Predicting the Severity of Road Traffic Accidents

Dmitry Vasilyev

September, 2020

1. Introduction

1.1. Problem

Traffic accidents are a significant source of deaths, injuries, property damage, and a major concern for public health and traffic safety. Accidents are also a major cause of traffic congestion and delay. Effective management of accident is crucial to mitigating accident impacts and improving traffic safety and transportation system efficiency. Accurate predictions of severity can provide crucial information for emergency responders to evaluate the severity level of accidents, locating and eliminating accident blackspots, estimate the potential impacts, and implement efficient accident management procedures.

1.2. Interest

Model can be applied to predictions of accident severity which is an essential step in accident management process. By recognizing those key influences, this report can provide suggestive results for government and local authorities to take effective measures to reduce accident impacts and improve traffic safety.

2. Data acquisition and cleaning

2.1. Data source

Dataset includes detailed information about collisions provided by Seattle Police Department and recorded by Traffic Records. It includes geo location, address type, weather and light conditions, time of the day and many others. Our dependent variable is severity code, from metadata file codes are divided into five categories, however dataset contains only two – property damage(1) and injury(2). This is cumulative dataset with time frame from 2004 till present.

2.2. Data cleaning

Multiple issues were found while working with the dataset.

First of all, several categorical features such as Address Type 'ADDRTYPE' and Collision Type 'COLLISIONTYPE' had number of missing values, around 2.5%, that I decided to replace with mode values.

Second, categorical features Junction Type 'JUNCTIONTYPE', Weather 'WEATHER', Road 'ROADCOND' and Light Conditions 'LIGHTCOND' already had ‘Uknown’ values, so all missing values were replaced as ‘Unknown’ .

Third, Inattention 'INATTENTIONIND', Speeding 'SPEEDING', Hit Parked Car 'HITPARKEDCAR' and Pedestrian Right Of Way Was Not Granted ‘PEDROWNOTGRNT’ features were converted from categorical to numeric values: ‘NaN’ or ‘N’ into 0, and ‘Y’ into 1. Column Under Influence ‘UNDERINFL’ had mixed type of values (NaN, ‘N’, ‘Y’, 0, 1), that were all converted into numeric.

Fourth, feature that reflects the data and time ‘INCDTTM’ of the incident was converted into datatime64 format to extract data easier.

After fixing missing values I checked for outliers in the dataset. First feature I looked at was Number of Persons involved in the Collision: there were around 237 (<0.1%) observations with more than 10 people, so for simplicity any value higher than 10 will be equal to 10. Same way I applied filter for number of vehicles involved in the incident, but limited it to 5. For incidents that involved pedestrians (‘PEDCOUNT’) and bicycles ‘PEDCYLCOUNT’ I used feature binarization: so either any number was in collision or not.

2.3. Feature selection

After data cleaning there were 194673 samples and 38 features. Upon examining the meaning of each feature, it was clear that there was some redundancy in the features. For example, a Collision code provided by SDOT and State Collision code contain very similar information, moreover feature Collision Type and Address type already had sufficient information related to the incident classification. So I decided to drop State Collision Code feature.

Some features were not relevant, contained only unique values related to each incident.

Table 1. Feature selection after data cleaning

Kept Features	Dropped Features	Reason
Numeric features: 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT'	‘X’, ‘Y’, ‘OBJECTID’, ‘INCKEY’, ‘REPORTNO’, ‘INTKEY’, ‘SDOTCOLNUM’, ‘ ‘EXCEPTRSNCODE’, ‘CROSSWALKKEY’, ‘SEGLANEKEY’	Irrelevant or unique data that has no correlation with target variable.
Binary features: 'INATTENTIONIND', 'UNDERINFL', 'PEDROWNOTGRNT', 'SPEEDING', 'HITPARKEDCAR'	SEVERITYCODE.1	Duplicate data
Categorical features: 'ADDRTYPE', 'COLLISIONTYPE', 'JUNCTIONTYPE', 'WEATHER', ROADCOND', 'LIGHTCOND', ‘LOCATION’, ‘SDOT_COLCODE’, ST_COLCODE’,	‘STATUS’, ‘EXCEPTRSNDESC’, ‘SEVERITYDESC’, ‘INCDATE’, ‘SDOT_COLDESC’	Duplicated data from other columns, or data is irrelevant.

3. Exploratory Data Analysis

3.1. Calculation of target variable

Severity code for each incident has multiple values according to the metadata file, however dataset contains only two values: 1 – any property damage, 2 – incidents that included injuries. Dataset contains 136485 observations with target value 1, and 58188 - with value 2. So we conclude that our dataset is imbalanced and we need to apply certain techniques to balance it before training the model. Our primary goal is to establish whether incident involved injuries (2) or not (1), so that local authorities can implement efficient accident management procedures.

Prediction of injury is the primary purpose of this model, so we have to reduce number of false positive results as much as possible. While increasing recall, we will most likely reduce the accuracy of the model: i.e. authorities might prefer dispatching emergency even though it's not needed.

3.2. Injuries ratio for SDOT Collision Codes

Let's see if there any specific SDOT collision codes that have high ratio of injuries and create new feature by applying function for anything that is higher than 50%.

```
v = df.groupby('SDOT_COLCODE').SEVERITYCODE.value_counts().unstack()
df['SDOT_INJRATIO'] = df.SDOT_COLCODE.map(v[2] / (v[2] + v[1]))
```

```
df.loc[ (df['SDOT_INJRATIO'] > 0.5),'SDOT_INJ'] = 1
df['SDOT_INJ'] = df['SDOT_INJ'].fillna(0)

df['SDOT_INJ'].value_counts()
```

```
0.0    182389
1.0     12284
Name: SDOT_INJ, dtype: int64
```

- And now let's review correlation between new binarized feature and target class

```
df[['SDOT_INJ', 'SEVERITYCODE']].corr().sort_values('SEVERITYCODE', ascending=False)['SEVERITYCODE']
```

```
SEVERITYCODE    1.000000
SDOT_INJ        0.330683
Name: SEVERITYCODE, dtype: float64
```

And we also say that there is a correlation between new feature and target class.

3.3. Number of collisions in each location.

I was trying to find black spots using ‘Location’ feature, as certain addresses had more than one collision incident reported, so another column with number of incident at the location was added for all samples : ‘NUMINCLOCATION’.
Distribution of collisions per location is shown below:

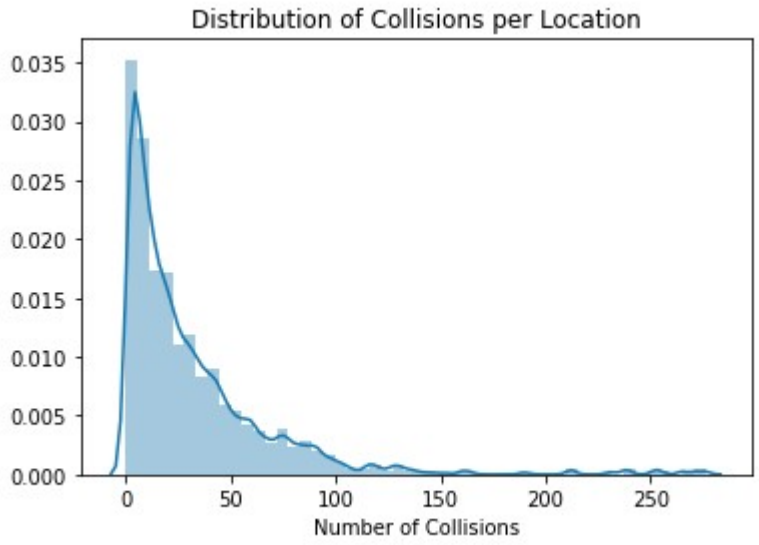

Figure 1. Distribution of Collisions per Location

Similar way we can find injury ratio for each location and create a new binary feature using following conditions:

- more than 5 collisions at the same place;
- more than 50% of collisions ended with injuries.

```
v = df.groupby('LOCATION').SEVERITYCODE.value_counts().unstack()
df['LOCATIONINJRATIO'] = df.LOCATION.map(v[2] / (v[2] + v[1]))
```

Now let's assume that any location that had more than 5 collisions and injury ratio is higher than 50% will be considered as risky.

```
df.loc[ (df['LOCATIONINJRATIO'] > 0.5) & (df['NUMINCLOCATION'] > 5) ,'INJLOCATION'] = 1
df['INJLOCATION'] = df['INJLOCATION'].fillna(0)

df['INJLOCATION'].value_counts()
```

```
0.0    175999
1.0     18674
Name: INJLOCATION, dtype: int64
```

3.4. Distribution of collisions during the week.

After creating another column ‘DAYOFWEEK’ containing Day of Week when incident happen, I was trying to find days with most collisions:

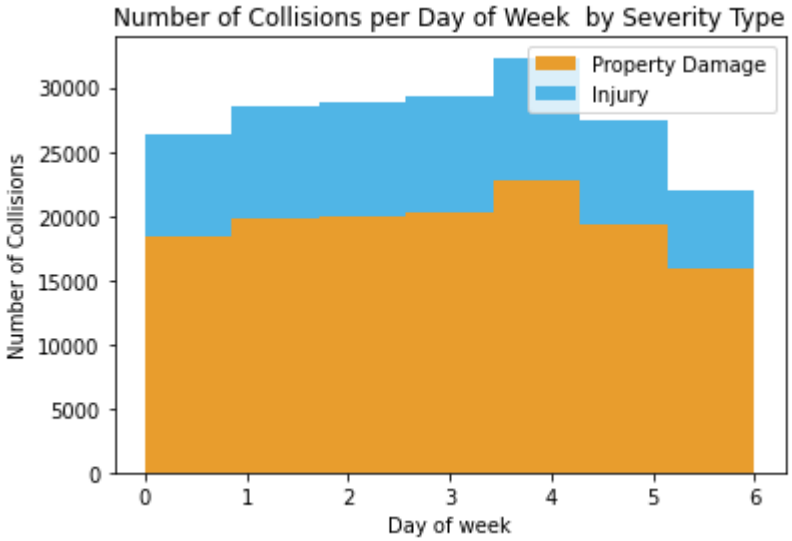

Figure 2. Distribution of Collisions During the Week

Despite higher number of collisions on Fridays, ratio between damages and injuries is constant during the week.

3.5. Distribution of collisions during the day.

After creating another column ‘INCHOUR’ containing hour of the day when incident happen, I was trying to find hours with most collisions:

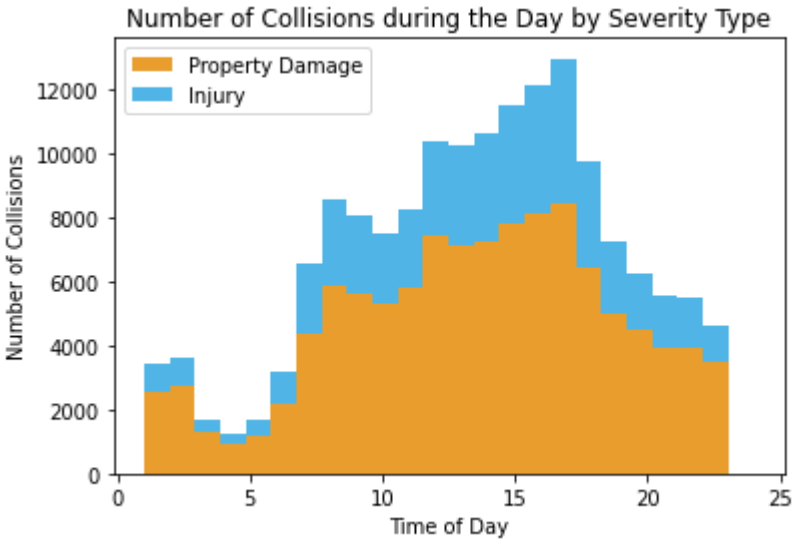

Figure 3. Distribution of Collisions During the Day

We see a spike of collisions during the day, so let's apply feature binarization for hours between 9:00 and 18:00. At the same time spike at midnight will not be taken into consideration as it also contains null values.

3.6. Number of persons involved in the collision by severity type

Majority of incidents that happened had 2 persons involved into the incident. Let's see the histogram showing number of persons by severity:

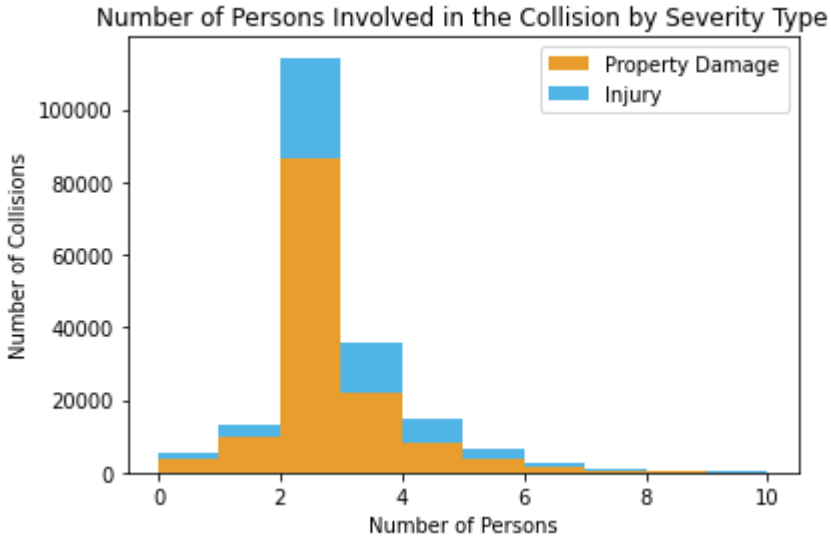

Figure 4. Number of Person Involved in the Collision by Severity Type

Let's assume that any value above 10 for 'Person Count' will be equal to 10 in order to remove outliers.

```
df.groupby('PERSONCOUNT')['SEVERITYCODE'].value_counts(normalize=True)
```

```
PERSONCOUNT  SEVERITYCODE
0            1               0.682179
             2               0.317821
1            1               0.749430
             2               0.250570
2            1               0.756537
             2               0.243463
3            1               0.621382
             2               0.378618
4            1               0.570600
             2               0.429400
5            1               0.549058
             2               0.450942
6            2               0.502221
             1               0.497779
7            2               0.563218
             1               0.436782
8            2               0.532833
             1               0.467167
9            2               0.597222
             1               0.402778
10           2               0.512329
             1               0.487671
Name: SEVERITYCODE, dtype: float64
```

Table 2. Grouped Number Of Persons Involved in the Collision Per Severity

3.7. Number of pedestrians involved in the collision by severity type

As we see from the graph below feature binarization can be applied to that feature: so either pedestrians were involved in the collision or not.

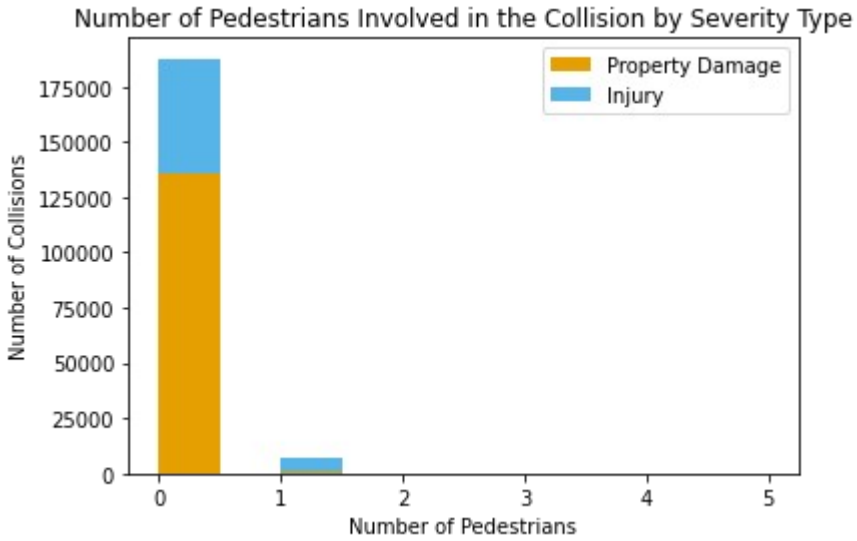

Figure 5. Number of Pedestrians Involved in the Collision by Severity Type

3.8. Number of bicycles involved in the collision by severity type

Same way we can apply feature binarization for Number of Bicycles involved in the collision column.

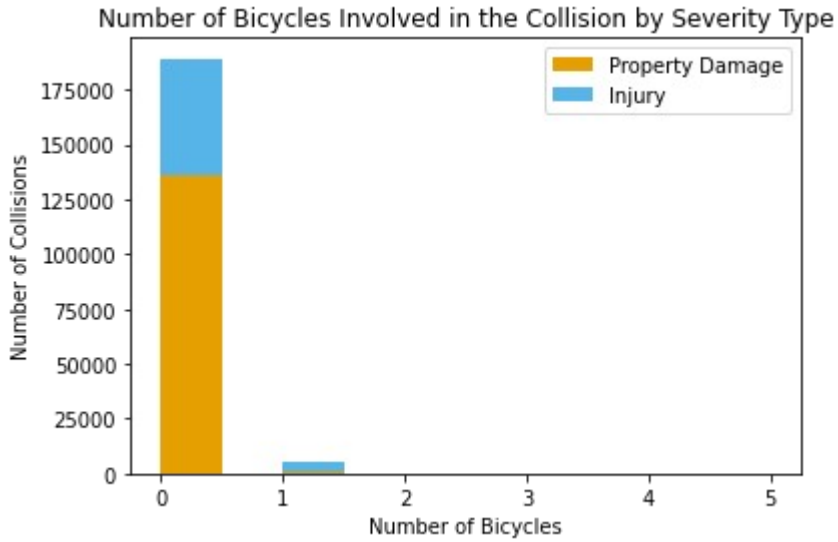

Figure 6. Number of Bicycles Involved in the Collision by Severity Type

3.9. Number of vehicles involved in the collision by severity type

Let’s see outliers in number of vehicles in the collision by severity type.

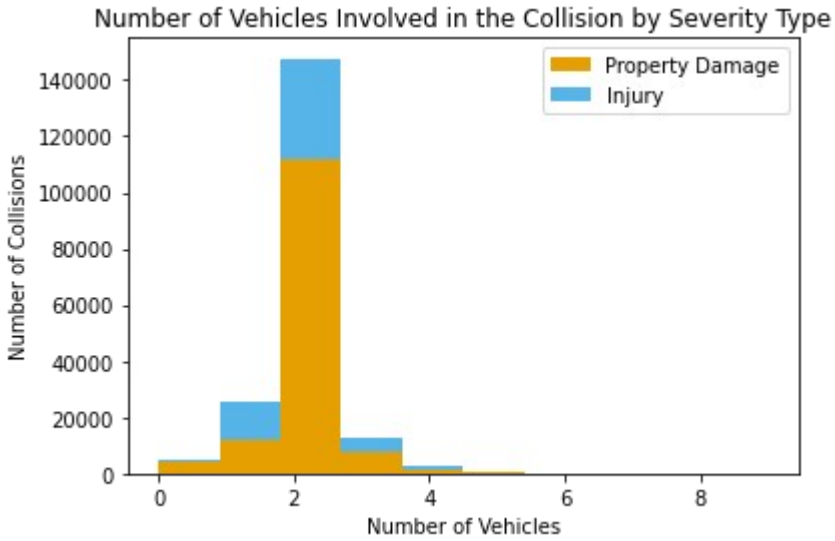

Figure 7. Number of Vehicles Involved in the Collision by Severity Type

We will assume that any value above 5 for 'Person Count' will be equal to 5 in order to remove outliers.

3.10. Feature Selection After Exploratory Data Analysis

Table 2. Feature selection after data exploratory

Kept Features	Dropped Features	Reason
Numeric features: 'PERSONCOUNT', 'PEDCOUNT', 'PEDCYLCOUNT', 'VEHCOUNT',		
Binary features: 'INATTENTIONIND', 'UNDERINFL', 'PEDROWNOTGRNT', 'SPEEDING', 'HITPARKEDCAR' 'SDOT_COLCODE', 'SDOT_INJRATIO', 'LOCATIONINJRATIO', 'RHOUR'		
Categorical features: 'ADDRTYPE', 'COLLISIONTYPE', 'JUNCTIONTYPE', 'WEATHER', ROADCOND', 'LIGHTCOND',	‘LOCATION’, 'SDOT_COLCODE'	Replaced by 'LOCATIONINJRATIO' after counting number of incidents at the location and injury ration. Replaced by 'RSDOT_COLCODE'

One hot encoding technique was applied to all categorical features in order to have each features to be represented in a binary format.

Our final dataset has **194673** observations and **61** features.

4. Predictive Modeling

Since we are dealing with typical classification problem, there are several algorithms that can help us to build model:

- K Nearest Neighbor (KNN)
- Decision Tree
- Support Vector Machine
- Logistic Regression

Just because the size of our dataset is quite big, building model with KNN and SVM might take significant amount of time without optimizing the dataset: like under-sampling, removing some features, etc. So I decided to start with simple Logistic Regression and Decision Tree Classifier. We also need to keep in mind that our dataset is imbalanced: number of accidents with injuries is several times less than with damages.

4.1. Data normalization

Following numeric features were standardized before building the model: ‘PERSONCOUNT’, ‘VEHCOUNT’.

4.2. Decision Tree Classifier

I used standard loop to find the best f1-score with different tree depth. After classification report and confusion matrix were build.

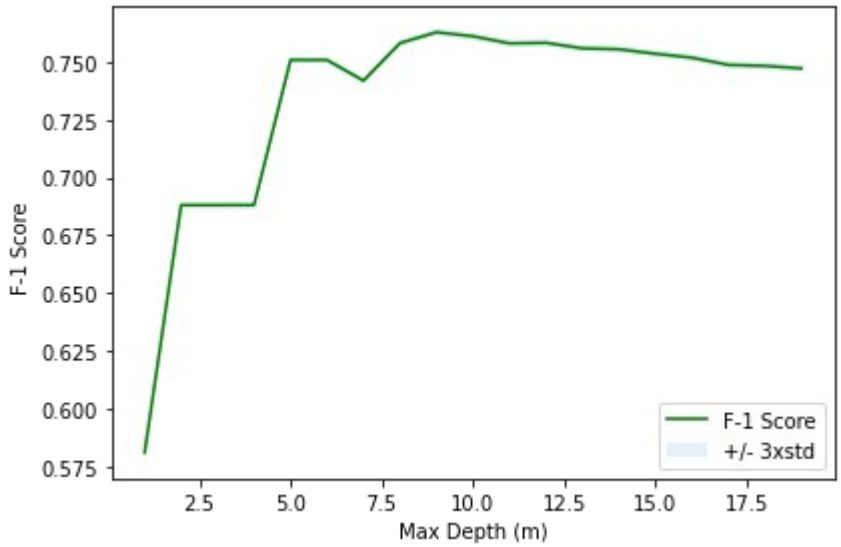

Figure 8. F-1 score for different Max Depth numbers

As we see from confusion matrix predictions of injuries, which is more important than prediction of collisions, are quite poor on imbalanced dataset. Let's see if other algorithms can perform better.

	precision	recall	f1-score	support
1	0.80	0.90	0.85	41083
2	0.67	0.47	0.56	17319
accuracy			0.78	58402
macro avg	0.74	0.69	0.70	58402
weighted avg	0.76	0.78	0.76	58402

```
Confusion matrix, without normalization
[[37092  3991]
 [ 9108  8211]]
```

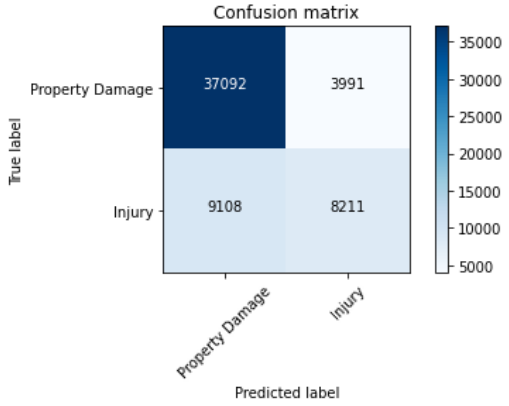

Figure 9. Decision Tree Classifier Confusion Matrix

4.3. Logistic Regression

Just like with previous algorithm Logistic Regression showed quite poor result predicting injuries on imbalanced dataset.

	precision	recall	f1-score	support
1	0.80	0.92	0.85	41083
2	0.69	0.44	0.54	17319
accuracy			0.77	58402
macro avg	0.74	0.68	0.69	58402
weighted avg	0.76	0.77	0.76	58402

```
Confusion matrix, without normalization
[[37596  3487]
 [ 9659  7660]]
```

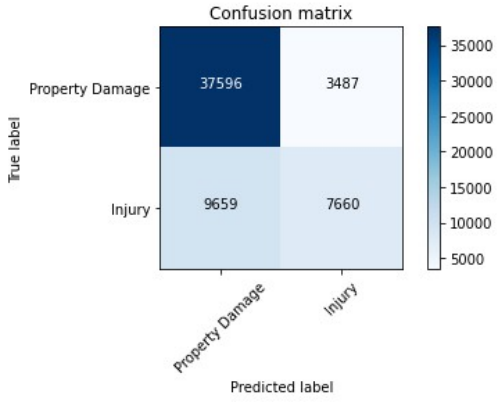

Figure 10. Logistic Regression Confusion Matrix

4.4. Balancing Classes

There's several methods for balancing classes:

- **Random-Undersampling of Majority Class.** You reduce the size of majority class to match size of minority class. Disadvantage is that you may end up with very little data.

- **SMOTE- Synthetic Minority Oversampling Technique.** Algorithm that creates a larger sample of minority class to match the size of majority class.

- **Inverting Class Ratios. (Turning minority into majority).** If you turn the minority into the majority, you may skew results towards better recall scores(detecting injuries correctly), as opposed to better specificity scores.(detecting damages correctly)

Dataset was balanced with a variant implementation of SMOTE, to see correlations.

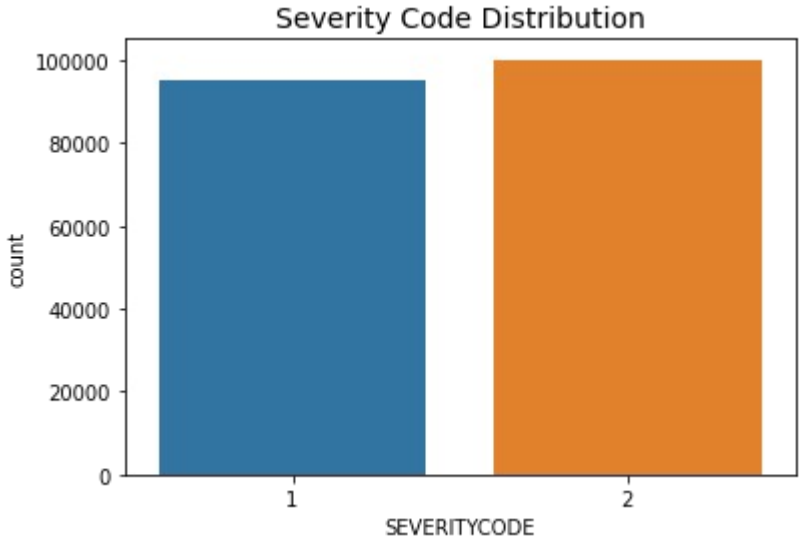

Figure 11. Severity Code Distribution After Applying SMOTE

4.5. Removing High-Correlated Outliers

This step must be taken after balancing classes. Otherwise, correlations will echo class-distributions. Based on a correlation matrix, we'll identify features with high correlations, and remove any collisions with outlying values in these. High correlation features have a high capacity to influence the algorithm predictions. Therefore it's important to control their anomalies.
This approach will reduce prediction bias because our algorithm will learn from more normally-distributed features.

Approach to removing outliers:

- For features of high positive correlation - remove damage outliers on the top range, (improve recall) and remove injury outliers on the bottom range. (improve specificity)
- For features of high negative correlation - remove damage outliers on the bottom range, (improve recall) and remove injury outliers on the top range. (improve specificity)

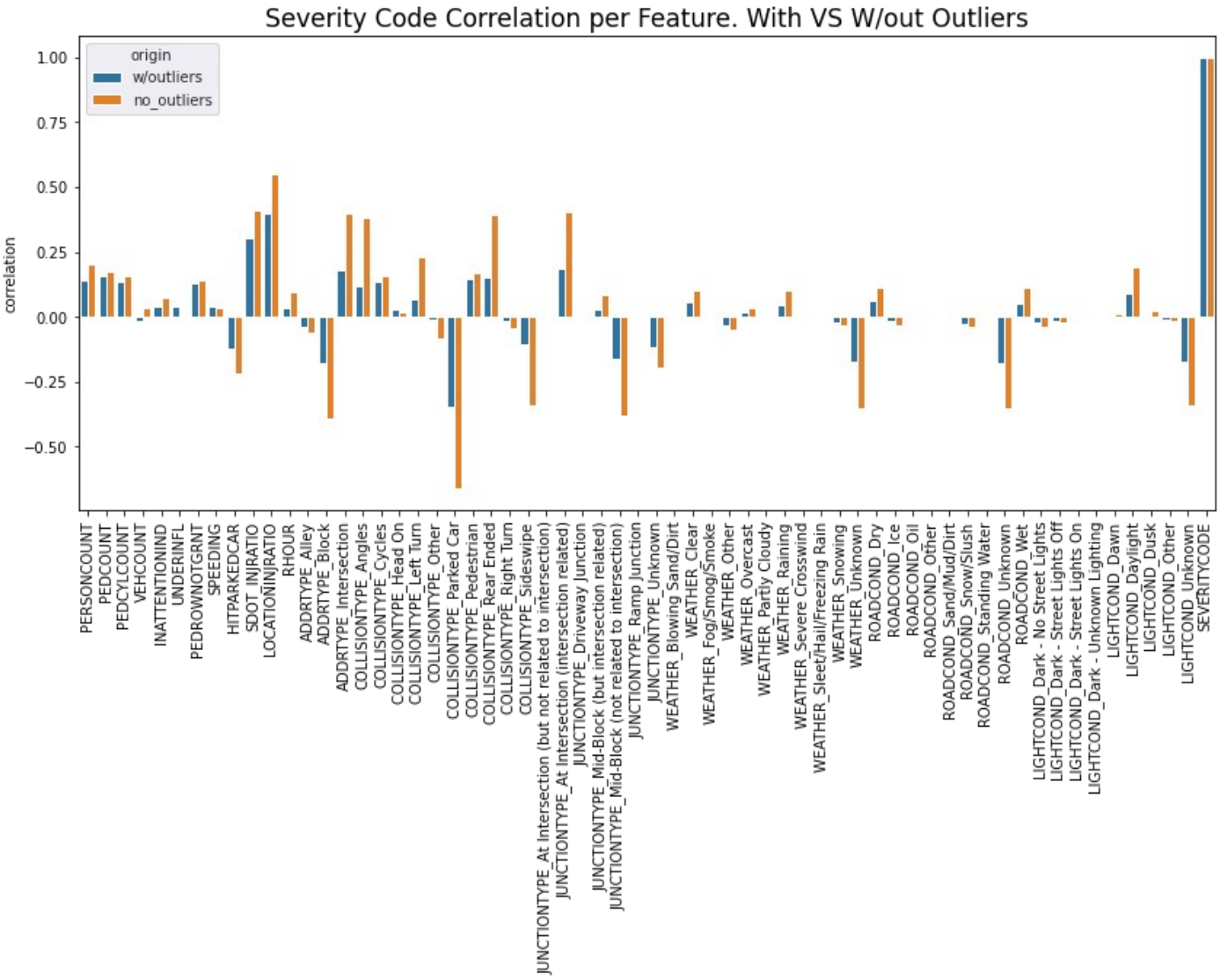

Figure 12. Severity Code Correlation Per Feature With vs Without Outliers

It's obvious that most features gained correlation power, regardless of direction. Positive correlations went higher up, negative correlations went lower down. Also, the highest correlations flattened out, while the smallest ones rose to relevance.

It is clearly an indicator that the outliers were causing noise, and therefore dimming the correlation-potential of each feature.

4.6. Test and Compare Classifiers

We'll evaluate improvements based on injury recall, since its crucial to predict injuries. This might come at the expense of more false-alarms, which would decrease the overall accuracy. The main purpose of this project will be to identify all injuries, while minimizing false-positives.

Let’s define outcomes and predictors, reduce model size, and classify.

	Train_Recall	Test_Recall	Test_Specificity
SVC_default	0.229751	0.602001	0.602001
LogisticRegression_default	0.66325	0.663559	0.663559
DecisionTreeClassifier_default	0.6989	0.513838	0.513838
KNeighborsClassifier_default	0.63355	0.619916	0.619916

Figure 13. Performance of Classifiers with Default Parameters

4.7. Final model and Grid Search

Results above might not be very promising, I’ve chosen Logistic Regression and now use GridSearch to find the best parameters for this classifier. Let’s review confusion matrix after finding the best parameters:

	precision	recall	f1-score	support
1	0.96	0.38	0.54	41083
2	0.40	0.96	0.56	17319
accuracy			0.55	58402
macro avg	0.68	0.67	0.55	58402
weighted avg	0.79	0.55	0.55	58402

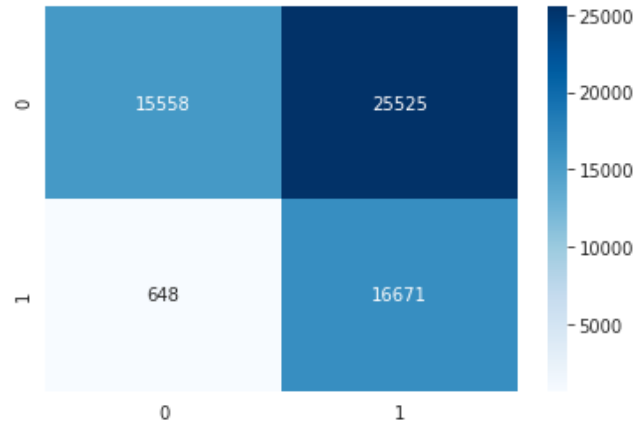

Figure 14. Logistic Regression Confusion Matrix after Balancing and Optimizing Data

5. Conclusions

In this study we analyzed relationship between different car collisions features and severity of the accident. Number of features such as SDOT collision code or Location have quite high correlation with target variable. We tried to tweak our model to increase injury **recall** and it reached 0.96, since its crucial to predict injuries. This has came at the expense of more false-alarms, which decreased the overall accuracy. As the main purpose of this project is identify all injuries, while minimizing false-positives, this model can now be used to predict injuries of any collision. Local and government authorities can decide how to balance the model: use it with less accuracy but higher recall, covering as many injury cases as possible, for example, dispatching emergency services, or use it with more accurate model, that will cover less accidents with injuries.

6. Future directions

I was able to increase the recall of injuries from 0,44 to 0,96 after balancing and optimizing the dataset. But that came at a high cost of false-alarms. In the scope of this study I've only used SMOTE to balance the dataset, so there is plenty of room to try other techniques and find the most accurate one while keeping the recall score for injuries at the same level.