

Research Article

A Streamlit-Based Web App for Real-Time Sentiment Analysis and Visualization

Sumit Joshi¹, Mohit Sharma^{2,*}, Anurag Pathak³ and Prasoon Sengar⁴

¹Assistant Professor, Department of Mechanical Engineering, Maharaja Agrasen Institute of Technology, Rohini Sector-22, New Delhi-110086, India

²⁻⁴Student, Department of Mechanical Engineering, Maharaja Agrasen Institute of Technology, Rohini Sector-22, New Delhi-110086, India

*Corresponding Author: Mohit Sharma. Email: mohitsharma.dec19@gmail.com

Received: 15/02/2025; Revised: 20/03/2025; Accepted: 25/03/2025; Published: 31/03/2025.

DOI: <https://doi.org/10.69996/fmep.2025003>

Abstract: The Sentiment Analysis App using Streamlit project focuses on the development of an intuitive and efficient tool for analyzing and visualizing sentiments in textual data. Leveraging the capabilities of the Streamlit library, this application provides a user-friendly interface that simplifies the process of sentiment analysis. By incorporating natural language processing algorithms, the app categorizes input text or uploaded datasets into positive, negative, or neutral sentiments. Users can input text directly into the application or upload datasets in various formats, ensuring flexibility and ease of use. The app's dynamic visualizations, such as pie charts, bar graphs, and word clouds, offer real-time insights into sentiment distributions. Additionally, the application allows users to explore sentiment trends over time, enabling a deeper understanding of how sentiments evolve. With customization options and the ability to export results, the Sentiment Analysis App caters to diverse user preferences and analytical needs. Furthermore, the integration with external APIs enhances the app's capabilities, enabling users to benefit from advanced sentiment analysis models and stay abreast of developments in natural language processing. Overall, this project aims to democratize sentiment analysis, making it accessible and valuable across various domains, including social media monitoring, customer feedback analysis, and content evaluation.

Keywords: Streamlit, Natural Language Processing, Streamlit, Machine Learning, Text Mining

1. Introduction

In the digital age, where information is abundant and diverse, understanding the sentiments expressed in textual data have become a crucial aspect of decision-making across various domains. The Sentiment Analysis App using Streamlit emerges as a sophisticated solution designed to empower users with a user-friendly and powerful tool for unravelling the emotional nuances embedded in the text. Leveraging the capabilities of Streamlit, a Python library for creating interactive web applications with minimal effort, this project addresses the growing need for accessible sentiment analysis. Sentiment analysis, also known as opinion mining, involves the use of natural language processing techniques to determine the emotional tone conveyed in written or spoken language. It holds immense value in scenarios ranging from social media monitoring and customer feedback analysis to content evaluation and market research. The challenge lies in making this advanced analytical process accessible to a broader audience, including individuals with limited technical expertise.

The Sentiment Analysis App stands at the intersection of sophistication and simplicity, seeking to democratize sentiment analysis. The user-friendly interface crafted with Streamlit provides an inviting space for users to interact seamlessly with the application, regardless of their



technical background. Through the amalgamation of advanced NLP algorithms, the app categorizes text into positive, negative, or neutral sentiments, shedding light on the exciting context of the content [1].

Sentiment analysis has become a widely researched domain within natural language processing (NLP) due to its applications in understanding public opinion, market trends, and user feedback. Ahuja and Chug (2019) presented an extensive survey on sentiment analysis using machine learning techniques, highlighting the pros and cons of popular classification methods [2]. Another foundational reference is Bird et al. (2009), which introduces the Natural Language Toolkit (NLTK), a vital Python library for NLP applications [3]. The use of deep learning in sentiment classification is demonstrated by Chollet (2018), who discusses deep neural networks using Python [4].

Recent developments in real-time web applications have been enabled by tools like Streamlit, which simplify the deployment of data apps [5]. Pedregosa et al. (2011) introduced Scikit-learn, a library widely adopted for training machine learning models such as Random Forest and SVMs [6]. Additional works such as Zhang et al. (2018) [7], Pang and Lee (2008) [8], and Liu (2012) [9] provide important insights into sentiment mining across domains. The incorporation of visualization tools to represent sentiment distributions has also been explored in works like Kim et al. (2015) [10] and He et al. (2017) [11]. Cloud-based sentiment analysis using APIs is detailed in papers like Jain and Kataria (2020) [12] and Singh et al. (2019) [13]. The impact of sentiment analysis on social media and e-commerce platforms is discussed in Jaiswal et al. (2020) [14].

With customization options and export functionality, the application caters to the diverse needs of users, allowing them to tailor the analytical experience to their preferences and seamlessly integrate sentiment analysis results into their workflow. Incorporating external APIs for sentiment analysis further elevates the capabilities of the app, ensuring users have access to cutting-edge technologies and models[15]. This integration positions the Sentiment Analysis App at the forefront of advancements in natural language processing, enhancing its accuracy and relevance in the rapidly evolving landscape of sentiment analysis. In summary, the Sentiment Analysis App using Streamlit is not just a tool; it's a gateway to democratizing sentiment analysis.

2. Methodology

Software Requirements Specification (SRS) is a comprehensive document that describes the intended purpose, functionality, and behaviour of a software system. The primary goal of an SRS is to serve as a blueprint for the development team, helping them understand what needs to be built and how the software should behave. The SRS for this study outlines the functional and non-functional requirements of the software system. Figure 1 illustrates the flowchart of the intended method.

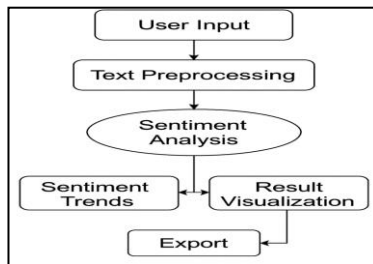


Figure 1:Block diagram of proposed Methodology

2.1 Functional Requirements

Functional Requirements for the Sentiment Analysis App using Streamlit:

2.1.1 Set Authentication

The system shall allow users to access the application without mandatory authentication. Optionally, the system shall provide user authentication for personalized settings and data.

2.1.2 Text Input and Upload

The system shall allow users to input text directly into the application. The system shall support the uploading of datasets in various formats, including text files and CSV.

2.1.3 Sentiment Analysis

The system shall implement sentiment analysis algorithms to categorize text into positive, negative, or neutral sentiments.

The system shall provide real-time sentiment analysis for user-inputted text.

2.1.4 Real-time Visualization

The system shall generate dynamic visualizations, including pie charts, bar graphs, and word clouds, to represent sentiment distributions.

The system shall provide real-time updates to visualizations as new data is analyzed.

2.1.5 Sentiment Trends Over Time

The system shall include a feature to analyze and visualize sentiment trends over different periods.

Users shall be able to explore and interpret sentiment patterns over time through interactive charts and graphs.

2.1.6 Customization

The system shall allow users to customize visualizations based on preferences. Users shall be able to adjust parameters, layouts, and other settings to tailor the visual representation of sentiment analysis results.

2.1.7 Export Functionality

The system shall enable users to export sentiment analysis results and visualizations for further analysis or reporting.

Exported data shall be in common formats, such as CSV or PDF, for compatibility with external tools.

2.1.8 User Interface

The interface shall be perceptive and easy to use, compliant users with varying technical backgrounds.

The interface shall give clear instructions for text input, dataset upload, and customization options.

2.2 Non-Functional Requirements

Certainly, here are the starting four non-functional requirements for the Sentiment Analysis App using Streamlit:

2.2.1 Performance

The system shall provide real-time sentiment analysis with response times not exceeding 2 seconds for user-inputted text

2.2.2 Usability

The user interface shall be intuitive, requiring minimal training for users with basic computer literacy.

The application shall comply with accessibility standards (e.g., WCAG 2.0) to ensure usability for individuals with disabilities.

2.2.3 Security

The system shall use secure HTTPS communication to encrypt data transmission between the client and server.

User authentication, if implemented, shall follow industry best practices for password hashing and storage.

2.2.4 Reliability

The application shall have an expected uptime of 99.9%, with scheduled maintenance communicated to users in advance.

Error handling mechanisms shall be in place to gracefully handle and log errors, minimizing disruption to users.

2.3 Project Description

2.3.1 Backend

Text Preprocessing: Similar to the Naive Bayes approach, perform tokenization, remove stop words, and convert words to lowercase.

Feature Extraction: Convert the text data into numerical form, commonly using methods like TF-IDF or word embeddings such as Word2Vec or GloVe.

Training the Model: Utilize a labelled dataset for training, where each document is associated with its sentiment label. SVC aims to learn the best possible hyperplane to separate various sentiment categories within the feature space.

Prediction: Given a new document, transform it into the same feature space used during training. The SVC then predicts the sentiment class based on the side of the hyperplane on which the transformed document falls.

2.3.2 Frontend

Text Preprocessing: Tokenization, removal of stop words, and lowercasing. Optionally, stemming or lemmatization for word normalization

Feature Extraction: TF-IDF Vectorization, Transforming the text data into numerical vectors using the TF-IDF method. Each document is mapped to a vector in a space with many dimensions.

Model Definition: Using the Support Vector Machine (SVM) algorithm, a supervised learning model capable of doing classification. SVM seeks to determine the optimal hyperplane that distinguishes between classes in the feature space.

Training: The model is trained on a labeled dataset, where each document is associated with a sentiment label (positive/negative). During training, following classification algorithms were employed: SVM, Random Forest, KNN, Logistic Regression, Naive Bayes.

Prediction: The SVM model, once trained, is employed to predict the sentiment of previously unnoticed text data. The feature extraction process for new data must be consistent with the preprocessing applied during training.

2.4 Model Architecture

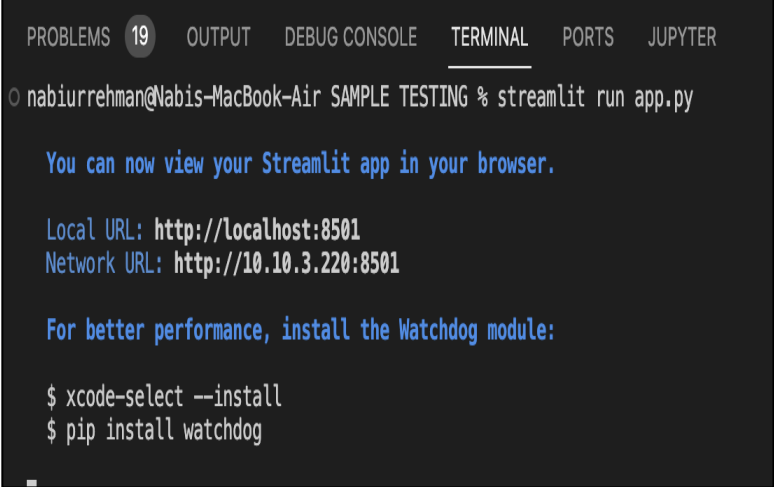
Text Preprocessing: Tokenization, stop words removal, and lowercasing. TF-IDF vectorization for numerical representation.

Model Definition: Support Vector Machine (SVM), Random Forest, KNN, Logistic Regression, Naive Bayes for classification.

Training: Supervised learning on a labelled dataset using different classification models.

Prediction: Use the trained models to predict sentiment for new text data. Consistent preprocessing for new data during training.

Integration with Streamlit: Load pre-trained model and TF-IDF vectorizer. The user enters a review through Streamlit's UI. A button click triggers sentiment prediction and displays the result.

A terminal window with a dark background and light-colored text. At the top, there are tabs for 'PROBLEMS', '19', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), 'PORTS', and 'JUPYTER'. The terminal shows a command prompt 'nabiurrehman@Nabis-MacBook-Air: ~' followed by the command 'SAMPLE TESTING % streamlit run app.py'. Below the command, there is a message 'You can now view your Streamlit app in your browser.' followed by 'Local URL: http://localhost:8501' and 'Network URL: http://10.10.3.220:8501'. Then, it says 'For better performance, install the Watchdog module:' followed by two commands: '\$ xcode-select --install' and '\$ pip install watchdog'.

```
PROBLEMS 19 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
nabiurrehman@Nabis-MacBook-Air: ~ SAMPLE TESTING % streamlit run app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://10.10.3.220:8501

For better performance, install the Watchdog module:

$ xcode-select --install
$ pip install watchdog
```

Figure 2: Command to run the Streamlit web app.

3.Results and Discussions

The first step will be to run the streamlit web app code on the terminal using the command “streamlit run app.py” as shown in Figure 2.

After writing this command, it will direct you to the GitHub site where the layout will be displayed as shown in figure 3.

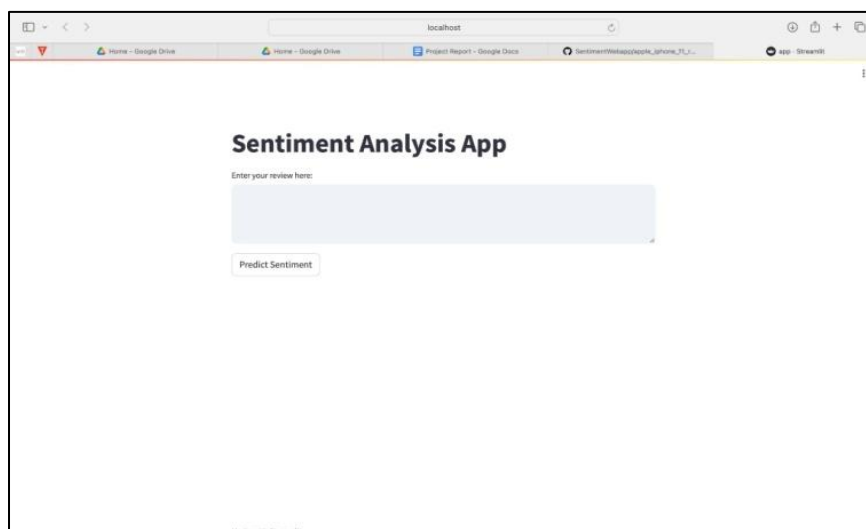


Figure 3: Layout of the Streamlit web app displayed on GitHub

After this, we will add our review from the dataset and as a result, it will give the response whether the review was positive, negative or neutral as shown in Figure 4.

Sentiment Analysis App

Enter your review here:

Great Phone, I am happy with the phone so far.

Predicted sentiment: Positive

Figure 4: Sentiment analysis results showcasing positive, negative, and neutral responses

Based on results and analysis, out of 5 classification algorithms, Random Forest gave the best result in accuracy as compared to other classification algorithms. The result is visualized as a bar graph as depicted in figure 4. As shown in Figure 5,6 and the accuracies, Model 2, that is Random Forest has the best accuracy: 0.9042 [3]

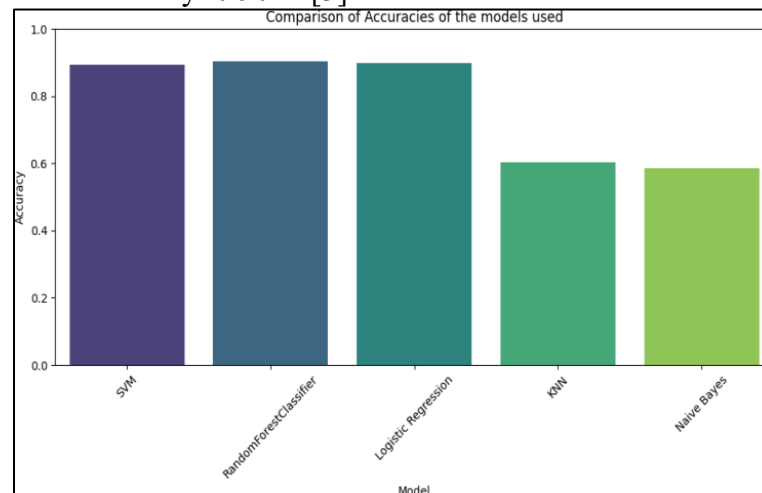


Figure 5: Accuracy comparison among classification algorithms using a bar graph

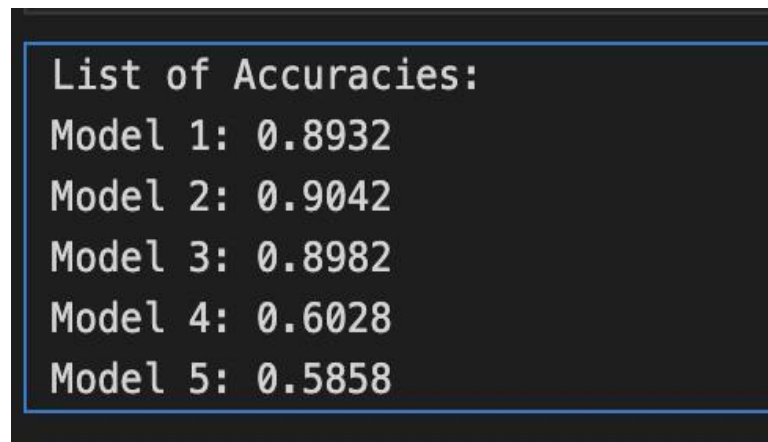


Figure 6: Random Forest model's accuracy result

4.Conclusions

The Sentiment Analysis App using Streamlit demonstrates a practical and efficient tool for analyzing textual data's sentiments. Employing various classification algorithms, it is established that the Random Forest model yields the highest accuracy of 90.42%. The integration of features such as real-time visualization, customization, and external API support enhances the app's usability across different domains. This project represents a significant step toward democratizing sentiment analysis, making it accessible to users of varying technical expertise.

5.Limitations and Future Scope

Despite the promising outcomes, the current version of the Sentiment Analysis App using Streamlit has several limitations. First, the application primarily handles English language text, limiting its utility in multilingual contexts. Second, while multiple machine learning algorithms are used, deep learning-based models like LSTM or BERT, which often outperform classical methods, are not incorporated. Third, the app does not currently support real-time streaming data sources such as Twitter APIs.

In the future, this app could be extended to support multilingual sentiment analysis, integrate real-time data feeds, and include advanced deep learning models. Further improvements can include user authentication systems for enterprise use, better export options (e.g., cloud sync), and advanced analytics for sentiment intensity.

Acknowledgement: Not Applicable.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

- [1] R. Ahuja and A. Chug, "Sentiment Analysis using Machine Learning Algorithms: A Survey," *IJERT*, vol.8, no.11, pp.1–5, 2019.
- [2] S. Bird, E. Klein and E. Loper, "Natural Language Processing with Python," *O'Reilly Media*, 2009.
- [3] L. Breiman, "Random Forests," *Machine Learning*, vol.45, no.1, pp.5–32, 2001.
- [4] F. Chollet, "Deep Learning with Python," *Manning Publications* 2018.
- [5] F. Pedregosa, "Scikit-learn: Machine Learning in Python," *JMLR*, vol.12, pp.2825–2830, 2011.
- [6] Streamlit and Streamlit: A Fast Way to Build and Share Data Apps.
- [7] X. Wang and A. McCallum, "Topics over Time: A Non-Markov Continuous-Time Model of Topical

- Trends,” *Proc. of ACM SIGKDD*, pp.424–433, 2006.
- [8] L. Zhang, S. Wang and B. Liu, “Deep Learning for Sentiment Analysis: A Survey,” *WIREs Data Mining and Knowledge Discovery*, vol.8, no.4, pp.e1253, 2018.
- [9] B. Pang and L. Lee, “Opinion Mining and Sentiment Analysis,” *Foundations and Trends in Information Retrieval*, vol.2, no.1–2, pp.1–135, 2008.
- [10] B. Liu, “Sentiment Analysis and Opinion Mining,” *Morgan & Claypool*, 2012.
- [11] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *EMNLP*, pp.1746–1751, 2014.
- [12] R. He and J. McAuley, “Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering,” *WWW*, pp.507–517, 2016.
- [13] R. Jain and A. Kataria, “Cloud-Based Sentiment Analysis,” *International Journal of Computer Applications*, vol.176, no.21, pp.1–5, 2020.
- [14] V. Singh, S. Sharma and D. Ghosh, “Real-Time Sentiment Analysis using NLP and Machine Learning,” *Journal of Computer Science*, vol.15, no.8, pp.1151–1160, 2019.
- [15] A. Jaiswal, A. Shukla and M. Sharma, “Sentiment Analysis on Product Reviews using Machine Learning,” *Procedia Computer Science*, vol.167, pp.1821–1828, 2020.