Applied Data Science Capstone (Seattle City Car Accident Analysis)

PREPARED BY MOHAMMED ALAWI

Introduction

In an attempt to build a model that can predict the accident, an algorithm was developed. The model aims to interpret the severity of a car collision in Seattle city. Several parameters were employed, such as weather, road, and visibility condition, to predict the accident severity. Undoubtedly, the weather condition somehow influences the number of collisions (for example, a rainy day with high speed could cause the car to drift, leading to an accident). The success of the project will help individuals to be more careful when one of the parameters changes. In the long term, the project can help governmental organizations, car insurance companies, and medical insurance companies to provide solutions based on the prediction of the collision rate (i.e., collisions per year) and the severity code (prop damage or injury).

Data Description

The data used in this project are the collisions for all years of Seattle city, and it is provided below. The data include a vast number of attributes. To build a model, the data was shortened and divided into two categories. The first category comprises of the independent variables, which are Road condition, light-condition, and weather. The second category is the dependent variable or the predictor, which is the severity code. The numbers in the severity code are indications that were put to assess the severity of an accident. The number 1 refers to prop damage while 2 referring to injury.

More description of the data

1. OBJECTID: Unique identifier

2. X: Coordinates

3. Y: Coordinates

4. LOCATION: Coordinates

3. COLLISIONTYPE: Collision type

4. PERSONCOUNT: The number of pedestrians involved in the collision. This is entered by the state.

More description of the data

5. UNDERINFL: Whether or not a driver involved was under the influence of drugs or alcohol.

6. WEATHER: A description of the weather conditions during the time of the collision.

7. ROADCOND: The condition of the road during the collision.

8. LIGHTCOND: The light conditions during the collision.

9. SPEEDING: Whether speeding was a factor in the collision or not. (Y/N)

10. SEVERITYCODE: A code that corresponds to the severity of the collision where 1 refers to prop damage and 2 refers to injury.

Data visualization and pre-processing

In this section the data are going to be pre-processed, investigated and analyzed comprehensively. This step is necessary for Machine Learning (ML) modeling.

Sample of the data before pre-processing and cleaning

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	...	ROADCOND	LIGHTCOND
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	...	Wet	Daylight
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	...	Wet	Dark - Street Lights On
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	...	Dry	Daylight
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	...	Dry	Daylight
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	...	Wet	Daylight

Attributes selection

```
## Select some attributes and attach it to a variable named features
df[['OBJECTID','X','Y','LOCATION','COLLISIONTYPE','PERSONCOUNT','UNDERINFL','WEATHER','ROADCOND','LIGHTCOND','SPEEDING','SEVE
features = df[['OBJECTID','X','Y','LOCATION','COLLISIONTYPE','PERSONCOUNT','UNDERINFL','WEATHER','ROADCOND','LIGHTCOND','SPEE
features.head()
```

8]:

ID	X	Y	LOCATION	COLLISIONTYPE	PERSONCOUNT	UNDERINFL	WEATHER	ROADCOND	LIGHTCOND	SPEEDING	SEVERITYCODE
1	-122.323148	47.703140	5TH AVE NE AND NE 103RD ST	Angles	2	N	Overcast	Wet	Daylight	NaN	2
2	-122.347294	47.647172	AURORA BR BETWEEN RAYE ST AND BRIDGE WAY N	Sideswipe	2	0	Raining	Wet	Dark - Street Lights On	NaN	1
3	-122.334540	47.607871	4TH AVE BETWEEN SENECA ST AND UNIVERSITY ST	Parked Car	4	0	Overcast	Dry	Daylight	NaN	1
4	-122.334803	47.604803	2ND AVE BETWEEN MARION ST AND MADISON ST	Other	3	N	Clear	Dry	Daylight	NaN	1
5	-122.306426	47.545739	SWIFT AVE S AND SWIFT AV OFF RP	Angles	2	0	Raining	Wet	Daylight	NaN	2

Balancing the data

```
##Balance the unbalanced data to avoid bias ML model
from sklearn.utils import resample

n=58188

features_cl1 = features[features.SEVERITYCODE==1]
features_cl2 = features[features.SEVERITYCODE==2]

features_dsample = resample(features_cl1,
                            replace=False,
                            n_samples=n,
                            random_state=123)
features_balanced = pd.concat([features_dsample, features_cl2])
features_balanced.SEVERITYCODE.value_counts()
features_balanced.head()
```

TID	X	Y	LOCATION	COLLISIONTYPE	PERSONCOUNT	UNDERINFL	WEATHER	ROADCOND	LIGHTCOND	SPEEDING	SEVERITYCODE
626	-122.292730	47.719265	33RD AVE NE AND NE 125TH ST	Angles	2	0	Raining	Wet	Dark - Street Lights On	NaN	1
557	-122.329124	47.608658	8TH AVE AND SPRING ST	Angles	2	0	Clear	Dry	Daylight	NaN	1
574	-122.292434	47.733753	LAKE CITY WAY NE AND NE 145TH ST	Angles	2	N	Unknown	Unknown	Unknown	NaN	1
907	NaN	NaN	ALASKAN WY VI NB BETWEEN SENECA ST OFF RP AND ...	Sideswipe	2	N	Clear	Dry	Daylight	NaN	1
842	-122.339185	47.625594	ROY ST BETWEEN 9TH AVE N AND VALLEY ST	Head On	3	0	Clear	Dry	Daylight	NaN	1

Data types

```
##Type and shape of data in the data frames (features, balanced features) and
print(features.dtypes)
print(features_balanced.dtypes)
print(features.shape)
print(features_balanced.shape)
```

```
OBJECTID          int64
X               float64
Y               float64
LOCATION         object
COLLISIONTYPE    object
PERSONCOUNT       int64
UNDERINFL        object
WEATHER          object
ROADCOND         object
LIGHTCOND        object
SPEEDING         object
SEVERITYCODE      int64
dtype: object
OBJECTID          int64
X               float64
Y               float64
LOCATION         object
COLLISIONTYPE    object
PERSONCOUNT       int64
UNDERINFL        object
WEATHER          object
ROADCOND         object
LIGHTCOND        object
SPEEDING         object
SEVERITYCODE      int64
dtype: object
(194673, 12)
(116376, 12)
```

Empty cells were replaced with nan

In [17]:

```
import numpy as np

# replace "?" to NaN
features_balanced.replace("?", np.nan, inplace = True)
features_balanced.head(5)
```

65280	71557	-122.329124	47.608658	8TH AVE AND SPRING ST	Angles	2	0	Clear	Dry	Daylight	Na
86292	94574	-122.292434	47.733753	LAKE CITY WAY NE AND NE 145TH ST	Angles	2	N	Unknown	Unknown	Unknown	Na
155111	172907	NaN	NaN	ALASKAN WY VI NB BETWEEN SENECA ST OFF RP AND ...	Sideswipe	2	N	Clear	Dry	Daylight	Na
64598	70842	-122.339185	47.625594	ROY ST BETWEEN 9TH AVE N AND VALLEY ST	Head On	3	0	Clear	Dry	Daylight	Na

Evaluation of missing values

Evaluating for Missing Data

The missing values are converted to Python's default. We use Python's built-in functions to identify these missing values. There are two methods to detect missing data:

1. **.isnull()**
2. **.notnull()**

The output is a boolean value indicating whether the value that is passed into the argument is in fact missing data.

"True" stands for missing value, while "False" stands for not missing value.

```
:  missing_data = features_balanced.isnull()
   missing_data.head(5)
```

[18]:

	OBJECTID	X	Y	LOCATION	COLLISIONTYPE	PERSONCOUNT	UNDERINFL	WEATHER	ROADCOND	LIGHTCOND	SPEEDING	SEVERITYC
25055	False	False	False	False	False	False	False	False	False	False	True	
65280	False	False	False	False	False	False	False	False	False	False	True	
86292	False	False	False	False	False	False	False	False	False	False	True	
155111	False	True	True	False	False	False	False	False	False	False	True	
64598	False	False	False	False	False	False	False	False	False	False	True	

Speeding attributes showed the highest number of missing values

```
UNDERINFL
False    113683
True       2693
Name: UNDERINFL, dtype: int64

WEATHER
False    113560
True       2816
Name: WEATHER, dtype: int64

ROADCOND
False    113612
True       2764
Name: ROADCOND, dtype: int64

LIGHTCOND
False    113528
True       2848
Name: LIGHTCOND, dtype: int64

SPEEDING
True     110396
False      5980
Name: SPEEDING, dtype: int64
```

Based on the summary above, each column has 116376 rows of data, nine columns containing missing data:

"X": 2914 missing data

"Y": 2914 missing data

"LOCATION": 1305 missing data

"COLLISIONTYPE" : 2710 missing data

"UNDERINFL": 2693 missing data

"WEATHER": 2816 missing data

"ROADCOND": 2764 missing data

"LIGHTCOND": 2848 missing data

"SPEEDING": 110396 missing data

Several iterations were undertaken to clean and balance the data

Based on the summary above, columns were reduced to 5600 rows of data, and non of the columns contains missing data

```
# Check the data balance again

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

```
23]: 2    3319
     1    2281
     Name: SEVERITYCODE, dtype: int64
```

```
# Balance the data again

##Balance the unbalanced data to avoid bias ML model
from sklearn.utils import resample

n=2281

features_balanced_cl3 = features_balanced[features_balanced.SEVERITYCODE==1]
features_balanced_cl4 = features_balanced[features_balanced.SEVERITYCODE==2]

features_dsample = resample(features_balanced_cl4,
                            replace=False,
                            n_samples=n,
                            random_state=123)
features_balanced_1 = pd.concat([features_dsample, features_balanced_cl3])
features_balanced_1.SEVERITYCODE.value_counts()
features_balanced_1.head()
```

Final balanced data

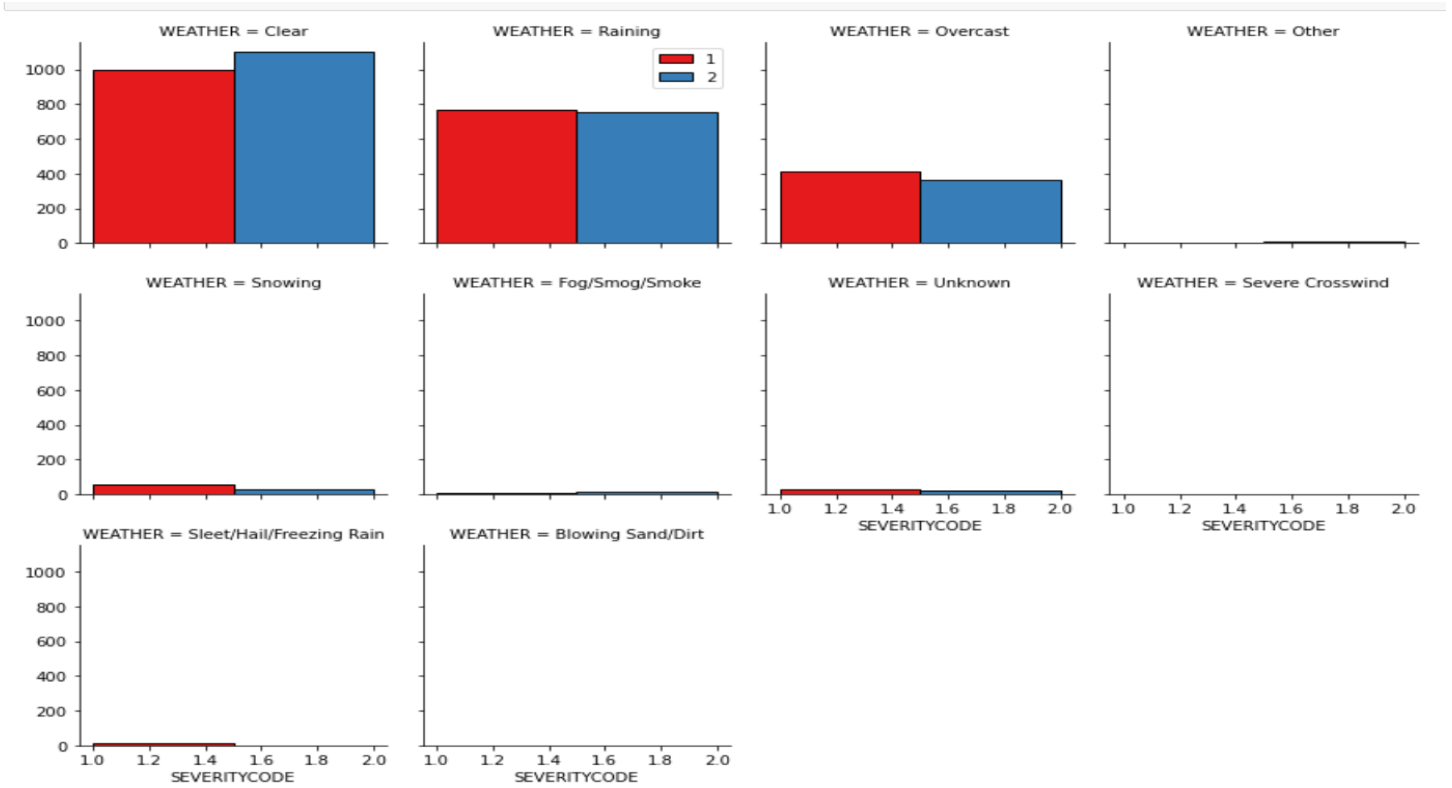
	OBJECTID	X	Y	LOCATION	COLLISIONTYPE	PERSONCOUNT	UNDERINFL	WEATHER	ROADCOND	LIGHTCOND	SPEEDING
4222	108957	-122.320760	47.611169	BROADWAY AND MADISON ST	Angles	3	N	Clear	Dry	Daylight	Y
4772	146119	-122.333450	47.546641	EAST MARGINAL WAY S BETWEEN 1ST AV S BR NB AND...	Rear Ended	6	N	Clear	Dry	Daylight	Y
3017	44860	-122.307202	47.543604	S CORGIAT DR BETWEEN URSULA PL S AND CORGIAT DR S	Other	1	1	Clear	Dry	Dark - No Street Lights	Y
5271	192417	-122.329554	47.734100	N 145TH ST BETWEEN SUNNYSIDE AVE N AND 1ST AVE NE	Rear Ended	4	N	Raining	Wet	Dawn	Y
4725	142762	-122.368082	47.568339	28TH AVE SW AND SW ANDOVER ST	Head On	3	N	Overcast	Wet	Dark - Street Lights On	Y

```
# Check the data balance again

features_balanced_1['SEVERITYCODE'].value_counts()
```

```
2    2281
1    2281
Name: SEVERITYCODE, dtype: int64
```

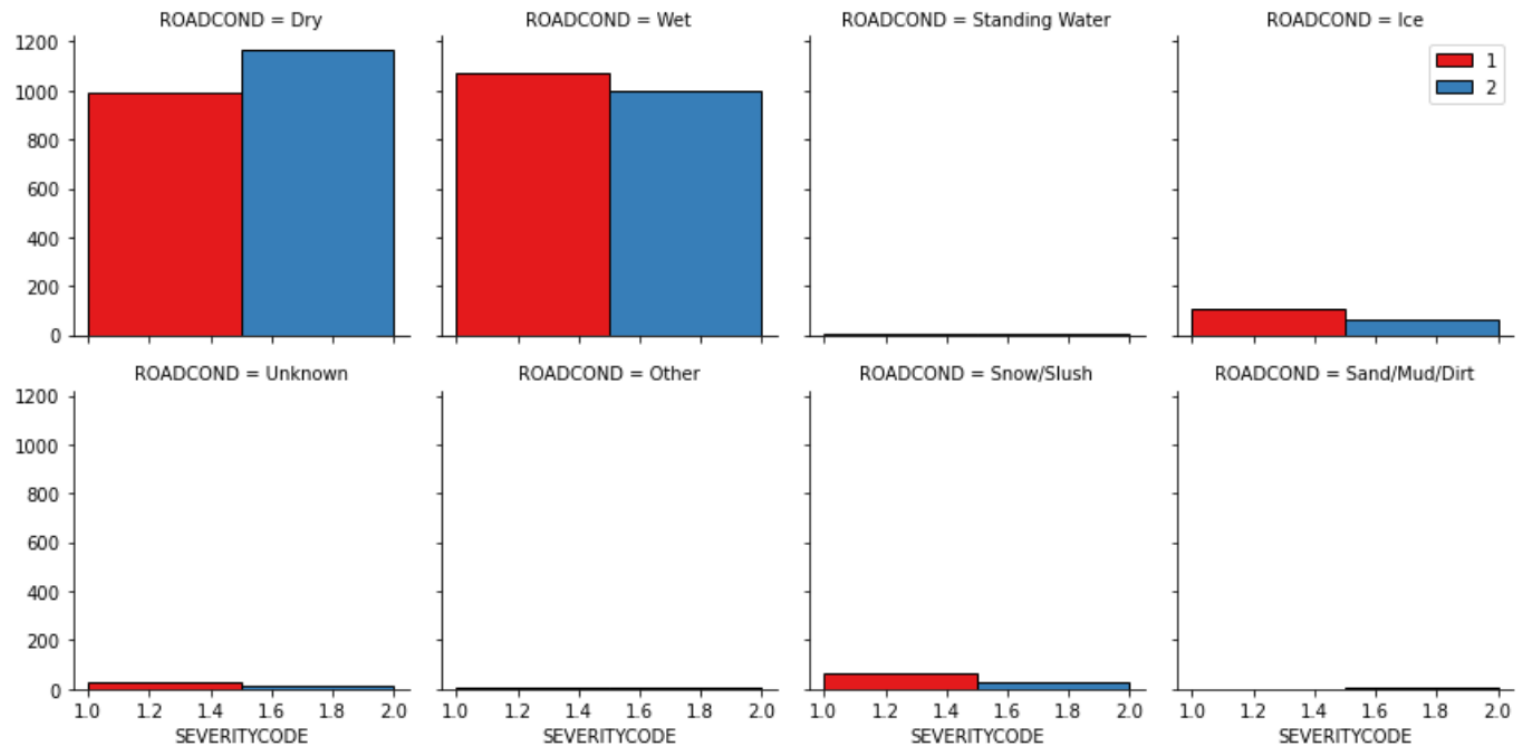
Weather contribution visualization



Weather contribution visualization

Weather contribution visualization (It is clear from the figures that the clear raining and overcast weather are the main contributer to the increased number of the care accident.)

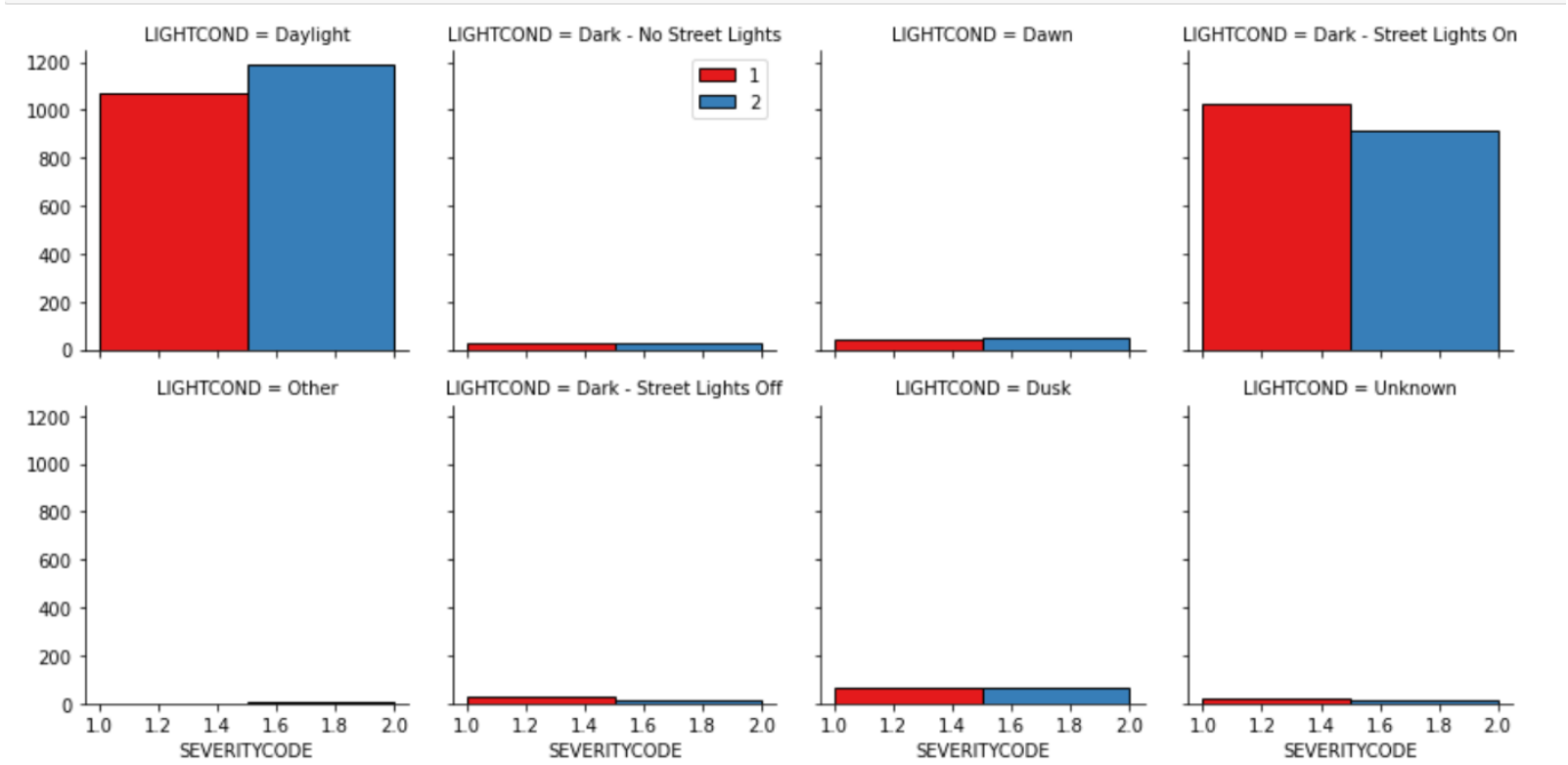
Road condition visualization



Road condition visualization

The condition of the road was visualized. The outcomes shows that the dry and wet were the main contributors.

Light condition visualization



All of the visualized factors were converted to numerical data

UNDERINFL	Blowing Sand/Dirt	Clear	Fog/Smog/Smoke	Other	Overcast	Raining	Severe Crosswind	Sleet/Hail/Freezing Rain	Snowing	...	Other	Unknown	Dry	Ice	Other	S
0	0	1	0	0	0	0	0	0	0	...	0	0	1	0	0	
0	0	1	0	0	0	0	0	0	0	...	0	0	1	0	0	
1	0	1	0	0	0	0	0	0	0	...	0	0	1	0	0	
0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	
0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	

Methodology

The data are cleaned and ready to be used in a machine learning model. There are several models will be tested as follows:

1. K-Nearest Neighbour (KNN)
2. Logistic Regression
3. SVM

Results and Discussion

1- KNN data testing and training

```
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: (3649, 27) (3649,)
Test set: (913, 27) (913,)
```

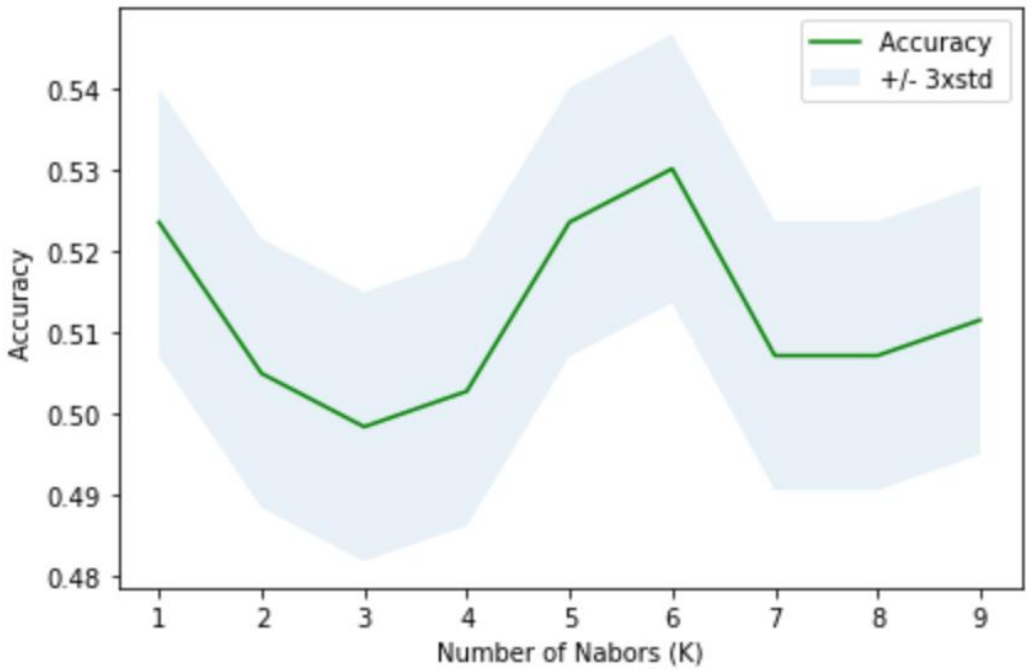
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
Ks = 10
mean_acc = np.zeros((Ks-1))
std_acc = np.zeros((Ks-1))
ConfustionMx = [];
for n in range(1,Ks):

    #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([0.52354874, 0.50492881, 0.49835706, 0.50273823, 0.52354874,
       0.53012048, 0.50711939, 0.50711939, 0.51150055])
```



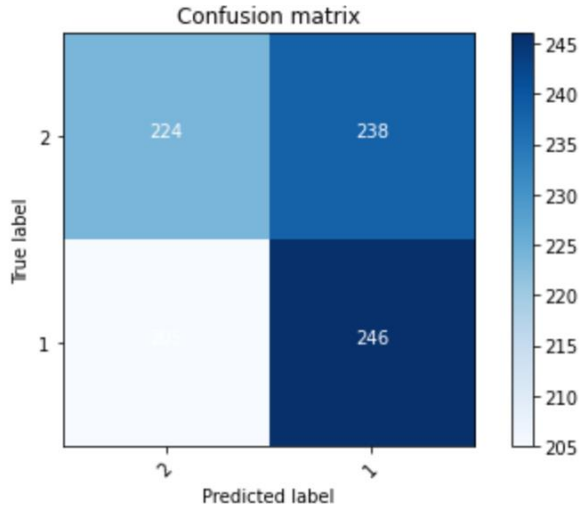
The best accuracy was with 0.5301204819277109 with k= 6

Results and Discussion

2- SVM

	precision	recall	f1-score	support
1	0.52	0.48	0.50	462
2	0.51	0.55	0.53	451
micro avg	0.51	0.51	0.51	913
macro avg	0.52	0.52	0.51	913
weighted avg	0.52	0.51	0.51	913

```
Confusion matrix, without normalization
[[224 238]
 [205 246]]
```



Results and Discussion

3- Logistics Regression

```
from sklearn.metrics import jaccard_similarity_score
jaccard_similarity_score(y_test, yhat2)
from sklearn.metrics import log_loss
log_loss(y_test, yhat_prob)
```

```
0.6912541423423092
```

Model Evaluation

```
yhatKNN=neigh.predict(X)
KNNJaccard = jaccard_similarity_score(y, yhatKNN)
KNNF1 = f1_score(y, yhatKNN, average='weighted')
print("KNNAvg F1-score: %.2f" % KNNF1 )
print("KNN Jaccard Score: %.2f" % KNNJaccard)


#yhatDEC = loneTree.predict(X)
#DTJaccard = jaccard_similarity_score(y, yhatDEC)
#DTF1 = f1_score(y, yhatDEC, average='weighted')
#print("Decision Tree Avg F1-score: %.2f" % DTF1 )
#print("Decision Tree Jaccard Score: %.2f" % DTJaccard)

yhatSVM=clf.predict(X)
SVMJaccard = jaccard_similarity_score(y, yhatSVM)
SVMF1 = f1_score(y, yhatSVM, average='weighted')
print("SVM Avg F1-score: %.2f" % SVMF1)
print("SVM Jaccard score: %.2f" % SVMJaccard)

yhatLOG = LR.predict(X)
yhatLOGproba = LR.predict_proba(X)
LogRJaccard = jaccard_similarity_score(y, yhatLOG)
LogRF1 = f1_score(y, yhatLOG, average='weighted')
Logloss = log_loss(y, yhatLOGproba)
print("LogLoss: : %.2f" % Logloss)
print("LOGR Avg F1-score: %.4f" % LogRF1)
print("LOGR Jaccard score: %.4f" % LogRJaccard)
```

```
KNNAvg F1-score: 0.52
KNN Jaccard Score: 0.53
SVM Avg F1-score: 0.55
SVM Jaccard score: 0.56
LogLoss: : 0.69
LOGR Avg F1-score: 0.5506
LOGR Jaccard score: 0.5568
```

Model Evaluation

Comparison and Evaluation

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.53	0.52	NA
SVM	0.56	0.55	NA
LogisticRegression	0.5568	0.5506	0.69

Conclusion

At the beginning of this project, the data pulled from the CSV file was not in the proper shape to perform machine learning model. Therefore, the data was cleaned, preprocessed, and attributes believed to influence the model were selected to pave the way for further analysis.

The label (severity code) was investigated. It was found that the prop damage (number 1) is almost 2.6 times the injury (number 2). To continue, it was necessary to balance the data that will be the input to the ML model. It was noticed that the speeding attribute was a vital factor in all accidents, and the majority of the cases were registered with yes as an answer for whether the driver was speeding or not. Therefore the data again were cleaned, and all the rows that have no answers were removed. The downsampling was conducted using the sklearn resample tool. The number of the rows at the end were 4562, and it was equally divided between number 2 and 1 in the severity code label.

Once the data analyzed, cleaned, and pre-processed, it was then fed to three ML models; KNN, SVM, and Logistic Regression. Even though KNN and SVM have presented acceptable results, logistic regression made the most sense because of its binary nature.

The models were evaluated using the evaluation matrices: Jaccard index, f-1 score, and logloss for logistic regression. The results were presented in a table, and the logistic Regression showed excellent results.