Seattle Traffic Accident Severity - A Case Study

September 25, 2020

By Kate Pickering

1 Introduction

Car accidents result in millions of dollars in property damage and injury annually nationwide.

Can we use existing accident data, such as speeding, inattention, number of vehicles, involvement of pedestrians , to calculate their influence in predicting the severity of an accident? These types of accident data are quite easy to obtain, and should be relatively free of bias, as they contain no identifying data.

Could the relationship between existing accident data and outcome severity, be useful in making future laws, or driver regulations?

2 Data

The data will be harvested from the Seattle Police Department collisions data set. We will primarily focus on the data for accident severity, speeding, driver inattention, number of vehicles involved, number of persons involved, and number of pedestrians and cyclists. We will then look at the independant variables listed above, to see if they can be used to accurately predict an outcome, which is the severity of accident, classified as property damage, injury, serious injury and fatality.

2.1 Initial Data Exploration

Initially, the data was imported into a pandas dataframe in raw form, from the Seattle Police Department Traffic Records department on September 23, 2020. This data is updated weekly, so it is important to note the date on which the data was obtained.

Since there are many columns of data included in the raw dataframe, we need to focus in on the data columns of interest, which are as follows:

1) Accident severity code (column name “SEVERITYCODE”). This is the dependant variable we would like to predict. This code contains text values as follows: 0: Unknown 1: Property Damage 2: Injury 2b: Serious Injury 3: Fatality

2) Number of people involved (column name “PERSONCOUNT”). This is one of the independant variables we would like to use, to see if we can predict the accident severity. This column contains intiger values.

3) Number of pedestrains involved (column name “PEDCOUNT”). This is one of the independant variables we would like to use, to see if we can predict the accident severity. This column contains intiger values.

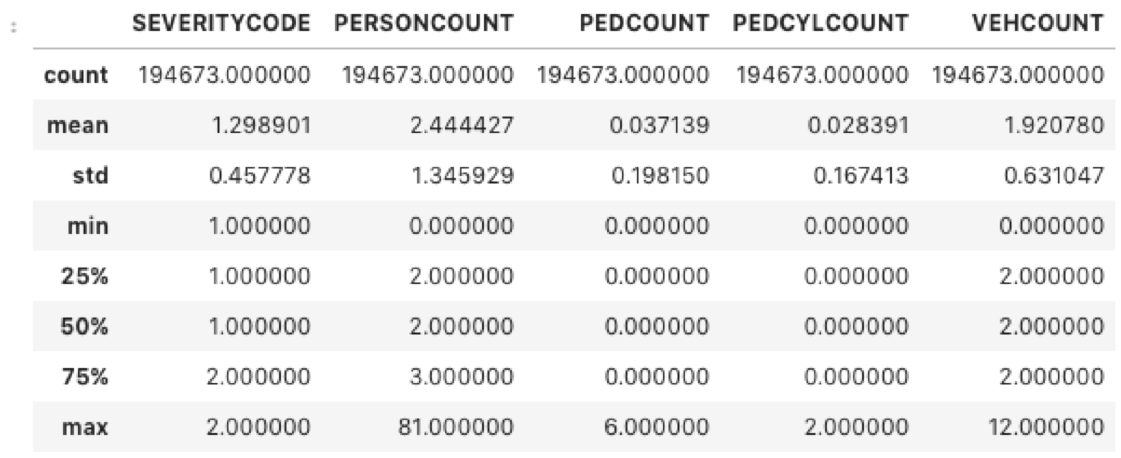
4) Number of cyclists involved (column name “PEDCYLCOUNT”). This is one of the independant variables we would like to use, to see if we can predict the accident severity. This column contains intiger values.

5) Number of vehicles involved (column name “VEHCOUNT”). This is one of the independant variables we would like to use, to see if we can predict the accident severity. This column contains intiger values.

6) If driver inattention was involved (column name “INATTENTIONIND”). This is one of the indepentant variables we would like to use, to see if we can predict the accident severity. This column contains text values of Y, if driver inattention was involved, and is left blank if no driver inattention was involved.

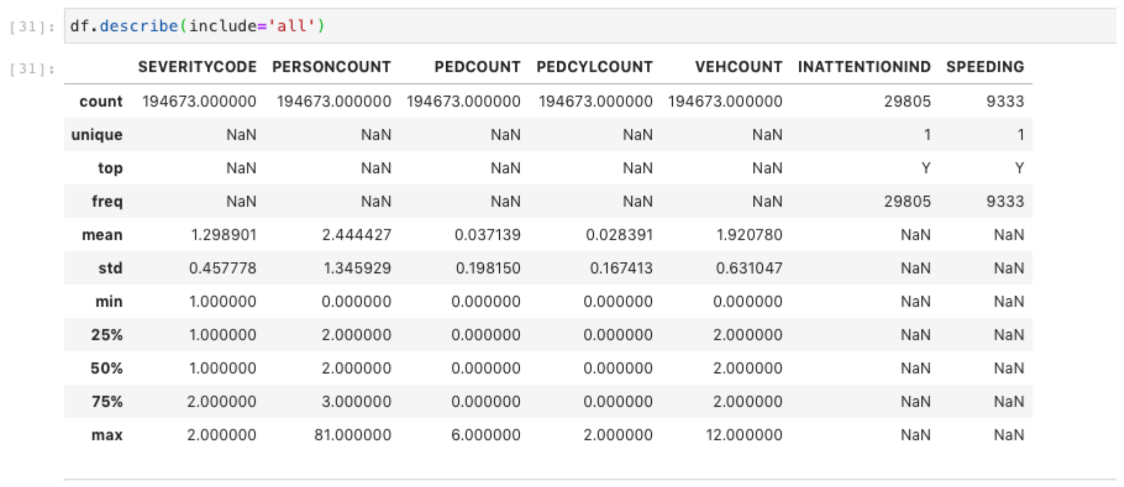
7) If speeding was involved (column name “SPEEDING”). This is one of the indepentant variables we would like to use, to see if we can predict the accident severity. This column contains text values of Y, if speeding was involved, and is left blank if no driver inattention was involved.

Next, a dataframe containing only the columns above was created. Then, to begin to initially explore the data, we used the pandas describe method, to find general information about the data, such as the number of rows (194673), and the value spread in each column.



	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT
count	194673.000000	194673.000000	194673.000000	194673.000000	194673.000000
mean	1.298901	2.444427	0.037139	0.028391	1.920780
std	0.457778	1.345929	0.198150	0.167413	0.631047
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	2.000000	0.000000	0.000000	2.000000
50%	1.000000	2.000000	0.000000	0.000000	2.000000
75%	2.000000	3.000000	0.000000	0.000000	2.000000
max	2.000000	81.000000	6.000000	2.000000	12.000000

From this inital exploration of data, we can see that there is no accident contained with a severity over 2, which is injury. We have only have severity as 1 or 2. Since we have two data columns which contain text, we will also need to run an all data descrition method, to see the number of occurrences of driver inattention, and speeding.



```
[31]: df.describe(include='all')
```

[31]:

	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND	SPEEDING
count	194673.000000	194673.000000	194673.000000	194673.000000	194673.000000	29805	9333
unique	NaN	NaN	NaN	NaN	NaN	1	1
top	NaN	NaN	NaN	NaN	NaN	Y	Y
freq	NaN	NaN	NaN	NaN	NaN	29805	9333
mean	1.298901	2.444427	0.037139	0.028391	1.920780	NaN	NaN
std	0.457778	1.345929	0.198150	0.167413	0.631047	NaN	NaN
min	1.000000	0.000000	0.000000	0.000000	0.000000	NaN	NaN
25%	1.000000	2.000000	0.000000	0.000000	2.000000	NaN	NaN
50%	1.000000	2.000000	0.000000	0.000000	2.000000	NaN	NaN
75%	2.000000	3.000000	0.000000	0.000000	2.000000	NaN	NaN
max	2.000000	81.000000	6.000000	2.000000	12.000000	NaN	NaN

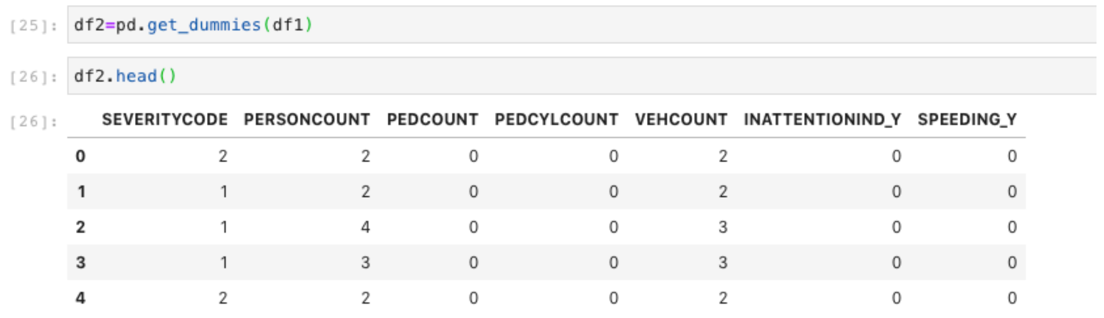
From this describe method, we can find that there were 29808 instances where driver inattention was a factor, and 9333 times where speeding was a factor.

2.2 Data Wrangling

Based on the above data descritions provided, it looks as if our data set is pretty clean already. We have 5 columns of intiger data, and 2 columns of text data.

The all columns of numerical data contain the same count of values, which indicates that we have no empty data columns for any of the rows. If we had empty data columns for Accident Severity, we would need to drop that row, as this is the variable we would like to predict.

The two columns containing text data, driver inattention and speeding, contain 29805 and 9333 entries, respectively. We can see that each column contains only one unique value, Y, which is what we would expect to see for this data type. As such, no manipulation needs to be done for this data. However, as we are going to try and model this data for predictions, it will be useful to cast the inattention and speed columns to intigers, where 1 is involvement, and 0 is no involvement. This was done using the Pandas get_dummies tool.



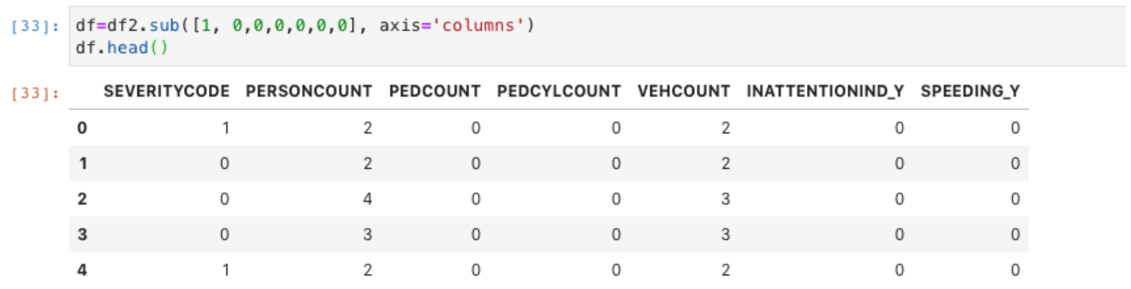
```
[25]: df2=pd.get_dummies(df1)
```

```
[26]: df2.head()
```

[26]:

	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND_Y	SPEEDING_Y
0	2	2	0	0	2	0	0
1	1	2	0	0	2	0	0
2	1	4	0	0	3	0	0
3	1	3	0	0	3	0	0
4	2	2	0	0	2	0	0

Since the outcome variable we wish to predict, accident severity, is represented by a value of either 1 or 2. Given that we want to predict the likelyhood of one of two outcomes, I chose to cast the column as a binary, with 0 meaning property damage, and 1 meaning injury. Since the data contained 1 or 2 as a value, a simple subtraction is all that is needed to accomplish this. This should simplify calculations later



```
[33]: df=df2.sub([1, 0,0,0,0,0,0], axis='columns')
      df.head()
```

[33]:

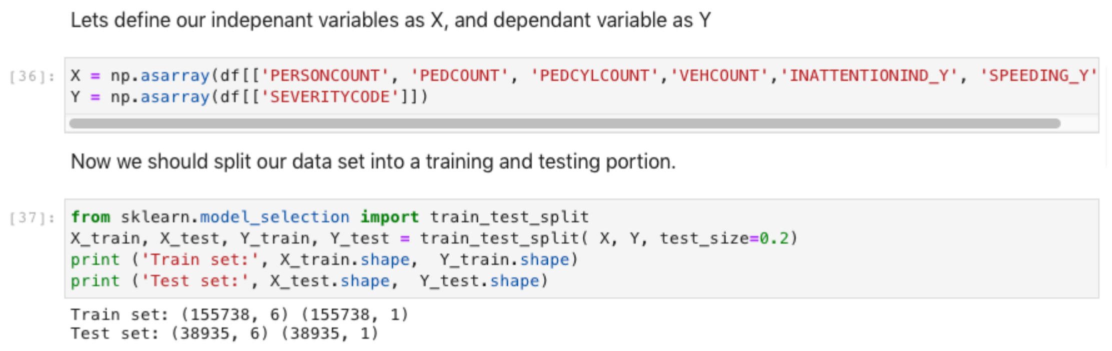
	SEVERITYCODE	PERSONCOUNT	PEDCOUNT	PEDCYLCOUNT	VEHCOUNT	INATTENTIONIND_Y	SPEEDING_Y
0	1	2	0	0	2	0	0
1	0	2	0	0	2	0	0
2	0	4	0	0	3	0	0
3	0	3	0	0	3	0	0
4	1	2	0	0	2	0	0

3 Methodology

Since we are looking for a binary outcome between two possibilities, it looks like a binary logistic regression would be the best fit.

To test that, I will first assign the X and Y variables as indepentant and dependant, respectively.

Since we have many accidents (194673) to use in our analysis, we should split our data sets into a training and testing set, so we can evaluate the performace of our potential model. I have chosen to use 80% of the data for training the model, and 20% for testing the model (Train set: (155738, 6) (155738, 1) Test set: (38935, 6) (38935, 1))



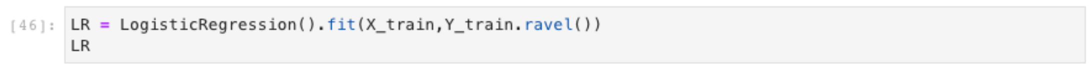
Lets define our indepenant variables as X, and dependant variable as Y

```
[36]: X = np.asarray(df[['PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT','VEHCOUNT','INATTENTIONIND_Y', 'SPEEDING_Y'
      Y = np.asarray(df[['SEVERITYCODE']])
```

Now we should split our data set into a training and testing portion.

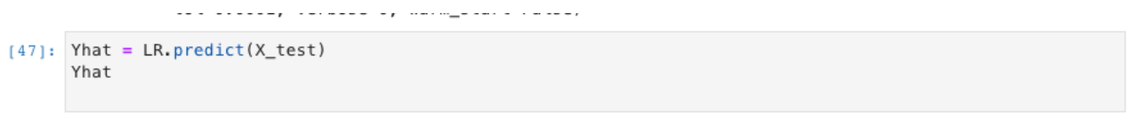
```
[37]: from sklearn.model_selection import train_test_split
      X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2)
      print ('Train set:', X_train.shape,  Y_train.shape)
      print ('Test set:', X_test.shape,  Y_test.shape)

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

Next, we are able to run our logistic regression, using the scikitlearn package installed at the beginning.



```
[46]: LR = LogisticRegression().fit(X_train,Y_train.ravel())
      LR
```

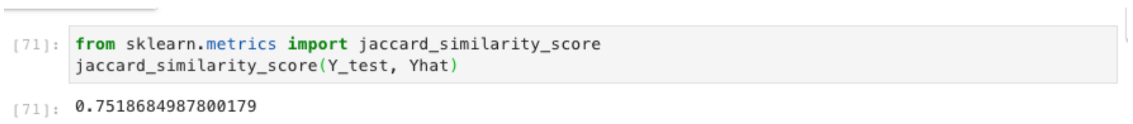
Now, we should run our test data through the regression model LR.



```
[47]: Yhat = LR.predict(X_test)
      Yhat
```

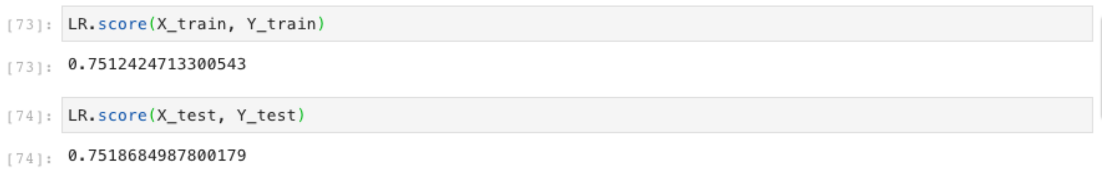
4 Model Evaluation and Results

Now that we have our predictions from our model, lets compare them to the actual outcomes, using the Jaccard Similarity from scikitlearn.



```
[71]: from sklearn.metrics import jaccard_similarity_score
      jaccard_similarity_score(Y_test, Yhat)

[71]: 0.7518684987800179
```

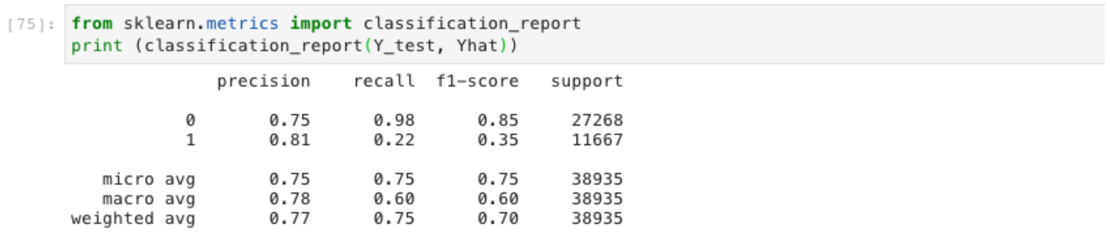
So, it looks like our Y__test data set and Yhat predictions from our model overlap 75% of the time. Lets check out the R^2 values between our test and training sets, and make sure that we aren't over or underfitting our model.



```
[73]: LR.score(X_train, Y_train)

[73]: 0.7512424713300543

[74]: LR.score(X_test, Y_test)

[74]: 0.7518684987800179
```

Based on the R^2 scores between the test and training set, we are getting approximately the same outcomes, so there isnt a substancial difference in the model's fit between the training and test data sets.

Lets take a look at the precision, recall and F1-scores of our model, using the scikitlearn classification report.



```
[75]: from sklearn.metrics import classification_report
      print (classification_report(Y_test, Yhat))

              precision    recall  f1-score   support

           0       0.75      0.98      0.85     27268
           1       0.81      0.22      0.35     11667

   micro avg       0.75      0.75      0.75     38935
   macro avg       0.78      0.60      0.60     38935
weighted avg       0.77      0.75      0.70     38935
```

It looks like the model is decent at correctly predicting property damage accidents and injury accidents (precision of 0.75 and 0.81 respectively) , but is squewed to not pick up all the correct injury accidents, instead misclassifying them as property damage (recall of 0.22).

The overall weighted F1-score is 0.7, so our model is overall, correct in predicting accident severity 70% of the time.

5 Discussion

Binary Logistic Regression appears to be a decent model to build to predict the severity of outcome between property damage and injury accidents, but it is stongly skewed towards predicting property damage accidents, over injury accidents.

The independant variables of number of people, pedestrains, cyclists and vehicles involved, along with driver inattention and speeding, are rather easy to collect, but given the overall 0.7 F1-score of our model, more indepth input variables are required to better predict when injury accidents will occur.

6 Conclusions

Most jurisdictions have laws against speeding, and against distracted driving, which is one of the catagories of driver inattention. Given this model, those regulations are validated by the accident data from Seattle.

7 Data Reference

The raw data can be found here:

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv

The metadata can be found here:

https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Metadata.pdf