:
```
f1_score(y_test, knn_y_pred, average='macro')
```

[111]:
```
0.5477714681769319
```

Decision Tree

[112]:
```
## Decision Tree

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

dt.fit(X_train,y_train)
```

[112]:
```
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')
```

[113]:
```
dt_y_pred = dt.predict(X_test)
```

[115]:
```
# Decision Tree Evaluation
jaccard_similarity_score(y_test, dt_y_pred)
```

[115]:
```
0.5664365709048206
```

[116]:
```
f1_score(y_test, dt_y_pred, average='macro')
```

[116]:
```
0.5450597937389444
```

Linear Regression

```
## Linear 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_y_pred = LR.predict(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
```

ML Model	Jaccard Score	F1 Score
KNN	0.56	0.55
Decision Tree	0.57	0.55
Linear Regression	0.52	0.51

Based on the machine learning models, Decision Tree is the best model to predict car accident severity.

Conclusion

After using machine learning models to help predict the severity of a car accident, given the weather, road, and visibility conditions, one can see that these conditions can somewhat predict if the relative risk a driver has of having a car accident that could result in property damage (class 1) or injury (class 2).