CAPSTONE PROJECT - CAR ACCIDENT SEVERITY

BY KHUSHI THAKKAR

TABLE OF CONTENTS

1. INTRODUCTION

For the final capstone project in the IBM certificate course, we want to analyse the accident "severity" in terms of human fatality, traffic delay, property damage, or any other type of accident bad impact. The data was collected by Seattle SPOT Traffic Management Division and provided by Coursera via a link. This dataset is updated weekly and is from 2004 to present. It contains information such as severity code, address type, location, collision type, weather, road condition, speeding, among others.

Target Audience: The target audiences of this study are those people who really care about the traffic records, especially in the transportation department. Also, we want to figure out the reason for collisions and help to reduce accidents in the future.

2. DATA

There are 194,673 observations and 38 variables in this data set. Since we would like to identify the factors that cause the accident and the level of severity, we will use SEVERITYCODE as our dependent variable Y, and try different combinations of independent variables X to get the result.
Since the observations are quite large, we may need to filter out the missing value and delete the unrelated columns first. Then we can select the factor which may have more impact on the accidents, such as address type, weather, road condition, and light condition. The target Data to be predicted under (SEVERITYCODE 1-prop damage 2-injury) label.

Other important variables include:

- ADDRTYPE: Collision address type: Alley, Block, Intersection
- LOCATION: Description of the general location of the collision
- PERSONCOUNT: The total number of people involved in the collision helps identify severity involved
- PEDCOUNT: The number of pedestrians involved in the collision helps identify severity involved
- PEDCYLCOUNT: The number of bicycles involved in the collision helps identify severity involved
- VEHCOUNT: The number of vehicles involved in the collision identify severity involved
- JUNCTIONTYPE: Category of junction at which collision took place helps identify where most collisions occur
- WEATHER: A description of the weather conditions during the time of the collision
- ROADCOND: The condition of the road during the collision

- LIGHTCOND: The light conditions during the collision
- SPEEDING: Whether or not speeding was a factor in the collision (Y/N)
- SEGLANEKEY: A key for the lane segment in which the collision occurred
- CROSSWALKKEY: A key for the crosswalk at which the collision occurred
- HITPARKEDCAR: Whether or not the collision involved hitting a parked car

3. METHODOLOGY

We used Jupyter Notebook to do the data analysis. To generate the table and graph for the dataset, we imported Python libraries (Pandas, Numpy, Matplotlib, and Seaborn).

First, we imported the data through pd.read_csv. We noticed that it had 194,673 rows and 38 columns. Therefore, we narrowed it down to 8 columns ('Severity', 'X', 'Y', 'Location', 'Vehcount', 'Weather', 'Roadcond', 'Lighdcond') and delete the missing values, which made the final dataset with 184,167 observations and 8 variables.

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.

Pre-processing

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 analysing 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.

```
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()
```

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

Next, 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.

```
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
```

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
- Logistic Regression

After importing necessary packages and splitting pre-processed data into test and train sets, for each machine learning model, I have built and evaluated the model.

Tools and Technologies

To help implement the solution, I have used a **Github repository** and running **Jupyter Notebook** from the **IBM WATSON Studio** to pre-process data and build Machine Learning models.

Model Development

When it comes to coding, I have used Python and its popular packages such as Pandas, NumPy and Sklearn. After balancing SEVERITYCODE feature, and standardizing the input feature, the data has been ready for building machine learning models.

Data Normalisation:

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

```
/opt/conda/envs/Python36/lib/python3.6/site-packages
dation.py:595: DataConversionWarning: Data with inpu
nverted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages
dation.py:595: DataConversionWarning: Data with inpu
nverted to float64 by StandardScaler.
  warnings.warn(msg, DataConversionWarning)
```

```
9]: 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]])
```

Train and Test

```
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,)
```

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:]
```

```
2]: array([2, 2, 1, ..., 2, 2, 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:]
```

```
4]: array([2, 2, 1, ..., 2, 2, 2])
```

Logistic Regression

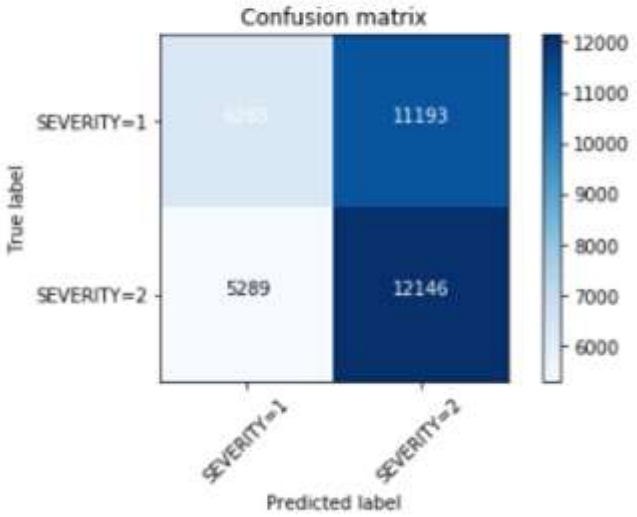
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train, y_train)
LRpred = LR.predict(X_test)
LRprob = LR.predict_proba(X_test)
LRpred[0:]
```

```
0]: array([1, 2, 2, ..., 2, 2, 2])
```

4. RESULTS

Confusion Matrix:

```
Confusion matrix, without normalization
[[ 6285 11193]
 [ 5289 12146]]
```



Classification Report:

```
print (classification_report(y_test, LRpred))
```

	precision	recall	f1-score	support
1	0.54	0.36	0.43	17478
2	0.52	0.70	0.60	17435
micro avg	0.53	0.53	0.53	34913
macro avg	0.53	0.53	0.51	34913
weighted avg	0.53	0.53	0.51	34913

Comparison:

	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

5. OBSERVATIONS

At the beginning of this, we had a dataset that had a lot of features of type object, we did convert the columns into category type, then we encoded the labels to contain numerical values.

Once we converted our data to the proper type that can be fed to our models, we faced the problem of unbalanced dataset, we fixed this issue through resampling the dataset.

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

We can still improve 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.

6. 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).