WHEN MACHINES TALK

a data challenge by SNCB-NMBS

check out our GitHub repository

TEAM MEMBERS

Jorge Ignacio del Río Sánchez: 000607231
Kristóf Balázs: 000612294
Neri Pérez Sebastian Alberto: 000605855
Stefanos Kypritidis : 000606810

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- CUSTOM PIPELINE
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- CUSTOM PIPELINE
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

Context of the problem and description of the challenge

Recent railway vehicles are equipped with sensors on most of their Subsystems. The latter report some states via a wired network to a central on-board computer

They receive thousands of sequences of events from thousands of vehicles. Since machines tend to degrade, sometimes technical failures appear

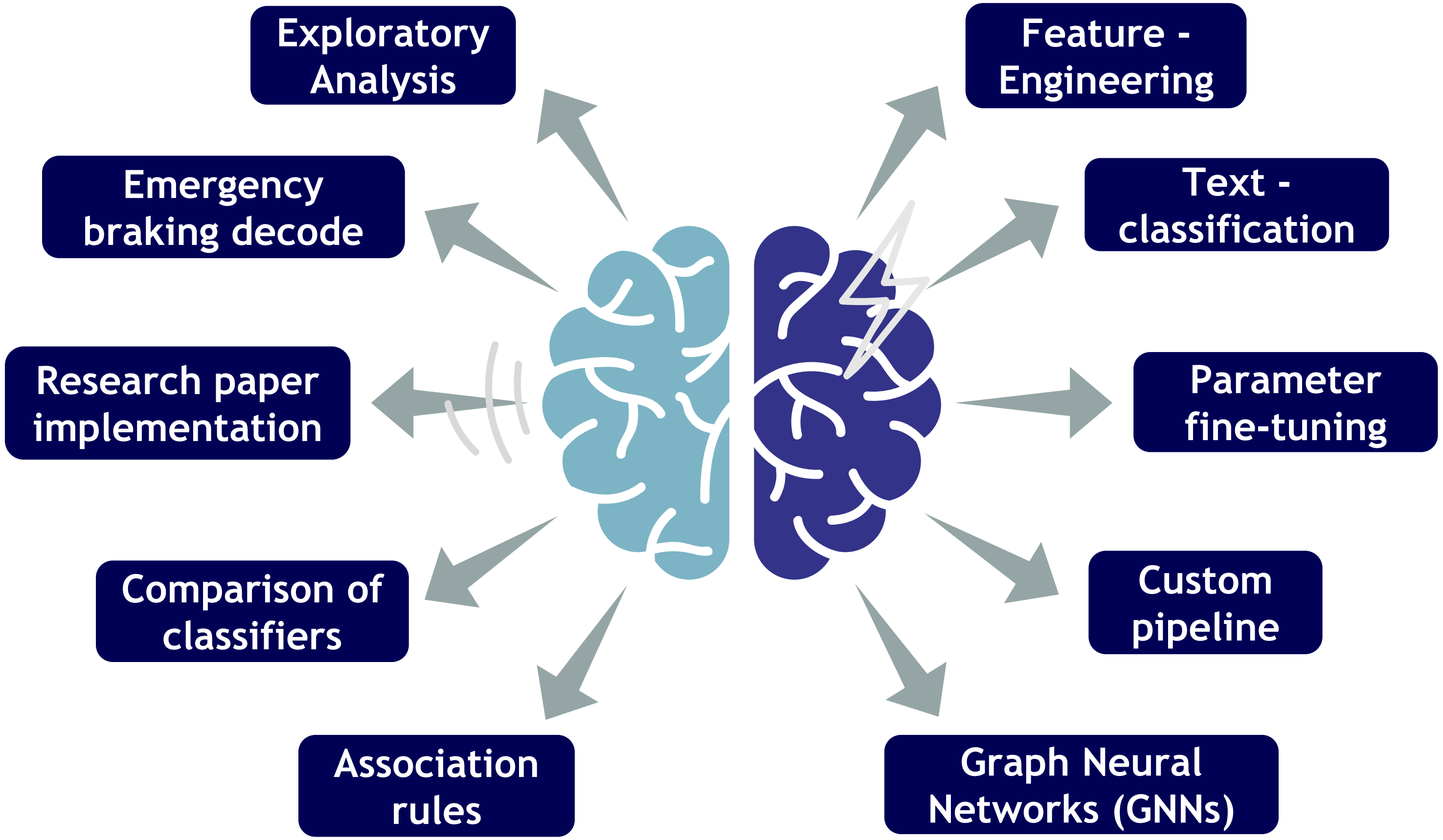
With that information, they labeled dataset of sequence of events with technical incident types.

The main challenges are to, find sub sequences of events (scenarios) that seem to be highly associated to some types of incidents and automatically suggest incident types based on new sequences of events

Source: SNCB presentation challenge

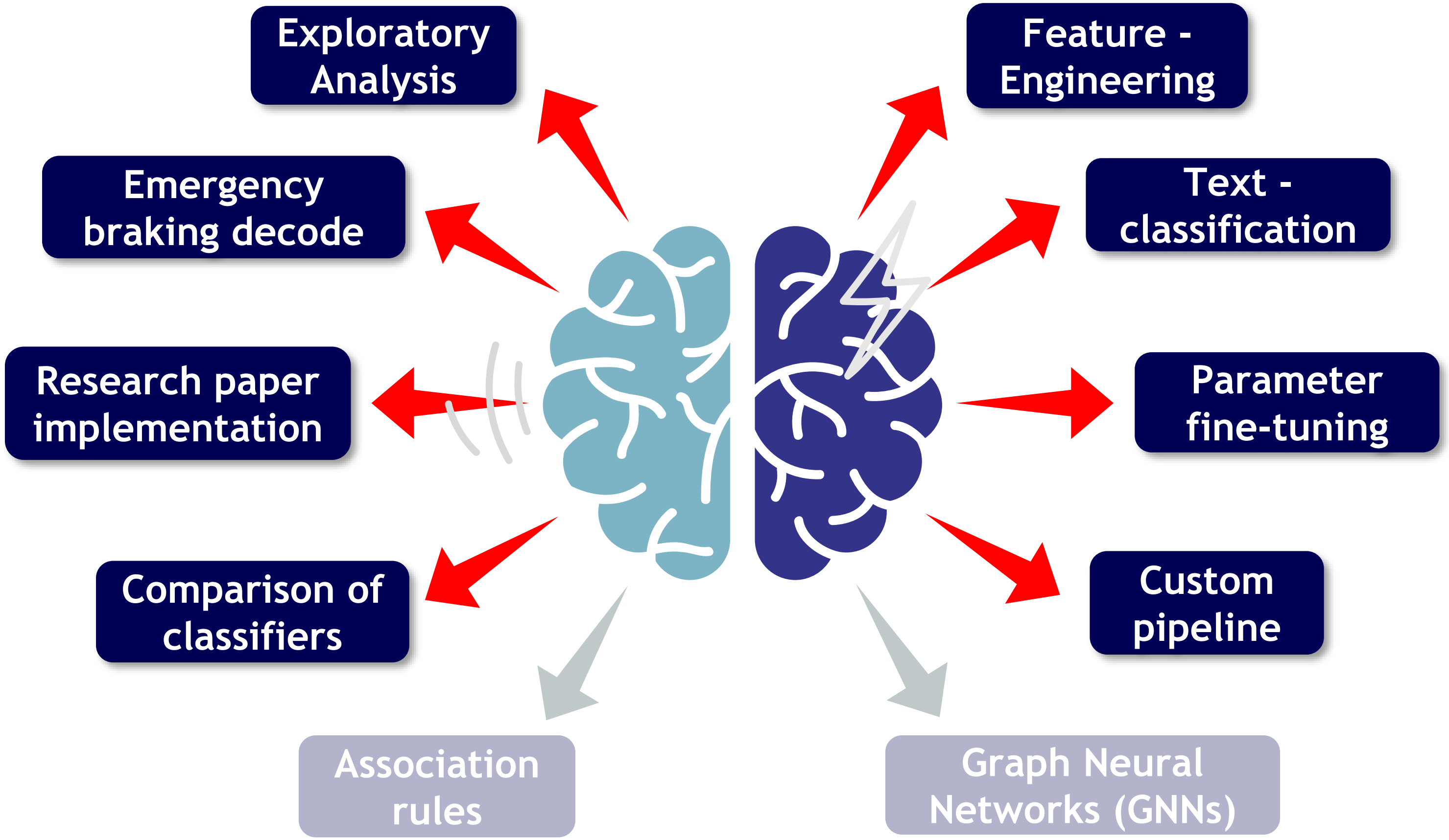
Idea Map to address the challenge

We used several methodologies to explore options and different points of view, ranging from exploratory analysis and feature engineering to applying multiple types of models



Idea Map to address the challenge

We used several methodologies to explore options and different points of view, ranging from exploratory analysis and feature engineering to applying multiple types of models



CONTEXT OF THE PROBLEM

DATA AND EXPLORATORY ANALYSIS

RESEARCH PAPER METHOD

TEXT CLASSIFICATION

CUSTOM PIPELINE

RESULTS

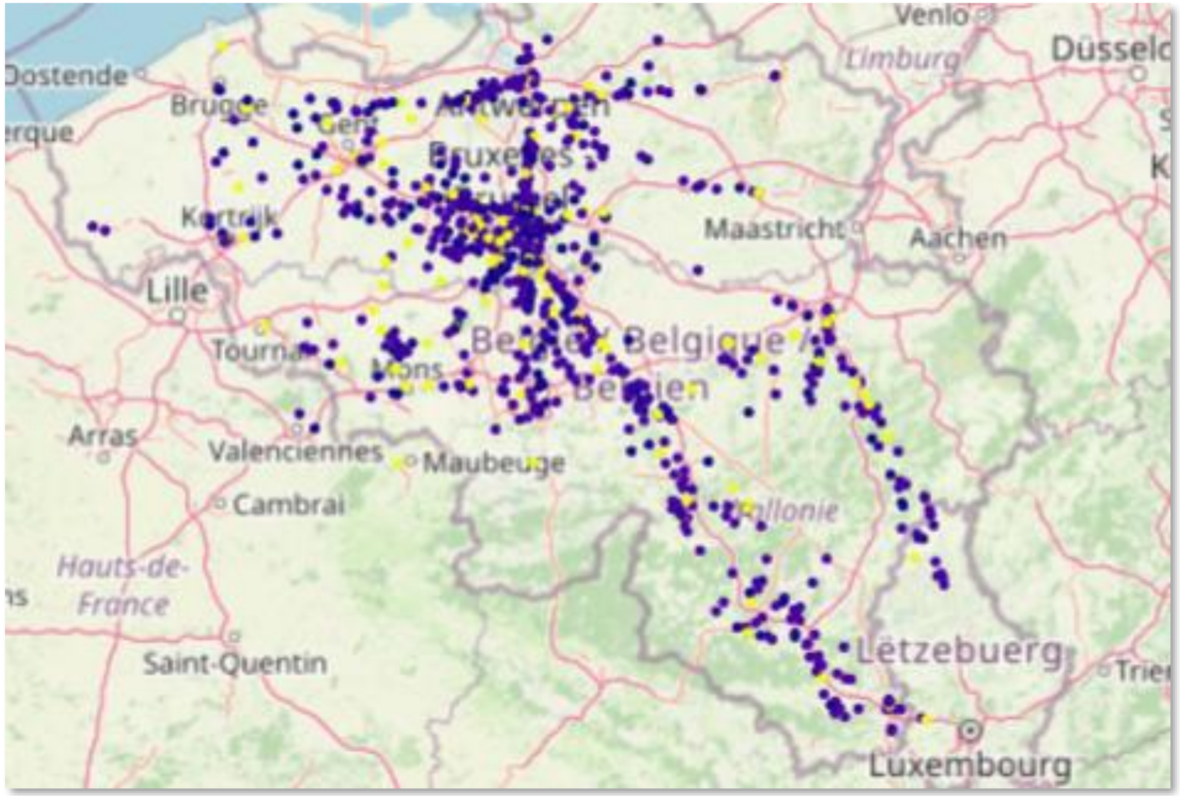
DISCUSSION AND CONCLUSION

FUTURE STEPS

General description of the dataset

As expected, most events are in Belgium, with high density near Brussels. However, some "data errors" show incidents in Africa.

Map of incidents



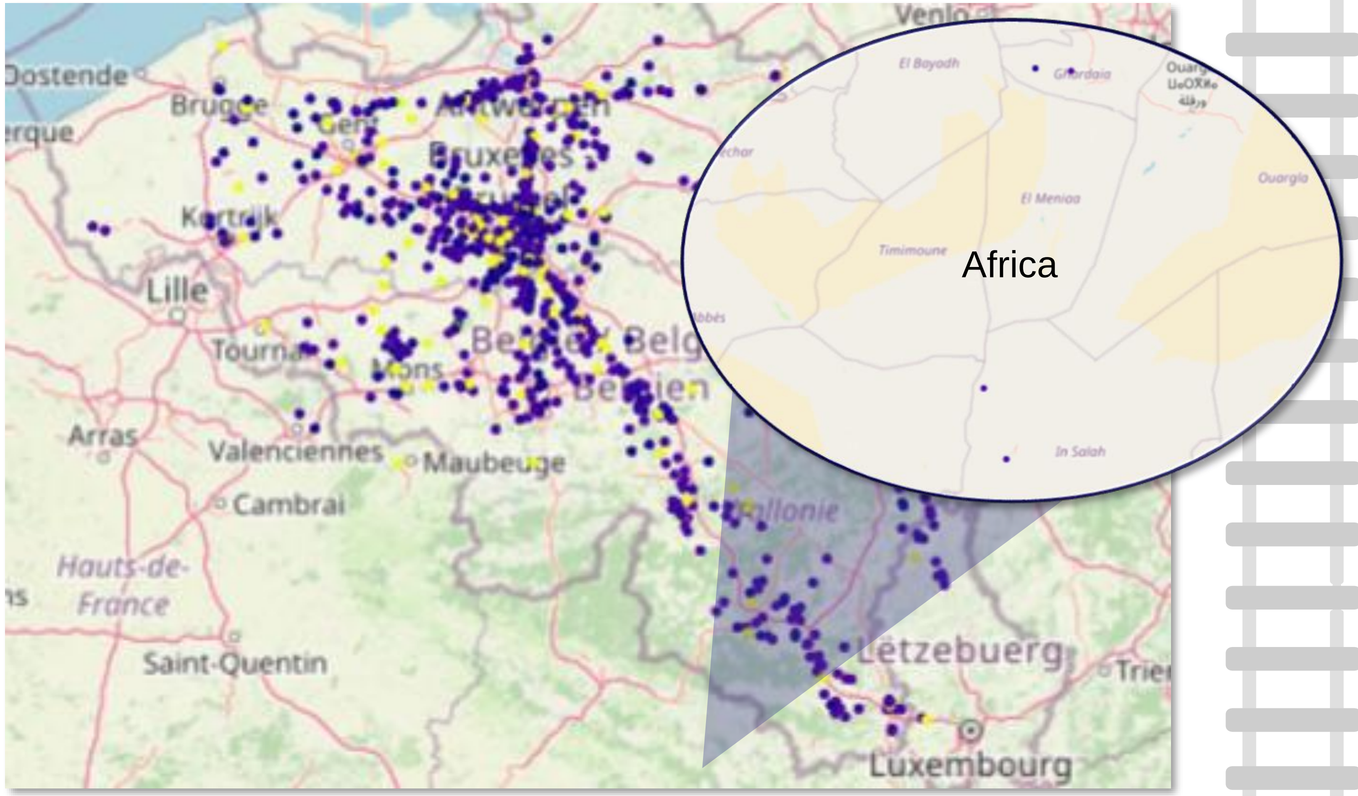
Main columns	Description
Incident ID	ID of an incident
Vehicle sequence	List of sequence of vehicles that have reported an event
Event sequence	List of sequence of events reported by each vehicle
Seconds to incident sequence	List of the time in seconds to the incident

Source: Exploratory analysis from SNCB data challenge

General description of the dataset

As expected, most events are in Belgium, with high density near Brussels. However, some "data errors" show incidents in Africa.

Map of incidents



Main columns	Description
Incident ID	ID of an incident
Vehicle sequence	List of sequence of vehicles that have reported an event
Event sequence	List of sequence of events reported by each vehicle
Seconds to incident sequence	List of the time in seconds to the incident

Source: Exploratory analysis from SNCB data challenge

General description of the dataset

The first overview of the data show us that the dataset is “small” and the classes to be predicted are imbalanced. This indicates that we have to deal with those inconvenient when developing the models

Incident type frequency

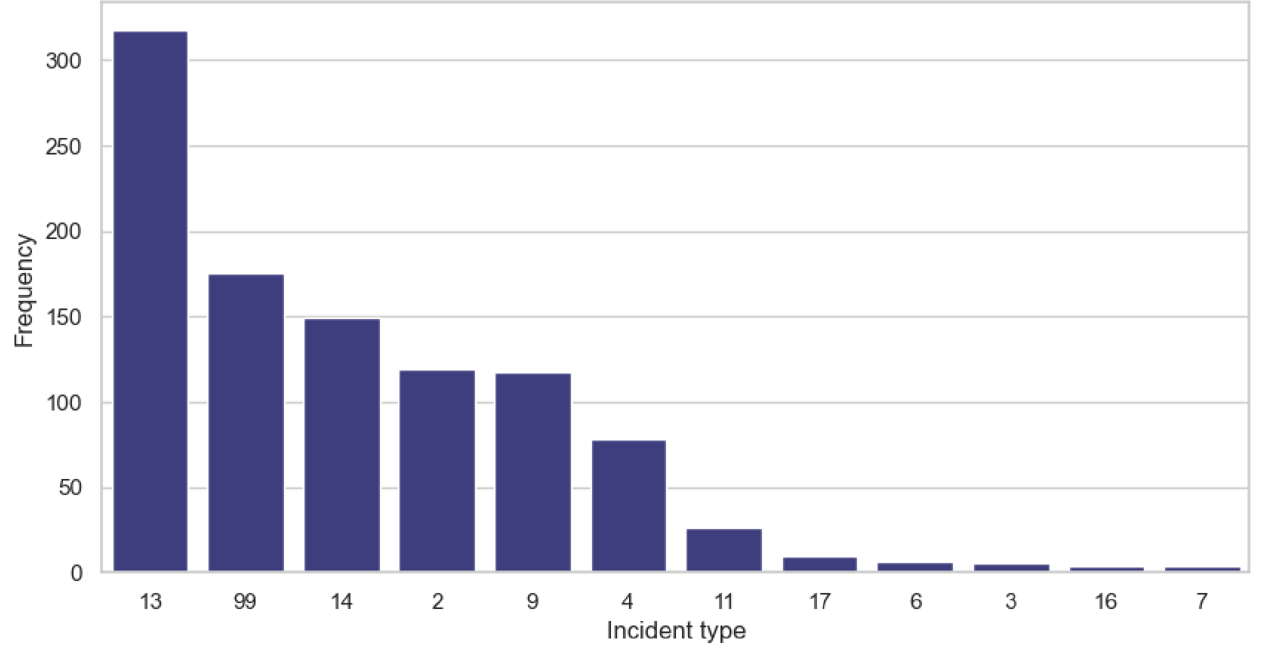


Table description

Dataset with 1011 rows and 10 columns. No null values in the attributes

There are 12 different classes that represent the incidents.

Dataset is heavily imbalanced. There are at least 5 minority classes.

Source: Exploratory analysis from SNCB data challenge

General description of the dataset

The first overview of the data show us that the dataset is “small” and the classes to be predicted are imbalanced. This indicates that we have to deal with those inconvenient when developing the models

Incident type frequency

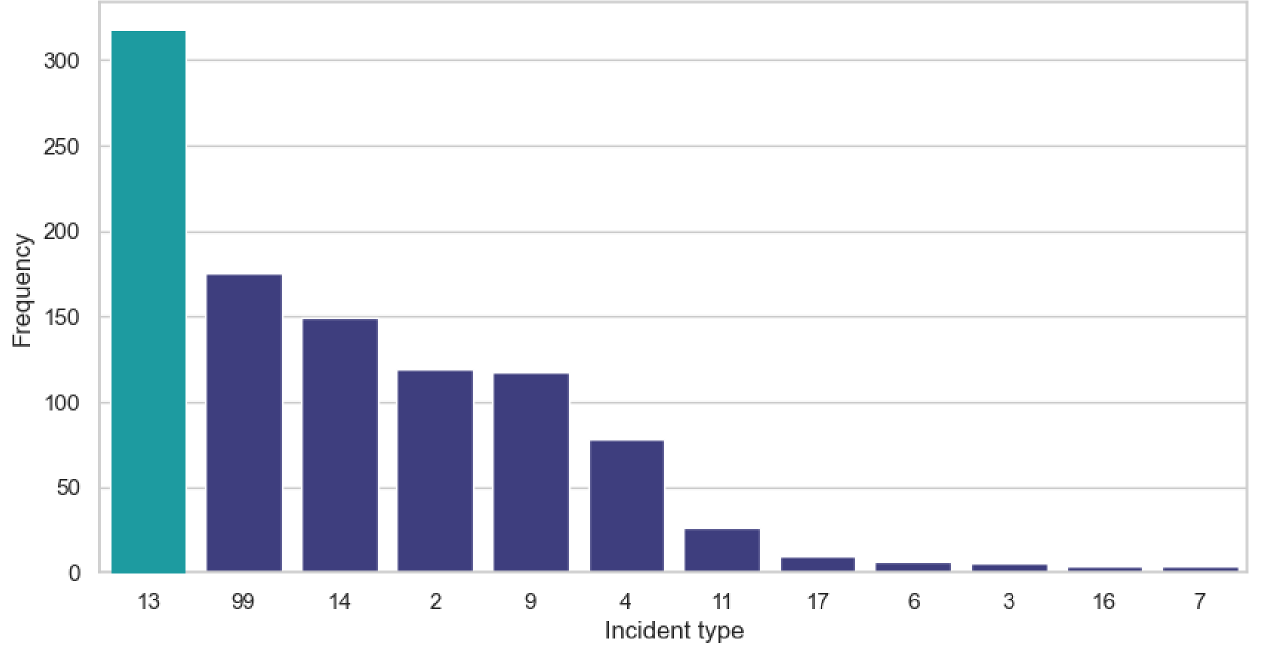


Table description

Dataset with 1011 rows and 10 columns. No null values in the attributes

There are 12 different classes that represent the incidents.

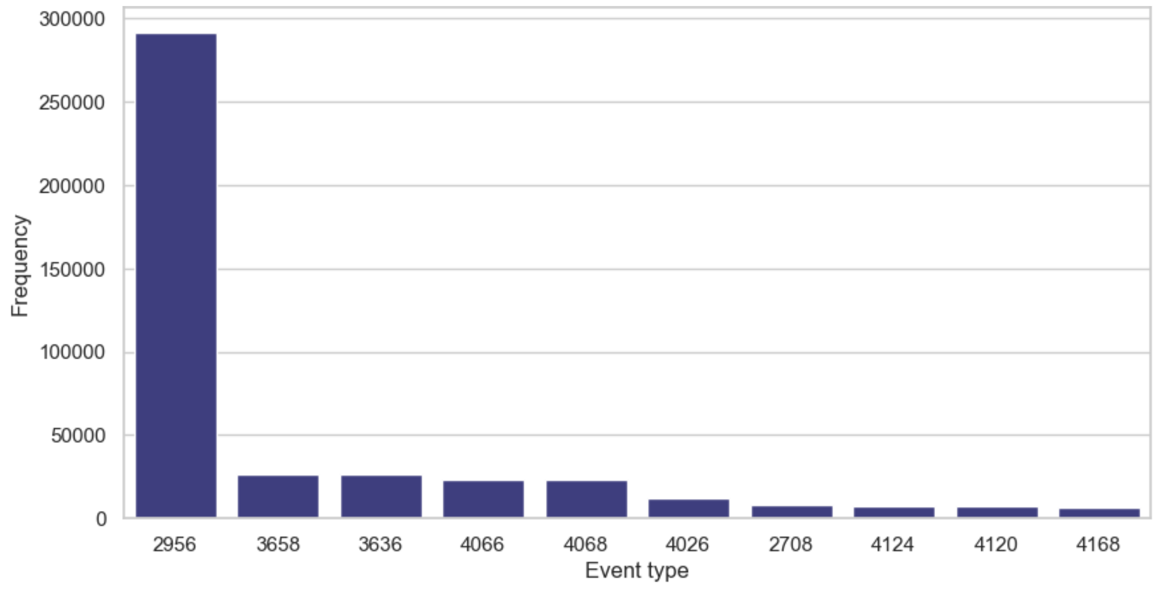
Dataset is heavily imbalanced. There are at least 5 minority classes.

Source: Exploratory analysis from SNCB data challenge

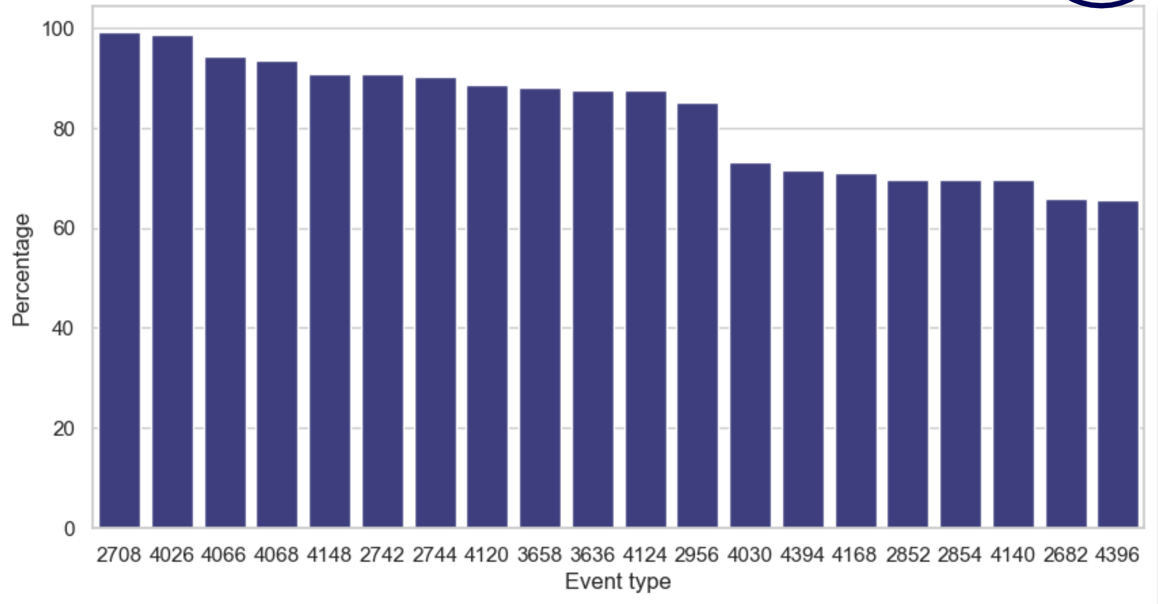
Incidents and event type frequency

It is possible to observe that some event types appears in more than 90% of the events sequences, which show us that those could not provide valuable information

Event type frequency top 10



Event type percentage top 20



- Event type 2956 happens very often compared to the other events
- There are 917 events that occur at least one time in this dataset

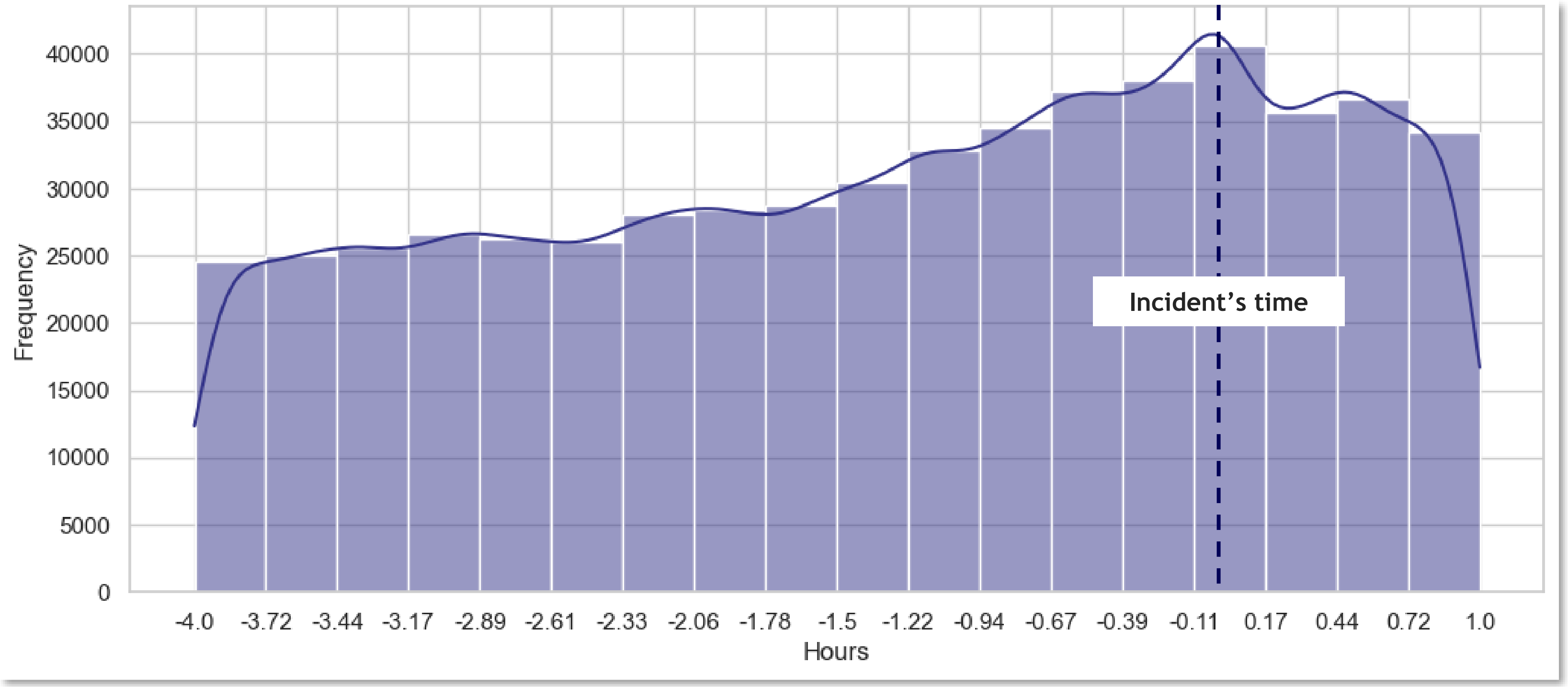
- Multiple events appear in more than 90% of the events sequences. Those do not provide enough information
- Event types that appears with less frequency could provide more valuable information

Source: Exploratory analysis from SNCB data challenge

Chronology of the events

We observed that it is possible to find events 4 hours before the incident and one hour after the incident. Most of the registered events happened before incident

Histogram of the incidents time



Key points

- It is possible to observe a peak just before the event happens
- Most incidents registered occurs before the events

- It is possible to filter incidents using the time criteria.
- For example, braking emergency incidents could happen seconds before event

Source: Exploratory analysis from SNCB data challenge

Emergency Braking Analysis

The latter gave us the idea of identifying Emergency Braking Codes for Feature Engineering for the Classification Model

General steps to develop the Braking emergency feature

Calculate the change in speed (Δv) in m/s and the change in time (Δt) in s between subsequent events. Determine the deceleration in m/s^2 between subsequent events

Identify the event codes where:

>The end speed is 0

>Deceleration is greater than 1 m/s^2

>Δt is less than 10 s, but greater than 0 s

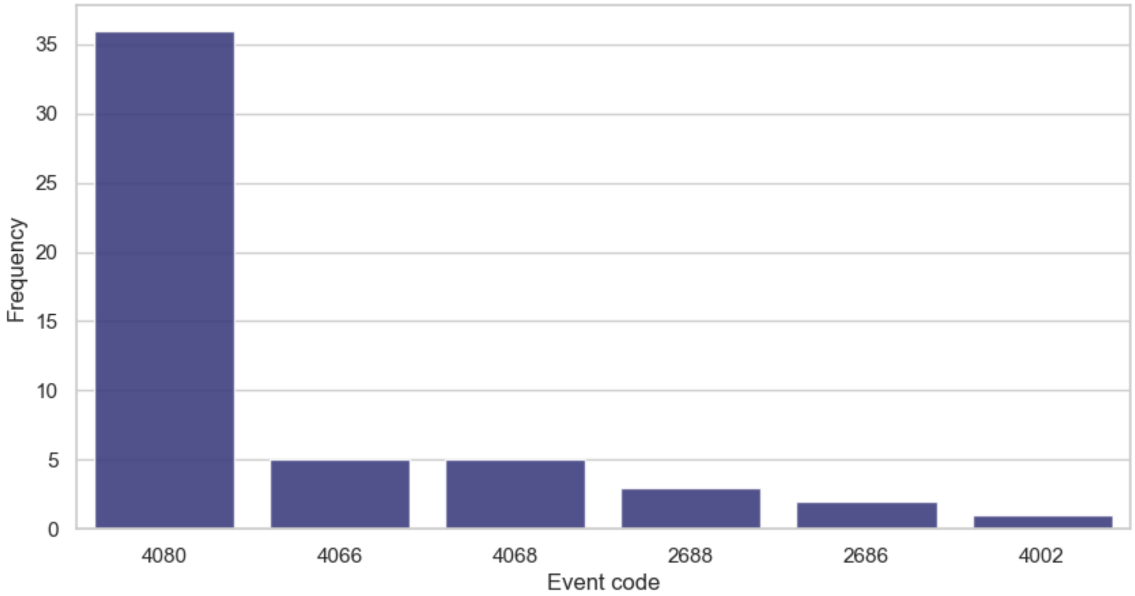
Calculate average deceleration of the most frequent event codes of previous step (excluding outlier values)

Source: sciencedirect.com

Emergency Braking Analysis

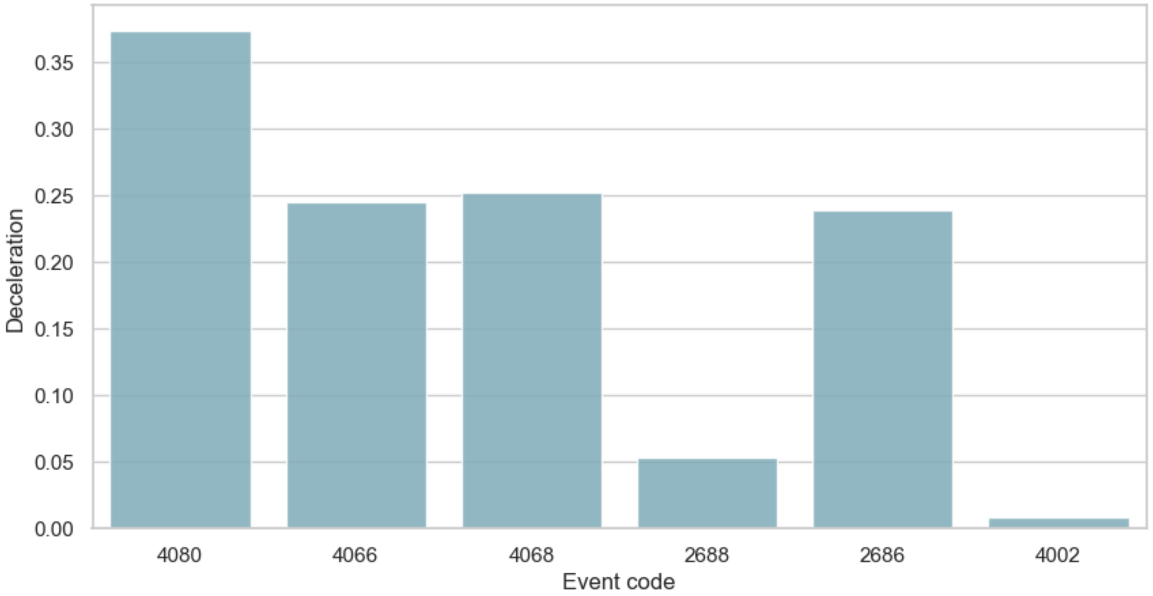
Analyzing the frequency and the average deceleration it was possible to find that the 4080 event may be related to an emergency braking

Comparison of frequency and average deceleration



- Event 4080 shows the higher frequency in decelerations events.
- Many times, there was a short event before the 4080 which led to deceleration being 0.

Average deceleration (no filtering)



- Event 4080 shows the highest deceleration with 0.38 m/s2.
- For codes 4066 and 4068 there were cases with acceleration (negative deceleration).

Source: sciencedirect.com

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- CUSTOM PIPELINE
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

Paper implementation

Machine-Learning Model implementation based on SNCB paper

Filter out events based on r metric (threshold 0.5)

$$r = \frac{h_{\text{in class}}}{h_{\text{in all classes}}}$$

↓

Reinclude filtered event codes with avg F1 ≥ 0.6 using CountVectorizer

↓

Identify the most frequent one- and two-event sequences using the FP-Growth algorithm with a minimum support threshold of 0.05

↓

Add binary columns for each frequent event sequence in each incident row

↓

Fine-tune the parameters of a Naive Bayes classifier using GridSearchCV with 5-fold stratified cross-validation

MAIN ISSUES DURING IMPLMEMENTATION

Apply Apriori algorithm for the whole data set

MemoryError: Unable to allocate **60.9 GiB** for an array with shape (12926995, 5, 1011).

The number of possible itemsets **grows exponentially.**

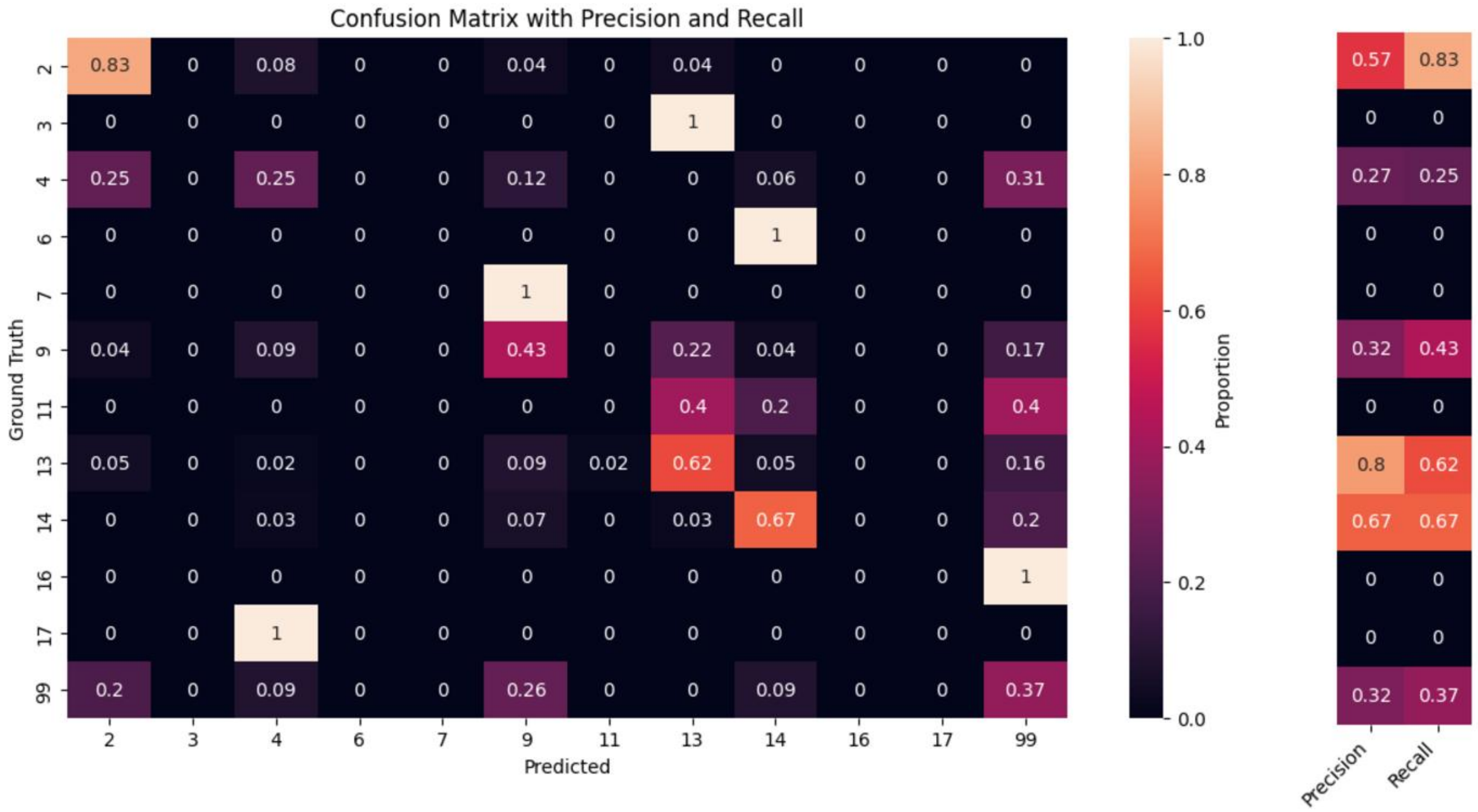
Same issues with Fpgrowth for sequences > 4 or with adding new Boolean attributes where sequences > 2.

Source: “Augmenting train maintenance technicians with automated incident diagnostic suggestions”

Paper implementation

Test Dataset Results

F1-score: **0.52**
Accuracy: **0.53**



POSSIBLE IMPROVEMENTS

Use longer sequences, not just the most frequent one- and two-event sequences.

For full testing, we would need **Virtual Machines with high memory** or **distributed processing** (e.g., Spark's FP-Growth implementation).

Source: "Augmenting train maintenance technicians with automated incident diagnostic suggestions"

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- CUSTOM PIPELINE
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

Text classification Methodology

Current tabular ML Techniques do not support sequential data and do not perform well on severely imbalanced and scarce data

Sequence Representation: Represent events in sequences as “words” in a sentence

Class Imbalance: Test oversampling techniques for improved performance

Brute-force approach: Test multiple embedding representations, oversampling strategies and classification models

Source: Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1), 67-82.

Text classification Methodology

Current tabular ML Techniques do not support sequential data and do not perform well on severely imbalanced and scarce data

Sequence Representation: Represent events in sequences as “words” in a sentence

Class Imbalance: Test oversampling techniques for improved performance

Brute-force approach: Test multiple embedding representations, oversampling strategies and classification models

Goal: Maximize accurate classifications and minimize mistakes (Maximize F1-Score)

Source: Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1), 67-82.

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210 different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210 different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

We will **train embedding space generators on the training fold** and **transform the test fold** using the learned representations. Below are the strengths and weaknesses of each generator:

METHOD	STRENGTHS	WEAKNESSES
TF-IDF	• **Mitigates noise** from frequent events across sequences. • **Highlights unique patterns** by emphasizing events frequent in one sequence but rare in others.	• **Ignores sequential and semantic relationships**. • Produces **high-dimensional, sparse vectors**, challenging computational efficiency and interpretability.
Count Vectorizer	• Provides a **straightforward representation of event frequency**. • Indicates the significance of **event frequency over order** in outcomes.	• **Disregards sequential and semantic information**. • Produces **sparse, high-dimensional vectors**, hindering computational and analytical efficiency.
Word2Vec	• **Captures semantic relationships** by grouping similar events. • **Encodes event context**, preserving order and meaning.	• **Requires substantial training data** for quality embeddings and **struggles with limited datasets**.

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210 different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Extra Processing:

We **duplicated training records for classes with fewer than three samples**, ensuring **oversamplers could function for all classes.**

Oversampling **at first** was **infeasible** since it requires **at least three** samples per class**.**
(5-fold split would leave folds with only two samples)

We applied the **default oversampling strategy** for each sampler **exclusively to the training data** to prevent information leakage into the test data.

This approach enhanced the training dataset while **maintaining the validity of the evaluation process.**

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210 different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

We adopted **a brute-force approach, inspired by the No Free Lunch Theorem**, to determine the best-performing model for this dataset.

We **tested 10 different models** available in the scikit-learn library.

For benchmarking, we **initially trained all models using their default parameters**.

Once the best-performing model is identified, we will **fine-tune its hyperparameters**.

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210 different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

Embedding Space Generators

- TFIDF Vectorizer
- Count Vectorizer
- Word2Vec

Over-samplers

- SMOTE
- Borderline-SMOTE
- ADASYN
- RandomOversampler
- SMOTE-ENN
- SMOTE-Tomek
- No Oversampler

Models

- LogisticRegression
- DecisionTree
- RandomForest
- ExtraTreesClassifier,
- GradientBoostingClassifier
- AdaBoostClassifier
- GaussianNB
- KNN
- SVM
- XGBoost

Ensambler

210

different models

The pipeline at a glance

This methodology could be divided in four main steps that implies trying more than 200 different models.

To construct the ensemble model, we **trained 210 individual models.**

We used the **differential evolution (DE) algorithm** to **determine the optimal weights** for combining their predictions.

DE is well-suited for optimizing ensemble models due to its **global search capability**, which **avoids local minima** and **effectively explores the solution space**.

The objective was to **maximize predictive performance**, specifically the **F1-score**.

The **fitness function**, defined as the F1-score of the ensemble model's predictions compared to true values, **ensured weight optimization aligned with the performance metric**.

Ensambler

210 different models

- TFIDF V
- Count V
- Word2V

Dataset with no extra processing results

Data Preprocessing

Load the data

→

Transform the sequences stored as strings into lists

→

Pipeline

Dataset with no extra processing results

Test Dataset Results

F1-score: **69.10%**
Accuracy: **69.92%**

Embedding Generator + Over Sampler + Model

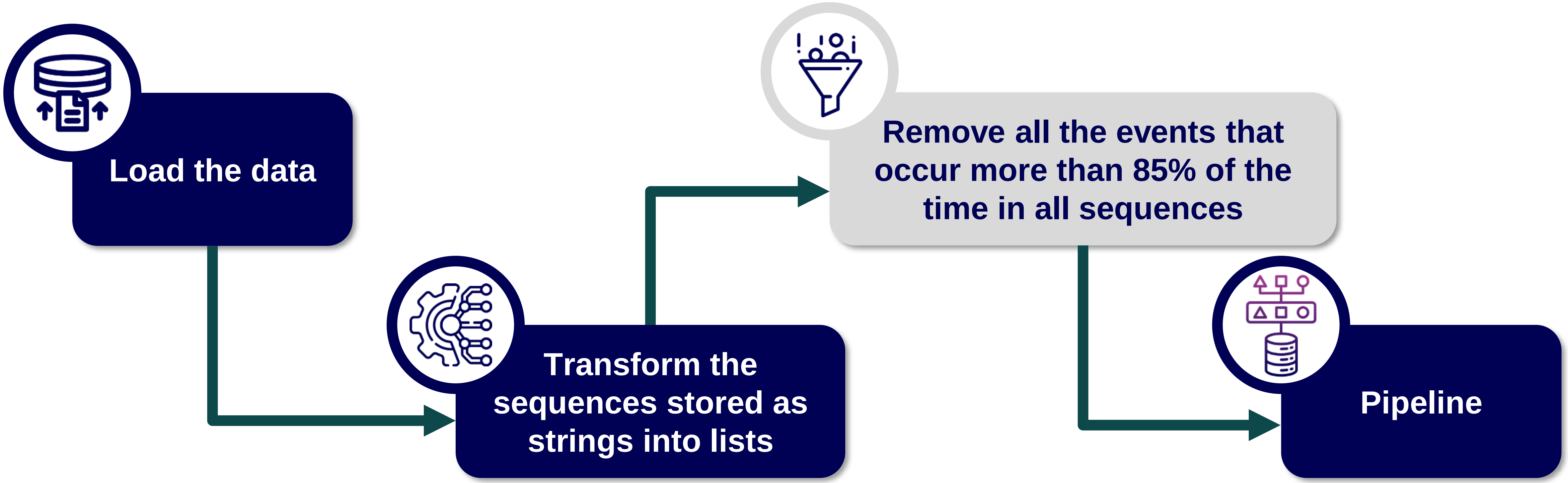
	Model	Vectorizer	Sampler	Accuracy Mean	Accuracy Std	Recall Mean	Recall Std	Precision Mean	Precision Std	F1 Mean	F1 Std
94	GradientBoostingClassifier	Count	ADASYN	0.699298	0.009880	0.699298	0.009880	0.693292	0.009541	0.691044	0.009947
104	GradientBoostingClassifier	Count	RandomOversampler	0.694333	0.019053	0.694333	0.019053	0.705052	0.017707	0.689394	0.018435
74	GradientBoostingClassifier	Count	SMOTE	0.693347	0.023312	0.693347	0.023312	0.698465	0.015648	0.687498	0.021960
84	GradientBoostingClassifier	Count	Borderline-SMOTE	0.688407	0.021736	0.688407	0.021736	0.684475	0.023663	0.680387	0.022421
54	GradientBoostingClassifier	TFIDF	SMOTE-Tomek	0.677515	0.032724	0.677515	0.032724	0.708091	0.018617	0.677958	0.028411
124	GradientBoostingClassifier	Count	SMOTE-Tomek	0.680500	0.015156	0.680500	0.015156	0.681979	0.012754	0.675060	0.014813
34	GradientBoostingClassifier	TFIDF	RandomOversampler	0.677486	0.034528	0.677486	0.034528	0.686205	0.026626	0.674007	0.031581
64	GradientBoostingClassifier	TFIDF	NoSamp	0.679496	0.027137	0.679496	0.027137	0.691487	0.023965	0.671905	0.024160
24	GradientBoostingClassifier	TFIDF	ADASYN	0.673555	0.022593	0.673555	0.022593	0.697297	0.016998	0.671654	0.016708
134	GradientBoostingClassifier	Count	NoSamp	0.680491	0.018784	0.680491	0.018784	0.689769	0.025222	0.669836	0.018454

Ensambler(Embedding Generator + Over Sampler + Model) = Ensemble Model

	Model	Vectorizer	Sampler	Accuracy Mean	Accuracy Std	Recall Mean	Recall Std	Precision Mean	Precision Std	F1 Mean	F1 Std
0	Ensemble	Multiple	Multiple	0.675574	0.019847	0.675574	0.019847	0.676288	0.019961	0.667454	0.018378

Dataset keeping the events that occur less than 85% of the time

Data Preprocessing



Dataset keeping the events that occur less than 85% of the time

Test Dataset Results

F1-score: **68.89%**
Accuracy: **69.23%**

Embedding Generator + Over Sampler + Model

	Model	Vectorizer	Sampler	Accuracy Mean	Accuracy Std	Recall Mean	Recall Std	Precision Mean	Precision Std	F1 Mean	F1 Std
104	GradientBoostingClassifier	Count	RandomOversampler	0.692362	0.015539	0.692362	0.015539	0.713046	0.015279	0.689194	0.014109
4	GradientBoostingClassifier	TFIDF	SMOTE	0.677564	0.026962	0.677564	0.026962	0.704565	0.021570	0.677456	0.025040
134	GradientBoostingClassifier	Count	NoSamp	0.685451	0.010444	0.685451	0.010444	0.701284	0.012506	0.675476	0.008441
74	GradientBoostingClassifier	Count	SMOTE	0.678501	0.023227	0.678501	0.023227	0.680099	0.023338	0.672320	0.023137
9	XGBoost	TFIDF	SMOTE	0.679564	0.029786	0.679564	0.029786	0.676154	0.037139	0.670813	0.030060
94	GradientBoostingClassifier	Count	ADASYN	0.678520	0.020992	0.678520	0.020992	0.674466	0.023335	0.670713	0.021415
59	XGBoost	TFIDF	SMOTE-Tomek	0.676564	0.022819	0.676564	0.022819	0.679385	0.027989	0.669826	0.022084
84	GradientBoostingClassifier	Count	Borderline-SMOTE	0.675545	0.015661	0.675545	0.015661	0.679817	0.011690	0.669716	0.014653
124	GradientBoostingClassifier	Count	SMOTE-Tomek	0.673599	0.011871	0.673599	0.011871	0.681708	0.008188	0.668974	0.013821
64	GradientBoostingClassifier	TFIDF	NoSamp	0.676569	0.028997	0.676569	0.028997	0.689269	0.031692	0.667489	0.029637

Ensambler(Embedding Generator + Over Sampler + Model) = Ensemble Model

	Model	Vectorizer	Sampler	Accuracy Mean	Accuracy Std	Recall Mean	Recall Std	Precision Mean	Precision Std	F1 Mean	F1 Std
0	Ensemble	Multiple	Multiple	0.684485	0.009068	0.684485	0.009068	0.671497	0.02027	0.669622	0.013434

Dataset keeping the events that occur less than 85% of the time

Test Dataset Results

KEY INSIGHTS

F1-score: 68.89%
Accuracy: 69.23%

Embedding Generator + Over Sampler + Model

	Model								[illegible] Std	F1 Mean	F1 Std
104	GradientBoostingClassifier	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.689194	0.014109
4	GradientBoostingClassifier	TFIDF	SMOTE	0.677564	0.026962	0.677564	0.026962	0.704565	0.021570	0.677456	0.025040
134	GradientBoostingClassifier	Count	NoSamp	0.685451	0.010444	0.685451	0.010444	0.701284	0.012506	0.675476	0.008441
74	GradientBoostingClassifier	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.023338	0.672320	0.023137
9	XGBoost	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.670813	0.030060
94	GradientBoostingClassifier	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.670713	0.021415
59	XGBoost	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.669826	0.022084
84	GradientBoostingClassifier	Count	Borderline-SMOTE	0.675545	0.015661	0.675545	0.015661	0.679817	0.011690	0.669716	0.014653
124	GradientBoostingClassifier	Count	SMOTE-Tomek	0.673599	0.011871	0.673599	0.011871	0.681708	0.008188	0.668974	0.013821
64	GradientBoostingClassifier	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0.667489	0.029637

The count vectorizer revealed that **classifying incidents relies solely on the frequency of events in sequences**.

It **does not consider relationships among events** in terms of semantics or sequential information.

This insight can **guide future analysis** to uncover the rules the model learned.

Ensambler(Em[illegible]

	Model	Vectorizer	Sampler	Accuracy Mean	Accuracy Std	Recall Mean	Recall Std	Precision Mean	Precision Std	F1 Mean	F1 Std
0	Ensemble	Multiple	Multiple	0.684485	0.009068	0.684485	0.009068	0.671497	0.02027	0.669622	0.013434

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- **CUSTOM PIPELINE**
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

Data Preparation

Manipulation of data to align with custom pipeline

Reordering chronologically all the events, speeds, states of an incident based on the relative timestamp

Store the event sequence column without commas and [] symbols

Add new columns for the number of vehicles and the index of the timestamp right before 0 sec.

The attributes events, speed, pantograph states, and vehicle sequences are **not sorted in absolute chronological** order based on the relative timestamp. Instead, they are sorted by vehicle, and within each vehicle, they are ordered chronologically.

Custom ML pipeline

Enhancing Model Performance: Incorporating Unexploited Attributes

EXISTING ATTRIBUTE	NEW ATTRIBUTE
Events sequence	The **last n events leading up to the incident** (timepoint 0 sec)
Train kph sequence	**Average speed during the last k events** prior to the incident
Dj ac state & dj dc state	**State of each pantograph in the last x events** before the incident
Events sequence	**Emergency braking occurrence in the last m events** prior to the incident
Events sequence	New event sequence that **excludes the most common event codes appearing** in at least a certain percentage of incidents
Vehicles sequence	Number of vehicles in incident

Custom ML pipeline

Custom Machine-Learning Pipeline with Feature Engineering and Hyperparameter Optimization

- **Binary** features of the **last n events before the incident**, where **n is a hyperparameter**, created using a MultiLabelBinarizer
- **Numeric** feature of the **average speed in the last k events before the incident**, with **k as a hyperparameter**
- **Boolean** features for the **two pantograph states in the last x events before the incident**, where **x is a hyperparameter**
- **Boolean** feature of **emergency braking occurrence in the last m events**, with **m as a hyperparameter**
- Event sequence **excluding the most common event codes** (occurring in at least a certain percentage **per** incidents)
- **Removal of unnecessary attributes** used in earlier steps
- **Event codes transformed into numerical features** using count vectorizer
- **Oversampling** technique of ADASYN
- Use of **Gradient Boosting classifier**

Custom ML pipeline

Optimized Parameters and Key Attributes after RandomSearchCV

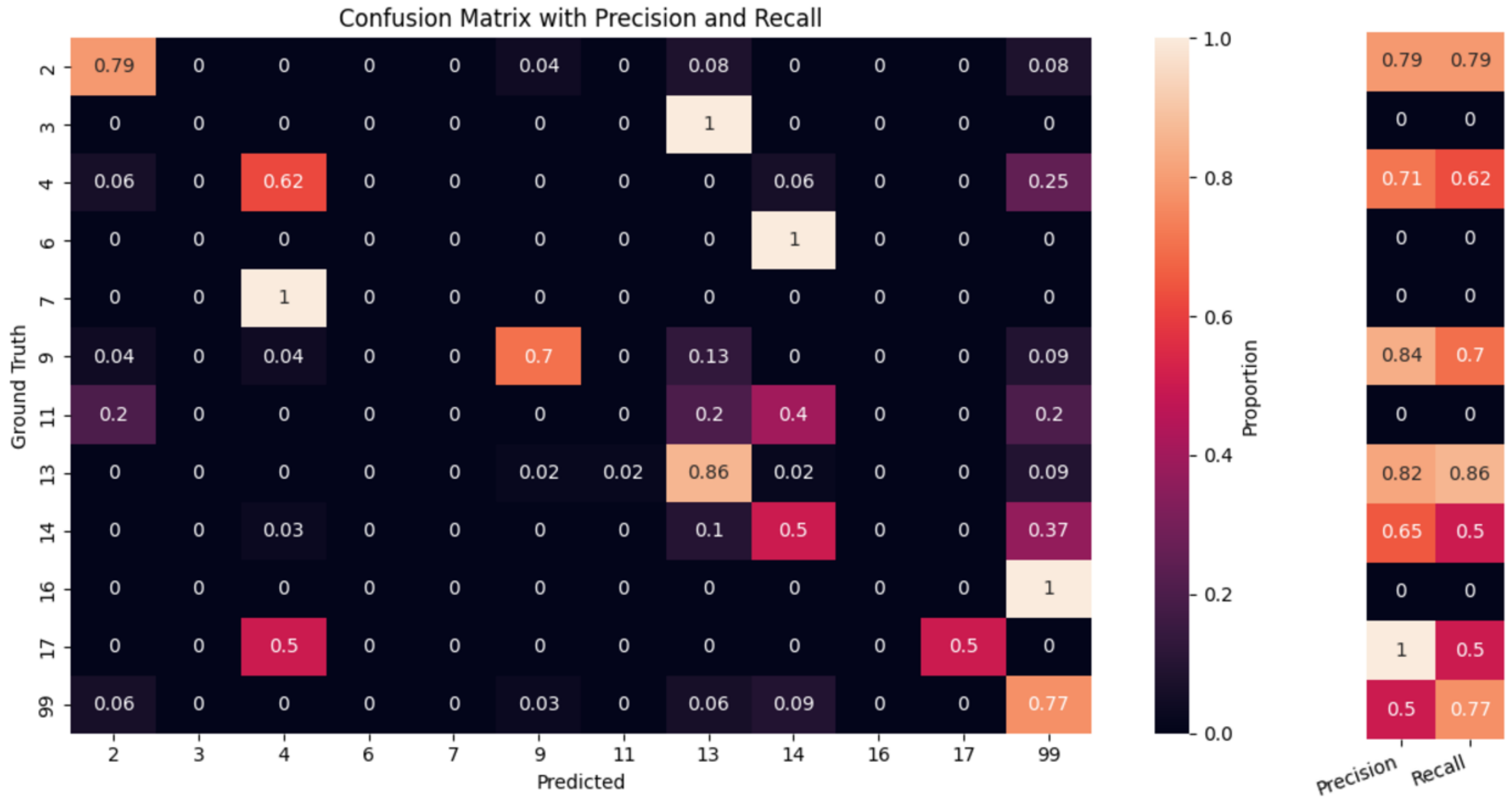
- **Fine-tuning parameters** using RandomSearchCV with 1500 iterations
- **Comparison** with cross-validation weighted F1 and F1 score of validation dataset

ATTRIBUTE	OPTIMAL PARAMETER
Last n events prior to incident	**n = 6**
Average speed in the last k events before the incident	**k = 12**
Boolean state of pantographs in the last x events	**x = 3**
Emergency braking occurrence in the last m events	**m = 15**
Event sequence **excluding the most common event codes**	**percentage = 80**
ADASYN Oversampling	Add **30% of the majority class count** to the current count of any non-majority class
Gradient Boosting classifier	**n_estimators = 400, max_depth = 5, learning_rate = 0.05, max_features = 'sqrt'**

Custom ML pipeline

Test Dataset Results – not included in randomsearchCV fine-tuning

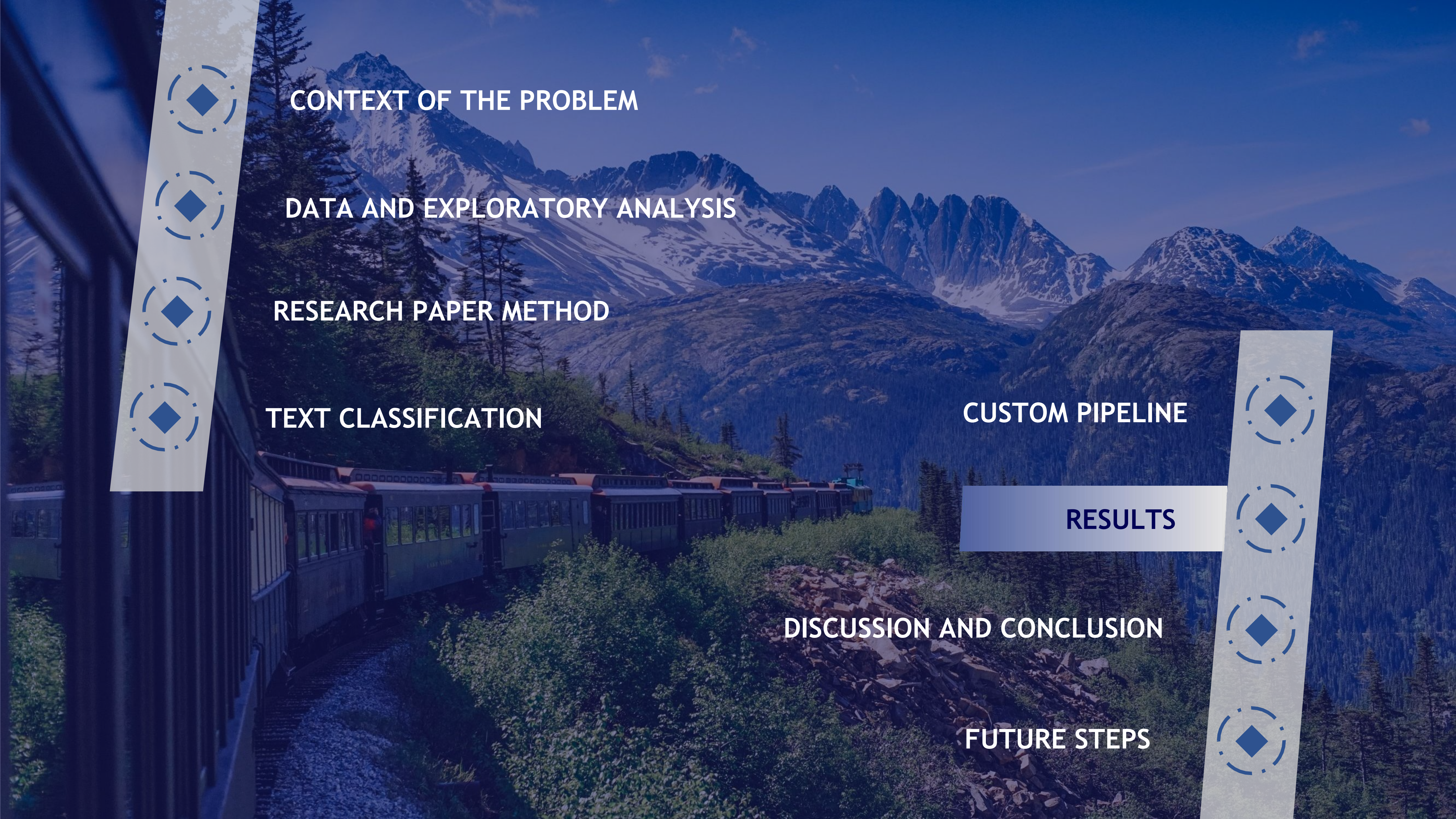
F1-score: **69.2%**
Accuracy: **70%**



INSIGHTS

The pipeline performs poorly in all minority classes except for *17*

Excluding the minority classes, the performance would reach
72.5% F1-score
74% accuracy


CONTEXT OF THE PROBLEM
DATA AND EXPLORATORY ANALYSIS
RESEARCH PAPER METHOD
TEXT CLASSIFICATION
CUSTOM PIPELINE
RESULTS
DISCUSSION AND CONCLUSION
FUTURE STEPS

F1-Score

The metric we choose to measure the performance of the models and makes the comparison was the F1-Score

Precision

- Measures how accurate a model is at identifying positive cases.
- It evaluates the model's overall ability to distinguish relevant objects from irrelevant ones

$$Precision = \frac{TP}{TP + FP}$$

Recall

- Ratio of true positives to all actual positives
- Focuses on how well a model captures all the positive cases

$$Recall = \frac{TP}{TP + FN}$$

F1-Score

- It incorporates the trade-off between precision and recall
- The value of the F1 Score lies between 0 and 1
- It is the harmonic mean of precision and recall

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Pros

- It can handle imbalanced data sets: It penalizes low precision or recall. It rewards when high values for both
- It can be used to compare different classifiers on the same data set

Cons

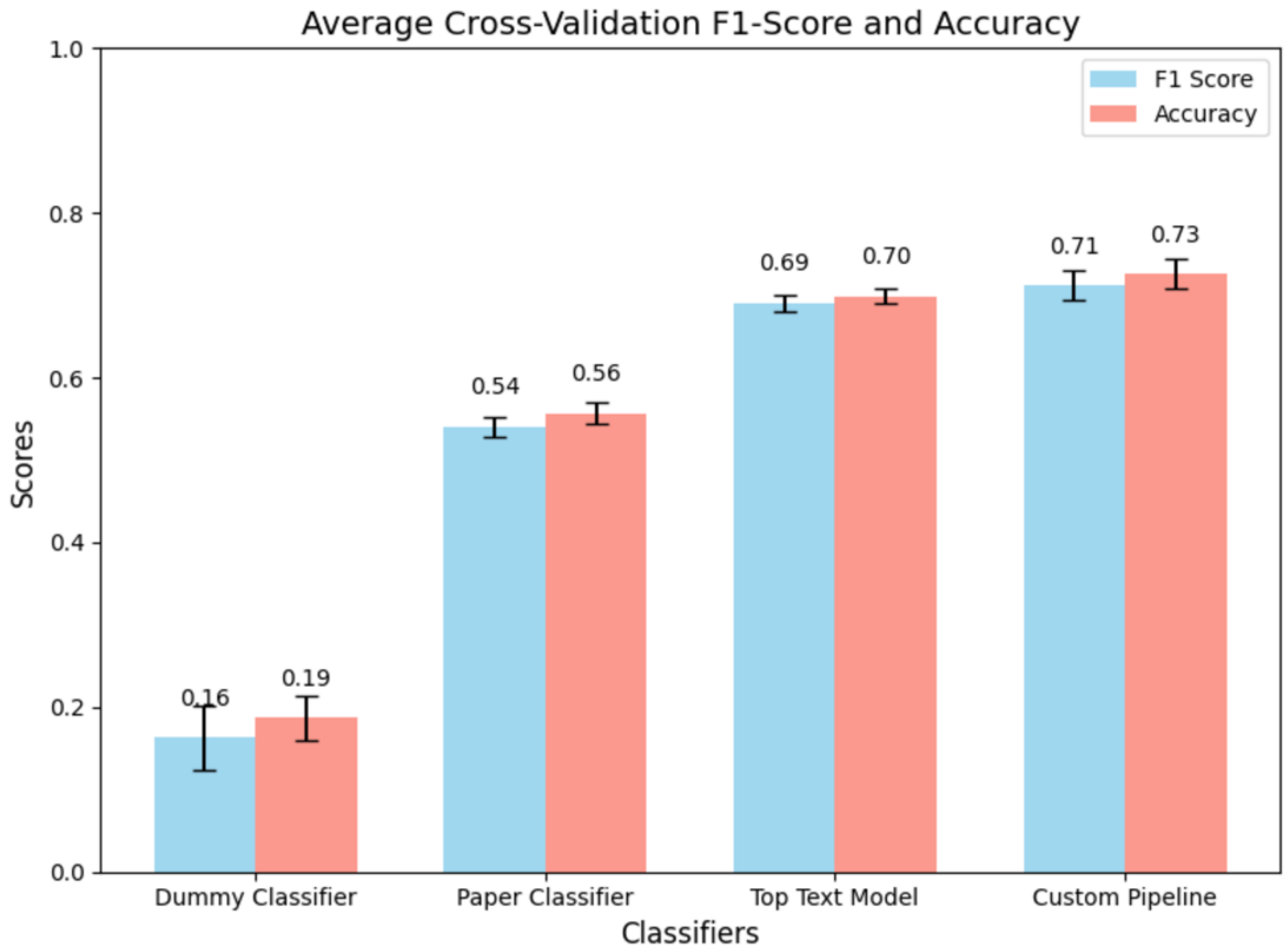
- It assumes that precision and recall are equally important
- It does not consider the distribution of errors or the confidence of predictions

We selected the F1-score because the nature of the problem requires and equilibrium in the predictions of TP and FN considering an imbalanced data set. Also, it is a fair metric to compare different models including the paper's results

Source: blog.roboflow.com

Results

Comparison of cross-validation F1 and Accuracy



INSIGHTS

All approaches outperform the baseline model of Dummy Classifier.

Custom pipeline slightly increases performance of best text classification approach.

CONTEXT OF THE PROBLEM

DATA AND EXPLORATORY ANALYSIS

RESEARCH PAPER METHOD

TEXT CLASSIFICATION

CUSTOM PIPELINE

RESULTS

DISCUSSION AND CONCLUSION

FUTURE STEPS

Conclusions - Discussion

The encryption of event and incident codes made it harder to understand the data and see the connections between different events which is crucial for problem solving.

The scarcity of minority class incidents made it nearly impossible to achieve reliable results for those classes.

Representing the events as sequences increased the complexity of the task, yet allows room for creative solutions.

The text classification approach proved efficient due to the limited vocabulary size (a maximum of 917 events) as well as its simplicity and effectiveness.

With more computational resources, better results could be achieved using algorithms like FP-growth or GNNs, especially with more data samples and balanced data.

- CONTEXT OF THE PROBLEM
- DATA AND EXPLORATORY ANALYSIS
- RESEARCH PAPER METHOD
- TEXT CLASSIFICATION
- CUSTOM PIPELINE
- RESULTS
- DISCUSSION AND CONCLUSION
- FUTURE STEPS

Predictive maintenance

Predictive maintenance system greatly helps in preventing damage and accidents using artificial intelligence, thereby saving companies time and money.

Predictive maintenance is a data-driven system that anticipates when a particular part of a train or railway is likely to fail.

Condition-based maintenance and predictive maintenance are related maintenance strategies.

Both use real-time monitoring system and analysis to improve effectiveness and reduce costs.

- CBM monitors equipment in real-time to identify potential issues based on current condition and performance.
- The main operator of the system is artificial intelligence, which can be used to predict errors on the railways.

Top 5 Benefits of Predictive Maintenance

1. Minimize the duration of system or service unavailability.
2. Take measures to avoid harm to both the locomotive and the railway infrastructure.
3. Enhance the ability to identify empty spaces or gaps.
4. Reduce the occurrence of failures in the point machines.
5. The combination of anticipating maintenance needs and ensuring protection against cyber threats.

Companies which had implemented this technology

Source: littlefish.co.uk; railacademy.deutschebahn.com; up.com; sncf.com

Predictive maintenance - Implementers

Many companies around the world deal with predictive maintenance because their goal is to provide the best solutions and services to companies operating in all industries.

EKE - Electronics

SmartVision™ Track Condition Monitoring measures ride smoothness of in-service trains, complementing track geometry measurements.

It detects issues like broken rails and wheel slip, enabling early and cost-effective repairs.

Largest partners

VR FLEETCARE

unife THE EUROPEAN RAIL INDUSTRY

Siemens

Predictive Maintenance is often used in factories and industries with automation systems like Siemens.

These systems produce significant data used to implement supervised machine learning algorithms.

Largest partners

AZIMA DLI Knowing What's Ahead

ARMSTRONG FLIGHT RESEARCH CENTER

Alstom

HealthHub is a predictive maintenance tool, which utilizes advanced data analytics to monitor the health of trains, infrastructure, and signaling assets.

HealthHub is supported by TrainScanner, a high-tech data capture solution that measures the condition of key train components.

Largest partners

SNCF

NetworkRail

Source: eke-electronics.com; alstom.com; iot-analytics.com

Predictive maintenance - Best Practices

Predictive maintenance is very popular in the railway industry, more and more companies are using the technology to save money and increase their traffic, like SBB Cargo and GE Transportation.

SBB Cargo

SBB Cargo plays a critical role in Switzerland's freight services.

To maintain its position as a trusted service provider, SBB Cargo has been using Railnova's remote monitoring solutions.

This allows the company to gather telematic data from its trains and prevent potential failures.

This approach helps to eliminate the inconvenience caused by small component failures.

METRÔ

The São Paulo Metro has created an AI-based predictive maintenance system, called the Asset Monitoring System (AMS).

It can forecast potential malfunctions in various systems, including escalators, lifts, trains, tunnel ventilation, and power supply systems.

The AMS provides data to expedite decision-making.

GE Transportation

GE Research's predictive maintenance technologies have been incorporated into the Expert on AlertTM solution for GE Transportation.

AlertTM solution provides instant health checks and diagnostic reports for essential locomotive components.

Allowing for proactive planning of resources and parts.

Source: ge.com; railjournal.com; railnova.eu

WHEN MACHINES TALK

a data challenge by SNCB-NMBS

check out our GitHub repository

TEAM MEMBERS

Jorge Ignacio del Río Sánchez: 000607231
Kristóf Balázs: 000612294
Neri Pérez Sebastian Alberto: 000605855
Stefanos Kypritidis : 000606810