Car Accident Severity Report

• Understanding the Problem

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

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. The dependent variable, “SEVERITYCODE”, contains numbers that correspond to different levels of severity caused by an accident from 0 to 4. Severity codes are as follows:

0: Little to no probability (Clear Conditions)
1: Very low probability - Chance of 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

The dataset in the original form in not ready for data analysis. First we check the datatype of every column. After analysing the data I decided to work only on the following four attributes because rest of the attributes are not that relevant to the problem. The following attributes are:

1. SEVERITYCODE
2. ROADCOND
3. LIGHTCOND
4. WEATHER

Now, I have applied value_counts function from pandas to all the four attributes to know about these attributes. After this we notice that all these three attributes (LIGHTCOND, ROADCOND, WEATHER) have significant amount of unknown values.

```
In [6]: df["SEVERITYCODE"].value_counts()

Out[6]: 1    136485
        2     58188
        Name: SEVERITYCODE, dtype: int64

In [7]: df["ROADCOND"].value_counts()

Out[7]: 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 [8]: df["LIGHTCOND"].value_counts()

Out[8]: 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

In [9]: df["WEATHER"].value_counts()

Out[9]: 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
```

Now, I have created a new dataframe including only these four important attributes and removed all the rows from the dataframe which contains unknown in any of these columns.

In [10]:
```
df2 = df[["SEVERITYCODE", "ROADCOND", "LIGHTCOND", "WEATHER"]]
df2.head(5)
```

Out[10]:

	SEVERITYCODE	ROADCOND	LIGHTCOND	WEATHER
0	2	Wet	Daylight	Overcast
1	1	Wet	Dark - Street Lights On	Raining
2	1	Dry	Daylight	Overcast
3	1	Dry	Daylight	Clear
4	2	Wet	Daylight	Raining

In [11]:
```
df2.replace("Unknown", np.nan, inplace=True)
df2.head(20)
```

In [12]:
```
df2.dropna(subset=["LIGHTCOND", "ROADCOND", "WEATHER"], axis=0, inplace=True)

df2.reset_index(drop=True, inplace=True)

df2.head(20)
```

After this I have changed the datatype of these three attributes from object to categorical datatype and resampled the dataset because the target variable (‘SEVERITYCODE’) has unbalanced categorical numbers.

In [13]:
```
df2["SEVERITYCODE"].value_counts()
```

Out[13]:
```
1    114659
2     55851
Name: SEVERITYCODE, dtype: int64
```

In [14]:
```
df2["WEATHER"] = df2["WEATHER"].astype('category')
df2["ROADCOND"] = df2["ROADCOND"].astype('category')
df2["LIGHTCOND"] = df2["LIGHTCOND"].astype('category')
```

In [15]:
```
from sklearn.utils import resample
```

In [16]:
```
df2_maj = df2[df2.SEVERITYCODE==1]
df2_min = df2[df2.SEVERITYCODE==2]

df2_maj_resample = resample(df2_maj, replace=False, n_samples=55851, random_state=123)

df3 = pd.concat([df2_maj_resample, df2_min])
df3.SEVERITYCODE.value_counts()
```

Out[16]:
```
2    55851
1    55851
Name: SEVERITYCODE, dtype: int64
```

Now, I have added 3 new columns (LIGHTCOND_NUM, WEATHER_NUM , ROADCOND_NUM) in the dataframe which contains numerical value for each category in these columns. After this I have created an array X and y to preprocess the data.

- Methodology

After importing all the evaluation scores from sklearn.metrics and splitting the data into train and test sets. As this is a case of classification, I have applied the following classification algorithms:

1. KNN
2. Decision Tree
3. Logistic Regression

- KNN

Applying KNN algorithm after finding the best K and then evaluating accuracy scores.

```
In [30]: #Best k is 18
         k = 18
         knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)

         knn_y_pred = knn.predict(X_test)
         knn_y_pred[0:5]

Out[30]: array([2, 2, 1, 2, 1])

In [31]: print("KNN Jaccard index: %.2f" % jaccard_similarity_score(y_test, knn_y_pred))
         print("KNN F1-score: %.2f" % f1_score(y_test, knn_y_pred, average='weighted') )

         KNN Jaccard index: 0.51
         KNN F1-score: 0.51
```

- Decision Tree

Applying decision tree algorithm and evaluating the accuracy scores.

```
[32]: from sklearn.tree import DecisionTreeClassifier
      dt = DecisionTreeClassifier(criterion="entropy", max_depth = 7)

      dt.fit(X_train,y_train)
```

```
Out[32]: 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')
```

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

```
[34]: print("DT Jaccard index: %.2f" % jaccard_similarity_score(y_test, dt_y_pred))
      print("DT F1-score: %.2f" % f1_score(y_test, dt_y_pred, average='weighted') )
```

```
DT Jaccard index: 0.52
DT F1-score: 0.47
```

- Logistic Regression

Applying logistic regression and evaluating the scores.

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

```
[43]: LR_y_pred = LR.predict(X_test)
```

```
[44]: LR_y_prob = LR.predict_proba(X_test)
```

```
[45]: print("LR Jaccard index: %.2f" % jaccard_similarity_score(y_test, LR_y_pred))
      print("LR F1-score: %.2f" % f1_score(y_test, LR_y_pred, average='weighted') )
      print("LR LogLoss: %.2f" % log_loss(y_test, LR_y_prob))
```

```
LR Jaccard index: 0.52
LR F1-score: 0.51
LR LogLoss: 0.69
```

- Result

Algorithm	Jaccard	F1-score	LogLoss	Accuracy
KNN	0.51	0.51	NA	0.51
DecisionTree	0.52	0.47	NA	0.52
LogisticRegression	0.52	0.51	0.69	0.51

- Conclusion

After seeing the result we can say that lighting condition, weather, road condition can have an impact on the severity of an accident. From the result we can see that decision tree is the most accurate but not by much. By seeing this the government of Seattle can put various safety measures which can help in preventing accidents by judging these three parameters.