Applied Data Science Capstone Project –

on

Car accident severity

Final Report

By

Shengyu Wu

Table of Contents

1. Introduction

2. Data

3. Methodology

4. Results

5. Discussion

6. Conclusion

1.Introduction

Car accidents have a significant impact on individuals, their families and the nation. It is always one of the top issues in society. According to NSC, in US, an estimated 38,800 people lost their lives to car crashes in 2019 – a 2% decline from 2018 (39,404 deaths) and a 4% decline from 2017 (40,231 deaths). About 4.4 million people were injured seriously enough to require medical attention in crashes last year. Therefore, if there was an algorithm that can predict severity of car accidents, it could be efficient and faster for police to arrive the accident scene and give right help.

This project is attempting to classify varies factors that will cause the accidents and predict the level of severity by leveraging data collected from different kinds of car accidents.

2.Data

For this dataset, we can see there are 38 different attributes, some of them are relatively not important to analyze the car accident severity. As a result, we drop them to emphasize the main factors. For data cleaning, miss values will be replaced by 'other' or 'unknown' based on the information in each column.

The dataset we used can be found at https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

After loading data from csv file to data frame, we do main feature selection:

]:
```
df1 = df.drop([ "SEVERITYCODE.1", "OBJECTID", "INCKEY", "COLDETKEY", "REPORTNO","STATUS", "ADDRTYPE", "INTKEY",
         "EXCEPTRSNCODE","EXCEPTRSNDESC", "PEDCYLCOUNT", "PEDCOUNT", "SDOT_COLCODE", "SDOT_COLDESC", "ST_COLCODE", "SEGLANEKEY", "CROSSWALKKEY", "SDOTCOLNU
          "INCDATE", "INCDTTM", "PEDROWNOTGRNT", "UNDERINFL", "HITPARKEDCAR", "ST_COLDESC", "SEVERITYDESC"], axis=1)
df1.head()
```

Out[6]:

	SEVERITYCODE	X	Y	LOCATION	COLLISIONTYPE	PERSONCOUNT	VEHCOUNT	JUNCTIONTYPE	INATTENTIONIND	WEATHER	ROADCOND	LIGH
0	2	-122.323148	47.703140	5TH AVE NE AND NE 103RD ST	Angles	2	2	At Intersection (intersection related)	NaN	Overcast	Wet	
1	1	-122.347294	47.647172	AURORA BR BETWEEN RAYE ST AND BRIDGE WAY N	Sideswipe	2	2	Mid-Block (not related to intersection)	NaN	Raining	Wet	Dark L
2	1	-122.334540	47.607871	4TH AVE BETWEEN SENECA ST AND UNIVERSITY ST	Parked Car	4	3	Mid-Block (not related to intersection)	NaN	Overcast	Dry	
3	1	-122.334803	47.604803	2ND AVE BETWEEN MARION ST AND MADISON ST	Other	3	3	Mid-Block (not related to intersection)	NaN	Clear	Dry	
4	2	-122.306426	47.545739	SWIFT AVE S AND SWIFT AV OFF RP	Angles	2	2	At Intersection (intersection related)	NaN	Raining	Wet	

Then we observed that there are too many missing value in each columns, so we replace them for data cleaning:

```
]: df1['ROADCOND'].replace(np.NaN, "Unknown", inplace=True)
```

```
]: df1['LIGHTCOND'].replace(np.NaN, "Unknown", inplace=True)
```

```
]: df1['SPEEDING'].replace(np.NaN, "N", inplace=True)
```

```
]: avg_X = df1["X"].astype("float").mean(axis=0)
   df1['X'].replace(np.NaN, avg_X, inplace=True)
```

```
]: avg_Y = df1["Y"].astype("float").mean(axis=0)
   df1['Y'].replace(np.NaN, avg_Y, inplace=True)
```

```
]: df1['LOCATION'].replace(np.NaN, "Unknown", inplace=True)
```

```
]: df1.isna().sum()
```

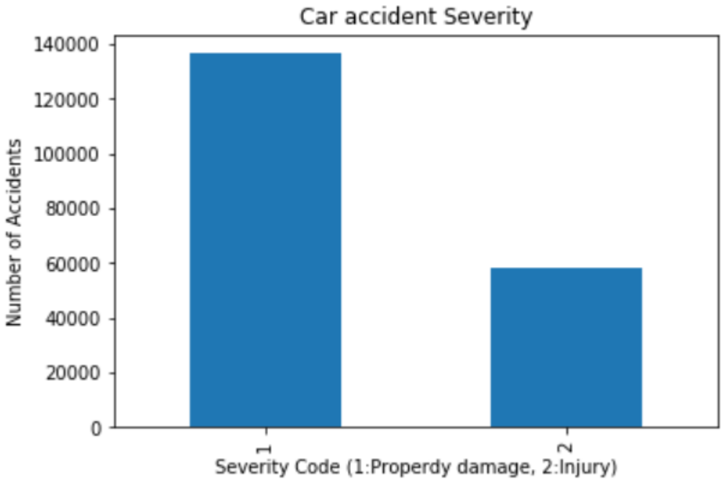
```
Out[25]: SEVERITYCODE      0
         X                 0
         Y                 0
         LOCATION          0
         COLLISIONTYPE     0
         PERSONCOUNT       0
         VEHCOUNT          0
         JUNCTIONTYPE      0
         INATTENTIONIND    0
         WEATHER           0
         ROADCOND          0
         LIGHTCOND         0
         SPEEDING          0
         dtype: int64
```

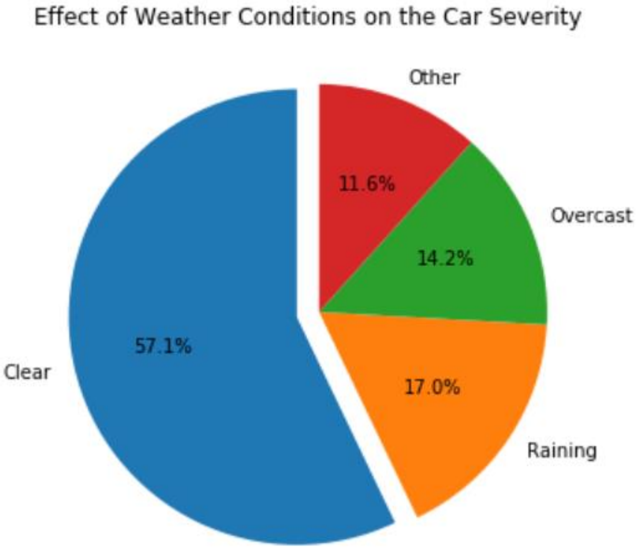
3.Methodology

In this approach, supervised learning wll be used to map inputs to outputs as the dataset is labelled. The question we defined is "Are there any factors having bigger effect on causing the car accident and increasing the severity?" Since this is a Yes/No question, we will use 3 classification algorithms to create machine learning models:

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

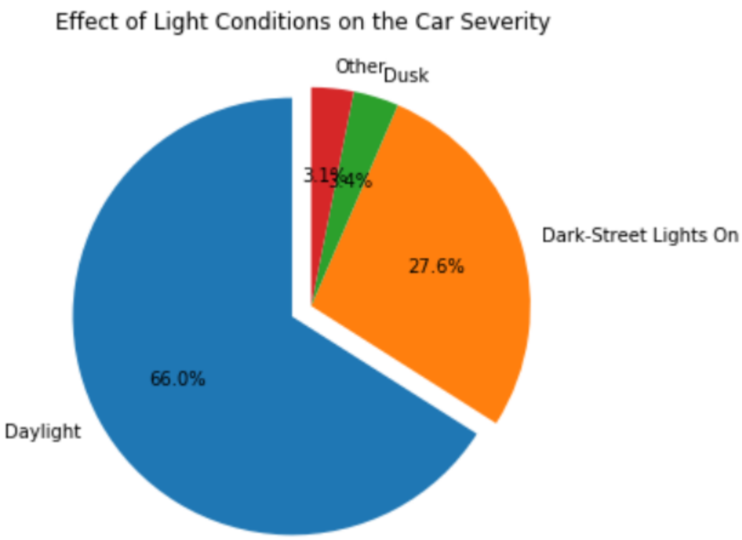
For data visualization, we plot bar chart for different types of accident and pie charts for relationships between relative features and the car severity.





6]:
```
df_w = df1['WEATHER'].value_counts()
df_w
```

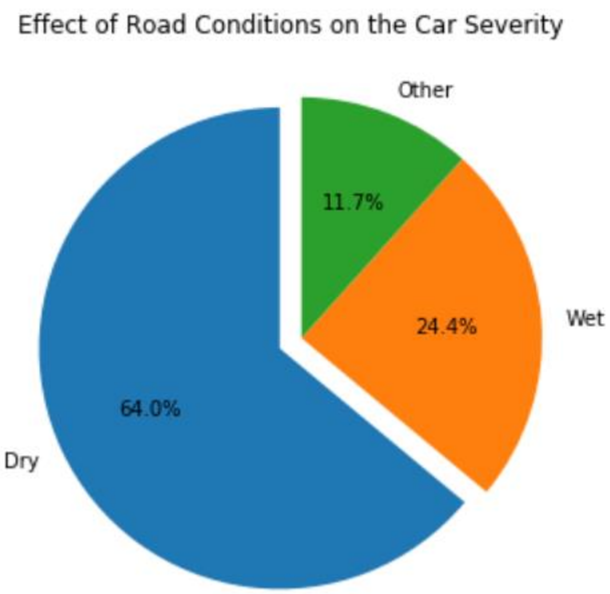
Out[26]:
```
Clear                       111135
Raining                      33145
Overcast                     27714
Unknown                      20172
Snowing                        907
Other                          832
Fog/Smog/Smoke                 569
Sleet/Hail/Freezing Rain       113
Blowing Sand/Dirt               56
Severe Crosswind                25
Partly Cloudy                    5
Name: WEATHER, dtype: int64
```




```
df_l = df1['LIGHTCOND'].value_counts()
df_l
```

Out[28]:

```
Daylight                    116137
Dark - Street Lights On      48507
Unknown                      18643
Dusk                          5902
Dawn                          2502
Dark - No Street Lights       1537
Dark - Street Lights Off      1199
Other                          235
Dark - Unknown Lighting         11
Name: LIGHTCOND, dtype: int64
```



]:
```
df_r = df1['ROADCOND'].value_counts()
df_r
```

Out[31]:
```
Dry               124510
Wet                47474
Other              20222
Ice                 1209
Snow/Slush          1004
Standing Water       115
Sand/Mud/Dirt         75
Oil                   64
Name: ROADCOND, dtype: int64
```

3.Results

Import scikit-learn library, then we can use an evaluation approach called Train/Test Split.

]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (155738, 11) (155738,)
Test set: (38935, 11) (38935,)
```

Logistic Regression

5]:
```
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

Out[45]:
```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
                   tol=0.0001, verbose=0, warm_start=False)
```

5]:
```
yhat = LR.predict(X_test)
print("Logistic Regresion's Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

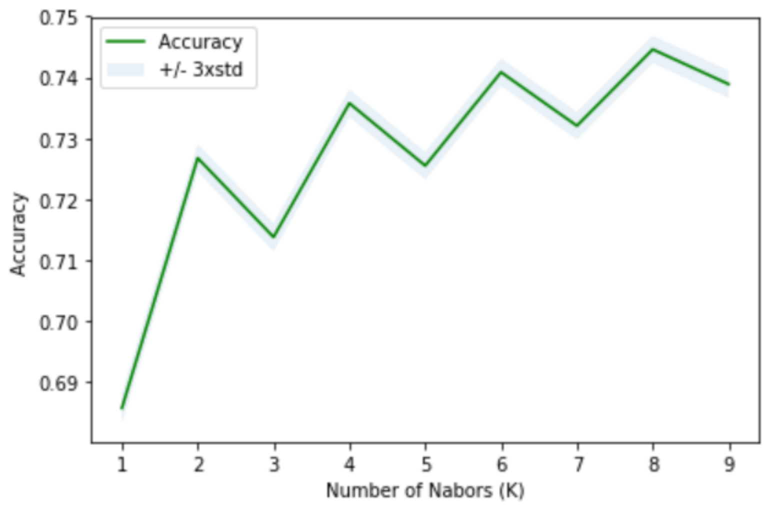
```
Logistic Regresion's Accuracy:  0.7358674714267369
```

K nearest neighbor (KNN)

8]:
```
#Finding the best k
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #Train Model and Predict
    kNNeigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat1 = kNNeigh.predict(X_test)


    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat1);

    std_acc[n-1]=np.std(yhat1==y_test)/np.sqrt(yhat1.shape[0])

mean_acc
```

Out[48]:
```
array([0.68573263, 0.72674971, 0.71377938, 0.73573905, 0.7254912 ,
       0.74082445, 0.73204058, 0.74457429, 0.73887248])
```

|:
```
plt.plot(range(1,Ks),mean_acc,'g')
plt.fill_between(range(1,Ks),mean_acc - 1 * std_acc,mean_acc + 1 * std_acc, alpha=0.10)
plt.legend(('Accuracy ', '+/- 3xstd'))
plt.ylabel('Accuracy ')
plt.xlabel('Number of Nabors (K)')
plt.tight_layout()
plt.show()
```



Decision Tree

```
0]:  DTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
     DTree.fit(X_train, y_train)
```

```
Out[50]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
                    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')
```

```
0]:  yhat2 = DTree.predict(X_test)
     print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, yhat2))
```

```
DecisionTrees's Accuracy:  0.7520226017721844
```

4.Discussion

Based on the result, it can be considered that if we incorporate with more factors, the accuracy might be higher. In addition, we observed that there are two types of severity, one of them was property damage and another was injury. Since the data was unbalance between these two types, the result would be different after we balance the dataset or predicting severity based only on accidents that cause injury. During modelling, the processing time of KNN was much longer than the other two algorithms and ended with lowest accuracy, Jaccard and F1 score. So Decision Tree and Logistic Regression should be more suitable for this prediction.

5.Conclusion

In this study, we found that more than half of car accidents happen on dry road during clear weather days under daylight, which was the most common condition in daily life. Other than that, more than 15 percent of accidents happen in rainy days on wet and dark street with lights on. Explanatory data analysis has done to give insight into the relationship between the features and the severity of the accidents. For improving the accuracy of this machine learning process, we could collect more data on speeding and add more features that should be relevant to the accidents such as age of driver, years of driving license and so on.