Final Project Report

Fabrizio Rossi, Mina Makar, Matteo Orsini

December 27, 2020

1 Abstract

In this document we present our project for the Foundations of Data Science course. Since United States is one of the busiest countries in terms of road traffic with nearly 280 million vehicles, the level of traffic is one of the reasons leading to more traffic accidents.

In order to help traffic management, we compared the performances of several models to predict the impact on the traffic of an accident. We present an Exploratory Data Analysis of one dataset offered by Kaggle which contains about 3.5 million records of accidents, and after a feature engineering phase, we compared various models implemented using the scikit-learn library showing, which model performs the best.

2 Introduction

The problem we faced in our project was to predict the impact of an accident on the traffic, called severity, which goes from 1, a low impact on the traffic, to 4, an high impact on the traffic. We thought that predicting this value could be useful for an automated traffic control service, in order to take decisions according to the predicted severity. So we tried to answer the question: "What is the severity of an accident which happens in a particular place and in some particular conditions?".

3 Related work

As related work we inspected the several notebooks available on Kaggle and on the Web which used the same dataset, or tried to accomplish the same task. In this Medium article (Link), there is an in-depth analysis of the dataset state by state, and a comparison of different models deployed on our chosen dataset where the Random Forest performed the best.

Here (Link), instead, there is a Kaggle notebook in which the severity of an accident is predicted using just the description. This (Link) other Kaggle notebook, presents an approach which only predicts if an accident will have severity 4 or not. In this paper (Link) a deep neural model is proposed to predict in real-time a geographical region of reasonable size, and a fine-grained time period in which an accident could occur using an augmentation of the dataset. This Kaggle notebook (Link) contains a similar work to ours but it's limited just on the Fairfield county.

In the end we used two works to compare our results: this Kaggle notebook (Link) and this Medium article (Link), which both used the same dataset to carry out our same goal using different models.

4 Dataset analysis

We based our work on a dataset found on Kaggle (Link), which contains about 3.5 million of records about accidents happened in the US. In the Table 1, the features contained in the dataset are shown with a brief description on the second column.

In the attached notebook, we carried out an additional **Exploratory Data Analysis (EDA)** that we will not report in this document.

5 Proposed strategy

In this section we will describe our methodology of work to treat the features included in the dataset. First of all, we decomposed the **Start_Time** feature in year, month, day, weekday, hour and minutes, in order to feed them to the models.

Then, we computed the **correlation matrix** among all the feature, to inspect if we could remove some features highly correlated.

From the matrix, shown in Figure 1, we can see that the start and end GPS coordinates of the accidents are highly correlated.

In fact, from the medium distance shown in the EDA, the end of the accident is usually close to the start, so we dropped the end GPS coordinates.

Name	**Description**
ID	Unique identifier of the accident record
Source	API that reported the accident. Can be ”Bing”, ”MapQuest”, or ”MapQuest-Bing”
TMC	Traffic Message Channel code, which provides a description of the accident and sometimes the impact on the traffic
Severity	Value between 1 and 4 which describes the impact on the traffic
Start_Time, End_Time	Report the accident time. In particular, End_Time reports the timestamp at which the effects of the accident on the traffic flow ended
Start_Lat, Start_Lng, End_Lat, End_Lng	Report the GPS coordinates of the start and end point of the accident
Distance(mi)	The length of road affected by the accident
Description	Natural language description of the accident
Number, Street, City, County, State	Indicate the place where the accident occurred
Side	Show the side of the road in which the accident occurred, can be ”L” for left, or ”R” for right
Timezone	Timezone of the accident
Airport_Code	Indicates the closest weather station to the location of the accident
Weather_Timestamp	The time-stamp of the weather observation
Temperature(F), Wind_Chill(F), Humidity(%), Pressure(in), Visibility(mi), Wind_Direction, Wind_Speed(mph), Precipitation(in), Weather_Condition	Report the conditions of the weather when the accident occurred
Amenity, Bump, Crossing, Give_Way, Junction, No_Exit, Railway, Roundabout, Station, Stop, Traffic_Calming, Traffic_Signal, Turning_Loop	Report the point of interest near the accident, can be true or false
Sunrise_Sunset, Civil_Twilight, Nautical_Twilight, Astronomical_Twilight	Report the period of the day (day or night) according to the subcategory of twilight

Table 1: Dataset features description

Moreover, the wind chill is directly proportional to the temperature, so we decided to drop the **Wind_Chill(F)**.
From the matrix we can also note that we couldn’t compute the covariance with **Turning_Loop**, and that’s because it’s always False.
Upon a further analysis, we also decided to drop the following features for the reasons reported:

- **ID, Source**: since they don’t carry any information for the severity, and we wanted to reduce the number of features;
- **TMC**: because it could already contains information about the accident severity that should be not available for the prediction when an accident is reported. For example, TMC 106 means ”stationary traffic for 10 km”, but we should not have this information, since it’s our goal to predict it;
- **End_Time**: because we cannot know in advance when the traffic flow will become regular again;
- **Description**: most descriptions only report the name of the road of the accident, and so we decided to omit this feature for simplicity;
- **Number, Street, County, State, Zipcode, Country**: because we just focus on the City where the accident happened;

- **Timezone, Airport_Code, Weather_Timestamp**: because we think they are not useful for our task;
- **Turning_Loop**: since it's always False;
- **Sunrise_Sunset, Nautical_Twilight, Astronomical_Twilight**: because they are redundant.

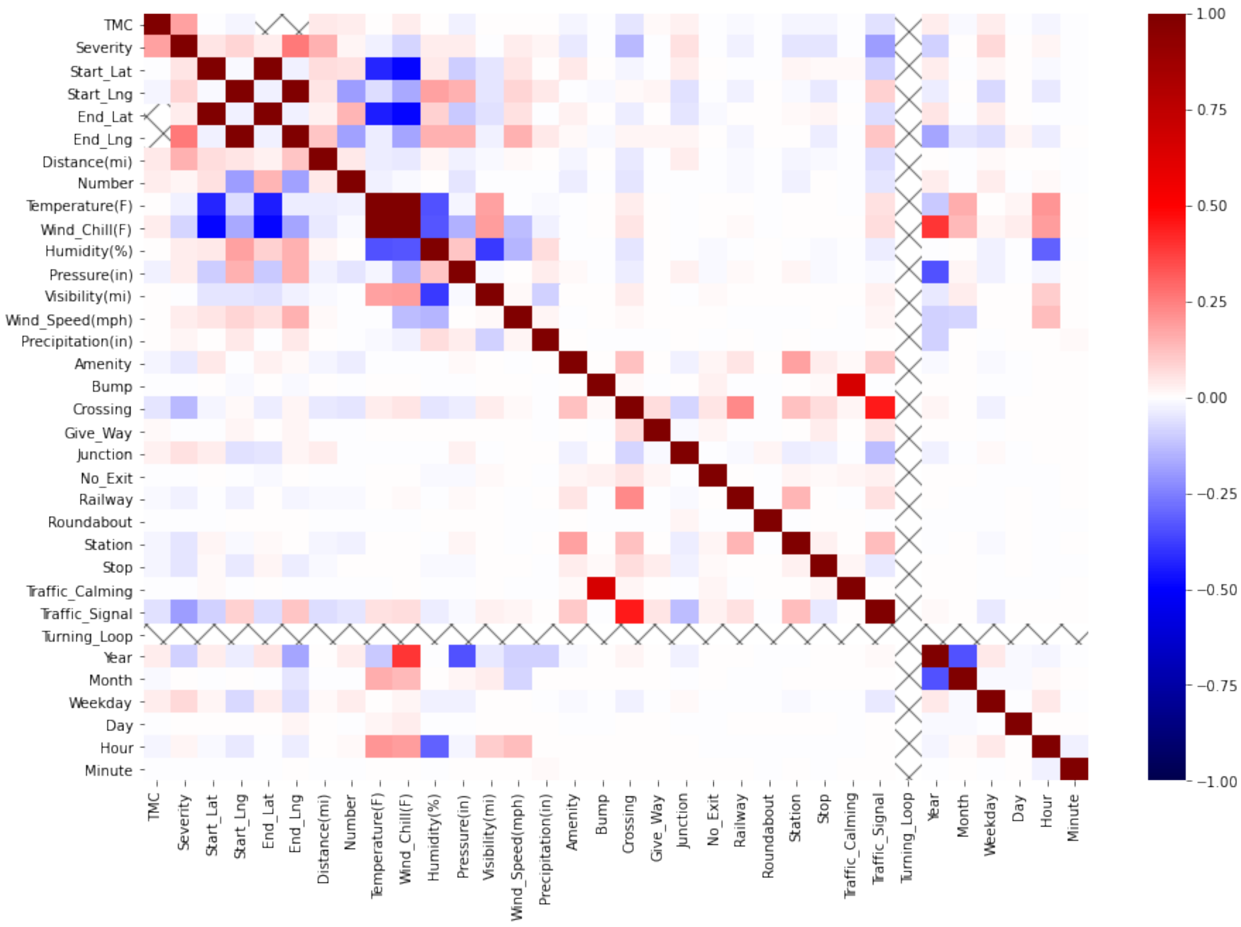

Figure 1: Correlation matrix between the dataset features

Then, we **removed duplicated** records and we inspected the dataset to check if there were some anomalous or missing values. We **fixed some errors** in the Pressure(in) and the Visibility(mi) features, by removing records with impossible values, and we reduced the number of weather conditions and wind directions by combining similar but not equal values to reduce the number of classes for those features.

In the end we **fixed missing values**, by filling them with the mean, for numerical features, and by dropping the entire record for categorical values, since the dataset contains lots of data.

At this point, we were almost ready to feed the dataset to the Machine Learning models, but first we needed to handle a few more aspects.

The most important of all was the number of records for each severity: as we can see from Figure 2 the dataset was highly unbalanced.

For this reason, we used the **undersampling** technique to even the number of records to the minority class which is severity 1. After the operation, each class contained about 26.000 records.

Then, we proceeded to **scale the numerical features** using the MinMaxScaler, and to **encode the categorical ones**: boolean features were encoded simply transforming false and true to 0s and 1s; the remaining ones were encoded using the one-hot encoding except City, which was encoded using binary encoding since it would have produced too many columns (about 12.000).

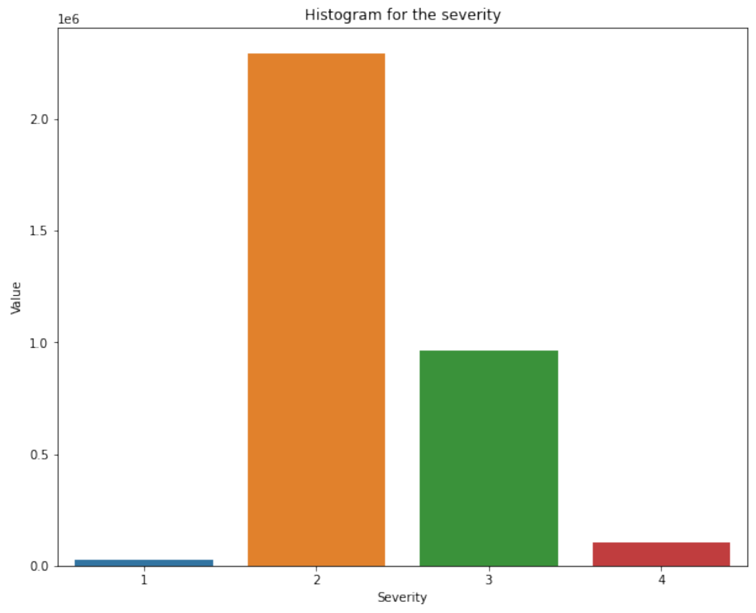

Figure 2: Number of records for each severity

6 Experimental results

In this section we present the results obtained with several Machine Learning models.

We **split the dataset** into three parts: 64% of the records formed the training set, which was used to train the models, 16% formed the validation set, used to tune the hyperparameters, and the remaining 20%, formed the test set, used to evaluate the best model.

We tested the following models to find the best one:

- Logistic Regression;
- Support Vector Machine;
- Decision Tree;
- Random Forest;
- Naive Bayes;
- Multi-Layer Perceptron.

For each model we tuned the hyper-parameters on the validation set, either by using the grid search or by checking manually some values.

For the **Logistic Regression** model, we experimented with the following solvers: Newton's solver, SAG and SAGA. We obtained the same accuracy scores , but Newton's solver took almost four times to run due to the computational power required to compute the inverse of the Hessian matrix.

Hyper-parameters tuning for the **SVM** was conditioned by the high computational cost needed by the training of the SVM. So, we used a small number of records to feed the grid search, and the best parameters found could be not the true best parameters for the entire dataset. For 5.000 records, the best parameters are: C = 1.0, kernel = polynomial with degree 2.

With respect to the **Decision Tree** model, we saw that without limiting its max depth, we would get a little bit of over-fitting since we scored 100% accuracy on the train set, while obtaining 68% accuracy on the validation set. Using 10 as max depth, we reduced the over-fitting obtaining an higher accuracy of 74% on the validation set.

For the **Random Forest**, we saw that increasing the number of estimators and the max depth, we obtained better results, but increasing the risk of over-fitting. We obtained the best results using 500 estimators and 30 as maximum depth for each tree.

We saw that for **Naive Bayes** models, the best one is the Bernoulli Naive Bayes probably because most of the feature are just binary features which can assume 0s or 1s as values.

For the **MLP** we tested the architecture, the solver, the activation functions and the number of epochs. From the results we could only see that Adam solver obtains slightly better results than SGD.

The accuracy obtained by each model on the validation set is shown in the Figure 3. Logistic Regression, SVM and Bernoulli Naive Bayes have comparable performances while the most successful models are the tree based ones: the Decision Tree and Random Forest. We can see that the latter is the best performer followed by the Multi-Layer Perceptron.

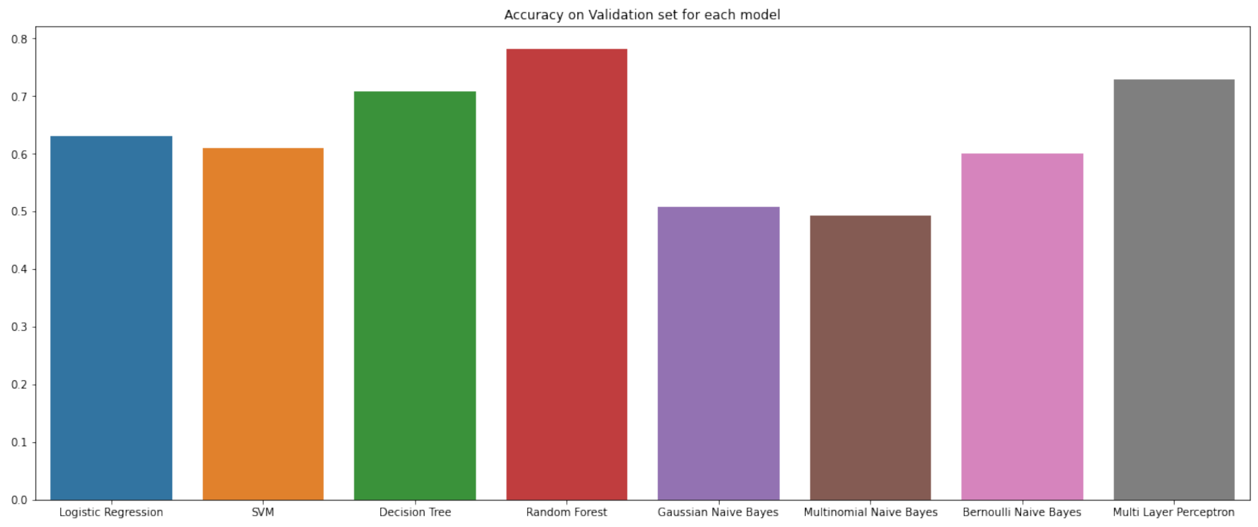

Figure 3: Accuracy obtained for each model

In Figure 4 and 5, we can see respectively the ROC and Precision-Recall curves for each model, confirming what we said about the accuracy.

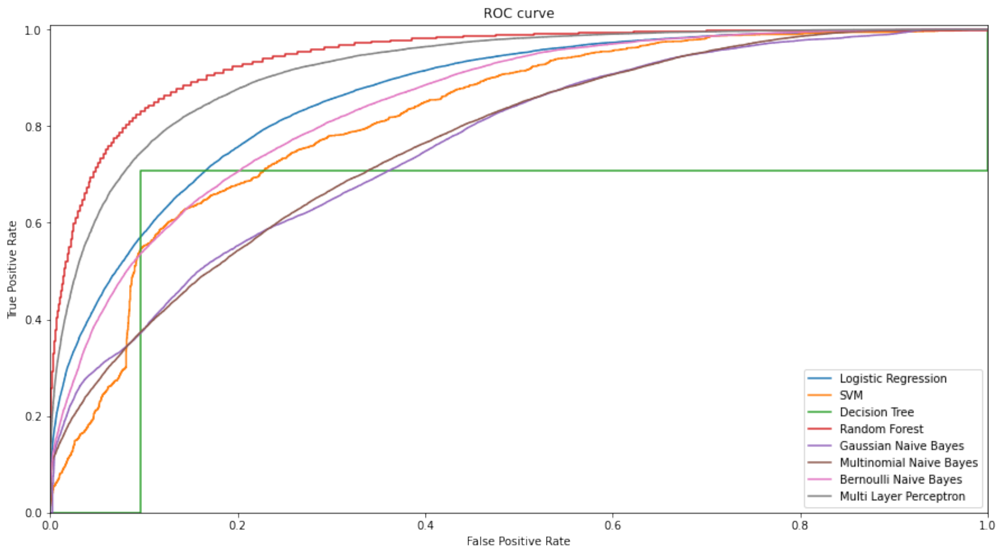

Figure 4: ROC Curves computed for each model

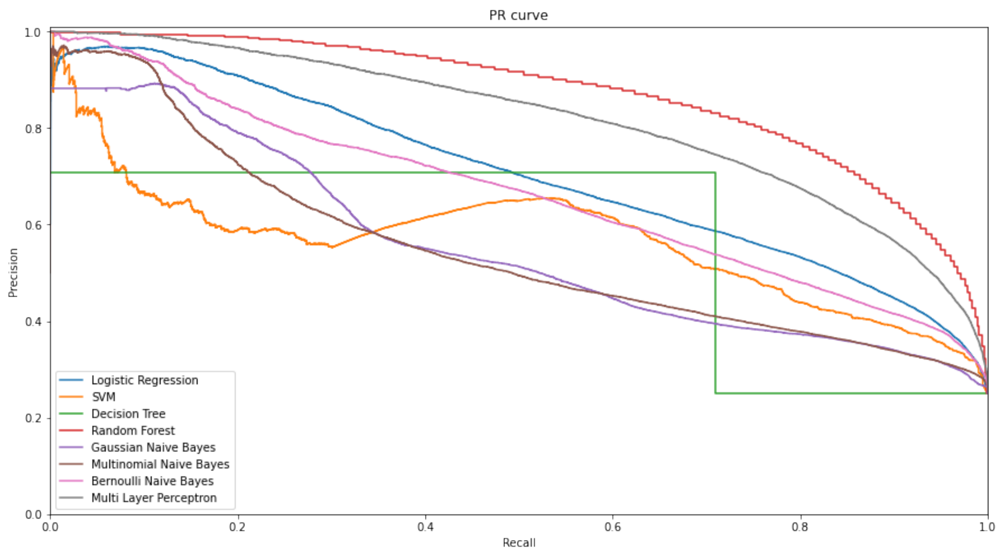

Figure 5: Precision-Recall Curves computed for each model

Since the **Random Forest** is the best model for our task, we also computed the confusion matrix, depicted in Figure 6, where we can see that the most accurately classified classes are severity 1 and 4. In the Table 2, instead, we reported the metrics collected from this model on the test set.

Class	**Precision**	**Recall**	**F1-Score**
1	0.88	0.95	0.91
2	0.73	0.57	0.64
3	0.69	0.63	0.66
4	0.73	0.91	0.81
Macro-AVG	0.76	0.76	0.75

Table 2: Precision, Recall, F1-Score for the Random Forest model computed on the test set

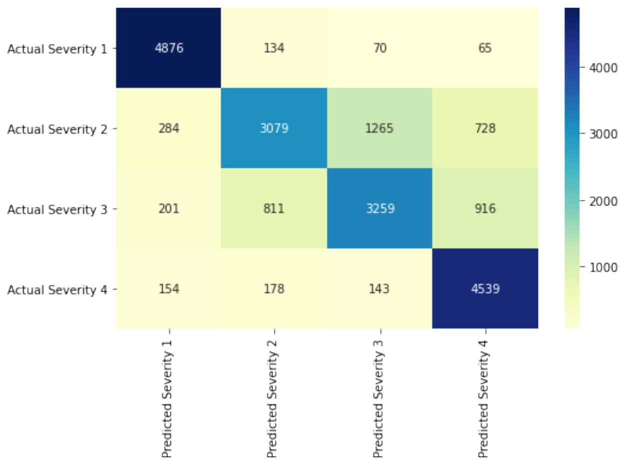

Figure 6: Confusion matrix of Random Forest model on test set

In the end, in the Figure 7, we present a comparison between our best model, the Random Forest, which obtained 76% as the accuracy, and a few models that we found inspecting the related works that used the same dataset. As we said in the related work section, we found a Kaggle notebook reporting the results obtained using a Neural Network and a Medium article which used several models to accomplish the same task as us.

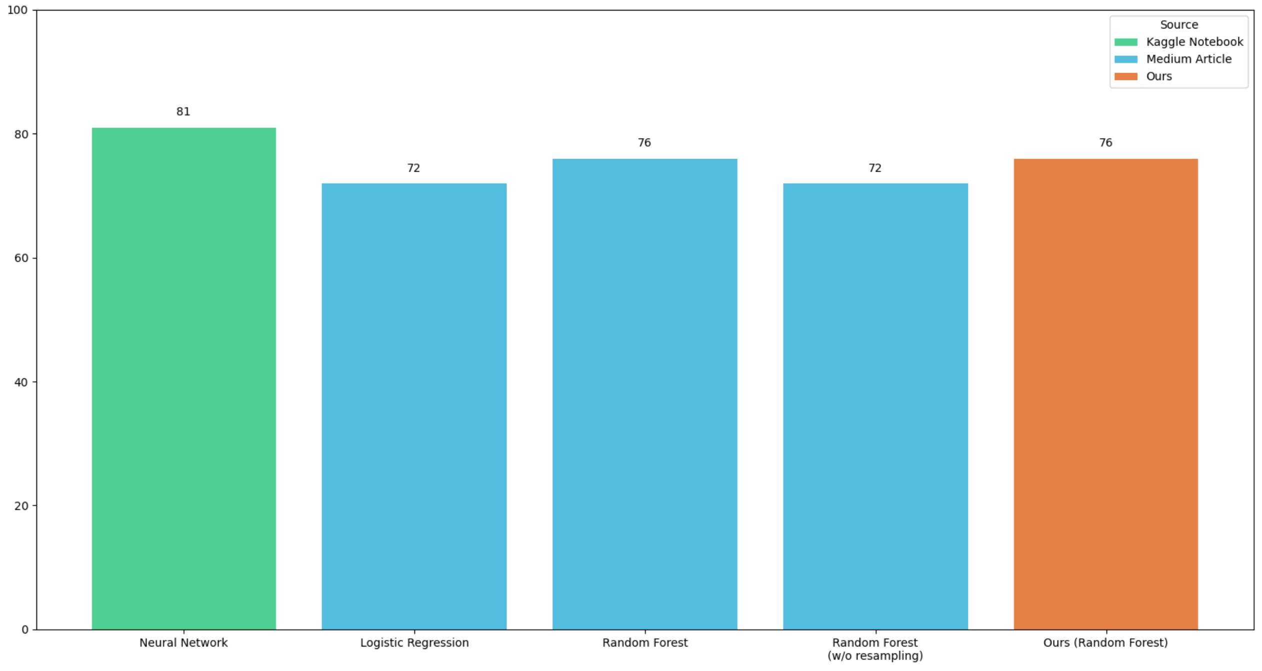

Figure 7: Accuracy comparison between different related models

7 Conclusions

In conclusion, we can say that accidents with severity 2 and 3 are the most difficult to predict as shown by the confusion matrix for each model. Instead, when the accident has a severity of 1 or 4, the prediction is more accurate as shown by the precision and recall of those classes.

8 Contributions

Mina Makar

- Exploratory Data Analysis:
  - Most frequent words in the description of an accident with severity 4
  - Number of accidents for weekday
- Data preprocessing:
  - Handle unbalanced data
  - Feature scaling
- Model:
  - Logistic Regression
  - Decision Tree

Matteo Orsini

- Exploratory Data Analysis:
  - Number of Accidents per State
  - Weather condition histogram
- Data preprocessing:
  - Feature addition
  - Drop duplicates
  - Feature encoding
- Model:
  - Support Vector Machine
  - Random Forest
  - Multi Layer Perceptron

Fabrizio Rossi

- Exploratory Data Analysis:
  - Most frequent road features
  - Medium distance by severity
- Data preprocessing:
  - Check correlation between features
  - Handle erroneous and missing values
  - Check feature variance
- Model:
  - Gaussian Naive Bayes
  - Bernoulli Naive Bayes
  - Multinomial Naive Bayes