Mining information from social media during crisis events

Raphaël Pinault [21801579], Camille Pousse [22200255] et Cindy Puech [21804198]

Abstract

This project explores the application of machine learning techniques to extract insights from social media data during crisis events, focusing on real-time information from platforms like Twitter. Through detailed analysis and modeling, we address challenges such as data imbalance and feature selection. Despite notable achievements, the study highlights the need for further validation in diverse crisis contexts and emphasizes the potential for enhanced performance with additional optimization. This project also emphasizes the utilization of text mining techniques like TF-IDF, embedding, and the exploitation of rich information inherent in graph networks.

Contents

1 Introduction **3**

2 Data presentation **3**

3 Definition and implementation of features **4**

3.1 About the tweet . . . 4

3.1.1 TF IDF . . . 4

3.1.2 Word embedding . . . 5

3.2 About the user . . . 7

3.3 About the event . . . 7

3.4 About the hashtag . . . 7

4 Data preparation for the training of the model **8**

4.1 Formating the response variable . . . 8

4.2 Spliting the data . . . 8

5 Predictive model used and the scores **9**

5.1 MultiOutputClassifier . . . 9

5.2 Metrics . . . 9

5.3 The models . . . 10

5.3.1 Random Forest . . . 10

5.3.2 Histogram gradient boosting classifier (HGBC) . . . 10

5.3.3 Ridge classifier . . . 10

5.3.4 First results . . . 10

6 Model improvement **11**

6.1 Data imbalance . . . 11

6.2 Feature selection . . . 11

6.3 Hyperparameter optimisation and results . . . 12

7 Conclusion and limits of our project **13**

8 Annexe **15**

1 Introduction

In recent years, the proliferation of social media platforms has transformed the landscape of information dissemination, particularly during crisis events. With the advent of platforms like Twitter, real-time updates from affected individuals have become readily accessible, providing invaluable insights for emergency response teams. In this project, we delve into the real of crisis informatics, leveraging machine learning techniques to harness the wealth of information embedded within Twitter posts during various crisis events.

The dataset under study comprises real-world Twitter posts collected during a spectrum of crisis events, ranging from floods and fires to tornadoes and earthquakes. This rich dataset has been provided by the organizers of the NIST TREC Incident Stream Initiative [odlNTIS20], an endeavor aimed at catalyzing research in automatically processing social media data during emergencies. At its core, the primary objective of this initiative is to facilitate the categorization of information and help requests from citizens, thereby aiding emergency operators in swiftly and efficiently coordinating response efforts.

We present the comprehensive workflow, encompassing all stages from data ingestion and preprocessing to label predictions. By offering a holistic view of the entire process, we aim to provide insights into the efficacy and feasibility of employing such methodologies in real-world scenarios.

2 Data presentation

The dataset encompasses real-world Twitter posts amassed during a diverse array of crisis events, including floods, fires, tornadoes, and earthquakes. This valuable dataset has been made available by the organizers of the NIST TREC Incident Stream Initiative [odlNTIS20], with the objective of fostering research in the automated processing of social media data during emergency situations. Central to this initiative is the imperative to classify information and aid requests from citizens, thereby facilitating response coordination by emergency operators.

Presented as a directed multi-graph structure, the dataset comprises a total of 109,627 nodes, delineated as follows:

- 43,141 User Nodes: Representing individual users
- 55,986 Tweet Nodes: Capturing individual Twitter posts during crisis events
- 34 Event Nodes: Signifying specific crisis events
- 10,441 Hashtag Nodes: Denoting hashtags associated with tweets
- 25 Category Nodes: the categories used for classifying information and aid requests

Of particular significance is the presence of edges linking these nodes. Notably, the dataset features a diverse array of edge types, including:

- has_hashtag: Establishing connections between tweets and associated hashtags,
- has_category: Linking tweets to one of the 25 predefined categories,
- reply_to: Facilitating interactions between tweets and users.

Despite the richness of the dataset, not all information will be utilized in the scope of this project. Nonetheless, we will provide detailed insights into the components that we think are pertinent for our analysis and model.

3 Definition and implementation of features

Initially, we choose the variables that appear relevant for analysis with the aim of predicting the category of a tweet. In this section we will explore 4 categories of features: about the tweet, more precisely about the content of the tweet, about the user, about the Event and finally about the hashtag.

3.1 About the tweet

Firstly, regarding the tweet, we retain the full text of the tweet, which we will subsequently analyze, along with the tweet's topic and its number of retweets to gauge its popularity.

Regarding the full text of the tweet, we start by cleaning the text, separating the raw text from the hashtags that beggins with a "#" symbol, links such as those starting with "https," and mentions that appear with an "@" symbol. Next, we identify the important words through a series of operations describ as following :

- initially, we tokenize the tweet, meaning we split the sentence into words,
- we convert everything to lowercase
- we remove punctuation that doesn't provide information
- we eliminate non-alphabetic terms
- we discard words with fewer than 3 letters
- we lemmatize, meaning we retain only the root form of words; and finally, we remove stop words, which are commonly used words in the language that carry little or no information.

	Node	Text_original	topic	Hashtags	HTTPLinks	Mentionner	Text_Clean
0	582	#colorado. Told you its #amazing http://t.co/6...	TRECIS-CTIT-H-001	#colorado;#amazing	http://t.co/6z0Qq7qe		colorado. Told you its amazing
1	583	RT @northfortynews: Tanker helicopter heads up...	TRECIS-CTIT-H-001	#HighParkFire	http://t.co/7atRS5cy	@northfortynews	RT : Tanker helicopter heads up to Paradise Pa...
2	584	#Evacuation center Cache La Poudre Middle Scho...	TRECIS-CTIT-H-001	#Evacuation;#Colorado;#Wildfire			Evacuation center Cache La Poudre Middle Schoo...
3	585	20F degrees cooler tomorrow in North Central &...	TRECIS-CTIT-H-001	#Colorado;#HighParkFire;#COwx;#Heat;#Coolbreak			20F degrees cooler tomorrow in North Central &...
4	586	FEMA has authorized the use of federal funds t...	TRECIS-CTIT-H-001	#HighParkFire	http://t.co/whxlpPEP		FEMA has authorized the use of federal funds t...
5	587	#Media Large wildfire in N. Colorado prompts e...	TRECIS-CTIT-H-001	#Media;#Politics;#News	http://t.co/ju1BGTKH		Media Large wildfire in N. Colorado prompts ev...
6	588	Large wildfire in northern Colorado prompts ev...	TRECIS-CTIT-H-001		http://t.co/BvG6xSRH		Large wildfire in northern Colorado prompts ev...
7	589	RT @LarimerSheriff: #HighParkFire update http:...	TRECIS-CTIT-H-001	#HighParkFire	http://t.co/hBy5shen	@LarimerSheriff	RT : HighParkFire update
8	590	India Large wildfire in N. Colorado prompts ev...	TRECIS-CTIT-H-001		http://t.co/uprHT63g		India Large wildfire in N. Colorado prompts ev...
9	591	Western wildfires forcing evacuations: Firefig...	TRECIS-CTIT-H-001		http://t.co/BOaVjrRO		Western wildfires forcing evacuations: Firefig...

Figure 1: Dataframe with a view on the original and clean tweets

From these more simplified tweets, we then apply two distinct methods of text mining : TF-IDF and word embedding, which we will present thereafter.

3.1.1 TF IDF

Firstly, we apply a TF IDF vectorizer on the important words we have selected using the TfidfVectorizer function presented below.

```
vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, max_features=200, stop_words='
    english')
```

The max_df option filters out words that appear in more than 50% of documents to select only relevant words. The min_df option, set to 2, filters out words appearing in fewer than 2 documents, while max_features = 200 specifies the desired number of selected words. Here we retained the 200 most important words from the whole tweet network. Lastly, stop_words = 'english' filters out English stop words.

	abfire	affected	africa	ago	aid	alberta	alert	amp	area	attack	...	weather	week	wildfire	wind	work	world	year	ymm	ymmfire	yycflood
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.479692	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.371099	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 2: View on the TF IDF of the 200 selected words

3.1.2 Word embedding

Furthermore, as a subsequent step, we establish a word embedding method, which involves representing the tweet as a vector. For this purpose, we utilize the Word2Vec function, which enables us to obtain a numerical vector for each word present just below.

```
word2vec_model = Word2Vec(sentences=tokenized_text, vector_size=100, window=5,
    min_count=1, workers=16)
```

We specify several options, where we limit the size of the vector to 150. The "window" parameter indicates that for each word, the model considers words within a window of 5 words on each side. Finally, the "min_count" parameter set to 1 indicates that we are interested in words that appear at least once in the tweet, meaning all words in the tweet. This function allows us to obtain a vector of dimension 150 for each word present in the tweet. The tweet is nothing more than the sum of the vectorial representation of the word composing the tweet. To leverage this very rich information, we propose the following workflow :

- For each of the 25 category, we compute the vectorial representation of this category by computing the average vector of all the tweets belonging to this category.
- We then compute for each tweet the cosine similary with these category representant vector.
- Finally we obtain 25 new variables.

We compute the centrality score for each tweet node as well. We use 3 different centrality measures : closeness centrality, degree centrality and eigenvector centrality.

Node	0	1	2	3	4	5	6	7	8	9	...	140	141	142	143
582	0.544621	0.036198	-1.148637	-0.166836	0.431369	-0.431074	-0.320243	0.072626	0.385461	-0.388199	...	0.096144	0.242564	0.564463	0.163971
583	0.232817	0.357720	-1.068029	-0.145546	0.095105	-0.467353	-0.135425	-0.081899	0.034090	-0.031806	...	0.088263	0.134747	0.604798	0.516315
584	0.108889	0.397931	-1.373557	-0.017393	0.443378	-0.360247	-0.140677	-0.130147	0.277302	-0.351053	...	0.297206	0.016786	0.284720	0.400287
585	0.196755	0.247272	-0.829006	0.090893	0.379409	-0.317811	-0.035636	0.006532	0.132287	-0.104672	...	0.154519	-0.011499	0.446425	0.477192
586	0.354599	0.328523	-0.571975	0.036006	-0.054318	-0.261417	-0.144169	-0.125810	-0.078325	-0.173220	...	0.100808	0.195887	0.261231	-0.070670
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49023	0.557445	0.297902	-0.762604	0.100374	1.010133	-0.310543	-0.442874	0.370698	0.159584	-0.684839	...	0.217661	-0.677957	0.811629	-0.149449
49024	0.549562	0.198930	-0.403479	0.370339	0.318818	0.208498	-0.519078	0.766130	0.569787	-0.543364	...	-0.179537	-0.629642	0.409304	-0.281176
49025	0.793260	0.431944	-0.584306	0.388745	0.592595	-0.031930	-0.481955	1.110904	0.558426	-0.703180	...	-0.180554	-0.920219	0.860963	-0.496005
49026	0.622282	0.378066	-0.815334	0.346435	0.830485	-0.285820	-0.533546	0.652171	0.132862	-0.560795	...	0.343880	-0.763294	0.599812	-0.016201
49027	0.719376	0.403435	-0.520568	0.637285	0.787084	0.116462	-0.504227	1.089063	0.812971	-0.863553	...	-0.188784	-0.935545	0.339700	-0.503896

Figure 3: View on the vectorial representation of the tweets

	sim_562	sim_573	sim_564	sim_572	sim_570	sim_578	sim_558	sim_579	sim_566	sim_569	...	sim_563	sim_581	sim_576	sim_565	sim_560
0	0.768522	0.680289	0.695466	0.573142	0.770826	0.795718	0.777337	0.765728	0.749593	0.572826	...	0.759255	0.747735	0.692164	0.771831	0.719033
1	0.624730	0.720794	0.780027	0.643423	0.735902	0.829799	0.876742	0.681712	0.734869	0.649901	...	0.696899	0.784979	0.806169	0.768349	0.747720
2	0.602406	0.621127	0.656107	0.623323	0.751208	0.834251	0.866858	0.712779	0.734487	0.490209	...	0.735798	0.765436	0.725826	0.703400	0.641754
3	0.727019	0.783672	0.768165	0.781692	0.848363	0.899586	0.910470	0.808128	0.833571	0.631606	...	0.824145	0.885320	0.850291	0.837608	0.767749
4	0.720862	0.770637	0.900095	0.573174	0.689298	0.722792	0.714380	0.638227	0.665477	0.913671	...	0.672931	0.682472	0.778593	0.763324	0.836244

Figure 4: View on the similarity score between the tweets and the categories

	Node	degree_centrality	closeness_centrality	eigenvector_centrality
0	582	0.000046	0.000009	1.276952e-18
1	583	0.000055	0.000009	-2.052775e-18
2	584	0.000055	0.000009	-2.784416e-18
3	585	0.000073	0.000009	2.817689e-18
4	586	0.000036	0.000009	3.062136e-18

Figure 5: View on the centrality measures for each node

3.2 About the user

We decided to keep the attributes 'follower_count' and 'isVerified' of the User nodes, as it allows us to determine whether the tweeting user is a more influential person or not. For instance, the number of followers can indicate the reach and influence of a user's tweets, which may be critical for the filter of information during a crisis. Additionally, verified accounts might be considered more credible sources of information, which could impact how their tweets are perceived and utilized in emergency response efforts.

3.3 About the event

Regarding the event discussed in a tweet, we kept the event name (attribute 'id') as well as the event type ('eventType'). Indeed, we hypothesize that the 'eventType' variable provides context regarding the nature of the event, enabling better categorization and analysis of tweets based on the type of event.

fireColorado2012	costaRicaEarthquake2012	floodColorado2013
typhoonPablo2012	laAirportShooting2013	westTexasExplosion2013
guatemalaEarthquake2012	italyEarthquakes2012	philipinnesFloods2012
albertaFloods2013	australiaBushfire2013	bostonBombings2013
manilaFloods2013	queenslandFloods2013	typhoonYolanda2013
joplinTornado2011	chileEarthquake2014	typhoonHagupit2014
nepalEarthquake2015	flSchoolShooting2018	parisAttacks2015
floodChoco2019	earthquakeCalifornia2014	earthquakeBohol2013
hurricaneFlorence2018	shootingDallas2017	fireYMM2016
albertaWildfires2019	cycloneKenneth2019	southAfricaFloods2019
philippinesEarthquake2019	coloradoStemShooting2019	sandiegoSynagogueShooting2019

Table 1: Exemple de EventType

3.4 About the hashtag

The hashtags undergo a comparable treatment to the tweets: following formatting into word lists, we calculate cosine similarity with each category's representative vector, mirroring the process used for tweets. Separately computing these similarities adds depth to our analysis, potentially yielding richer insights. This step contributes an additional 25 features to our dataset, further enhancing the scope of information available for analysis.

Node	n562_#	n573_#	n564_#	n572_#	n570_#	n578_#	n558_#	n579_#	n566_#	n569_#	...	n563_#	n581_#	n576_#	n565_#	n560_#
582	0.680710	0.588461	0.620153	0.518834	0.700250	0.734135	0.704369	0.697022	0.672211	0.501962	...	0.682449	0.684364	0.621269	0.686541	0.623772
583	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
584	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
585	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
586	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
56563	0.510046	0.660271	0.656550	0.437678	0.532519	0.639851	0.807521	0.426408	0.569950	0.546015	...	0.497984	0.609836	0.634874	0.713546	0.712914
56564	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
56565	0.436765	0.542142	0.516865	0.337095	0.424298	0.489752	0.665040	0.335043	0.460550	0.428223	...	0.402602	0.488221	0.478170	0.604666	0.599219
56566	0.493031	0.645252	0.623501	0.433497	0.515908	0.610629	0.787018	0.409461	0.555930	0.513669	...	0.485731	0.592470	0.608826	0.697449	0.692909
56567	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 6: View on the similarity score between the hashtags in a tweet and the categories

4 Data preparation for the training of the model

4.1 Formating the response variable

A single tweet may belong to multiple classes, implying that the response variable contains as many dimensions as there are categories in the project? 25 in this instance. This response variable essentially forms a vector, and precise coding is crucial to ensure compatibility with the multiclassprediction function. It should be noted that it's represented as a 2D array object.

category	n557	n558	n559	n560	n561	n562	n563	n564	n565	n566	...	n572	n573	n574	n575	n576	n577	n578	n579	n580	n581
Node																					
582	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
583	0	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
584	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
585	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	0	0	0	0	0
586	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49023	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
49024	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	1	1	0	0
49025	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	1	0	0
49026	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
49027	1	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	1	0	1	0	0

Figure 7: View on formating of the tweets response variable before turning it into a 2D array object

4.2 Spliting the data

We have opted to partition our dataset into a 30% test set and a 70% training set. This decision aims to mitigate overfitting, a phenomenon where a machine learning model memorizes the training data rather than learning the underlying patterns and relationships. Overfitting leads to high performance on the training data but poor performance on new, unseen data. By splitting the dataset into training and test sets, we can assess the model's performance on unseen data, thus preventing overfitting. A successful performance on the test set indicates that the model has learned to generalize and can make accurate predictions on new data. Conversely, if the model performs poorly on the test set compared to the training set, it suggests overfitting, necessitating adjustments such as regularization techniques or reducing model complexity to enhance generalization.

Moreover, we took meticulous measures to prevent any potential data leakage between the training and testing datasets. Given that the response variable of tweets indicates all possible categories they may belong to, it's crucial to avoid overlap between the two datasets. If a tweet from the test dataset were also present in the training dataset, it could inadvertently grant the model access to its response variable during training, undermining the integrity of the evaluation process. Considering that some tweets can be associated with up to 6 or 7 categories, alongside the duplication of certain tweets to address the imbalance issue mentioned earlier, dividing the datasets based on row numbers (index in dataframe) would greatly increase the likelihood of overlap.

To address this concern, during the splitting phase, we use the node ID assigned to each tweet rather than simply dividing based on index numbers. This ensured that no tweet appeared in both the training and test databases, maintaining the integrity and independence of the datasets for robust model evaluation.

5 Predictive model used and the scores

We will employ three distinct machine learning algorithms for making predictions. Although delving into their principles is beyond the scope of this project, we will briefly outline their methodologies. To assess the performance of each model and compare them, we will compute performance metrics presented just below.

5.1 MultiOutputClassifier

We utilize the MultiOutputClassifier function to fit a classifier for each target. This serves as a straightforward approach to expand classifiers that lack native support for multi-target classification. The desired classifier is specified as a parameter of the MultiOutputClassifier function. When combined with the classification algorithm, this function effectively functions as a multi-class classifier algorithm.

5.2 Metrics

To evaluate our models, we use two metrics: the F1 score and Cohen's Kappa measure.

The **F1 score** is a measure of a classification model's precision. It takes into account both precision (the classifier's ability to not mislabel a negative observation as positive) and recall (the classifier's ability to find all positive observations).It is derived as follows:

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Where precision is the ratio of true positives to all observations predicted as positive and recall is the ratio of true positives to all actual positive observations. The closer the F1 score is to 1, the better the model's performance in terms of precision and recall.

Cohen's Kappa score measures agreement between two raters. In the context of classification, it assesses agreement between the labels predicted by the model and the actual labels, while accounting for the possibility of agreement by chance. It is derived as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Where:p_0 is the observed proportion of agreement between the two raters and p_e is the expected proportion of agreement if the raters were to randomly assign labels. In this formula, p_0 is computed as the ratio of the number of agreements to the total number of observations, and p_e is calculated as the product of the proportions of items labeled by each rater. A score of 1 indicates perfect agreement,

0 indicates agreement by chance, and -1 indicates complete disagreement. This measure is particularly interesting in this context of multi-label classification.

5.3 The models

5.3.1 Random Forest

Random Forest is an ensemble learning technique employed for classification and regression purposes. It builds numerous decision trees during training and outputs the class (for classification) predicted by each tree. Each tree is trained on a subset of the training data and makes independent predictions. By averaging or voting these predictions, Random Forest mitigates overfitting and enhances generalization compared to individual trees. Its robustness and capability to handle high-dimensional datasets make it particularly compelling for various applications.

5.3.2 Histogram gradient boosting classifier (HGBC)

HistGradientBoostingClassifier is a machine learning algorithm that can be used both for classification and regression tasks. It belongs to the gradient boosting family and utilizes a histogram-based technique for optimizing the model's performance. HistGradientBoostingClassifier handles large datasets by binning features into histograms, reducing computational overhead. The algorithm discretizes the continuous features into bins or intervals. Each feature's range is divided into bins, and the algorithm counts the occurrences of feature values within each bin. These counts are then used to construct histograms for each feature. The predictive model is built in a sequential manner by iteratively minimizing the loss function.

5.3.3 Ridge classifier

The Ridge Classifier incorporates L2 regularization to address multicollinearity (when independent variables are highly correlated) and overfitting in linear regression-based classification tasks. The L2 regularization adds a penalty term to the regression equation's cost function, which is proportional to the square of the coefficients' magnitude. By introducing this penalty term, the Ridge Classifier effectively controls the complexity of the model by favoring simpler models with smaller coefficient values. This preference results in improved generalization performance on unseen data. This regularization technique effectively shrinks the coefficients towards zero, reducing the model's sensitivity to noise and multicollinearity while enhancing its ability to generalize well to new data.

5.3.4 First results

These first results are obtained from the original test dataset.

Model	**Precision**	**Recall**	**F1 Score**	**Cohen Kappa Score**
Random Forest	0.775	0.460	0.533	0.274
HGBC	0.748	0.560	0.615	0.348
Ridge	0.684	0.424	0.475	0.209

Table 2: Performance Metrics of Different Models

Based on these initial findings, it appears that the HistGradientBoostingClassifier (HGBC) performs the best, exhibiting the highest Cohen's Kappa and F1 scores among the three models tested.

6 Model improvement

In this section, we will elucidate the methods employed to enhance the performance of our models, and then the updated results following the implementation of these methodologies.

6.1 Data imbalance

Among the 55,986 tweets in the database, we observe 36,611 tweets associated with at least one of the 25 categories. This reveals a significant class imbalance within the database: the most prevalent class comprises over 10,000 tweets, whereas the smallest class contains nearly 200 tweets. Such a skewed distribution implies that our model will primarily learn from the more dominant classes, potentially overlooking the underrepresented ones. In fact, data imbalance can significantly affect performance metrics such as the F1 score and Cohen's Kappa measure. When classes are unevenly distributed, these metrics may be biased towards the majority class, resulting in inflated scores for dominant classes and lower scores for minority classes.

To address this imbalance, we opt to resample the database, a popular practiced in this field. Specifically, we will upsample the smaller classes by duplicating their instances. Ultimately, we construct a new training database where each class represents at least 3% of the dataset. We recognize that this distribution does not accurately reflect real-world proportions. However, our main objective is to construct an optimal training dataset for the model, with the aim of enhancing our F1 and Kappa scores.

6.2 Feature selection

The classifier method allows us to see which features have been used the most on average. Since we have a high number of features, we will retain only the ones that have been used a lot, thus indicating they have strong discriminant power to distinguish between classes. The Random Forest Classifier method allows us to have a look at the importance of each feature, so we select the most important variables to be used using the 'feature_importances_' attribute of the model which cannot be utilized for the other two models.

For each of the 25 categories, a classifier model is fitted. For each of these 25 classifier instances, we can retrieve the importance score of each variable. The overall feature importance is obtained by computing the mean of these importance scores across the 25 categories.

```
Number_of_var = 100

feat_impts = []
for clf in randomforest_model1.estimators_:
    feat_impts.append(clf.feature_importances_)

feature_importance = np.mean(feat_impts, axis=0)
x = np.argsort(feature_importance)[::-1][:Number_of_var]

features = X_train.columns
selected_features = list(features[x])
```

We decided to retain the 100 most important variable identified by the Random Forest Classifier, listed in descending order of importance in the appendix.

Number of Variables	Precision	Recall	F1 Score	Cohen Kappa Score
50	0.4771	0.2496	0.3241	0.1036
100	0.7607	0.4792	0.5600	0.2988
150	0.4899	0.2521	0.3286	0.1086
257	0.6897	0.4942	0.5366	0.2568

Table 3: Performance Metrics for Different Numbers of Variables

6.3 Hyperparameter optimisation and results

In this section, we focused on the model that exhibited the best performance, namely the HistGradientBoostingClassifier. Our goal was to enhance its performance by tuning its hyperparameters. Initially, we considered using GridSearchCV to explore different hyperparameter combinations and identify the optimal configuration. However, we quickly realized that this approach was too costly in terms of computation time. The GridSearchCV process never concluded due to time constraints (sometimes after 10 hours without any result), prompting us to switch to an alternative approach with RandomizedSearchCV, limiting the number of iterations to make the process more efficient.

Unfortunately, despite our efforts, the results obtained with this new model did not surpass those of the baseline model. Several factors may explain this limitation. Firstly, the random nature of RandomizedSearchCV may not always effectively explore the hyperparameter space. Additionally, the constraint on the number of iterations may limit the ability to find optimal combinations. Moreover, there is an inherent risk of overfitting when evaluating optimal parameters, which can lead to poor performance on test data. Lastly, the complex interactions between hyperparameters can make selecting an optimal combination challenging.

In conclusion, although hyperparameter tuning is essential for optimizing model performance, our attempt with RandomizedSearchCV did not lead to significant improvements compared to the baseline model. Below, the results for the hyperparameters found :

Model	Hyperparameters
HGBC	'estimator__max_depth': 20
	'estimator__max_leaf_nodes' : 100
	'estimator__max_iter' : 500
	'estimator__learning_rate' : 0.09
	'estimator__l2_regularization' : 3.755

Table 4: Hyperparameters of HXGBoost model

Below, the results obtained after enhancing the model. It is observed that overall, the metrics of the second results are close to those of the first; however, they exhibit lower performance.

Model	Precision	Recall	F1 Score	Cohen Kappa Score
HGBC	0.725	0.556	0.608	0.349

Table 5: Performance metrics of updated HXGBoost model

7 Conclusion and limits of our project

In this project, we aimed to explore the potential of machine learning techniques in extracting valuable insights from social media data during crisis events. Our endeavor demonstrated a detailed application of effective methods for categorizing and analyzing this data to support emergency response efforts, leveraging real-time information available on online social platforms like Twitter during emergencies. Additionally, the project shed light on the rich content of text data through the utilization of text mining techniques such as TF-IDF, as well as the wealth of information derived from network graphs such as those found on Twitter.

We traversed through various stages of data processing, feature extraction, model training, and evaluation. Beginning with the presentation of the dataset provided by the NIST TREC Incident Stream Initiative, which comprised a diverse array of Twitter posts associated with different crisis events, our analysis encompassed not only the textual content of tweets but also user attributes, event details, and hashtag usage, providing a holistic view of the information landscape during crises.

For predictive modeling, we experimented with three machine learning algorithms: Random Forest, Histogram Gradient Boosting Classifier (HGBC), and Ridge classifier. Through rigorous evaluation using performance metrics such as F1 score and Cohen's Kappa score, we iteratively refined our models, addressing challenges such as class imbalance, feature selection, and engineering. The HGBC model yielded the best performances. However, despite our attempts to enhance its performance using GridSearchCV or RandomizedSearchCV, computational times prevented significant improvements.

However, our project is not without limitations. While our models achieved commendable performance on the provided dataset, their efficacy in diverse and dynamic crisis situations remains to be validated. Furthermore, the interpretability of our models and the generalizability of our findings to different contexts warrant further investigation. Given more time for fine-tuning the model and access to more powerful machines, this project holds even greater potential.

In conclusion, our project highlights the immense potential of machine learning in extracting valuable information from social media during crisis events. By leveraging advanced algorithms and methodologies, we can empower emergency responders with timely and actionable insights, ultimately contributing to more efficient and effective crisis management strategies.

References

[odlNTIS20] Les organisateurs de l'initiative NIST TREC Incident Stream. Description de l'initiative nist trec incident stream 2020, 2020.

8 Annexe

Category	Number of tweets
Hashtags	10605
News	10355
Sentiment	9912
MultimediaShare	9391
Irrelevant	7563
Location	7087
Factoid	6838
ThirdPartyObservation	6196
FirstPartyObservation	4894
Discussion	3864
OriginalEvent	3371
Weather	3109
EmergingThreats	2460
Advice	2253
ContextualInformation	2102
Official	1638
ServiceAvailable	1564
Donations	952
NewSubEvent	868
MovePeople	387
InformationWanted	281
SearchAndRescue	261
CleanUp	219
Volunteer	210
GoodsServices	186

Table 6: Number of tweets per category

Feature	Description
degree_centrality	Degree centrality of nodes
topic_encoded	Encoded topic
sim_559	Similarity score 559
sim_569	Similarity score 569
sim_561	Similarity score 561
sim_572	Similarity score 572
sim_558	Similarity score 558
sim_575	Similarity score 575
sim_577	Similarity score 577
EventPlace_encoded	Encoded event place
sim_580	Similarity score 580
sim_564	Similarity score 564
sim_560	Similarity score 560
sim_578	Similarity score 578
sim_562	Similarity score 562
sim_574	Similarity score 574
sim_579	Similarity score 579
sim_567	Similarity score 567
sim_573	Similarity score 573
sim_576	Similarity score 576
sim_557	Similarity score 557
sim_581	Similarity score 581
sim_571	Similarity score 571
sim_566	Similarity score 566
sim_565	Similarity score 565
sim_570	Similarity score 570
sim_563	Similarity score 563
sim_568	Similarity score 568
event_type_encoded	Encoded event type
retweet_count	Number of retweets
closeness_centrality	Closeness centrality of nodes
rescueph	Presence of 'rescueph' keyword
mozambique	Presence of 'mozambique' keyword
n562_#	Frequency of hashtag 'n562'
wildfire	Presence of 'wildfire' keyword
n561_#	Frequency of hashtag 'n561'
n559_#	Frequency of hashtag 'n559'

Table 7: Feature importance (1)

Feature	Description
n576_#	Frequency of hashtag 'n576'
n558_#	Frequency of hashtag 'n558'
n577_#	Frequency of hashtag 'n577'
n567_#	Frequency of hashtag 'n567'
n569_#	Frequency of hashtag 'n569'
help	Presence of 'help' keyword
n564_#	Frequency of hashtag 'n564'
need	Presence of 'need' keyword
abfire	Presence of 'abfire' keyword
n580_#	Frequency of hashtag 'n580'
n574_#	Frequency of hashtag 'n574'
n575_#	Frequency of hashtag 'n575'
n573_#	Frequency of hashtag 'n573'
n572_#	Frequency of hashtag 'n572'
n557_#	Frequency of hashtag 'n557'
n563_#	Frequency of hashtag 'n563'
n560_#	Frequency of hashtag 'n560'
n578_#	Frequency of hashtag 'n578'
cyclone	Presence of 'cyclone' keyword
n571_#	Frequency of hashtag 'n571'
n568_#	Frequency of hashtag 'n568'
ha	Presence of 'ha' keyword
n565_#	Frequency of hashtag 'n565'
n570_#	Frequency of hashtag 'n570'
n581_#	Frequency of hashtag 'n581'
n579_#	Frequency of hashtag 'n579'
ymmfire	Presence of 'ymmfire' keyword
n566_#	Frequency of hashtag 'n566'
evacuation	Presence of 'evacuation' keyword
people	Presence of 'people' keyword
high	Presence of 'high' keyword
cyclonekenneth	Presence of 'cyclonekenneth' keyword
alberta	Presence of 'alberta' keyword
level	Presence of 'level' keyword
donate	Presence of 'donate' keyword
earthquake	Presence of 'earthquake' keyword
shooting	Presence of 'shooting' keyword
flood	Presence of 'flood' keyword

Table 8: Feature importance (2)

Feature	Description
affected	Presence of 'affected' keyword
school	Presence of 'school' keyword
emergency	Presence of 'emergency' keyword
kenneth	Presence of 'kenneth' keyword
philippine	Presence of 'philippine' keyword
florida	Presence of 'florida' keyword
nepal	Presence of 'nepal' keyword
lake	Presence of 'lake' keyword
amp	Presence of 'amp' keyword
area	Presence of 'area' keyword
water	Presence of 'water' keyword
home	Presence of 'home' keyword
power	Presence of 'power' keyword
flooding	Presence of 'flooding' keyword
update	Presence of 'update' keyword
resident	Presence of 'resident' keyword
hit	Presence of 'hit' keyword
synagogue	Presence of 'synagogue' keyword
say	Presence of 'say' keyword
relief	Presence of 'relief' keyword
wa	Presence of 'wa' keyword
news	Presence of 'news' keyword
safe	Presence of 'safe' keyword
typhoon	Presence of 'typhoon' keyword
ymm	Presence of 'ymm' keyword

Table 9: Feature importance (3)