International Journal of Social Science and Human Research

ISSN (print): 2644-0679, ISSN (online): 2644-0695

Volume 07 Issue 08 August 2024

DOI: 10.47191/ijsshr/v7-i08-90, Impact factor- 7.876

Page No: 6563-6579

Digital Echoes: Empowering Crisis Management and Urban Resilience Strategies with Real-Time Social Media Insights

Denis-Cătălin ARGHIR

Bucharest University of Economic Studies, Piața Romană, 6, Bucharest, Romania

ABSTRACT: This paper proposes an innovative approach for decision-makers, introducing a web application that employs a microservices architecture to analyze data gathered from social networks. In the context of smart cities, citizens act as "sensors", providing real-time insights through their online activities. By sharing locations and posting content, they generate data that can be analyzed using supervised machine learning algorithms and natural language processing techniques to identify significant urban events. The application was trained on a pre-labeled dataset from X, utilizing various machine learning models and NLP preprocessing techniques to achieve high accuracy in message classification. The dataset comprises texts with keywords related to disruptive events, such as fires, floods, heatwaves, and more severe contemporary issues like terrorism, war, or epidemics, labeled as either disruptive or neutral. Comprehensive testing was conducted using X's API, focusing on acquiring messages from specific areas. This approach can enable more proactive city management and timely resource allocation, improving overall crisis response and urban planning.

KEYWORDS: X social network, microservices web application, machine learning, natural language processing, resilient smart city, crisis management

I. INTRODUCTION

Human settlements have evolved continuously over millennia, adapting to environmental factors, social organization, technological advances, industrial revolutions, and economic opportunities. Today, with a global population of around 8 billion (Population Media Center, 2024), over half reside in urban areas. This urbanization emphasizes the need for innovative urban planning, sustainable development, and advanced technologies to create resilient and efficient urban environments (Arghir, 2024).

In recent years, technology has developed rapidly and has changed the manner in which people interact. This progress influences multiple aspects of citizens' lives, determining the need for integration into as many activities as possible to optimize them. Cities, hosts of billions of citizens, are also part of this growing need for connectivity to become smarter, more sustainable, and more resilient. Cities integrate smart devices, infrastructure, sensors, and monitoring systems. These allow analyses of large datasets that improve urban management. In this context, mobile devices constitute a technical means that cannot be neglected. Thus, citizens themselves become "sensors" due to digital interconnection (Sagl & Resch, 2015). Users share their location, post messages, and multimedia content on various platforms and social networks. Experiences and feedback shared on social media become valuable sources for understanding community problems. Problems identified by citizens allow for solutions in various stages, such as mitigation, preparedness, response, and recovery, thus contributing to urban area resilience (Kanteler & Bakouros, 2024).

The primary objective of this study was to develop a web application based on a microservice architecture that interacts with data in text format from social networks, which can serve as a decision-making aid for urban planners and policymakers. The application works in two directions: the first, is represented by the automation of training based on some sets of pre-labeled data, on which multiple Machine Learning algorithms are applied, to create an optimized model for producing predictions. In this way, the texts are preprocessed, various vectorization techniques are applied, and the models are fine-tuned. The second direction involves the capture of historical or real-time messages from the selected social media API - X, classifying new messages based on trained models, and improving models based on newly acquired relevant messages.

The paper is structured into six key sections. Chapter Two provides a comprehensive literature review, establishing the foundation for the study. Chapter Three delves into the primary classification algorithms utilized within the developed web application. Chapter Four outlines the methodological approach, detailing the framework and procedures employed in the research. Chapter Five presents and analyzes the main results and findings. The final section summarizes the insights and implications drawn from the study.

II. STATE OF THE ART

Social networks emerged in the early 1990s, revolutionizing interpersonal communication by enabling virtual connections and transcending geographical limitations (Maryville University, 2020). The objectives of social networks are multifaceted, encompassing the management of professional careers, facilitating interpersonal communication, enhancing the visibility of professional endeavors, sharing opinions and reactions, and distributing multimedia content such as images and videos (Portal Management, 2022). Their popularity surged as Internet users increased. Over time, the advent of high-speed Internet connections has brought performance parity between mobile and fixed devices. Social media platforms have developed mobile-optimized applications that enhance user interaction and increase the time spent on such platforms (Mehta, 2023). Mobile devices have empowered users to share their opinions directly from the heart of their attending events, in stark contrast to the era when desktop computers were predominant and information dissemination was significantly delayed.

In the context of the existence of numerous social networks, X, formerly known as Twitter, was selected for the experimental study because it presents unique characteristics that make it ideal for capturing messages focused on relevant topics. X is primarily perceived as a valuable source on which topical discussions and real-time posts about ongoing events can be shared, along with metadata that are crucial for geographically focused experiments. According to a study (Robertson, 2023), 25% of users consider X an excellent platform for information, and 20% appreciate the debates and comments that accompany posts. Moreover, 17% of users believe that the platform offers perspectives not addressed in traditional media, and 11% believe that it offers content dedicated to personal needs and interests. In all these categories, X is considered the leader, remaining a valuable source of relevant messages and preferred by premium users and opinion leaders. This network is less entertainment-oriented (only 10% use it for this) compared to other platforms such as TikTok (35%), Instagram (23%), YouTube (21%), and Facebook (16%). This orientation toward serious and rapid information, active debates, and diverse perspectives makes it ideal for impact studies. This platform lets users share thoughts in messages up to 280 characters, with premium accounts allowing up to 25,000. It serves as a global forum for discussions. The interest subjects are signaled by keywords annotated with hashtags (O'Brien, 2023), and retweets and likes give amplitude to important messages of the moment. X remains relevant, with ongoing investments and developments such as the Grok Chatbot by its AI division. Grok, currently in early access, uses a large language model (LLM) to generate content and answer user questions. It promises to be an essential open-source tool for real-time data interaction from Twitter (AI/ML API, 2024).

X has proven over time to be an extremely reliable tool during emergency situations, proving its ability to interconnect people from all over the world during critical situations, as was addressed in the paper (Uchida, et al., 2016). However, the raw data from X represents a first step in the crisis management. Using advanced processing and analytical techniques, simple messages can be transformed for the automated identification and monitoring of various situations, both positive and negative, that impact human settlements. Thus, with the help of Natural Language Processing (NLP), the contexts addressed in the messages, the main topics of discussion, and the entities mentioned in the messages, from places, objects, and people, to specific events, can be captured, leading to an understanding of the texts as close as possible to the human one (Clarissa & Marlo, 2018). Artificial intelligence is employed to automatically interpret messages, classifying them into specific categories or classes based on their content or context.

Research indicates various approaches to early warning systems using messages from X. These systems, analyzed through data science, complement traditional measurement methods. The ability of social media platforms to capture timely signals of significant events has been observed, such as natural disasters and severe weather phenomena (Yigitcanlar, et al., 2022), public health crises (Sigalo & Frias-Martinez, 2023), market trends (Reveiu & Arghir, 2020), power outages (Bauman, Tuzhilin, & Zaczynski, 2017), road accidents (Azhar, et al., 2022). These data can be utilized to implement preventive actions, swiftly mobilize resources, and promote strategies to restore the initial conditions. By monitoring the information shared on these platforms, authorities and organizations can gain a real-time overview of ongoing situations and intervene more effectively to reduce the negative impact on affected communities.

In the context of urban resilience, it is not the social network itself but the valuable output that is crucial for analysis. A series of studies have demonstrated the pivotal role of social media data in understanding and enhancing the capacity of cities to recover from various disturbances. In the study conducted in a Chinese region, data from Weibo microblogging platform was used to evaluate urban resilience in the face of flooding. By utilizing technologies like GooSeeker crawler, Correlation Explanation (CorEx) topic model, TF-IDF, and SentiStrength sentiment analysis, researchers provided a detailed assessment of how infrastructure and the psychological state of the population were impacted and recovered post-disaster. The study emphasizes the significance of recovery alongside infrastructure restoration (Siqing & Feng, 2023). Similarly, in another Chinese experimental study, geo-tagged data and sentiment analysis were employed to understand the spatio-temporal patterns of public reactions on social media during floods. Robust regression models suggested policy measures to improve infrastructure and resilience plans for efficiently urban resources management (Wang, Loo, Zhen, & Xi, 2020). Another comprehensive approach focuses on real-time data analysis, big data, AI/Machine Learning, sensor data, Earth Observation, and geolocation data to enhance cyber-resilience and risk identification. These underscores the critical role of advanced technologies in improving adaptability to unforeseen events and managing smart infrastructure (ERRIN, 2024). In a Germanian study, Twitter data and an online survey were used to assess community knowledge and perceptions about floods. The thematic and spatio-temporal analysis provided valuable insights into how communities perceive and respond to natural disasters, highlighting the importance of community participation and public perceptions in managing urban resilience (Moghadas, Fekete, Rajabifard, & Kotter, 2023). In a greek-related study, an innovative project developed and applied Digital Twin technology to monitor, simulate, and analyze urban systems in real-time at the neighborhood level. Advanced data analytics, predictive models, and interactive platforms were used to dynamically monitor the city. This study concludes that Digital Twin technology can significantly improve urban functionality, resilience, and residents' quality of life (Gkontzis, Kotsiantis, Feretzakis, & Verykios, 2024). These studies highlight the critical role of social media and advanced analytics in enhancing urban resilience. They show that integrating these technologies into urban monitoring and management can lead to more effective interventions, better planning, and ultimately more resilient cities. The findings represent models of good practices that can be applied to improve urban quality of life and safety.

Facilitating a complete process that allows easy data gathering, machine learning model training, and data analysis is valuable, especially in the context of the use by people who are not necessarily specialists from a technical point of view but hold decision-making roles in the urban management and planning; the democratization of the use of AI is a topical direction (Tjaden & Tjaden, 2023). Decomposing an application into small and independent components that can be run separately, called microservices, is a modern methodological approach that allows for scalable, flexible, and agile development. This allows frequent and isolated updates to different parts of the application without disrupting the operation of the entire application in the event of a service failure. Many studies have emphasized the advantages of such microservice architectures (Tapia, et al., 2020), confirming their efficiency and relevance in various fields and applications.

III. MACHINE LEARNING CLASSIFIERS

Machine learning algorithms play a significant role in knowledge extraction and predictive modeling, offering significant applications across various domains. In particular, supervised classification algorithms constitute an essential component for predictive analytics, allowing efficient identification and categorization of new data based on models generated from trained datasets. In Table 1, an analysis was conducted to navigate part of the varied landscape of classification techniques, summarizing the way of work and the importance for the analysis.

Table 1 - Overview of Selected Algorithms

Classifier	Summary Explanation	Importance for the Analysis
Logistic Regression (LR)	- Binary classification algorithm; - Model the relationship between the dependent variable and the independent variables using a logistic function; - The logistic function transforms the linear result into a probability between 0 and 1; - Implemented in sklearn/ linear_model/ LogisticRegression Python library (scikit-learn developers, 2024)[1].	- Easy to interpret and implement; - Ideal for initial classification tasks on vectorized text; - Coefficients provide clarity on the influence of each feature.
Logistic Regression Cross Validation (LRCV)	- Extend logistic regression by cross-validation to choose the best parameters; - It helps to avoid overfitting and improve model robustness; - Implemented in the sklearn/ linear_model/ LogisticRegressionCV Python library (scikit learn developers, 2024)[2].	- Enhances model generalization; - Suitable for text datasets with high variability.
Stochastic Gradient Descendent (SGD)	- Optimization method for fast model training on large data sets; - Updates model parameters based on each training subset or small mini-batch; - Implemented in the sklearn/ linear_model/ SGDClassifier Python library (scikit learn developers , 2014)[3].	- Ensures rapid convergence, ideal for adapting to new forms of text or changes in data distribution; - Manages sparse data, common in vectorized text representations.
Linear Support	- Classification algorithm that constructs an optimal hyperplane to separate two distinct classes;	- Highly effective for text classification;

Vector Classification (LSVC)	- It maximizes the margin between classes to reduce the risk of overfitting; - Implemented in the sklearn/ svm/ LinearSVC library (scikit learn developers , 2024)[4], (Zisserman, 2015).	- Manages high-dimensional feature vectors well; - Offers clear separation between classes.
Random Forest Classifier (RFC)	- Ensemble algorithm that builds a forest of decision trees trained on subsets of the data; - Use the bootstrap aggregating (bagging) method; - Implemented in the sklearn/ ensemble/ RandomForestClassifier library (scikit learn developers , 2024)[5].	- Robust and flexible; - Handles complexity and large dimensions of text vectors; - Captures complex relationships between features.
Gradient Boosting (GB)	- Ensemble algorithm that builds sequential decision trees to correct errors made by previous trees; - Minimize the loss function using the gradient descent method; - Implemented in the sklearn/ ensemble/ GradientBoostingClassifier library (scikit learn developers, 2024)[6].	- High performance on complex datasets; - Good for advanced text classification; - Captures subtle relationships in the data.
Decision Tree (DT)	- Build a classification model in the form of a decision tree; - Each internal node represents a feature and each leaf represents a class or decision value; - Implemented in the sklearn/ tree/ DecisionTreeClassifier library (scikit learn developers , 2024)[7], (Breiman, Friedman, Olshen, & Stone, 2017).	- Simple to understand and visualize; - Handles non-linear relationships well; - Suitable for smaller datasets or as part of ensemble methods.
K Neighbors (KN)	- Classifies a data point according to its k-nearest neighbors in the training set. - Effective for datasets with few variables and well separated classes; - Implemented in the sklearn/ neighbors/ KNeighborsClassifier library (scikit learn developers, 2024)[8].	- Useful when there is a clear structure in the vectorized data; - Performs well on well-separated classes; - Simple and intuitive.
Bernoulli Naive Bayes (BNB)	- Variant of the Naive Bayes classifier for binary data; - Calculates the probability that an instance belongs to a class based on the presence or absence of a feature; - Implemented in the sklearn/ naive_bayes/ BernoulliNB library (scikit learn developers, 2024)[9].	- Highly efficient for text classification; - Performs well on sparse binary text data; - Fast and easy to implement.

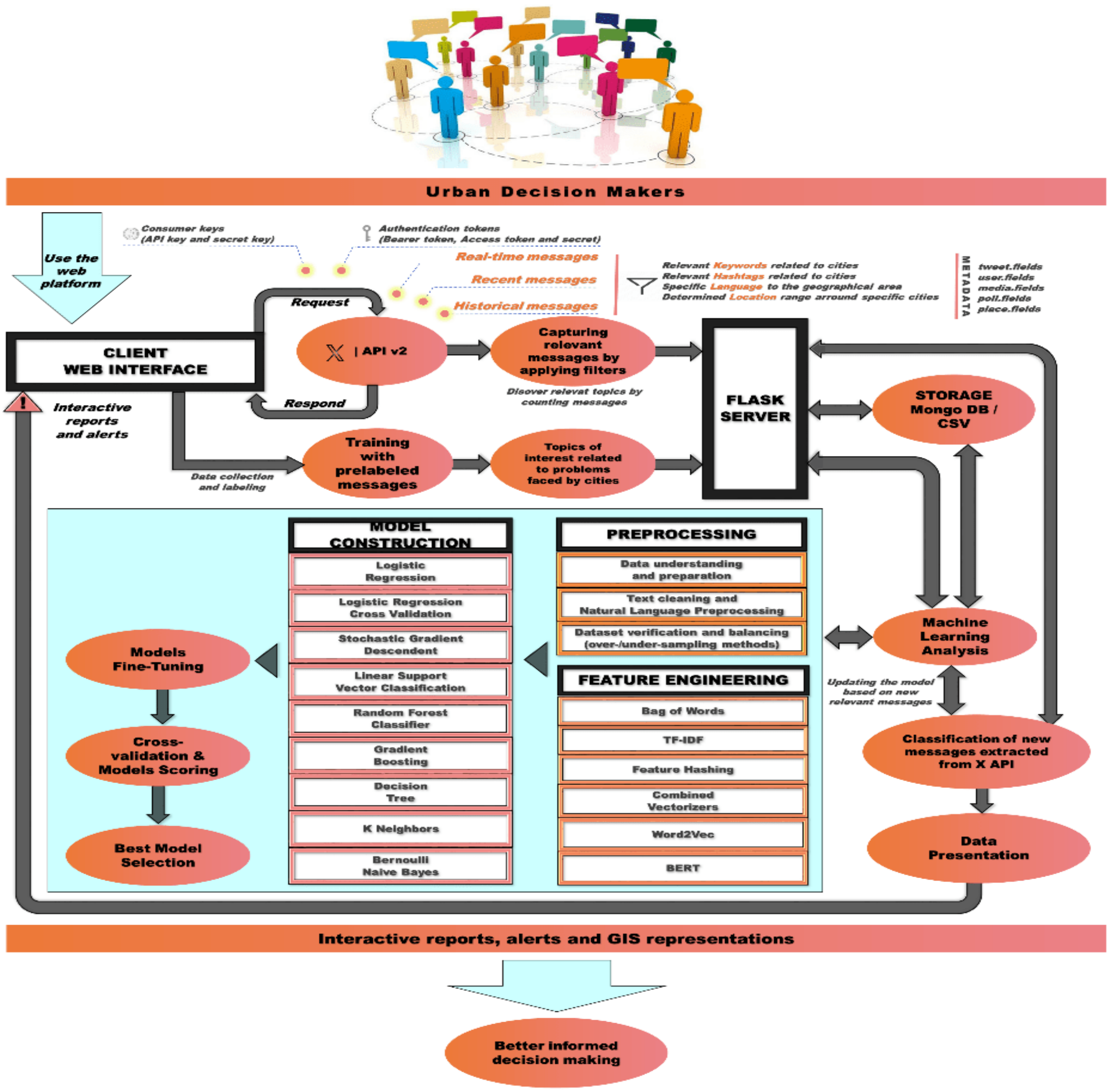
(Source: author's own representation)

The web application incorporates these supervised classification algorithms, each optimized for performance through specific fine-tuning. This approach enhances the adaptability, scalability, and effectiveness of the machine learning solutions, forming a robust foundation for data-driven decisions.

IV. METHODOLOGICAL APPROACH

This paper proposes a web application based on a microservice architecture. The proposed model is based on the Python ecosystem and Flask microframework. Unlike monolithic architectures, this type of architecture allows for scalable development, allowing the addition of new functionalities as development progresses (Tapia, et al., 2020). The application implements a responsive design, working on both mobile and desktop devices. Data processing is handled server-side, ensuring efficient performance regardless of the device. The microservice implementation is performed using the Flask framework, and the execution involves the WSGI Gunicorn server. The application runs with two inputs. The first input is used for training, which allows the introduction of pre-labeled data files on which the analysis is based. The second input is the real-time search of messages on the X platform. The message queues are optimized via asynchronous processing with the RabbitMQ message broker.

The stream of real-time messages is managed and then transmitted to microservices that handle processing, analysis, and interpretation. The number of messages that can be obtained in a month through the X API is quite limited; the storage of messages is important in the system testing process to explore their valuable characteristics. Despite this limitation, the experimental study used the existing API constraints as an initial framework to test and demonstrate key concepts. In a real-world implementation, the project can be scaled by opting for a higher-tier API package or integrating third-party libraries like Tweepy or Twython to compensate for the official API limitations, thus meeting the specific requirements of an extended project that can be used in daily life, enabling decision-makers to draw pertinent conclusions. Data persistence is managed using a schema-less NoSQL MongoDB database. The working pipeline of the entire application is presented in the Figure 1:

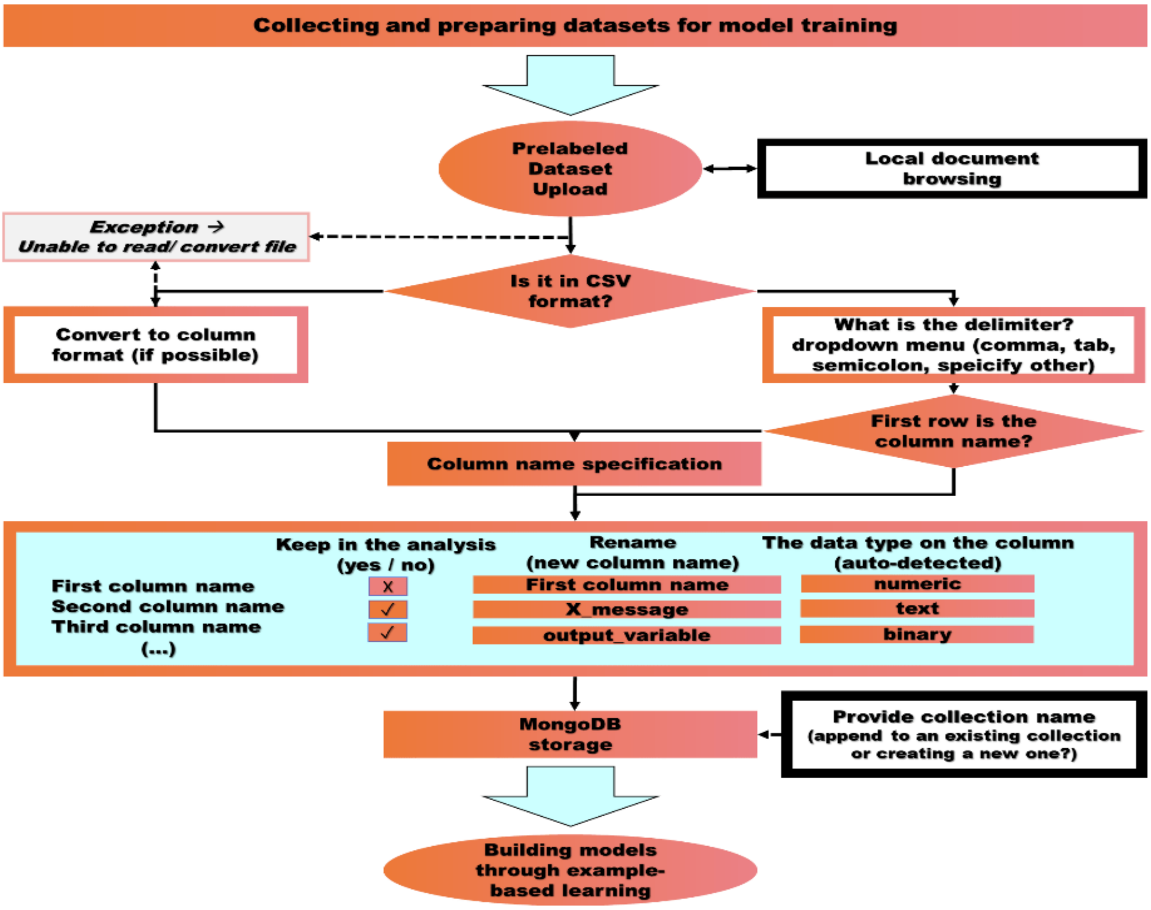


(Source: author's own representation)
Figure 1 - Application Pipeline

The proposed microservice approach involves the following steps:

- *Loading the pre-labeled dataset* – the client-type graphic interface facilitates the loading of pre-tagged CSV data files in the web application to substantiate the classification models. A dedicated microservice retrieves and manipulates the data files. The dataset is initially displayed for the client's preview in a Pandas DataFrame tabular format. The columns for storage can be specified, various formatting can be applied, and the columns can be renamed. The column of the target variable allows processing to replace the values with 0 and 1 if the column is completed with other data (eg "yes"/"no"). Once the dataset is properly formatted, the microservice receives the changes for storage in a MongoDB collection. The interaction is performed using PyMongo, the process of saving having a dedicated microservice. To consume the microservice, a POST request is made to the route "/upload_file" with the data to be uploaded. If a new name is specified, a dedicated collection will be created in the database, or an existing collection can be selected to be appended. The flow is presented in Fig 2.
- *Understanding the dataset* – the data is taken from the database and de-serialized from the JSON document through interconnection with PyMongo. To consume the microservice, a GET request is made to the "/read_file" route, which is handled with the help of Pandas. Once retrieved, Exploratory Data Analysis is performed to evaluate the dataset. It is followed if there are duplicates, anomalies, or missing values and if the corresponding structure and format are respected. For the field associated with the text message, it is verified that the length is between 0 and 280 characters. From the perspective of the dependent variable (target), is verified that it exclusively contains values of 0 and 1, according to the binary classification task. These changes are reflected by making a POST request to the "analyze_file" route. Various

representations, including graphics, are generated using Matplotlib and Seaborn. Changes are pushed to the MongoDB set to reflect the handling of anomalies identified by a PUT ”update_file” request.



(Source: author's own representation)

Figure 2 - Prelabeled Dataset Upload

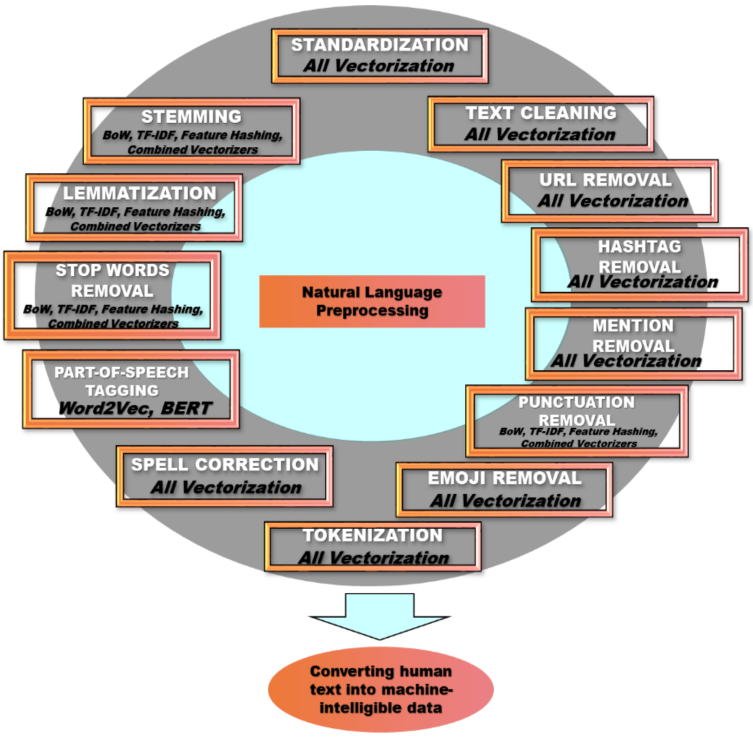
- *Balance of the dataset* – the data is taken from the database and de-serialized from the JSON document through interconnection with PyMongo. If the operation of understanding the dataset identifies an imbalance in the dataset, the microservice dedicated to balancing it is called, respectively, the weight of messages classified as 0 compared to those classified as 1 should be balanced as much as possible. Depending on the specific requirements, the subsampling method "imblearn.under_sampling" or over-sampling "imblearn.over_sampling" will be applied. To consume the microservice, a POST request is made to the "/balance_file" route. The limit thresholds at which one generally opts for balancing are generally represented by moderate to extreme values; however, the necessity at the time of the analysis will also be considered. In Table 2, the degrees of imbalance are presented:

Table 2 - Overview of Class Imbalance

Degree of Imbalance	Proportion of Minority Class
Mild	20-40% of the data set
Moderate	1-20% of the data set
Extreme	under 1% of the data set

(Source: (Google, n.d.))

- *Preprocessing messages with NLP-type processing* – messages on the X platform are preprocessed from the perspective of the column containing the text, eliminating other associated metadata. This microservice achieves the standardization of English messages, which are then passed through an NLP-type processing flow. The task includes standardization, text cleaning, removal of URLs, hashtags, mentions, punctuation and emoticons, tokenization (breaking messages into individual words), spelling correction and part-of-speech recognition, removal of linking words, lemmatization (reduction of words to the canonical form), stemming (reducing words to their basic form). Applying the best possible flow in the preprocessing stage improves the performance of placing messages in the correct classes. The processing is integrated with the initial data by mapping the data with a unique identifier. After processing, the microservice updates the data in the client interface. The stages through which messages are passed are illustrated in Fig 3.



(Source: author's own representation)

Figure 3 - NLP Preprocessing Based on Vectorization Technique

- *Feature Engineering (vectorization of messages)* – this microservice processes messages to transform unstructured textual data into structured numerical values. This process allows the use of ML techniques and algorithms that operate on numerical data. The preprocessing stages are applied separately for the types of vectorization; for the first vectorization techniques, the entire spectrum of preprocessing is performed because the techniques are not strongly based on the context of the messages; for Word2Vec, milder preprocessing is performed, maintaining the stop words to preserve the context of the messages. For BERT, minimal preprocessing is applied to maintain the message context. The specific preprocessing steps are presented in Fig 3.
  - Bag of Words (BoW) – the messages are represented as vectors of word frequencies, allowing quantitative analysis of the text (Khanna, 2024);
  - Term Frequency–Inverse Document Frequency (TF-IDF) – messages are transformed into vectors that reflect the relative importance of words, thereby helping filter common terms and emphasize relevant terms (Weiss, Indurkhya, Zhang, & Damerau, 2005);
  - Feature Hashing – messages are transformed into fixed-size vectors by applying a hash function to individual words, providing efficient and fast representation of text;
  - Combined Vectorization – messages are represented by a combination of vectorization techniques by adding BoW CountVectorizer(), TF-IDF TfidfVectorizer(), and Feature Hashing HashingVectorizer() vectorizations to the set of combined vectorizations through FeatureUnion. A composite vector is obtained that allows a more complete representation of the messages;
  - Word2Vec – the messages are transformed into vectors that capture the semantic and syntactic relations between the words, thereby allowing understanding of the context of the words in the sentence (Khanna, 2024);
  - Bidirectional Encoder Representations from Transformers (BERT) – the messages are transformed into vectors that capture the bidirectional context of the words, thereby providing an advanced contextual representation (Khanna, 2024).

- *Training models based on pre-labeled sets* – the dedicated microservice passes properly formatted data to the training phase of the supervised classification model. The dataset is divided into two subsamples: training and testing. The dataset is analyzed using multiple algorithms, which are described in Section 2. The models are evaluated using various metrics to evaluate performance objectively. The hyperparameters of the models are adjusted to maximize the classification quality. The best-performing model is saved for the classification of new messages.
- *Acquisition of data from the X platform* – The X Developer Platform provides four access tiers (API version 2) - Free, Basic, Pro, and Enterprise. The Free plan allows 1,500 posts monthly, Basic includes 3,000 posts and 10,000 retrievals, and Pro extends to 300,000 posts and 1,000,000 retrievals with additional features like complex filters and historical data. The Enterprise plan is customizable for specific high-demand projects. Table 3 shows the filters that can be applied to limit the content of messages to fit specific needs in the context of cities.

Table 3 - X API Specific Filters

Type	Filter	Explanation
Keywords	“exact match phrase”	match a specific phrase within messages
	“OR”	contain at least one of the specified terms
	“emoji”	contain a specific emoticon
	“from:”	message from a specific user
	“to:”	reply to a particular user
	“url:”	contain validly-formatted URL
	“entity”	specific entity
	“has:links”	contain links and media
	“has:media”	contain photo, GIF, or video
Location	“place”	specified location or ID
	“place coutry”	country code
	“point radius”	longitude, latitude and radius in km or mi
	“bounding box”	west long, south lat, east long, north lat
	“ has:geo”	return geolocated messages
Hashtag	“#”	contain a specific hashtag
	“has:hashtags”	contain at least one hashtag
Language	“lang:”	particular language specific for the country to which the city belongs
Retweets	“retweets of”	retweets of a specified user
	“is:retweet”	return retweeted messages

(Source: author's own representation based on (X Developer Platform, 2024)[1])

Table 4 lists the main endpoints queried in the developed application. In addition, multiple other components were used to facilitate specific searches. For example, an endpoint was integrated that transforms the username into the user id to facilitate queries. The longitude and latitude of a city, as well as its radius, is determined automatically by entering the name of the city and the country in which it is located.

Table 4 - The Main X API Endpoints

X Endpoints	Analyzed period	Explanation
GET /2/tweets/search/recent	Recent search	Returns historical messages, filtered by various criteria.
GET /2/tweets/search/all	Full-archive search	Returns historical messages, filtered by various criteria.
GET /2/tweets/search/stream/rules	Real-time stream	Returns real-time messages, filtered by various criteria.
GET /2/tweets/search/stream	Real-time stream	Returns real-time messages, filtered by various criteria.
GET /2/users/:id/tweets	Recent search	Returns messages created by a specific user.
GET /2/tweets/counts/recent	Recent search	Returns the number of messages for a specific search query, filtered by various criteria.
GET /2/tweets/counts/all	Full-archive counts	Returns the number of messages for a specific search query, filtered by various criteria.
GET /2/tweets/search/recent	Recent search	Returns historical messages, filtered by various criteria.

(Source: author's own representation based on (X Developer Platform, 2024)[2])

- *X Message verification* – In addition to text message analysis, the associated metadata such as the number of likes on a particular post helps in assessing the impact of the message. Also, the user credibility is evaluated based on account age and posting frequency. A microservice filters potential spam by excluding new accounts (under 30 days old) and those with unusually high activity. It checks for irregular and repetitive posting patterns indicative of automation. A notable follower-following imbalance might also suggest spam or bot activity. Behavioral analysis using rule-based models filters accounts showing signs of automation.
- *Data management* – involves storing messages in MongoDB for historical access or managing them asynchronously in a queue for real-time processing to handle large data volumes efficiently, all while ensuring user anonymity and privacy.
- *Geolocation of messages* – the associated metadata regarding the geolocation of the authors of the X posts, or regarding the location from the moment the messages were posted, depending on their availability, will be decoded to create a geographic representation that shows their distribution. This is achieved through a dedicated microservice based on the open-source library Open Steet Map "Nominatim". Generally, high volumes of geolocalized messages for analysis and the Nominatim API are limited in terms of the number of requests and are processed through various procedures. First, messages identified as having a specified longitude/latitude are no longer decoded; the common locations of several messages are treated only once. The messages are divided into smaller batches and treated in parallel, to avoid overloading the API. To consume the microservice, a GET request is made to the "/geocode_message" route.
- *Classification of new messages* – the dedicated microservice is based on data obtained from the X platform, either historical or real-time, and is passed through the preprocessing stage with NLP-type processing. A pre-trained classification model is used to make predictions regarding the classification in one of the identified classes. Relevant messages are used to update the models.
- *Data presentation* – the microservice enhances data presentation by generating interactive reports and alerts for captured and classified messages. It employs the Leaflet GIS library to display each message on an interactive map. Messages that signal potential dangers are distinctly marked in red, and the size of the pin varies to indicate how recent each message is. This setup provides an effective visual tool for rapidly identifying and responding to key data points and trends.

V. RESULTS AND DISCUSSIONS

A. Training with prelabeled X messages

Testing of the system based on web technologies was carried out by going through the developed flow. Thus, a dataset from the Kaggle platform was identified. The topics covered in this dataset encompass major events related to disruptive occurrences. The messages in the dataset were collected during a study in 2020 (Stepanenko & Liubko, 2020), with the aim of capturing reactions associated with keywords linked to disasters, indicating a methodology focused on global interest events. The dataset maintains the basic structure of earlier versions, having been updated to include current topics, the most notable being the COVID-19 pandemic. Although the messages are predominantly geolocated outside Europe, the typology of disruptive events and the nature of reactions are applicable to other geographical areas. Similar events, such as natural disasters or public emergencies, can occur anywhere in the world, making the insights gained from this dataset a valuable framework for identifying situations before they occur.

The dataset is in CSV format, with 5 columns "id", "keyword", "location", "text", and "target". The data understanding operation reveals that the file contains 11,370 messages from the X platform, pre-labeled on the "target" column with the value 0 - 9,256 messages and with the value 1 - 2,114 messages. The value 1 represents messages classified as disruptive events, containing key terms related to communities' problems and reported on social media. In contrast, the 0 value indicates that the messages are neutral under this aspect. The other variables are represented by numerical columns "id" - the order number of the message, and text, which contain the keyword that summarizes disturbances with potential impact on the cities, the body of the messages, and the location, which can be the country, region, or city (either the location where the user who posted the message was geolocated, or the location detected at the time of posting, or the declared location). Figure 4 presents the words that suggest disturbances, weighted according to the number of messages included in these categories. The proposed web application allows data files to be uploaded directly from the interface. Thus, the CSV file containing the messages from the X platform is passed through a typical initial verification phase, and each column is detected to contain a common type of data. Records with missing values are checked. After preliminary checks, the dataset is stored in the MongoDB database. The text column containing the message is selected for applying the NLP preprocessing.

(Source: author's own representation based on analysis)

Figure 4 - Word Cloud Generated Based on the Dataset

The dataset is divided into training and testing sets. This is achieved using the web platform by parameterizing the percentage of data to be passed for training and testing. According to the proposed methodology, the text column is passed through multiple stages of text preprocessing using NLP techniques and then through multiple vectorization techniques to be transformed into numerical values. Based on the Machine Learning process by applying the classification algorithms selected in the analysis the system evaluates and determines the most effective model for accurately predicting outcomes based on the dataset.

B. Models Fine-Tuning

The classifiers included in the analysis, such as LR, LRCV, SGD, LSVC, RFC, GB, DT, KN, and BNB, were passed through a hyperparameter optimization operation to obtain the most reliable models. For this purpose, the GridSearchCV library was used. Each of the algorithms includes specific sets of hyperparameters, some of which were adjusted according to the vectorization method used. The obtained results are presented in Table 5:

Table 5 - Hyperparameter Tuning Results

LR	LRCV	SGD	LSCV	RFC	GB	DT	KN	BNB
Candidate Hyperparameters:								
C: [0.01, 0.1,1, 10, 100]	Cs: [1, 10, 100]	alpha: [0.0001, 0.001, 0.01, 0.1]	C: [0.01, 0.1, 1, 10, 100]	n estimators: [50, 100, 200]	n estimators: [50, 100, 200], learning rate: [0.01, 0.1, 0.5]	max depth: [None, 10, 20, 30, 40]	n neighbors: [3, 5, 7, 9]	alpha: [0.5, 1.0, 2.0]
Selected Combination of Hyperparameters for BoW:								
C: 1	Cs: 100	alpha: 0.001	C: 0.1	n estimators: 200	n estimators: 200, learning rate: 0.5	max depth: None	n neighbors: 3	alpha: 0.5
Selected Combination of Hyperparameters for TF-IDF:								
C: 100	Cs: 10	alpha: 0.0001	C: 1	n estimators: 50	n estimators: 200, learning rate: 0.5	max depth: 40	n neighbors: 3	alpha: 0.5

Selected Combination of Hyperparameters for Feature Hashing:								
C: 10	Cs: 100	alpha: 0.0001	C: 1	n estimators: 200	n estimators: 200, learning rate: 0.5	max depth: 30	n neighbors: 3	alpha: 0.5
Selected Combination of Hyperparameters for Combined Vectorization:								
C: 1	Cs: 100	alpha: 0.001	C: 0.1	n estimators: 200	n estimators: 200, learning rate: 0.5	max depth: 40	n neighbors: 3	alpha: 0.5
Selected Combination of Hyperparameters for Word2Vec:								
C: 100	Cs: 10	alpha: 0.0001	C: 100	n estimators: 200	n estimators: 200, learning rate: 0.5	max depth: 10	n neighbors: 3	alpha: 0.5
Selected Combination of Hyperparameters for BERT:								
C: 1	Cs: 100	alpha: 0.001	C: 0.1	n estimators: 100	n estimators: 200, learning rate: 0.1	max depth: 10	n neighbors: 9	alpha: 0.5

(Source: author's own representation based on analysis)

For LR and LRCV, the values for the parameter C and respectively Cs, vary significantly between the vectorization methods, indicating the need to adjust the level of regularization according to the complexity of the feature space. For example, TF-IDF requires a larger C to allow more flexible fitting of denser data. The alpha parameter for SGD is slightly adjusted to balance regularization against data density and complexity, with lower values used for methods that produce denser feature representations, such as Word2Vec. LSVC shows variations in the C parameter, reflecting different regularization needs to control the balance between bias and variance depending on the type of vectorization.

The RFC and GB use n_estimators to control the number of trees in the model, with a larger number indicating a more robust model. The GB further adjusts learning_rate to handle the contribution of each tree to the final model. For DT the max_depth parameter is variable, which is set to prevent overfitting to the characteristics of the data space, with tighter bounds for methods that produce large feature spaces. KN uses a consistent number of neighbors in most cases; thus, proximity in the feature space is a robust indicator of similarity between data points, regardless of the vectorization method used. For BNB, the alpha parameter is constant, suggesting a uniform approach to smoothing feature frequencies that is independent of the vectorization method used.

This hyperparameter optimization reveals a deep understanding of the interplay between the nature of the data, its representation via various vectorization techniques, and the complexity of the model. Careful selection of hyperparameters according to data characteristics can improve performance, which is essential for application to disruptive urban events.

C. Models Scoring and Best Model Selection

Monitoring machine learning algorithm performance can maximize the potential to create reliable and effective models for detecting disruptive events in smart cities. Indicators such as accuracy, precision, recall, and the F1 score provide a clear picture of the system's ability to correctly process and classify messages; thus, the model can then be replicated in real-life scenarios, from certain geographical areas, from certain periods, or in real-time, with the ability to self-learn as new relevant data are captured.

To evaluate the performance of the algorithms, for each of the 6 vectorization techniques (BoW, TF-IDF, Feature Hashing, Combined Vectorization, Word2Vec, BERT) the performances were tested by combining them with each of the 9 algorithms selected in the analysis.

Table 6 - Confusion Matrix and Performance Monitoring

Algorithm	TN	FP	FN	TP	Accuracy	Precision	Recall	F1
Vectorization technique: BoW								
LR	1817	61	157	239	0.9041	0.8989	0.9041	0.8987
LRCV	1820	58	166	230	0.9015	0.8959	0.9015	0.8951
SGD	1831	47	183	213	0.8989	0.8935	0.8989	0.8901
LSCV	1811	67	153	243	0.9033	0.8980	0.9033	0.8984

RFC	1853	25	217	179	0.8936	0.8921	0.8936	0.8791
GB	1820	58	204	192	0.8848	0.8764	0.8848	0.8739
DT	1743	135	184	212	0.8597	0.8534	0.8597	0.8560
KN	1867	11	305	91	0.8610	0.8653	0.8610	0.8251
BNB	1731	147	135	261	0.8760	0.8775	0.8760	0.8767
Vectorization technique: TF-IDF								
LR	1800	78	143	253	0.9028	0.8982	0.9028	0.8993
LRCV	1811	67	147	249	0.9059	0.9011	0.9059	0.9016
SGD	1827	51	181	215	0.8980	0.8922	0.8980	0.8897
LSCV	1814	64	151	245	0.9055	0.9005	0.9055	0.9007
RFC	1855	23	226	170	0.8905	0.8896	0.8905	0.8744
GB	1793	85	199	197	0.8751	0.8650	0.8751	0.8664
DT	1788	90	228	168	0.8602	0.8459	0.8602	0.8479
KN	1876	2	317	79	0.8597	0.8763	0.8597	0.8188
BNB	1731	147	135	261	0.8760	0.8775	0.8760	0.8767
Vectorization technique: Feature Hashing								
LR	1771	107	201	195	0.8646	0.8541	0.8646	0.8571
LRCV	1822	56	242	154	0.8690	0.8567	0.8690	0.8519
SGD	1819	59	239	157	0.8690	0.8565	0.8690	0.8527
LSCV	1782	96	198	198	0.8707	0.8606	0.8707	0.8629
RFC	1860	18	242	154	0.8857	0.8867	0.8857	0.8663
GB	1785	93	208	188	0.8676	0.8562	0.8676	0.8584
DT	1811	67	254	142	0.8588	0.8426	0.8588	0.8404
KN	1874	4	308	88	0.8628	0.8759	0.8628	0.8252
BNB	1743	135	232	164	0.8386	0.8244	0.8386	0.8294
Vectorization technique: Combined Vectorization								
LR	1812	66	151	245	0.9046	0.8995	0.9046	0.8999
LRCV	1817	61	160	236	0.9028	0.8974	0.9028	0.8971
SGD	1820	58	175	221	0.8975	0.8914	0.8975	0.8902
LSCV	1806	72	149	247	0.9028	0.8978	0.9028	0.8986
RFC	1853	25	218	178	0.8931	0.8916	0.8931	0.8785
GB	1792	86	195	201	0.8764	0.8668	0.8764	0.8683
DT	1795	83	230	166	0.8624	0.8482	0.8624	0.8493
KN	1868	10	299	97	0.8641	0.8698	0.8641	0.8299
BNB	1689	189	114	282	0.8668	0.8779	0.8668	0.8712
Vectorization technique: Word2Vec								
LR	1837	41	293	103	0.8531	0.8368	0.8531	0.8235
LRCV	1816	62	268	128	0.8549	0.8370	0.8549	0.8331
SGD	1878	0	396	0	0.8259	0.6820	0.8259	0.7471
LSCV	1851	27	312	84	0.8509	0.8385	0.8509	0.8143
RFC	1848	30	286	110	0.8610	0.8520	0.8610	0.8323
GB	1749	129	235	161	0.8399	0.8247	0.8399	0.8298
DT	1710	168	279	117	0.8034	0.7815	0.8034	0.7902
KN	1746	132	268	128	0.8241	0.8017	0.8241	0.8089
BNB	1873	5	391	5	0.8259	0.7703	0.8259	0.7512
Vectorization technique: BERT								
LR	1787	91	176	220	0.8826	0.8750	0.8826	0.8768
LRCV	1801	77	189	207	0.8830	0.8743	0.8830	0.8751
SGD	1755	123	155	241	0.8777	0.8741	0.8777	0.8757
LSCV	1794	84	181	215	0.8835	0.8754	0.8835	0.8768
RFC	1861	17	269	127	0.8742	0.8751	0.8742	0.8488
GB	1809	69	216	180	0.8747	0.8637	0.8747	0.8627

DT	1713	165	242	154	0.8210	0.8077	0.8210	0.8132
KN	1776	102	189	207	0.8720	0.8631	0.8720	0.8656
BNB	1420	458	100	296	0.7546	0.8399	0.7546	0.7799

(Source: author's own representation based on analysis)

The confusion matrix is a useful tool to evaluate the performance scores of classification algorithms, providing a detailed insight into the types of errors committed by models and the correctness of predictions. This is essential in the context of the proposed application, in which correct identification of disruptive messages can help minimize the effects of critical events through early alerts by tracking citizens' message flows. By examining the confusion matrix we can extract valuable insights:

- True Positives (TP) represent messages correctly identified as disruptive. High TP counts are necessary, indicating that the system can effectively detect real disruptive events that require immediate attention.
- True Negatives (TN) are correctly classified as non-disruptive messages that are critical for reducing false alarms and maintaining a state of normalcy when managing crisis response resources.
- False Positives (FP), although preferably avoided, are non-disruptive messages that are misclassified as disruptive. These errors can lead to the waste of intervention resources; however, a moderate level of FP may be acceptable in scenarios where the sensitivity of the system is maximized.
- False Negatives (FN) are the most critical errors in our context, representing disruptive messages that are not detected by the system. Reducing FN is essential because each unprocessed incident can have negative consequences. Therefore, identifying parameters that maximize prediction quality is an essential task in the application workflow.

Keeping in the foreground of the analysis an optimal balance between sensitivity (recall) and precision, which maximizes TP and minimizes FN, is thus desirable. LRCV and LSVC applied to TF-IDF vectorization demonstrated the best rates of these metrics, suggesting a superior ability to discriminate between disruptive and non-disruptive messages.

Accuracy represents the total percentage of correct predictions (both TP and TN) relative to the total number of messages. Equation 1 shows how to calculate accuracy. It is a useful indicator to evaluate model overall performance. The accuracy of the LRCV model along with the TF-IDF vectorization technique, was 90.59%, indicating an overall good prediction accuracy rate.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad [1]$$

Precision indicates the model's ability to correctly classify disruptive messages from all predictions labeled as disruptive. The Equation 2 shows how to calculate precision. High precision is crucial in the context of managing some crisis situations; it can focus attention on the really important alerts. The accuracy of the same model reached 90.11%, which means that most alerts generated by the system are relevant and worthy of further investigation.

$$Precision = \frac{TP}{TP+FP} \quad [2]$$

Recall indicates the percentage of actual disruptive events that the system was able to detect. The Equation 3 shows how the recall is calculated. In emergency situations, high recall is vital because an unprocessed event can have negative consequences. A recall value of 90.59% obtained by the same model indicates that the system can detect the most disturbing messages, which is an essential aspect of preventing crises.

$$Recall = \frac{TP}{TP+FN} \quad [3]$$

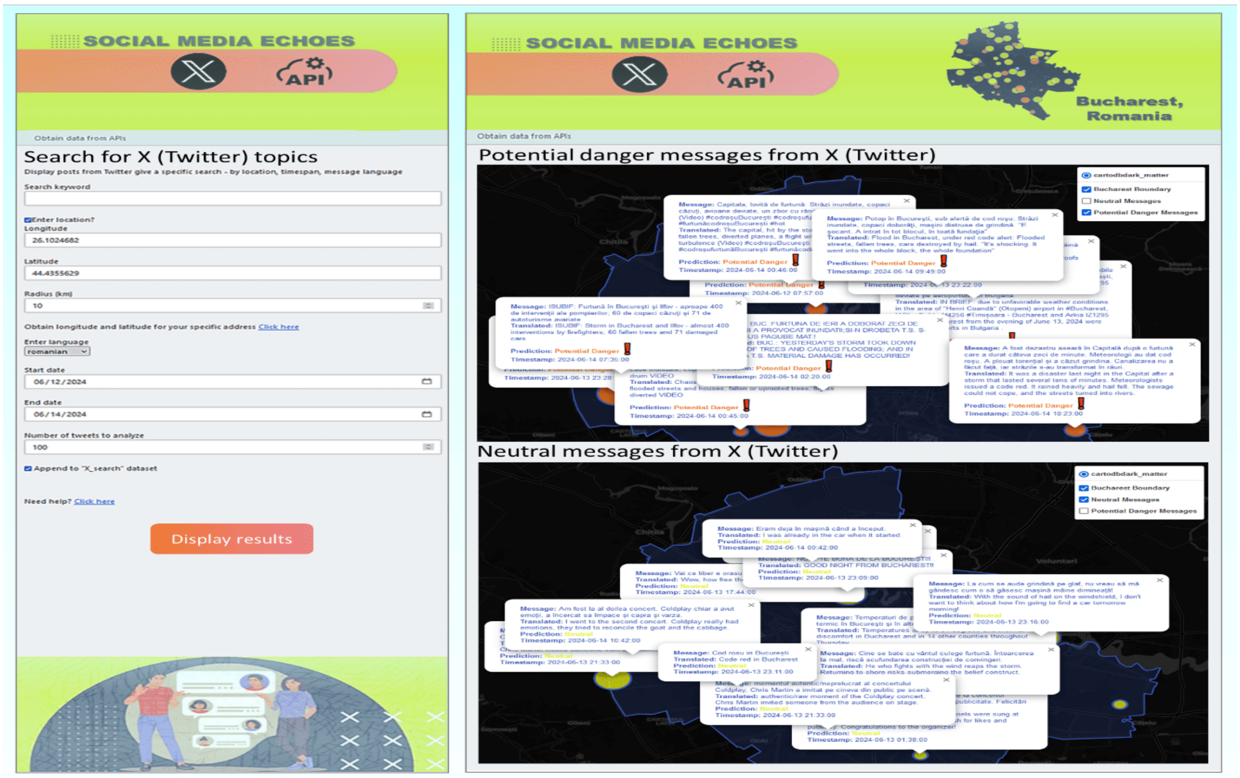
A high F1 Score indicates an effective balance between accuracy and the ability to reliably detect disruptive events, which maximizes the ability to evaluate model performance. The Equation 4 shows how to calculate the F1-score. The F1 score of 90.16% reflects a good balance between precision and recall for application in the management of disruptive events.

$$F1 = 2 * \frac{Precision * Recall}{Precision+Recall} \quad [4]$$

D. Testing the application in a real scenario

To validate the selected model, the X API was used to capture immediate impact messages. This integration allows the model to be tested under dynamic and diverse real-life conditions, thereby assessing its applicability. For example, an analysis was carried out for the classification of messages for 2 days in the interval June 12, 2024 to June 14, 2024 in the area of the city of Bucharest, Romania. The experimental analysis represents a micro-study for a nowcasting meteorological event, announced minutes before its occurrence by the National Meteorological Institute and the Emergency Situations Inspectorate through the RO-ALERT service. The storm had a local impact at the level of some neighborhoods in Bucharest, without having very violent effects, quantified by the injury of people.

The search on X was carried out using geographical restrictions in the area of the city of Bucharest, without other keyword limitation to automatically capture relevant subjects, exemplified in the Figure 5.



(Source: author's own representation)

Figure 5 - Capture-Search and Classification of New Obtained Messages

During the specified period, two major events occurred: a severe weather event causing damage to the city's infrastructure (hail, heavy rain, damage to the sewage and transport network, fallen trees, damaged cars, etc.), and a concert by an international band. Both events were discussed on the X network as main topics. In total, 50 messages were captured and analyzed using the selected classification model, which identified 14 as potentially dangerous due to the weather event. The distribution of these messages is illustrated in Figure 5, with map representation to simplify visualization despite the lack of precise street-level geolocation. The first map includes messages deemed potentially dangerous, mostly related to the severe weather event, and the second shows those classified as neutral. A subjective analysis of the messages labeled as potentially dangerous highlighted their significant impact. However, three similarly themed messages were classified as neutral, indicating a classification discrepancy. The generalization capability of the NLP model trained on the pre-labeled dataset was confirmed through testing on new data obtained from the Bucharest region. This demonstrated that the model can effectively identify and classify similar messages within a different geographical context. By implementing such an approach, authorities could more effectively identify high-risk topics.

By carefully analyzing the flow of these type of messages identified as having a potential risk, authorities can rapidly assess the impact of impending events, thus facilitating prompt and effective intervention. This strategic approach allows the triggering of well-structured action plans and the allocation of resources in an optimized way, essential for the effective management of crises. The study continues after this phase to capture other topics of interest expanding the learning base. This experimental analysis can be considered a preliminary statistic, providing initial insights into the system's behavior and allowing for its further refinement as more data is collected.

VI. CONCLUSION

This study demonstrates the effectiveness of integrating machine learning technologies into event monitoring and crisis management in the context of resilient smart cities. The process begins by training classification models on pre-labeled sets of messages from the X platform using multiple supervised classification algorithms. These algorithms were fine-tuned to maximize performance scores, including also text preprocessing and NLP vectorization techniques.

The selected model that provided the best predictions in terms of accuracy and F1 score was LRCV with TF-IDF, and it was then tested under dynamic real-world conditions. Using the X platform API, real-time messages were extracted from the Bucharest city area to evaluate the model's ability to automatically classify relevant messages. This testing allowed the verification of the predictive power of the model in a diverse context, highlighting how events can be detected and classified effectively. The model is subjected to continuous learning to improve its ability to face urban challenges.

The performance of the model under real-life conditions highlights its potential for implementation in the early detection of potentially disruptive incidents. By "listening" to the echoes from the digital world, systems can improve their ability to respond to challenging situations, strengthening communities' confidence in managing various crises. In the current phase, the system does not have the capacity to determine complex nuances of urban situations, requiring more granular and contextual classifications. The following studies will also explore other data sources exposed online for integration into a decision-making support system that will improve the application's ability to interpret and respond to urban problems.

REFERENCES

1) AI/ML API. (April 01, 2024). Deciphering Grok: An In-Depth Look into Elon Musk's AI Chatbot. (AI/ML API) Retrieved June 11, 2024, from AI/ML API: https://aimlapi.com/blog/grok-elon-musk-ai-chatbot.
2) Arghir, D.-C. (2024). From Ancient Streets to Connected Cities: Analyzing the Implementation of Smart Initiatives in Romania. Theoretical and Empirical Researches in Urban Management, 19(2), 5-27.
3) Azhar, A., Rubab, S., Khan, M. M., Bangash, Y. A., Alshehri, M. D., Illahi, F., & Bashir, A. K. (2022). Detection and prediction of traffic accidents using deep learning techniques. Cluster Computing-The Journal of Networks Software Tools and Applications, 26(1), 477-493, Available: https://doi.org/10.1007/s10586-021-03502-1.
4) Bauman, K., Tuzhilin, A., & Zaczynski, R. (2017). Using Social Sensors for Detecting Emergency Events: A Case of Power Outages in the Electrical Utility Industry. ACM Transactions on Management Information Systems, 8(2-3), 1-20, Available: http://dx.doi.org/10.1145/3052931.
5) Breiman, L., Friedman, J., Olshen, R., & Stone, C. J. (2017). Classification and Regression Trees. New York: Chapman and Hall/CRC.
6) Clarissa, X. C., & Marlo, S. (2018). Extraction and Classification of Semantic Data from Twitter. WEBMEDIA'18: Proceedings of the 24th Brazilian Symposium on Multimedia and the Web (pg. 15-18). Slavador: Association for Computing Machinery, Available: https://doi.org/10.1145/3243082.3264606.
7) ERRIN. (June 27, 2024). Advanced real-time data analysis used for infrastructure resilience. (European Regions Research and Innovation Network) Preluat pe July 17, 2024, de pe European Regions Research and Innovation Network: https://errin.eu/calls/advanced-real-time-data-analysis-used-infrastructure-resilience.
8) Gkontzis, A. F., Kotsiantis, S., Feretzakis, G., & Verykios, V. S. (2024). Enhancing Urban Resilience: Smart City Data Analyses,Forecasts, and Digital Twin Techniques at the Neighborhood Level. future internet, 16(47), 1-42, Available: https://doi.org/10.3390/fi16020047.
9) Google. Imbalanced Data. (Google) Retrieved May 20, 2024, from https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data.
10) Kanteler, D., & Bakouros, I. (2024). A collaborative framework for cross-border disaster management in the Balkans. International Journal of Disaster Risk Reduction, 108, 1-19. Availabe: https://doi.org/10.1016/j.ijdrr.2024.104506.
11) Khanna, V. (February 16, 2024). Text Becomes Insight with Vectorization. (Shelf) Retrieved June 12, 2024, from Shelf: https://shelf.io/blog/text-becomes-insight-with-vectorization.
12) Maryville University. (May 28, 2020). The Evolution of Social Media: How Did It Begin, and Where Could It Go Next? (Maryville University) Retrieved June 21, 2024, from Maryville University: https://online.maryville.edu/blog/evolution-social-media.
13) Mehta, I. (June 27, 2023). Twitter now allows subscribers to post 25,000-character-long tweets. (Tech Crunch Yahoo News) Retrieved June 10, 2024, from Yahoo News: https://uk.news.yahoo.com/twitter-now-allows-subscribers-post-115307993.html.
14) Moghadas, M., Fekete, A., Rajabifard, A., & Kotter, T. (2023). The wisdom of crowds for improved disaster resilience: a near-real-time analysis of crowdsourced social media data on the 2021 flood in Germany. GeoJournal, 88, 4215-4241, Available:

https://doi.org/10.1007/s10708-023-10858-x.

15) O'Brien, C. (December 28, 2023). How to Use Hashtags Effectively on Social Media. (Digital Marketing Institute) Retrieved June 10, 2024, from Digital Marketing Institute: https://digitalmarketinginstitute.com/blog/how-to-use-hashtags-in-social-media.
16) Population Media Center. (2024). 8 Billion Opportunities. (Population 8 Billion) Retrieved June 20, 2024, from https://www.populationmedia.org/population-8-billion.
17) Portal Management. (2022). 13 positive effects that social networks have on today's society. (Portal Management) Retrieved June 10, 2024, from https://www.portalmanagement.ro/efecte-pozitive-retelele-socializare-asupra-societatii-actuale.
18) Reveiu, A., & Arghir, D.-C. (2020). Mining Social Media to Identify the Immediate Impact of Covid-19 Pandemic on the Romanian Retailers: Early findings. 2020 Basiq International Conference: New Trends in Sustainable Business and Consumption (pg. 1225-1232). Messina: ASE.
19) Robertson, C.T. (October 25, 2023). Here’s what our research says about news audiences on Twitter, the platform now known as X. Retrieved Jul 17, 2024, from Reuters Institute: https://reutersinstitute.politics.ox.ac.uk/news/heres-what-our-research-says-about-news-audiences-twitter-platform-now-known-x.
20) Sagl, G., & Resch, B. (2015). Mobile Phones as Ubiquitous Social and Environmental Geo-Sensors. În Encyclopedia of Mobile Phone Behavior (pg. 1194-1213). IGI Global. Available: https://doi.org/10.4018/978-1-4666-8239-9.ch098.
21) scikit-learn developers. (2024) [1]. Logistic Regression. (scikit-learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
22) scikit learn developers. (2024) [2]. LogisticRegressionCV. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html.
23) scikit learn developers . (2014) [3]. Stochastic Gradient Descent. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/0.15/modules/sgd.html.
24) scikit learn developers . (2024) [4]. LinearSVC. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.
25) scikit learn developers . (2024) [5]. RandomForestClassifier. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.
26) scikit learn developers. (2024) [6]. GradientBoostingClassifier. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.
27) scikit learn developers . (2024) [7]. DecisionTreeClassifier. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.
28) scikit learn developers. (2024) [8]. KNeighborsClassifier. (scikit learn developers) Retrieved May 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.
29) scikit learn developers. (2024) [9]. BernoulliNB. (scikit learn developers) Retrieved 24, 2024, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html.
30) Sigalo, N., & Frias-Martinez, V. (2023). Using COVID-19 Vaccine Attitudes Found in Tweets to Predict Vaccine Perceptions in Traditional Surveys: Infodemiology Study. JMIR Infodemiology, 3(1), 1-15, Available: https://doi.org/10.2196/43700.
31) Siqing, S., & Feng, Z. (2023). Social media-based urban disaster recovery and resilience analysis of the Henan deluge. Natural Hazards, 118, 377-405. Available: https://doi.org/10.1007/s11069-023-06010-0.
32) Stepanenko, V., & Liubko, I. (2020). Kaggle - Disaster Tweets Dataset. Retrieved May 10, 2024 from https://doi.org/10.34740/KAGGLE/DSV/1640141.
33) Tapia, F., Mora, M., Fuertes, W., H, A., E, F., & T, T. (2020). From Monolithic Systems to Microservices: A Comparative Study of Performance. Applied Sciences-Basel, 1-35, Available: https://doi.org/10.3390/app10175797.
34) Tjaden, J., & Tjaden, B. (2023). MLpronto: A tool for democratizing machine learning. PLOS ONE, 18(11), 1-12. Available: https://doi.org/10.1371/journal.pone.0294924.
35) Uchida, O., Kosugi, M., Endo, G., Funayama, T., Utsu, K., Tajima, S., . . . Yamamoto, Y. (2016). A Real-Time Information Sharing System to Support Self-, Mutual-, and Public-Help in the Aftermath of a Disaster Utilizing Twitter.

IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, E99A(8), 1551-1554, Available:
https://doi.org/10.1587/transfun.E99.A.1551.

36) Wang, B., Loo, B. P., Zhen, F., & Xi, G. (2020). Urban resilience from the lens of social media data: Responses to urban flooding in Nanjing, China. Cities, 106, 1-13, Available:
https://doi.org/10.1016/j.cities.2020.102884.
37) Weiss, S. M., Indurkhya, N., Zhang, T., & Damerau, F. J. (2005). Text Mining - Predictive Methods for Analyzing Unstructured Information. Sydney: Springer.
38) X Developer Platform. (2024) [1]. Search Tweets. (X Developer Platform) Retrived June 25, 2024, from X Documentation:
https://developer.x.com/en/docs/twitter-api/tweets/search/introduction.
39) X Developer Platform. (2024) [2]. Twitter API v2 data dictionary. (X Developer Platform) Retrived June 25, 2024, from X Documentation:
https://developer.x.com/en/docs/twitter-api/data-dictionary/object-model/tweet.
40) Yigitcanlar, T., Regona, M., Kankanamge, N., Mehmood, R., D'Costa, J., Lindsay, S., . . . Brhane, A. (2022). Detecting Natural Hazard-Related Disaster Impacts with Social Media Analytics: The Case of Australian States and Territories. Sustainability, 14(2), 1-23, Available:
https://doi.org/10.3390/su14020810.
41) Zisserman, A. (2015). Lecture 2: The SVM classifier. Retrived May 30, 2024, from
https://www.robots.ox.ac.uk/~az/lectures/ml.