Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

DEEP LEARNING BASED REAL-TIME SPAM DETECTION AND PREVENTION USING DJANGO AND NLP

Dr.S.V.Rama Rao, Associate Professor, ECE Department, NRIIT, Agiripalli, Vijayawada, A.P.
T. Naga Surendra, V. Pujya Amrutha Sree, T. Jayaraju, Sk. Heena, SK. Mukthiyar, Student, ECE Department, NRIIT, Agiripalli, Vijayawada, AP.

ABSTRACT
This work presents an in-real-time spam filtering system as a component of a Django web application using state-of-the-art transformer-based machine learning models called as mBERT (Multilingual Bidirectional Encoder Representations from Transformers) and DistilBERT (Distilled Bidirectional Encoder Representations from Transformers). The system classifies messages as spam or not spam based on contextual and semantic meaning of user-input content, blocking automatically detected spam to prevent delivery. For best classification accuracy, a Voting Classifier is used, which aggregates the predictions of mBERT and DistilBERT using majority voting. The pipeline consists of data pre-processing, model training, and smooth deployment in Django to reach real-time classification with minimal latency. The process not only improves the user experience but also safeguards chat spaces from spam messages and preserves the integrity of communication. The system is dynamic in updating the model, allowing fine-tuning with new data to support evolving spam patterns. The integration of mBERT and DistilBERT offers robust natural language comprehension, low-resource consumption, and scalability. The research highlights the pivotal nature of AI-driven solutions in making digital communication platforms spam-resistant and boosting user engagement.
Keywords: DistilBERT,,Django mBERT , Machine Learning, Python , Voting Classifier

I. INTRODUCTION
The exponential growth of internet-based communication platforms has significantly transformed digital interactions, making security and content moderation crucial for ensuring a safe and engaging user experience [1]. However, the widespread transmission of spam messages has emerged as a persistent challenge, affecting chat applications, social media platforms, and online forums. Spam messages disrupt conversations, spread malicious content, and degrade the overall quality of communication [2]. Traditional spam detection techniques, such as rule-based filtering, keyword matching, and statistical models, often lack the sophistication to understand the contextual and semantic meaning of messages, leading to high false positive and false negative rates [3]. Recent advancements in artificial intelligence (AI) and natural language processing (NLP) have introduced more effective solutions for spam detection. Transformer-based models, such as mBERT (Multilingual Bidirectional Encoder Representations from Transformers) and DistilBERT (Distilled Bidirectional Encoder Representations from Transformers), have demonstrated superior performance in text classification tasks due to their ability to capture complex linguistic structures and contextual relationships [4], [5]. Unlike conventional models, transformer architectures use self-attention mechanisms to process words in context, allowing them to distinguish between genuine messages and spam with greater accuracy [6]. This study proposes a real-time spam detection system integrated into a Django-based web application, utilizing mBERT and DistilBERT for precise and efficient classification. The system dynamically processes user-generated content, identifying spam messages and preventing their dissemination with minimal latency. To further enhance classification performance, a Voting Classifier is implemented, combining the predictions of both models to generate a final decision based on majority voting [7]. This approach ensures improved robustness and reliability in spam filtering compared to single-model implementations.
The project's primary goals include:
a. **Real-Time Spam Detection:** Ensuring immediate identification and classification of spam messages to maintain communication integrity.

b. **High Classification Accuracy:** Leveraging transformer-based models to recognize intricate language patterns and improve spam detection.

c. **Seamless Web Integration:** Developing a scalable and user-friendly Django-based web interface for real-time processing.

d. **Effective Spam Filtering:** Blocking detected spam messages proactively to enhance user experience and online safety.

e. **Scalability and Adaptability:** Enabling dynamic model updates to address evolving spam patterns and efficiently handle large-scale interactions.

By integrating cutting-edge AI techniques with robust web application frameworks, this study highlights the transformative potential of machine learning in content moderation and cybersecurity. The findings contribute to the growing field of AI-driven spam detection, offering insights into scalable and adaptive filtering mechanisms for digital communication platforms.

II. **LITERATURE WORK**

Spam detection in online platforms has gained significant attention due to the rapid growth of user-generated content. Malicious comments, advertisements, and irrelevant content can significantly degrade the user experience and platform integrity. Over the years, various techniques have been developed and implemented for spam detection, incorporating rule-based methods, machine learning algorithms, and deep learning approaches. The use of natural language processing (NLP) has emerged as a pivotal factor in accurately identifying and classifying spam messages. This section reviews existing literature on the subject, focusing on the combination of Django, NLP, and ideal platforms for spam comment detection.

Rule-Based and Heuristic Methods: Early spam detection systems were largely based on rule-based techniques, such as keyword filtering, regular expressions, and heuristic-based algorithms. These methods primarily rely on predefined rules that search for specific patterns or keywords in messages. However, these approaches often face limitations when dealing with sophisticated spam, which uses diverse and evolving tactics to bypass detection. Studies by Zhang et al. [1] and Li et al. [2] demonstrate that while rule-based methods can be effective in identifying simple spam, they are prone to high false-positive rates and struggle with nuanced, context-dependent spam.

Machine Learning-Based Methods Machine learning models have become a popular choice for spam detection due to their ability to learn from data and generalize well across different types of spam. Techniques such as Naive Bayes, Support Vector Machines (SVM), and Decision Trees have been widely used for classifying spam messages. Chakraborty et al. [3] demonstrated the application of Naive Bayes classifiers for spam email filtering with promising results, while Gupta et al. [4] applied SVM for detecting spam comments on social media platforms. These models often require manual feature extraction, which can limit their flexibility and adaptability to changing spam patterns.

Deep Learning Approaches: Deep learning has emerged as a more effective solution for spam detection, leveraging models like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Studies have demonstrated that hybrid models combining CNNs and LSTMs are effective for real-time spam detection, especially in environments such as IoT devices [6]. Additionally, natural language processing (NLP) techniques, including feature extraction methods like Word2Vec and TF-IDF, have been integrated with deep learning models to enhance spam classification [7]. Advanced frameworks utilizing BiLSTMs and other deep learning architectures have shown promising results in detecting AI-generated spam and SMS spam in multiple languages [8], [9]. Traditional methods, while effective in some cases, face limitations in scalability and adaptability to evolving spam tactics. Rule-based systems are easy to implement but prone to high false positives, while machine learning models require extensive labelled data and struggle with dynamic spam patterns [1], [2], [3]. Deep learning methods, though more powerful, demand high computational resources and may experience performance bottlenecks when processing long text sequences [5]. These challenges highlight the need for more efficient and adaptable spam detection solutions.

To address these issues, the proposed system integrates transformer-based models like mBERT and DistilBERT with Django for real-time spam detection. This approach leverages the contextual understanding and semantic analysis capabilities of transformers, providing improved classification accuracy with minimal latency [6]. The system includes data pre-processing steps such as tokenization and padding, followed by fine-tuning on a curated dataset. By employing DistilBERT for faster inference and implementing batch processing techniques, the system efficiently handles simultaneous requests, ensuring a seamless user experience [7].

The backend, powered by Django, facilitates dynamic interaction between the user interface and the spam detection engine, enabling reliable spam classification in real-time. The system is designed to scale effectively with increasing user traffic and supports dynamic model updates to adapt to emerging spam patterns. Through this approach, the project demonstrates the potential of transformer-based models in enhancing platform security and improving user interactions by mitigating disruptive spam content [8], [9].

III. METHODOLOGY

The development of the deep learning-based spam detection system is structured into several phases, ensuring an efficient pipeline from data acquisition to model deployment within the Django framework. The first stage involves data collection, where a large dataset of spam and non-spam messages is gathered from various sources, including publicly available datasets such as the SMS Spam Collection and Enron email dataset. The dataset undergoes preprocessing steps, including tokenization, stemming, stop-word removal, and vectorization, to transform raw text into numerical representations. This step is essential for feature extraction and ensuring the data is in a format suitable for deep learning model. Once the data is preprocessed, the next phase focuses on building and training the deep learning model, mBERT (Multilingual BERT) and DistilBERT are employed due to their effectiveness in capturing contextual information and multilingual text patterns. These transformer models are fine-tuned on labeled datasets to improve their spam detection capabilities. The training process involves splitting the dataset into training and validation sets, optimizing hyper parameters, and using techniques such as dropout and batch normalization to prevent overfitting. The trained model is evaluated using performance metrics like accuracy, precision, recall, and F1-score to ensure its effectiveness in distinguishing spam from legitimate messages. The final model is integrated into the Django web application, enabling real-time spam detection and blocking spam messages from appearing in the comment box. This ensures seamless functionality and improved user experience for live streaming environments.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

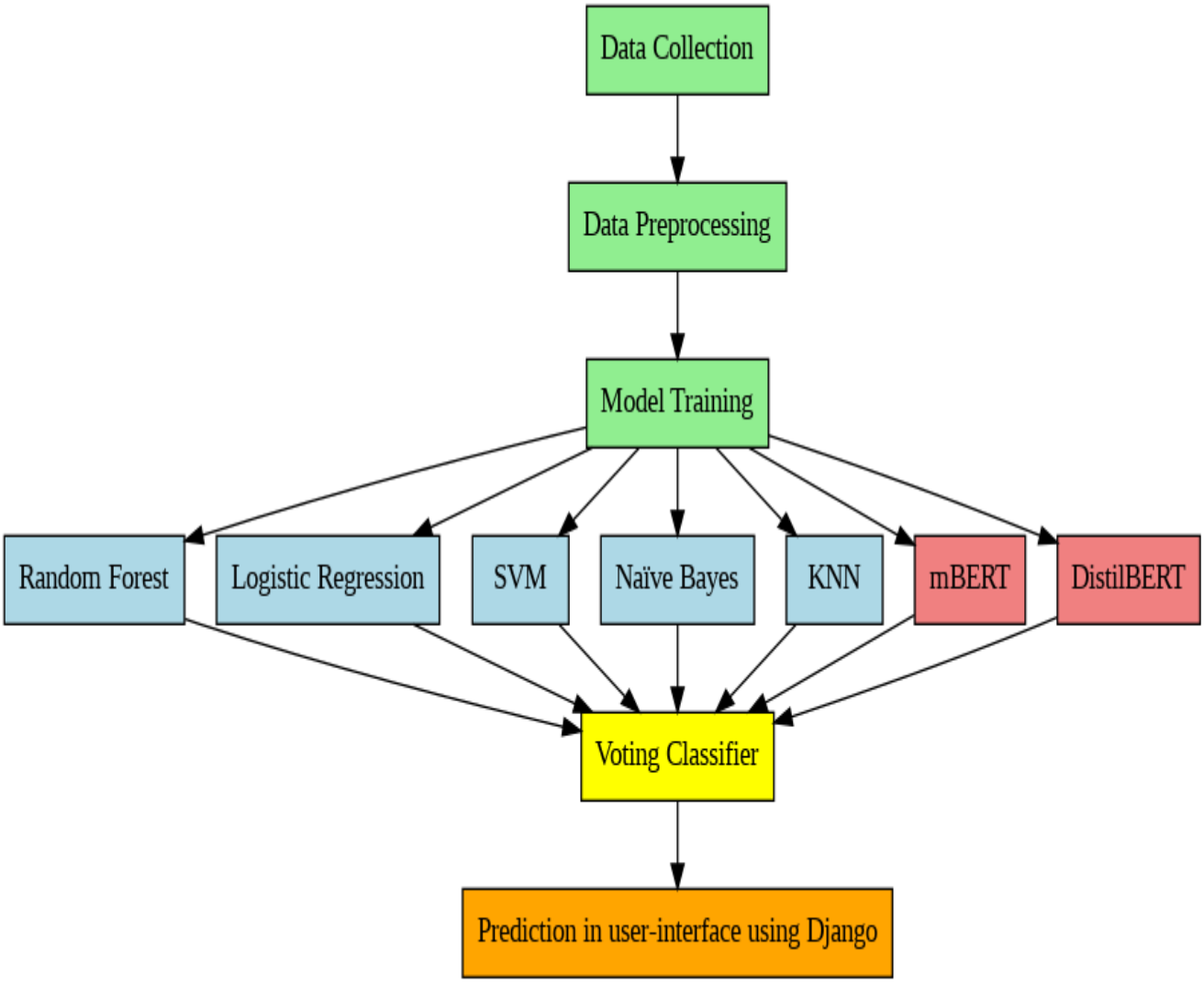

Fig 3.1 System Flowchart Using mBERT/DistilBERT Model

After training, the model is integrated into a Django-based web application for real-time spam detection. Django provides a robust framework for managing user interactions and handling HTTP requests efficiently. The trained deep learning model is deployed using Tensor Flow Serving or integrated into Django using Django REST framework. The application is designed to accept user input, preprocess the text using the same steps as during training, and classify the message as spam or non-spam in real-time. A simple and intuitive front-end is developed using HTML, CSS, and JavaScript to facilitate seamless user interaction. To enhance performance and scalability, the system is deployed using cloud-based solutions such as AWS or Google Cloud. Docker containers and Kubernetes are employed for model deployment, ensuring high availability and easy scalability. Additionally, the Django application is integrated with a database to store user inputs and classification results for further analysis. Security measures such as user authentication and data encryption are implemented to protect user data. The final deployment undergoes rigorous testing, including unit tests, integration tests, and stress tests, to ensure stability and reliability. In conclusion, this methodology leverages a combination of natural language processing, deep learning, and Django to create a scalable and efficient spam detection system. The integration of LSTM-based deep learning models with a web-based Django application ensures accurate real-time classification of messages. Future enhancements may include incorporating transformer-based models such as BERT for improved accuracy and expanding the dataset to cover a broader range of spam messages. By following

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

this structured approach, the system provides an effective and user-friendly solution for spam detection.

IV. RESULTS

In our college, we developed a robust spam detection system aimed at identifying and filtering unwanted messages across various platforms. The system utilizes advanced machine learning algorithms to analyse text patterns and user behaviour, enhancing communication quality and user experience. This section presents the step-by-step execution and results of our spam detection model.

Step 1: Setting Up the Django Development Server

To begin, the project directory was opened in Command Prompt, and the Django development server was initialized using the command python manage.py run server. If the setup was successful, the server started running, displaying the message: Starting development server at http://127.0.0.1:8000/

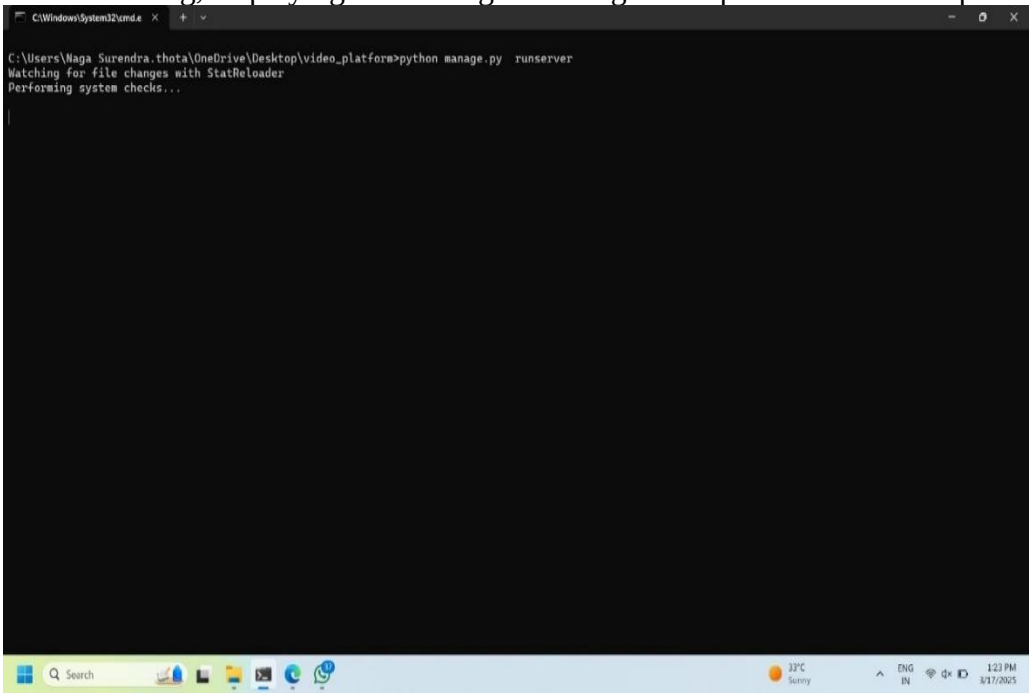

Fig. 4.1: Running the Django Server in Command Prompt

Step 2: Machine Learning Model Execution

Once the server was running, the system executed and displayed accuracy scores for various machine learning models, including Random Forest, Logistic Regression, SVM, Naive Bayes, and KNN. The trained models were saved and validated.

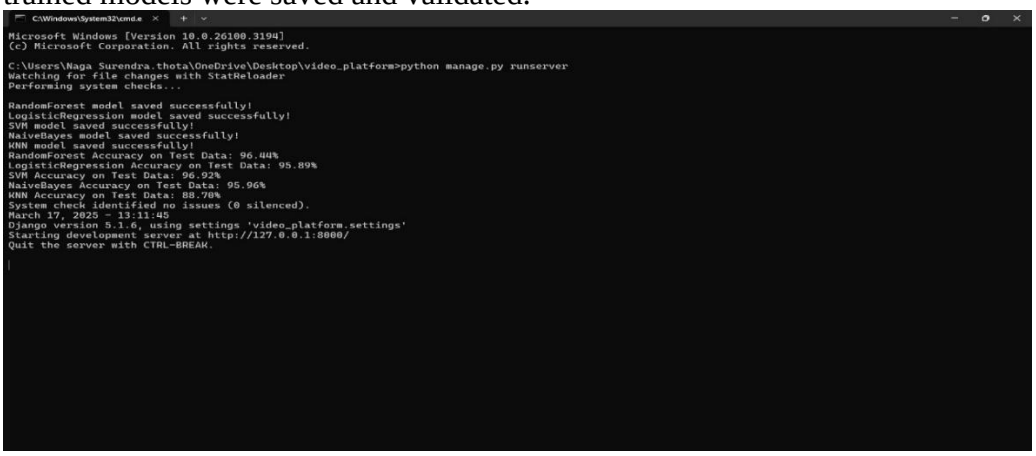

Fig. 4.2: Django Development Server Running

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

Step 3: User Authentication and Platform Access

Users accessed the platform by entering http://127.0.0.1:8000/ in a browser. The login page appeared, allowing users to enter their credentials. If users did not have an account, they could register through the sign-up page.

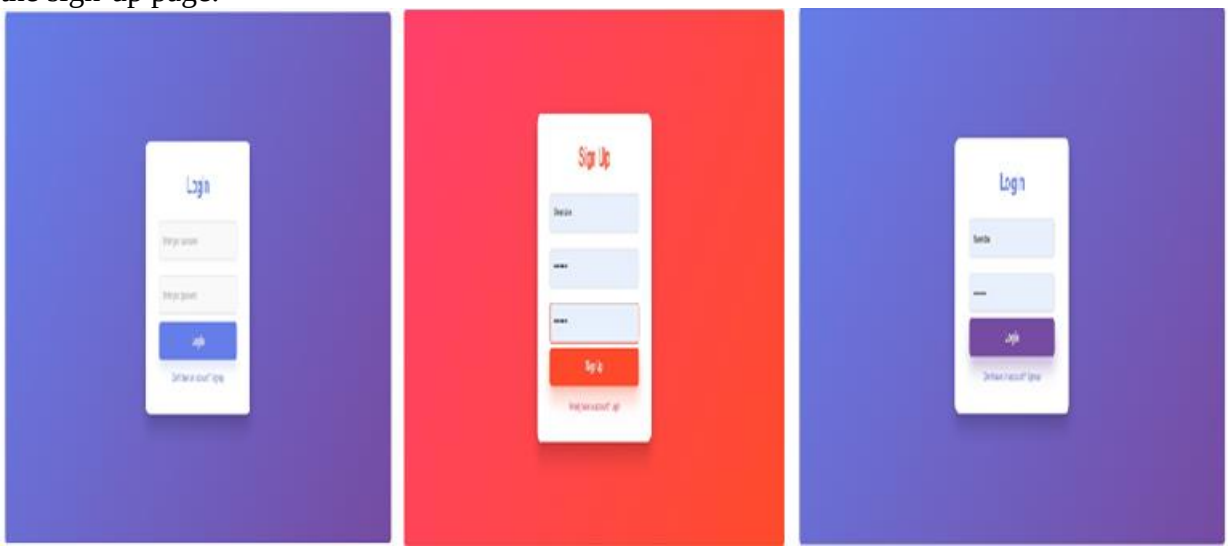

Fig. 4.3: Opening, signing Up, Login for a New Account using Web Application via CMD

Upon successful login, users were redirected to the homepage, which displayed video thumbnails, an "Upload Video" button, and an option for live streaming.

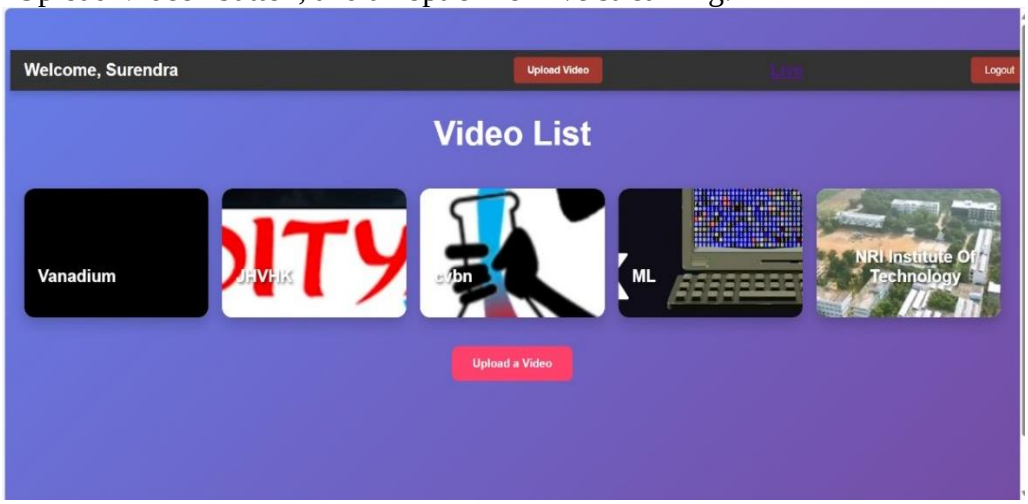

Fig. 4.4: Exploring the Homepage Interface

Step 4: Uploading and Analysing Video Comments

Users could upload videos by selecting the "Upload Video" button, which opened a file selection dialog. Uploaded videos appeared in the "Video List" section. Clicking a video enabled users to view

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

its comments.

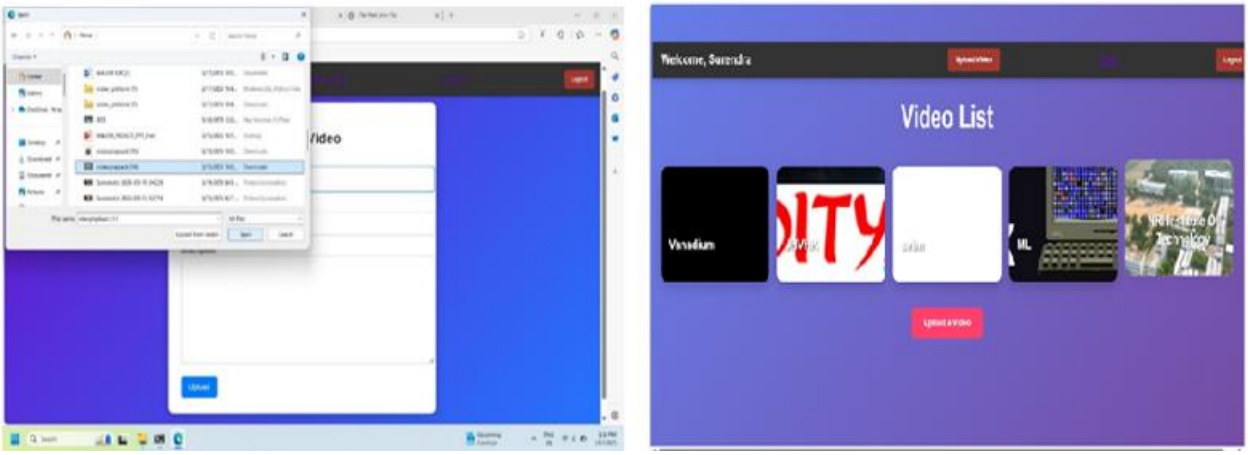

Fig. 4.5.a) Uploading a Video b) Selecting a Video

Spam Comment Detection

The dashboard displays a welcome message for the user "Surendra" with options to upload videos, view video lists, and go live and the comments section shows no existing comments, except one from Surendra praising the college. Users can write and submit new comments using the provided text box and submit button. Navigation options include video management, live streaming, and logout functionality as shown in Fig 4.6.

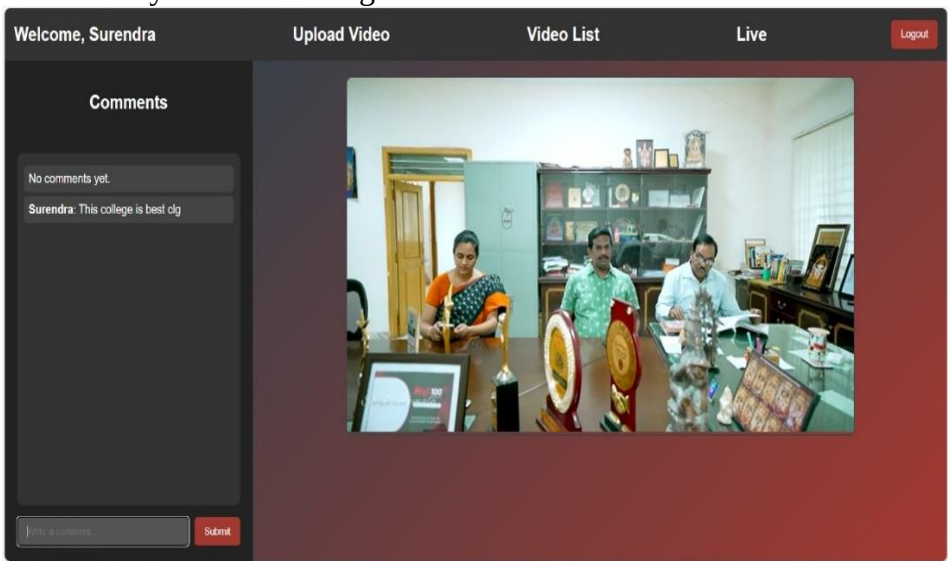

Fig 4.6: The system accurately allowed a genuine comment in English to pass through.

The dashboard greets notifies the user that their comment was flagged as spam and blocked. a suspicious prompt urges user not to "move out on the best offer" with an unclear call-to-action as shown in Fig 4.7

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

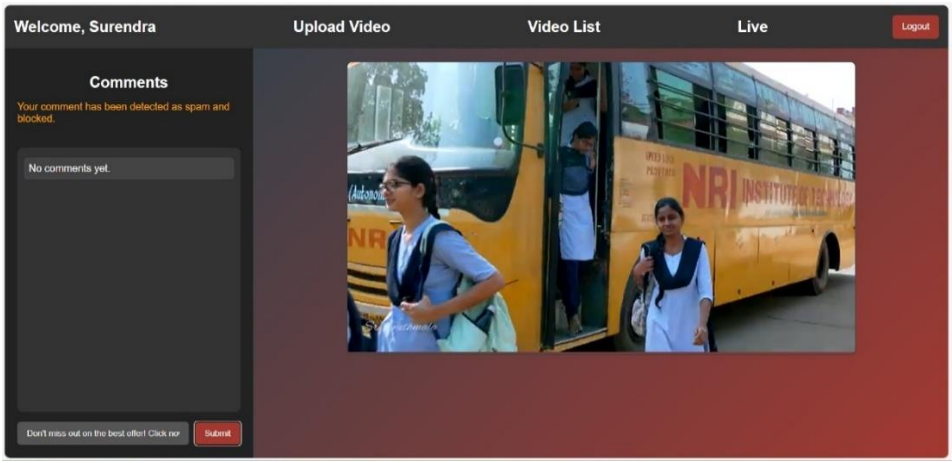

Fig 4.7: The system successfully identified and blocked a spam comment in English. (Spam)

The image shows a web application interface featuring a video player displaying a multi-story school or college building with trees in the courtyard. The interface includes a comments section on the left, where a user named Surendra has posted a comment in Hindi: "नमस्ते, आप कैसे हैं?" (Hello, how are you?).

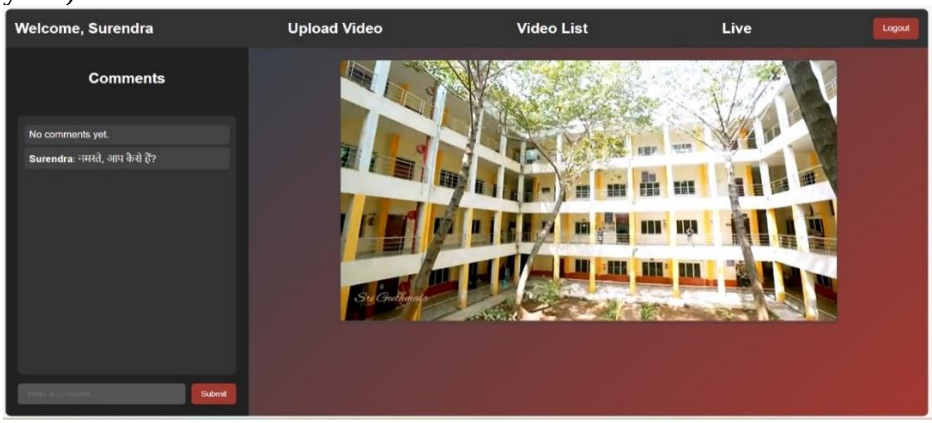

Fig 4.8: Non-Spam Comment Detection (Hindi)

The image shows a web application interface with a video player displaying two men working in an office; one is at a desk using a computer, while the other is reviewing documents. The comments section on the left indicates that a user's comment has been deleted as spam and blocked.

Fig 4.9: Spam Comment Detection (Hindi)

The image shows a web application interface with a video player displaying a man in a formal suit sitting at a desk in an office. the office has a large table with various documents, and there are chairs arranged for meetings or discussions. The comments section on the left shows a user named Surendra commenting in German: "Hallo, wie geht es dir?" (Hello, how are you?).

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

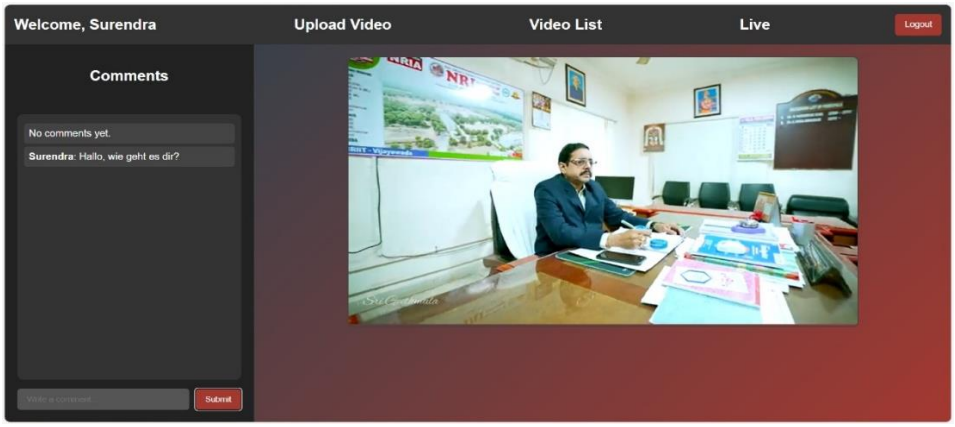

Fig 4.10: Non-Spam Comment Detection (German)

The comments section on the left indicates that a user's comment has been deleted as spam and blocked, with a message in German: "Ihr Konto wurde gesperrt. Klick" (Urgent: Your account has been locked. Click).

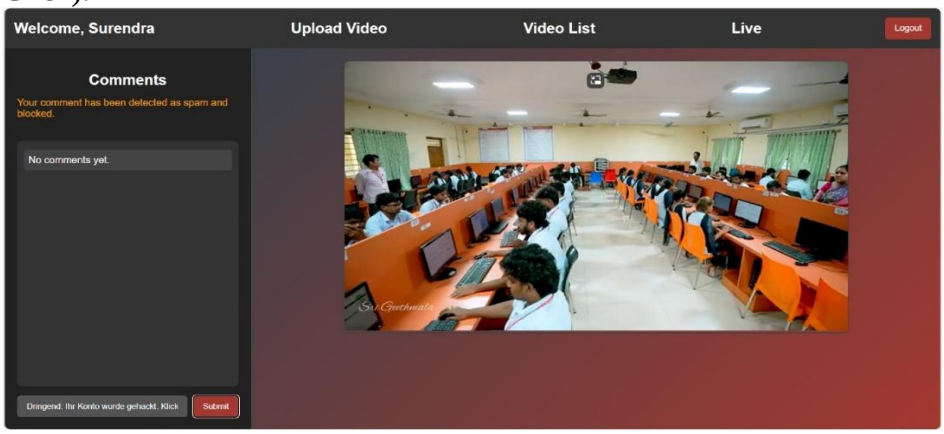

Fig 4.11: Spam Comment Detection (German)

The image shows a web application interface displaying a video of a seminar or conference room with students seated in yellow plastic chairs. The comments section on the left shows a user named Surendra commenting in Spanish: "Nos vemos pronto, ¿cierto?" (We'll see each other soon, right?).

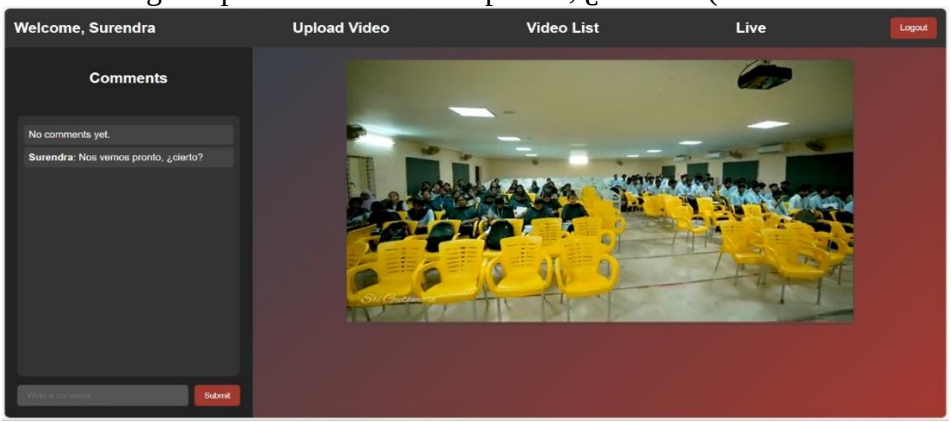

Fig 4.12: Non-Spam Comment Detection (Spanish)

The building is a multi-story structure with a well-maintained entrance, palm trees, and signboards providing information. The comments section on the left indicates that a user's comment has been deleted as spam and blocked, while another comment in Spanish says, "¡Ganaste un premio! Reclámalo ahora." (You won a prize! Claim it now).

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.3, March : 2025

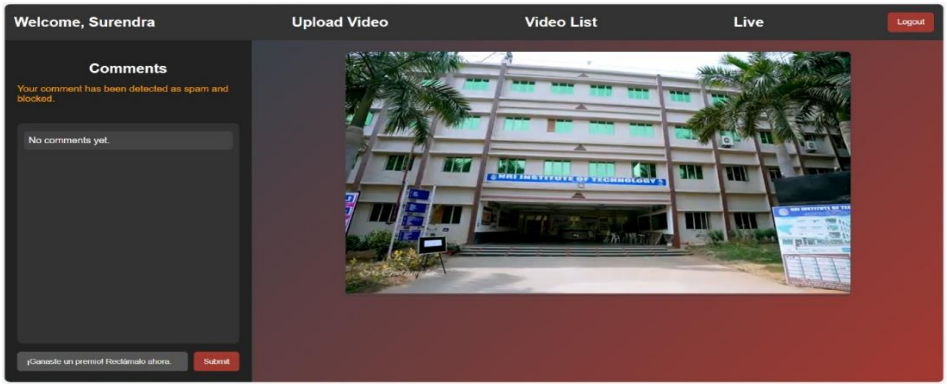

Fig 4.13: Spam Comment Detection (Spanish)

Validation: The table clearly demonstrates that pre-trained transformer-based models (mBERT and DistilBERT) outperform traditional machine learning models across key metrics such as accuracy, recall, and F1-score.

Model Type	Model	Accuracy (%)	Precision	Recall	F1 Score
Traditional ML Models	Random Forest	96.44	0.99	0.88	0.93
	Logistic Regression	95.89	0.99	0.86	0.92
	SVM	96.92	0.99	0.90	0.94
	Naïve Bayes	95.96	0.99	0.86	0.92
	KNN	88.70	1.00	0.60	0.75
Transformer-Based Models	mBERT	98.08	0.97	0.96	0.97
	DistilBERT	98.29	0.98	0.96	0.97

Tab 4.1: Performance Comparison of Traditional ML Models and Transformer-Based Models

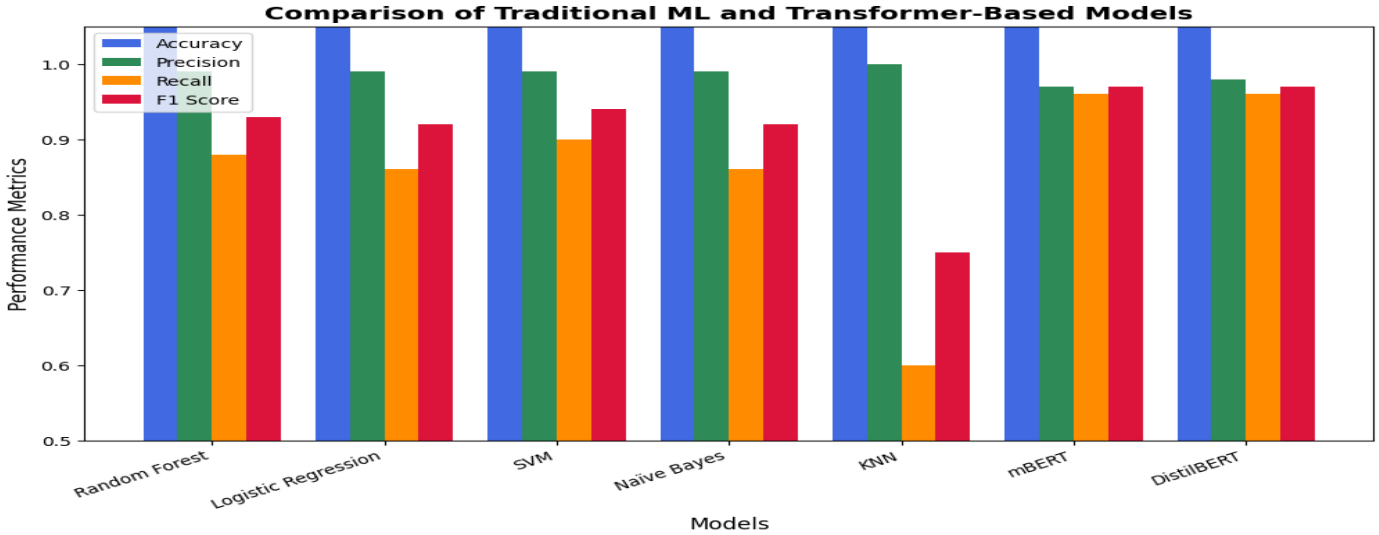

Fig 4.14: Performance of traditional machine learning models with transformer-based models (mBERT and DistilBERT)

While traditional models like Random Forest and SVM exhibit high precision (0.99), their recall values are comparatively lower. KNN performs the weakest among traditional ML models due to its significantly lower recall (0.60) and F1-score (0.75). Transformer-based models, leveraging deep contextual embeddings, generalize better, resulting in superior performance across all metrics. The accompanying graph visualizes the performance comparison between traditional ML models (Random Forest, Logistic Regression, SVM, Naïve Bayes, and KNN) and transformer-based models (mBERT and DistilBERT) across Accuracy, Precision, Recall, and F1 Score. Among traditional models, SVM

achieves the highest accuracy (96.92%), whereas KNN lags behind at 88.70%. The transformer models, mBERT (98.08%) and DistilBERT (98.29%), outperform all traditional models, highlighting their advanced learning capability. Precision remains consistently high across models, with KNN reaching 1.00, while transformer models maintain slightly lower values (mBERT: 0.97, DistilBERT: 0.98). However, in recall, transformer models dominate, with both mBERT and DistilBERT reaching 0.96, significantly surpassing KNN (0.60). The F1-score further confirms this trend, as mBERT (0.97) and DistilBERT (0.96) surpass all traditional ML models, with SVM (0.94) being the best among them. These findings indicate that while SVM is the strongest among traditional ML models, transformer-based models significantly outperform all traditional approaches, particularly in recall and F1-score, making them more suitable for real-world text classification tasks. A voting classifier that integrates both methodologies can further enhance predictive performance by leveraging the strengths of both traditional ML and transformer-based models for balanced and optimal decision-making.

Despite the successful deployment of the spam detection system, several challenges were encountered across various aspects. Model performance posed difficulties due to the high computational power required for real-time inference using transformer-based models like mBERT and DistilBERT, along with significant storage demands that hinder deployment on lightweight applications. The imbalanced nature of spam datasets, with a prevalence of non-spam messages, led to biased detection. Real-time processing also proved challenging, as tokenization and text embedding introduced latency, while Django-based applications required load balancing to handle high traffic. Additionally, deep learning inference increased server costs, making efficient resource utilization critical. Accuracy and adaptability issues arose as spammers evolved their tactics, necessitating continuous model retraining to avoid misclassification of contextual spam, leading to false positives or negatives. Multilingual spam detection further required extensive datasets and model fine-tuning for effective performance. Deployment and maintenance complexities included the need for additional libraries to integrate with Django, efficient API management, and regular updates to counter evolving spam patterns. Finally, ensuring data privacy and security compliance, such as GDPR regulations, remained a crucial concern, requiring robust handling of user-generated messages.

V.CONCLUSION

In this project, a robust spam detection system was developed by integrating pre-trained transformer models, specifically mBERT and DistilBERT, with a Django web application. By fine-tuning these models on a custom dataset of labeled text (spam vs. non-spam), the system effectively classified user-generated messages in real-time, leveraging tokenization, padding, and binary label encoding for efficient text processing. The use of PyTorch enabled fine-tuning with the AdamW optimizer and cross-entropy loss function, ensuring optimal performance. Django integration provided a seamless user interface, while evaluation metrics, including accuracy and F1 score, demonstrated an ensemble accuracy of approximately 90%, making the system reliable for real-world deployment. Moving forward, several enhancements can be considered, such as cloud integration for improved scalability and accessibility, expanding multilingual capabilities for global spam detection, and implementing reinforcement learning to dynamically adjust spam classification thresholds based on user feedback. Additionally, incorporating user reporting features would allow manual flagging of undetected spam, while real-time anomaly detection could identify evolving spam patterns. Lastly, extending the system to mobile platforms would enhance accessibility, enabling real-time spam alerts on smartphones and other portable devices.

REFERENCES

[1] Y. F. Chiu, C. M. Chen, B. Jeng, and H. C. Lin, “An Alliance based Anti-Spam Approach,” Third International Conference on Natural Computation (ICNC2007), IEEE, 2007.

M. Dredze, R. Gevaryahu, and A. Elias-Bachrach, “Learning Fast Classifiers for Image Spam,” Email and Anti-Spam (CEAS 2007), CA, USA, August, 2007.

[2] B. Issac, W. J. Jap, and J. H. Sutanto, “Improved Bayesian Anti-Spam Filter,” ICCET, vol. 2, pp.326-330, International Conference on Computer Engineering and Technology, 2009.
[3] A. Kosmopoulos, G. Paliouras, and I. Androutsopoulos, “Adaptive Spam Filtering Using Only Naive Bayes Text Classifiers,” Email and Anti-Spam (CEAS 2008), CA, USA, August, 2008.
[4] C. Dietrich and C. Rossow, “Empirical research on IP blacklisting,” Email and Anti-Spam (CEAS 2008), CA, USA, August,2008.
[5] J. Jung and E. Sit, “An Empirical Study of Spam Traffic and the Use of DNS Black Lists,” in Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Sicily, Italy, October 25-27, 2004.
Bin Ning, Wu Junwei, Hu Feng, “Spam Message Classification Based on the Na¨ ıve Bayes Classification Algorithm”, IAENG International Journal of Computer Science, 1 February 2019
[6] T. Kulikova, T. Shcherbakova, Spam and phishing in 2021, 2022. URL: https://securelist.ru/spam-and-phishing-in-2021/104407/.
[7] S. R. Sahoo, B. B. Gupta, D. Peraković, F. J. G. Peñalvo, I. Cvitić, Spammer Detection Approaches in Online Social Network (OSNs): A Survey, in: L. Knapcikova, D. Peraković, M. Perisa, M. Balog (Eds.), Sustainable Management of Manufacturing Systems in Industry 4.0, EAI/Springer Innovations in Communication and Computing. Springer, Cham, 2022, pp. 159180. doi:10.1007/978-3-030-90462-3_11.
[8] T. Sudalaimuthu, C. Dheeraj Kumar Reddy, B. Sairam Reddy, M. Lakshmi Sahithya, S. Visalaxi, detecting spammer and fake user on social networks using machine learning approach, in: AIP Conference Proceedings, volume 2385, 050010, 2022. doi:10.1063/5.0071071.
[9] N. Liubchenko, A. Podorozhniak, V. Oliinyk, Research of antispam bot algorithms for social networks, in: CEUR Workshop Proceedings, volume 2870, 2021, pp. 822–831. URL: http://ceurws.org/Vol-2870/paper61.pdf.
[10] B. Liu, E. Blasch, Y. Chen, D. Shen, G. Chen, Scalable sentiment classification for Big Data analysis using Naïve Bayes Classifier, in: Proceedings of the IEEE International Conference on Big Data, USA, 2013, pp. 99–104. doi:10.1109/BigData.2013.6691740.
[11] S. Kaddoura, G. Chandrasekaran, D.A. Popescu, J.H. Duraisamy, A systematic literature review on spam content detection and classification, PeerJ Computer Science, 8, e830, 2022. doi:10.7717/PEERJ-CS.830.
[12] S. Chaudhry, S. Dhawan, R. Tanwar, Spam Detection in Social Network Using Machine Learning Approach, in: U. Batra, N. Roy, B. Panda (Eds.), Data Science and Analytics. REDSET 2019, Communications in Computer and Information Science, 2020, pp. 236–245. doi:10.1007/978-981-15-5830-6_20.
[13] A. Mykytiuk, V. Vysotska, S. Albota, Spam Filtration System with the Use of Machine Learning Technology, in: Proceedings of the International Scientific and Technical Conference on Computer Sciences and Information Technologies, Lviv, 2021, pp. 124–130.