Car accident severity project - Seattle

Course/IBM Data Science capstone project

By Stein Myhre, 18.10.2020

Contents

1. Introduction

Road traffic collisions are a leading cause of death in the United States for people aged 1–54[1]. To reduce this problem road planers, law enforcement officers, and others need decision support from data scientists. Vast amount of data exists about the nature of accidents and how they occur, but modern data science techniques most be applied to go from information overload to insight and understanding.

This project aims to build a model that predicts, using selected features, what the severity of an accident will be. This model can then be used by responsible authorities to simulate different type of accidents by adjusting features such as weather conditions, road conditions, and if the driver is speeding or not. From these simulations decision-makers can select the optimal strategy to reduce the severity of accidents in Seattle.

As an extra bonus the findings will also be valuable for car drivers, enabling them to predict in which situation they are most likely to be in a severe accident.

2. Data collection and understanding

In this submission I will be using a dataset of collisions in Seattle from 2004 to 2020. The dataset contains 194.673 accidents described with 38 features (before adjusting for potential duplicates). These features include a categorical severity code, geolocation, information on the type of collision, how many where involved in the accident, in what type of junction it occurred, if the driver was under the influence or inattentive, weather, road, and light conditions and more.

The severity code is a categorical variable with 1 representing "property damage", and 2 representing "injury collision". This will be the variable that the model will attempt to predict using selected features.

2.1 Feature selection

For this model I will be using three features describing internal driver factors:

Feature name	Feature description
INATTENTIONIND	Whether or not collision was due to inattention. (Y/N)
UNDERINFL	Whether or not a driver involved was under the influence of drugs or alcohol.
SPEEDING	Whether or not speeding was a factor in the collision. (Y/N)

I will also be using three features that are external to the driver:

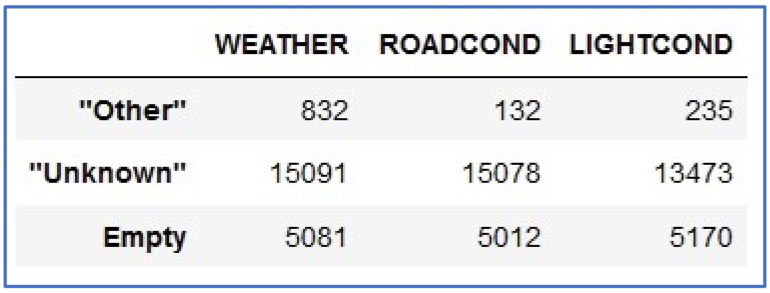
WEATHER	A description of the weather conditions during the time of the collision.
ROADCOND	The condition of the road during the collision.
LIGHTCOND	The light conditions during the collision.

2.2 Data cleaning

[1] https://www.cdc.gov/injury/features/global-road-safety/index.html

As with all real-world data the quality of the data is varying and needs cleaning before applying machine learning algorithms. The first problem to address is that of zero values in the external features (WEATHER, ROADCOND, LIGHTCOND).

Zero values in external features is represented as “Other”, “Unknown”, or empty cells:



	WEATHER	ROADCOND	LIGHTCOND
"Other"	832	132	235
"Unknown"	15091	15078	13473
Empty	5081	5012	5170

Figure 1: Showing how many rows have missing values

Dropping the zero values will reduce the dataset with 24.716 rows, a reduction of 12.7%. It will probably have some effect on the how well the machine learning algorithms perform, but for now I am willing to make the sacrifice. If on a later stage I assess that the performance is to low, I will come back and adjust by randomly assigning values by frequency, or simply choosing the most common value.

The feature WEATHER has these categories of labels, ranging from most common to least common:

Clear

Raining

Overcast

Snowing

Fog/Smog/Smoke

Sleet/Hail/Freezing Rain

Blowing Sand/Dirt

Severe Crosswind

The other external features, ROADCOND and LIGHTCOND, are likewise categorical and will be prepared with one hot encoding before being used for modelling. Testing shows that it gives better performance on all machine learning algorithms. One hot encoding works by converting the categorical values to row labels with binary values. With one-hot encoding WEATHER: “Raining” will be represented like this:

Clear	Raining	Overcast	Snowing	Fog/Smog/Smoke	Sleet/Hail/Freezing Rain
0	1	0	0	0	0

For internal features (“NATTENTIONIND, UNDERINFL, SPEEDING) the data is cleaned by converting «Y» to 1, «N» to 0. For empty values I will assume that these are negative and will assign 0. Lastly names of labels are changed and will now be addressed as INATTENTION, DUI, and SPEEDING.

Dependent variable SEVERITYCODE is converted to categories 0 - “property damage” - and 1 - “injury collision”.

3. Methodology

3.1 Exploratory data analysis

A dataset is biased if one of the categories in the dependent variable are overrepresented in the dataset. In our case the dependent variable is SEVERITYCODE. A histogram of SEVERIYCODE shows that category 0 – “property damage” is overrepresented, and dataset is unbalanced.

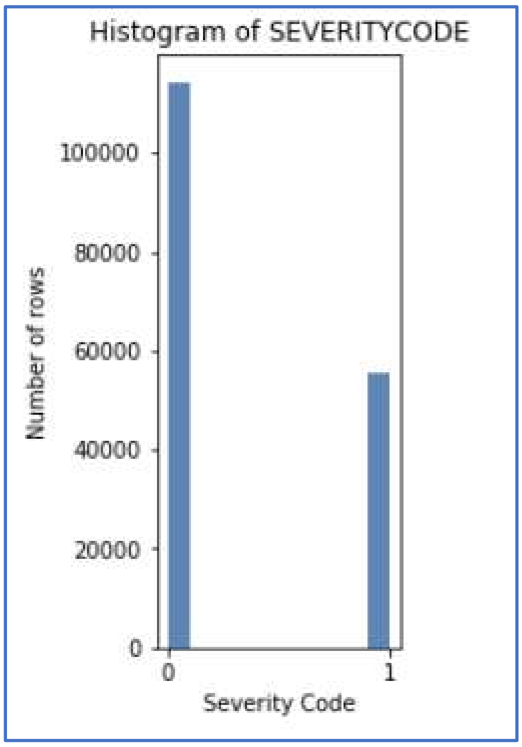


Figure 2: Histogram of SEVERITYCODE.
0 = “property damage” and 1 = “injury collision”

Common wisdom should dictate that features SPEEDING, DUI, and INATTENTION should predict an increased likelihood of severe accidents (SEVERITYCODE=1). A pandas dataframe derived from our dataset indicates that this is true:

SEVERITY	DUI	Not DUI	SPEEDING	Not SPEEDING	INATTENTIVE	ATTENTIVE
0	60.9%	67.6%	61.9%	67.5%	64.2%	67.9%
1	39.1%	32.4%	38.1%	32.5%	35.8%	32.1%

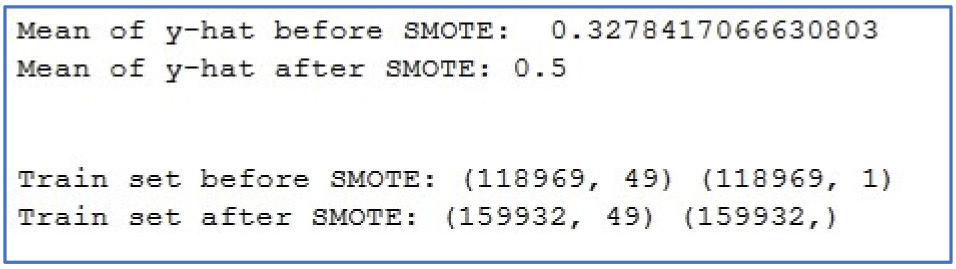
From this we can conclude that driving while under the influence (DUI) correlates to a 6.7 percentage points increase in accidents becoming severe, speeding 5.6 percentage points increase, and inattention increases the correlation with 3.7 percentage points.

3.2 Test/train split and SMOTE balancing

Balancing should be done after test/train splitting in order to avoid overfitting. I choose to use 70% of the data for training, and 30% for testing.

SMOTE, or Synthetic Minority Over-sampling TEchnique, is an algorithm that creates additional rows of the minority class using statistical techniques. It is a good alternative to simply duplicating instances of the minority class. I will use SMOTE method from the Imbalanced-Learn library.

This is the result of an analysis of the datasets before and after SMOTE balancing:



```
Mean of y-hat before SMOTE:  0.3278417066630803
Mean of y-hat after SMOTE: 0.5


Train set before SMOTE: (118969, 49) (118969, 1)
Train set after SMOTE: (159932, 49) (159932,)
```

Figure 3: analysis before and after SMOTE

A balanced binary class, where there are equal parts of 1 and 0, should result in a mean of 0.5. We can therefor conclude that dataset is balanced after SMOTE. We can also see that the balanced training has 159.932 rows, while the unbalanced dataset has 118.969.

3.2 Machine learning algorithms, selection and brief explanation

For this project I will use Decision tree, Logistic Regression, and K-nearest neighbor. Support vector machine was also considered but was found to be unsuitable because of the size of the dataset.

A decision tree is built by calculating each attributes significance relative to the target. The most significant attribute is split into the unique values of that attribute. Each split is a node, and each leaf is class. This is done until it can give a prediction of the target.

Logistic Regression involves using a logistic function to predict a target variable. It is often used to predict binary targets, such as the target in this project. It also enables us to calculate the probability of the prediction.

K-nearest neighbor calculates the distance from other datapoints, and then predicts the target variable based on the closest neighbors/datapoints. The number of neighbors a model uses is called the “K”. Finding the best K can be achieved by building several models that uses different K’s, and then selecting the models that gives the best accuracy.

4. Results

4.1 Metrics and scoring – brief explanation

In the following I will evaluate each machine learning techniques using a few different metrics from Scikit-Learn. Measurements will be done on models built from the training data, and then tested on the test data. I will be using accuracy, recall, precision, F1, and confusion matrix.

Accuracy is how often the predicted label corresponds to the actual label. An accuracy of 1 means 100% correlation between prediction and actual label, while an accuracy of 0.5 would be 50%.

Recall and precision give score based on how many false positives and false negatives a model produces. A true positive is a correctly predicted positive value, a false positive is a incorrectly predicted positive value. Likewise, a true negative is a correctly predicted negative value, and a false negative is a incorrectly predicted negative value.

Recall is the number of true positives divided by the number of true positives + number of false positives. Precision is the number for true positives divided by true positives + number of false negatives. A F1 score is based on both recall and precision.

For each model I will show a confusion matrix. A confusion matrix counts the number of true positives, true negatives, false positives, and false negatives, and displays thenm in a format like this:

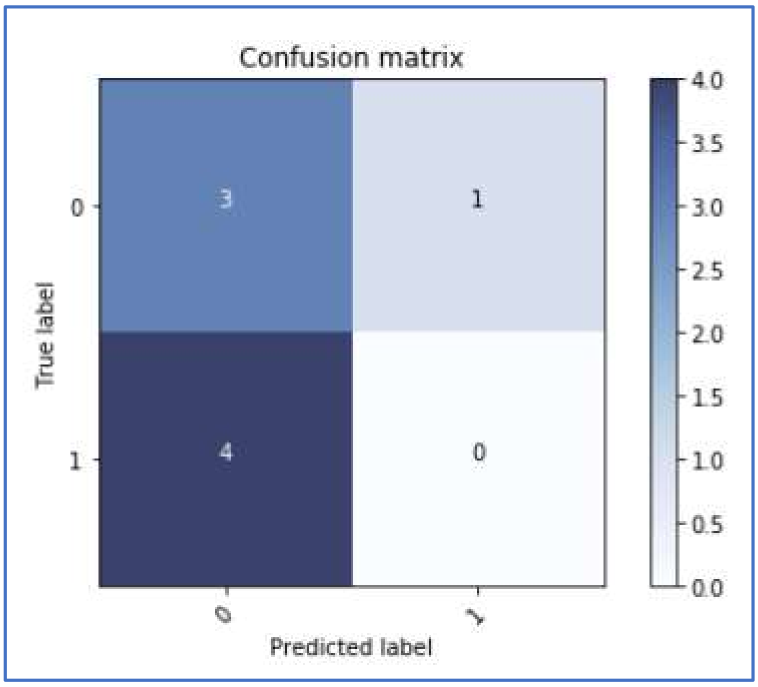


Figure 4: Confusion matrix example[2]

In this example the model has predicted 3 true negatives, 1 false positive, 4 false negatives, and 0 true positives.

[2] Credit to Saeed Aghabozorgi for writing the plot_confusion_matrix function

4.2 Decision tree

The Scikit-Learn Decision Tree classifier predicts the severity of an accident with an accuracy of 0.577 with an F1-score of 0.578. From the Confusion matrix we can see that it does fairly well at predicting true negatives, however it does less well on predicting true positives. The decision tree model is biased towards predicting SEVERITYCODE 0 - "property damage".

Classification report:

	Precision	Recall	F1-score	Support
0	0.69	0.67	0.68	34308
1	0.36	0.38	0.37	16680
accuracy			0.57	50988
macro avg	0.52	0.53	0.52	50988
weighted avg	0.58	0.57	0.58	50988

Confusion matrix:

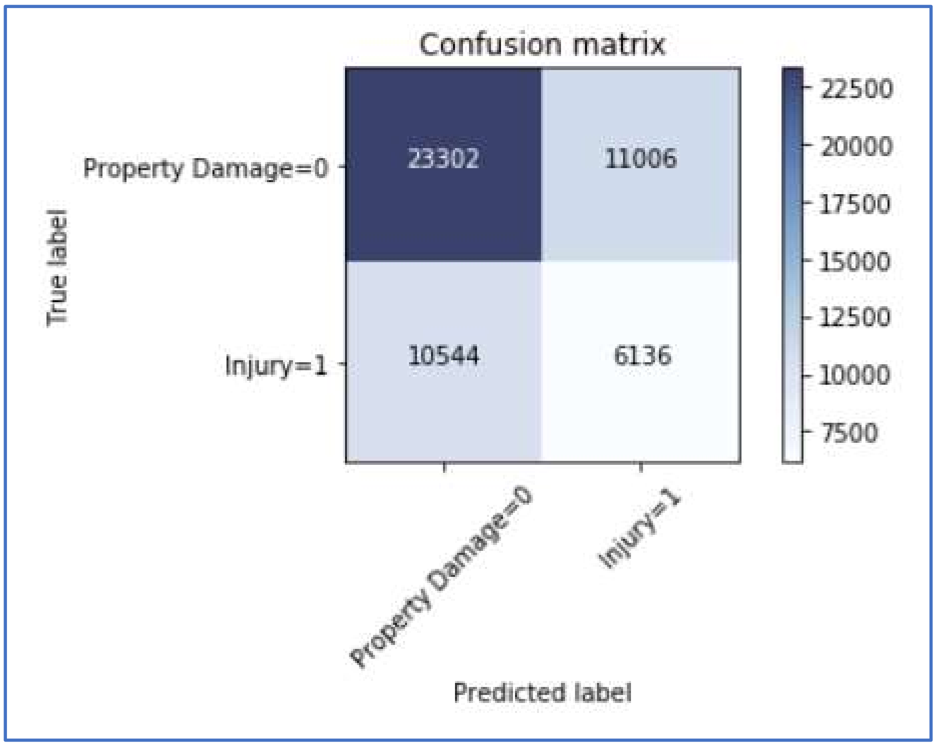


Figure 5: Confusion matrix – Decision Tree model

4.3 Logistic regression

The Scikit-Learn Logistic Regression classifier performs on-par with Decision Tree classifier with an accuracy of 0.586 and F1-score of 0.584. It also leans towards predicting more negatives, meaning more less severe accidents predicted.

Classification report:

	Precision	Recall	F1-score	Support
0	0.69	0.70	0.69	34308
1	0.36	0.35	0.36	16680
accuracy			0.59	50988
macro avg	0.53	0.52	0.52	50988
weighted avg	0.58	0.59	0.58	50988

Confusion Matrix:

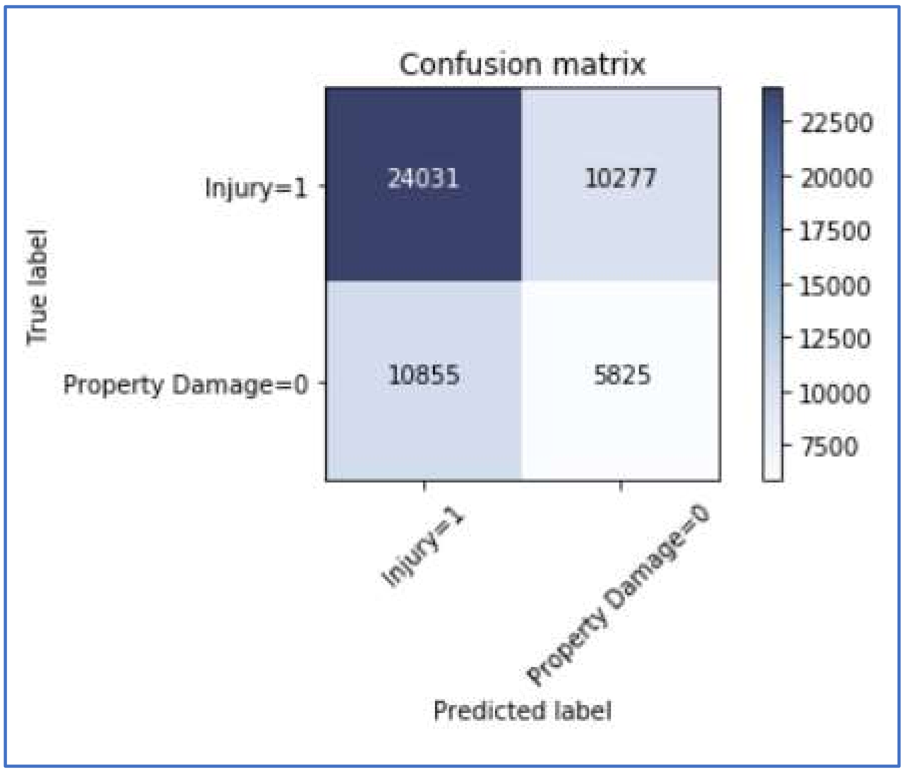


Figure 6: Confusion matrix – Logistic Regression model

4.4 K-nearest neighbor

The accuracy of Scikit-Learn K-nearest neighbor classifier varied from 0.589 for K=7, to 0.670 for K=10. However, a model trained with K=7 performs well because it simply predicts almost exclusively SEVERITYCODE to be 0. Since the injury collisions are rather rare, the accuracy gets high but false negatives are plentiful.

If we however measure the model's performance with recall, we will emphasis the ability to avoid false negatives. Doing so we will find that selecting K=7 gives us the highest recall, even though it has the lowest accuracy. The low accuracy raises the question if the model should simply be dropped. However, if used in conjunction with logistic regression or decision tree, there might be some value in this model.

Classification report (for K=7):

	Precision	Recall	F1-score	Support
0	0.68	0.55	0.61	34308
1	0.33	0.46	0.39	16680
accuracy			0.52	50988
macro avg	0.50	0.51	0.50	50988
weighted avg	0.56	0.52	0.54	50988

Confusion Matrix (for K=7):

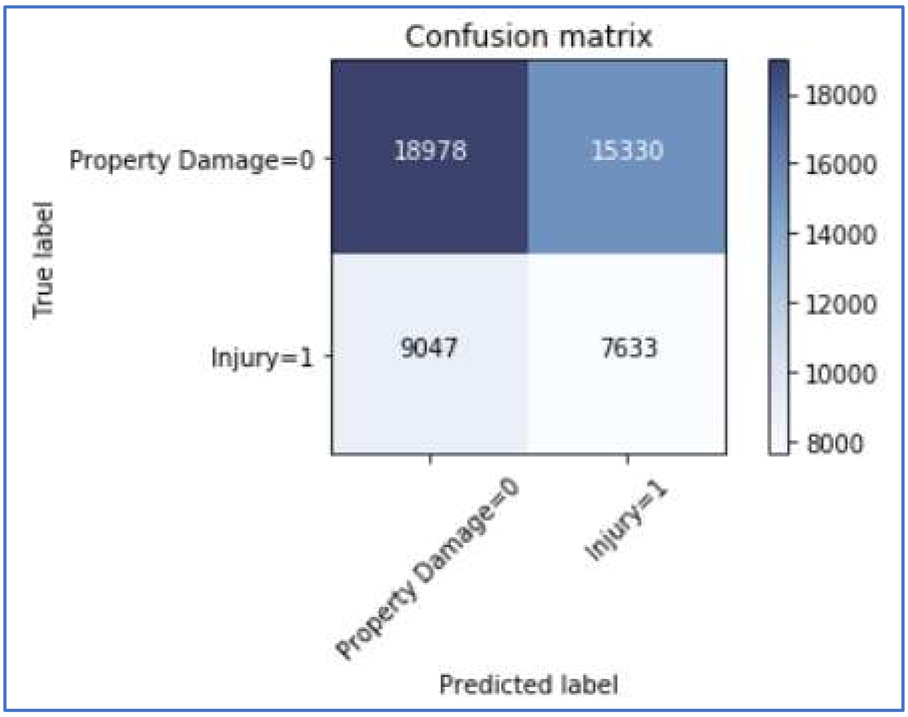
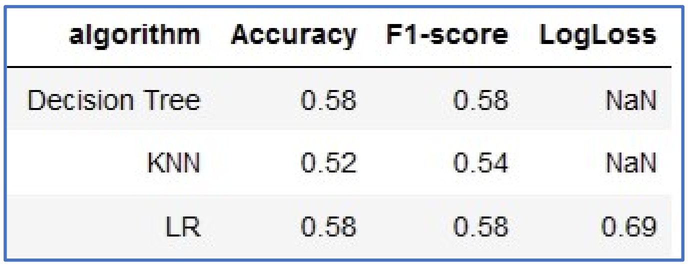


Figure 7: Confusion matrix – K nearest neighbor

5. Discussion



algorithm	Accuracy	F1-score	LogLoss
Decision Tree	0.58	0.58	NaN
KNN	0.52	0.54	NaN
LR	0.58	0.58	0.69

Figure 8: Summary of model accuracy and F1-score

5.1 Choosing the machine leaning algorithm

Whether one should choose a model with a bias towards false negatives or one with a bias towards false positives, is ultimately up the client/decision maker. False negatives could result in not predicting a collision that leads to loss of life, but false positives can result in the inefficient use of resources. By not effectively using resources to mitigate severe collisions, false positives can also lead to loss of life. The customer needs to understand that not all mistakes are not equal. A decision must be made for whether one wants an alarmist model, or a model that sometimes misses a severe accident.

In this project K-nearest neighbor is the alarmist model, with a recall of 0.46 compared to 0.35 (Logistic Regression) and 0.38 (Decision Tree). It does however have a high rate of false positives, and therefore gets a lower F1 score then the other two.

Logistic Regression and Decision Tree perform with the same accuracy rate and F1-score, and they are preferable to K-nearest neighbor.

All models perform better than chance, but more should be done to increase their performance. This project is sufficient to deliver a proof of concept to the costumer, but more should be done to get better predictions. More data may be used from the existing dataset, and more data should be collected from accidents.

5.2 Recommendations to stakeholders

- The models should be used by government officials for data-driven strategies for countering severe accidents on the roads of Seattle. Simulations can indicate if a particular intersection or highway are needs better lighting or better road conditions. It can also show the impact of DUI, speeding, and drivers inattentiveness.

- Driving while under the influence had the highest correlation with severe accidents. Assuming that DUIs are relatively infrequent compared to speeding and inattentive drivers, one should conclude that this a leading cause for severe accidents. Law enforcement and public official should make a note of this, as well as the individual drivers.

- Speeding and inattentive drivers does also affect the chances of an accidents being severe

- 12.7% of the data was dropped because of missing values. First responders should be encouraged to report accurate data. They need to understand that reporting data could save lives.

- Collecting more data can increase the effectiveness of the models. The drivers age, how long he/she have been driving, and the type of car, are all examples of relevant and probably significant data.

6. Conclusion

This project is meant to give decision support to officials with the responsibly of road safety in Seattle. It achieves this objective by creating a model that allows the decision maker to simulate accidents, and the get a prediction of whether this accident would be severe or not.

Data was collected from Seattle accidents from 2004 to 2020. This data was then cleaned and prepared for machine learning algorithms such as Decision Tree, K-nearest neighbor, and Logistic Regression. Decision Tree and Logistic regression where the preferable models. On a later stage they may on be used in conjunction with K-nearest neighbor.

Several recommendations have been given to the decision makers. Advice are also given to the individuals drivers.

This project should be considered a proof of concept, and further work is needed to make the models more accurate.