

Machine Learning-Based Email Spam and Phishing Detection on Raspberry Pi Edge Devices

Ghandhamalla Harshitha
M.Tech, Embedded Systems
CVR College of Engineering
Hyderabad, Telangana, India - 500086
harshithaghandhamalla@gmail.com

Malleswari Akurati
Senior Assistant Professor
Department of Electronics and Communication
CVR College of Engineering
Hyderabad, Telangana, India - 500086
mali.akurati@gmail.com

Abstract- This paper presents an edge-oriented email-security framework that combines classical machine learning, bio-inspired feature optimization, contextual transformer classification, and Raspberry Pi deployment. The conventional study evaluates Naive Bayes, Support Vector Machine, Random Forest, Decision Tree, and Multi-Layer Perceptron models, while Genetic Algorithm and Particle Swarm Optimization are used to reduce irrelevant features and support resource-efficient inference. A complementary software path uses a fine-tuned DistilBERT sequence classifier to distinguish Legitimate, Spam, and Phishing emails from sender, subject, and body fields. The browser interface reports class, confidence, risk level, and an explanation. Recorded software outputs show sample-level confidence values of 99.20%, 99.02%, and 98.77% for Legitimate, Spam, and Phishing examples, respectively. The embedded path uses TF-IDF with Naive Bayes for lightweight local inference and Gmail monitoring. In the recorded Raspberry Pi dashboard state, 427 messages were processed, including 369 spam and 58 safe messages. A saved validation confusion matrix contains 28 Legitimate, 48 Spam, and 24 Phishing samples on the diagonal for the compact validation split. The results establish end-to-end software and edge execution while emphasizing that sample confidence and dashboard counts should not be interpreted as unrestricted real-world benchmark accuracy.

Keywords- Email spam detection, phishing detection, Raspberry Pi, DistilBERT, Naive Bayes, TF-IDF, Genetic Algorithm, Particle Swarm Optimization, edge AI, Streamlit.

I. INTRODUCTION

Electronic mail remains a primary communication channel for academic, commercial, and personal activity, but its low delivery cost also enables unsolicited advertising, fraudulent messages, and credential-oriented phishing. Automated filtering is therefore a supervised text-classification problem in which incoming messages are transformed into numerical or contextual representations and mapped to security-relevant categories. Conventional spam filtering has been studied with Naive Bayes, Support Vector Machine, tree-based learners, neural networks, and hybrid feature-selection strategies [1]-[11]. The project considered here extends this line of work by combining a workstation-oriented contextual model with a compact Raspberry Pi inference path.

The framework has two complementary execution modes. The software mode concatenates sender, subject, and body fields, tokenizes the resulting sequence, and uses a fine-tuned DistilBERT classifier with three labels: Legitimate, Spam, and Phishing. The embedded mode emphasizes low-cost continuous monitoring and uses sparse TF-IDF features

with a lightweight Naive Bayes classifier. This division reflects the practical difference between contextual NLP experiments and repeated inference on constrained edge hardware. Scikit-learn, Python development tools, and machine-learning libraries used by the project are supported by established references and documentation [12]-[28], while Ling-Spam and Enron-Spam represent important public corpora in the broader spam-filtering literature [29], [30].

A) Contributions

The work provides: (1) a unified email-security pipeline covering input validation, preprocessing, model inference, and browser visualization; (2) comparative consideration of Naive Bayes, SVM, Random Forest, Decision Tree, and MLP with GA/PSO optimization; (3) a three-class DistilBERT software interface with confidence and risk mapping; (4) an edge path based on TF-IDF and Naive Bayes; and (5) Raspberry Pi Gmail monitoring with local browser presentation and result logging. The study separates functional evidence from statistical evidence so that interface confidence scores and dashboard message counts are not reported as global accuracy.

II. RELATED WORK

Early and conventional spam-filtering research demonstrates that probabilistic and discriminative methods remain effective for sparse email features. Feng et al. combined SVM processing with Naive Bayes [1], while broad reviews describe the continuing role of supervised learning and the challenges created by changing spam vocabulary [2]. Comparative email-classification studies have evaluated common machine-learning models under shared preprocessing conditions [3], [4]. Hybrid classification and optimization approaches further show that predictive quality depends on both the model and the representation supplied to it [5]-[11].

Feature optimization is especially relevant for edge deployment because email text can produce a high-dimensional vocabulary. Particle Swarm Optimization, Genetic Algorithm, Ant Colony Optimization, Firefly optimization, and related methods have been investigated for selecting informative features or improving classifier settings [6], [8]-[11]. These studies motivate the use of GA and PSO in the project as mechanisms for reducing irrelevant dimensions before compact inference.

The implementation also reflects the transition from sparse text models to contextual language models. Classical toolkits and textbooks provide the statistical-learning foundations used by the conventional path [12]-[14], [20]-[28]. The software path uses a distilled transformer because contextual representations can capture relationships across sender, subject, and body fields that are difficult to express with independent word counts. At the same time, the Raspberry Pi path retains a lighter model because model size, startup time, memory, and CPU latency are practical constraints for continuous local monitoring. Public spam corpora such as Ling-Spam and Enron-Spam illustrate the established benchmark ecosystem for broader future evaluation [29], [30].

A) Classical Spam Filtering and Feature Optimization

Classical email filtering remains relevant because sparse bag-of-words and TF-IDF vectors can be processed with low memory and predictable CPU cost. Naive Bayes is particularly attractive for word-frequency data because its probability estimates are simple to compute, while Support Vector Machine provides a discriminative large-margin alternative. Random Forest and Decision Tree models introduce nonlinear decision rules, and Multi-Layer Perceptron provides a neural baseline. The project retains these alternatives because model suitability depends on the available corpus, feature representation, and deployment platform rather than on a single universally optimal classifier.

Bio-inspired optimization addresses a second problem: the feature space generated from email vocabulary can be much larger than the set of truly informative terms. Genetic Algorithm evaluates candidate feature subsets through selection, crossover, and mutation, whereas Particle Swarm Optimization updates a population of candidate positions according to personal and swarm-level search experience. In an edge setting, reducing the feature space can lower vectorization cost, reduce stored vocabulary size, and simplify repeated inference. The project therefore treats optimization as a resource-aware preprocessing stage rather than as an isolated accuracy-enhancement mechanism.

B) Contextual Classification and Edge NLP

Transformer-based sequence classifiers represent words according to their surrounding context and can therefore use relationships across sender identity, subject urgency, and body content. This is useful for phishing detection because suspicious intent is often distributed across several fields rather than expressed by a single keyword. The DistilBERT software path demonstrates this contextual approach with three output labels. Distillation reduces the size of the pretrained encoder relative to full BERT, but the model remains considerably heavier than a sparse TF-IDF classifier, particularly when initialization time and memory are considered on a single-board computer.

The dual-path architecture is consequently an engineering compromise. Context-rich analysis is retained for

workstation-oriented software experiments, while the Raspberry Pi path emphasizes low-complexity local monitoring. This design is consistent with the broader edge-NLP principle that predictive quality must be evaluated together with model size, latency, power, and maintainability. It also provides a clear experimental framework for future controlled comparison: the same fixed corpus can be processed by both paths while accuracy-resource trade-offs are measured independently.

III. SYSTEM DESIGN AND METHODOLOGY

A) Overall Architecture

The system receives email either through the Streamlit software interface or from a Gmail monitoring path on Raspberry Pi. Sender, subject, and body fields are normalized and converted into the representation expected by the selected classifier. The workstation path uses DistilBERT for three-class classification, whereas the edge path uses a lightweight TF-IDF and Naive Bayes model. Both routes terminate in a browser interface that exposes the prediction and associated evidence.

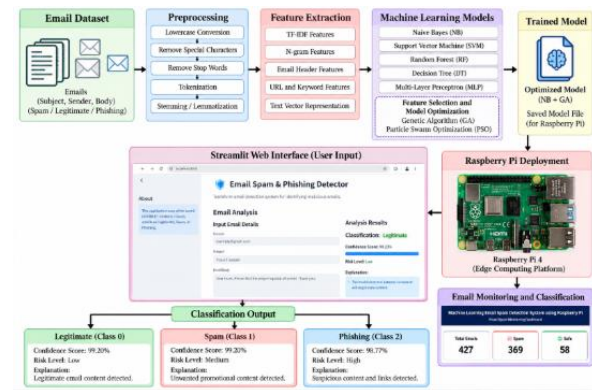


Fig. 1. System Architecture.

B) Dataset and Preprocessing

The implemented emails.csv structure contains text information and a categorical label. The final software mapping is Legitimate = 0, Spam = 1, and Phishing = 2, and the training path uses an 80/20 train-validation split with random_state = 42. Sender, subject, and body are preserved as separate semantic fields before being combined into structured text for transformer input. The classical route performs lowercase conversion, URL and punctuation handling, whitespace normalization, and TF-IDF feature construction. The contextual route uses the DistilBERT tokenizer with padding and truncation and stores the tokenizer together with the trained checkpoint for consistent inference.

Path	Representation	Model	Output
Software	Sender + subject + body tokens	DistilBERT	Legitimate / Spam / Phishing
Edge	TF-IDF sparse vector	Naive Bayes	Spam / Safe monitoring

Model study	Text features	SVM, RF, DT, MLP	Comparative classification
Optimization	Selected feature subset	GA / PSO	Reduced feature space

Table I. Functional model paths used in the implementation.

C) Training, Inference, and Edge Deployment

The software classifier is instantiated as a three-label DistilBERT sequence model. Structured email text is tokenized and converted to input identifiers and attention masks; the classifier returns three logits, softmax produces class probabilities, and the maximum probability determines the displayed class and confidence. The interface then maps Legitimate, Spam, and Phishing to Low, Medium, and High risk levels. The selected model and tokenizer are persisted so that inference uses the same representation as training.

On Raspberry Pi, TF-IDF and Naive Bayes provide a lower-complexity alternative suitable for repeated CPU inference. The device operates as both the local inference computer and browser-facing service. Gmail application authentication, local model loading, message retrieval, classification, dashboard refresh, and result storage are integrated in the edge workflow. Model loading is performed once rather than for every message, reducing repeated initialization overhead.

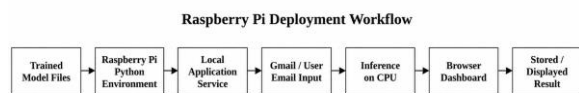


Fig. 2. Raspberry Pi edge-deployment workflow used for local inference and browser presentation.

D) Evaluation and Reproducibility

The training and evaluation path uses a deterministic 80/20 partition with `random_state = 42` so that the same validation subset can be reconstructed across repeated runs. The software project stores the trained sequence classifier, tokenizer configuration, label mapping, confusion matrix, and training-history artifact. Retaining these items is essential because a saved model is only reproducible when the corresponding tokenizer and output-label semantics are preserved. The prediction module caches model resources after the first request, switches the network to evaluation mode, and disables gradient calculation during inference to reduce avoidable memory and computation.

Evaluation is organized at three levels. First, labelled validation samples support class-level error inspection through the confusion matrix. Second, interface validation verifies that sender, subject, and body fields reach the model and that class, confidence, risk, and explanation fields are rendered correctly. Third, edge validation verifies mailbox access, local model loading, network serving, dashboard refresh, and message logging on Raspberry Pi. This separation prevents operational counts or individual confidence scores from being mistaken for global predictive measures.

E) Security and Resource Constraints

Resource constraints influence the model chosen for the deployment environment. DistilBERT provides richer contextual representations but has a larger model footprint and initialization cost than a sparse classical classifier. The Raspberry Pi path therefore prioritizes TF-IDF and Naive Bayes for repeated CPU inference. Relevant engineering measures for a controlled edge benchmark include startup time, steady-state latency, peak memory, model size, processor utilization, and device temperature. These measurements are not substituted with estimated values; they remain explicit targets for future controlled evaluation.

Local email processing also requires operational safeguards. Application-specific Gmail authentication should be separated from the user interface, credentials should not be embedded in source code, and prediction logs should remain traceable to the corresponding message record. A production system would additionally require hashed credentials, secure session management, controlled access to stored email content, audit logging, and versioned model artifacts. These controls are distinct from classifier accuracy but are necessary for responsible deployment of an email-security service.

F) Implementation Environment and Module Interaction

The implementation environment is based on Python with pandas and NumPy for data handling, scikit-learn for conventional machine-learning components, PyTorch and Hugging Face Transformers for contextual classification, and Streamlit for the browser-based software interface. The project additionally uses standard plotting and evaluation libraries to save training history and confusion-matrix artifacts. The Raspberry Pi deployment reuses the Python execution environment but substitutes the compact TF-IDF and Naive Bayes path where repeated inference and local service responsiveness are more important than transformer capacity.

Module separation supports maintainability and traceability. Data loading defines the field structure and class mapping; cleaning and tokenization transform input into model-ready features; the prediction module loads persisted resources and returns a class with confidence; the interface validates required fields and renders the response; and the monitoring component updates the edge dashboard. Because these modules communicate through stable input-output contracts, the underlying classifier can be replaced without redesigning the complete browser workflow.

Module	Technology / Primary role
Data	pandas / NumPy - CSV loading and field handling
Classical ML	scikit-learn - TF-IDF and classifier execution
Contextual NLP	PyTorch / Transformers - DistilBERT training and inference
Software UI	Streamlit - Interactive email analysis

Edge platform	Raspberry Pi / Python - Local monitoring and inference
Mailbox path	Gmail application authentication - Message acquisition for monitoring

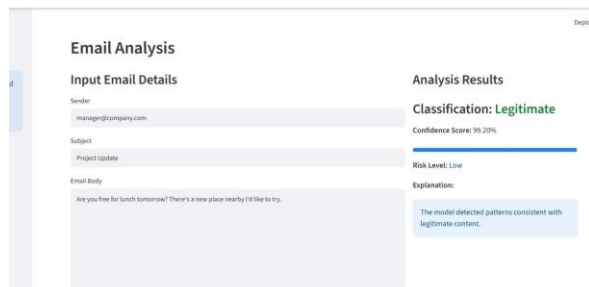
Table III. Principal implementation modules and their roles.

IV. EXPERIMENTAL RESULTS

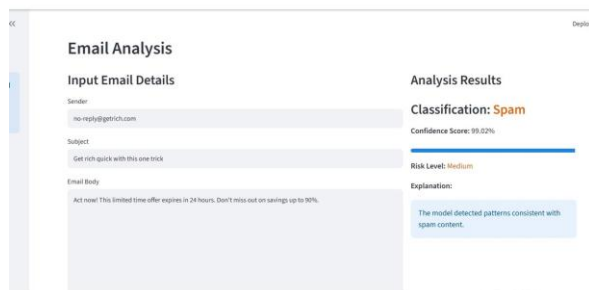
A) Software Output Results

The software interface was verified with representative Legitimate, Spam, and Phishing messages. For the recorded cases, the Legitimate sample is reported with 99.20% confidence and Low risk, the Spam sample with 99.02% confidence and Medium risk, and the Phishing sample with 98.77% confidence and High risk. These values are per-message softmax confidence scores; they describe the model output for the selected inputs and are not treated as dataset-level accuracy.

Legitimate - 99.20% / Low



Spam - 99.02% / Medium



Phishing - 98.77% / High

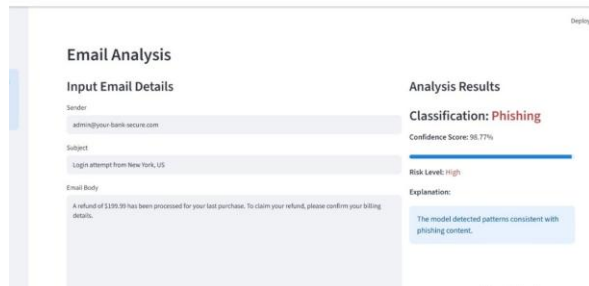


Fig. 3. Representative software results for Legitimate, Spam, and Phishing emails.

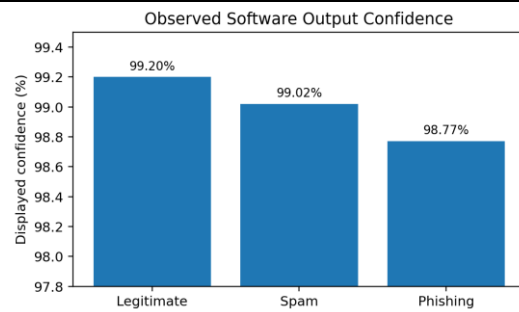


Fig. 4. Displayed confidence values for the three recorded software cases.

Class	Confidence	Risk	Interpretation
Legitimate	99.20%	Low	Normal correspondence
Spam	99.02%	Medium	Promotional / unwanted content
Phishing	98.77%	High	Credential-oriented deceptive content

Table II. Observed software output values.

B) Validation Evidence

The saved validation confusion matrix contains 100 evaluated messages: 28 Legitimate, 48 Spam, and 24 Phishing examples. In the recorded run, all observations lie on the main diagonal and no off-diagonal errors are shown. This confirms correct execution on the compact validation split. Because the project dataset contains a finite set of class-specific senders, subjects, and message templates, the result should be interpreted as project-level validation evidence rather than unrestricted real-world generalization.

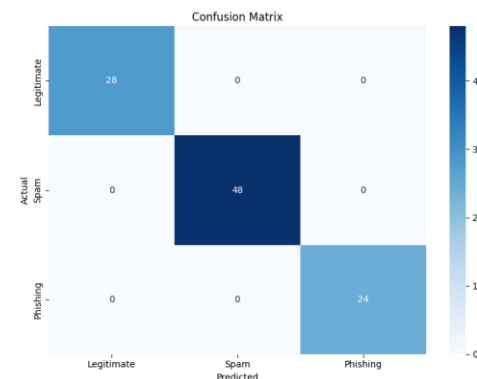


Fig. 5. Saved three-class validation confusion matrix.

C) Raspberry Pi Hardware Results

The hardware deployment verifies that email monitoring and local inference execute on the Raspberry Pi environment. The browser dashboard presents processed-message counts together with recent decisions and spam/ham probabilities. In the recorded populated dashboard state, 427 emails were processed, including 369 messages marked as spam and 58 marked as safe. These values are operational counts from the monitored inbox and are not benchmark accuracy measurements.

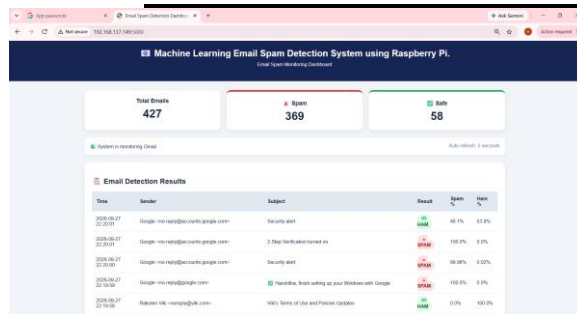


Fig. 6. Raspberry Pi Gmail monitoring dashboard

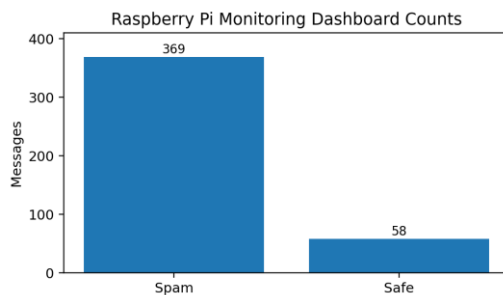


Fig. 7. Recorded spam and safe message counts on the Raspberry Pi dashboard.

D) Training Behaviour and Discussion

The stored training-history artifact shows rapid reduction of training loss and a stable recorded validation-accuracy trace for the compact project split. Together with the confusion matrix, this provides evidence that the fine-tuning and evaluation stages are integrated in the training pipeline. However, the software and hardware paths should not be treated as identical benchmark experiments: the workstation path performs three-class contextual classification, whereas the Raspberry Pi dashboard emphasizes continuous spam/safe monitoring with a compact local model.

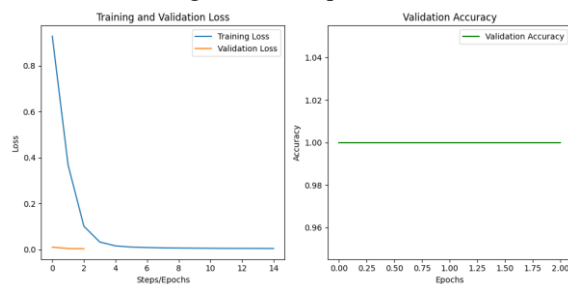


Fig. 8. Training and validation history stored by the classification pipeline.

The results therefore support three conclusions. First, the interface correctly propagates sender, subject, and body information through preprocessing and inference to a class-specific output. Second, the saved validation artifacts confirm reproducible execution on the configured dataset. Third, Raspberry Pi deployment demonstrates that local email monitoring can be served from a compact edge platform. Stronger claims about real-world phishing generalization require a larger independently collected

corpus, fixed test partitions, class-wise metrics, and measured edge latency and memory consumption.

E) Cross-Path Interpretation

The software and hardware paths address the same security objective but operate under different computational and label constraints. The Streamlit path explicitly separates Legitimate, Spam, and Phishing and is suitable for studying contextual relationships across sender, subject, and body. The Raspberry Pi dashboard emphasizes continuous local spam/safe monitoring and low-cost execution. Consequently, the two paths should be compared through engineering characteristics and controlled common test sets rather than by assigning an unsupported accuracy ranking between them.

The observed software confidence values are tightly clustered near 99%, but confidence is not equivalent to correctness. Softmax confidence expresses relative preference among the configured classes for a specific input and can be poorly calibrated outside the training distribution. Similarly, the dashboard counts of 369 spam and 58 safe messages describe the state of one monitoring run and do not reveal the number of false positives or false negatives. The confusion matrix provides stronger labelled-validation evidence because actual and predicted classes are compared directly.

F) Limitations and External Validation

The validation split is compact and is derived from a finite project dataset. The diagonal confusion matrix therefore demonstrates successful execution on the configured samples but does not establish resilience to unrestricted commercial email, multilingual phishing, adversarial obfuscation, unseen domains, or evolving attack templates. A publication-grade benchmark should freeze an independent test corpus, report class-wise precision, recall, and F1-score, analyze calibration, and repeat the experiment across multiple seeds or cross-validation folds.

Future hardware evaluation should submit a fixed request list after a defined warm-up interval, measure per-message latency over repeated runs, and record memory consumption and thermal behaviour. Such a protocol would quantify the trade-off between contextual transformer inference and compact sparse classification on the same hardware. The present system already separates preprocessing, model loading, prediction, and presentation, so these measurements can be introduced without redesigning the application.

The structured email representation also supports controlled field-sensitivity analysis. Because sender, subject, and body are concatenated into a single contextual sequence, one field can be changed while the others remain fixed. Such tests can reveal whether the classifier is relying excessively on sender domains, urgent subject terms, or body-level lexical patterns. This is particularly relevant for phishing because an apparently neutral message body may be paired with a suspicious sender or a high-pressure subject line. Recording

the full probability vector for these controlled cases provides more insight than retaining only the winning class.

Risk-level presentation should likewise remain traceable to the underlying class decision. In the software interface, Legitimate, Spam, and Phishing are mapped to Low, Medium, and High risk respectively. This mapping improves user interpretation but does not alter the model output. For evaluation, the semantic class should therefore remain the primary target, and the derived risk label should be treated as a presentation layer. Separating these two concepts prevents changes to interface wording from being mistaken for changes in model performance.

For the Raspberry Pi path, continuous operation introduces additional integrity checks that are not visible in a one-time desktop prediction. The vectorizer and classifier should be loaded once, preprocessing should remain identical for every retrieved message, and result logging should preserve the association between the processed email and the displayed decision. Monitoring counts are useful for operational status, but a controlled study should additionally store ground-truth labels when available so that false-positive and false-negative behaviour can be quantified over time.

G) Functional Verification and Traceability

Functional verification examines the complete application path rather than the classifier in isolation. A valid software test begins with sender, subject, and body input, confirms that all required fields are present, invokes the prediction module, maps the numerical output back to the semantic label, and renders confidence, risk level, and explanation. The three recorded software cases verify each target class through the same interaction path. This consistency is important because it shows that the user interface is not hard-coded for one response category and that the presentation layer remains independent of the predicted label.

The Raspberry Pi workflow adds a second set of operational checks. Before monitoring begins, the Python environment must load the vectorizer and classifier, application authentication must be accepted, the local service must bind to the intended network address, and the dashboard must be reachable from the browser. During monitoring, each processed email should remain traceable to its sender, subject, decision, and probability display. Aggregate message counts are useful for operational observation, but traceable row-level results are necessary for later error analysis.

The saved confusion matrix and training-history plot provide an additional link between model development and interface execution. The matrix records the distribution of actual and predicted classes for the compact validation subset, while training history records optimization and validation behaviour. Storing these artifacts together with the model and tokenizer makes later reproduction possible and provides a basis for comparing future model revisions under the same evaluation procedure.

H) Error Analysis and Practical Boundaries

Meaningful email-security error analysis should distinguish several failure modes. A false Legitimate decision on a phishing message is a high-impact false negative; a Spam-to-Phishing confusion changes the displayed risk level from Medium to High; and a Legitimate-to-Spam error can unnecessarily interrupt normal correspondence. Other difficult cases include obfuscated URLs, mixed-language text, contrastive wording, unfamiliar sender domains, and messages whose subject is suspicious while the body appears neutral. Controlled test sets should therefore preserve sender, subject, body, true class, predicted class, and class probabilities for every record.

The present evidence is strongest for implementation correctness on the configured dataset and hardware environment. External validity requires independent corpora with different senders, writing styles, organizations, and attack templates. A future study should reserve a test set before tuning, report per-class confusion matrices and macro-averaged metrics, evaluate repeated seeds, and compare resource consumption of contextual and sparse models on the same Raspberry Pi. These additions would convert the current edge-AI implementation into a more rigorous cross-platform benchmarking study without changing the fundamental system architecture.

V. CONCLUSION AND FUTURE SCOPE

This work integrates machine-learning-based email filtering with contextual NLP and Raspberry Pi edge deployment. The conventional branch considers Naive Bayes, SVM, Random Forest, Decision Tree, and MLP classifiers together with GA and PSO optimization. The software branch uses DistilBERT to distinguish Legitimate, Spam, and Phishing emails and exposes class confidence and risk information through a Streamlit interface. The Raspberry Pi branch uses TF-IDF and Naive Bayes to support local Gmail monitoring and browser-based visualization. Recorded software outputs show high sample-level confidence for the three representative classes, while the saved validation confusion matrix documents correct classification of the 100-message project split. The hardware dashboard further confirms local processing of 427 messages in the recorded run.

Future work should evaluate all model paths on the same independently preserved test corpus, report class-wise precision, recall, F1-score and confidence intervals, and measure Raspberry Pi startup time, steady-state latency, peak memory, and temperature. Additional work may examine quantized contextual encoders, incremental model updates, adversarially obfuscated phishing content, and calibrated confidence scores. Security-oriented deployment should also strengthen credential management, access control, logging, and model-version traceability.

REFERENCES

- [1] W. Feng, J. Sun, L. Zhang, C. Cao, and Q. Yang, "A support vector machine based Naive Bayes algorithm for spam filtering," in Proc. IEEE35th Int. Perform. Comput. Commun.

- Conf. (IPCCC), Dec. 2016, pp. 1-8, doi: 10.1109/ipccc.2016.7820655.
- [2] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: Review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, Jun. 2019, Art. no. e01802, doi: 10.1016/j.heliyon.2019.e01802.
- [3] W. Awad and S. ELseuo, "Machine learning methods for spam E-Mail classification," *Int. J. Comput. Sci. Inf. Technol.*, vol. 3, no. 1, pp. 173-184, Feb. 2011, doi: 10.5121/ijcsit.2011.3112.
- [4] S. Mohammed, O. Mohammed, and J. Fiaidhi, "Classifying unsolicited bulk email (UBE) using Python machine learning techniques," *Int. J. Hybrid Inf. Technol.*, vol. 6, no. 1, pp. 43-55, 2013. [Online]. Available: https://www.researchgate.net/publication/236970412_Classifying_Unsolicited_Bulk_Email_UBE_using_Python_Machine_Learning_Techniques
- [5] A. Wijaya and A. Bisri, "Hybrid decision tree and logistic regression classifier for email spam detection," in *Proc. 8th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE)*, Oct. 2016, pp. 1-4, doi: 10.1109/ICITEE.2016.7863267.
- [6] K. Agarwal and T. Kumar, "Email spam detection using integrated approach of Naive Bayes and particle swarm optimization," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2018, pp. 685-690, doi: 10.1109/ICCONS.2018.8662957.
- [7] R. Belkebir and A. Guessoum, "A hybrid BSO-Chi2-SVM approach to arabic text categorization," in *Proc. ACS Int. Conf. Comput. Syst. Appl. (AICCSA)*, Ifran, Morocco, May 2013, pp. 1-7, doi: 10.1109/AICCSA.2013.6616437.
- [8] A. I. Taloba and S. S. I. Ismail, "An intelligent hybrid technique of decision tree and genetic algorithm for E-Mail spam detection," in *Proc. 9th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Cairo, Egypt, Dec. 2019, pp. 99-104, doi: 10.1109/ICICIS46948.2019.9014756.
- [9] R. Karthika and P. Visalakshi, "A hybrid ACO based feature selection method for email spam classification," *WSEAS Trans. Comput.*, vol. 14, pp. 171-177, 2015. [Online]. Available: <https://www.wseas.org/multimedia/journals/computers/2015/a365705-553.pdf>
- [10] S. L. Marie-Sainte and N. Alalyani, "Firey algorithm based feature selection for arabic text classification," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 3, pp. 320-328, Mar. 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S131915781830106X>
- [11] E. A. Natarajan, S. Subramanian, and K. Premalatha, "An enhanced cuckoo search for optimization of bloom filter in spam filtering," *Global J. Comput. Sci. Technol.*, vol. 12, no. 1, pp. 75-81, 2012.
- [12] A. Géron, *Hands-On Machine Learning With Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Newton, MA, USA: O'Reilly Media, 2019, Ch. 1.
- [13] (2019). 1. Supervised Learning, Scikit-Learn 0.22.2 Documentation. Accessed: Oct. 9, 2019. [Online]. Available: https://scikit-learn.org/stable/supervised_learning.html
- [14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825-2830, Oct. 2011. [Online]. Available: <http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>
- [15] S. Zhu and F. Chollet. (2019). Working With RNNs. Accessed: Nov. 2, 2019. [Online]. Available: https://keras.io/guides/working_with_rnns/
- [16] (2019). TensorFlow Core | Machine Learning for Beginners and Experts. Accessed: Nov. 2, 2019. [Online]. Available: <https://www.tensorflow.org/overview>
- [17] (2019). Spyder: The Scientific Python Development Environment Documentation Spyder 3 Documentation. Accessed: Nov. 2, 2019. [Online]. Available: <https://docs.spyder-ide.org/>
- [18] (2019). User Guide Anaconda Documentation. Accessed: Nov. 9, 2019. [Online]. Available: <https://docs.anaconda.com/ae-notebooks/user-guide/>
- [19] (2020). Google Colaboratory. Accessed: Mar. 18, 2020. [Online]. Available: <https://colab.research.google.com>
- [20] S. Sawla. (2018). Introduction to Naive Bayes for Classification. Accessed: Oct. 9, 2019. [Online]. Available: <https://medium.com/@srishtisawla/introduction-to-naive-bayes-for-classification-baefefb43a2d>
- [21] GeeksforGeeks. (2019). Naive Bayes Classifiers-GeeksforGeeks. Accessed: Nov. 10, 2019. [Online]. Available: <https://www.geeksforgeeks.org/naive-Bayes-classifiers/>
- [22] G. Bonaccorso, *Machine Learning Algorithms*, 2nd ed. Birmingham, U.K.: Packt Publishing, 2018.
- [23] G. Singh, B. Kumar, L. Gaur, and A. Tyagi, "Comparison between multinomial and Bernoulli Naive Bayes for text classification," in *Proc. Int. Conf. Autom., Comput. Technol. Manage. (ICACTM)*, London, U.K., Apr. 2019, pp. 593-596, doi: 10.1109/ICACTM.2019.8776800.
- [24] (2018). Implementing 3 Naive Bayes classifiers in Scikit-Learn | Packt Hub. Accessed: Nov. 13, 2019. [Online]. Available: <https://hub.packtpub.com/implementing-3-naive-Bayes-classifiers-in-scikit-learn>
- [25] (2019). Sklearn.Linear_Model.SGDClassifier Scikit-Learn 0.22.2 Documentation. Accessed: Nov. 29, 2019. [Online]. Available: https://scikitlearn.org/stable/modules/neural_networks_supervised.html#neural-networks-supervised
- [26] A. Géron, *Hands-on Machine Learning With Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Newton, MA, USA: O'Reilly Media, 2019, Ch. 6.
- [27] T. Yiu. (2019). Understanding Random Forest. Medium. Accessed: Jan. 17, 2020. [Online]. Available: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
- [28] (2020). 1.17. Neural Network Models (Supervised) Scikit-Learn 0.22.2 Documentation. Accessed: Mar. 17, 2020. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#neural-networks-supervised
- [29] I. Androustopoulos. (2000). Ling-Spam. AUEB. Accessed: Oct. 11, 2019. [Online]. Available: http://www.aueb.gr/users/ion/data/lingspam_public.tar.gz
- [30] I. Androustopoulos, V. Metsis, and G. Paliouras. (2006). The Enron-Spam Datasets. Accessed: Oct. 11, 2019. [Online]. Available: <http://www2.aueb.gr/users/ion/data/enron-spam/>