ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Spam and Bot Detection Across Multiple Communication Channels Using Machine Learning

M. Thoshanth, N. Lahari, N. Pranav, M. Sanjay, Dharmendra Roy

Department of Computer Science and Engineering, Hyderabad Institute of Technology and Management (HITAM), Hyderabad, Telangana, India

Published November 2026

Abstract

Digital communication platforms face a growing volume of spam, fraudulent content, and automated social media accounts, and the data available differ from one channel to another. This paper presents a multi-channel system that classifies Email and Short Message Service (SMS) messages as spam or legitimate (ham) and identifies suspicious automated accounts on Twitter (X). Email and SMS text is processed using Natural Language Processing (NLP) preprocessing and Term Frequency–Inverse Document Frequency (TF-IDF) features, which are classified by Random Forest models. Twitter accounts are analyzed separately from account metadata, namely follower count, following count, account age, and tweet volume, using a distinct Random Forest classifier. A FastAPI backend provides real-time prediction, batch processing, and input validation, and a web-based interface returns the predicted class, a confidence value, and dynamically generated explanations. The system reports an overall classification accuracy of 94.25% and an average prediction latency of 32 ms per request. The work shows how text-based spam classification and metadata-based bot detection can be integrated within a single application, and it identifies the evaluation details that remain to be documented.

Keywords: bot detection; FastAPI; machine learning; natural language processing; random forest; spam detection; TF-IDF; Twitter

1 Introduction

Email, text messaging, and social media are central to personal, commercial, and institutional communication. Their wide adoption has also created opportunities for unsolicited advertising, fraudulent messages, malicious links, and automated social media activity. Spam can mislead recipients into disclosing information or visiting unsafe destinations, while automated accounts can amplify content and imitate ordinary user behavior.

Traditional filters often depend on fixed keyword lists or manually defined rules. Such rules are easy to interpret, but their effectiveness declines when message wording changes or senders deliberately alter terms to avoid known patterns. Machine-learning classifiers offer an alternative by learning patterns from labeled examples. The data available on each channel, however, differs in structure. Email and SMS spam detection is primarily a text-classification task, whereas Twitter bot detection can draw on account-level information such as follower count, following count, account age, and tweet volume. A single feature representation therefore does not suit all inputs.

This paper describes an application that combines three channel-specific detection options behind one interface: Email spam detection, SMS spam detection, and Twitter account bot detection. Email and SMS messages follow a text pipeline, and Twitter accounts follow a metadata pipeline. A FastAPI backend validates requests, supports real-time and batch prediction, and returns a predicted class, a

confidence value, and dynamically generated reasons to a web-based frontend. The contributions of this work are: (1) the integration of three detection options in one application; (2) TF-IDF and Random Forest classification for Email and SMS; (3) metadata-based Random Forest classification for Twitter accounts; and (4) a web interface offering real-time inference, batch processing, input validation, confidence values, and generated explanations.

The remainder of this paper reviews related work, states the problem and objectives, presents the proposed system and the methodology, describes the implementation and interface, discusses the reported results and limitations, and concludes with future work.

2 Related work

The Enron corpus released by Klimt and Yang [1] gave researchers a common collection of email on which classification methods could be developed and compared. It is relevant to the Email component of the present system, although its scope is limited to email.

Almeida *et al.* [2] assembled a new SMS collection and reported filtering results on it, which gave later work a basis for separating spam from legitimate short messages. The present work likewise treats SMS as a text-classification channel.

Benevenuto *et al.* [3] approached Twitter spammer detection through attributes of users and of the tweets they post, which shows the value of platform-specific characteristics when judging suspicious activity. The present system applies a related idea at the account level, using structured metadata, and does not implement a Twitter-post spam classifier.

Varol *et al.* [4] examined how humans and bots interact online, combining user, content, network, and activity characteristics to detect, estimate, and characterize bot accounts. Their work shows the benefit of drawing on several signals to identify automated behavior.

Table 1 summarizes the reviewed studies. Within them, each addresses a single data source and detection problem. The gap considered here is the absence of one interface that combines Email spam detection, SMS spam detection, and Twitter account bot detection. This work addresses that integration requirement at the application level, using separate pipelines rather than forcing all inputs into one representation. It does not claim that the underlying algorithms are new.

Table 1. Summary of related work.

Ref.	Area and contribution	Scope relative to this work
[1]	Enron corpus for email classification research	Email only
[2]	SMS spam collection and filtering results	SMS only
[3]	Twitter spammer detection using user and tweet characteristics	Twitter only
[4]	Online human–bot detection and characterization	Bot-focused; no Email or SMS classification

3 Problem statement and objectives

Email and SMS messages must be classified as spam or ham primarily from their textual content. Twitter bot detection requires a different input representation, because a single message may not

reveal whether an account is automated, whereas account-level metadata can provide behavioral indicators. A practical system must therefore route each input to a suitable pipeline while giving the user a consistent experience. The problem addressed here is the design and implementation of a unified application that (i) classifies Email and SMS messages as spam or ham from text; (ii) classifies Twitter accounts as bot or non-bot from account metadata; (iii) supports real-time and batch inference; and (iv) returns predictions with confidence values and dynamically generated reasons.

The objectives are to develop a unified web application for Email, SMS, and Twitter account analysis; to apply NLP preprocessing and TF-IDF features to Email and SMS text; to use Random Forest classifiers for both text channels; to extract account-level metadata and classify it with a separate Random Forest model for Twitter; to provide real-time prediction, input validation, and batch processing through a FastAPI backend; and to present confidence values and generated explanations with each prediction.

The scope comprises the three implemented detection options above. The web interface also lists a selection for X/Twitter post spam detection; that option is not an implemented detection pipeline and is outside the scope of this paper.

4 Proposed system

The proposed system is a unified application with three channel-specific detection options. Email and SMS inputs are treated as text and processed through NLP preprocessing, TF-IDF feature extraction, and Random Forest classification. Twitter account inputs are represented by structured account metadata and classified by a separate Random Forest model. The three pipelines converge at a common result-handling stage. Table 2 summarizes the pipelines.

Table 2. Implemented detection pipelines.

Option	Input	Representation	Classifier
Email	Message text	NLP preprocessing; TF-IDF vector	Random Forest (spam / ham)
SMS	Message text	NLP preprocessing; TF-IDF vector	Random Forest (spam / ham)
Twitter account	Account metadata	Structured numerical and Boolean features	Separate Random Forest (bot / non-bot)

4.1 Email and SMS spam detection

For each text channel, the submitted message is preprocessed and transformed into a TF-IDF feature vector. The vector is passed to the Random Forest classifier of that channel, which predicts whether the message is spam or ham.

4.2 Twitter account bot detection

The Twitter option analyzes account-level information rather than the text of an individual post. The attributes include follower count, following count, the relationship between followers and following, account age, and tweet volume. These features form the input to a separate Random Forest classifier, which predicts bot or non-bot.

4.3 Prediction and explanation

The application returns a predicted class together with a confidence or probability value, and it generates human-readable reasons from available indicators. Regular-expression-based detection of URLs, mentions, and hashtags serves as a supporting indicator when interpreting suspicious text patterns. These indicators are explanatory aids and do not replace the Random Forest decision.

4.4 System architecture

The system follows a frontend–backend architecture, shown in Figure 1. A user submits an Email message, an SMS message, or Twitter account information through the web frontend. The frontend sends the request to the FastAPI backend, which validates it and routes it to the relevant pipeline. The pipeline performs feature processing and invokes its trained Random Forest model, and the prediction, confidence information, and generated reasons are returned to the frontend for display.

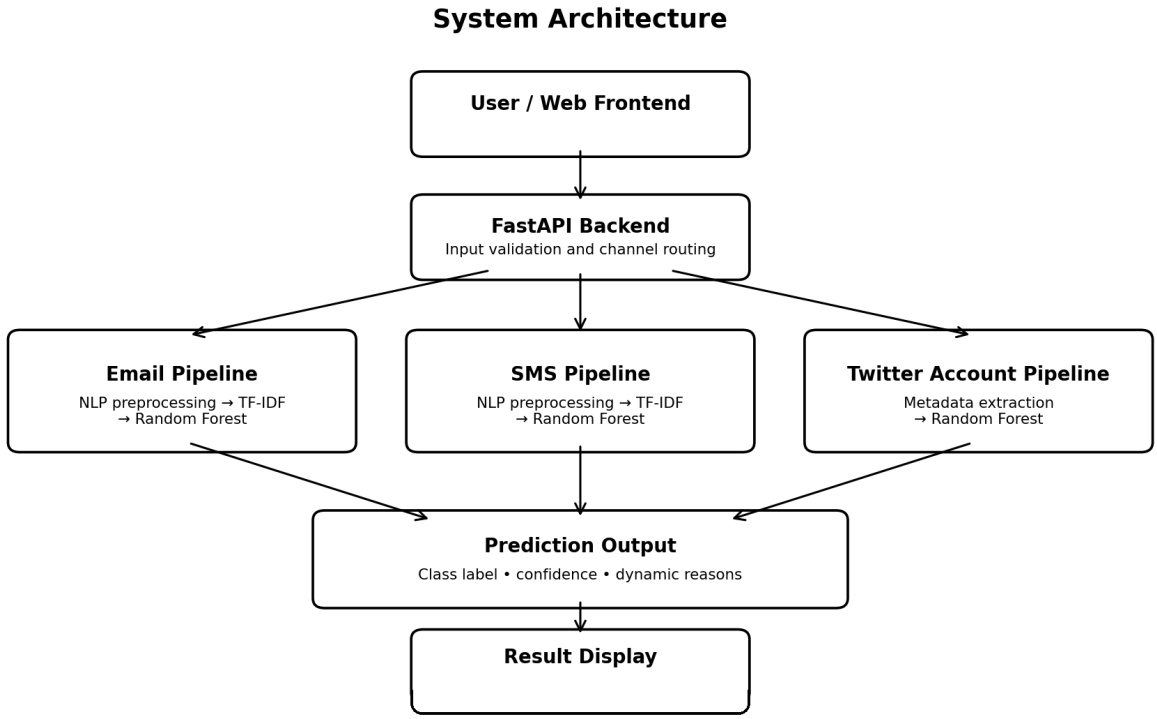


Figure 1. System architecture of the implemented multi-channel detection application.

5 Methodology

5.1 Processing workflow

The methodology consists of input submission, validation and routing, channel-specific feature processing, Random Forest classification, generation of the prediction and explanation, and display of the result (Figure 2).

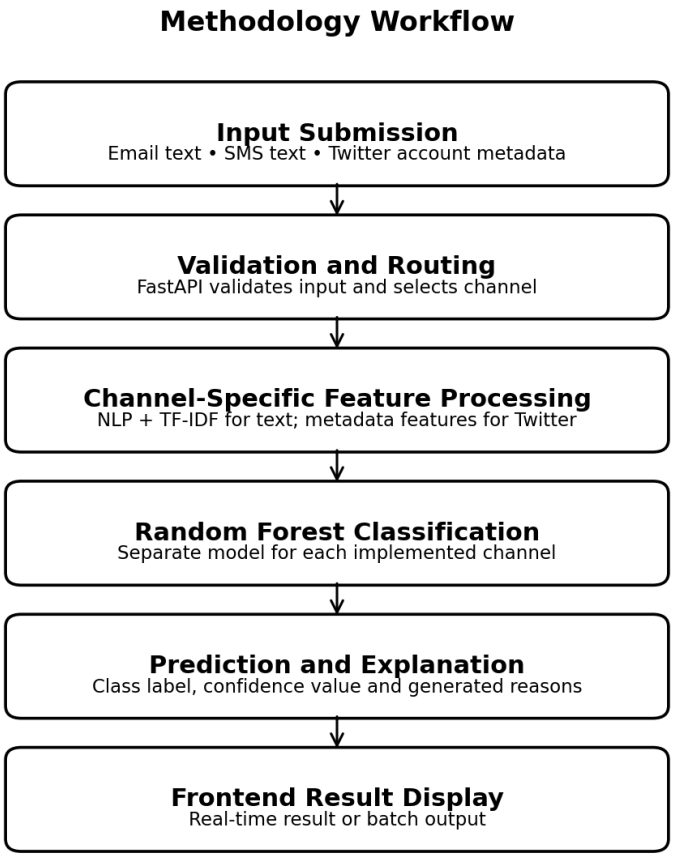


Figure 2. Processing workflow followed by the proposed system.

5.2 Data preparation

The available project documentation does not specify the training corpora, dataset sizes, class distributions, train/test split, or data-cleaning procedure. These details are therefore not stated in this paper.

5.3 Text preprocessing

Email and SMS text passes through an NLP preprocessing stage that prepares the text for feature extraction. The individual preprocessing operations and their configuration are not specified in the project documentation and are therefore not described here.

5.4 TF-IDF feature extraction

TF-IDF converts a collection of documents into numerical feature vectors that reflect the importance of each term. Term frequency measures the relative occurrence of a term within a document, and inverse document frequency reduces the influence of terms that occur in many documents. In their general form,

$$TF(t,\ d) = \frac{f(t,\ d)}{\sum_{v} f(v,\ d)} \tag{1}$$

$$IDF(t) = \log\left(\frac{N}{df(t)}\right) \tag{2}$$

$$TF\text{-}IDF(t,\ d) = TF(t,\ d) \times IDF(t) \tag{3}$$

where $f(t,d)$ is the frequency of term t in document d, N is the number of documents in the collection, and $\mathrm{df}(t)$ is the number of documents containing t. These equations describe the general representation; the vectorizer settings used in the implementation are not documented.

5.5 Twitter metadata features

The Twitter pipeline uses structured numerical and Boolean metadata instead of a TF-IDF representation. The features include follower count, following count, the follower-to-following relationship, account age, and tweet volume, and they are organized into a feature vector for the Twitter Random Forest classifier. Feature transformations, missing-value treatment, and scaling are not specified in the available material.

5.6 Supporting pattern indicators

Regular-expression-based detection of URLs, mentions, and hashtags can contribute contextual reasons that accompany a text prediction. These are supporting signals only, since the class label is produced by the corresponding Random Forest model. The fidelity and usefulness of the generated explanations have not been systematically evaluated.

5.7 Random Forest classification

Random Forest is an ensemble method that combines the outputs of multiple decision trees. For an input feature vector $\mathbf{x}$, each tree k produces a class prediction $h_k(\mathbf{x})$, and the ensemble selects the class that receives the most votes:

$$\hat{y} = \mathrm{mode}\left\{h_k(x) : k = 1,\ \ldots,\ K\right\} \tag{4}$$

where K is the number of trees. Email and SMS text features are classified by Random Forest models, and Twitter metadata is classified by a separate Random Forest model. The number of trees, maximum depth, split criterion, class weights, and other hyperparameters are not specified in the project documentation and are not assumed here.

6 Implementation and interface

The backend is implemented with FastAPI and exposes prediction functionality to the web frontend. For each request, the backend receives the input, validates it, identifies the selected channel, and invokes the corresponding pipeline, which prepares the features, applies the trained Random Forest model, and returns the class, confidence information, and generated reasons. The system supports individual real-time prediction and batch processing, and input validation is applied before inference. A complete API specification, deployment configuration, and frontend technology stack are not documented, so they are not described here.

The web application has four sections: Dashboard, Detect, Datasets, and Analytics. Figure 3–7 show screenshots of these sections and are included to document the interface.

6.1 Dashboard

The Dashboard (Figure 3) provides an overview of detection metrics through summary cards for total samples, spam detected, ham detected, bot accounts, model accuracy, average confidence, and false positive rate, followed by a short welcome note that directs users to the Detect page.

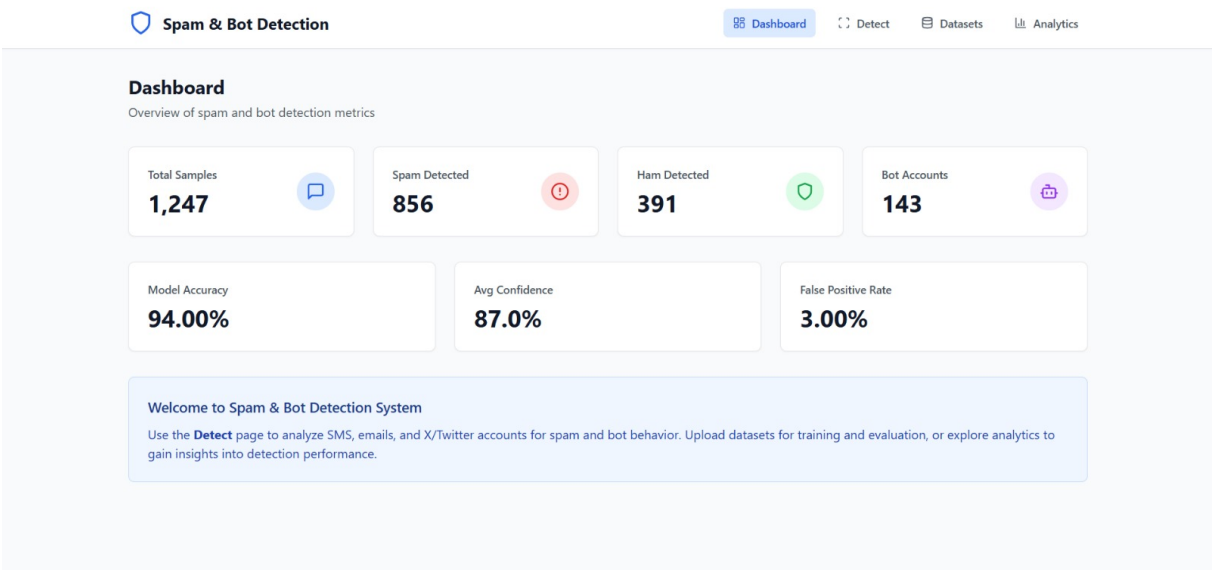


Figure 3. Dashboard page of the web application showing the overview metric cards.

6.2 Detection interface

The Detect page (Figure 4) presents a "Select Detection Type" panel followed by an "Input Data" panel whose fields depend on the selected option. In the screenshot, SMS is selected, and the panel shows a text area for the SMS message. The panel also lists an X/Twitter post option, which, as stated in the problem statement, is not an implemented detection pipeline.

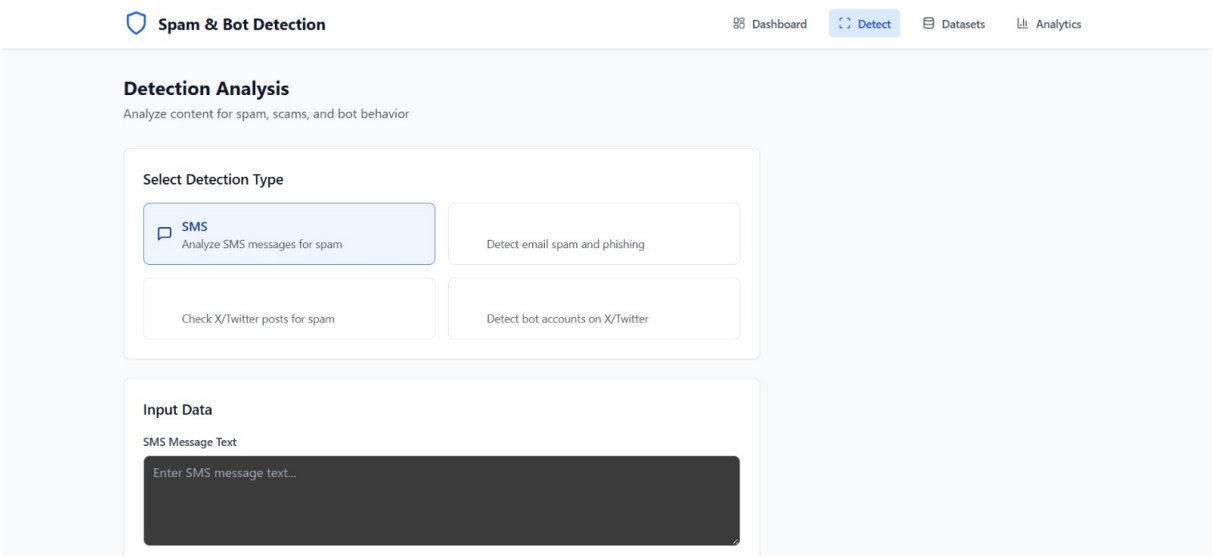


Figure 4. Detect page with the SMS option selected and the SMS message input area.

Figure 5 shows the Email option, which accepts a subject line, an email body, and an optional sender address. For the example input, with the subject "Amazon" and a body announcing an order shipment, the result panel shows the class HAM, a LOW RISK label, a risk score of 0/100, a confidence of 86.2%,

and the generated reason "High ham probability (86%)." It also shows the model identifier (email_classifier v1.0.0) and a processing time of 24 ms. This single example illustrates the output format and is not used as a performance measurement.

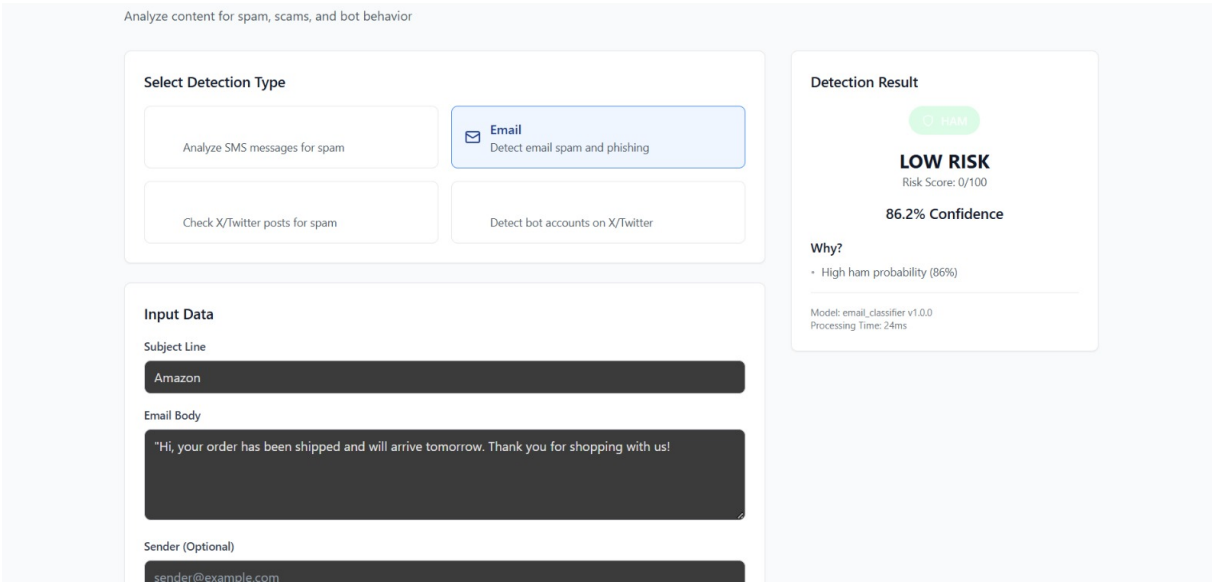


Figure 5. Email detection result showing class, risk level, confidence, generated reason, and model details.

6.3 Dataset management

The Datasets page (Figure 6) provides an upload option for datasets intended for training and evaluation. At the time of capture, no dataset had been uploaded, and the page displays an empty state. The upload guidelines specify CSV files up to 50 MB: for SMS and Email, a text column and a label column with the values HAM or SPAM; and for X accounts, account metrics with a label of HUMAN, BOT, or SUSPICIOUS. The guidelines also list a format for X posts, which corresponds to the option that is outside the implemented scope.

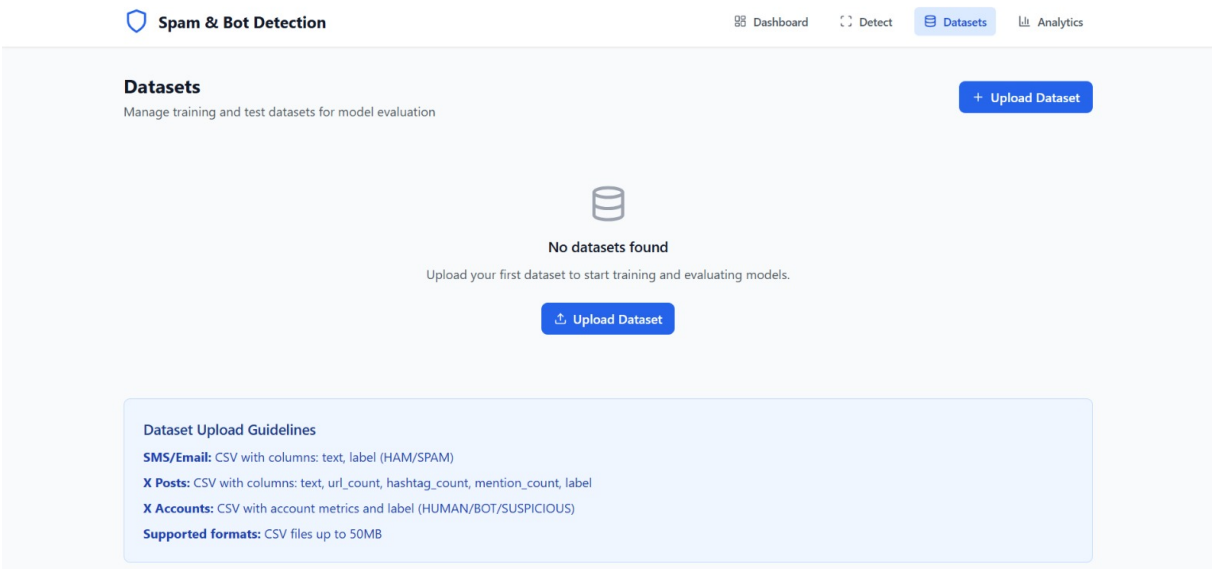


Figure 6. Datasets page with the upload option and dataset upload guidelines.

6.4 Analytics

The Analytics page (Figure 7) combines summary cards for total detections, average accuracy, false positive rate, and processing speed, each with a month-over-month comparison, with a bar chart titled “Detections by Channel” and a pie chart titled “Risk Level Distribution.” The values rendered on the Dashboard and Analytics pages are interface outputs and are not treated as evaluation results in this paper.

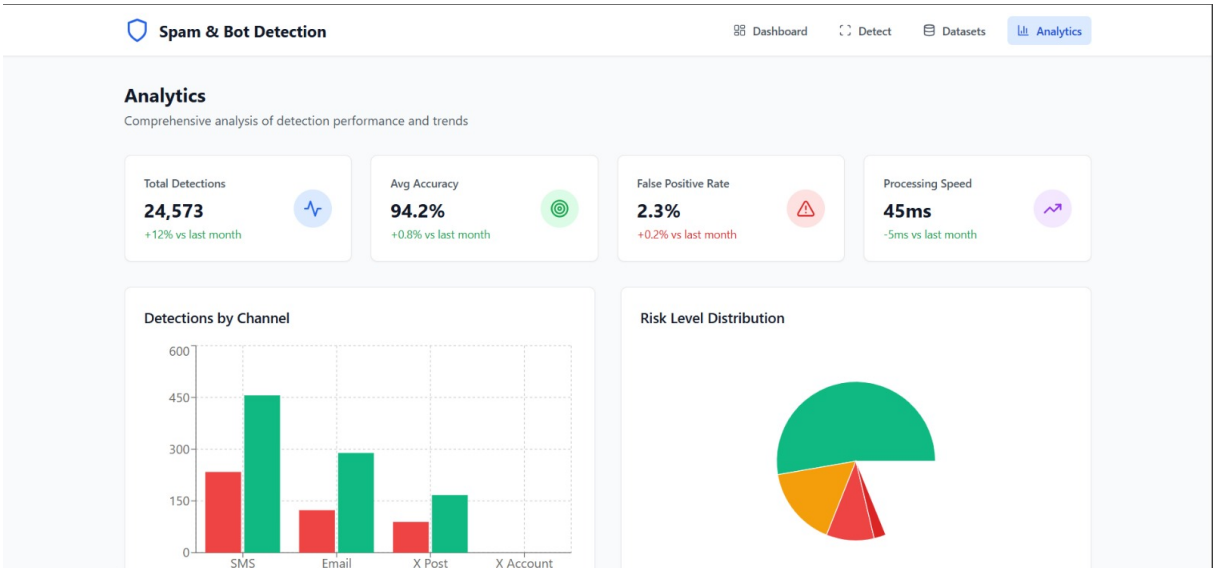


Figure 7. Analytics page with summary cards, the “Detections by Channel” bar chart, and the “Risk Level Distribution” pie chart.

7 Results and discussion

The project reports an overall classification accuracy of 94.25% and an average prediction latency of 32 ms per request (Table 3). These are the project-level results of the system, and they are the only performance figures used in this paper.

Table 3. Reported system performance.

Metric	Reported value
Overall classification accuracy	94.25%
Average prediction latency	32 ms per request

The documentation does not provide the evaluation protocol, test-set size, per-channel accuracy, confusion matrices, precision, recall, F1-score, hardware configuration, or latency measurement procedure. The reported values therefore cannot be independently reproduced from the available material, and they do not show that one channel performs better than another. The latency figure indicates that the inference flow is intended for interactive use, but without information about the test environment, request size, and concurrency it should not be read as a benchmark.

8 Limitations

Model performance depends on the representativeness and quality of the training data, and performance on previously unseen spam and bot behavior has not been documented. TF-IDF captures lexical patterns but not full semantic context or message intent. The Twitter pipeline depends on the availability, quality, and freshness of account metadata, which may be incomplete or deliberately manipulated. Maintaining separate channel-specific models requires independent maintenance and evaluation, and the supporting pattern indicators may not cover every suspicious pattern. Finally, the reported evidence is limited to overall accuracy and average latency. Per-channel and class-wise metrics, dataset details, hyperparameters, baseline comparisons, confidence calibration, and explanation quality have not been documented.

9 Conclusion and future work

This paper presented a multi-channel system that integrates Email spam detection, SMS spam detection, and Twitter account bot detection within one web application. Email and SMS messages are classified using NLP preprocessing, TF-IDF features, and Random Forest models, and Twitter accounts are classified from account metadata using a separate Random Forest model. A FastAPI backend supports validation, real-time prediction, and batch processing, and the frontend displays predicted classes, confidence values, and generated reasons. The project reports an overall accuracy of 94.25% and an average latency of 32 ms per request. Because the evaluation procedure and per-channel results are not documented, these figures should be interpreted with caution.

Future work includes training on broader and more recent datasets, evaluating on previously unseen spam and bot patterns, and extending text features with contextual or semantic representations. The Twitter pipeline could be evaluated with additional account-level and behavioral attributes, subject to data availability. Further evaluation should report per-channel precision, recall, F1-score, confusion matrices, dataset composition, the train/test methodology, model hyperparameters, and repeatable latency measurements, together with studies of confidence calibration and explanation quality. These are proposed extensions and are not features of the current implementation.

Acknowledgment

The authors thank Dr. Dharmendra Roy for his guidance and support during the project.

References

[1] Klimt B, Yang Y. The Enron corpus: A new dataset for email classification research. In: *Proc*. 15th Eur. Conf. Mach. Learn. (ECML); 2004. p. 217–226. https://doi.org/10.1007/978-3-540-30115-8_22

[2] Almeida TA, Hidalgo JMG, Yamakami A. Contributions to the study of SMS spam filtering: New collection and results. In: *Proc*. 11th ACM Symp. Document Eng. (DocEng); 2011. p. 259–262. https://doi.org/10.1145/2034691.2034742

[3] Benevenuto F, Magno G, Rodrigues T, Almeida V. Detecting spammers on Twitter. In: *Proc*. 7th Annu. Collaboration, Electron. Messaging, Anti-Abuse and Spam Conf. (CEAS); 2010.

[4] Varol O, Ferrara E, Davis CA, Menczer F, Flammini A. Online human-bot interactions: Detection, estimation, and characterization. In: *Proc*. 11th Int. AAAI Conf. Web Social Media (ICWSM); 2017. p. 280–289. https://doi.org/10.1609/icwsm.v11i1.14871