Predict Single-Vehicle Traffic Accident Severity with Machine Learning using Apache Spark

Tian Qi, Anna Zeng, Harrison Mamin, Yihan Wang

Research Question and Data

While traffic accidents have declined globally over the past few decades, they remain the leading cause of death for young adults and injure tens of millions of people every year[1]. From an economic standpoint, the United Kingdom Department of Transport (DofT) estimated the cost of all traffic accidents to be about £36 billion ($46.7 billion) per year[2].

The motivation for this project is to understand the factors that contribute to accidents and use machine learning to predict accident severity level: *slight, severe,* and *fatal*. The insights we gain could potentially have a large social impact in reducing the economic and human costs of accidents through improved public policy.

Our data comes from the UK Department of Transportation in the form of two .csv files[3], which we will join.

- *Accident_Information.csv* (629.9 MB)**:** every line in the file represents a unique traffic accident, featuring various properties related to the accident as columns. Date range: 2005 - 2017
- *Vehicle_Information.csv* (614.6 MB)**:** every line in the file represents the involvement of a unique vehicle in a unique traffic accident, featuring various vehicle and passenger properties as columns. Date range: 2004 - 2016

We chose to examine single-vehicle accidents to better isolate the effects of driver and vehicle attributes on accident severity. In total, we have *578,526* examples of single-car accidents with which to build our model.

[1] https://researchbriefings.parliament.uk/ResearchBriefing/Summary/CBP-7615#fullreport
[2] https://www.hughjames.com/news/comment/2017/11/cost-of-road-traffic-accidents/#.XED0AfxRdWM
[3] https://www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles

Predict Single-Vehicle Traffic Accident Severity with Machine Learning using Apache Spark

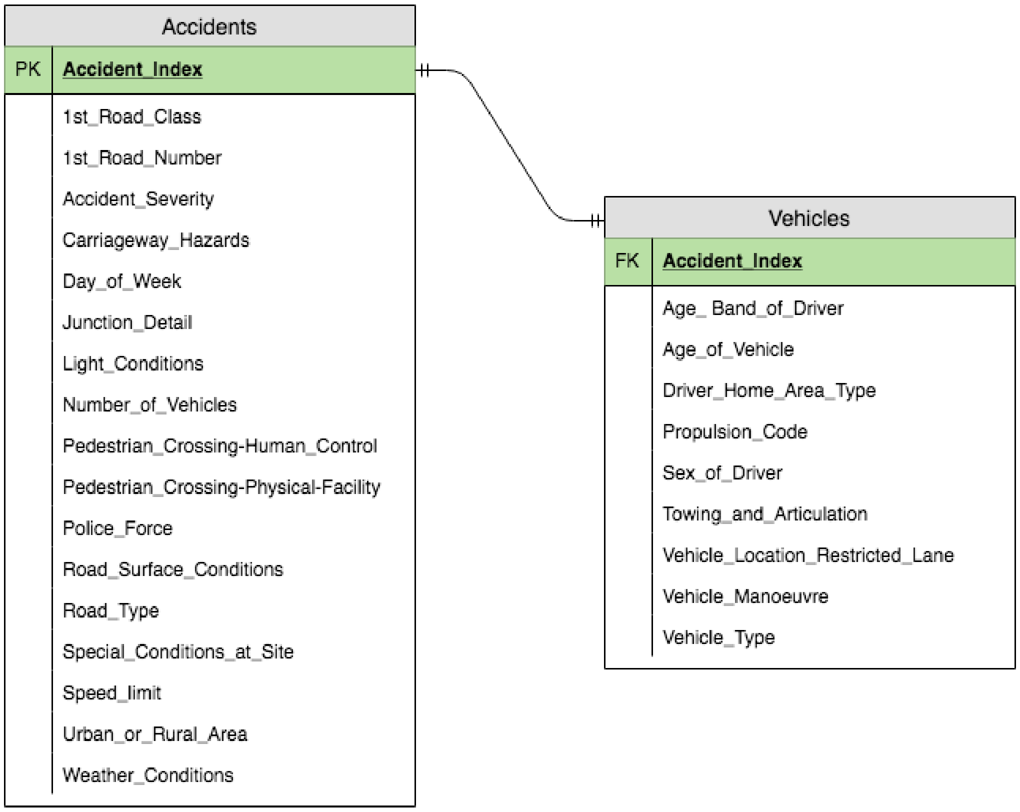

Figure 1: Accidents and Vehicles tables joined using accident index

MongoDB and Spark Configuration

We launched 10 EC2 t2.micro instances, each with 1 vCPU and 1 GiB memory. We use them to create 2 shards with 3 replicas each, along with 3 config servers and 1 mongos server.

We launched an EMR instance with 1 master and 2 slave nodes, each being an m3.xlarge with 8 vCore, 15 GiB memory, and 80 SSD GB storage.

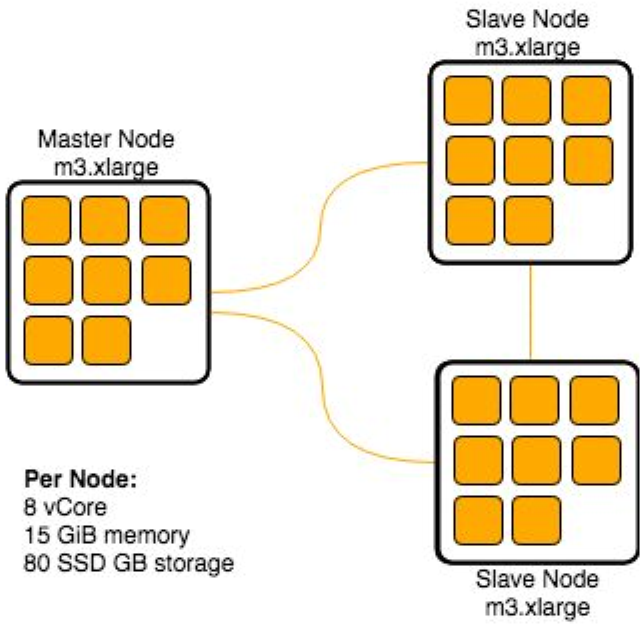

Figure 2: EMR configuration

Machine Learning

The most pressing question is what conditions or attributes tend to increase the chance of high severity accidents. We are therefore more concerned with our model's ability to predict the *fatal* class than its ability to identify *slight* accidents. For the sake of caution, it would be preferable to be sensitive in detecting potentially dangerous situations rather than risk missing them. Therefore, an optimizing metric of recall for *fatal* accidents was selected. Satisficing metrics of macro-F1 score and algorithm runtime were set as well to avoid models with excessively skewed predictions or unfeasible training times.

For a number of reasons, tree-based algorithms seemed like a promising approach. They aid our goal of prioritizing model interpretability and integrate easily into a distributed computing framework. In addition, they should be able to handle our high-dimensional data, and the large number of training examples should be enough to mitigate the risk of decision tree overfitting.

Predict Single-Vehicle Traffic Accident Severity with Machine Learning using Apache Spark

After we fit our initial Random Forest model and evaluated it on the test set, we found that it predicted *slight*, the most frequent class, for all samples. This was due to our highly imbalanced training data consisting mostly of *slight* accidents. A model that predicts a single class in all cases would not be useful in practice. *Slight* was downsampled and *fatal* was upsampled to make our dataset relatively balanced.

Results

Our initial models all suffered from high bias. Some hyperparameter tuning was required. For the decision tree, increasing max depth yielded some improvements. Entropy proved to be a more effective splitting criteria than Gini impurity. Note that entropy can be slightly more computationally expensive due to the log function, but this was not an issue because our final models all ran relatively quickly on our distributed setup.

$$I(X_n) = -\Sigma_i^c p_i log(p_i)$$
$$I(X_n) = \Sigma_i^c p_i * (1 - p_i)$$

Figure 3: Two different measures of impurity calculated for a given node X_n and summed over c classes. Entropy (eqn. 1) was more effective than gini impurity (eqn. 2).

The Naive Bayes model actually had a higher recall for *fatal* accidents, but it failed to satisfy our macro-F1 threshold of 0.5 and was consequently dropped from consideration. A 20 tree Random Forest implementation was also considered, but took far longer to run and did not meet our minimum processing time requirement so it was dropped as well. We kept the 10 tree version instead.

Ultimately, we chose the 10 tree Random Forest as our final model because it performed the best on Recall for *fatal*, our optimizing metric. Processing speed was not a determining factor because all the models ran in a reasonable amount of time.

Predict Single-Vehicle Traffic Accident Severity with Machine Learning using Apache Spark

The following metrics were computed on the test set predictions:

Model	Recall (Fatally Injured)	F1 score (Overall)	Processing time (Wall time)
Random Forest	***0.6084***	*0.5538*	*3 min 29 sec*
Decision Tree	*0.5857*	*0.5829*	*1 min 0 sec*
Naive Bayes	*0.6468*	*0.4396*	*2.17 sec*

Table1: Model evaluation

Random Forests are particularly well-suited to distributed computing, and our 10 tree implementation only took roughly two minutes longer than the single decision tree.

Lessons Learned

Model interpretation:

From our final Random Forest model, we were able to extract the five most important features for predicting accident severity (their relative importance is in parentheses):

1. Speed limit ***(.244)*** : Unsurprisingly, speed limit was a major factor. Higher speeds create the potential for more destructive accidents.
2. Vehicle maneuver ***(.135)*** : This represents what the driver was doing before the accident occured, such as going ahead of another car, turning right, or slowing/stopping. Vehicle maneuver determines the type and degree of collision, so it is not surprising that maneuver is an important contributing factor to accident severity.
3. Junction detail ***(.121)*** : This represents the layout of the road at the accident site, such as a T-shaped intersection, crossroads, or roundabout. It is possible that certain road layouts are more dangerous than others. This would be a good topic for further research.

4. Police force district ***(.114)*** : This is a proxy for the region of the country. A more detailed breakdown of the feature importance by district can provide insight into which areas of the country have more *serious* accidents.
5. Vehicle location - restricted lane ***(.050)*** : This represents whether or not the car was in a restricted lane around the time of the accident (such as sidewalk or bike lane).

Environmental factors such as weather or day of the week did not play as large of a role as we initially suspected. Similarly, driver characteristics were less meaningful than expected. Concrete actions (such as turning, passing a car, moving through an intersection, or entering a zone with a different speed limit) proved to be more important than characteristics regarding identity (e.g. driver sex). A simple effort to educate people about safe driving practices could be a worthwhile use of time. The junction detail statistics could also provide some guidance for improved road layout designs.

Data processing lessons

The initial MongoDB we launched crashed after two days, so we had to re-do the entire setup process again, which was time-consuming. One thing we learned from this experience is to launch multiple copies the first time in case one copy goes down.

When working with large datasets, we found that sampling can be powerful tool for experimentation. Manipulating data and training models can be a highly iterative process, and using the whole dataset at all times slows productivity. It is often useful to run tests on subsets of the data, then return to the full set after debugging.

Along the same lines, it would have been easier if we had performed more of the data cleaning before importing it to MongoDB. This is not always possible, of course, but in our case the dataset size was small enough that this was an option.

Predict Single-Vehicle Traffic Accident Severity with Machine Learning using Apache Spark

Potential Improvements

Due to time constraints, we were not able to devote as much time to feature engineering and hyperparameter tuning as we initially planned. We also would have liked to further explore the Random Forest feature importances and potentially exclude less predictive features from our model in an effort to reduce noise.

If we had more time, we also would have done additional exploratory data analysis and visualization before building our machine learning models. These tasks would have aided our feature selection and model tuning.

More broadly, we found it to be extremely important to define a clear, specific question early in the modeling process. Honing in on a goal was a challenge, and we considered several different variations and approaches.