Sockpuppet Detection in Wikipedia Using Machine Learning and Voting Classifiers

Rafeef Baamer[a] and Mihai Boicu[b]
Department of Information Sciences and Technology, George Mason University, University Dr, Fairfax, VA, U.S.A.

Keywords: Sockpuppet, Machine Learning, Classifier, Random Forest, Naive Bayes, Support Vector Machine, K-Nearest Neighbour, AdaBoost, XGBoost, Logistic Regression, Voting Classifier, Soft Voting, Hard Voting.

Abstract: Sockpuppet accounts, deceptive identities created by individuals on social networks, present significant challenges to online integrity and security. In this study, we analyse various approaches for detecting Sockpuppet accounts through the computation of several distinct features and the application of seven different classifiers. To enhance detection accuracy, we employ simple majority voting to aggregate the predictions from multiple classifiers, involving all seven classifiers, or only best five or three of them. This approach allows us to leverage the strengths of different classifiers while mitigating their individual weaknesses. Our experimental results show a significant improvement over previous studies, achieving an accuracy rate of 88% with 87% precision. Additionally, our experiments highlight the critical importance of feature engineering, demonstrating how carefully selected features directly influence classification performance. The findings also emphasize the value of human-in-the-loop involvement, where iterative feedback refines the models and improves their predictive capabilities. These insights offer meaningful contributions toward strengthening the security and integrity of online social networks by enabling more accurate and robust detection of Sockpuppet accounts.

1 INTRODUCTION

With the rise of online platform usage, there has been an increase in a special type of malicious account, Sockpuppet accounts. Sockpuppet accounts can be defined as the creation of multiple accounts by an individual or a coordinated group of people for malicious or deception intents, including activities such as spamming, fake reviewing, and the dissemination of fake news and misinformation (Baamer & Boicu, 2024). These accounts are often created to manipulate opinions, spread false information, or avoid bans on primary accounts. Detecting these Sockpuppet accounts is essential to keep online interactions and content trustworthy.

In the next sections we discuss the impact of Sockpuppet accounts, formalize the research problem and briefly present the current state of the art for Sockpuppet detection, which identifies machine learning as a promising solution to this problem by detecting anomalies in user behaviour, interaction networks, and account characteristics. In this paper, we describe an experiment that studies the efficacy of various machine learning classifiers in detecting Sockpuppet accounts. Additionally, we employ a simple majority voting system among these classifiers and study if they improve overall detection performance. Our analysis shows that the proposed method achieved an accuracy of 88% with a precision of 87%. Finally, this research aims to advance the state of the art in Sockpuppet detection while offering valuable methodologies and future directions to enhance the security and reliability of online communities. A key future direction involves integrating human-in-the-loop processes for iterative refinement, which we think will improve the models' detection accuracy and adaptability.

[a] https://orcid.org/0009-0000-6611-7381
[b] https://orcid.org/0000-0002-6644-059X

Baamer, R. and Boicu, M.
Sockpuppet Detection in Wikipedia Using Machine Learning and Voting Classifiers.
DOI: 10.5220/0013210100003944

In *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security (IoTBDS 2025)*, pages 39-47
ISBN: 978-989-758-750-4; ISSN: 2184-4976

2 RESEARCH PROBLEM

Sockpuppet accounts present a complex problem for online social networks. Firstly, they undermine trust and authenticity within online communities by pretending to be real users who engage in conversations, support ideas, or endorse causes. For instance, there is a correlation between highly active users in political hashtags and those who tweet across multiple political hashtags, supporting the idea that social media activism is driven by a small, highly active group of politically engaged users (Bastos et al., 2013). This deceptive behaviour affects user trust and creates an environment where users doubt the authenticity of the content they encounter (Bhatia et al., 2023). Secondly, Sockpuppet accounts are often used to manipulate public opinion and spread false or misleading information. Also, their involvement in online discussions can distort the perceived popularity of promoted views, potentially affecting democratic processes and policy decisions (Bhopale et al., 2021). Moreover, these accounts can be employed for cyber harassment (Boididou et al., 2017), cyberbullying (Borkar et al., 2022), and even cyberattacks, posing significant risks to user safety and well-being (Boshmaf et al., 2015).

Our study aims to address several key research questions in the domain of Sockpuppet accounts detection specifically within the context of Wikipedia. Firstly, we seek to determine which machine learning classifiers perform best in identifying Sockpuppet accounts. By evaluating and comparing various classifiers using Wikipedia accounts and editing data, we aim to identify the most effective algorithms for this specific application. Secondly, we investigate whether aggregating the results of individual classifiers using a voting system improves overall classification accuracy on Wikipedia. This involves examining hard and soft voting techniques to see if combining multiple classifiers leads to better performance in the Wikipedia environment. Finally, we explore the common characteristics of Wikipedia accounts that are consistently misclassified by all classifiers. Understanding these characteristics can provide insights into the limitations of current detection methods and highlight areas for improvement.

3 RELATED WORKS

We performed an extensive literature review for Sockpuppet accounts (Baamer & Boicu, 2024) in which we identified 125 published papers and 13 other recent survey papers. Only ten papers out of 125 papers specifically addressed Sockpuppet accounts on Wikipedia, highlighting a significant gap in the literature that our study aims to fill.

Machine learning-based detection techniques are used most to identify Sockpuppet accounts (81 out of 125 papers). They extracted attributes and incorporated supervised, semi-supervised, and unsupervised models. About 25 studies used one machine learning classification method, while others applied and compared several machine learning classifiers (56 papers). However, only two studies aggregated the results of the different classifiers by using a voting system.

One of the first studies to detect Sockpuppet accounts was conducted by Solorio, Hasan, and Mizan in 2013 for Wikipedia. This study used a Support Vector Machine classifier based on 24 features (Solorio et al., 2013). On the other hand, most other studies on Wikipedia run more than one classifier and compare their results. For example, (Sakib et al., 2022) considered different classifiers, which are Logistic Regression, Gaussian Naive Bayes, Decision Tree, Multilayer Perceptron, Random Forest, ExtraTree, and a Long Short-Term Memory network. Similarly, (Yamak et al., 2016) also considered six classifiers: Support Vector Machine, Random Forest, Naive Bayes, k-Nearest Neighbors, Bayesian network, and AdaBoost. Despite this common platform (Wikipedia), the previous three studies employed different categories of features: (Sakib et al., 2022) and (Yamak et al., 2016) both used account-based, user activity and interaction-based, and temporal-based features in their analyses. In contrast, both (Yamak et al., 2016) and (Solorio et al., 2013) used content-based features. A comparison with Solorio et al. and Sakib et al. studies on Wikipedia is performed in the analysis section.

The remaining six papers focused on Wikipedia used similar methods to the previously discussed papers. For instance, one of these papers was conducted by the same authors of (Solorio et al., 2013), and they achieved similar results with their previous work (Solorio et al., 2014). Other studies, such as (Tsikerdekis et al., 2014) and (Yu et al., 2021), computed several features like verbal and non-verbal features and applied various classifiers, including Support Vector Machine, Random Forest, and AdaBoost. Tsikerdekis et al. computed several non-verbal features, such as time-independent and time-dependent features. Then, the authors applied the machine learning classifiers on different binary

models that consist of different combinations of variables to identify the differences in accuracy as time progress. On the other hand, Yu et al. extracted verbal and non-verbal features. Then, they applied multi-source feature fusion adaptive selection technique to combine the extracted features. Lastly, they applied the machine learning classifiers (SVM, RF, and AdaBoost) on these adaptive multi-source features. The results of Solorio et al., Tsikerdekis et al. and Yu et al. are shown in Table 1. Additionally, some studies employed alternative detection techniques, such as graph-based techniques (Kuruvilla et al., 2015; Yamak et al., 2018; and Liu et al., 2023) or statistical-based techniques like (He et al., 2021). These approaches do not significantly differ in outcomes or methodologies from the previously discussed studies.

Table 1: Previous Results Notations: accuracy (A), F1 score, precision (P) and recall (R). Dash indicates that the results were not available. Results are provided as A/F1/P/R.

	Solorio et al., 2014	Tsikerdekis et al., 2014	Yu et al., 2021
SVM	-/74/-/-	69/66/70/62	-/80/85/76
RF	-	71/67/75/61	-/81/80/81
AB	-	71/69/73/65	-/78/72/90

Looking at a different platform (PTT), both (Nguyen et al., 2021) and (Wang et al., 2023) further expand their research by utilizing a diverse set of classifiers. (Nguyen et al., 2021) applied Naive Bayes, XGBoost, AdaBoost, k-Nearest Neighbours, and Random Forest classifiers, while (Wang et al., 2023) utilized LightGBM, XGBoost, Random Forest, ConcNet, and Fully Connected classifiers. These two papers are notable for applying both hard and soft voting classifiers to aggregate the results of the individual classifiers, enhancing the robustness and accuracy of the Sockpuppet detection. Additionally, both studies computed several features of different categories, including content-based, sentiment-based, statistical-based, temporal-based, and user activity and interaction-based features. Also, (Nguyen et al., 2021) used social network-based features. The results of these two studies are discussed in detail in the Analysis section below.

4 EXPERIMENTS

To evaluate the validity of our assumptions, we conducted six experiments. For each experiment, we applied various machine learning classifiers to assess the performance and accuracy of our models. The key distinction between the six experiments lies in the scope of feature engineering and the number of features utilized. Through these experiments, we aimed to answer our three research questions.

For these experiments, we selected the dataset collected from Wikipedia by (Sakib et al, 2022) and publicly available on GitHub (Sakib, 2022).

4.1 Feature Engineering

We started by removing features such as 'top,' 'text hidden,' and 'comment hidden' because most of these variables contained mostly empty or null values, which could negatively impact the accuracy of the models. Additionally, these features were not useful in any of the computed features or analysis, so retaining them would not contribute meaningfully to the model's performance. The remaining raw features were user ID, username, revision ID, namespace, title, timestamp, comment, and class label.

Based on these raw features, we aggregated the entries for the same account and computed twenty-one features for each account: 1. Number of digits in the username, 2. The ratio of digits to total alphabet characters in a username, 3. Number of leading digits, 4. The unique character ratio in the username, 5. Average comment length for each user, 6. Average title length, 7. Average time difference between two consecutive edits, 8. Average number of alphabetic in comments, 9. Average number of punctuation in comments, 10. Average number of sentences in comments, 11. Average number of characters in comments, 12. Total number of revisions, 13. Number of namespaces the user participates in, 14. Number of titles the user participates in, 15. Account age, 16. The ratio of namespace contribution to total contribution, 17. The ratio of title contribution to total contribution, 18. The ratio of the number of comments to total revision per user, 19. Average number of comments per article, 20. Average user comments per day, 21. Average number of comments per active day for a user. Some of these features were inspired by previous work, which are (Ashford et al., 2020), (Nguyen et al., 2021), (Sakib et al., 2022), (Solorio et al., 2013), and (Yamak et al., 2016). However, we proposed four new features: 15, 16, 20, and 21.

In experiment 1, we used the first seven features, while in experiment 2, we used features 1-11. For experiment 3, we used features 1-15. Then, in experiments 4 and 5, we used features 1-17 and 1-21, respectively. Lastly, we ran the feature selection method provided by the Scikit-learn library, SelectKBest, and we selected the best 15 features for

experiment 6, which are features 1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, and 20. Moreover, most of the features were computed as aggregations (e.g., average, count, sum) of the raw data, providing a global view of each account. For example, features 8-11 were averaged to aggregate the contributions made by a single account. This aggregation allowed us to keep a single entry for each account.

4.2 Dataset Preparation

After aggregating the data and removing duplicates, we have refined our dataset. Despite this cleaning process, the dataset remains substantial, with approximately 116,000 unique accounts.

We then randomly split the dataset using an 80/20 rule, designating 80% of the data for training and 20% for testing. To ensure that the model is not biased, we did not use feedback from testing to improve the models. Additionally, the same training and testing datasets were used for all experiments.

4.3 Classification Models

Based on the analysis of previous research, we identified seven promising classifiers: Support Vector Machine (SVM), Random Forest (RF), Naive Bayes (NB), Logistic Regression (LR), k-Nearest Neighbours (KNN), AdaBoost (AB), and XGBoost (XGB). Most of these classifiers were implemented using their default parameters from the Scikit-learn library. However, for the Random Forest, KNN, AdaBoost, and XGBoost classifiers, we observed noticeable discrepancies between their performance on the training and testing datasets, which indicated the presence of overfitting. To address this, we utilized GridSearchCV to select the optimal parameters from a range of values, as shown in Table 2, to mitigate overfitting and improve generalization. In contrast, the remaining classifiers demonstrated comparable and reasonable performance on both training and testing datasets, with no significant signs of overfitting. Consequently, their default parameters were retained without further tuning.

For Random Forest, we modified the number of splits that each decision tree is allowed to make (max_depth) to 7, the number of trees (n_estimators) to 50, and the minimum number of samples that are required to split an internal node (min_samples_split) to 3. For KNN, we set the number of neighbours (k) to 9. For AdaBoost, we set the number of weak learner (n_estimators) to 100 and the weight applied to each regressor to each iteration (learning_rate) to be 1.0, and we used the same values of (n_estimators) and (learning_rate) for XGBoost classifiers. For other parameters used in XGBoost, we set the (max_depth) to 3 and (alpha) value to 15.

We implemented simple majority voting by utilizing both hard and soft voting classifiers with these seven models. Each voting classifier, implemented using the Scikit-learn library, was executed twice—once with the hard voting parameter and once with the soft voting parameter. Detailed explanation of these two types of voting can be found in (Scikit-Learn, 2014).

Table 2: Hyperparameter Tuning for RF and KNN.

Classifier	Hyperparameter	Values Range	Best Value
RF	max_depth	2,3,5,7,9	7
	n_estimators	50,100,200	50
	min_samples_split	2,3,4,5	3
KNN	n_neighbors	1-15	9
AB	n_estimators	50,100,200	100
	learning_rate	0.01, 0.02, 1.0	1.0
XGB	n_estimators	50,100,200	100
	max_depth	3,5,7,9	3
	learning_rate	0.01, 0.02, 1.0	1.0
	alpha	5,7,10,15	15

After analysing the aggregation of the seven classifiers' results, we implemented more voting classifiers that aggregate the results of the best classifiers with different combinations. For example, we added voting classifiers that aggregate the results of different combinations of five models and other voting classifiers of three different classifiers. The added voting classifiers are hard and soft voting on (RF, SVM, LR, AB, XGB), (RF, SVM, KNN, AB, XGB), (RF, LR, KNN, AB, XGB), (RF, LR, KNN), (SVM, RF, KNN), (SVM, RF, LR), (RF, LR, AB), (RF, KNN, AB), (RF, KNN, XGB, RF, LR, XGB), and (RF, AB, XGB). By the end of the experiments, we had 19 classifiers.

The classifiers were trained on the training data and saved using the pickle library provided by Scikit-Learn. These saved models were imported and used on testing data, which was prepared in a similar way of the training data.

4.4 Classification Results

For each classifier, we generated evaluation metrics: accuracy, F1 score, precision, and recall. The results achieved by experiment 5, which utilized all features, are shown in Table 3 and Table 4. The results of all experiments and different combination of classifiers are shown in Appendix Table A1 to Table A24.

Table 3: Experiment 5- Individual classifiers performance with 21 features; (Abbreviations: 1=Training data, 2=Testing data, DS=Data set, A=Accuracy, F1=F1-score, P= Precision, R=Recall, H=Hard, S=Soft).

	SVM		RF		LR		NB		KNN		AB		XGB	
DS	1	2	1	2	1	2	1	2	1	2	1	2	1	2
A	87	87	88	87	87	87	22	22	88	86	88	88	**89**	88
F1	83	83	84	83	83	83	20	20	86	84	86	86	**88**	**87**
P	85	85	87	86	84	84	82	82	87	83	86	86	**88**	**87**
R	87	87	88	87	87	87	22	22	88	86	88	88	**89**	88

Table 4: Experiment 5- Voting classifiers aggregating all classifiers' results with 21 features (Abbreviations: 1=Training data, 2=Testing data, DS=Data set, A=Accuracy, F1=F1-score, P= Precision, R=Recall, H=Hard, S=Soft).

	H		S	
DS	1	2	1	2
A	88	88	88	88
F1	86	85	86	85
P	87	86	87	86
R	88	88	88	88

5 ANALYSES

In the analysis of our six experiments, we observed distinct trends in the performance of the classifiers based on the number of features used. The first two experiments that utilized 7 and 11 features produced lower results than the subsequent experiments. In experiments 3 through 6, which employed 15, 17, 20, and again 15 features, respectively, the performance of all models, except NB, showed a slight but consistent improvement in each successive experiment. However, the performance of the NB model decreased dramatically in experiments 4 and 5 but then improved in experiment 6. This decrease in performance highlights the sensitivity of the NB model to the number and type of features used, contrasting with the more stable improvement observed in the other models. Additionally, it is worth pointing out that our experiments likely avoided overfitting, as we obtained similar results for both the training and testing for each model.

The voting classifiers, both hard and soft, consistently outperformed most other classifiers, showcasing their robustness in aggregating predictions to improve overall accuracy. However, AdaBoost and XGBoost classifiers showed almost equal performance to the voting classifiers but slightly better. Furthermore, attempts to improve the voting system's performance by implementing several voting classifiers that aggregate the results of different combinations of classifiers and excluding the worst-performing classifiers have results in slight enhancements. For example, aggregating the result of RF, AdaBoost and XGBoost, which are the best-performing classifiers, achieved the highest results of all the individual classifiers and voting classifier with 88% accuracy, 86% F1 score, 87% precision and 88% recall. Tables 5 and Table 6 show the percentage of change in the performance of individual models and voting classifiers between experiment 1 and 5. As indicated in the tables, the improvement ranged between 1% and 13%, emphasizing the overall benefit of adding more features, except for the NB model.

Table 5: Change of Performance Between Experiment 1 and Experiment 5 for Individual Models (in %).

	SVM		RF		LR		NB		KNN		AB		XGB	
DS	1	2	1	2	1	2	1	2	1	2	1	2	1	2
A	+1	+1	+2	+1	+1	+1	-64	-64	+1	+1	+2	+2	+3	+2
F1	+3	+3	+4	+3	+3	+3	-60	-60	+4	+4	+6	+6	+7	+7
P	+11	+11	+13	+12	+9	+10	+8	+8	+4	+4	+3	+1	+4	+4
R	+1	+1	+2	+1	+1	+1	-64	-64	+1	+1	+2	+2	+3	+2

Table 6: Change of Performance Between Experiment 1 and Experiment 5 for Voting Classifiers aggregating all classifiers' results (in %).

	H		S	
DS	1	2	1	2
A	+2	+2	+2	+2
F1	+6	+5	+6	+5
P	+13	+12	+13	+12
R	+2	+2	+2	+2

Our classification results surpass those presented in the works of (Sakib et al., 2022) and (Solorio et al., 2013) across all classifiers. However, the results by (Yamak et al., 2016) demonstrate superior performance with a Random Forest classifier reaching 99% accuracy. Yamak et al. suggest that this high performance might be attributed to an overfitting problem. Additionally, after analysing the proposed models, we found that the authors did not remove duplicate accounts, and some similar accounts within

one group were assigned to both the training and testing data. This increases the possibility of overfitting, as the models were trained on similar accounts used for testing. Thus, we exclude the results of that study from our comparison table. This comparative analysis is summarized in Table 7, which compares the seven classifiers testing results of experiment 5 with the results of (Sakib et al., 2022) and (Solorio et al., 2013).

Table 7: Comparative analysis of individual classifiers by accuracy (A), F1 score, precision (P) and recall (R). Dash indicates that the results were not available. Results are provided as A/F1/P/R.

	Sakib et al., 2022	Solorio et al., 2013	Ours
SVM	-	69/72/68/75	87/83/85/87
RF	-/82/-/-	-	87/83/86/87
NB	-/60/-/-	-	25/24/82/25
LR	-/75/-/-	-	87/83/84/87
KNN	-	-	86/84/84/86
AB	-	-	88/86/86/88
XGB	-	-	88/86/86/88

Regarding voting classifiers, we compared our voting classifiers results, which aggregate the results of seven classifiers, with those of papers (Nguyen et al., 2021) and (Wang et al., 2023), which employed both hard and soft voting classifiers to aggregate the outcomes of five models (RF, NB, KNN, XGB, and AB) and (XGB, LightGBM, RF, ConvNet, FC), respectively. Our results exhibit higher performance across all evaluation metrics compared to both studies. The comparison between the results is shown in Table 8.

Table 8: Comparative analysis of voting classifiers by accuracy (A), F1 score, precision (P) and recall (R). Dash indicates that the results were not available. Results are provided as A/F1/P/R.

	Nguyen et al., 2021	Wang et al., 2023	Ours
Hard Voting	83/83/85/80	-/49/-/-	88/85/86/88
Soft Voting	82/81/87/76	-/51/-/-	88/85/86/88

Including additional features significantly improved the performance of the Random Forest, AdaBoost and XGBoost classifiers, as demonstrated in Figure. 1, Figure. 2, and Figure. 3. This enhancement is attributed to the capacity to effectively handle and leverage the increased feature set, leading to better overall model accuracy. The bar charts clearly illustrate the upward trend in performance as more features were added. Moreover, in experiment 6 in which we selected the best 15 features, the results were comparable with previous experiments.

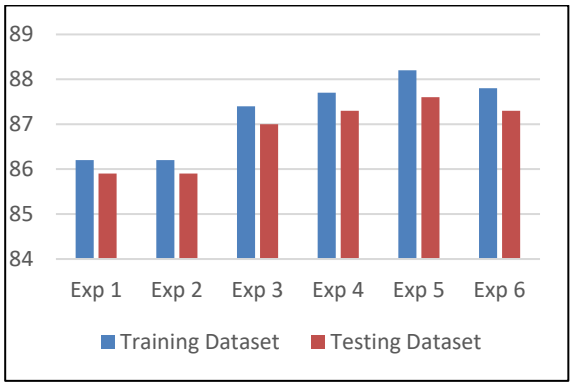

Figure 1: Random Forest performance.

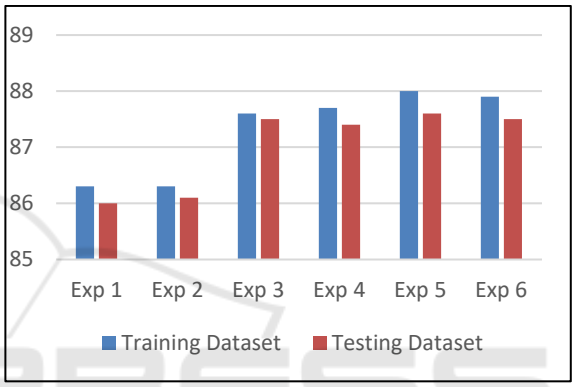

Figure 2: AdaBoost performance.

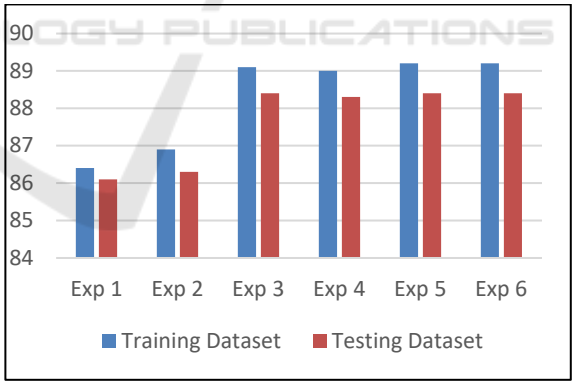

Figure 3: XGBoost performance.

5.1 Misclassified Accounts Analysis

We also reported the number of misclassified accounts that were misclassified by all classifiers for each phase separately, training and testing. Thus, we computed for each account, its target class, and the classification results from each classifier to show whether they generated similar classifications. Also, we computed how many classifiers correctly classified each account and how many accounts were

correctly classified by each classifier. As shown in Table 9, adding more features decreased the number of misclassified accounts, except in the case of experiment 5, where the number of misclassified accounts increased slightly, and this might indicate that some of the features added in experiment 5 affected the performance of the models. On the other hand, in experiment 6, the numbers also increased because the number of features was reduced. However, these numbers (4,830-1,309) are slightly better than the result reported by experiment 3 (4,894-1,325) even though both experiments have the same number of features, 15 features. Additionally, Figure 4 shows the percentage of misclassified accounts for both the training and testing datasets. As shown in the graph, the percentage of misclassified accounts in the testing dataset is lower, which indicates that our models are performing well and do not exhibit overfitting.

Table 9: Misclassified accounts by all classifiers.

	Training (~93,000)	Training (%)	Testing (~23,125)	Testing (%)
Experiment 1	11,340	12.19%	3,025	13.08%
Experiment 2	6,631	7.13%	1,810	7.82%
Experiment 3	4,894	5.26%	1,325	5.73%
Experiment 4	629	0.68%	175	0.76%
Experiment 5	673	0.72%	207	0.90%
Experiment 6	4,830	5.19%	1,309	5.66%

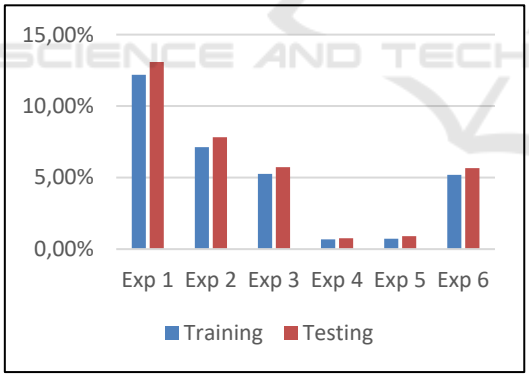

Figure 4: Percentage of misclassified accounts by all classifiers.

We analysed these misclassified accounts and found that all of them were genuine accounts mistakenly classified as Sockpuppet accounts. For this reason, we compared these misclassified accounts with Sockpuppet accounts in the detailed observations below.

First, the average time difference between their comments ranged from 15 minutes to 24 hours, closely resembling the patterns observed in real Sockpuppet accounts, which range between 1 minute and 23 hours. Additionally, the total number of revisions for these accounts ranged between 1 and 200, similar to Sockpuppet accounts. Thirdly, even though there are 28 namespaces on Wikipedia, contributions by namespace for these accounts ranged between 1 and 7, aligning with the participation in 1 to 8 namespaces by real SP accounts, except for 48 accounts out of 115,620 who participated in 9-16 namespace. Similarly, the average contributions by title for these accounts ranged from 1 to 100, with some accounts editing one article multiple times. For example, user CD1975 made a total of 5 revisions, all in one article. This mirrors the behaviour of Sockpuppet accounts, except for one account that edited 14,407 titles.

These observations indicate that the behaviour of genuine users and Sockpuppet accounts can be quite similar. However, data analysis can significantly improve the accuracy of distinguishing between these types of accounts. One notable difference is the account age; Sockpuppet accounts typically have a much shorter lifespan compared to normal accounts. Additionally, Sockpuppet accounts tend to write longer sentences. For instance, when we analysed the average number of sentences per contribution, we found that normal accounts ranged from 0 to 6 sentences. In contrast, Sockpuppet accounts ranged from 0 to 50 sentences per contribution, with some accounts contributing as many as 147 sentences. This distinct behavioural pattern provides a valuable indicator for accurately identifying Sockpuppet accounts, even in the presence of misleading similarities with normal accounts.

Moreover, we generated the number of accounts misclassified by Random Forest, AdaBoost, and XGBoost classifiers in each experiment. Table 10 Table 11, and Table 12 show that the improvement in these classifiers' performance, and it is evident in the reduction of misclassified accounts. In other words, adding more features to these classifiers decreased the number of misclassified accounts. Figures 5, Figure 6, and Figure 7 show the percentages of misclassified accounts for each classifier, which highlighting the improvements achieved by each classifier across the experiments.

Table 10: Misclassified accounts by Random Forest.

	Training (~93,000)	Training (%)	Testing (~23,125)	Testing (%)
Experiment 1	12,718	13.67%	3,240	14.01%
Experiment 2	12,718	13.67%	3,240	14.01%
Experiment 3	11,532	12.40%	2,988	12.92%
Experiment 4	11,286	12.14%	2,936	12.69%
Experiment 5	10,935	11.75%	2,857	12.35%
Experiment 6	11,198	12.04%	2,928	12.66%

Figure 5: Percentage of misclassified accounts by Random Forest.

Table 11: Misclassified accounts by AdaBoost.

	Training (~93,000)	Training (%)	Testing (~23,125)	Testing (%)
Experiment 1	12,693	13.64%	3,225	13.95%
Experiment 2	12,659	13.61%	3,207	13.86%
Experiment 3	11,434	12.29%	2,894	12.52%
Experiment 4	11,396	12.25%	2,900	12.54%
Experiment 5	11,083	11.92%	2,864	12.39%
Experiment 6	11,182	12.02%	2,881	12.46%

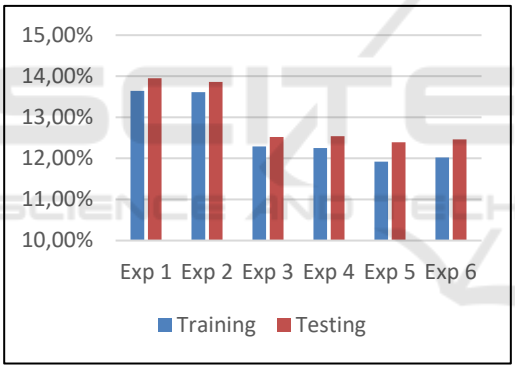

Figure 6: Percentage of misclassified accounts by AdaBoost.

Table 12: Misclassified accounts by XGBoost.

	Training (~93,000)	Training (%)	Testing (~23,125)	Testing (%)
Experiment 1	12,545	13.49%	3,205	13.86%
Experiment 2	12,095	13.00%	3162	13.67%
Experiment 3	10,054	10.81%	2673	11.56%
Experiment 4	10,095	10.86%	2690	11.63%
Experiment 5	9,916	10.66%	2,675	11.57%
Experiment 6	9,980	10.73%	2,680	11.59%

6 CONCLUSIONS

In this study, we performed a comprehensive analysis of various methods for detecting Sockpuppet accounts on Wikipedia. We analysed previous

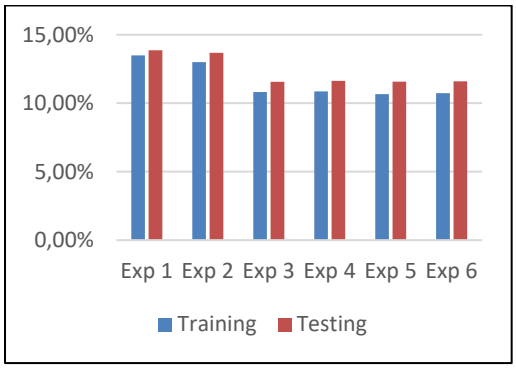

Figure 7: Percentage of misclassified accounts by XGBoost.

features and proposed four new ones, that improved the performance of the classifiers. We studied seven distinct basic classifiers (SVM, RF, LR, NB, KNN, AB, and XGB) and two voting classifiers applied on various combinations of the basic ones. The best results were obtained by aggregating the result of Random Forest, AdaBoost, and XGBoost achieving an accuracy rate of 88% with 87% precision.

Additionally, our study highlights several important points and findings regarding Sockpuppet detection. First, feature engineering plays a crucial role in enhancing the performance of all models, as demonstrated by the significant improvements observed when additional features were incorporated. Also, parameter tuning, and refinement are equally important to avoid overfitting, and this can be achieved manually and through automated solutions like GridSearchCV.

Despite achieving results that surpass previous work, there is still room for improvement. One potential enhancement is implementing a rule-based voting system built on the analysis of training results. Additionally, integrating cognitive assistants and human-in-the-loop methodologies with machine learning techniques can offer substantial benefits. Cognitive assistants can provide contextual insights and recognize complex patterns that automated systems might overlook. Incorporating human expertise is also valuable, as humans can identify anomalies and provide feedback to the machine learning models, leading to continuous improvement and adaptation, particularly for accounts misclassified by automated systems. For example, when our analysis of misclassified normal accounts revealed that these accounts have longer lifespans than Sockpuppet accounts, we added the account age feature in Experiment 3, and it improved the results of all classifiers. This demonstrates how human intervention can enhance detection accuracy. By

leveraging both automated systems and human insights, we can refine the Sockpuppet detection process and achieve even better results.

REFERENCES

Ashford, J. R., Turner, L. D., Whitaker, R. M., Preece, A., & Felmlee, D. (2020, December). Assessing temporal and spatial features in detecting disruptive users on Reddit. In 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM) (pp. 892-896). IEEE.

Baamer, R., Boicu. M. (2024). "Sockpuppet accounts detection on Wikipedia: Research report." Under review

Bastos, M. T., Puschmann, C., & Travitzki, R. (2013, May). "Tweeting across hashtags: overlapping users and the importance of language, topics, and politics," In Proceedings of the 24th ACM conference on hypertext and social media (pp. 164-168).

Bhatia, T., Manaskasemsak, B., & Rungsawang, A. (2023, March). "Detecting fake news sources on Twitter using Deep Neural Network," In 2023 11th International Conference on Information and Education Technology (ICIET) (pp. 508-512). IEEE.

Bhopale, J., Bhise, R., Mane, A., & Talele, K. (2021, September). "A Review-and-Reviewer based approach for Fake Review Detection," In 2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT) (pp. 1-6). IEEE.

Boididou, C., Papadopoulos, S., Apostolidis, L., & Kompatsiaris, Y. (2017, June). "Learning to detect misleading content on twitter," In Proceedings of the 2017 ACM on international conference on multimedia retrieval (pp. 278-286).

Borkar, B. S., Patil, D. R., Markad, A. V., & Sharma, M. (2022, November). "Real or fake identity deception of social media accounts using Recurrent Neural Network," In 2022 International Conference on Fourth Industrial Revolution Based Technology and Practices (ICFIRTP) (pp. 80-84). IEEE.

Boshmaf, Y., Logothetis, D., Siganos, G., Lería, J., Lorenzo, J., Ripeanu, M., & Beznosov, K. (2015, February). "Integro: Leveraging victim prediction for robust fake account detection in OSNs," In NDSS (Vol. 15, pp. 8-11).

He, B., Ahamad, M., & Kumar, S. (2021, August). Petgen: Personalized text generation attack on deep sequence embedding-based classification models. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (pp. 575-584).

Kuruvilla, A. M., & Varghese, S. (2015, March). A detection system to counter identity deception in social media applications. In 2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015] (pp. 1-5). IEEE.

Liu, Wei & Xie, Chencheng & Zong, Shijie. (2023). A rumor source identification method based on node embeddings and community detection in social networks. 104-109. 10.1109/CBD63341.2023.00027.

Nguyen, N. L., Wang, M. H., & Dow, C. R. (2021). "Learning to recognize sockpuppets in online political discussions," IEEE Systems Journal, 16(2), 1873-1884.

Sakib, M., & Spezzano. F,. (2022). "Automated Detection of Sockpuppet accounts in Wikipedia," 2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey, 2022, pp. 155-158, doi: 10.1109/ASONAM55673.2022.10068604.

Sakib, M. N. (2024) Wikipedia Sockpuppetry, GitHub, Available at: https://github.com/Mostofa-Najmus-Sakib/Wikipedia-Sockpuppetry/tree/main/data Accessed on: April 14, 2023

Solorio, T., Hasan, R., & Mizan, M. (2013, June). "A case study of sockpuppet detection in wikipedia," In Proceedings of the Workshop on Language Analysis in Social Media (pp. 59-68).

Solorio, T., Hasan, R., & Mizan, M. (2014). Sockpuppet detection in wikipedia: A corpus of real-world deceptive writing for linking identities. arXiv preprint arXiv:1310.6772.

Tsikerdekis, M., & Zeadally, S. (2014). Multiple account identity deception detection in social media using nonverbal behavior. IEEE Transactions on Information Forensics and Security, 9(8), 1311-1321.

Scikit Learn (2014). Voting Classifier. API Reference. Accessed on May 26, 2024 https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

Wang, Ruei-Yuan & Chen, Hung-Hsuan. (2023). Detecting Inactive Cyberwarriors from Online Forums. 10.48550/arXiv.2308.15491.

Yamak, Z., Saunier, J., & Vercouter, L. (2016). "Detection of multiple identity manipulation in collaborative projects," In Proceedings of the 25th International Conference Companion on World Wide Web (pp. 955-960).

Yamak, Z., Saunier, J., & Vercouter, L. (2018). SocksCatch: Automatic detection and grouping of sockpuppets in social media. Knowledge-Based Systems, 149, 124-142

Yu, H., Hu, F., Liu, L., Li, Z., Li, X., & Lin, Z. (2021). Sockpuppet detection in social network based on adaptive multi-source features. In Modern Industrial IoT, big data and supply chain: proceedings of the IIoTBDSC 2020 (pp. 187-194). Springer Singapore.

APPENDIX

Link: https://drive.google.com/file/d/1uDDc-eld14-Opok92RbYq0qyqrQ8msfG/view