Predicting the Car Accident's Severity

1. Introduction

   In an effort to reduce the frequency of car collisions in a community, an algorithim must be developed to predict the severity of an accident given the current weather, road and visibility conditions. The high frequency of car collisions/accident can affect the day to day economical condition as it may affect the goods delivery from one place to another place. In this case, we need to create a model that will be able to predict the severity based on the condition mentioned and review what is the most important factor influencing the severity and frequency of the accident.

2. Data Understanding

   The data was downloaded from the example data set on first week part of the capstone project. Here is the summary of the dataset:

Data Set Summary

Data Set Basics	
Title	Collisions—All Years
Abstract	All collisions provided by SPD and recorded by Traffic Records.
Description	This includes all types of collisions. Collisions will display at the intersection or mid-block of a segment. Timeframe: 2004 to Present.
Supplemental Information	
Update Frequency	Weekly
Keyword(s)	SDOT, Seattle, Transportation, Accidents, Bicycle, Car, Collisions, Pedestrian, Traffic, Vehicle
Contact Information	
Contact Organization	SDOT Traffic Management Division, Traffic Records Group
Contact Person	SDOT GIS Analyst
Contact Email	DOT_IT_GIS@seattle.gov

Our target variable will be severity itself. We will classify this as:

0 : No probability (clear condition).

1: Very low probability that will cause property damage.

2: Low probability that will cause injury.

3: Medium probability that will cause serious injury.

4:High probability that will cause fatality.

Meanwhile, attributes used to weight the severity of an accident are ‘VEHCOUNT’, 'WEATHER', 'ROADCOND' and 'LIGHTCOND'.

In it's original form, this data is not fit for analysis. For one, there are many columns that we will not use for this model. Also, most of the features are of type object, when they should be numerical type. We also need to check the datatype for each column on the table and convert it to suitable format (int or float). Furthermore, we need to do the dataset balancing and data normalization. In this report, we’re given a set of data that consists of very low and low probability accident only.

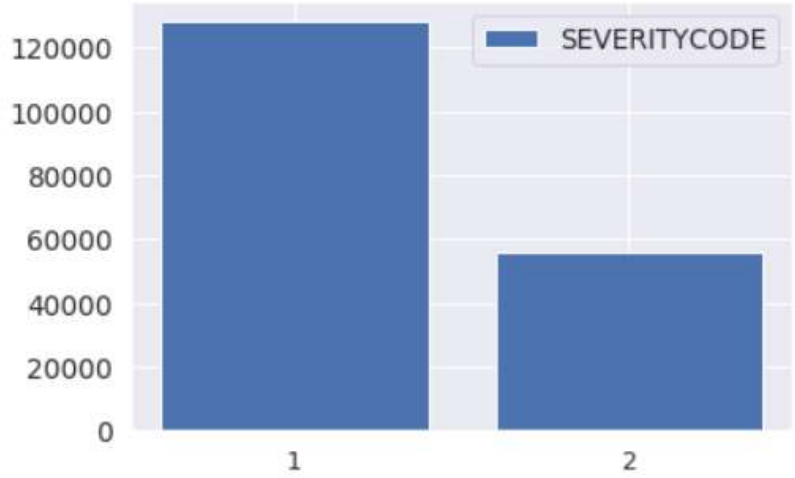
3. Data Preparation

The raw data was not ready to be analysed as there are many columns which were not required to do the analysis. Beside of this, we also need to drop the row which has NaN reading, label the data to numeric type, and change the data type on the column which has ‘object’ data type as it is not compatible to do the analysis using machine learning library. To build and test a machine learning model, we need to change the data type as float or integer.

	SEVERITYCODE	X	Y	OBJECTID	INCKEY	COLDETKEY	REPORTNO	STATUS	ADDRTYPE	INTKEY	LOCATION	VEHCOUNT	WEATHER	ROADCOND	LIGHTCOND
0	2	-122.323148	47.703140	1	1307	1307	3502005	Matched	Intersection	37475.0	5TH AVE NE AND NE 103RD ST	2	Overcast	Wet	Daylight
1	1	-122.347294	47.647172	2	52200	52200	2607959	Matched	Block	NaN	AURORA BR BETWEEN RAYE ST AND BRIDGE WAY N	2	Raining	Wet	Dark - Street Lights On
2	1	-122.334540	47.607871	3	26700	26700	1482393	Matched	Block	NaN	4TH AVE BETWEEN SENECA ST AND UNIVERSITY ST	3	Overcast	Dry	Daylight
3	1	-122.334803	47.604803	4	1144	1144	3503937	Matched	Block	NaN	2ND AVE BETWEEN MARION ST AND MADISON ST	3	Clear	Dry	Daylight
4	2	-122.306426	47.545739	5	17700	17700	1807429	Matched	Intersection	34387.0	SWIFT AVE S AND SWIFT AV OFF RP	2	Raining	Wet	Daylight

Screenshot of Raw Data

Beside of this, the data distribution and balance need to be checked as if it was not distributed correctly, it may give the wrong prediction based on machine learning model. Hence, we need to do data balancing and data standardization based on the number of each severity category.



Distribution of severity code 1 and 2 on the data

Based on the histogram above, the severity code in class 1 is nearly three time the size of class 2. We can fix this by downsampling the majority class.

```
Out[42]: 1    132285
         2     57052
         Name: SEVERITYCODE, dtype: int64
```

The number of data after doing downsampling on majority class, perfectly balanced.

After doing the balancing on the data, we need to make a good distribution on the data using StandardScaler methods on preprocessing command in sklearn library. Once it's done, we get the final dataset as shown below:

	SEVERITYCODE	VEHCOUNT	WEATHER	ROADCOND	LIGHTCOND
138325	1	2	4	0	5
68611	1	2	1	0	5
95270	1	2	4	8	5
147691	1	2	10	8	2
47598	1	2	1	0	5

First 5 rows of final dataset

4. Methodology

A correlation matrix on each parameter was done to check the correlation between the severitycode and another parameters. As shown below:

	SEVERITYCODE	VEHCOUNT	WEATHER	ROADCOND	LIGHTCOND
SEVERITYCODE	1.000000	-0.078667	-0.064129	-0.053807	-0.069152
VEHCOUNT	-0.078667	1.000000	-0.012531	-0.013101	0.042109
WEATHER	-0.064129	-0.012531	1.000000	0.825706	0.019084
ROADCOND	-0.053807	-0.013101	0.825706	1.000000	-0.011944
LIGHTCOND	-0.069152	0.042109	0.019084	-0.011944	1.000000

Correlation matrix table

This is done by finding correlation coefficient between two parameters for all parameters on the dataset. The target of doing this process is knowing what kind of factor influencing the risk of accident represented by the severity code. This part was done as exploratory data analysis.

Moreover, we use the final dataset which has been prepared to be fed into machine learning models. We will use the following models:

a. KNN (K-Nearest Neighbour)
   KNN will help us predict the severity code of an outcome by finding the most similar to data point within k distance.
b. Decision Tree
   A decision tree is a mechanical way to make a decision by dividing the inputs into smaller decisions. Like other models, it involves mathematics. But it's not very complicated mathematics. The approach is to look at the decisions and the factors that led to that decision. It is based on the concept of **entropy**. This looks at the frequency distribution of decisions and then calculates a logarithm.
c. SVM (Support Vector Machine)
   Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

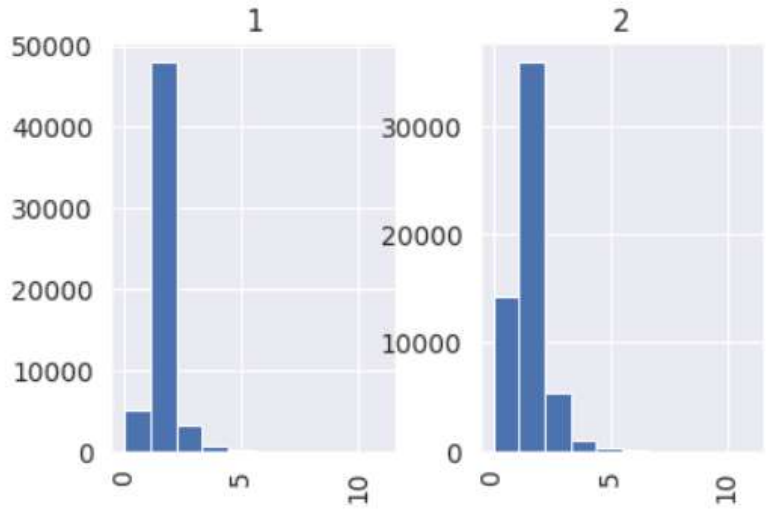
d. Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes. In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc. In this case, we use binomial mode as the model will predict 2 kinds of severity code only.

Then, we will check the similarity score to choose the best models using jcard, log loss, and f1_score.

5. Result and Evaluation

As shown on the correlation matrix, the strongest relationship shown between the vehicle count with the correlation coefficient -0.078667. To confirm this condition, the histograms have been made as below:



Relationship between the vehicle count(x-axis) and the frequency of severity class 1(y-axis) (left-side) and the vehicle count(x-axis) and the frequency of severity class 2 (y-axis) (right-side)

As shown on the histogram above, the number of accidents has reversed relationship relatively with the severity code. That means the number of vehicles involved on the accident increase along the reduction of accident frequency. More accident happened on the severity class 1.

Moreover, the dataset needs to be fed on each of machine learning fed to train the model and make a prediction. First of all we define the X and Y as shown below:

```
In [57]: X = df_downsampled.drop('SEVERITYCODE', axis=1)
```

```
In [58]: Y = df_downsampled["SEVERITYCODE"]
         Y
```

```
Out[58]: 82770     1
         122946    1
         102968    1
         13906     1
```

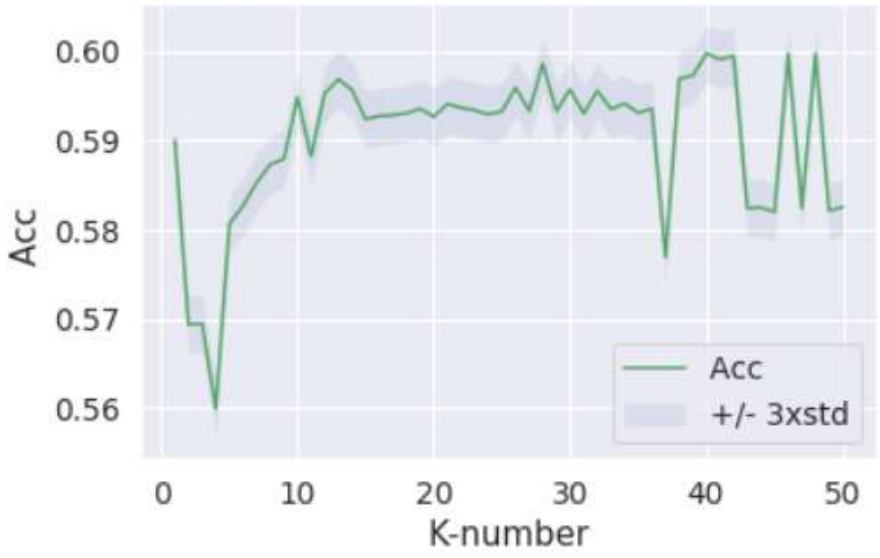
Then, the data standardization was done using Standardscaler as shown below:

```
X= preprocessing.StandardScaler().fit(X).transform(X)
```

a. KNN (K-Nearest Neighbor)

We tested several K numbers and found that the best K-number if we are going to use KN N model is 40 with the accuracy as 0.5997546119801936. Here is the mean accuracy for K =1 to K=51:

```
Out[61]: array([0.59002673, 0.56934403, 0.56947548, 0.55992288, 0.5806494 ,
                0.58266509, 0.58525043, 0.58730993, 0.5879234 , 0.59484685,
                0.58823014, 0.59528504, 0.59686254, 0.59567942, 0.59239297,
                0.59274353, 0.59283116, 0.59305026, 0.59357609, 0.59265589,
                0.59410192, 0.59366373, 0.59340082, 0.5929188 , 0.59326936,
                0.59594233, 0.59331318, 0.59861531, 0.59331318, 0.59572324,
                0.59300644, 0.59559178, 0.59353227, 0.59414574, 0.59309408,
                0.59357609, 0.57692476, 0.59695018, 0.59725691, 0.59975461,
                0.59909732, 0.5994917 , 0.58240217, 0.58248981, 0.58196398,
                0.59971079, 0.58240217, 0.59971079, 0.58209544, 0.58253363])
```




```
The best accuracy was with 0.5997546119801936 with k= 40
```

Relationship of K-number and accuracy

b. Decision Tree

After training the data using decision tree algorithm using the criterion of entropy and max depth = 4, we got the accuration result as below:

```
In [64]: from sklearn.tree import DecisionTreeClassifier
```

```
In [65]: dtm = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
         dtm.fit(X_train,Y_train)
```

```
Out[65]: 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')
```

```
In [66]: Y_pred=dtm.predict(X_test)
         dtm_acc=metrics.accuracy_score(Y_test,Y_pred)
         dtm_acc
```

```
Out[66]: 0.6075982647561456
```

The accuration score shows as 0.6075982647561456. It is relatively higher than using KNN algorithm. However, we still need to check the accuracy on SVM and logistic regression model.

c. SVM (Support Vector Machine)

After training the data using SVM algorithm, the result of accuration's shown as below:

```
In [67]: from sklearn import svm
         svm_m=svm.SVC(kernel='rbf')
         svm_m.fit(X_train,Y_train)

/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/svm/base.py:19
o account better for unscaled features. Set gamma explicitly to 'auto' or '
  "avoid this warning.", FutureWarning)

Out[67]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)

In [68]: Y_pred=svm_m.predict(X_test)

In [69]: metrics.accuracy_score(Y_test,Y_pred)

Out[69]: 0.6076859033346479
```

The accuracy score is relatively higher than the two previous model.

d. Logistic Regression

Here is the accuracy score of using logistic regression algorithm to build the model:

```
In [70]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import confusion_matrix
         lrm=LogisticRegression(C=0.01,solver='liblinear').fit(X_train,Y_train)

In [71]: Y_pred=lrm.predict(X_test)

In [72]: metrics.accuracy_score(Y_test,Y_pred)

Out[72]: 0.5405985714911704
```

So far, it is relatively lower than another models. However, we still need to check the similarity score using several methods as shown below:

In [49]:
```
# ACCURACY SCORES
# knn
yhat = knnmodel.predict(X_test)
yhat
print("Train set KNN Accuracy: ", metrics.accuracy_score(Y, knnmodel.predict(X)))
print("Test set KNN Accuracy: ", metrics.accuracy_score(Y_test, yhat))
knn_jaccard = jaccard_similarity_score(Y_test, yhat)
knn_f1_score = f1_score(Y_test, yhat, average='weighted')

# Decission tree
yhat = dtm.predict(X_test)
yhat
print("Train set Decission Tree Accuracy: ", metrics.accuracy_score(Y, dtm.predict(X)))
print("Test set Decission Tree Accuracy: ", metrics.accuracy_score(Y_test, yhat))
tree_jaccard = jaccard_similarity_score(Y_test, yhat)
tree_f1_score = f1_score(Y_test, yhat, average='weighted')

# SVM
yhat = svm_m.predict(X_test)
yhat
print("Train set SVM Accuracy: ", metrics.accuracy_score(Y, svm_m.predict(X)))
print("Test set SVM Accuracy: ", metrics.accuracy_score(Y_test, yhat))
svm_jaccard = jaccard_similarity_score(Y_test, yhat)
svm_f1_score = f1_score(Y_test, yhat, average='weighted')

# Logistic regression
yhat = lrm.predict(X_test)
yhat_proba = lrm.predict_proba(X_test)
yhat
print("Train set Logistic regression Accuracy: ", metrics.accuracy_score(Y, lrm.predict(X)))
print("Test set Logistic regression Accuracy: ", metrics.accuracy_score(Y_test, yhat))
lr_jaccard = jaccard_similarity_score(Y_test, yhat)
lr_f1_score = f1_score(Y_test, yhat, average='weighted')
lr_log_loss = log_loss(Y_test, yhat_proba)
```

```
Train set KNN Accuracy:  0.5795765266774171
Test set KNN Accuracy:  0.5825336313045002
Train set Decission Tree Accuracy:  0.6067622519806493
Test set Decission Tree Accuracy:  0.6075982647561456
Train set SVM Accuracy:  0.6081732454602818
Test set SVM Accuracy:  0.6076859033346479
Train set Logistic regression Accuracy:  0.5335132861249386
Test set Logistic regression Accuracy:  0.5405985714911704
```

In [50]:
```
report = pd.DataFrame(data=np.array([["KNN", knn_jaccard, knn_f1_score, np.nan],
                                     ["Decision Tree", tree_jaccard, tree_f1_score, np.nan],
                                     ["SVM", svm_jaccard, svm_f1_score, np.nan],
                                     ["LogisticRegression", lr_jaccard, lr_f1_score, lr_log_loss]]), columns=["Algorithm", "Jaccard", "F1-score", "LogLoss"])
report = report.set_index(["Algorithm", "Jaccard", "F1-score", "LogLoss"])
report
```

Out[50]:

Algorithm	Jaccard	F1-score	LogLoss
KNN	0.5825336313045002	0.5743848524368107	nan
Decision Tree	0.6075982647561456	0.5841367886192402	nan
SVM	0.6076859033346479	0.5978099476132452	nan
LogisticRegression	0.5405985714911704	0.5349626716190827	0.6846179823388177

From several methods to check the similarity result above, the model built from SVM algorithm has the highest score.

6. Discussion

After watching on the raw dataset, we could see that there were too many columns containing categorical data. We could not use all of these data to fed the machine learning models hence we simplified the dataset to consider the VEHCOUNT', 'WEATHER', 'ROADCOND' and 'LIGHTCOND'

only. Later, we gave the label to each categories on the columns and change the data type from ‘object’ to ‘integer’ or ‘float’ as the ‘object’ could not be used to fed the machine learning algorithm/models.

We could see that there were NaN values so we drop the rows containing NaN value then solve the issue with imbalance data where severity class 1 is almost three times larger than class 2. We did a downsampling on the majority class and match the minority class with 57052 values each.

Once we analyzed and cleaned the data, it was then fed through four ML models; K-Nearest Neighbor, Decision Tree, SVM and Logistic Regression. We found that the SVM has highest similarity and accuration score as the model built using this algorithm has flexibility on any condition wherever different category was placed with minor or major conditions. When the algorithm couldn’t place a ‘plane’ between two groups, it will change the values by giving the same behaviour (for example, powering x and y values) on x and y reading till it is separated perfectly. Evaluation metrics used to test the accuracy of our models were jaccard index, f-1 score and logloss for all of algorithm.

7. Conclusion

Based on historical data from vehicle count pointing to certain classes, we can conclude that the number of vehicles involved on the accident increase along the reduction of accident frequency. More accident happened on the severity class 1. In another word, the accident involving many vehicles are rarely happen.

Beside of this, the light condition was also the second strongest factor influencing the frequency of accidents. We recommend to put enough lighting on the spot where the frequency of accident is high.