Seattle Car Accident Severity

Business Understanding

The Seattle government is going to prevent avoidable car accidents by employing methods that alert drivers, health system, and police to remind them to be more careful in critical situations.

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.

Data

We chose the unbalanced dataset provided by the Seattle Department of Transportation Traffic Management Division with 194673 rows (accidents) and 37 columns (features) where each accident is given a severity code. It covers accidents from January 2004 to May 2020. Some of the features in this dataset include and are not limited to Severity code, Location/Address of accident, Weather condition at the incident site, Driver state (whether under influence or not), collision type. Hence we think its a good generalized dataset which will help us in creating an accurate predictive model.

The unbalance with respect to the severity code in the dataset is as follows.

SEVERITY CODE Count

1 — 136485

2 — 58188

Data Preprocessing

The dataset in the original form is not ready for data analysis. In order to prepare the data, first, we need to drop the non-relevant columns. In addition, most of the features are of object data types that need to be converted into numerical data types.

After analyzing the data set, I have decided to focus on only four features, severity, weather conditions, road conditions, and light conditions, among others.

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.

```
In [26]:  1  pre_df["SEVERITYCODE"].value_counts()

Out[26]: 1    136485
         2     58188
         Name: SEVERITYCODE, dtype: int64
```

As you 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 downsampling the class 1.

```
In [27]:
from sklearn.utils import resample

```

```
In [28]:
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()
```

```
Out[28]: 2    58188
         1    58188
         Name: SEVERITYCODE, dtype: int64
```

Methodology

For implementing the solution, I have used Github as a repository and running Jupyter Notebook to preprocess data and build Machine Learning models. Regarding coding, I have used Python and its popular packages such as Pandas, NumPy and Sklearn.

Once I have load data into Pandas Dataframe, used '*dtypes*' attribute to check the feature names and their data types. Then I have selected the most important features to predict the severity of accidents in Seattle. Among all the features, the following features have the most influence in the accuracy of the predictions:

- “WEATHER”,
- “ROADCOND”,
- “LIGHTCOND”

Also, as I mentioned earlier, “SEVERITYCODE” is the target variable.

I have run a value count on road (‘ROADCOND’) and weather condition (‘WEATHER’) to get ideas of the different road and weather conditions. I also have run a value count on light condition (’LIGHTCOND’), to see the breakdowns of accidents occurring during the different light conditions. The results can be seen below:

```
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
```

```
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["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
```

After balancing SEVERITYCODE feature, and standardizing the input feature, the data has been ready for building machine learning models.

I have employed three machine learning models:

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

After importing necessary packages and splitting preprocessed data into test and train sets, for each machine learning model, I have built and evaluated the model and shown the results as follw

KNN

K Nearst Neigbours

```
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], dtype=int64)
```

KNN Evaluation

```
jaccard_score(y_test, knn_y_pred)
```

```
0.3091637411108111
```

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

```
0.5477714681769319
```

Decision Tree

Decision Tree

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

dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=7)
```

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

Decision Tree Evaluation

```python
jaccard_score(y_test, dt_y_pred)
```

```
0.2873687679487783
```

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

```
0.5450597937389444
```

Linear Regression

Linear Regression

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

```python
LR_y_pred = LR.predict(X_test)
```

```python
LR_y_prob = LR.predict_proba(X_test)
```

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

```
0.6849535383198887
```

Linear Regression Evaluation

```python
jaccard_score(y_test, LR_y_pred)
```

```
0.2720073907879108
```

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

```
0.511602093963383
```

Results and Evaluations

The final results of the model evaluations are summarized in the following table:

ML Model	Jaccard Score	F1 Score	Accuracy
KNN	0.30	0.55	0.56
Decision Tree	0.28	0.54	0.57
Linear Regression	0.27	0.51	0.53

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).