Road Accident All-Casualty Severity Prediction

[redacted] Machine Learning Project Report

Students:

Doron Battino [redacted]

Pavel Finkelbaum [redacted]

Abstract

Every year, the lives of approximately 1.3 million people are cut short because of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, with many incurring a disability because of their injury.
Motivated by the above, traffic accident prediction is as crucial as it is a challenging issue in the domain of intelligent traffic safety management system; it is of great significance for analysing the future development trend of traffic accidents and implementing proactive prevention measures under existing road traffic conditions. To improve traffic safety management and control, it is necessary to seek timely and accurate methods for predicting traffic accident **severity**.
In our project, we aim to predict the number of casualties of a given accident, based on given features regarding the accident's circumstances. In addition, we would consider the models developed to be used in a specific context:
The model might be implemented by emergency services to be used when they receive an emergency call to report an accident. Thanks to the information given by the caller, which is usually the place of the accident along with some additional details, the emergency services would be able to predict the number of casualties caused by the accident. By possessing this kind of information before sending emergency staff and equipment to the place of the accident, we can plan the need of resources for this accident more precisely. Moreover, it would be possible to anticipate by contacting in advance the nearest hospitals to warn them about a serious case coming. By matching the resources sent with the actual need, it also helps letting more resources available for other emergencies that can happen at the same time.
In order to successfully accomplish the task, we used unstructured and structured machine learning models and algorithms. We have elaborated our different models according to prediction accuracy and by the processing time of each model, led by the understanding that the shorter processing time of the model and the more accurate it was, the better chance we would have at saving lives at the actual occurrence point of the accident.

Problem Definition

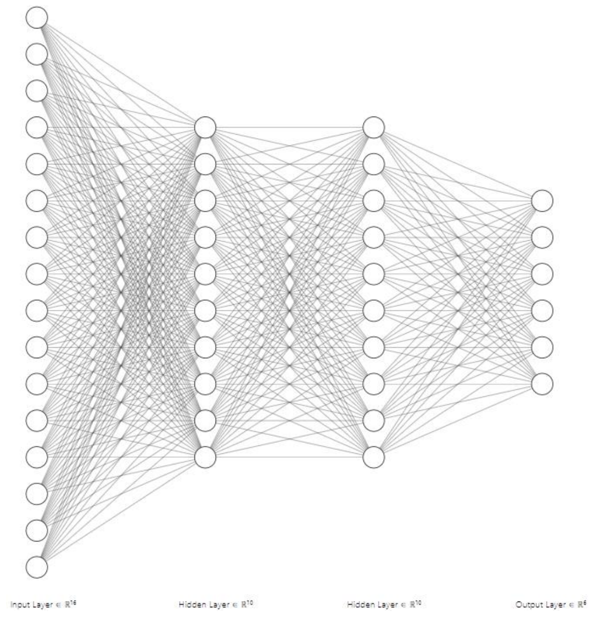
To get a clearer view of the problem we hoping to solve, we first start by examining our data source and the pre-processing part preceding the Model and Algorithms part. We based our project on the Kaggle.com website's **Accidents in France from 2005 to 2016** dataset, containing 1+M records, comprised namely of 5 tables with a total spread of 57 columns:
Characteristics: 16 features in total. For example: Day of the accident, Lightning conditions, Type of collision (frontal, from the rear), Type of intersection etc.
Holidays: 2 features in total :Date and the holiday happening on that date.
Places: 18 features in total. For example: Category of road, Traffic regime, surface condition etc.
Users: 12 features in total. For example: User category, sex, date of birth, reason for traveling in time of accident etc.
Vehicles: 9 features in total. For example: Category of vehicle, Identification of vehicle etc.

To be consistent with our desire for the implementation of the model by the emergency services, we decided to choose only visible features likely to be known by a caller to the emergency services. Out of total 57 features, we ended choosing 16 features:
Catr- Category of road (int 1-9)
Circ- Traffic regime(int 1-4)
Nbv- Total number of traffic lanes (int)
Vosp- Indicates the existence of a reserved lane (int 1-3)
Prof- Longitudinal profile describes the gradient of the road at the accident site (int 1-4)
Plan- Drawing in plan (int 1-4)
Surf- surface condition (int 1-4)
Jour- Day of the accident (datetime – later reduced to int 1-31)
Mois- Month of the accident (date time- later reduced to int 1-12)
Hrmn- Time of the accident in hour and minutes (datetime-later reduced to int)
Lum- Lighting: lighting conditions in which the accident occurred (int 1-5)
Agg- Agglomeration(int 1-5)
Int- Type of Intersection (int 1-9)
Atm- Atmospheric conditions (int 1-9)
Col- Type of collision (int 1-7)
Dep- Departmeent : INSEE Code
The features' values were largely manipulated by us, as a large portion of data was missing\ filled with incoherent values. for each feature, we either evaluated the average value of the valid records or decided of a constant value which would fit the blank spots best, based on the meaning of the values themselves, regarding the selected feature. other features, such as hrmn and dep were rounded to fit larger categories, with the aim of providing a better overall prediction accuracy. a column named num_of_inj, which is the label we wish to predict, was created by careful extraction from the records, and added to our current dataset.
Due to large variety of possible casualty number in each accident, that is according to the dataset, we decided to represent the number of casualties within 6 groups:

- [0], [1], [2], [3], [4], and [5+]
- Each category corresponding to the number of casualties within the given accident.

Models and Algorithms

In order to examine the feasibility of reaching our main learning objective, we decided to evaluate and compare between multiple learning algorithms, models, and approaches. For our unstructured approach we implemented the KNN, Logistic Regression, SVM and Random Forest learning algorithms and compared their performances.
For our structured approach, we designed a Feedforward neural network with a Structured Perceptron algorithm, using hyperbolic tangent function as activation function and MSE (mean squared error) as the Loss function. The Weight Initiation matrix was sampled from uniform distribution with parameters [0,1]. We constructed our graphical model according to the following method:



The network has 16 input nodes, as the number of features used for the prediction, 2 hidden layers and 6 final output nodes, corresponding to the number of labels.
For the advanced part, we decided to implement some deeper variations of our graphical model, with the purpose of improving our accuracy and execution time.
We understood that our model's accuracy rate and execution time might be significantly affected by a combination of factors, ranging from the activation function used and to the architecture of the network itself. We decided to try and figure out connections between these hyperparameters of a FFN and the prediction accuracy along with execution time.
For the creative part we chose a different combination of effecting factors, mostly regarding the Random forest algorithm, which we tried to tune and therefore, to predict with better accuracy.

Experiments and Results

Throughout the evaluation and analysis process for our models and algorithms, we used 2 main metrics:

1. Accuracy – percentage of correct labeling (higher is better)
2. Execution time– how fast did the algorithm perform (in seconds).

In order to evaluate our learning model for the basic part, we initially ran all our different algorithms on a 100,000-sample dataset with a test size of 0.2 and with 6 possible prediction labels refer to the number of casualties of the accident sample, and got the following results:

Comparison of all algorithms for the basic part

Classifier	Accuracy	Execution time
KNN	0.149	0.0289 sec
Logistic Regression	0.570	2.086 sec
SVM	0.571	226.253 sec
Random Forest	0.553	5.927 sec
FFN	0.568	20.56 sec

The results from the comparison surprised us, mainly because of the out-performance of Logistic Regression and SVM over the Random Forest and the FFN, indicating that even for the simpler learning objective, the data may be linearly separable since 2 of the unstructured algorithms gave higher accuracy rate than the random forest and the structured FFN. Despite of SVM's and Logistic Regression accuracy rates, we have agreed among ourselves that the execution time for SVM is too long, especially when the gap between the accuracy rate is relatively small. As for the Logistic Regression algorithm, we decided that there aren't enough significant hyperparameters to be adjusted to improve its prediction accuracy, apart from those which are data driven. Considering those decisions, we have concluded that to continue our analysis with improved results, we would attempt to improve our classifiers and models, mainly the hyperparameters of the Feedforward network, while also analyzing our feature set, the effects of different parameters per other classifiers and model variations.

<u>Unstructured model analysis:</u>
While analyzing our unstructured classifiers described in the Models and algorithms section, we chose to focus on the following parameters:
Dataset size
Experimenting with different dataset sizes showed that the best results were received for a 100,000-sample dataset. Decreasing the dataset size to 50,000 samples decreases results accuracy rate across all algorithms by at least 6%, not considering the KNN algorithm whose accuracy rate and execution time is only slightly affected by the dataset size. Increasing the size to 150,000 produce improvement only with SVM, but increases its execution time to 780.645 sec, making it still not effective enough timewise. A possible explanation might be that a too small dataset simply doesn't consist of enough records, as we have 16 features, creating at minimum 4^{16} possible vectors, and only a fraction of samples to learn from. The decrease in accuracy with the 150,000-sample dataset might be explained with a highly different type of data distribution inside the samples, presenting itself only within the last 50,000 samples, obstructing the models to make accurate prediction on highly different portion of samples.
Important Features
Our analysis showed that the 5 most important features for Random Forest, using the 100,000-sample dataset, were DEP, JOUR, HRMN, MOIS, and CATR. As we examined these features, we got the following results:

<u>Classifier</u>	<u>Accuracy</u>	<u>Execution time</u>
Random Forest	0.573	6.419 sec

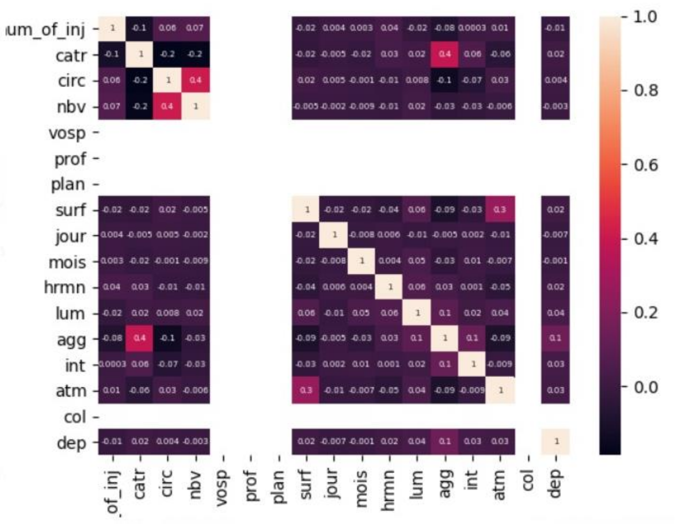
<u>Structured model analysis:</u>
While analyzing our structured model described in the Models and Algorithms section (FFN), we chose to focus on the following parameters:
Dataset size
Our analyses showed that changing the dataset size did not have a significant direct effect on our classifiers performance and that the best results were received for our 100,000-sample dataset with a test size of 0.2 while 150,000-sample dataset was very close behind it.

Important features

Since our structured model contain joint features between data features and labels, we could derive which features were highly correlated to each other. we comprised a correlation matrix and exhibited the 5 most significant features by distance from 0 (the matrix is presented below): CATR, NBV, AGG, CIRC, HRMN. After modifying the number of the input nodes for the FFN, We got the following results:

Classifier	Accuracy	Execution time
FFN	0.579	14.68 sec

The feature importance improved the accuracy rate by 1.5% and reduced the time taken by 5 sec. Here we would also like to point out an interesting experiment: we decided to run the FFN with the 5 most significant features derived for Random Forest, and the results for both algorithms were almost identical. Our assumption to explain those results concluded that HRMN and CATR were the 2 significant features for both algorithms respectively, and the other 3 features were nearly dispensable.
Following the experiment, we decided to drop the next step, where we planned to only remove the 5 worst features.



Activation function (FFN)

Since the activation function defines how the weighted sum of the input is transformed into an output in every layer of the network, we decided to apply different functions to both the hidden layers, and the output layers. We got the following results:

Activation function	Accuracy	Execution time
Sigmoid	0.5755	13.57sec
Tanh	0.579	14.91 sec
ReLu	0.5698	16.69 sec

Number of iterations (FFN)

We already understood that the number of iterations on a neural network deeply affects the backpropagation and thus the model's accuracy, but we found the above 100 iterations there was almost no effect on our classifier's performance.

Initiation of weight matrix (FFN)

We tried to differently initiate the weight matrix, with the belief and understanding that with a limited number of iterations, the initial weight might deeply affect the prediction accuracy, even if affected by back-propagation over the course of the iterations. We got the following results:

Initiation distribution	Accuracy	Execution time
normal	0.5741	15.59sec
Uniform[0,1]	0.5702	15.02 sec
Constant (all w=1)	0.5754	14.75 sec

Network Architecture
We assumed that modifying the structure of the network might also affect the results, given that more nodes in the hidden layers will provide a more efficient back-propagation and ultimately, better prediction accuracy. after several modification attempts, we found that 12:8:8:6 gave the best result on accuracy and time combined.

Creative part:
In this section, we decided to tune the hyperparameters of the Random Forest, as our attempts with the FFN didn't produce significant improvements. We modified the following parameters:

Number of trees in the forest
After trying out different values, we came to best result with a relatively low number of trees: 7. We will elaborate at the end of the creative section.

Max number of features considered for splitting a node
As we have seen in previous sections, when chosen the most significant features, the random forest algorithm improves its accuracy, while taking slightly longer to achieve the result. We reached best result with maximum of 6 features.

Max number of levels in each decision tree
The most significant hyperparameter from our experience. Increasing its number from 5 to 15 increased the accuracy rate by 4%, even while running the same algorithm with 5 levels but with 100 number of trees, still resulted with lower improvement percentage.

Discussion and Conclusions

The main challenge we faced throughout the project was understanding the relationships between the different features and models and constructing a model that accurately exhibits a structure which logically fits the data, and thus, providing reliable prediction for our dataset, and for all accidents of the same format.

Our main conclusions from this project are, that while it is possible to learn this task, its complexity within the fine details of the algorithms, parameters, and the data manipulation techniques proved to be much harder for this level of machine learning methods, than we would've expected. We believe that to provide a better prediction accuracy with a bounded, or at least acceptable execution time, a more complex approach is required, that incorporates in the learning model a deeper understanding of factors and the cross-validating between them, as well as better analyses of features and structures and the hidden structural patterns between them. We strongly believe this learning task should be further explored, using different structured learning algorithms which we didn't explore, as well as a full, network-understanding of hyperparameters tuning of the different algorithms, while considering the representation of the dataset, its features and some-artificially inserted values.

While we are somewhat disappointed with our overall prediction accuracy across different models, we feel that we came to some interesting conclusions, explored different ideas, and came upon some interesting results along the way. We enjoyed every hour of preparation towards our project and would be eager to continue exploring such challenges in the future.