Predicting Accident Severity in Great Britain

IBM Data Science Professional Certificate Capstone Project

Anne Bode, September 2020

1. Introduction

Certain everyday decisions come with a calculated risk, whether conscious or unconscious. It is relatively safe to say that the decision to get behind the wheel of a car and drive (alongside fellow cars) at speeds upwards of 60 mph comes with an inherent risk. However, more often than not we decide that the risk of getting into a fatal accident is outweighed by the reward of getting to where we need to be. Some of these calculations prove to be woefully inaccurate, with the result a matter of life and death.

What if we could enhance our subconscious decision-making with a tool that would allow us to look out the window, contemplate our route, and decide whether we still like the odds? A tool like the one described could both save lives and improve traffic conditions, by reducing our likelihood of getting into a fatal accident… and causing a bumper to bumper headache for fellow drivers while we are at it. In this project, we look at accident severity data in Great Britain, specifically.

2. Data

The dataset used in this project is provided by the UK's Department of Transport and can be found here. The "Road Safety Data – Accidents 2018" CSV file includes data on personal injury road accidents in Great Britain between 2005 and 2018. Additional information about the variables included in the dataset can be found here. Our goal is to use certain variables to predict how severe an accident would be, should a particular driver get into one over the course of her or his drive.

2.1 Data Pre-Processing

We are looking to create a model that will *predict* accident severity *throughout Great Britain.* Therefore, only a select number of variables within this dataset will be useful as predictive criteria for a generalized model. For example, latitude, longitude, and police force all have to do with specific location and therefore would not allow us to create a generalized tool for the entire Great Britain population. Separately, a variable such as whether a police officer arrived at the scene would not help us create a predictive tool, as this is likely backwards looking, with police officers more likely to be on scene for more severe accidents. Consequently, we will restrict our dataset to just include features which an end user might reasonably be able to plug in *before* embarking on his or her trip, regardless of where in GB she or he may live. These include features such as the Day of the Week, Light/Weather/Road Conditions, and Urban or Rural area, among others (see notebook for full list). We will use these features to predict our target variable: Accident Severity.

Once our features our selected, null values are dropped. A correlation analysis is run to determine which of the selected features is most likely to be useful in a predictive accident severity model. Of the selected features, Weather_Conditions, Light_Conditions, Speed_limit, and Urban_or_Rural_Area all seem to be features that are more highly correlated with Accident_Severity. Additionally, these are features a user could easily assess before embarking on a trip. These features (except for Speed Limit), and the target variable Accident_Severity are recoded with their full string values so that exploratory data analysis/visualization can be more user-friendly and readable. Values that are difficult to interpret and would therefore complicate the model (i.e. Weather_Conditions = 'Other') are dropped.

For Accident_Severity, "Fatal" accidents are coded as "Fatal" while "Serious" and "Slight" interactions are coded as "Non-Fatal". This is because our logistic regression model (see details below) had difficulty

distinguishing between a “Serious” and “Slight” accident, rendering the model relatively unhelpful for those two categories. A confusion matrix displaying this situation can be found in the appendix (Fig. 5).

3. Methodology

3.1 Balancing the Data

With our cleaned up data, ready for modeling, we can check to see whether our data is balanced between Accident_Severity=“Fatal” and Accident_Severity=“Non-Fatal”. Of course, in everyday life we are aware that minor accidents are much more common than fatal accidents. It therefore follows that our dataset is very heavily weighted towards non-fatal accidents. We must balance our data so that our model is not biased towards nearly always predicting non-fatal accidents, just from the fact that there are more of them. This would not serve us well since our aim is to avoid fatal accidents, specifically.

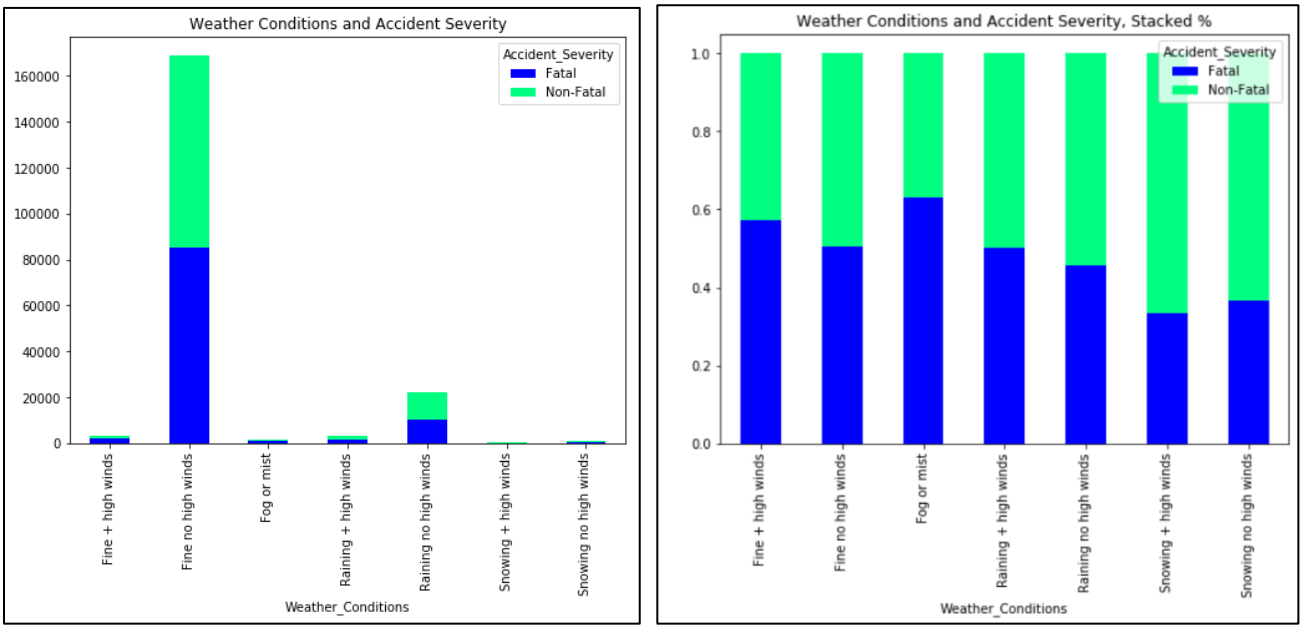
Due to this imbalance, we “upsample” Accident_Severity=“Fatal” through randomized multiplication of fatal accident datapoints. We also “undersample” Accident_Severity=“Non-Fatal” through randomized culling of the datapoints. Ultimately, each case now has 100,000 datapoints to be used to train and test our model.

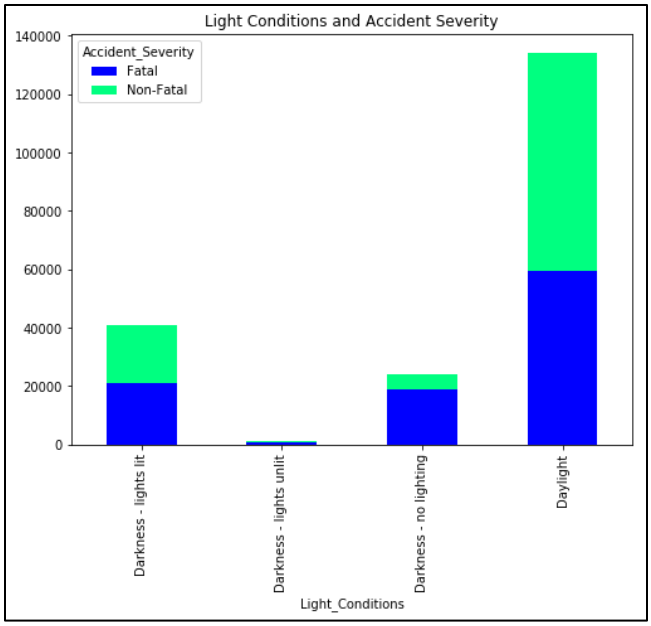
3.2 Data Visualization/Exploration

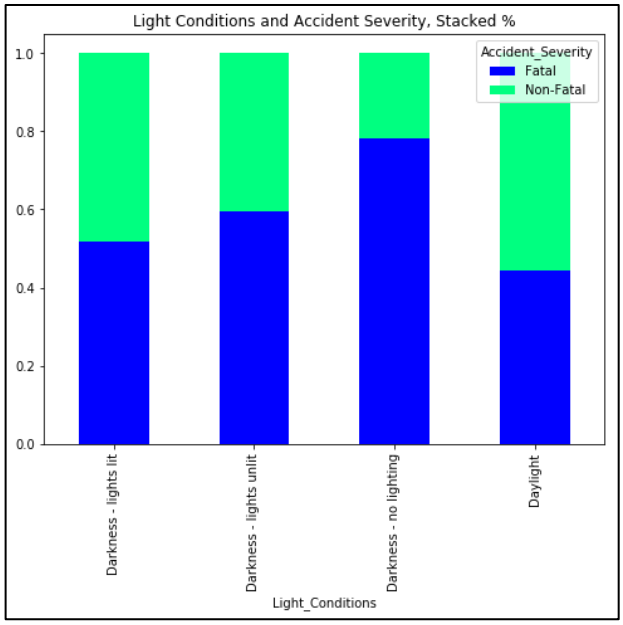
With our balanced data, we can now perform some helpful data visualizations to gain a better understanding of the distribution of our data and relationships between our selected features and Accident_Severity. We visualize our data in two ways per feature:

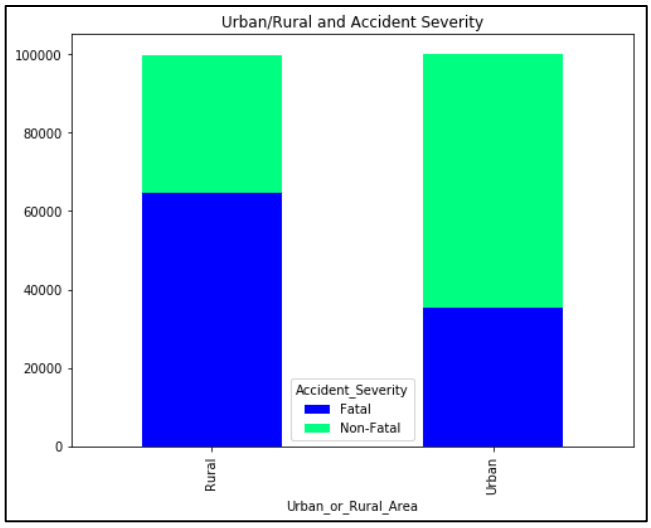
1: Stacked histogram of feature, with Accident_Severity labels stacked (differentiated by color)
2: Stacked percentage of Accident_Severity labels (differentiated by color) per feature value

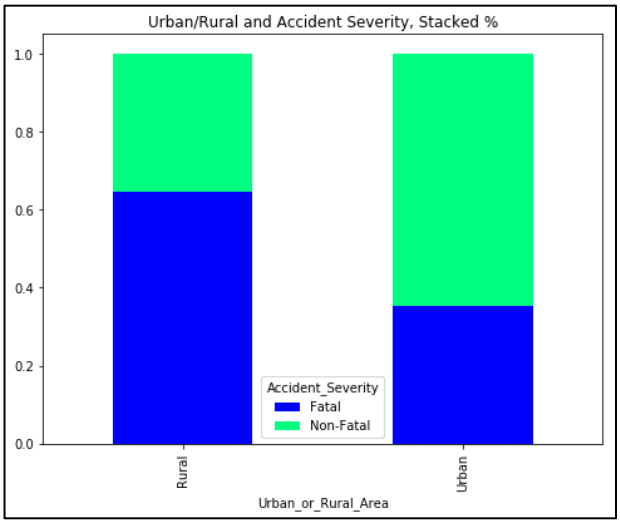
Fig 1. Feature set visualization

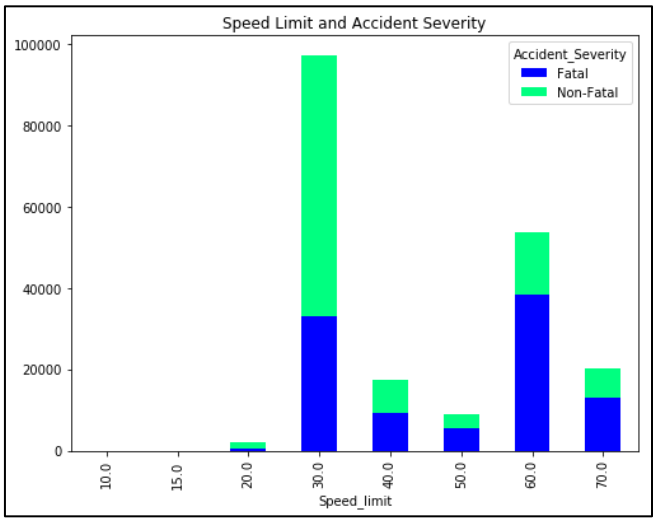


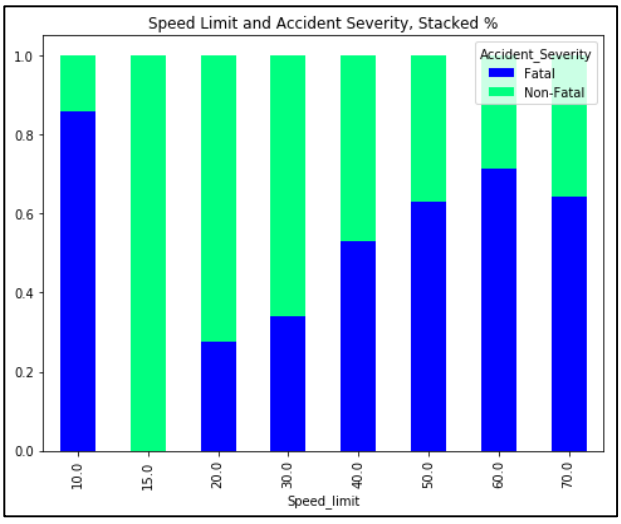

Light Conditions and Accident Severity
Accident_Severity
Fatal
Non-Fatal
140000
120000
100000
80000
60000
40000
20000
0
Darkness - lights lit
Darkness - lights unlit
Darkness - no lighting
Daylight
Light_Conditions


Light Conditions and Accident Severity, Stacked %
Accident_Severity
Fatal
Non-Fatal
1.0
0.8
0.6
0.4
0.2
0.0
Darkness - lights lit
Darkness - lights unlit
Darkness - no lighting
Daylight
Light_Conditions


Urban/Rural and Accident Severity
Accident_Severity
Fatal
Non-Fatal
100000
80000
60000
40000
20000
0
Rural
Urban
Urban_or_Rural_Area


Urban/Rural and Accident Severity, Stacked %
Accident_Severity
Fatal
Non-Fatal
1.0
0.8
0.6
0.4
0.2
0.0
Rural
Urban
Urban_or_Rural_Area


Speed Limit and Accident Severity
Accident_Severity
Fatal
Non-Fatal
100000
80000
60000
40000
20000
0
10.0
15.0
20.0
30.0
40.0
50.0
60.0
70.0
Speed_limit


Speed Limit and Accident Severity, Stacked %
Accident_Severity
Fatal
Non-Fatal
1.0
0.8
0.6
0.4
0.2
0.0
10.0
15.0
20.0
30.0
40.0
50.0
60.0
70.0
Speed_limit

3.3 Feature Selection, Train/Test Split, and Model Set-up

After performing our initial correlation analysis, bolstered by our exploratory visualization of the data, we are now ready to create a model based on the four features stated above. We therefore create a dataframe with just these features. We also now switch the feature values back to numerical values (as opposed to the categorical values we had converted them into, above). We do this via one hot encoding, whereby for each possible value, we have a new column that can either take on the value of 0 or 1 (binary). We do this for Weather_Conditions, Light_Conditions, and Urban_or_Rural_Area. This is not necessary for Speed_limit, which we have kept as a float datatype. We also create our target variable dataframe, just consisting of the column Accident_Severity.

Now that we have our X and y dataframes, we use scikit-learn's train_test_split to split the data into a training set (80% of datapoints) and test set (20% of datapoints). Note that since we are employing a logistic regression, normalization of the data is not necessary. We are now ready to model our data.

3.4 Creating a classification model: Logistic Regression

Due to the size of our dataset (200,000 datapoints with now 15 features following our one hot encoding), and due to the fact that probability of our accident being fatal/non-fatal is helpful to the end user, we decide to employ a Logistic Regression model. This model will allow us to not just predict how severe an accident will be (fatal or non-fatal), given a set of conditions, but also understand the probability of our accident being fatal or non-fatal. Scikit-learn's LogisticRegression function is utilized to create a model trained on our X_train and y_train dataset.

3.5 Evaluating our model

With our testing data (X_test, y_test), we can now evaluate the accuracy and precision of our model. Using scikit-learn's predict function, we can now predict accident fatality based on our X_test data. These values (our predicted "yhats") can be compared against the true labels – our y_test dataset. A Jaccard index, F1 score, and LogLoss are calculated.

Of course, our model isn't perfect, so to help understand where its major shortfalls are arising, we produce a confusion matrix to better visualize our model's mistakes. Finally, we produce a report on our model's precision and recall. Remember, precision represents the percentage of our positive labels that are true positives. Recall represents the percentage of true positives that we were able to predict.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \qquad recall = \frac{true\ positives}{true\ positives\ + false\ negatives}$$

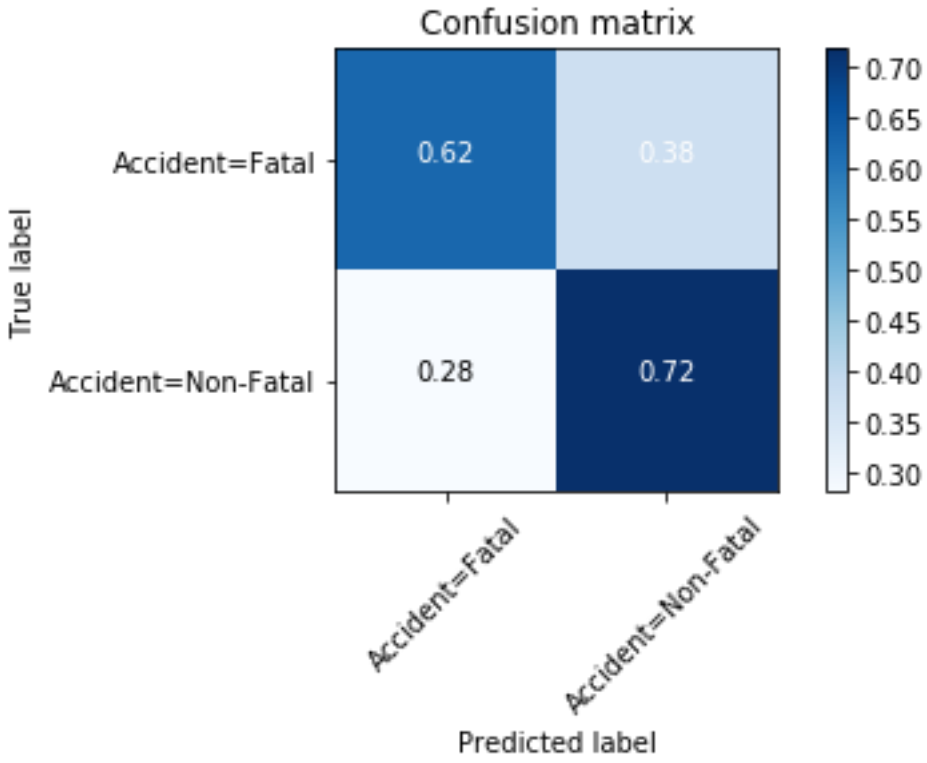
3.6 Creating a user-friendly tool

Armed with our logistic regression model, we now create a function that will allow a user to plug in their expected Weather_Conditions, Light_Conditions, Speed_limit, and Urban_or_Rural_Area values, to predict whether an accident they may get into during their trip will result in a fatality or not.

4. Results

Our model ultimately proved to be relatively accurate in predicting fatal vs. non-fatal accidents. With a Jaccard index of 0.67, F1-score of 0.67, and LogLoss of 0.62, we can be reasonably satisfied. However, there is still much room for error. Our confusion matrix helps us understand where our model is making the most errors (Fig. 2). With the test dataset, our model correctly labels fatal accidents 62% of the time, while our model correctly labels non-fatal accidents at a moderately higher 72% rate.

Fig. 2. Accident_Severity logistic regression model confusion matrix



Our classification report provides additional detail (Fig. 3). With our precision values, we know that 69% of the accidents that our model labels “Fatal” are in fact fatal accidents, while 66% of the accidents that our model labels “Non-Fatal” are in fact non-fatal accidents. Our recall values (same value as in our confusion matrix) indicate that we are only labeling 62% of all fatal accidents “Fatal” and 72% of all non-fatal accidents “Non-Fatal”.

Fig. 3. Accident_Severity logistic regression model classification report

	precision	recall	f1-score	support
Fatal	0.69	0.62	0.66	19951
Non-Fatal	0.66	0.72	0.69	20049

Our user-friendly tool produces results as expected. Fig. 4 displays the predicted label (“Fatal” or “Non-Fatal”) and probability underlying the prediction for two different sets of user inputs. One is a

highspeed, nighttime, snowstorm drive through a rural neighborhood while the other is a slower, daytime, fair weather drive through an urban center. You can likely guess what the predicted accident severity will be!

Our first scenario is predicted to be “Fatal” with a 61% probability. Our second scenario is predicted to be “Non-Fatal” with a 71% probability.

Fig. 4. Predicted labels and probabilities given two user-input scenarios

```
In [122]:  1 print(Predict_Severity(LR_Model,'Snowing + high winds','Darkness - no lighting', 'Rural', 60.0))
           (array(['Fatal'], dtype=object), array([[0.61, 0.39]]))

In [123]:  1 print(Predict_Severity(LR_Model,'Fine no high winds','Daylight', 'Urban', 30.0))
           (array(['Non-Fatal'], dtype=object), array([[0.29, 0.71]]))
```

5. Discussion

Ultimately, our model did a decent job of predicting whether an accident, given a certain set of conditions, would result in a fatality. There is clearly still an element of unpredictability behind each car accident, meaning no driver can put all her or his trust in one logistic regression. However, this model could serve as an extra tool to help inform the driver’s decision-making process. Additionally, it is worth pointing out that this model does not venture to predict whether a driver will get into an accident generally, but just whether an accident that the drive gets into would be fatal.

One interesting point about the model is that very few, and very simple, variables were needed to create a moderately useful model. Light_Conditions, Urban_or_Rural_Area, and Speed_limit were the most impactful variables to include in the model. Weather_Conditions, while slightly helpful in improving the performance of our model, was ultimately not critical. This was surprising, as “checking the weather” is probably one of the most natural and common ways that drivers decide whether to get behind the wheel.

A second interesting point is how the model had a very difficult time labeling accidents as “Serious” (vs. “Slight” and “Fatal”) in our initial version of the model. It turns out that the factors that can help predict a “Serious” accident overlap a bit more cleanly with those of “Slight” accidents, meaning “Fatal” accidents appear to be in more of a league of their own.

6. Conclusion

Before embarking upon a road trip, a driver must weigh a number of factors when ascertaining the safety of such a trip. Accidents are inherently difficult to predict, and so a driver must understand that this is a risk she or he is inevitably taking on. However, given a large, labeled dataset of accident details, we can now better predict whether one of these “inevitable” accidents would result in a fatality. Given a driver’s innate common sense, personal understanding of the route chosen, and a logistic regression model predicting that an accident would be “Non-Fatal”, the drivers of Great Britain can now face the unexpected with a bit more confidence.

7. Appendix

Fig. 5. Confusion matrix using original Accident_Severity labels