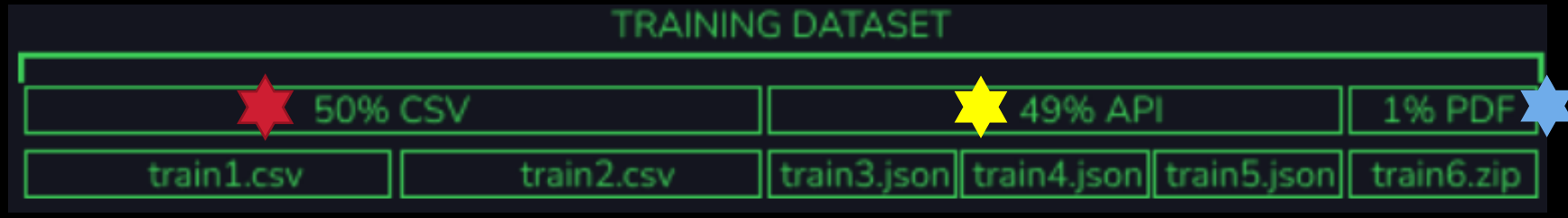
SEHackathon

Carlos Núñez
Ingrid Sancho

Preprocessing



We imported the .cvs easily with the **pandas** command read_csv. We took in consideration the different separators used in each csv

With the given URLS we opened them with the **urllib** library and then with pandas function .read_json we got the text and converted it to csv

We first unzipped the file with the use of **zipfile** and **os** libraries. Then we looped for each file, read it and extracted the important information:

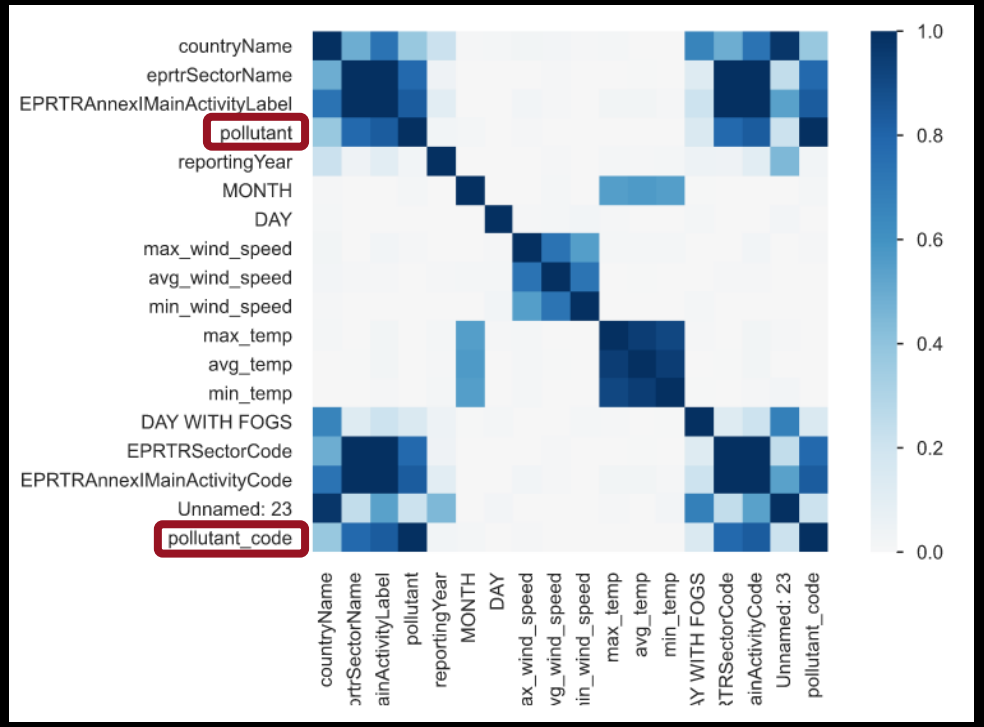
· **Emissions** was a variable that did not appear in any of the other datasets, therefore we chose not to take it into account since we would have had 99% of nulls in this column

· **max/min/avg of wind and temp** were variables that were corrupted, we tried to make sense of them (changing max and min values) but we decided to not spend more time on those variables since we son found out they were not going to be very useful for our classification model

· **EPRTRAnnexIMainActivityLabel** did not appear in the pdfs, therefore we retrieved such information from EPRTRAnnexIMainActivityCode with the information of the other training datasets

At the end we ended up concatenating the 6 datasets and creating the “pollutant_code” column

Descriptive Analysis



We decided to start by doing an Exploratory Data Analysis. From which we want to point out the Correlation Matrix that you can see on the left.

We find it very interesting since you can clearly see how the time, wind and temperature variables are completely independent from the "pollutant"/"pollutant code" variable. Because of this we decided to not focus on these columns but rather do our model based on the columns that have at least some correlation with our target variable.

The columns we decided to keep going with are mostly categorical variables, which is something we needed to fix before building the model. We first tried to transformed them to dummies but we ended up with a massive database which was too time-consuming to process. Therefore, we decided to rethink our approach and use **Label Encoders**!
When using those we needed to be careful and ensure that the test dataset also had the same labels associated to the different variables, therefore we followed the following steps:

1-we joined the train and test datasets and created a new column ("TRAIN/TEST") that distinguished them.
2-we used Label Encoder to transform categorical variables to numeric ones.
3-we separated the data frames back with the use of the previously mentioned variable and eliminated what was necessary.

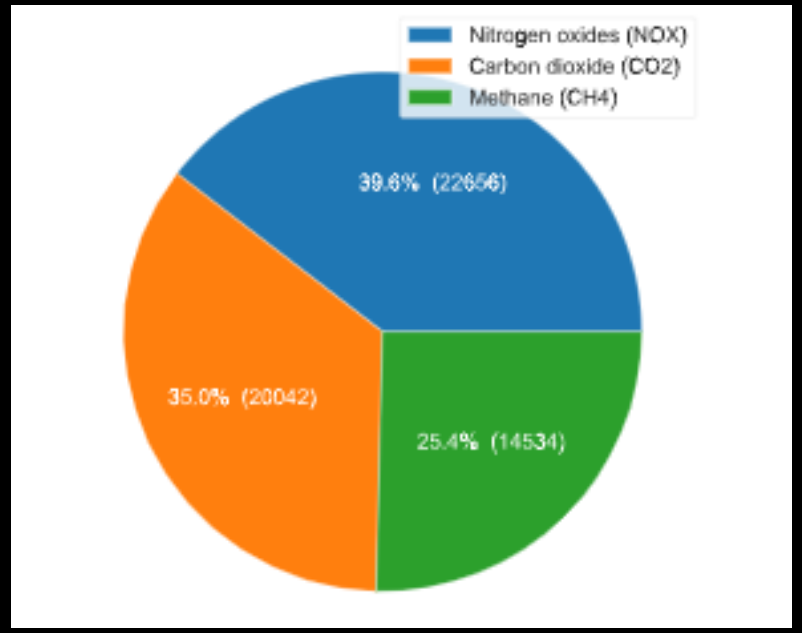
Model Creation

We first split our train dataset in an 80/20 way so that we could both train and test the models!

We then ran 2 classification models:

- Random Forest (<60% accuracy)
- Decision Tree (<60% accuracy)

We got the results and realized that, because of the imbalance in the dataset, some pollutants were predicted better than others, As you can see in the pie plot, we do not have a uniform distribution of the 3 variables, therefore we applied imblearn function SMOTE that made sure that we had the same number of registers for the three gases.



Then we decided to run both classification models again with the balanced dataset:

- Random Forest (>65% accuracy)
- Decision Tree (>65% accuracy)

Tests

With the best performing model (Random Forest) we predicted the pollution_code column for the actual test dataset (test_x.csv). Then we prepared the output so that it contained the “test_index” and “pollutant” columns.

test_index	pollutant
0	2
1	2
2	0
3	2
4	1
5	1
6	1

On the left we have as a sample the first 7 registers of the prediction and on the right, we have the distribution of the pollutant variable of the predictions.

We can see that, similarly to the train test, the Methane (2 label) es the one with fewer registers even though we trained the model on a balanced dataset.

To us, this is a good indicator of the prediction since we assume that the test dataset and train datasets have a similar behavior, and this is reflected in the target variable