USA Traffic Accident Forecasting

IBM Advanced Data Scientist Certification Capstone Project

Pedro Roig Aparicio | Master student @ETH Zürich
Zurich | 12.10.2020

Contents

Introduction.
The dataset

The US-Accidents dataset is a very complete data source

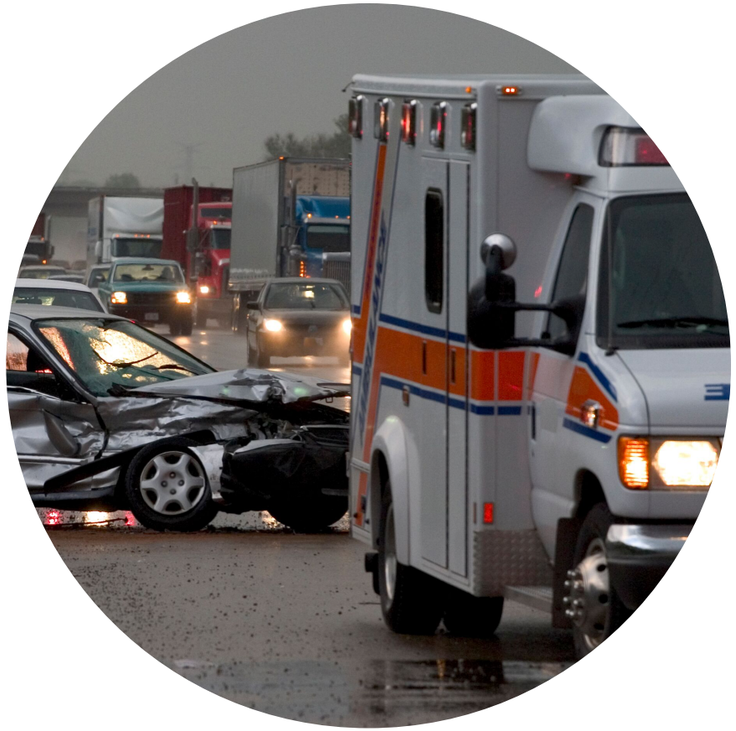
About the *'US-Accidents: A Countrywide Traffic Accident Dataset'* (*US-Accidents dataset* in short):

- More than **3.5 million** accident records.
- Merges traffic data from **several entities**, such as the US and state departments of transportation, law enforcement agencies, traffic cameras and sensors.
- Covers **49 states** of the United States.
- Has continuously been collected since **February 2016**.
- Contains a total of **49 features**.

Each accident is classified in one of 4 severity levels, depending on the time needed to clear out the traffic and the length of the traffic jam. I'll predict the severity of the accidents via supervised learning.

Introduction.
The use case

Traffic continuously impacts todays society in many ways



Death cause: 36,560 people lost their life in the US during 2018. That is more than 100 persons per day [1].

Traffic jams: 5 out of the top 10 most-gridlocked cities in the world in 2018 were in the US [2].

Environmental threat: Transportation emissions are a major source of air pollution in urban areas [3].

[1] National Highway Traffic Safety Administration (NHTSA)

[2] INRIX

[3] United States Environmental Protection Agency

Benefits of traffic data analysis and accident prediction

Some of the many advantages of traffic accident forecasting are:

- Understanding the drivers of car crashes enables preventing future accidents.
- Accurate prediction can be used for optimizing resource planning during peak times in order to decrease the death and serious injury number.
- Being able to forecast future congestions and accident hot spots can be used for a better city growth and spatial planning.

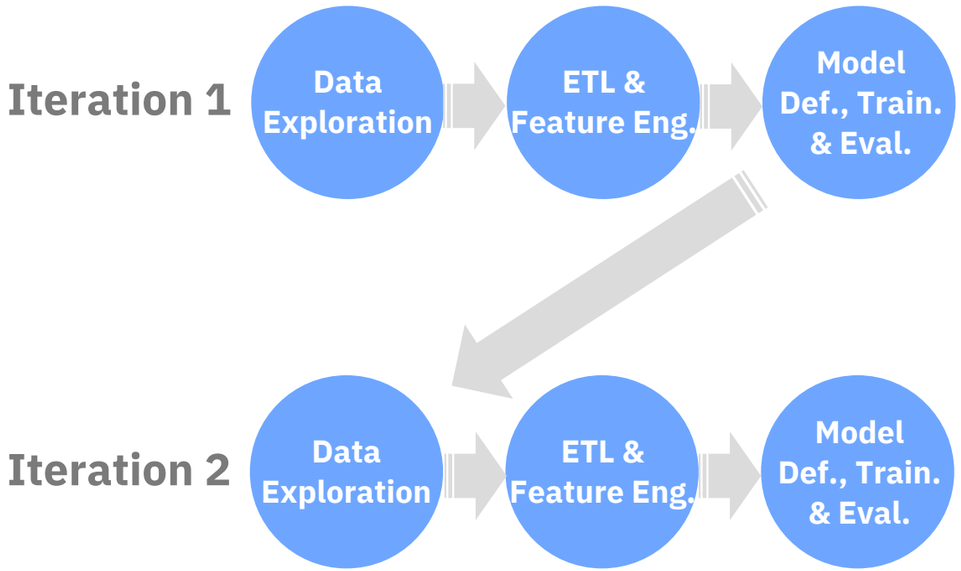
‘By 2020, halve the number of global deaths and injuries from road traffic accidents.’, UN 2015

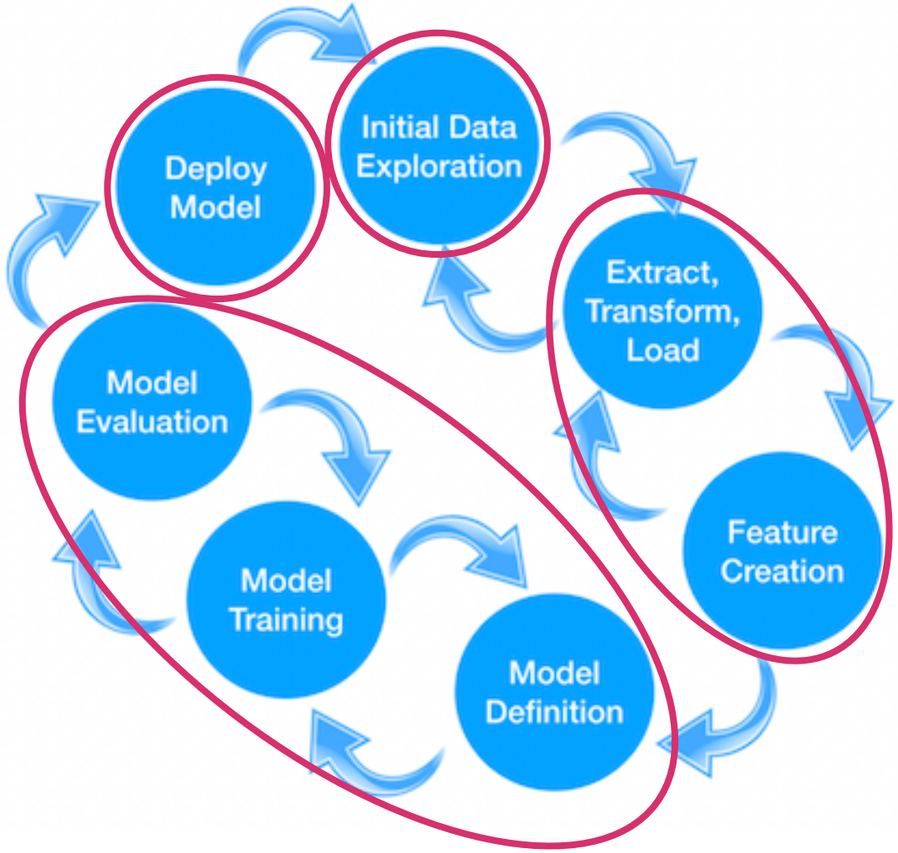
[1] Transforming Our World: The 2030 Agenda for Sustainable Development, United Nations (2015): https://sustainabledevelopment.un.org/post2015/transformingourworld

Solution.
Approach structure

Following the *Lightweight IBM Cloud Garage Method for DS*, the approach is structured in an iterative way

Below a brief description of my specific project was structured. Each of the tasks (the bubbles below) has its own notebook:





Lightweight IBM Cloud Garage Method for Data Science Process Model [1]

[1] https://developer.ibm.com/articles/the-lightweight-ibm-cloud-garage-method-for-data-science/

The project is contained in a single GitHub repository

File description:

- README.md: Contains the project description.
- *_[task]X.0.ipynb: The notebooks follow the naming convention of the IBM Garage method. The [task] placeholder refers to the task or phase that the notebook is focused on. The X.0 placeholder shows the iteration number.
- Lightweight_IBM Cloud_Garage_Method_for_Data_Science_ADD_Template.docx: Project description and Architectural Decisions Document (ADD).

N.B.: The *_data_exp*- and *_etl_and_feature_eng*-modules output a csv-file each. These files are ‘gitignored’ in order to save storage. One can easily get those files by running the respective scripts.

The tasks from the previous slide follow the IBM Garage Method for Data Science. Here the respective notebooks:

- _data_exp: import of the original dataset, feature understanding, visualization and data limitations.
- _etl_and_feature_engineering: typecasting, value imputation, feature transformation, feature encoding.
- _model_def_train_evaluation: data augmentation to deal with class imbalance, comparison of performance of different machine learning models.

Solution.
Iteration 1

Iteration 1 description

Data Exploration

Feature understanding:

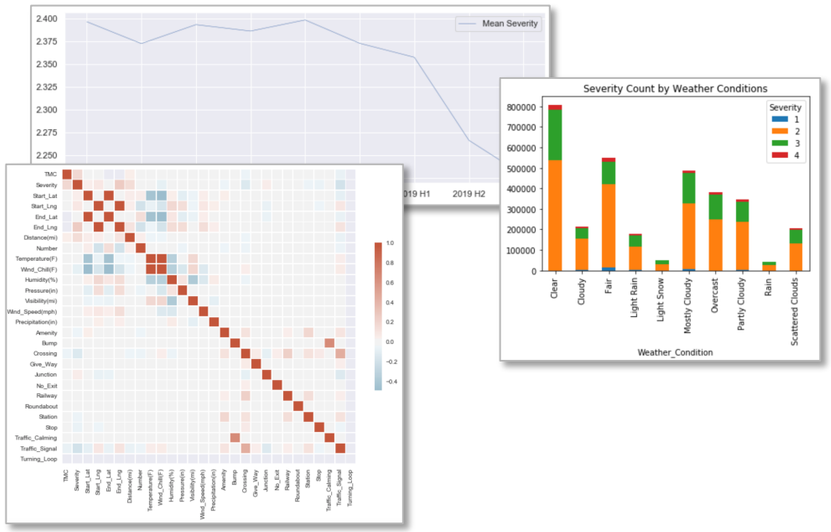
- Variable type
- Unique values
- Univariate statistics

```
***** Source *****
Type: <class 'str'>
Number of unique values: 3
Unique values: ['MapQuest' 'MapQuest-Bing' 'Bing']
```

```
***** Precipitation(in) *****
Type: <class 'numpy.float64'>
Max val: 25.0
Min val: 0.0
Average value: 0.01598255881559659
Number of unique values: 262
```

Visualization:

- Time progression
- Histograms
- Pie diagrams
- Correlation matrix



ETL & Feature Engineering

Typecasting & redundancy:

- *Wind_Direction*, *Weather_Condtions*

Feature relevance and value imputation

Feature transformation:

- *Minute*, *Street* to frequency domain

Feature encoding

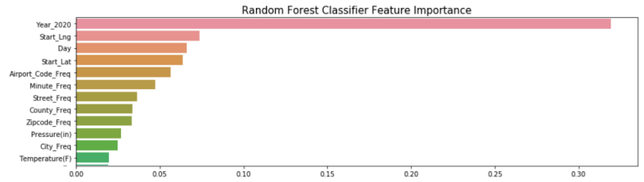
Model Definition, Train & Eval

Data augm (re-/sampling) for class imbalance (50’000 dp)

Model performances:

	Model	Train_Accuracy[%]	Test_Accuracy[%]*
0	Linear SVM	66.1	65.7
1	Decision Tree	80.4	70.3
2	Random Forest	95.9	71.8
3	Neural Network	80.21	69.6

Feature importance:



* The test accuracy refers to the accuracy achieved with the validation set

Iteration 1 key takeaways: A big overfit and not sensible feature importance points towards wrong assumptions in the model

Overfit:

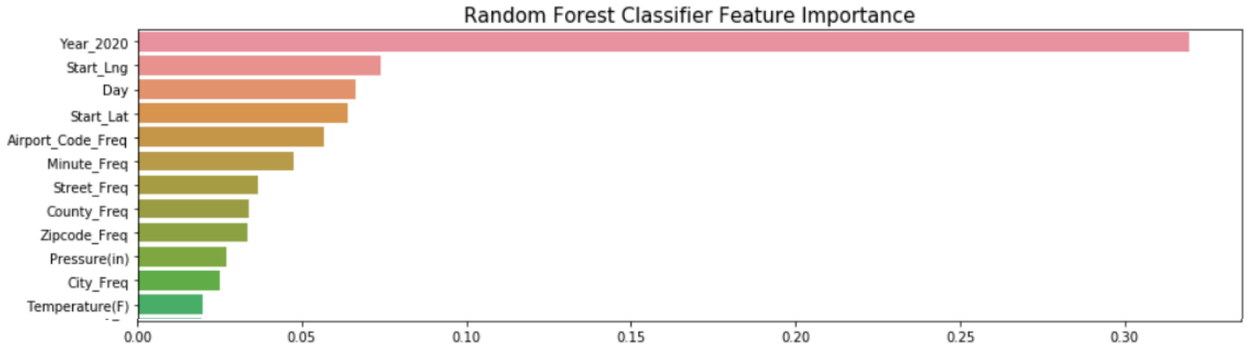
Performance of models during training is much higher than in the validation set, which points towards an overfit of the model to the training data.

	Model	Train_Accuracy[%]	Test_Accuracy[%] *
1	Decision Tree	80.4	70.3
2	Random Forest	95.9	71.8
3	Neural Network	80.21	69.6

Big overfit

Feature importance:

The most important feature is the one-hot encoded year 'Year_2020' with around 0.30% importance for both models (here on the right I only show the Random Forest). This means the model is giving great importance to such a specific feature and in my opinion could mean some of the assumptions made about the model are wrong.



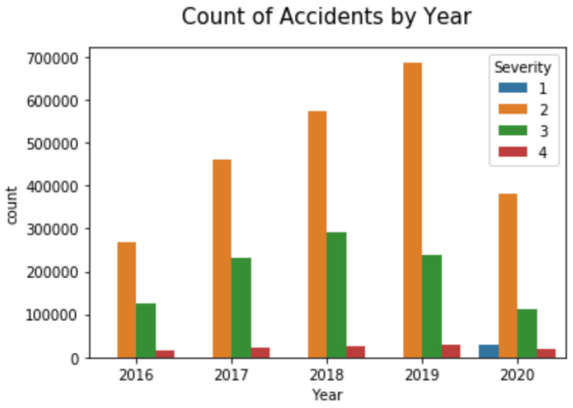
* The test accuracy refers to the accuracy achieved with the validation set

Solution.
Iteration 2

Iteration 2 description

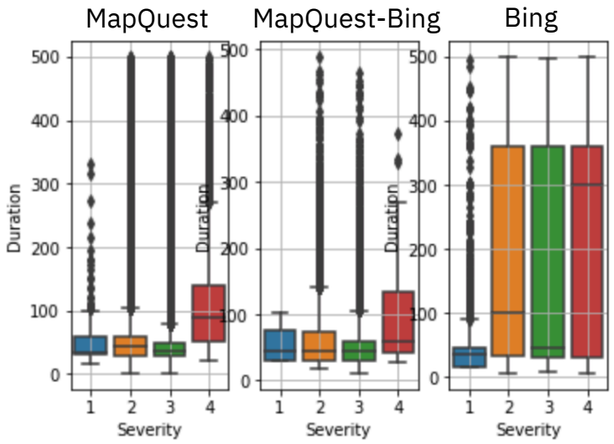
Data Exploration

Time severity analysis



There is a big class imbalance: few Severity 1 and Severity 4 dp. Since no other mayor anomalies were found, I look deeper into the definition of 'Severity'.

Severity definition analysis



There is apparently a different definition of Severity between datasources, so I got rid of the data from Bing.

ETL & Feature Engineering

(same as in Iteration 1)

Typecasting & redundancy:

- *Wind_Direction*, *Weather_Condtions*

Feature relevance and value imputation

Feature transformation:

- *Minute*, *Street* to frequency domain

Feature encoding

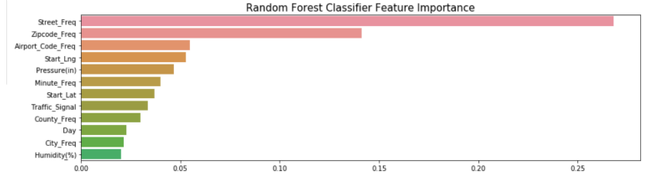
Model Definition, Training & Evaluation

Data augm to only 5'000dp (**10x less dp**), due to limited data for Severity levels 1 and 4.

Model performances:

	Model	Train_Accuracy[%]	Test_Accuracy[%]*
0	Linear SVM	60.5	61.7
1	Decision Tree	91.9	69.4
2	Random Forest	96.9	78.7
3	Neural Network	88.69	70.2

Feature importance:



* The test accuracy refers to the accuracy achieved with the validation set

Iteration 2 key takeaways:

Performance enhancement:

Even though the basic models' performance is worse during the Iteration 2, there is a notable enhancement in accuracy for the Random Forest model. Please note this is a great improvement, since as described in the previous slide, the training data amount used during Iteration 2 was 10x less than for Iteration 1.

Iteration 1:

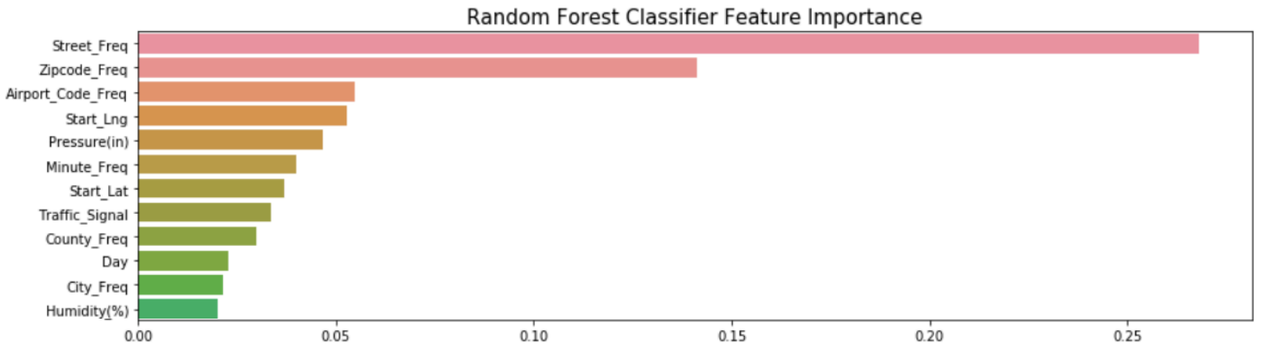
	Model	Train_Accuracy[%]	Test_Accuracy[%]*
0	Linear SVM	66.1	65.7
1	Decision Tree	80.4	70.3
2	Random Forest	95.9	71.8
3	Neural Network	80.21	69.6

Iteration 2:

	Model	Train_Accuracy[%]	Test_Accuracy[%]*
0	Linear SVM	60.5	⌄ 61.7
1	Decision Tree	91.9	⌄ 69.4
2	Random Forest	96.9	⌃⌃ 78.7
3	Neural Network	88.69	⌃ 70.2

Sensitive feature importance:

These results look also more promising. The most important feature is now 'Street_Frequency'. This confirms our hypothesis of both sources (*MapQuest* and *Bing*) defining Severity in different ways. For the data collected by MapQuest, there is a trend towards more frequent accidents having less Severity.



* The test accuracy refers to the accuracy achieved with the validation set

Conclusions & outlook.

The Lightweight IBM Garage Method for DS lead to satisfying results, but there is much room for improvement

Developed a traffic accident forecasting module that allows to compare the performance of different SoA machine learning models. In order to do so:

- The data was first visualized and analyzed for better understanding.
- Data was adapted and features were transformed to be ingestible for predicting modules.
- Different SoA prediction models were created and evaluated.

Some further research could include:

- Study how the performance of the different models is influenced if data with Severity level 1 is ignored (3 class classification), in order have more data available. In order to deal with class imbalance much data was ‘thrown away’ since the data for Severity 1 accidents was very limited.
- Apply more complex models, i.e. an LSTM neural network is often used for time series data forecasting.

Thank you for your attention.

Don’t hesitate to contact me via LinkedIn or GitHub for any questions or feedback.

GitHub: *https://github.com/peroap/USA_Traffic_Accident_Prediction*

LinkedIn: *https://www.linkedin.com/in/pedro-roig-aparicio-27b419126/*

References were checked as per 07.10.2020.