Université libre de Bruxelles

Project: Car accident analysis in the UK

INFO-H423: DATA MINING

Report

Authors:
Ait Oujkal Abdellatif 000430127
Outmane Mouad 000427221
Pappas Akilleas 000425456

Supervisors:
Mahmoud Sakr

December 18, 2019

Introduction

During this project, we were asked to analyse some traffic data from 2012-2014 amassed from the UK government. The dataset records nearly half a million accidents and contains 35 attributes. Our task is to help the government by identifying hotspots for car accidents, understand the causes of fatal accidents and try to prevent them. The software we used is *RapidMiner* and we also used *Python/R* language.

Hotspots identification

The first idea that comes in mind in order to find hotspots for car accidents is obviously to plot accidents on a map using longitude and latitude information and see where big number of accidents happen. However, plotting half a million points do not help us to identify zones with a high concentration of accidents because it makes the map unreadable. Therefore, we used a heatmap in order to pinpoint these hotspots.

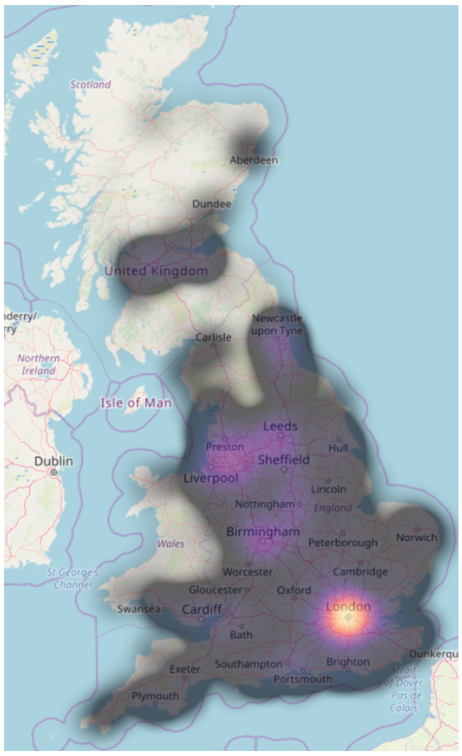


Figure 1: Heatmap

Figure 1 represents the obtained heatmap. As we can see, most accidents happened locally within cities instead on highways. The reason could be that the traffic is more congested locally than on highways. We can see what area has the most accidents, there are 4 hotspots: Liverpool-Manchester, Birmingham, Leeds and London. This is obviously explained by the high density of population in these zones. At the national level, the south of the United Kingdom, where the population density is higher compared to the other region, concentrate 90% of the accidents. We can see that London is the city that contains the most accidents so we decided to study the British capital case. We zoomed into London in order to identify the hotspots at the city level.

Figure 2 shows that there are two hotspots: one in the surroundings of the *City Of London* (right) and a bigger and more concentrated one circled in red in the figure. We can obviously see that accidents are concentrated in the more urbanized areas, indeed, if we look at the data there is a huge difference in the numbers between the 2 zones. We zoomed in the red rectangle to clearly see where are the hotspots at a street level.

Figure 3 shows that the majority of trouble spots in *London* are in big crossings and important roundabouts, since the traffic flows are more important there, the more accidents there are. In addition, we can observe that *Piccadilly Circus* (circled in red in the figure) which is a road junction of London's West End has really high concentration of accidents which is normal since it is a busy meeting place and a tourist attraction in its own right. It is known as a major traffic junction.

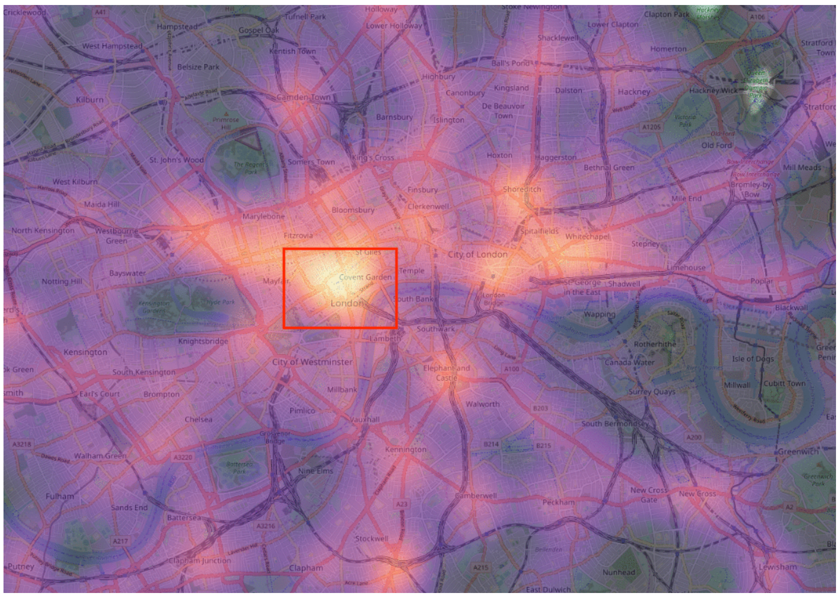


Figure 2: London

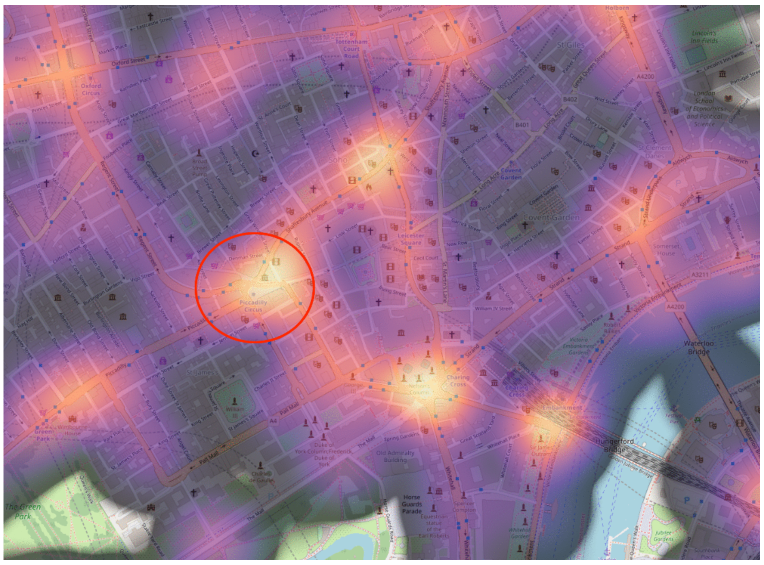
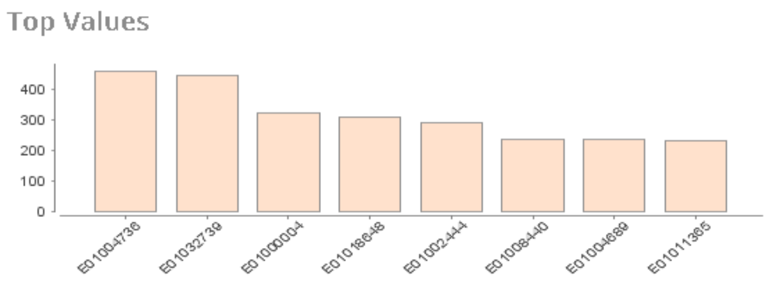


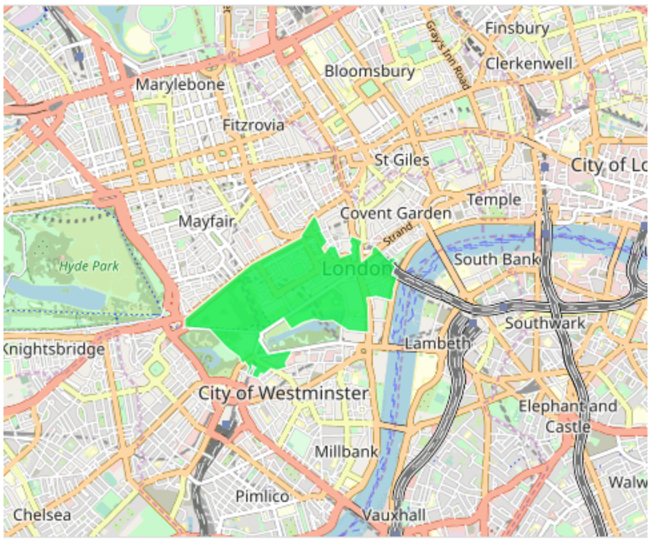
Figure 3: Zoom into London

We also analysed the given dataset in order to determine these hotspots and verify our results. We analysed the LSOA_of_Accident_Location attribute to see where the most accidents happens.

We can observe on the histogram of figure 4 (a) that *E01004736* is the zone containing the most accidents. We looked over the internet to see the correspondence with the zone, it gives us the representation in figure 4 (b) which is exactly the hotspot we have found.



(a) Histogram of LSOA attribute



(b) LSOA zone

Figure 4: LOSA attribute analysis

Data pre-processing

Before trying to understand the causes of fatal accidents and try to detect some patterns in the traffic conditions, we need to prepare the data. Basically, this step consists in removing irrelevant features, cast data to unify the structure and clean the data.

First of all, we checked whether there were any correlated features that we could remove. Figure 5 shows the correlations between all features and we can see that there are 6 high correlated features : *Latitude* with *Location_Northing_OSGR*, *Longitude* with *Location_Easting_OSGR* and *Local_Authorithy_(District)* with *Police_Force*. Since these features are irrelevant to detect patterns in the traffic conditions we removed them.

Once all correlated features removed, we removed all the irrelevant features which are represented in table 1. The attributes *Junction_Detail* and *Junction_Control* contained respectively 100% of missing values and 40% of missing values, this is why we decided to remove them since they do not give any relevant information. Also, the attributes *Special_Conditions_at_Site* and *Carriageway_Hazards* contained respectively 97.78% and 98.31% *None* values so it is self-evident that they will be removed.

Then, we analysed each attribute to look for any missing values. We did not impute any mean or median value since it can lead to data bias. We decided to delete the entire record even if we will use a mining algorithm that tolerates missing values because our data set is big enough to perform analysis.

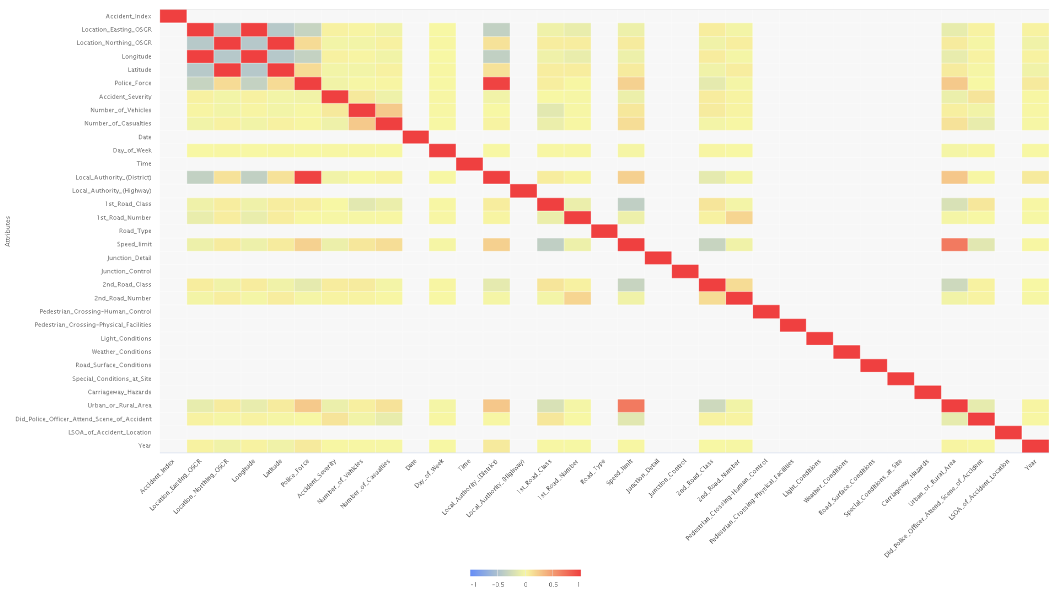


Figure 5: Correlation Matrix

Local_Authority_(Highway)	**Pedestrian_Crossing-Human_Control**	**Did_Police_Officer_Attend_Scene_of_Accident**
Date	**2nd_Road_Number**	**Junction_Detail**
Accident_Index	**2nd_Road_Class**	**1st_Road_Number**
1st_Road_Class	**Special_Conditions_at_Site**	**Carriageway_Hazards**
Junction_Control	**LSOA_of_Accident_Location**	**Year**

Table 1: Irrelevant Features

Finally, we casted data to unify the structure. More precisely, the transformations we have done are represented in table 2.

Attribute name	**Values**	**New values**
Accident_Severity	**1**,**2** or **3**	**fatal** (1) or **non-fatal** (2 and 3)
Day_of_Week	from **1** to **7**	**true** if it is a day of week else **false**
Time	**all times**	**Night**, **Morning**, **Afternoon** and **Evening**

Table 2: Data cast

Detection of patterns in the traffic condition

Now that the data is clean we will try to understand the causes of fatal accidents and detect patterns in the traffic condition that often lead to fatal accidents. The features we selected are traffic parameters, i.e. the relevant informations for the authorities. For example, the light conditions or the surface conditions of the road.

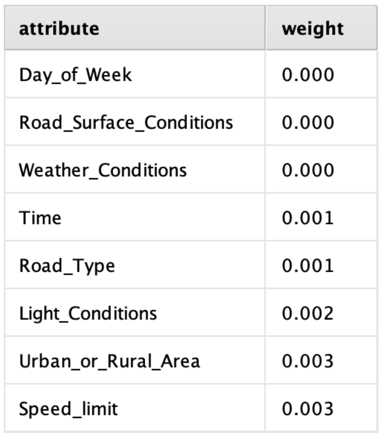
We ran the FP-Growth algorithm with the relevant attributes on the two types of accident severity (fatal (1) and non-fatal (2-3)), to see which item-sets (different combinations of attributes values) are more frequent. We can see that two types of patterns stick out of all possibilities in both types of severity of accidents. Indeed, 30% of the accident (fatal or not) happens in normal weather conditions ("**Fine without high winds**"), the surface of the road being **dry** and in **single-carriage roads** during the day ("**Daylight: Street light present**"). The high frequency of this pattern can be explained by the fact that these are probably the standard and the most common conditions under which British drivers drive or just the most represented situations in the dataset.

We can notice a difference on this pattern by adding an element: the **Urban or Rural area**. In fact, if the accident is not fatal and follows the "standard" pattern (32.56% of non fatal accidents), 21.83% of all non-fatal accidents are in an urban area and 10.73% are in a rural zone. On the other hand, if the accident is fatal and it follows the "standard" pattern (31.08% of fatal accidents), 21.55% are in a rural zone and 9.53% are in an urban area.
We can conclude by saying that fatal accidents under standard conditions occur more in **rural areas** than in urban areas.

The third most common pattern in fatal accidents is basically the standard pattern but at **night** in **rural areas** where there is **no street lighting**. It represents 5.64% of all fatal accidents of the data-set.

The speed limitation seems to be an important factor. Indeed, most of the time in fatal accidents the **speed limitation of the road is 60** compared to the non-fatal accidents where most of the accidents have a speed limitation of 30.

This is confirmed when we see the relevance given by the weight information gain (That uses the OneR classifier) in *RapidMiner* in figure 6.



attribute	weight
Day_of_Week	0.000
Road_Surface_Conditions	0.000
Weather_Conditions	0.000
Time	0.001
Road_Type	0.001
Light_Conditions	0.002
Urban_or_Rural_Area	0.003
Speed_limit	0.003

Figure 6: Attributes selection based on weight

Prediction of fatal accident

Imbalanced Data

After analysing the target attribute we noticed that we were dealing with an unbalanced dataset. Indeed, as we can see on figure 7, there are a lot of *fatal* accidents compared to *non-fatal* accidents. In this scenario, if we use this dataset for training our predictive model then the model will perform badly because the model is not trained on a sufficient amount of data representing non-fatal accidents. Indeed, the classifier can have good accuracy on the majority class (99%) but very poor accuracy on the minority class. There are a lot of solutions to solve this problem, however, the most famous techniques are called **Resampling Methods**. There are two kind of resampling methods. On one hand, we have **Oversampling** which consists on duplicating minority class instances to achieve a more balanced class distribution. However this method is used when we do not have a lot of data to work with and can lead to over-fitting. On the other hand, we have **Undersampling** which consists in removing some observations of the majority class. We used this method since our dataset is big enough but we have to be careful because we could remove information that may be valuable and can lead to **underfitting**.

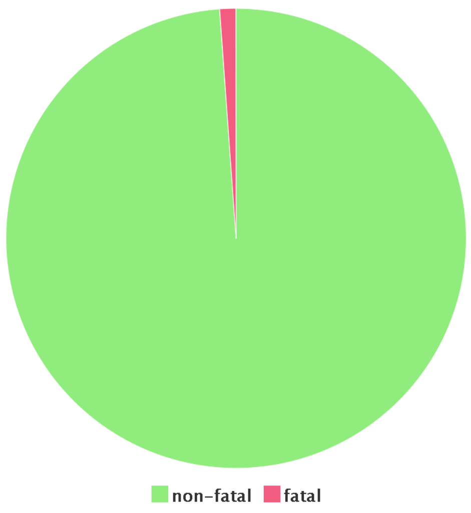


Figure 7: Target distribution

Predictive model

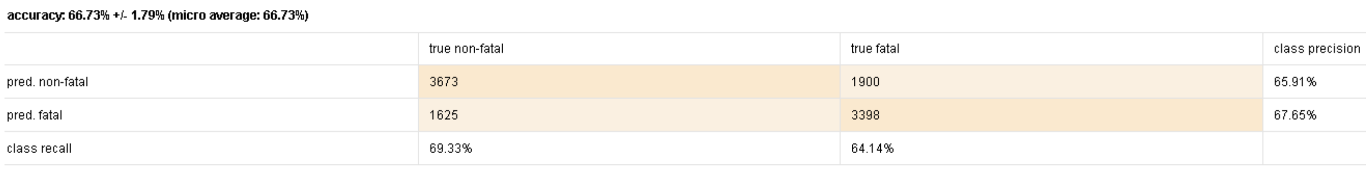
Now that our data is ready, we have to create and train a model that, for a given traffic situation, predicts if it would lead to a fatal accident or not. We chose *Random Forests* as classification algorithm because it performs good compared to the state-of-art models. It uses bagging and consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. Moreover, *Random Forests* can handle huge multidimensional datasets as well as categorical predictors. Concerning the parameters of the algorithm, we used the *RapidMiner*'s built-in *Optimization* operator in order to tune the hyper-parameters and we obtained the following results:

- Number of trees: 60
- Criterion: accuracy
- Maximal depth: 6
- Voting strategy: majority vote

Model Validation

There are many ways to validate the quality of predictions of our model. We chose to use K-fold cross validation which basically consists on dividing the dataset into k folds, of approximately equal size. Each fold is treated as a validation set and the method is fit on the remaining k−1 folds. We set the value of K to 10 since it is very common in the field of applied machine learning.

Also, it has been found through experimentation that it results in a model skill estimate with low bias and modest variance. Figure 8 shows the results of *K-fold cross validation* using *Random Forest*:



accuracy: 66.73% +/- 1.79% (micro average: 66.73%)

	true non-fatal	true fatal	class precision
pred. non-fatal	3673	1900	65.91%
pred. fatal	1625	3398	67.65%
class recall	69.33%	64.14%	

Figure 8: Results of K-fold cross validation using Random Forests

We can see that our predictor has an accuracy of 66.73%. However, recall, precision and F1-score need to be used to measure the performance of the model. Indeed, we could have a model with 97% of accuracy and think that it is extremely good but in reality the model might only predict correctly the majority class and not the minority class. We can get those information by looking at the confusion matrix that summarizes the actual vs. predicted labels where the X axis is the actual label and the Y axis is the predicted label. We can see that there are few false positives (1625) and false negatives (1900) meaning that there were few accidents that were incorrectly classified. Overall the classifier is balanced, the recalls of the classes are similar but with a 3% difference.

The recall score, the ability of the classifier to find the relevant results, is 64.14%. The precision, 67.65%, is also high, this score means that the classifier returned more relevant results than irrelevant ones.

Recall and precision are often reported pairwise because these metrics report the relevance of the model from two perspectives, especially when the data is imbalanced so the accuracy is not useful. But in our case the accuracy results are relevant because we balanced our data before training the model so the classes are represented in the same quantity.

Conclusion

There are two things that were clear from this project. First, the most of the accidents occurred locally and need to be treated at this level. Secondly, we looked for patterns and understood what were the biggest factors in fatal accidents that may be interesting for authorities.