IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 16 October 2023, accepted 11 December 2023, date of publication 13 December 2023,
date of current version 19 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3342472

RESEARCH ARTICLE

Interpretable Classification of Wiki-Review Streams

SILVIA GARCÍA-MÉNDEZ[1], FÁTIMA LEAL[2], BENEDITA MALHEIRO[3,4], AND JUAN CARLOS BURGUILLO-RIAL[1]

[1] Information Technologies Group, atlanTTic, University of Vigo, 36310 Vigo, Spain
[2] Research on Economics, Management and Information Technologies, Universidade Portucalense, 4200-072 Porto, Portugal
[3] ISEP, Polytechnic of Porto, 4249-015 Porto, Portugal
[4] Institute for Systems and Computer Engineering, Technology and Science, 4200-465 Porto, Portugal

Corresponding author: Silvia García-Méndez (sgarcia@gti.uvigo.es)

This work was supported in part by Xunta de Galicia, Spain, under Grant ED481B-2021-118; and in part by the Portuguese National Funds through FCT–Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) under Project UIDB/50014/2020.

ABSTRACT Wiki articles are created and maintained by a crowd of editors, producing a continuous stream of reviews. Reviews can take the form of additions, reverts, or both. This crowdsourcing model is exposed to manipulation since neither reviews nor editors are automatically screened and purged. To protect articles against vandalism or damage, the stream of reviews can be mined to classify reviews and profile editors in real-time. The goal of this work is to anticipate and explain which reviews to revert. This way, editors are informed why their edits will be reverted. The proposed method employs stream-based processing, updating the profiling and classification models on each incoming event. The profiling uses side and content-based features employing Natural Language Processing, and editor profiles are incrementally updated based on their reviews. Since the proposed method relies on self-explainable classification algorithms, it is possible to understand why a review has been classified as a revert or a non-revert. In addition, this work contributes an algorithm for generating synthetic data for class balancing, making the final classification fairer. The proposed online method was tested with a real data set from Wikivoyage, which was balanced through the aforementioned synthetic data generation. The results attained near-90 % values for all evaluation metrics (accuracy, precision, recall, and *F*-measure).

INDEX TERMS Data reliability and fairness, data-stream processing and classification, synthetic data, transparency, vandalism, wikis.

I. INTRODUCTION

Wiki-based platforms, like Wikipedia,[1] WikiVoyage[2] or WikiNews[3] are collaboratively maintained by voluntary editors who share their wisdom about a topic, entity, or city. When editors create and refine wiki articles, they generate a continuous stream of events indistinctly referred to as edits or reviews. Specifically, wiki editors can add, edit, and revert reviews. As such, wikis are modern-day oracles maintained by and for the crowd, simultaneously empowering and impacting it.

Moreover, this information-gathering model, known as crowdsourcing, accumulates the digital legacy of the crowd, allowing the scrutiny of interested parties. In this respect, wikis, discussion forums, blogs, and social networks can be mined to profile editors with the help of Artificial Intelligence (AI) [1], [2]. This activity is essential in crowdsourcing platforms since crowdsourced data are unmediated by default, exposing platforms to social manipulation. Examples of social manipulation include fake information in social networks, biased feedback in evaluation-based platforms, and undesired content in Wiki articles. Such adverse contributions

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Magno.

[1] Available at https://en.wikipedia.org, December 2023.
[2] Available at https://www.wikivoyage.org, December 2023.
[3] Available at https://www.wikinews.org, December 2023.

may damage brands, products, services, and articles, affecting the overall reliability of the targeted platforms.

One of the most popular techniques for spreading disinformation online is through standalone or coordinated brigades of false data generator bots. In the case of Wikipedia, sockpuppets – individuals who create multiple online identities to increase their influence in online communities – constitute a severe problem [3], [4]. To mitigate misinformation, wikis rely on highly ranked editors – administrators – to patrol the contents. Interestingly, these threats to the reliability of wikis can only be offset by instantly reverting damaging reviews and swiftly outcasting unreliable editors [5]. By classifying reviews and editors in real-time, the current work aims to address misinformation and reliability simultaneously.

This work proposes an interpretable classification solution to recognize in real-time which reviews to revert. Therefore, this paper contributes with real-time transparent identification of deceitful wiki reviews and editors. By anticipating the reversion of undesirable reviews, this early discarding of reviews has a positive impact on both the quality and reliability of wiki data. The proposed method employs stream-based processing, updating the profiling and classification models on each incoming event. The profiling uses side and content-based features employing Natural Language Processing (NLP). Since the proposed method relies on self-explainable classification algorithms (*e.g.*, decision trees), it is possible to understand why a review has been classified as a revert or a non-revert.

In addition, this paper contributes with a data synthetic generation algorithm for class balancing, aiming to make the final classification fairer. In fact, synthetic data generation has been reported to be highly beneficial [6], [7] in (*i*) testing stochastic and multispectral scenarios, (*ii*) creating relevant scenarios absent in real data, (*iii*) automatically labeling entries, (*iv*) overcoming data restrictions and making the process more affordable, (*v*) protecting sensible information, and (*vi*) speeding up data analytic processes. However, it comes with relevant constraints that must be taken into account, such as (*i*) the complexity for specific data scenarios, (*ii*) bias and outliers that can be reflected from real data, (*iii*) the dependent quality on the data source, and (*iv*) laborious and time-consuming validation and evaluation against the original data. The proposed synthetic data generation module seeks to take advantage of the first three and the last benefits while aiming to address all four constraints pointed out in the literature.

The experiments were conducted with a real data set collected from Wikivoyage with 285 698 reviews, including 8305 reverts, and 70 260 editors. Despite the original imbalanced class distribution, the proposed method presents macro and micro class classification metrics near-90 %.

The rest of this paper is organized as follows. Section II overviews the relevant related work concerning profiling, classification, transparency, and fairness of wiki data and states the current contribution. Section III introduces the proposed method, detailing the offline and stream processing. Section IV describes the experimental set-up and presents the empirical evaluation results considering the online revert classification and explanation. Finally, Section V concludes and highlights the achievements and future work.

II. RELATED WORK

In wiki-based platforms, problems such as transparency, fairness, and real-time modeling still need to be explored [8].

A. PROFILING

Wiki profiling methods model editors through their interactions within the platform. In addition, in stream-based modeling, profiles are continuously updated and refined. Based on the contents of crowdsourced data, the literature contemplates multiple types of wiki profiling methods:

Graph embedding profiling is an unsupervised learning technique representing the learned graph nodes through low-dimensional vectors. Reference [9] created and represented profiles based on side features via graph embedding to detect unbiased vandalism.

Stylometric profiles are based on textual patterns of style, *i.e.*, rely on the contents. Reference [10] built stylometric profiles to detect vandalism in Wikipedia articles. Reference [11] used standard stylometric metrics (*e.g.*, digit *n*-gram frequency, word *n*-gram frequency, *etc.*) to identify the authorship in collaborative documents. Reference [12] identified style patterns using artificial neural networks to generate the linguistic model that represents a text.

Trust & reputation profiling represents the reliability of wiki editors. By definition, while trust is based on one-to-one relationships, reputation considers third-party experiences. Trust-based models are popular among wikis. References [13] and [14] proposed TrustWiki to establish the trustworthiness of wiki reviews based on the social context of editors. Hence, TrustWiki creates clusters of editors, using content and demographic features, and presents the reader with content from similar groups of editors. Reference [15] implemented WikiTrust, which highlights trustworthy and untrustworthy words in wiki articles with different background colors. As such, WikiTrust explores content and side features. To prevent malicious and unreliable users, [16] adopted SigmoRep to compute the reputation of editors in collaborative environments from side features. Reference [17] used WikiTrust side and content-based features to recognize the authorship of crowdsourced content.

The literature shows several wiki editor and review profiling approaches that explore content, side, and social features. Besides, most surveyed works implemented offline processing. The only exception is the stream-based quality and popularity profiling proposed by [18], which enables model updating in real-time, along with the user profiling work of [19] that identifies benign and malign human and non-human (bots) contributors. Since the classification problem in the latter work is different, the content of the review is not considered.

B. ANALYSIS OF REVIEWS

According to the literature, the classification of wiki edits encompasses the detection of paid [20], puffery [21], reverted [22], [23], [24], toxic [25], [26] and vandal [9], [10], [12], [13], [17], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38] reviews. Similarly, prediction focuses on review quality [39], [40], [41] as well as on editor and article quality [18], [42], [43], [44], [45].

1) VANDALISM DETECTION

Vandalism and error detection methods explore side and content-based features to identify unethical behaviors and unintentional errors. Specifically, unethical behaviors are practiced mainly by unregistered editors [46]. In this context, the literature provides several vandalism detection methods employing distinct profiling and classification approaches.

Profiles use, separately or in combination, side and content-based features of the contributions. The Random Forest classifier [47] holds the best results and the highest popularity. Except for the detectors by [34] and [36], the remaining works implement offline processing.

In addition to the features extracted by the described profiling methods, several vandalism detection solutions rely on scores from the Objective Revision Evaluation Service (ORES), a public Application Programming Interface (API) for wiki platforms [48]. ORES is a Machine Learning (ML) system that predicts the quality of edits and article drafts in real-time. Regarding edits, ORES predicts the probability of being done in good faith, damaging, and reverted in the future. In the case of article drafts, ORES returns the probability of being spam, vandalism, an attack, and OK. These scores are used as input features by many of the surveyed works, *e.g.*, [9], [34], [36], [37], and [41], to classify reviews. Moreover, ORES is currently used on wiki platforms to help volunteers reduce the burden of manually screening content.

2) REVERT DETECTION

On wikis, reverting consists of completely removing a previous edit. Although it is a means to eliminate involuntary errors and malicious reviews, scant research is dedicated to revert classification:

- Reference [22] designed a textual analysis algorithm to detect reverts and the corresponding target reviews. The algorithm analyses editor actions considering the inserted and deleted words. The unchanged paragraphs are removed, and the insert and delete actions are analyzed using text difference methods. The computational cost of this solution is the main drawback.
- Reference [23] proposed a content-agnostic, metadata-driven classification to detect Wikipedia reverts. The profiling model is based on the editor roles defined by Wikipedia and side-based information. The authors use a Support Vector Machine classifier.

All surveyed revert identification approaches implemented offline processing.

3) QUALITY PREDICTION

Wikis, while unmediated collaborative environments, suffer from data quality and trustworthiness issues. To address this problem, predictive models can be used to anticipate the quality of individual wiki reviews, editors, and articles. To predict the quality of reviews, [39] employed orthographic similarity of lexical units to predict the quality of new reviews. Using side-based and stylometric profiles, the solution applies a deep neural network to extract quality indicators. Reference [41] shared an annotated data set of English Wikipedia articles based on Wikipedia templates, *e.g.*, original research, contradictory, unreliable sources, *etc*. The data set was used to predict the content reliability using Logistic Regression [49], Random Forest, and Gradient Boosted Trees [50]. Except for the work by [18], all surveyed quality prediction works implemented offline techniques.

C. TRANSPARENCY

Interpretability and explainability are essential for users to understand ML-generated models and outputs, improving the experience and developing trust. While interpretability is defined as the ability to describe an ML model, explainability is the interface that enables the user to understand the model [51], [52]. ML models can be divided into interpretable and opaque. Opaque models behave as black boxes (*e.g.*, artificial neural networks, or matrix factorization), whereas interpretable ones are self-explainable (*e.g.*, decision trees, rules, or regression). A transparent model – interpretable or explainable – enhances decision-making and contributes to responsible ML.

Frequently, the ML algorithms used to mitigate the negative impact of malicious behaviors in wikis are opaque. To address this problem, several research works attempted to explain the operation of vandalism detectors [38], [53], classifiers [54], or profiling [55], [56]. These explainability efforts take the form of the following:

Graph-based explanations rely on a knowledge representation graph to explain a context. References [55] and [56] represented and explained the relationship between wiki entities through graph-based profiling.

Model agnostic explanations rely on surrogate models to explain the outcomes of opaque ML models. Model agnostic interpretability techniques include the Local Interpretable Model-agnostic Explanations (LIME) [57], the Shapley Additive Explanations (SHAP) [58] and, more recently, the local trust-based explanation plugin (Explug) advanced by [59]. Considering wiki contents, [60] compared multiple explanation models, including LIME and SHAP.

Word embedding explanations are based on text processing, namely NLP. In this context, [54] combined word embedding with LIME to assess the reliability of a toxic comment classifier while [61] integrated a word embedding technique with graph-based explanations.

Visual explanations adopt non-textual formats easily interpretable by users. References [38], [53] explained vandalism detection visually. While [38] correlated inputs and outputs with the parameter spaces based on the edit frequency and reverts, [53] analyzed statistically vandal behaviors and displayed the results graphically.

D. FAIRNESS

Fairness embraces equal opportunity against biased algorithms or data to avoid prejudicial or unethical results. A fair model ensures that data biases do not affect its performance. While accuracy evaluates the performance of an ML model, fairness indicates the practical implications of deploying the model in the real world. Therefore, the selection of ML predictive algorithms must be guided not only by performance but also by fairness.

Wikis tend to discriminate against unregistered or new editors regarding vandalism detection [46], [62], article ranking [63] or reverts based on the contents of talk pages [64]. These issues have been addressed through class balancing, a pre-processing technique that balances the number of samples across classes. Regarding wiki vandalism detection, [27] performed random over-sampling to address the imbalanced class problem, whereas [35] re-sampled wiki data with Local Neighbourhood Synthetic Minority Over-sampling Technique (SMOTE).

E. RESEARCH CONTRIBUTION

The literature on real-time vandalism detection and revert classification reveals a lack of fundamental studies on vandal behavior [65]. Currently, misinformation and quality control in wiki platforms are addressed by a group of dedicated voluntary users named *patrollers*[4] together with ORES. When compared to ORES, the proposed method classifies incoming edits and explains the verdict on a stream basis. Moreover, given its modular design, it enables the integration of the latest technological advances, such as using Large Language Models (LLMs) to engineer new features for effective classification and better explainability. Such optional improvements in detection accuracy will lead to greater computational load.

Consequently, this work contributes to mitigating the vulnerabilities of the crowdsourcing model through real-time classification and explanation of the content posted in wikis. The designed pipeline, which applies existing AI methods, constitutes an original wiki vandalism detection method. Specifically, it employs:

- Standard feature analysis, engineering, and selection techniques to ensure the high quality of the experimental data and take advantage of the full potential of the classification models.
- Online editor profiling to capture the expected profile evolution with time.
- Synthetic data generation techniques to create artificial samples and improve the fairness of the final classification.
- Stream-based ML classification and explainability to detect and justify which reviews to revert in real-time.

In summary, this work contributes with (*i*) a stream-based method that, unlike existing solutions, analyses and exploits textual content for classification purposes; (*ii*) generation of synthetic data to perform stochastic and multispectral tests; and (*iii*) visual and natural language explanations of the classifications. Furthermore, the solution was validated with a comprehensive set of experiments, including ad hoc tests, to determine its performance in offline scenarios with different partition sets of the experimental data.

III. PROPOSED METHOD

This paper proposes an explainable and fair method to identify which wiki reviews will be reverted. Figure 1 introduces the proposed multi-stage solution, which adopts offline synthetic wiki data generation for class balancing followed by stream-based classification of wiki reviews. The offline stage encompasses (*i*) data pre-processing (Section III-A1) based on feature-target pairwise correlation (Section III-A1a), feature engineering (Section III-A1b) and selection (Section III-A1c), and (*ii*) synthetic data generation (Section III-A2) to balance the data set classes. The online stage performs (*i*) incremental profiling (Section III-B1), (*ii*) stream-based classification (Section III-B2) evaluated through standard classification metrics, and (*iii*) outcome explanation (Section III-B3) supported by model interpretability.

A. OFFLINE PROCESSING

Offline processing comprises mainly data pre-processing techniques and synthetic data generation. However, the former is a three-phase stage. Algorithm 1 overviews the offline processing method.

1) DATA PRE-PROCESSING

Data pre-processing translates raw data into features usable by ML classifiers and selects the most promising independent features to predict the target feature. The new

[4] Available at https://en.wikipedia.org/wiki/Wikipedia:Patrolling, December 2023.

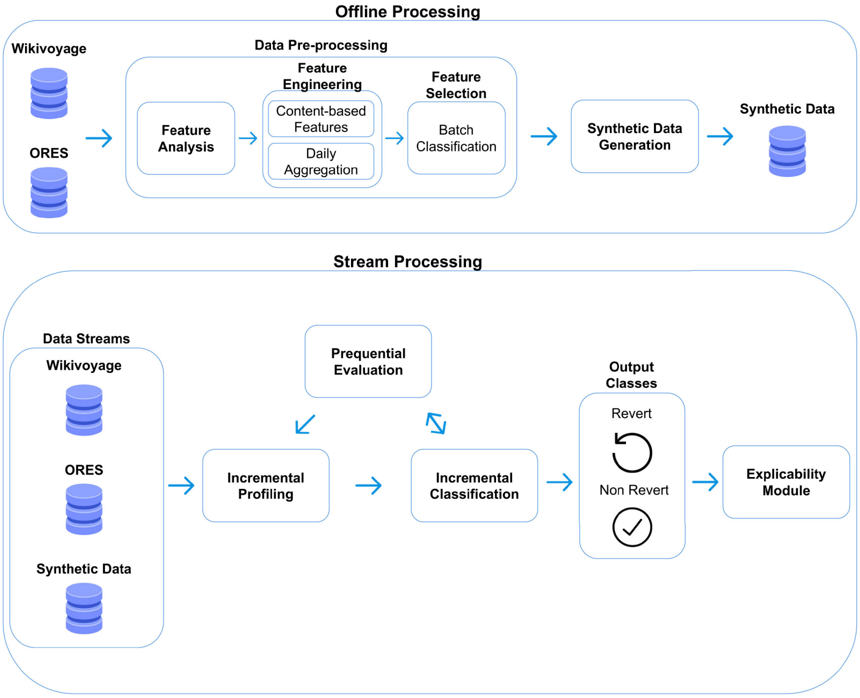

FIGURE 1. Fair and transparent classification of Wikivoyage reviews as reverts and non-reverts.

Algorithm 1 Offline Processing Algorithmic Description

```
function offline_processing(dataset)
    // Feature analysis
    Spearman_coefficients = compute_Spearman(dataset);
    // Feature engineering
    for sample in dataset do
        sample.get_text().process();    // The specific text processing techniques will be detailed in Section IV-B1b
        sample.get_text().compute_features();    // The specific features engineered will be detailed in Section IV-B1b
    end for
    // Feature selection
    selected_features = meta_transformer_wrapper(dataset, RF, configuration_parameters);
    // Synthetic data generation
    synthetic_data_generation(); // Detailed in Algorithm 2.
end function
```

computed features relevant to the revert prediction task are employed to model editors and reviews. Specifically, data pre-processing is a three-phase stage composed of (*i*) feature analysis, (*ii*) feature engineering, and (*iii*) feature selection tasks. First, feature analysis performs an in-depth screening of the features highly correlated with the target variable. Then, feature engineering enables valuable data generation for classification. Finally, feature selection takes the most relevant correlated features identified in feature analysis and those created during feature engineering. Moreover, the three data pre-processing techniques are applied offline before stream-based classification, as illustrated in Figure 1.

a: FEATURE ANALYSIS

The statistical dependence between the rankings of the independent features and the target feature is computed using Spearman's rank correlation coefficient [66] as shown in Equation (1), where x and y are the rank variables and n represents the sample size.

$$r_s = \frac{n\sum xy - \sum x \sum y}{\sqrt{n\sum x^2 - (\sum x)^2}\sqrt{n\sum y^2 - (\sum y)^2}} \qquad (1)$$

This non-parametric measure of rank correlation assesses monotonic (linear or not) relationships among continuous and discrete features. Spearman correlation values range from −1 to +1, with the limits corresponding to the case when each feature is a perfect monotone function of the other.

b: FEATURE ENGINEERING

This stage produces new side and content-derived features about articles, editors, and reviews to improve the classification of reviews as reverts and non-reverts. Side features characterize the properties of an entity (*e.g.*, size of the revision or the number of links in the review). These features allow us to characterize the type of review performed by the editors (*e.g.*, a large revision may indicate either a thorough correction, an excess of unnecessary changes, or the addition of spam/vandalism content). In contrast, content-derived features result from the analysis of the introduced or deleted text to provide ML models with in-depth knowledge of the content of the reviews.

Numeric features contain average values regarding editor revisions per article, revisions per week, articles revised per week, article quality probabilities from ORES, review size, number of links, bad words, and number of inserted and deleted characters. Categorical features identify the creator of the revision, whether the editor is a bot, and hold the polarity of inserted and deleted text. In the end, textual features represent the cumulative characters and word *n*-grams of the inserted and deleted text. The generation of these features will be further explained in Section IV-B1b.

Finally, the original data set consisting of individual timestamped reviews and related features is transformed into daily reviews and associated features per editor. The remaining stages explore these editor daily activity features instead of the original individual features.

c: FEATURE SELECTION

Feature selection is performed through a meta-transformer wrapper method. Mainly, a meta-transformer method can be used with any estimator for feature selection, while a wrapper method allows the exploitation of an underlying ML model for the feature importance computation [19]. It wraps the classification algorithm – Random Forest (RF) classifier – and selects features based on importance weights. The algorithm establishes relative feature importance using a forest of trees to find meaningful features and, thus, reduce the feature space dimension. The features are considered irrelevant if the corresponding importance of the feature values is below the specified threshold. The resulting profile feature vector comprises:

- Side features related to editors;
- ORES probabilities related to articles and reviews;
- Side and content-derived features related to reviews.

2) SYNTHETIC DATA GENERATION

Synthetic data generation allows testing ML models in a fully stochastic and multispectral scenario, including significant layouts absent from real data. The proposed synthetic data generation module is mainly used to balance the experimental data set.

More in detail, the designed data generation method produces feasible incremental samples of the editor's daily activity concerning the revert category. This process allows us to balance the experimental data set. Note that only reverted entries are created since they are less represented. The created artificial samples include the features listed in Table 3, except for the char and word *n*-grams. The latter are randomly selected by using a clustering procedure. To maintain the inter-feature correlation (see Section IV-B2), the data generated for each feature is based on its statistical measures (quartile distribution, median, minimum and maximum values) considering four intervals: (*i*) from minimum to the first quartile (Q1); (*ii*) from Q1 to median; (*iii*) from median to third quartile (Q3); (*iv*) from Q3 to the maximum. Algorithm 2 summarizes the synthetic data generation procedure.

For example, to generate a Q1 value for numeric feature f, it retrieves all samples from the original data set with an f value between its Q1 and median. Then, it generates the remaining features based on the above subset with random values between Q1 (minimum) and Q3 (maximum). The latter avoids the generation of outliers (caused by applying cluster-based filtering and having a minimal subset) and enhances the dispersion of the synthetically generated data. Once all non-cumulative numeric features of the synthetic feature vector are generated, the values of cumulative features are randomly selected from a hash map that holds all possible values for those features in the subset. The date is randomly selected within the period of the experimental data set to prevent revert and non-revert samples from grouping and, thus, obtaining an unrealistic time distribution.

B. STREAM PROCESSING

Stream processing is a three-phase stage that involves incremental profiling, classification, and the generation of explainable descriptions regarding the predictions. Algorithm 3 overviews this process.

1) INCREMENTAL PROFILING

The incremental profiling models the online evolution of the editor's daily activity through the selected feature vectors from the offline data pre-processing stage. Editor profiles are incrementally updated using the balanced data set as a data stream. The built profiles encompass side features (*e.g.*, number of inserted or deleted characters) and content-derived ones (*e.g.*, the polarity, bad words, or *n*-grams of reviews). Several numeric features store cumulative averages and sums.

Algorithm 2 It Creates *count* Revert Samples Using Cluster-Based Filtering and Maintaining Inter-Feature Correlation

```
function synthetic_data_generation($min, Q1, median, Q3, max, count$)
    $ranges = \{min, Q1, median, Q3, max\}$; // Quartile distribution
    $synthetic\_data = []$; // Synthetic samples
    for $r \in ranges - 1$ do
        for $i \in count/4$ do
            $synthetic\_entry = []$; // Synthetic sample
            for $f \in features$ do // Table 3 features
                $Q1 = Kmeans[f, r, r + 1]$
                $Q3 = Kmeans[f, r, r + 1]$
                //Random sample from Q1 to Q3 obtained by the $\boldsymbol{K}$-means
                $synthetic\_entry[f] = random(Q1, Q3)$
            end for
            $synthetic\_data.append(synthetic\_entry)$
        end for
    end for
    $synthetic\_data.sortByTimestamp()$ // Returns the synthetic samples
end function
```

Algorithm 3 Stream Processing Algorithmic Description

```
function stream_processing($dataset$)
     // Incremental profiling
    dataset = compute_average_values(dataset);
    dataset = compute_cumulative_values(dataset);
     // Incremental classification
    ml_models = load_ml_models();
    for model in ml_models do
        for sample in dataset do
            predict(model, sample);
            evaluate(model, sample);
            train(model, sample);
            print(model.evaluation_metrics());
        end for
    end for
     // Explainability
    explainability_graph = RF.compute_graph(dataset.get_random_sample());
    visualize(explainability_graph);
    nl_description = apply_template(dataset.get_random_sample());
    print(nl_description);
end function
```

2) INCREMENTAL CLASSIFICATION

This work comprises both batch and stream-based ML classification. The batch experiments select the most promising features and classification algorithm (baseline results), whereas the stream-based experiment explores the batch findings online. The binary classification algorithms selected are well-known interpretable models with promising performance [9], [67], [68], [69]:

- Naive Bayes (NB) is a simple probabilistic classifier based on Bayes' theorem [70].
- Ridge Classifier (RC) exploits Ridge regression by converting the target feature values into {-1, 1} [71].
- Decision Tree (DT) is a discrete-target predictive model based on traversing a tree structure from observation branches (conjunctions of features) to a target class label leaf [72].
- Random Forest (RF) is an ensemble learning model based on multiple DT classifiers [47].
- Boosting Classifier (BC) is an ensemble of weak predictive models that allows the optimization of a differentiable loss function [73].

The model evaluation relies on standard metrics: classification accuracy, precision, recall, and F-measure in macro and micro-averaging computing scenarios. Furthermore, macro results enable comprehensive evaluation considering the whole set of target classes, while micro-averaging considers the target classes individually. The latter is beneficial in imbalanced classification problems [74], [75], [76]. Finally, run-time is measured to compare the performance of the different models.

3) EXPLAINABILITY

The proposed method relies on interpretable classifiers to explain classification outcomes. Being self-explainable, interpretable models can make their reasoning explicit, offering insight into the classification process. Decision rules, decision trees, Naive Bayes, and logistic regression are examples of interpretable binary classification algorithms.

In the current case, the selected classification algorithms are interpretable and, thus, can explain why a review has been classified as a revert or a non-revert. This explainability may rely on graph-based, natural language, visual, or hybrid formats to present the user with the reasons learned by the classifier, *e.g.*, the relevant DT path, decision rules, or the impact of the different features on the outcome.

IV. EXPERIMENTAL RESULTS

All experiments were performed using a server with the following hardware specifications:

- Operating System: Ubuntu 18.04.2 LTS 64 bits
- Processor: IntelCore i9-9900K 3.60 GHz
- RAM: 32 GB DDR4
- Disk: 500 GB (7200 rpm SATA) + 256 GB SSD

The experiments comprise (*i*) offline feature analysis, engineering, and selection, (*ii*) offline synthetic data generation for class balancing, (*iii*) incremental profiling, online classification with a balanced data stream and prediction explanation.

A. DATA SET

The data collection[5] relied on a well-known set of utilities for extracting and processing MediaWiki[6] data in Python. To retrieve the contents of the reviews, direct GET requests were issued to the Wikivoyage endpoint[7] using the *compare* action functionality. The data span from 1st January 2004 to 31st December 2019 contains 285 698 samples from 70 260 editors regarding 3369 different articles. Considering the target feature, the data is deeply imbalanced with a total of 8305 reverted reviews (0.03 % of the samples).

B. OFFLINE PROCESSING

As previously mentioned, offline processing comprises several relevant tasks: (*i*) offline feature analysis, engineering and selection; and (*ii*) offline synthetic data generation for class balancing.

TABLE 1. Wikivoyage data set transformation.

Data set	Contents	Total
Original (imbalanced)	Articles	3369
	Editors	70 260
	Reviews / editor:	285 698
	Non-reverts	277 393
	Reverts	8305
Daily original[8] (imbalanced)	Articles	3369
	Editors	70 260
	Daily reviews / editor:	45 353
	Non-reverts	41 996
	Reverts	3357
Daily synthetic[9] (imbalanced)	Articles	3369
	Editors	70 260
	Daily reviews / editor:	40 000
	Non-reverts	0
	Reverts	40 000
Daily combined (balanced)	Articles	3369
	Editors	70 260
	Daily reviews / editor:	85 353
	Non-reverts	41 996
	Reverts	43 357

1) DATA PRE-PROCESSING

The data pre-processing starts with the statistical dependence analysis described in Section III-A1a over the rankings of the numeric features listed in Table 2. These exclude identifier features 1 to 5 and textual features 11 and 12. The correlation results were computed using Spearman's rank correlation coefficient. Finally, it performs feature engineering and selection.

a: FEATURE ANALYSIS

Table 2 presents the independent (1 to 22) and target (23) features considered for the classification of reviews as reverts and non-reverts. The correlation between the independent and the target features in Table 2 is moderate and can be grouped into:

- **Features with negative correlation**: from 6 to 8, 14, 19 (false damaging & true good faith probabilities), 20 (E probability), 21 (OK probability), 22 (star & stub probabilities).
- **Features with positive correlation**: 9, 10, 13, from 15 to 19 (true damaging & false good faith probabilities), 20 (except E probability), 21 (except OK probability), 22 (except star & stub probabilities).

b: FEATURE ENGINEERING

The contents of the revisions are processed with the English Natural Language Processing pipeline optimized for CPU, named `en_core_web_lg`[10] provided by the spaCy library.[11] This processing removes URL instances and special

[5] Data and code will be available from the corresponding author on reasonable request.

[6] Available at https://pypi.org/project/mediawiki-utilities, December 2023.

[7] Available at https://en.wikivoyage.org/w/api.php, December 2023.

[8] It groups editor activity per day.

[9] It creates new synthetic samples per editor and day (only for imbalanced revert activity).

[10] Available at https://spacy.io/models/en#en_core_web_lg, December 2023.

[11] Available at https://spacy.io, December 2023.

TABLE 2. Features considered for the classification and target feature.

Number	Feature name
1	Date
2	Review ID
3	Editor name and ID
4	Creator name and ID
5	Article name
6	Bot flag
7	Editor is the creator of the article
8	Size of the revision
9	Number of links in the review
10	Number of repeated links in the review
11	Inserted characters
12	Deleted characters
13	Amount of inserted characters
14	Amount of deleted characters
15	Amount of common reverted words added
16	Amount of bad words added
17	Polarity of inserted characters
18	Polarity of deleted characters
19	ORES edit quality probability: false/true damaging & good faith
20	ORES item quality probability: A/B/C/D/E
21	ORES article quality probability: OK/attack/spam/vandalism
22	ORES article quality probability (wp10): B/C/FA/GA/start/stub
23	Revert flag

characters like accents. Then, it lemmatizes the resulting text and removes stop words.[12] The polarity of the revisions, considering added and deleted characters, is computed with the `spaCyTextBlob`[13] pipeline that performs sentiment analysis using the `TextBlob` library.[14] To determine the number of common words reverted and bad words in revisions, it reuses the corresponding lists of words provided by Wikimedia Meta-wiki.[15] The char and word *n*-grams are extracted from the accumulated textual data (see Section III-B1) with the help of the `CountVectorizer`[16] Python library. Based on performance tests, the configuration parameters were set to `max_df_in=0.7`, `min_df_in=0.001`, `wordgram_range_in=(1,4)`, `chargram_range_in=(1,4)`, `max_features_in=None`.

Finally, it aggregates the individual reviews and associated features into daily reviews and associated features per editor, removing the hour and minute from the date. The remaining stages explore these daily features.

c: FEATURE SELECTION

The `SelectFromModel`[17] feature selection algorithm wraps the RF classifier to identify the higher-importance features. The configuration parameters were set to `n_estimators=500`, `n_jobs=-1`, `random_state=0`. These experiments use a reduced balanced subset comprising 3357 revert and 4000 non-revert samples and 10-fold cross-validation [77] to avoid over-fitting, biased, or over-estimated values. Table 3 lists the independent features selected for classifying reviews as reverts and non-reverts (features from 1 to 18 correspond to side features and features 19 and 20 to content features). To establish the best set of features, the batch classifier explored the three sets identified in Table 3:

TABLE 3. Independent features selected for the classification.

			Number	Feature name
Set C	Set B	Set A	1	Bot flag
			2	Editor is the creator of the article
			3	Average number of revisions per article
			4	Average number of revisions per week
			5	Average number of articles revised per week
			6	Average ORES edit quality probability
			7	Average ORES item quality probability
			8	Average ORES article quality probability
			9	Average ORES article quality probability (wp10)
			10	Average size of the revision
			11	Average number of links in the revision
			12	Average number of repeated links in the revision
			13	Average number of common reverted words
			14	Average number of bad words
			15	Average number of inserted characters
			16	Average number of deleted characters
			17	Average polarity of inserted characters
			18	Average polarity of deleted characters
			19	Cumulative char and word n-grams of inserted text
			20	Cumulative char and word n-grams of deleted text

A) Side features related to editors;
B) ORES probabilities related to articles and reviews plus A features;
C) Content-derived features related to reviews plus B features.

Ultimately, only some char and word *n*-grams features were discarded.

Finally, Table 4 lists the results of the four offline ML classifiers with these sets of features using 10-fold cross-validation. The non-revert and revert classes correspond to #0 and #1, respectively. Set A of features report near 60 % - 70 % accuracy, precision, macro recall, and macro and non-revert *F*-measure values. Even though non-revert values for recall are promising for the NB and RC classifiers, revert values for recall and *F*-measure are significantly low. The results with set B (comprising set A and ORES probabilities) are generally better for all classifiers and metrics. Results improve further with set C, attaining near 80 % with RF and BC in all metrics. Set C comprises side features related to editors, ORES features related to articles, plus content-derived features related to reviews. The best classifier considering all metrics is RF.

2) SYNTHETIC DATA GENERATION

The quality of the synthetic data was determined by statistically comparing the synthetic against the original daily data. Table 5 displays the results, excluding the features without statistical variation (1, 2, 14 in Table 3).

Specifically, it shows the relative change in percentage between the original and synthetic data related to the first,

[12] Available at https://gist.github.com/sebleier/554280, December 2023.

[13] Available at https://spacy.io/universe/project/spacy-textblob, December 2023.

[14] Available at https://github.com/sloria/TextBlob, December 2023.

[15] Available at https://meta.wikimedia.org/wiki/Research:Revision_scoring_as_a_service/Word_lists/en, December 2023.

[16] Available at https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html, December 2023.

[17] Available at https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html, December 2023.

TABLE 4. Offline revert classification results with a balanced data set of 7357 original samples using 90 % for training and 10 % for test (10-fold cross-validation).

Set	Classifier	Accuracy	Precision			Recall			*F*-measure			Time
			Macro	#0	#1	Macro	#0	#1	Macro	#0	#1	(s)
A	NB	0.60	0.67	0.58	0.77	0.56	0.96	0.17	0.50	0.72	0.27	0.02
	RC	0.61	0.69	0.59	0.78	0.58	0.95	0.20	0.53	0.73	0.32	0.04
	DT	0.58	0.57	0.61	0.54	0.57	0.63	0.51	0.57	0.62	0.52	0.15
	RF	0.59	0.59	0.63	0.56	0.59	0.63	0.55	0.59	0.63	0.55	2.09
	BC	0.64	0.64	0.64	0.64	0.62	0.77	0.47	0.62	0.70	0.54	3.51
B	NB	0.65	0.69	0.62	0.77	0.62	0.92	0.33	0.60	0.74	0.46	0.02
	RC	0.65	0.67	0.63	0.71	0.63	0.86	0.40	0.62	0.73	0.51	0.05
	DT	0.67	0.67	0.72	0.62	0.67	0.64	0.70	0.67	0.68	0.66	0.68
	RF	0.74	0.73	0.76	0.71	0.74	0.75	0.72	0.73	0.75	0.71	2.54
	BC	0.69	0.69	0.69	0.68	0.68	0.77	0.58	0.68	0.73	0.63	16.65
C	NB	0.59	0.72	0.57	0.86	0.55	0.98	0.11	0.46	0.72	0.20	0.90
	RC	0.76	0.77	0.73	0.80	0.75	0.87	0.62	0.75	0.79	0.70	2.52
	DT	0.73	0.74	0.78	0.69	0.74	0.71	0.77	0.73	0.74	0.73	14.09
	RF	**0.83**	**0.84**	**0.82**	**0.85**	**0.83**	**0.88**	**0.77**	**0.83**	**0.85**	**0.81**	7.58
	BC	0.83	0.83	0.82	0.84	**0.83**	0.88	0.77	0.83	0.85	0.81	307.20

TABLE 5. Relative change (%) between the synthetic and original daily data related to first, second, and third quartiles.

	3	4	5	6			
				damaging false	damaging true	goodfaith false	goodfaith true
Q1	18.61	19.33	24.81	2.74	3.21	0.12	0.15
Q2	20.63	11.41	9.51	0.01	0.06	0.06	0.06
Q3	17.99	20.11	13.44	3.26	2.80	0.15	0.12
	7					8	
	A	B	C	D	E	OK	attack
Q1	0.01	0.02	0.23	0.61	0.79	1.34	0.24
Q2	0.00	0.00	0.01	0.07	0.08	0.00	0.05
Q3	0.01	0.01	0.22	0.59	0.88	1.77	0.23
	8		9				
	spam	vandalism	WP10B	WP10C	WP10FA	WP10GA	WP10Start
Q1	1.17	1.09	2.08	0.44	0.16	0.03	1.12
Q2	0.03	0.33	1.71	0.29	0.15	0.02	0.06
Q3	1.26	0.85	2.63	0.48	0.24	0.03	1.19
	9	10	11	12	13	15	16
	WP10Stub						
Q1	3.03	0.00	21.75	23.75	0.00	17.77	15.37
Q2	1.03	0.00	14.43	31.60	37.89	6.56	10.70
Q3	2.68	0.01	21.05	27.55	24.15	15.27	28.93
	17	18					
Q1	2.00	3.19					
Q2	1.23	1.17					
Q3	2.25	3.45					

second, and third quartiles of the synthetically generated samples. The minimal statistical variations observed in most features result from the synthetic data generation algorithm maintaining the inter-feature correlation. The exceptions are the features that do not represent probabilistic values (3 to 5 and 11 to 16). After adding the 40 000 synthetic to the original daily feature vectors, the final distribution of classes in the resulting balanced data set is 43 357 revert and 41 996 non-revert daily feature vectors.

C. STREAM PROCESSING

Stream processing takes advantage of the insights obtained during the batch experiments to select the most promising features and classification algorithms. Mainly, the stream-based experiment explores the batch findings online. Moreover, graph and natural language descriptions of the model predictions are provided.

1) INCREMENTAL PROFILING

In the stream processing mode, the profiles of the editors are updated with each new daily feature vector (see Table 3), depending on the type of feature:

- Static features (1 and 2) remain unchanged since they correspond to identifiers;
- Average value features (3 to 18) update their contents to the new incremental average;

- Cumulative value features (19 and 20) update their contents to the new incremental sum.

2) INCREMENTAL CLASSIFICATION

The following binary classification algorithms were selected from scikit-learn[18] and scikit-multiflow[19] and applied without hyper-parameter optimization.

- NB[20]
- RC.[21]
- DT[22]
- RF[23]
- BC[24]

The following stream-based classification experiments were performed exclusively with the best classifier – the RF model – and best set of features – set C. These experiments apply the `EvaluatePrequential`[25] algorithm that uses each incoming sample first to test and evaluate and, finally, to train the model.

The final classification experiments compare online and offline performance with the balanced data (85 353 samples) ordered chronologically. The online models are built from scratch and incrementally updated and evaluated, whereas the offline models are trained and then tested using distinct data partitions. The first set of experiments compares the classification results of the last 90 % of the data for the online and the best offline model (trained with the first 10 % of the data). The second set of experiments compares the results of the last 10 % of the data for the online and the best offline model (trained with the first 90 % of the data). Table 6 displays these classification results. In both cases, the best offline results are obtained with set C features and decision tree classifiers (DT in the first case and RF in the second case). In the first set of experiments, the revert class (#1) presents an online performance of near-90 %, 69 percent points than the offline baseline in the recall metric, whereas, in the second set of experiments, the revert class (#1) presents an online performance of near-100 %, 3 percent points than the offline baseline in the recall metric. As expected, stream-based outperforms batch classification.

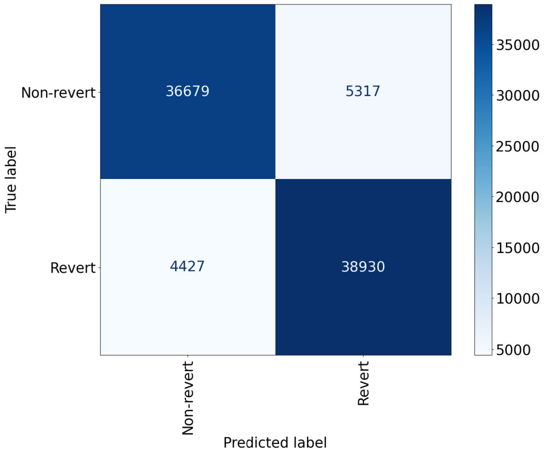

FIGURE 2. Confusion matrix of the online RC classifier with balanced data.

Figure 2 displays the confusion matrix, showing the impact of false positives and negatives on the classification results.[26] Accordingly, the vast majority of the samples were correctly classified as they concentrated on the first diagonal of the matrix. Even though the model's performance is comparable in non-revert and revert detection tasks, the prediction error is slightly superior for the non-revert class. This means our solution is conservative when identifying editions to be reverted.

Finally, these results are better than those found in the literature, *e.g.*, 7, 16, 12 percent points in precision, recall, and *F*-measure when compared with the closely related offline revert identification by [23]. They explore an original data set from Simple English Wikipedia[27] composed of 3.1 million edits, 240 000 articles and 175 000 users. The experiments, performed with the Weka suite[28] and 10-fold cross-validation, use a subset of 825 000 edits (750 000 for training and 75 000 for test), ignoring article contents and relying on a Support Vector Machine classifier. Our comparable results, obtained with approximately one-tenth of the samples, are better than the proposed method.

3) EXPLAINABILITY

The RF classifier provides the best results and builds and explores decision trees, which are interpretable models. These explanations cover the relevant subset of branches from root to classification leaf and materialize as the corresponding sub-graph and/or subset of learned rules.

Since the implemented online RF model uses ten estimators, the first step is to select the smallest decision path leading to the classification of a given sample. The next step

[18] Available at https://scikit-learn.org/stable, December 2023.

[19] Available at https://scikit-multiflow.readthedocs.io/en/stable/api/api.html, December 2023.

[20] Available at https://scikit-learn.org/stable/modules/naive_bayes.html, December 2023.

[21] Available at https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html, December 2023.

[22] Available at https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, December 2023.

[23] Available at https://scikit-learn.org/stable/modules/tree.html and https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.meta.AdaptiveRandomForestClassifier.html#skmultiflow.meta.AdaptiveRandomForestClassifier, December 2023.

[24] Available at https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, December 2023.

[25] Available at https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.evaluation.EvaluatePrequential.html, December 2023.

[26] It corresponds to the online classification within the first set of experiments.

[27] Available at https://simple.wikipedia.org/wiki/Main_Page, December 2023.

[28] Available at https://www.cs.waikato.ac.nz/ml/weka, December 2023.

TABLE 6. Online versus best offline revert classification results with balanced data.

Processing	Train/Update	Test/Evaluate	Classifier	Accuracy	Precision			Recall			*F*-measure			Time
					Macro	#0	#1	Macro	#0	#1	Macro	#0	#1	(s)
Offline	First 10 %	Last 90 %	DT	0.48	0.53	0.45	0.60	0.52	0.83	0.21	0.45	0.59	0.31	1.96
Online	100 %	Last 90 %	RF	**0.89**	**0.89**	**0.89**	**0.88**	**0.89**	**0.87**	**0.90**	**0.89**	**0.88**	**0.89**	1868.38
Offline	First 90 %	Last 10 %	RF	**0.96**	0.96	0.93	**0.99**	**0.96**	**0.99**	0.93	**0.96**	**0.96**	**0.96**	10.40
Online	100 %	Last 10 %	RF	**0.96**	**0.97**	**0.96**	0.97	**0.96**	0.97	**0.96**	**0.96**	**0.96**	**0.96**	1772.77

```
For sample 1, the model decision is based on the following facts:
 The average number of repeated links < 0.03
 Average ORES article quality probability - WP10GAAvg > 0.01
 Average ORES article quality probability - WP10FAAvg > 0.01
 Average ORES edit quality probability - damagingTrueAvg < 0.19
 Average ORES item quality probability - EAvg < 0.96
 Predicted class non-revert
For sample 2, the model decision is based on the following facts:
 Average ORES edit quality probability - damagingTrueAvg > 0.19
 Average ORES item quality probability - CAvg < 0.02
 The revision contains ['wiki']
 Predicted class revert
For sample 3, the model decision is based on the following facts:
 The average number of repeated links > 0.01
 Average ORES article quality probability - WP10BAvg < 0.22
 The revision contains ['speciality', 'barbeque']
 Predicted class revert
For sample 4, the model decision is based on the following facts:
 The average number of repeated links < 0.01
 The revision contains ['jpg', 'wikidata', 'pgname', 'long']
 Predicted class non-revert
```

LISTING 1. Natural language explanations built from the RF classifier.

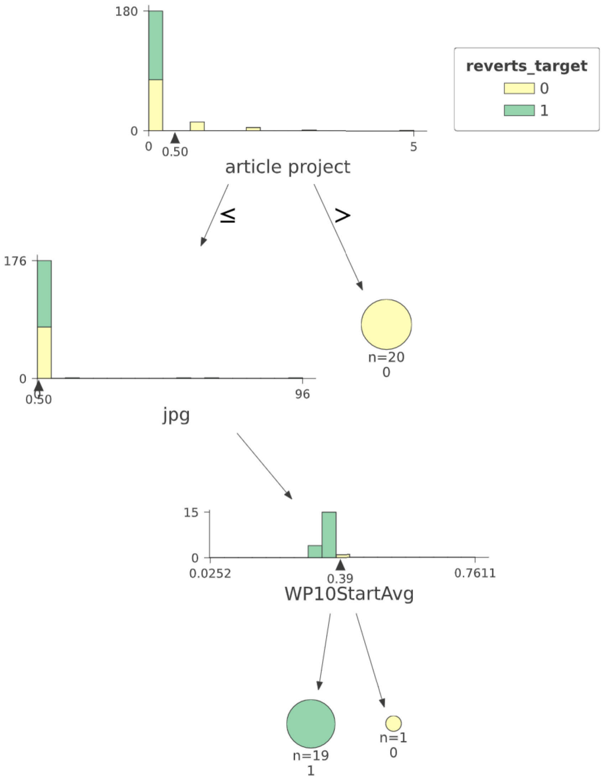

FIGURE 3. Enhanced graph-based explanations built from the RF classifier. The non-revert and revert classes correspond to 0 and 1, respectively.

uses the `get_model_description`[29] method from scikit-multiflow to traverse the selected decision tree and create the templates to display this knowledge in natural language. Listing 1 provides four natural language explanations, two for each class (revert and non-revert), detailing the model decisions based on side and content-derived features as well as the predicted class (0 represents the non-revert class and 1, the revert class). The features correspond to Table 3.

Figure 3 displays a partial view of the decision tree learned and used by the RF algorithm in three cases. It depicts the revert and non-revert leaves using different colors (green for reverts and yellow for non-reverts) and styles and was obtained using `dtreeviz`[30] library. More in detail, the first decision is based on the bigram `article project` (feature 19 in Table 3). If the frequency of the bigram in the revision is superior to 0.5, the reasoning continues through the right branch (predicted class non-revert); otherwise, it goes to the left. Using the `plot_tree` library[31] we can obtain the `gini` index (see Figure 4). The latter coefficient expresses the probability of an incorrect prediction. For the first bifurcation, the `gini` index reported was close to 0.5. In the latter case, if the frequency of the unigram `jpg` is superior to 0.5, the model checks if the average ORES article quality probability (feature 9 in Table 3 – `WP10StartAvg`) is higher than 0.386. If this test succeeds, the sample is classified as non-revert. Otherwise, it is considered a revert. As the decision

[29] Available at https://scikit-multiflow.readthedocs.io/en/stable/api/generated/skmultiflow.trees.HoeffdingTreeClassifier.html, December 2023.

[30] Available at https://pypi.org/project/dtreeviz, December 2023.

[31] Available at https://scikit-learn.org/stable/modules/generated/sklearn.tree.plot_tree.html, December 2023.

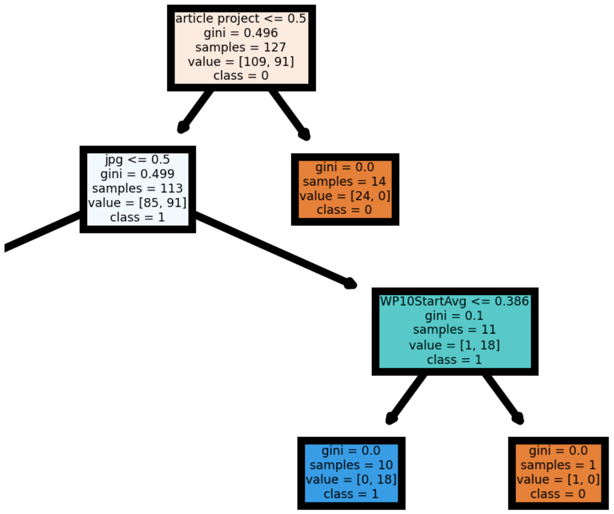

FIGURE 4. Graph-based explanations built from the RF classifier. The non-revert and revert classes correspond to 0 and 1, respectively.

tree is traversed, the gini index reduces, reaching 0.1 in the last bifurcation for this example.

Both explanations presented – natural language and graph-based – rely on the underlying model's interpretability and make the classifier's reasoning transparent and understandable for the user.

V. CONCLUSION

Wiki platforms like Wikivoyage display articles created and maintained by a community of volunteer editors. This crowdsourcing model based on free content and open collaboration is vulnerable to unethical behavior, raising concerns about data quality in wiki repositories. One of the most critical issues behind this social manipulation is disinformation.

In light of the above, this work proposes a transparent, fair method to identify which wiki reviews to revert. Notably, the designed solution includes (*i*) offline synthetic wiki data generation to ensure model fairness against class imbalance, (*ii*) stream-based classification of wiki reviews supported by side and content-based features, and (*iii*) interpretable ML models to explain to editors why their reviews were classified as reverts.

The stream-based experiments were performed with a balanced data set with 43 357 revert and 41 996 non-revert reviews made by 70 260 editors to 3369 articles, resulting from the combination of collected and synthetic Wikivoyage data. The proposed method was evaluated using standard classification metrics and attained near-90 % in all classification metrics considered (precision, recall, *F*-measure). This result shows that it is possible to explain and predict in real-time whether a review will be reverted and use this information to take preemptive actions to protect the quality of wiki articles and to help well-intentioned users improve their reviews.

The results obtained with the proposed method will immediately positively impact wiki users, who will enjoy more reliable content thanks to real-time review classification. The crowd volunteers will also benefit from explaining review classifications and reducing the editorial screening burden.

The challenges of integrating the proposed method into wiki platforms include (*i*) labeling large volumes of data, (*ii*) ensuring the privacy of sensitive data, and (*iii*) assessing the trust of editors. These may be addressed by exploring LLMs for data management, ad hoc algorithms for sensible data detection and removal, and reinforcement learning to model editor trust.

The future work plan will also explore the current result to automatically revert the identified reviews, outcast their authors, and evolve from the current mixed offline and online processing to a fully online processing pipeline, combined with hyper-parameter optimization, for further improvement.

REFERENCES

[1] F. Alattar and K. Shaalan, "Using artificial intelligence to understand what causes sentiment changes on social media," *IEEE Access*, vol. 9, pp. 61756–61767, 2021.

[2] W. Jiang and Y. Sun, "Social-RippleNet: Jointly modeling of ripple net and social information for recommendation," *Int. J. Speech Technol.*, vol. 53, no. 3, pp. 3472–3487, Feb. 2023.

[3] C.-B. Zhang, N. Li, S.-H. Han, Y.-D. Zhang, and R.-J. Hou, "How to alleviate social loafing in online brand communities: The roles of community support and commitment," *Electron. Commerce Res. Appl.*, vol. 47, May 2021, Art. no. 101051.

[4] M. Tajrian, A. Rahman, M. A. Kabir, and M. R. Islam, "A review of methodologies for fake news analysis," *IEEE Access*, vol. 11, pp. 73879–73893, 2023.

[5] V. Lageard and C. Paternotte, "Trolls, bans and reverts: Simulating Wikipedia," *Synthese*, vol. 198, no. 1, pp. 451–470, Jan. 2021.

[6] M. Mukherjee and M. Khushi, "SMOTE-ENC: A novel SMOTE-based method to generate synthetic data for nominal and continuous features," *Appl. Syst. Innov.*, vol. 4, no. 1, pp. 18–29, Mar. 2021.

[7] F. K. Dankar, M. K. Ibrahim, and L. Ismail, "A multi-dimensional evaluation of synthetic data generators," *IEEE Access*, vol. 10, pp. 11147–11158, 2022.

[8] F. Leal, B. Veloso, B. Malheiro, H. Gonzalez-Velez, and J. C. Burguillo, "A 2020 perspective on 'scalable modelling and recommendation using wiki-based crowdsourced repositories': Fairness, scalability, and real-time recommendation," *Electron. Commerce Res. Appl.*, vol. 40, pp. 100951–100952, Mar. 2020.

[9] S. Heindorf, Y. Scholten, G. Engels, and M. Potthast, "Debiasing vandalism detection models at Wikidata," in *Proc. World Wide Web Conf.*, May 2019, pp. 670–680.

[10] M. Harpalani, M. Hart, S. Singh, R. Johnson, and Y. Choi, "Language of vandalism: Improving Wikipedia vandalism detection via stylometric analysis," in *Proc. Annu. Meeting Assoc. Comput. Linguistics, Human Lang. Technol.*, 2011, pp. 83–88.

[11] E. Dauber, R. Overdorf, and R. Greenstadt, "Stylometric authorship attribution of collaborative documents," in *Proc. Cyber Secur. Cryptography Mach. Learn. Conf.* Cham, Switzerland: Springer, 2017, pp. 115–135.

[12] J. R. Martinez-Rico, J. Martinez-Romo, and L. Araujo, "Can deep learning techniques improve classification performance of vandalism detection in Wikipedia?" *Eng. Appl. Artif. Intell.*, vol. 78, pp. 248–259, Feb. 2019.

[13] H. Zhao, W. Kallander, T. Gbedema, H. Johnson, and F. Wu, "Read what you trust: An open Wiki model enhanced by social context," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust, IEEE 3rd Int. Conf. Social Comput.*, Oct. 2011, pp. 370–379.

[14] H. Zhao, W. Kallander, H. Johnson, and S. F. Wu, "SmartWiki: A reliable and conflict-refrained Wiki model based on reader differentiation and social context analysis," *Knowl.-Based Syst.*, vol. 47, pp. 53–64, Jul. 2013.

[15] B. T. Adler, "WikiTrust: Content-driven reputation for the Wikipedia," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. California, CA, USA, 2012.

[16] A. A. Kardan, R. Salarmehr, and A. Farshad, "SigmoRep: A robust reputation model for open collaborative environments," in *Proc. IEEE 13th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Sep. 2014, pp. 505–510.

[17] P. P. Paul, M. Sultana, S. A. Matei, and M. Gavrilova, "Editing behavior to recognize authors of crowdsourced content," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 1676–1681.

[18] F. Leal, B. M. Veloso, B. Malheiro, H. González-Vélez, and J. C. Burguillo, "Scalable modelling and recommendation using wiki-based crowdsourced repositories," *Electron. Commerce Res. Appl.*, vol. 33, Jan. 2019, Art. no. 100817.

[19] S. García-Méndez, F. Leal, B. Malheiro, J. C. Burguillo-Rial, B. Veloso, A. E. Chis, and H. González–Vélez, "Simulation, modelling and classification of Wiki contributors: Spotting the good, the bad, and the ugly," *Simul. Model. Pract. Theory*, vol. 120, Nov. 2022, Art. no. 102616.

[20] N. Joshi, F. Spezzano, M. Green, and E. Hill, "Detecting undisclosed paid editing in Wikipedia," in *Proc. Web Conf.*, Apr. 2020, pp. 2899–2905.

[21] A. Bertsch and S. Bethard, "Detection of puffery on the English Wikipedia," in *Proc. 7th Workshop Noisy User-Generated Text (W-NUT)*, 2021, pp. 329–333.

[22] F. Flöck, D. Vrandecic, and E. Simperl, "Revisiting reverts: Accurate revert detection in Wikipedia," in *Proc. 23rd ACM Conf. Hypertext Social Media*, Jun. 2012, pp. 3–12.

[23] J. Segall and R. Greenstadt, "The illiterate editor: Metadata-driven revert detection in Wikipedia," in *Proc. Int. Symp. Open Collaboration*. New York, NY, USA: Association for Computing Machinery, 2013, pp. 1–8.

[24] J. Kiesel, M. Potthast, M. Hagen, and B. Stein, "Spatio-temporal analysis of reverted Wikipedia edits," in *Proc. Int. AAAI Conf. Web Social Media*, May 2017, vol. 11, no. 1, pp. 122–131.

[25] M. Ibrahim, M. Torki, and N. El-Makky, "Imbalanced toxic comments classification using data augmentation and deep learning," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 875–878.

[26] N. Chakrabarty, "A machine learning approach to comment toxicity classification," in *Proc. Comput. Intell. Pattern Recognit. Conf.* Cham, Switzerland: Springer, 2020, pp. 183–193.

[27] M. Potthast, B. Stein, and R. Gerling, "Automatic vandalism detection in Wikipedia," in *Proc. Adv. Inf. Retr. Conf.* Cham, Switzerland: Springer, 2008, pp. 663–668.

[28] B. T. Adler, L. de Alfaro, S. M. Mola-Velasco, P. Rosso, and A. G. West, "Wikipedia vandalism detection: Combining natural language, metadata, and reputation features," in *Proc. Comput. Linguistics Intell. Text Process. Conf.* Cham, Switzerland: Springer, 2011, pp. 277–288.

[29] S. Javanmardi, D. W. McDonald, and C. V. Lopes, "Vandalism detection in Wikipedia: A high-performing, feature-rich model and its reduction through lasso," in *Proc. 7th Int. Symp. Wikis Open Collaboration*, Oct. 2011, pp. 82–90.

[30] S. M. Mola-Velasco, "Wikipedia vandalism detection," in *Proc. 20th Int. Conf. Companion World Wide Web*, Mar. 2011, pp. 391–396.

[31] E. Alfonseca, G. Garrido, J.-Y. Delort, and A. Peñas, "WHAD: Wikipedia historical attributes data," *Lang. Resour. Eval.*, vol. 47, no. 4, pp. 1163–1190, Dec. 2013.

[32] K.-N. Tran and P. Christen, "Cross language prediction of vandalism on Wikipedia using article views and revisions," in *Proc. Adv. Knowl. Discovery Data Mining Conf.* Cham, Switzerland: Springer, 2013, pp. 268–279.

[33] S. Kumar, F. Spezzano, and V. S. Subrahmanian, "VEWS: A Wikipedia vandal early warning system," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 607–616.

[34] S. Heindorf, M. Potthast, B. Stein, and G. Engels, "Vandalism detection in Wikidata," in *Proc. 25th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2016, pp. 327–336.

[35] M. Shulhan and D. H. Widyantoro, "Detecting vandalism on English Wikipedia using LNSMOTE resampling and cascaded random forest classifier," in *Proc. Int. Conf. Adv. Inform., Concepts, Theory Appl. (ICAICTA)*, Aug. 2016, pp. 1–6.

[36] S. Heindorf, M. Potthast, G. Engels, and B. Stein, "Overview of the Wikidata vandalism detection task at WSDM cup 2017," in *Proc. Web Search Data Mining Cup*, 2017, pp. 1–9.

[37] A. Sarabadani, A. Halfaker, and D. Taraborelli, "Building automated vandalism detection tools for Wikidata," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 1647–1654.

[38] Z. Liu and A. Lu, "Explainable visualization for interactive exploration of CNN on Wikipedia vandal detection," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2019, pp. 2354–2363.

[39] S. Sarkar, B. P. Reddy, S. Sikdar, and A. Mukherjee, "StRE: Self attentive edit quality prediction in Wikipedia," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 3962–3972.

[40] S. Asthana, S. T. Thommel, A. L. Halfaker, and N. Banovic, "Automatically labeling low quality content on Wikipedia by leveraging patterns in editing behaviors," in *Proc. ACM Human-Comput. Interact. Conf.*, vol. 5. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1–23.

[41] K. Wong, M. Redi, and D. Saez-Trumper, "Wiki-reliability: A large scale dataset for content reliability on Wikipedia," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 2437–2442.

[42] T. Ruprechter, T. Santos, and D. Helic, "Relating Wikipedia article quality to edit behavior and link structure," *Appl. Netw. Sci.*, vol. 5, no. 1, pp. 61–80, Dec. 2020.

[43] Q.-V. Dang and C.-L. Ignat, "Measuring quality of collaboratively edited documents: The case of Wikipedia," in *Proc. IEEE 2nd Int. Conf. Collaboration Internet Comput. (CIC)*, Nov. 2016, pp. 266–275.

[44] Q.-V. Dang and C.-L. Ignat, "An end-to-end learning solution for assessing the quality of Wikipedia articles," in *Proc. 13th Int. Symp. Open Collaboration*, Aug. 2017, pp. 1–10.

[45] W. Lewoniewski and K. Wecel, "Relative quality assessment of Wikipedia articles in different languages using synthetic measure," in *Proc. Bus. Inf. Syst. Workshops*. Cham, Switzerland: Springer, 2017, pp. 282–292.

[46] A. Alkharashi and J. Jose, "Vandalism on collaborative web communities: An exploration of editorial behaviour in Wikipedia," in *Proc. Spanish Conf. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–4.

[47] A. Parmar, R. Katariya, and V. Patel, "A review on random forest: An ensemble classifier," in *Proc. Int. Conf. Intell. Data Commun. Technol. Internet Things*. Cham, Switzerland: Springer, 2019, pp. 758–763.

[48] A. Halfaker and R. S. Geiger, "ORES: Lowering barriers with participatory machine learning in Wikipedia," in *Proc. ACM Human-Comput. Interact. Conf.*, vol. 4. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–37.

[49] B. B. Frey, *Multiple Logistic Regression*. New York, NY, USA: SAGE, 2023.

[50] M. Mistry, D. Letsios, G. Krennrich, R. M. Lee, and R. Misener, "Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded," *Informs J. Comput.*, vol. 33, no. 3, pp. 1103–1119, Jul. 2021.

[51] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.

[52] I. Mollas, N. Bassiliades, and G. Tsoumakas, "LioNets: A neural-specific local interpretation technique exploiting penultimate layer information," *Int. J. Speech Technol.*, vol. 53, no. 3, pp. 2538–2563, Feb. 2023.

[53] S. S. Subramanian, P. Pushparaj, Z. Liu, and A. Lu, "Explainable visualization of collaborative vandal behaviors in Wikipedia," in *Proc. IEEE Symp. Visualizat. Cyber Secur. (VizSec)*, Oct. 2019, pp. 1–5.

[54] A. Mahajan, D. Shah, and G. Jafar, "Explainable AI approach towards toxic comment classification," in *Proc. Emerg. Technol. Data Mining Inf. Secur. Conf.* Cham, Switzerland: Springer, 2021, pp. 849–858.

[55] M. K. Sarker, J. Schwartz, P. Hitzler, L. Zhou, S. Nadella, B. Minnery, I. Juvina, M. L. Raymer, and W. R. Aue, "Wikipedia knowledge graph for explainable AI," in *Proc. Knowl. Graphs Semantic Web Conf.* Cham, Switzerland: Springer, 2020, pp. 72–87.

[56] N. Klein, F. Ilievski, and P. Szekely, "Generating explainable abstractions for Wikidata entities," in *Proc. 11th Knowl. Capture Conf.*, Dec. 2021, pp. 89–96.

[57] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1135–1144.

[58] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Int. Conf. Neural Inf. Process. Syst.* New York, NY, USA: Curran Associates, 2017, pp. 4768–4777.

[59] F. Leal, S. Garcia-Mendez, B. Malheiro, and J. C. Burguillo, "Explanation plug-in for stream-based collaborative filtering," in *Proc. World Conf. Inf. Syst. Technol.* (Lecture Notes in Networks and Systems), vol. 468, 2022, pp. 42–51.

[60] J. Risch, R. Ruff, and R. Krestel, "Offensive language detection explained," in *Proc. Workshop Trolling, Aggression Cyberbullying*, vol. 34, 2020, pp. 29–47.

[61] M. A. Qureshi and D. Greene, "EVE: Explainable vector based embedding technique using Wikipedia," *J. Intell. Inf. Syst.*, vol. 53, no. 1, pp. 137–165, Aug. 2019.

[62] Z. Ye, X. Yuan, S. Gaur, A. Halfaker, J. Forlizzi, and H. Zhu, "Wikipedia ORES explorer: Visualizing trade-offs for designing applications with machine learning API," in *Proc. Designing Interact. Syst. Conf.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 1554–1565.

[63] D. Lewandowski and U. Spree, "Ranking of Wikipedia articles in search engines revisited: Fair ranking for reasonable quality?" *J. Amer. Soc. Inf. Sci. Technol.*, vol. 62, no. 1, pp. 117–132, Jan. 2011.

[64] S. Ross, "Your day in 'Wiki-Court': ADR, fairness, and justice in Wikipedia's global community," *Osgoode Legal Stud. Res. Paper*, vol. 10, pp. 1–21, Mar. 2014.

[65] J. Tramullas, P. Garrido-Picazo, and A. I. Sanchez-Casabon, "Research on Wikipedia vandalism: A brief literature review," in *Proc. Spanish Conf. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–4.

[66] J. C. F. de Winter, S. D. Gosling, and J. Potter, "Comparing the Pearson and Spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data," *Psychol. Methods*, vol. 21, no. 3, pp. 273–290, Sep. 2016.

[67] J.-X. Mi, A.-D. Li, and L.-F. Zhou, "Review study of interpretation methods for future interpretable machine learning," *IEEE Access*, vol. 8, pp. 191969–191985, 2020.

[68] R. Haffar, D. Sánchez, and J. Domingo-Ferrer, "Explaining predictions and attacks in federated learning via random forests," *Int. J. Speech Technol.*, vol. 53, no. 1, pp. 169–185, Jan. 2023.

[69] M. M. Hossin, F. M. J. M. Shamrat, M. R. Bhuiyan, R. A. Hira, T. Khan, and S. Molla, "Breast cancer detection: An effective comparison of different machine learning algorithms on the Wisconsin dataset," *Bull. Electr. Eng. Informat.*, vol. 12, no. 4, pp. 2446–2456, Aug. 2023.

[70] D. Berrar, "Bayes' theorem and naive Bayes classifier," in *Encyclopedia of Bioinformatics and Computational Biology*, vol. 3. Amsterdam, The Netherlands: Elsevier, 2019, pp. 403–412.

[71] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, pp. 160–180, May 2021.

[72] A. Trabelsi, Z. Elouedi, and E. Lefevre, "Decision tree classifiers for evidential attribute values and class labels," *Fuzzy Sets Syst.*, vol. 366, pp. 46–62, Jul. 2019.

[73] C. Bentéjac, A. Csörgo, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, Mar. 2021.

[74] M. Okkalioglu and B. D. Okkalioglu, "AFE-MERT: Imbalanced text classification with abstract feature extraction," *Int. J. Speech Technol.*, vol. 52, no. 9, pp. 10352–10368, Jul. 2022.

[75] A. Hafeez, T. Ali, A. Nawaz, S. U. Rehman, A. I. Mudasir, A. A. Alsulami, and A. Alqahtani, "Addressing imbalance problem for multi label classification of scholarly articles," *IEEE Access*, vol. 11, pp. 74500–74516, 2023.

[76] A. Vanacore, M. S. Pellegrino, and A. Ciardiello, "Evaluating classifier predictive performance in multi-class problems with balanced and imbalanced data sets," *Qual. Rel. Eng. Int.*, vol. 39, no. 2, pp. 651–669, Mar. 2023.

[77] D. Berrar, "Cross-validation," in *Encyclopedia of Bioinformatics and Computational Biology*. Cambridge, MA, USA: Academic Press, 2019, pp. 542–545.

SILVIA GARCÍA-MÉNDEZ received the Ph.D. degree in information and communication technologies from the University of Vigo, Spain, in 2021. Since 2015, she has been a Researcher with the Information Technologies Group, University of Vigo. She is collaborating with foreign research centers as part of a postdoctoral stage. Her research interests include natural language processing techniques and machine learning algorithms.

FÁTIMA LEAL received the Ph.D. degree in information and communication technologies from the University of Vigo, Spain. She is an Assistant Professor with Universidade Portucalense, Portugal, and a Researcher with Research on Economics, Management, and Information Technologies (REMIT). Following a full-time postdoctoral fellowship funded by the European Commission, she continues collaborating with the Cloud Competency Center, National College of Ireland, Dublin. Her research is based on crowdsourced information, including trust and reputation, big data, data streams, and recommendation systems. Recently, she has been exploring blockchain technologies for responsible data processing.

BENEDITA MALHEIRO received the degree in electrical engineering and the M.Sc. and Ph.D. degrees in electrical engineering and computers from the University of Porto. She is a Coordinator Professor with Instituto Superior de Engenharia do Porto, School of Engineering, Polytechnic of Porto; and a Senior Researcher with INESC TEC, Portugal. Her research interests include artificial intelligence, computer science, and engineering education. She is a member of the Association for the Advancement of Artificial Intelligence (AAAI), the Portuguese Association for Artificial Intelligence (APPIA), the Association for Computing Machinery (ACM), and the Professional Association of Portuguese Engineers (OE).

JUAN CARLOS BURGUILLO-RIAL received the Ph.D. degree in telematics from the University of Vigo, Spain. He is currently a Full Professor with the Department of Telematic Engineering and a Researcher with the AtlanTTic Research Center in Telecommunication Technologies, University of Vigo. He has directed and participated in several research and development projects in the areas of telematics and computer science in national and international calls. His research interests include intelligent systems, evolutionary game theory, self-organization, and complex adaptive systems. He is a Regular Reviewer of several international conferences and journals, including the *Autonomous Agents and Multi-Agent Systems*, *Computers and Education*, *Engineering Applications of Artificial Intelligence*, *Computers and Mathematics with Applications*, and the *Journal of Network and Computer Applications*. He is also an Area Editor of the *Simulation Modelling Practice and Theory* (SIMPAT).