Capstone Project

Car accident severity

Business Problem

Road traffic accidents are a serious problem in our societies around the world. In most cases, insufficient attention while driving, drug and alcohol abuse or driving at very high speeds are the main causes of accidents, which can be prevented by adopting stricter regulations. In addition to the abovementioned reasons, weather, visibility or road conditions are the main uncontrollable factors that can be prevented by discovering hidden patterns in the data and issuing a warning to local authorities, police and drivers on targeted roads.

The predictive analysis performed here aims to analyze the severity of an accident / collision based on road conditions, lighting conditions, collision area, number of people involved, and many other factors as such. Knowing in advance the severity of any such collision will help prevent and take immediate action.

Data

The data for this project is taken from open source and it belongs to SDOT Traffic Management Division, Traffic Records Group. This dataset includes all types of road collisions. Road collisions will be displayed at the intersection or in the middle of the segment. The information on incidents contains data from 2004 till 2018. All collisions are provided of the Seattle Police Department and registered by Traffic Records.

There is total of 40 data columns in the dataset, including the 3 target columns. We will consider various aspects when deciding the importance of the particular column or the transformation that might be required before we introduce it into the model. The dependent variable "SEVERITYCODE" contains codes that correspond to different severity levels.

Level	Cases
1	137776
2	58842
0	21656
2b	3111
3	352

0: Unknown
1: Property Damage Only Collision
2: Injury Collision
2b: Serious Injury Collision
3: Fatality Collision

Property Damage Only Collision	137776
Injury Collision	58842
Unknown	21657
Serious Injury Collision	3111
Fatality Collision	352

In addition, due to the existence of null values in some records, the data must be pre-processed before any further processing.

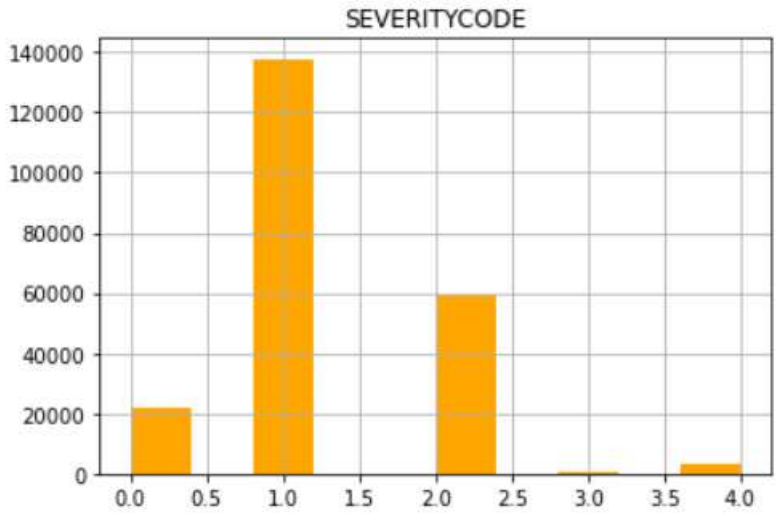
The dataset in its original form is not ready for direct data analysis. To prepare the data, we first need to remove the unused columns. Also, most of the functionality is related to the data types of the object, which must be converted to numeric data types.

```
1      137776
2       58842
0       21656
2b       3111
3         352
Name: SEVERITYCODE, dtype: int64
```

We also need to convert the SEVERITYCODE of our target variable to a numeric data type.

To get a good idea of the dataset, various values in the functions are checked. The results show that the target function is imbalance, so a simple statistical method is used to balance it.

The number of rows in class 1 is almost three times as many as the number of rows in class 2. The problem can be solved by down sampling class 1.



By sampling data for class 1, we get a balanced objective function. Although the overall record size has decreased, the built model will not be biased.

```
1.0    65000
2.0    58842
0.0    21656
4.0     3111
3.0      352
Name: SEVERITYCODE, dtype: int64
```

Methodology

To implement the solution, have been used Jupyter Notebook at IBM Skills Network Labs for data preprocessing and building machine learning models. In terms of coding, Python and its popular packages Pandas, NumPy and Sklearn have been used. Once the data was loaded into the Pandas Dataframe, the "dtypes" attribute was used to check the function names and their data types. The most important characteristics are then selected for predicting the severity of accidents in Seattle. Among all the characteristics, the following features have the greatest impact on the accuracy of predictions:

- COLLISIONTYPE
- PERSONCOUNT
- PEDCOUNT
- PEDCYLCOUNT
- VEHCOUNT
- INJURIES
- SERIOUSINJURIES
- FATALITIES
- JUNCTIONTYPE
- WEATHER
- ROADCOND
- LIGHTCOND
- SPEEDING

Also, as mentioned earlier, "SEVERITYCODE" is the target variable.

The road ("ROADCOND") and weather ("WEATHER") values were calculated in order to get an idea of the different road and weather conditions. Count values of lighting conditions ('LIGHTCOND') is also conducted to see the number of accidents occurring under different light conditions. Values were calculated for other attributes in a similar way.

The results of some of them can be seen below:

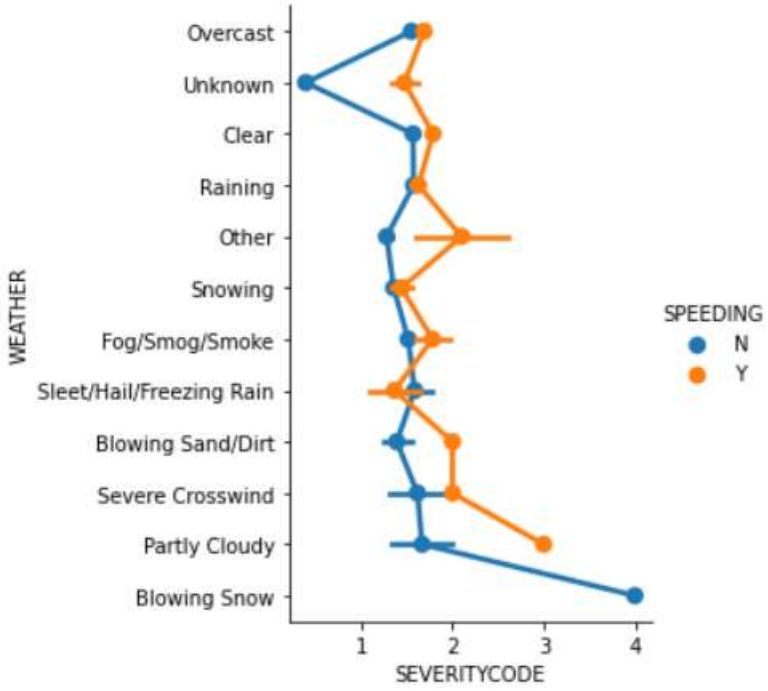
```
Clear                       74375
Raining                     22413
Overcast                    18489
Unknown                      7628
Snowing                       541
Other                         470
Fog/Smog/Smoke                374
Sleet/Hail/Freezing Rain       64
Blowing Sand/Dirt              37
Severe Crosswind               14
Partly Cloudy                   7
Blowing Snow                    1
Name: WEATHER  dtype: int64
```

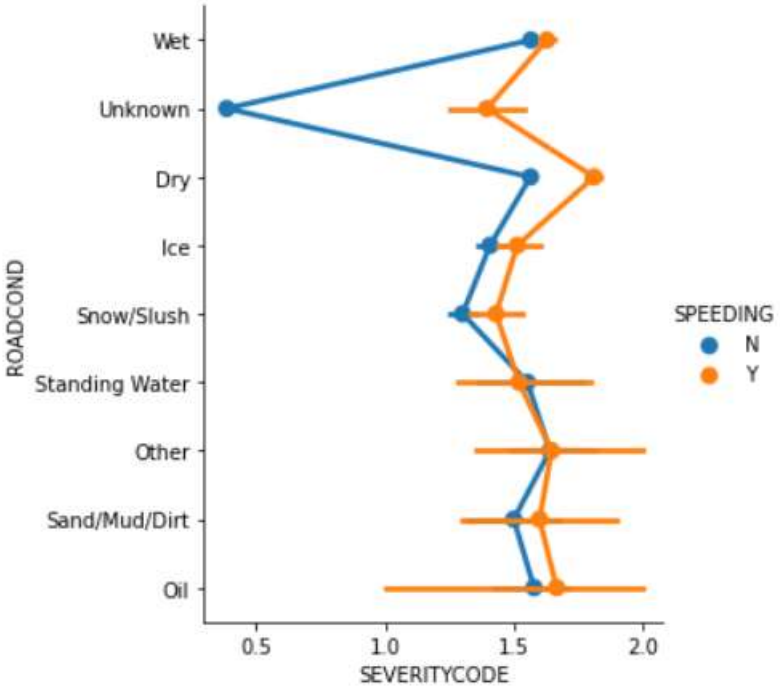
```
Dry               83334
Wet               31983
Unknown            7579
Ice                 755
Snow/Slush          584
Other                81
Standing Water       72
Sand/Mud/Dirt        44
Oil                  41
Name: ROADCOND  dtype: int64
```

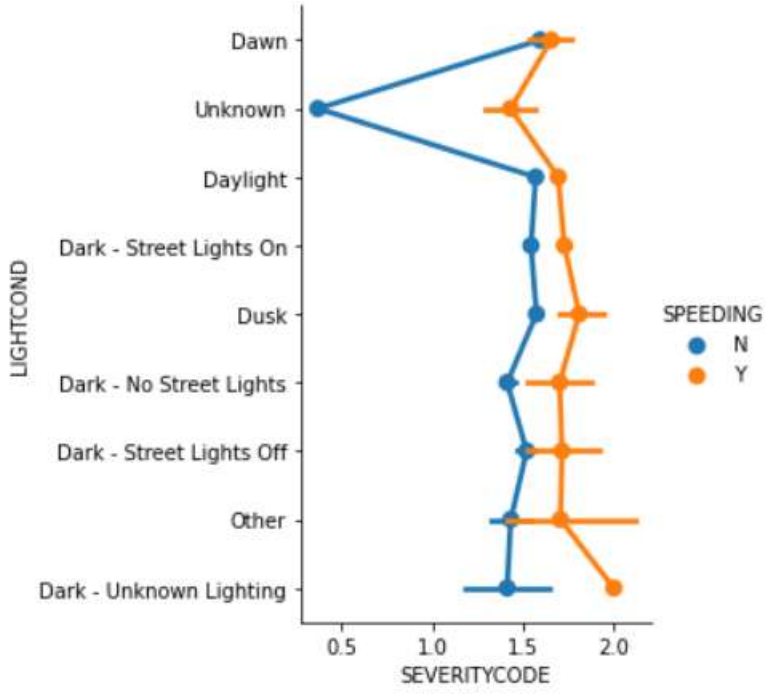
```
Daylight                    77920
Dark - Street Lights On     32090
Unknown                     6805
Dusk                        3964
Dawn                        1718
Dark - No Street Lights      951
Dark - Street Lights Off     764
Other                        142
Dark - Unknown Lighting       18
Name: LIGHTCOND  dtype: int64
```

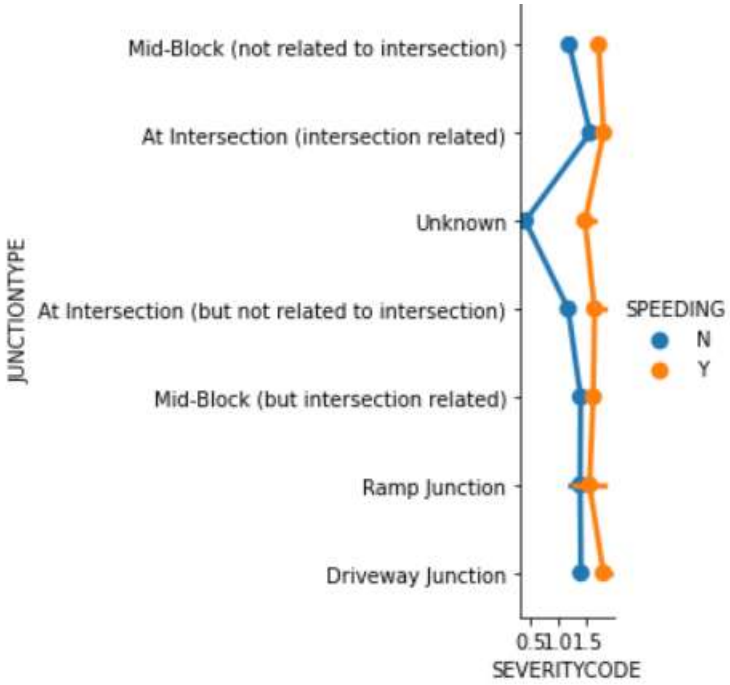
```
Mid-Block (not related to intersection)              64154
At Intersection (intersection related)               50311
Mid-Block (but intersection related)                 16294
Driveway Junction                                     7493
At Intersection (but not related to intersection)     1739
Ramp Junction                                          127
Unknown                                                 17
Name: JUNCTIONTYPE, dtype: int64
```

The below graphs show severity levels of accidents basis the different attributes like WEATHER, ROADCOND, LIGHTCOND and JUNCTIONTYPE




Wet
Unknown
Dry
Ice
Snow/Slush
Standing Water
Other
Sand/Mud/Dirt
Oil
ROADCOND
0.5
1.0
1.5
2.0
SEVERITYCODE
SPEEDING
N
Y


Dawn
Unknown
Daylight
Dark - Street Lights On
Dusk
Dark - No Street Lights
Dark - Street Lights Off
Other
Dark - Unknown Lighting
LIGHTCOND
0.5
1.0
1.5
2.0
SEVERITYCODE
SPEEDING
N
Y



Data Pre-processing

Using **df_car** as the final data after removing extra fields, declare the following variables:

- **X** as the **Feature Matrix** (data of df_car)
- **y** as the **response vector (SEVERITYCODE)**

As you may figure out, some features in this dataset are categorical such as **WEATHER, ROADCOND, LIGHTCOND,** etc. Unfortunately; Sklearn Decision Trees do not handle categorical variables. But still we can convert these features to numerical values. **pandas.get_dummies()** Convert categorical variable into dummy/indicator variables.

Normalize Data

Standardizing the data gives zero mean and unit variance, which is good practice, especially for algorithms such as KNN, which are based on observation distance.

Train Test Split

Out of Sample Accuracy is the percentage of correct predictions a model makes for data that the model has NOT been trained on. Performing training and testing on the same dataset is likely to have poor out-of-sample precision due to the likelihood of over-matching.

It is important that our models have high out-of-sample fidelity, because the goal of any model is, of course, to make correct predictions on unknown data. So how can we improve out-of-sample precision? One way is to use a scoring method called Train / Test split. Training / testing splitting involves splitting the dataset into training and test sets, respectively, which are mutually exclusive. Then you train with the training kit and test with the test kit.

MODELING

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 Neighbor (KNN)

- Decision Tree
- Logistic Regression

After importing necessary packages and splitting preprocessed data into test and train sets, for each machine learning model.

K Nearest Neighbor (KNN)

```
from sklearn.neighbors import KNeighborsClassifier

k = 4
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=4, p=2,
           weights='uniform')
```

```
yhat = neigh.predict(X_test)
yhat[0:5]
```

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

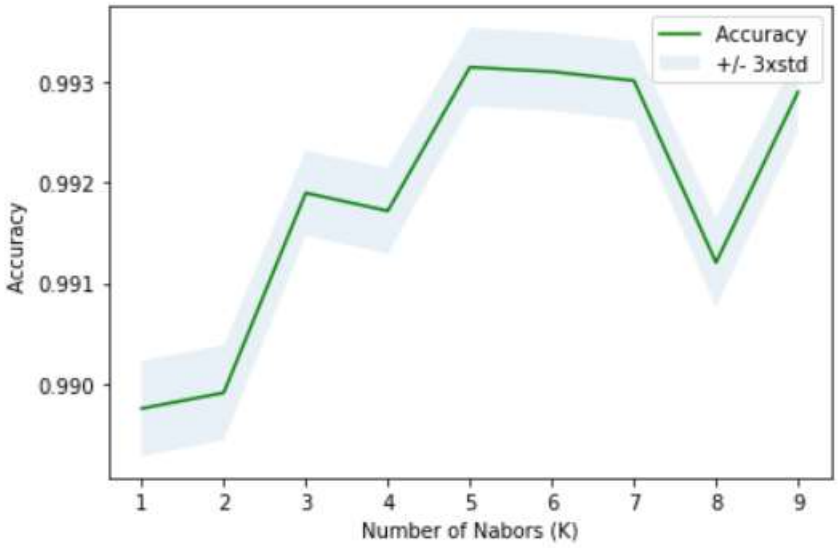
```
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))
```

```
Train set Accuracy:  0.9912057771426928
Test set Accuracy:  0.9917205576316319
```

```
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
    neigh = KNeighborsClassifier(n_neighbors = n).fit(X_train,y_train)
    yhat=neigh.predict(X_test)
    mean_acc[n-1] = metrics.accuracy_score(y_test, yhat)


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

mean_acc
```

```
array([0.98975139, 0.98990803, 0.99189957, 0.99172056, 0.99315268,
       0.99310792, 0.99301842, 0.99120589, 0.99290653])
```

Accuracy graph for Different number of Neighbors



The best accuracy 0.993152677392647 with k= 5. So we rebuild the model with best value for K=5.

Results and Evaluations

For the final forecast, we must preprocess the entire set of test data. When coding the function columns, we made sure that the one-time encodings are the same as in the train set, and the function hash transformer used must be set on the train data.

Below are the final results from the test data:

ML Model	F1-Score	Accuracy
KNN	0.99	0.993
Decision Tree	0.99	0.993
Logistic Regression	0.99	0.988

Conclusion

Based on the dataset provided for this capstone from weather, road, light conditions, etc. pointing to certain classes, we can conclude that particular conditions have a somewhat impact on whether or not travel could result in property damage or injury.

In conclusion, based on the work done below, the data set cannot tell us the possibility of getting into a car accident, but can tell us that if we are in an accident in certain conditions how severe it would be.