Car Accident Severity Analysis: Seattle Washington

(*Applied Data Science Capstone*)

The project is to predict severity of the accidents using different machine learning models

Submitted By: Sakshi Yadav
Date: 19th Sep 20
GitHub: https://github.com/sakshiakr/Coursera_Capstone

1. Introduction: Business Problem

In today's motor vehicle world, an accident risk is always there. Any accident results from a trivial injury to fatality. The severity of accident can be affected by multiple external factors/conditions. It would be very helpful, if somehow we can predict the severity of accident to avoid any unfortunate event.

We have **Seattle car accident data** from 2004 to 2020 to analyse and to provide some advice for the target audience to make insightful decisions for reducing the number of accidents and injuries for the city. Specifically, this report will be targeted to stakeholders interested in preventing avoidable car accidents by employing methods to alert local government, police and car drivers etc.

The target audience

- Local Seattle government
- Police
- Car drivers
- Rescue groups
- Car insurance institutes.

2. Data

The car accident data (i.e. Collisions-data) has been given by the Traffic Records Group in the **SDOT Traffic Management Division from Seattle, WA**. This includes all types of collisions. The collisions will display at the intersection or mid-block of a segment. The data consists of **37** independent variables and **194673** rows. The dependent variable **SEVERITYCODE**, contains numbers that correspond to different levels of severity caused by an accident from 0 to 3.

Severity codes

- **0:** No Probability - Clear Condition
- **1:** Low Probability - Chance of Property Damage
- **2:** Mild Probability - Chance of Injury
- **2b:** Probability - Chance of Serious Injury
- **3:** High Probability - Chance of Fatality

By analysing the provided Seattle car accident data, we have to train a model and it should predict the severity of an accident.

Data Processing

The provided data is not ready for data analysis, right away. So, I have to prepare the data, before 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 close analyses of provided data, I have decided to focus on only four features i.e. severity, weather conditions, road conditions, and light conditions among others. Apart from that, I can see that SEVERITYCODE data is not balanced, so I will use a simple statistical technique to balance it.

```
In [9]: pre_accident_df["SEVERITYCODE"].value_counts()

Out[9]: 1    136485
        2     58188
        Name: SEVERITYCODE, dtype: int64
```

From the above observation, we can deduce that 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 and balance the data.

```
In [10]: pre_accident_df_maj = pre_accident_df[pre_accident_df.SEVERITYCODE == 1]
         pre_accident_df_min = pre_accident_df[pre_accident_df.SEVERITYCODE == 2]

         pre_accident_df_maj_dsample = resample(pre_accident_df_maj,
                                         replace=False,
                                         n_samples=58188,
                                         random_state=123)

         balanced_accident_df = pd.concat([pre_accident_df_maj_dsample, pre_accident_df_min])

         balanced_accident_df.SEVERITYCODE.value_counts()

Out[10]: 2    58188
         1    58188
         Name: SEVERITYCODE, dtype: int64
```

The data for SEVERITYCODE is now balanced as both the classes have similar rows.

3. Methodology

To proceed with the solution, I have used below tools, language and libraries.

- **GitHub** - Version control tool to publish my project related files.
- **Python** - Language to use its popular packages like Pandas, Numpy, Sklearn etc.
- **Jupyter Notebook** - Tool to process data and build Machine Learning tools.

In the first step, I have collected required data and can see that **SEVERITYCODE** is having imbalance data, which needs to be balanced.

Second step in the analysis is exploration of severity of car accidents due to different influencing features like **WEATHER**, **ROADCOND** and **LIGHTCOND**.

In the third step, I will employ different Machine Learning models and try to perfect them to predict the accident severity.

Final step is the result and discussion on different Machine Learning models performance.

4. Analysis and Model Evaluation

After loading the data to Pandas data frame, I have used **dtypes** attribute to check the feature names and their data types. After analysing the outcome, I have figured out the most important features to predict the severity of accidents. Among all the features, the following features have the most influence in the accuracy of the predictions:

- **WEATHER**
- **ROADCOND**
- **LIGHTCOND**

As **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.

I have already balanced **SEVERITYCODE** feature. Also, after standardizing the input feature, now the data has been ready for building different machine learning models. I have run value count on **ROADCOND**, **WEATHER** and **LIGHTCOND** conditions. Below are the results

In [11]:
```
pre_accident_df["WEATHER"].value_counts()
```

Out[11]:
```
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
```

In [12]:
```
pre_accident_df["ROADCOND"].value_counts()
```

Out[12]:
```
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
```

In [13]:
```
pre_accident_df["LIGHTCOND"].value_counts()
```

Out[13]:
```
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
```

Training and Test Data

I have divided the provided Seattle car accident data into training and test dataset.

In [17]:
```
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 ('No. of training set rows:', X_train.shape[0])
print ('No. of test set rows:', X_test.shape[0])
```

```
No. of training set rows: 81463
No. of test set rows: 34913
```

Once the training and test data set are created, I have employed three different machine learning models.

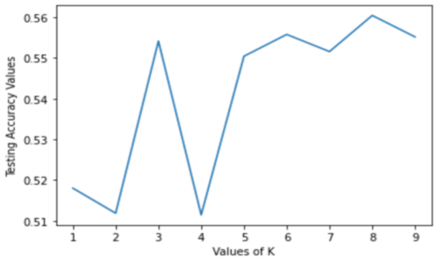
- **K Nearest Neighbour (KNN)**
- **Decision Tree**
- **Logistic Regression**

K Nearest Neighbour (KNN)

In [21]:
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(ks, mean_accuracy)
plt.xlabel('Values of K')
plt.ylabel('Testing Accuracy Values')
```

Out[21]: `Text(0, 0.5, 'Testing Accuracy Values')`



In [22]:
```
# As per above observation, best accuracy got at 8.
k = 8
knn_model = knn(n_neighbors = k)
knn_model.fit(X_train, y_train)
knn_yhat = knn_model.predict(X_test)
knn_yhat[0:5]
```

Out[22]: `array([2, 2, 1, 1, 2])`

KNN Evalution

In [31]:
```
knn_js = jaccard_score(y_test, knn_yhat)
knn_js
```
Out[31]: 0.32384561156150865

In [32]:
```
knn_f1 = f1_score(y_test, knn_yhat, average='macro')
knn_f1
```
Out[32]: 0.5517390361760366

In [33]:
```
knn_ac = accuracy_score(y_test, knn_yhat)
knn_ac
```
Out[33]: 0.5604502620800275

Decision Tree

In [24]:
```
result = pd.DataFrame([jaccard_similarity_score_, f1_score_], index = ['Jaccard', 'F1'], columns = ['d = 1','d = 2','d = 3','d = 4','d = 5','d = 6','d = 7','d = 8','d = 9'])
result.columns.name = 'Evaluation Metrices'
result
```
Out[24]:

Evaluation Metrices	d = 1	d = 2	d = 3	d = 4	d = 5	d = 6	d = 7	d = 8	d = 9
Jaccard	0.102573	0.120534	0.156570	0.156570	0.280563	0.28289	0.287917	0.277237	0.277052
F1	0.436513	0.451957	0.478074	0.478074	0.541223	0.54258	0.545345	0.541370	0.541281

In [25]:
```
# As per above observation, best matrics got at depth 7.
d = 7
dt = DecisionTreeClassifier(criterion = 'gini', max_depth = d)
dt.fit(X_train, y_train)
dt_yhat = dt.predict(X_test)
dt_yhat[0:5]
```
Out[25]: array([2, 2, 1, 1, 2])

Decision Tree Evaluation

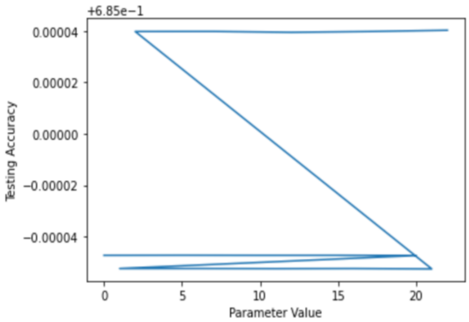
In [34]:
```
dt_js = jaccard_score(y_test, dt_yhat)
dt_js
```
Out[34]: 0.28791725434884813

In [35]:
```
dt_f1 = f1_score(y_test, dt_yhat, average='macro')
dt_f1
```
Out[35]: 0.5450788093962291

In [36]:
```
dt_ac = accuracy_score(y_test, dt_yhat)
dt_ac
```
Out[36]: 0.5661787872712171

Logistic Regression

In [27]:
```
lr_prob = lr_model.predict_proba(X_test)
log_loss(y_test, lr_prob)
plt.plot(index, accuracy_score_)
plt.xlabel('Parameter Value')
plt.ylabel('Testing Accuracy')
```
Out[27]: Text(0, 0.5, 'Testing Accuracy')



In [28]:
```
# As per above observation, best matrics got at 0.001 with sag solver.
lr = LogisticRegression(C = 0.001, solver = 'sag')
lr.fit(X_train, y_train)
lr_yhat = lr.predict(X_test)
lr_yhat[0:5]
```
Out[28]: array([1, 2, 1, 1, 1])

Logistic Regression Evaluation

```
In [37]: lr_js = jaccard_score(y_test, lr_yhat)
         lr_js

Out[37]: 0.2715826546335021

In [38]: lr_f1 = f1_score(y_test, lr_yhat, average='macro')
         lr_f1

Out[38]: 0.5115121139919065

In [39]: lr_ac = accuracy_score(y_test, lr_yhat)
         lr_ac

Out[39]: 0.5260791109328903
```

5. Results and Discussion

The final result of different model evaluations are summarized below:

```
In [44]: columns = ['KNN', 'Decision Tree', 'Logistic Regression']
         index = ['Accuracy Score', 'Jaccard', 'F1-Score', 'Logloss']
         accuracy_df = pd.DataFrame([as_list, js_list, f1_list, ll_list], index = index, columns = columns)
         accuracy_df1 = accuracy_df.transpose()
         accuracy_df1.columns.name = 'Algorithm'
         accuracy_df1
```

Out[44]:

Algorithm	Accuracy Score	Jaccard	F1-Score	Logloss
KNN	0.56	0.32	0.55	NA
Decision Tree	0.57	0.29	0.55	NA
Logistic Regression	0.53	0.27	0.51	0.69

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

Discussion

The most important observation has to be made with the dependent variable like **SEVERITYCODE**. Which indicates that there is no serious injury or fatality occurred. The **SEVERITYCODE** data signal that either somehow the data has been altered at the time of dataset creation or the sample data is incomplete and other important information has been missed from the report. Once the data for **SEVERITYCODE** has been corrected then different Machine Learning models can be trained to predict the severity of the accident.

As we can see that the results from different models are mediocre. We have to build/train the Machine Learning models for different conditions as well like speed of vehicle, mental conditions of driver etc. to predict the severity more accurate.

6. Conclusion

Purpose of this project was to alert the stakeholders about the severity of an accident so that they can take preventive measures accordingly.

We can conclude that particular conditions (i.e. weather, road, light) have a somewhat impact on whether or not travel could result in different severities like No Damage to Fatality.

Final decision on car accident severity will be made by stakeholders based on specific weather, road and light conditions.