Predicting the Car Accident Severity

1. Introduction

1.1 Background

In most cases, carelessness while driving, using drugs and alcohol or driving too fast are some of the main causes of accidents that can be avoided by implementing stronger regulations.

Besides the above reasons, weather, visibility or road conditions are the major uncontrollable factors which can be avoided by uncovering patterns hidden in the data and declaring a warning to local government, police and drivers on the roads. targeted routes or alerting the drivers before the road trips.

1.2 Problem Description

In order to reduce the frequency of car crashes in the community, an algorithm must be developed to predict accident severity given the current weather, road and light conditions.

The main goal of this study is to prevent accidents when conditions are bad, that's why this model is meant to alert drivers to remind them to be a little bit more careful, to switch roads or postpone their car rides.

There is another possible targeted audience for this study, the local police, health institutes, insurance companies etc. They can make good use of this model to know when to be fully ready to receive bad news about a specific road, and more importantly take prevention measures to avoid accidents on certain ones.

2. Data Acquisition & Cleaning

2.1 Source of Data

The data used in this project is the example dataset provided in the capstone project, including all types of collisions. Here is the metadata for the dataset.

It concerns the city of Seattle, WA. It is provided by SPD and recorded by Traffic Records, with a time frame of 2004 to Present.

2.2 Data Description

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 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 a huge unbalance in some attributes occurrences and the existence of null values in some records, the data needs to be pre-processed, cleaned and balanced before any further processing.

2.3 How the data will be used to solve the problem

We have to select the most important features to weigh the severity of accidents in Seattle. Among all the features, the following features have the most influence in the accuracy of the predictions:

The ‘WEATHER’, ‘ROADCOND’ and ‘LIGHTCOND’ attributes.

3. Data pre-processing

First step would be to count the missing values of the attribute columns that we are using to weigh the severity of the collision that we are going to study.

- Weather: 5081
- Roadcond: 5012
- Lightcond: 5170

The columns are of type object, with values as shown below:

```
print(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
```

```
print(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
```

```
print(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
```

Next would be to study the balance of the dataset in the SEVERITYCODE column,

```
print(df['SEVERITYCODE'].value_counts())

1     136485
2      58188
Name: SEVERITYCODE, dtype: int64
```

Class 1 has 136485 values whereas Class 2 has 58188 values. So, down sampling must be done to fix the balancing issue.

```
from sklearn.utils import resample

df_maj = df[df.SEVERITYCODE == 1]
df_min = df[df.SEVERITYCODE == 2]

df_sample = resample(df_maj, replace=False, n_samples=58188, random_state=123)
df = pd.concat([df_sample, df_min])

df['SEVERITYCODE'].value_counts()
```

```
0]:  2     58188
     1     58188
     Name: SEVERITYCODE, dtype: int64
```

In order to work with the features as categorical values, the type of the columns have to be changed.

```
df = df.astype({"WEATHER":'category', "ROADCOND":'category', "LIGHTCOND":'category'})
df.head()
```

Now the feature data frame has to be defined for the next steps

```
df["WEATHER_c"] = df["WEATHER"].cat.codes
df["ROADCOND_c"] = df["ROADCOND"].cat.codes
df["LIGHTCOND_c"] = df["LIGHTCOND"].cat.codes
Feature = df[['WEATHER','ROADCOND','LIGHTCOND','WEATHER_c','ROADCOND_c','LIGHTCOND_c']]
X = np.asarray(Feature[['WEATHER_c','ROADCOND_c','LIGHTCOND_c']])
```

```
Feature.head()
```

9]:

	WEATHER	ROADCOND	LIGHTCOND	WEATHER_c	ROADCOND_c	LIGHTCOND_c
25055	Raining	Wet	Dark - Street Lights On	6	8	2
65280	Clear	Dry	Daylight	1	0	5
86292	Unknown	Unknown	Unknown	10	7	8
155111	Clear	Dry	Daylight	1	0	5
64598	Clear	Dry	Daylight	1	0	5

4. Methodology

4.1 Tools and Technologies

- Github Repository
- Jupyter Notebook
- IBM Watson Studio

4.2 Exploratory Data Analysis

For the exploratory data analysis, it was all done in the previous section, where we uncovered flaws in the dataset, unbalance, missing values.

5. Model Building

When it comes to coding, Python and its popular packages such as Pandas, NumPy and Sklearn have been used.

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

5.1 Data Normalization

```
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:]
```

```
/opt/conda/envs/Python36/lib/python3.6/site-package
at64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-package
at64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
6]: array([[ 1.15236718,  1.52797946, -1.21648407],
           [-0.67488   , -0.67084969,  0.42978835],
           [ 2.61416492,  1.25312582,  2.07606076],
           ...,
           [-0.67488   , -0.67084969,  0.42978835],
           [-0.67488   , -0.67084969,  0.42978835],
           [-0.67488   , -0.67084969,  0.97854582]])
```

5.2 Splitting of Data

```
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)
print('Test set shape: ', X_test.shape, y_test.shape)
print('Training set shape: ', X_train.shape, y_train.shape)
```

```
Test set shape:  (34913, 3) (34913,)
Training set shape:  (81463, 3) (81463,)
```

5.3 Classification Models

The following machine learning algorithms have been used:

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

5.3.1 K-Nearest Neighbors (KNN)

```
from sklearn.neighbors import KNeighborsClassifier
k = 24
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh_pred = neigh.predict(X_test)
neigh_pred[0:]
```

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

5.3.2 Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion="entropy", max_depth = 7)
dt.fit(X_train, y_train)
pt = dt.predict(X_test)
pt[0:]
```

```
)]: array([2, 2, 2, ..., 2, 2, 2])
```

5.3.3 Logistic Regression

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train, y_train)
LRpred = LR.predict(X_test)
LRprob = LR.predict_proba(X_test)
LRpred[0:]
```

```
2]: array([2, 2, 2, ..., 2, 2, 2])
```

6. Results & Evaluation

6.1 KNN Results

```
from sklearn.metrics import f1_score, jaccard_similarity_score, log_loss
print("F1-Score of KNN is : ", f1_score(y_test, neigh_pred, average='macro'))
print("Jaccard Score of KNN is : ", jaccard_similarity_score(y_test, neigh_pred))
```

```
F1-Score of KNN is :  0.477320857528878
Jaccard Score of KNN is :  0.5034227938017357
```

6.2 Decision Tree Results

```
print("F1-Score of Decision Tree is : ", f1_score(y_test, pt, average='macro'))
print("Jaccard Score of Decision Tree is : ", jaccard_similarity_score(y_test, pt))
```

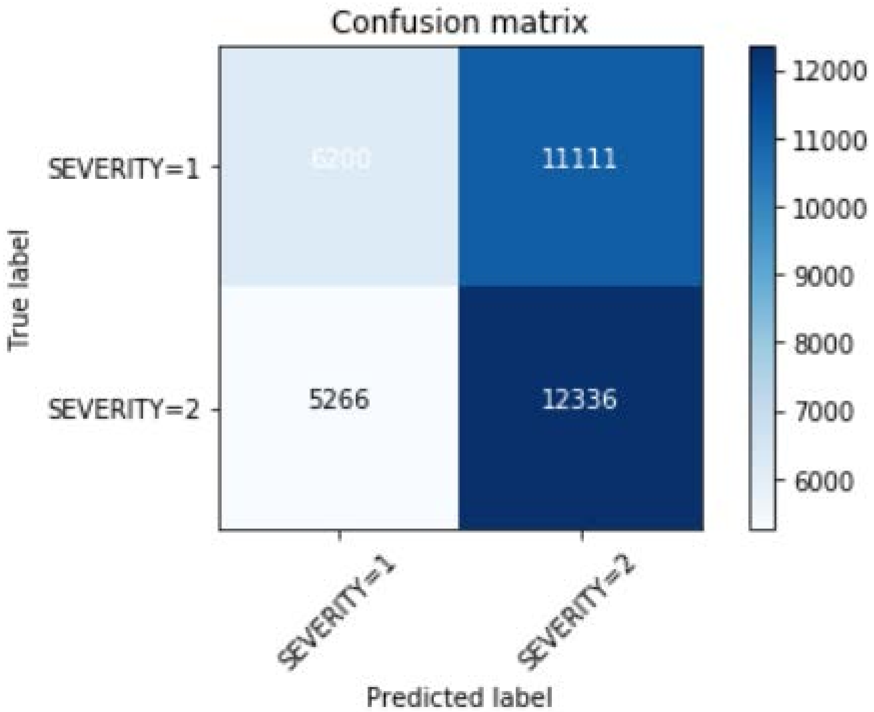
```
F1-Score of Decision Tree is :  0.5461236640287584
Jaccard Score of Decision Tree is :  0.5682410563400453
```

6.3 Logistics Regression Results

```
print("F1-Score of Logistic Regression is : ", f1_score(y_test, LRpred, average='macro'))
print("Jaccard Score of Logistic Regression is : ", jaccard_similarity_score(y_test, LRpred))
print("LogLoss of Logistic Regression is : ", log_loss(y_test, LRprob))
```

```
F1-Score of Logistic Regression is :  0.5159687304684568
Jaccard Score of Logistic Regression is :  0.5309197147194454
LogLoss of Logistic Regression is :  0.6840867853877896
```

6.4 Confusion Matrix



6.5 Classification Report

	precision	recall	f1-score	support
1	0.54	0.36	0.43	17311
2	0.53	0.70	0.60	17602
micro avg	0.53	0.53	0.53	34913
macro avg	0.53	0.53	0.52	34913
weighted avg	0.53	0.53	0.52	34913

6.6 Evaluation

	F1 Score	Jaccard Score	Log Loss
KNN	0.51	0.52	NA
Decision Tree	0.54	0.56	NA
Logistic Regression	0.52	0.53	0.68

Based on the above results, it can be seen that Decision Tree is the best model to predict car accident severity

7. Discussion

At the beginning of this, the dataset had a lot of features of type object, which was converted into columns into category type, then was encoded to contain numerical values.

Once the data was converted to the proper type that would be fed to the models, the problem of unbalanced dataset was seen, which was fixed via resampling the dataset.

Due to the binary aspect of the **'SEVERITYCODE'** attribute, that was to predict (**only classes 1 & 2 were in this dataset**), a **Logistic Regression** model was the first intuitive solution, but along with it, the **K-Nearest Neighbors** and **Decision Tree** models were tested to have more results, contrary to what was expected, the **Decision Tree model had better F1-score and Jaccard-score.**

It can still be improved from the above models, by better tuning of the hyper-parameters like the **"k"** in **KNN**, the **"max_depth"** in the **Decision Tree**, and the **"C"** parameter in the **Logistic Regression.**

8. 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).