Predicting the severity of road collisions in Seattle

Submitted by: Thang Le Dinh

Date: 10th October 2020

Introduction

Background

Seattle is a seaport city on the West Coast of the United States that is the largest city in both the state of Washington and the Pacific Northwest region of North America. According to U.S. Census data released in 2019, the Seattle metropolitan area's population stands at 3.98 million, making it the 15th largest in the United States and being the fastest-growing major U.S. city, with a 3.1% annual growth rate.

Problem

Since Seattle is a big and dynamic city, it is also known for its heavy traffic. Consequently, one of the most important challenges for the government of Seattle is to prevent car collisions.

For instance, hereafter is 2015 collision clock of Seattle[1]:

- A crash occurred every 4.5 minutes.
- A person died in a crash every 16 hours.
- A person was injured in a crash every 11 minutes.
- A motorcyclist was in a crash every 4 hours.
- A pedestrian or bicyclist was involved in a crash every 2 ½ hours.
- A pedestrian or bicyclist was killed in a crash every 4 days.
- A speeding driver was involved in a crash every 27 minutes.
- An inattentive/distracted driver was involved in a crash every 12 minutes.
- A person was killed by an impaired driver every 1 ½ days.

Motivation

This project aims at predicting the severity of car collisions for the Seattle Police Department (SPD) based on different factors, such as weather, road and light conditions as well as the number of peoples and vehicles involved.

[1] 2015 Annual Collision Summary :
https://www.wsdot.wa.gov/mapsdata/crash/pdf/2015_Annual_Collision_Summary.pdf

Data understanding

Data sources

Since 2004, the SPD has collected data related to road collisions recorded by the Traffic Records. Its dataset includes the following key attributes:

- Collision address types,
- Levels of the severity of the collision,
- Numbers of people and pedestrians involved,
- Numbers of bicycles and vehicles involved,
- Numbers of injuries, serious injuries and fatalities in the collision, and
- situations related to the inattention, drugs or alcohol, speeding, and pedestrian right of way as well as conditions of weather, road, and light.

Data cleaning

Firstly, the dataset is downloaded and read into a data frame using a Jupyter Notebook. Thus, this dataset is cleaned up to remove columns that are not informative to us for visualization (Figure 1). Finally, columns and rows, where at least one element is missing, have been removed.

Clean up the dataset to remove columns that are not informative to us for visualization

```
# Remove columns, which are not informative

collision_df.drop(["X", "Y", "INCKEY","COLDETKEY","REPORTNO","STATUS","INTKEY","LOCATION","EXCEPTRSNCODE","EXCEPTRSNDESC",
                   "SEVERITYCODE.1", "SEVERITYDESC", "COLLISIONTYPE", "UNDERINFL","PEDCOUNT","PEDCYLCOUNT","INCDATE","INCDTTM",
                   "JUNCTIONTYPE","SDOT_COLCODE","SDOT_COLDESC", "PEDROWNOTGRNT","SDOTCOLNUM","ST_COLCODE","ST_COLDESC",
                   "SEGLANEKEY","CROSSWALKKEY","HITPARKEDCAR"] ,
                    axis=1 , inplace=True)
collision_df.head()
```

	SEVERITYCODE	OBJECTID	ADDRTYPE	PERSONCOUNT	VEHCOUNT	INATTENTIONIND	WEATHER	ROADCOND	LIGHTCOND	SPEEDING
0	2	1	Intersection	2	2	NaN	Overcast	Wet	Daylight	NaN
1	1	2	Block	2	2	NaN	Raining	Wet	Dark - Street Lights On	NaN
2	1	3	Block	4	3	NaN	Overcast	Dry	Daylight	NaN
3	1	4	Block	3	3	NaN	Clear	Dry	Daylight	NaN
4	2	5	Intersection	2	2	NaN	Raining	Wet	Daylight	NaN

Figure 1: Cleaning up the dataset.

Feature selection

As mentioned in Table 1, ten key attributes were selected to determine certain patterns and correlations and then used for training a machine-learning model to predict the probability and severity of collisions.

Table 1: The feature selection.

Attributes	Description	Data type
OBJECTID	ESRI unique identifier	`int64`
SEVERITYCODE	A code that corresponds to the severity of the collision	`int64`
ADDRTYPE	Collision address type	`object`
PERSONCOUNT	The total number of people involved in the collision	`int64`
VEHCOUNT	The number of vehicles involved in the collision	`int64`

INATTENTIONIND	Whether or not collision was due to inattention	object
WEATHER	A description of the weather conditions during the time of the collision	object
ROADCOND	The condition of the road during the collision	object
LIGHTCOND	The light conditions during the collision	object
SPEEDING	Whether or not speeding was a factor in the collision	object

For verifying the consistency of the dataset, the types of the column labels have been examined to ensure that all column labels of type string.

Exploratory Data Analysis

It is observed that the dataset includes different categorical variables, such as Severity Code, Address type, Weather, Road condition, and Light condition. Therefore, we can firstly, explore the data related to each variables, and then explore the relationship between those variables.

Severity code

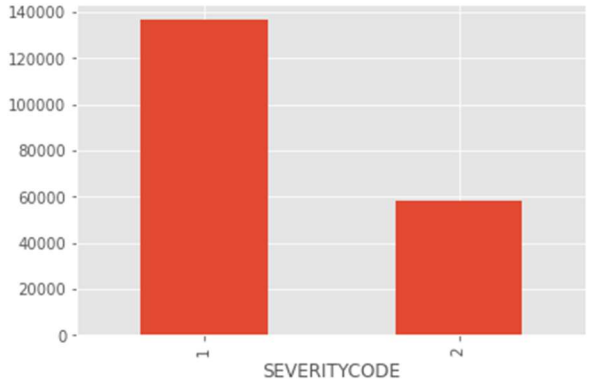

Figure 2: Exploring Severity code.

In the dataset, the majority of collisions is related to the “1-prop damage”, which means that there is the damage of the vehicles. There is another category related to the “2-injury”, which means that there is an injury. There are no fatality and serious injury collisions.

Address type

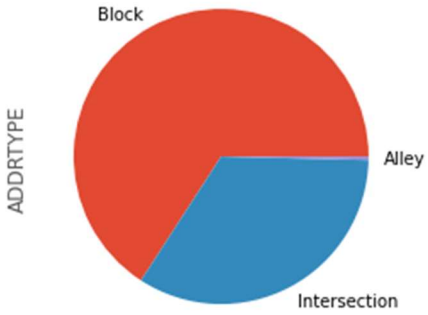

Figure 3: Exploring Address type.

There are three collision address types: Alley, Block and Intersection. The majority of collisions is related to block and then intersection.

Weather, Road and Light conditions

It is interesting to know that the majority of collisions had been happened in the good driving conditions: clear vision (weather), dry (road) and day light (light condition).

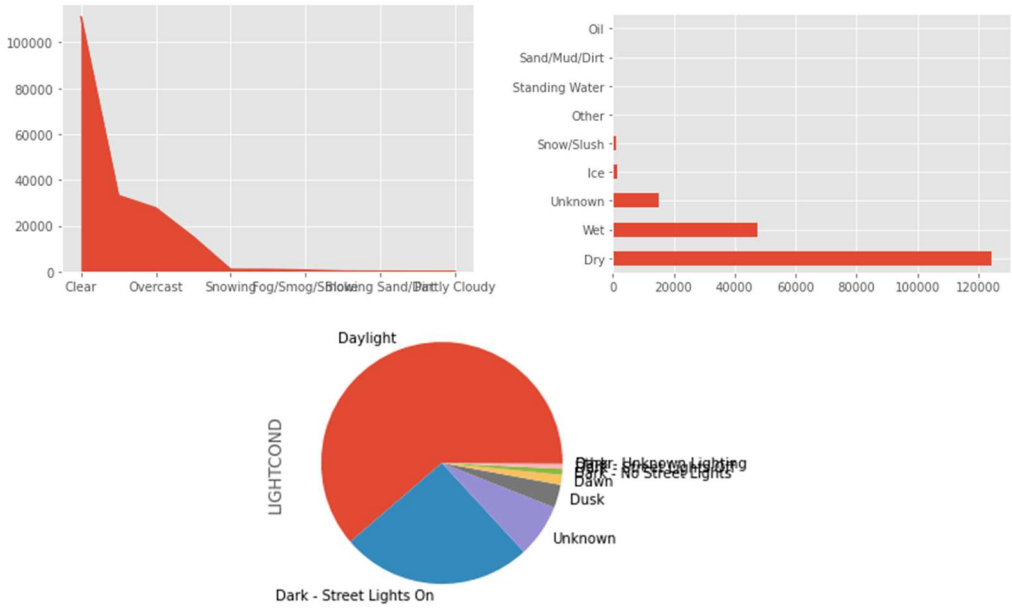

Figure 4: Exploring Weather, Road and Light conditions.

Relationships between the variables

Let us focus on the numerical variables of the dataset: Severity code, Person count and Vehicle count.

Figure 5: Heatmap for visualizing the potential relationships.

To determine the potential relationships between those variables, a heat map is used to visualize the relationships depicted by colour. It is observed that there may be the relationships between *Person count* and *Vehicle count* as well as *Person count* and *Severity code*.

Examining the potential relationships with scatter plots

The simple scatter plots have created for exploring about these two potential relationships.

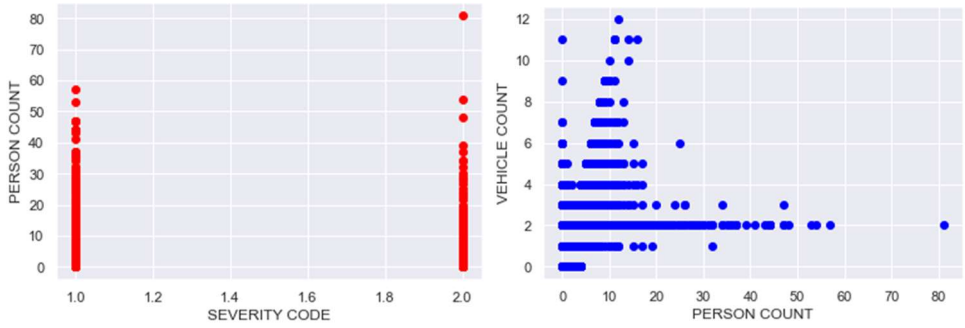

Figure 6: Examining the potential relationships with scatter plots.

As we can observe in the scatter plots, there is no relationship between *Person count* and *Severity code*; however, there might be a relationship between *Vehicle count* and *Person count*.

Simple linear regress for representing the relationship between Severity code and Person count

To observe the potential relationship between Severity code and Person count, we perform the Train/Test Split step to split the dataset into training and testing sets, which are mutually exclusive (80% of the entire data for training, and the 20% for testing).

Thus, Coefficient and Intercept are calculated as the parameters of the fit line.

```
|: from sklearn import linear_model
   regr = linear_model.LinearRegression()
   train_x = np.asanyarray(train[['PERSONCOUNT']])
   train_y = np.asanyarray(train[['VEHCOUNT']])
   regr.fit (train_x, train_y)
   # The coefficients
   print ('Coefficients: ', regr.coef_)
   print ('Intercept: ',regr.intercept_)

Coefficients:  [[0.17838949]]
Intercept:  [1.48456537]
```

Figure 7: Coefficient and Intercept of the linear model..

We can then plot the fit line over the data:

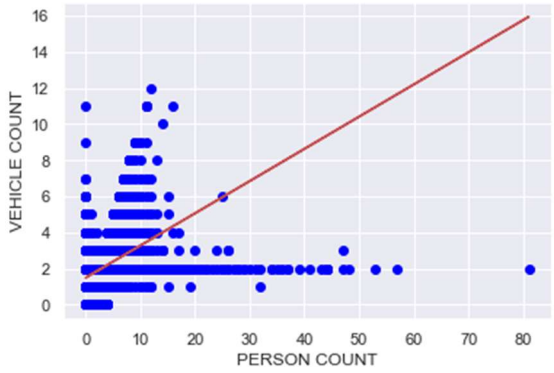

Figure 8: Scatter plot with the fit line.

Consequently, there does appear to be a correlation here and because the fit line drawn amongst these points would have a positive slope, that correlation is positive.

Machine learning modeling

In this section, we will continue to apply different machine learning models to capture more insights from this dataset.

Creating train and test dataset

We will split the dataset into train and test sets again, 80% of the entire data for training, and the 20% for testing.

```
: # Test/Train split
  from sklearn.model_selection import train_test_split
  X_train, X_test, y_train, y_test = train_test_split(X_df, y_df, test_size=0.2, random_state=4)
  y_train = y_train.ravel()
  print ('Train set:', X_train.shape,  y_train.shape)
  print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (155738, 6) (155738,)
Test set: (38935, 6) (38935, 1)
```

Figure 9: Spliting the dataset.

In the following, the three machine learning models such as K Nearest Neighbors, Decision Tree and Regression, which are often used to analyze the large dataset, will be selected for this project.

K Nearest Neighbors analysis

The k-nearest neighbors (KNN) algorithm is an easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

Let us begin with the K parameter. K in K Nearest Neighbors is the number of nearest neighbors to examine, which is supposed to be specified by the User. To choose the right value for K, we choose k =1 to calculate the accuracy of prediction using all samples in the test set and then increase the k to find which k is the best for the model.

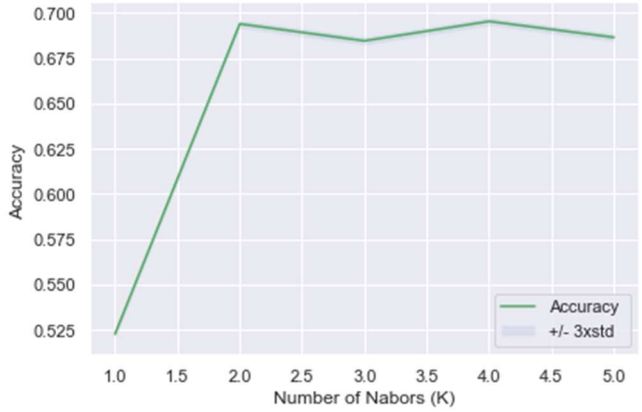

Figure 10: Best K for KNN model.

According to the above figure, the best accuracy was 0.6953127006549377, with K = 4.

```
# Best accuracy was with k = 4
k = 4
knn = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)

knn_y_pred = knn.predict(X_test)
knn_y_pred[0:5]
```

Figure 11: Predicting with the KNN model.

Continuing with K=4, we can train the data and evaluate the K Nearest Neighbours (KNN) using Jaccard score and F1 score. Jaccard is defined as the size of the intersection divided by the size of

the union of two label sets. Meanwhile, the F1 score is the harmonic average of the precision and recall.

```
: # KNN Evaluation
  from sklearn.metrics import jaccard_score
  from sklearn.metrics import f1_score

  knn_js = jaccard_score(y_test, knn_y_pred)
  knn_f1 = f1_score(y_test, knn_y_pred, average='macro')
  print ('KNN - Jaccard_score:', knn_js)
  print ('KNN - F1 score:', knn_f1)
```

```
KNN - Jaccard_score: 0.6927081984198938
KNN - F1 score: 0.4355822353605406
```

Figure 12: KNN evaluation.

Decision tree analysis

Decision tree is one of the machine learning approaches that uses a decision tree as a predictive model, which goes from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).

We will first create an instance of the *DecisionTreeClassifier* called *CollisionTree*. We then fit the data with the training feature matrix *X_train* and training response vector *y_train*. Finally, we make some predictions on the testing dataset and store it into a variable called *predTree*.

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
collisionTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

collisionTree.fit(X_train,y_train)
predTree = collisionTree.predict(X_test)
```

Figure 13: Predicting with Decision tree model.

In the following, we import metrics from *sklearn* and check the accuracy of our model.

```
: # Decision Tree Evaluation
  tree_js = jaccard_score(y_test, predTree)
  tree_f1 = f1_score(y_test, predTree, average='macro')
  print ('Decision tree - Jaccard_score:', tree_js)
  print ('Decision tree - F1 score:', tree_f1)
```

```
Decision tree - Jaccard_score: 0.7043715004880053
Decision tree - F1 score: 0.41336032198083383
```

Figure 14: Decision tree evaluation.

Regression analysis

The logistic regression is selected to model the probability of an event existing. The current version of Logistic Regression in *Scikit-learn*, support regularization, which is a technique used to solve the overfitting problem (based on the C parameter indicating inverse of regularization strength). We then fit the model with the train set, predict using the test set and try log loss for evaluation.

```
: # Regression
from sklearn.metrics import log_loss
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)

LR_pred = LR.predict(X_test)
LR_prob = LR.predict_proba(X_test)
reg_ll = log_loss(y_test, LR_prob)
print ('Regression - Log loss:', reg_ll)
```

```
Regression - Log loss: 0.5986679016584695
```

Figure 15: Predicting with Regression model.

Next, we try Jaccard index for accuracy evaluation and calculate the F1 scores for each label based on the precision and recall of that label.

```
: # Regression evaluation
reg_js = jaccard_score(y_test, LR_pred)
reg_f1 = f1_score(y_test, LR_pred, average='macro')
print ('Regression - Jaccard_score:', reg_js)
print ('Regression - F1 score:', reg_f1)
```

```
Regression - Jaccard_score: 0.704227884516593
Regression - F1 score: 0.41348440648219276
```

Figure 16: Regression model evaluation.

Discussion

Table 2: Machine learning model evaluations.

Model	K Nearest Neighbours	Decision Tree	Regression
Jaccard score	0.6927081984198938	0.7043715004880053	0.704227884516593
F1 score	0.4355822353605406	0.41336032198083383	0.41348440648219276
Log loss	N/A	N/A	0.5986679016584695

Firstly, the Jaccard score is used for accuracy evaluation. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise, it is 0.0. We can observe that accuracy of the three models is relative high (around 0.70). The best model in this case is Decision tree (highest Jaccard score = 0.7043715004880053).

Secondly, the F1 score is the harmonic average of the precision and recall. The F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. The best model based on this score is the K Nearest Neighbours (highest F1 score: 0.4355822353605406).

Thirdly, the Log loss (Logarithmic loss) measures the performance of a classifier where the predicted output is a probability value between 0 and 1. Since we can calculate only the Log loss for the Regression model, the only insight in that case is that the regression model is also an appropriate choice (Log loss = 0.5986679016584695 can be considered as an above average).

Discussion

Based on our analysis, hereafter are some important points:

- The majority of collisions causes injury and vehicle damage

- Most of the collision happened at certain specific address types: block and intersection.

- Most of the collisions can be happened in a good driving condition such as vision clear, day light and dry weather.

- When the collusions involved several vehicles, it may involve several persons.

- Decision tree model could be a good model to build a decision process to warn drivers the possibility of collisions.

- The K Nearest Neighbors model could be a good model to determine the patterns of collision to build an alert system.

- The Regression model could be an appropriate model to predict the severity of collisions in certain specific cases.

Recommendations

After exploring the dataset and predicting the severity of collisions using the three machine learning models, our recommendations are as the following:

- For the Public Development Authority of Seattle: Based on the patterns of vehicle collisions, the safety signs should be installed at and the lighting / road conditions should be improved the corresponding locations.
- For software developers, there is a need to develop a mobile application to display the potentiality and severity of vehicle collisions based on the patterns and predictions based on their locations. This application can integrate with GPS-based services (such as Google maps) to warn drivers at these locations or to suggest an alternative route if required.