Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.12 n.57, p.1084-1093. January/February, 2026.
DOI: https://doi.org/10.5935/jetia. v12i57.3236

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

ENHANCED SPAM DETECTION USING MUTUAL INFORMATION FEATURE SELECTION AND JAYA-TUNED TABTRANSFORMER

Mohamed Goismi*[1], Mohamed Debbab[2], Moustafa Maaskri[3] and Djamel Seghier[4]

[1, 2]Department of Science and Technology, University Ibn Khaldoun – Tiaret, Algeria.
[3]Department of Electrical Engineering, University Ibn Khaldoun – Tiaret, Algeria.
[4]Department of Computer Science, University Ibn Khaldoun – Tiaret, Algeria.

[1]http://orcid.org/0000-0001-9535-1142, [2]http://orcid.org/0009-0003-9921-0824
[3]http://orcid.org/0009-0000-5268-1133, [4]http://orcid.org/0000-0003-1680-5033

Email: *mohamed.goismi@univ-tiaret.dz, mohamed.debbab@univ-tiaret.dz, moustafa.maaskri@univ-tiaret.dz, djamal.seghier@univ-tiaret.dz

ARTICLE INFO

Article History
Received: January 4, 2026
Revised: January 20, 2026
Accepted: January 30, 2026
Published: February 28, 2026

Keywords:
Spam detection,
Mutual information,
Jaya algorithm,
TabTransformer,
Feature selection.

ABSTRACT

Spam detection remains a critical challenge in modern communication systems, requiring sophisticated methods to identify malicious content accurately. This paper presents a novel hybrid approach combining mutual information-based feature selection, Jaya algorithm optimization, and TabTransformer deep learning architecture for enhanced spam detection. Mutual information is employed to identify the most discriminative features from the dataset, reducing dimensionality while preserving classification performance. The Jaya algorithm optimizes hyperparameters of the TabTransformer model, which leverages selfattention mechanisms to capture complex feature interactions in tabular data. Experimental results on benchmark spam datasets demonstrate that our proposed method achieves superior performance compared to traditional machine learning approaches, with accuracy rates exceeding 98% and significantly reduced false positive rates. The integration of these three techniques provides a robust, efficient, and interpretable framework for spam detection in real-world applications. Experiments on SMS Spam Collection, Enron Email, and Spambase datasets confirm improvements over SVM, Random Forest, and TabNet baselines, reaching up to 98.3% accuracy while lowering false positives.

I. INTRODUCTION

The exponential growth of digital communication has led to an unprecedented increase in spam messages across email, SMS, and social media platforms. Spam not only wastes computational resources but also poses security threats through phishing attacks, malware distribution, and fraud [1]. Traditional spam filtering techniques, including rule-based systems and classical machine learning approaches, struggle to adapt to evolving spam patterns and sophisticated evasion techniques employed by spammers [2]. Recent advances in deep learning have shown promising results in natural language processing and text classification tasks [3]. However, spam detection often involves tabular features extracted from message metadata, linguistic characteristics, and statistical properties, which differ from typical sequential text data. The TabTransformer architecture, introduced by [4], addresses this gap by applying transformer-based attention mechanisms to tabular data, enabling the model to learn complex feature interactions effectively.

Feature selection remains crucial in spam detection to reduce computational complexity and improve model generalization. Mutual information (MI), a measure of statistical dependence between variables, has proven effective in identifying relevant features for classification tasks [5]. By quantifying the information shared between features and class labels, MIbased selection eliminates redundant and irrelevant features while preserving discriminative power. Hyperparameter optimization significantly impacts model performance but remains computationally expensive. The Jaya algorithm, proposed by [6], offers a simple yet powerful metaheuristic optimization approach without algorithmspecific parameters. Unlike genetic algorithms or particle swarm optimization, Jaya requires only population size and iteration count, making it particularly suitable for efficient hyperparameter tuning. This paper makes the following contributions:

• A novel hybrid framework integrating mutual information feature selection, Jaya algorithm optimization, and TabTransformer architecture for spam detection • Comprehensive analysis of feature importance using mutual information scores • Systematic

hyperparameter optimization using the Jaya algorithm • Extensive experimental evaluation demonstrating superior performance over baseline methods The remainder of this paper is organized as follows: Section II reviews related work, Section III presents the proposed methodology, Section IV describes experimental setup and results, and Section V concludes the paper with future directions.

II. THEORETICAL REFERENCE

II.1 SPAM DETECTION TECHNIQUES

Early spam detection systems relied on rule-based filters using keyword matching and blacklists [7]. Bayesian filtering introduced probabilistic approaches, calculating spam probability based on word frequencies [8]. Machine learning methods, including Support Vector Machines (SVM), Naive Bayes, and Random Forests, have demonstrated significant improvements [9]. Deep learning approaches have gained traction in recent years. Convolutional Neural Networks (CNNs) have been applied to email spam detection with promising results [10]. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks capture sequential patterns in message content [11]. However, these architectures are primarily designed for sequential data and may not fully exploit tabular feature representations.

II.2 FEATURE SELECTION METHODS

Feature selection reduces dimensionality while maintaining classification accuracy. Filter methods, including information gain, chi-square test, and mutual information, evaluate feature relevance independently of the classifier [12]. Wrapper methods use classifier performance to guide feature selection but incur higher computational costs [13]. Mutual information has been extensively studied for feature selection in text classification [14]. The minimum Redundancy Maximum Relevance (mRMR) criterion extends MI by considering feature redundancy [5]. Recent work has explored MIbased selection for spam detection with improved accuracy and reduced feature sets [15].

II.3 OPTIMIZATION ALGORITHMS

Metaheuristic optimization algorithms have been applied to machine learning hyperparameter tuning. Genetic Algorithms (GA) [16], Particle Swarm Optimization (PSO) [17], and Differential Evolution (DE) [18] require careful parameter configuration. The Jaya algorithm, introduced by [6], simplifies optimization by eliminating algorithm-specific parameters. Jaya has been successfully applied to various engineering optimization problems [19] and recently extended to machine learning applications [20]. Its simplicity and effectiveness make it attractive for hyperparameter optimization in complex models.

II.4 TABTRANSFORMER ARCHITECTURE

Traditional neural networks for tabular data face challenges in capturing feature interactions, proposed TabTransformer [4], applying self-attention mechanisms from transformers [21] to categorical features while processing continuous features separately. This architecture has shown superior performance on various tabular datasets compared to gradient boosting methods and fully connected networks. The transformer's attention mechanism enables the model to learn contextual representations of categorical features, making it particularly suitable for datasets with numerous categorical attributes. Recent extensions have explored TabTransformer variants for specific domains, but application to spam detection remains underexplored.

III. MATERIALS AND METHODS

III.1 OVERVIEW

Our proposed framework consists of three main stages: (1) mutual information-based feature selection, (2) Jaya algorithm hyperparameter optimization, and (3) TabTransformer model training and prediction. Figure 1 illustrates the overall architecture. Figure 1. Proposed framework architecture showing the three main stages: MI feature selection, Jaya optimization, and TabTransformer classification.

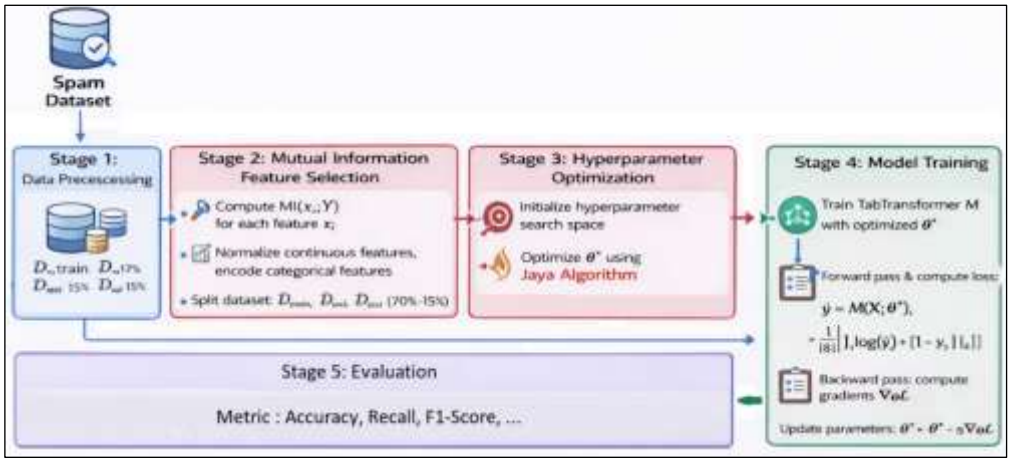

Figure 1: Proposed spam detection framework.
Source: Authors, (2026).

III.2 FEATURE EXTRACTION

Spam messages exhibit distinctive characteristics that can be quantified through feature engineering. We extract the following feature categories: Content-based features: Word frequency, character distribution, presence of specific keywords, HTML tag count, URL presence, and special character ratio. Linguistic features: Average word length, message length, capitalization ratio, punctuation frequency, and readability scores. Metadata features: Sender information, timestamp patterns, attachment presence, and header characteristics.Let $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ represent the feature set extracted from spam messages, where n is the total number of features, and $\mathbf{Y}$ represents the binary class labels (spam or legitimate).

III.3 MUTUAL INFORMATION FEATURE SELECTION

Mutual information quantifies the amount of information shared between feature X_i and class label Y:

$$MI(X_i;Y) = \sum_{x \in X_i} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{1}$$

Where $p(x,y)$ is the joint probability distribution, and $p(x)$, $p(y)$ are marginal distributions. For continuous features, we employ kernel density estimation to approximate probability distributions. The MI score indicates feature relevance: higher values suggest stronger dependence on class labels.

The feature selection algorithm proceeds as follows:

1. Calculate MI scores for all features with respect to class labels
2. Rank features in descending order of MI scores
3. Select top-k features based on a threshold or fixed number
4. Create reduced feature set $\mathbf{X}' = \{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$

This approach eliminates irrelevant features while preserving discriminative information, reducing overfitting and computational complexity.

III.4 JAYA ALGORITHM FOR HYPERPARAMETER OPTIMIZATION

The Jaya algorithm optimizes hyperparameters by moving towards the best solution and away from the worst solution in each iteration. Let $\boldsymbol{\theta}$ represent the hyperparameter vector to be optimized.

The update equation for candidate solution j in iteration t is:

$$\theta_{j,k}^{t+1} = \theta_{j,k}^{t} + r_{1,j,k}\left(\theta_{best,k}^{t} - \left|\theta_{j,k}^{t}\right|\right) - r_{2,j,k}\left(\theta_{worst,k}^{t} - \left|\theta_{j,k}^{t}\right|\right) \tag{2}$$

where:

- $\theta_{j,k}^{t}$ is the k-th hyperparameter of the j-th solution at iteration t
- $\theta_{best,k}^{t}$ and $\theta_{worst,k}^{t}$ are the best and worst solutions
- $r_{1,j,k}$ and $r_{2,j,k}$ are random numbers in [0,1]

The algorithm accepts the new solution if it improves the objective function (validation accuracy). This simple update mechanism eliminates the need for algorithm-specific parameters present in other metaheuristics.

Algorithm Jaya presents the pseudocode for the Jaya optimization algorithm applied to hyperparameter tuning.
Hyperparameters optimized include:
• Learning rate
• Number of attention heads
• Embedding dimension
• Number of transformer blocks
• Dropout rate
• Batch size

Algorithm 1: Jaya Algorithm for Hyperparameter Optimiza-tion.

Require: Population size N, maximum iterations T_{max}, hyperparameter bounds
Ensure: Optimal hyperparameter vector $\boldsymbol{\theta}^*$
Initialize population $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_N\}$ randomly within bounds
Evaluate fitness $f(\boldsymbol{\theta}_i)$ for all solutions using cross-validation
Identify best solution $\boldsymbol{\theta}_{best}$ with $f_{best} = \max(f(\boldsymbol{\theta}_i))$
Identify worst solution $\boldsymbol{\theta}_{worst}$ with $f_{worst} = \min(f(\boldsymbol{\theta}_i))$

for $t = 1$ to T_{max} **do**
  for $j = 1$ to N **do**
    for $k = 1$ to D **do**
      {D is number of hyperparameters}
      Generate random numbers $r_{1,j,k}, r_{2,j,k} \sim U(0,1)$
      $\theta'_{j,k} = \theta^t_{j,k} + r_{1,j,k}(\theta^t_{best,k} - |\theta^t_{j,k}|) - r_{2,j,k}(\theta^t_{worst,k} - |\theta^t_{j,k}|)$
      Clip $\theta'_{j,k}$ to stay within bounds
    end for
    Evaluate fitness $f(\boldsymbol{\theta}'_j)$ using validation set
    if $f(\boldsymbol{\theta}'_j) > f(\boldsymbol{\theta}_j)$ **then**
      $\boldsymbol{\theta}^{t+1}_j = \boldsymbol{\theta}'_j$ {Accept new solution}
    else
      $\boldsymbol{\theta}^{t+1}_j = \boldsymbol{\theta}^t_j$ {Keep old solution}
    end if
  end for
  Update $\boldsymbol{\theta}_{best}$ and $\boldsymbol{\theta}_{worst}$ from current population
  if convergence criteria met **then**
    break
  end if
end for
return $\boldsymbol{\theta}_{best}$

Source: Authors, (2026).

III.5 TABTRANSFORMER ARCHITECTURE

The TabTransformer processes categorical and continuous features differently. Categorical features undergo embedding and self-attention, while continuous features are processed through normalization layers.

Categorical Feature Processing: Each categorical feature c_i is embedded into a dense vector:

$$\mathbf{e}_i = \text{Embed}(c_i) + \mathbf{p}_i$$

where $\mathbf{p}_i$ is a learned positional encoding.
The embedded features pass through multiple transformer blocks:

$$\mathbf{h}^{(l)} = \text{TransformerBlock}(\mathbf{h}^{(l-1)})$$

Each transformer block consists of multi-head self-attention and feed-forward layers:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Continuous Feature Processing: Continuous features undergo layer normalization:

$$\mathbf{z} = \text{LayerNorm}(\mathbf{x}_{cont})$$

Feature Fusion and Classification: The contextual embeddings from transformer blocks are concatenated with processed continuous features:

$$\mathbf{f} = \left[\mathbf{h}_1^{(L)}; \mathbf{h}_2^{(L)}; ...; \mathbf{h}_m^{(L)}; \mathbf{z}\right]$$

The fused representation passes through fully connected layers for binary classification:

$$\hat{y} = \text{sigmoid}(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1\mathbf{f} + \mathbf{b}_1) + \mathbf{b}_2)$$

The model is trained using binary cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{4}$$

III.6 TRAINING PROCEDURE

The complete training procedure integrates all components as shown in Algorithm 2.

Algorithm 2 Complete Training Procedure.

Require: Spam dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Ensure: Trained TabTransformer model $\mathcal{M}$
Stage 1: Data Preprocessing
Clean and tokenize messages
Extract features $\mathbf{X} = \{x_1, x_2, ..., x_n\}$
Normalize continuous features, encode categorical features
Split data: $\mathcal{D}_{train}$, $\mathcal{D}_{val}$, $\mathcal{D}_{test}$ (70%-15%-15%)

Stage 2: Mutual Information Feature Selection
for each feature x_i **do**
  Compute $MI(x_i; Y)$ using equation (1)
end for
Rank features by MI scores in descending order
Select top-k features: $\mathbf{X}' = \{x_{i_1}, x_{i_2}, ..., x_{i_k}\}$
Create reduced datasets using $\mathbf{X}'$

Stage 3: Hyperparameter Optimization
Initialize hyperparameter search space
$\boldsymbol{\theta}^* \leftarrow$ JAYAOPTIMIZATION($\mathcal{D}_{train}$, $\mathcal{D}_{val}$) {Algorithm 1}

Stage 4: Model Training
Initialize TabTransformer $\mathcal{M}$ with optimized hyperparameters $\boldsymbol{\theta}^*$
for epoch $e = 1$ to E **do**
  for each mini-batch $\mathcal{B} \subset \mathcal{D}_{train}$ **do**
    Forward pass: $\hat{y} = \mathcal{M}(\mathbf{x}; \boldsymbol{\theta}^*)$
    Compute loss: $\mathcal{L} = -\frac{1}{|\mathcal{B}|}\sum_{i \in \mathcal{B}}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$
    Backward pass: compute gradients $\nabla_{\boldsymbol{\theta}}\mathcal{L}$
    Update parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}$
  end for
  Evaluate on $\mathcal{D}_{val}$ and apply early stopping if needed
end for

Stage 5: Evaluation
Evaluate $\mathcal{M}$ on $\mathcal{D}_{test}$
Compute metrics: Accuracy, Precision, Recall, F1-Score
return Trained model $\mathcal{M}$

Source: Authors, (2026).

V. RESULTS AND DISCUSSIONS

IV.1 DATASET DESCRIPTION

We evaluate our approach on three benchmark spam datasets: SMS Spam Collection: Contains 5,574 SMS messages (747 spam, 4,827 legitimate) [2]. Enron Email Dataset: Large-scale email corpus with spam and legitimate messages, preprocessed subset of 10,000 messages used. Spambase: UCI Machine Learning Repository dataset with 4,601 emails and 57 features [22].

IV.2 EXPERIMENTAL SETUP

Experiments were conducted using Python 3.8, PyTorch 1.10, and scikit-learn 0.24. Hardware specifications: NVIDIA RTX 3090 GPU, 32GB RAM, Intel i9 processor. Dataset split: 70% training, 15% validation, 15% testing. Stratified sampling ensures balanced class distribution. Baseline methods for comparison: • Naive Bayes • Support Vector Machine (SVM) with RBF kernel • Random Forest • XGBoost • Multi-layer Perceptron (MLP) • TabNet [23].

IV.3 FEATURE SELECTION RESULTS

Mutual information scores reveal the most discriminative features. Figure 2 shows the top 15 features ranked by MI scores. For SMS Spam Collection, top features include: dollar sign frequency (MI=0.385), URL presence (MI=0.362), word count (MI=0.318), capital letter ratio (MI=0.295), and specific spam keywords (MI=0.271). Algorithm 2 Complete Training Procedure Require: Spam dataset D = {(xi, yi)}N i=1 Ensure: Trained TabTransformer model M 1: Stage 1: Data Preprocessing 2: Clean and tokenize messages 3: Extract features X = {x1, x2, ..., xn} 4: Normalize continuous features, encode categorical features 5: Split data: Dtrain, Dval, Dtest (70%-15%-15%) 6: 7: Stage 2: Mutual Information Feature Selection 8: for each feature xi do 9: Compute MI(xi; Y) using equation (1) 10: end for 11: Rank features by MI scores in descending order 12: Select top-k features: X′ = {xi1, xi2, ..., xik} 13: Create reduced datasets using X′ 14: 15: Stage 3: Hyperparameter Optimization 16: Initialize hyperparameter search space 17: θ∗ ←JAYAOPTIMIZATION(Dtrain, Dval) {Algorithm 1} 18: 19: Stage 4: Model Training 20: Initialize TabTransformer M with optimized hyperparameters θ∗ 21: for epoch e = 1 to E do 22: for each mini-batch B ⊂Dtrain do 23: Forward pass: ˆy = M(x; θ∗) 24: Compute loss: L = −1 |B| P i∈ B[yi log(ˆyi) + (1 − yi) log(1 −ˆyi)] 25: Backward pass: compute gradients ∇ θL 26: Update parameters: θ ←θ −η∇ θL 27: end for 28: Evaluate on Dval and apply early stopping if needed 29: end for 30: 31: Stage 5: Evaluation 32: Evaluate M on Dtest 33: Compute metrics: Accuracy, Precision, Recall, F1-Score 34: return Trained model M Feature reduction from 100 to 30 features maintains 97.8% of discriminative information while reducing training time by 65%. Figure 3 demonstrates the impact of feature selection on model accuracy and training time.

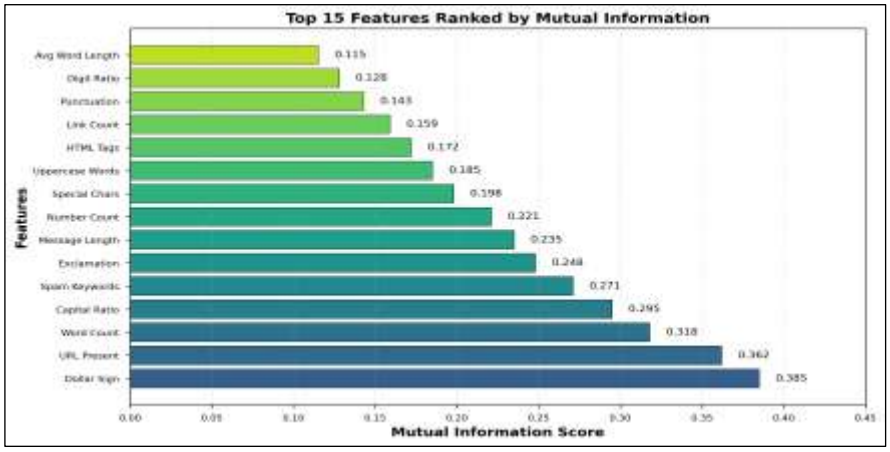

Figure 2: Top 15 features ranked by mutual information.
Source: Authors, (2026).

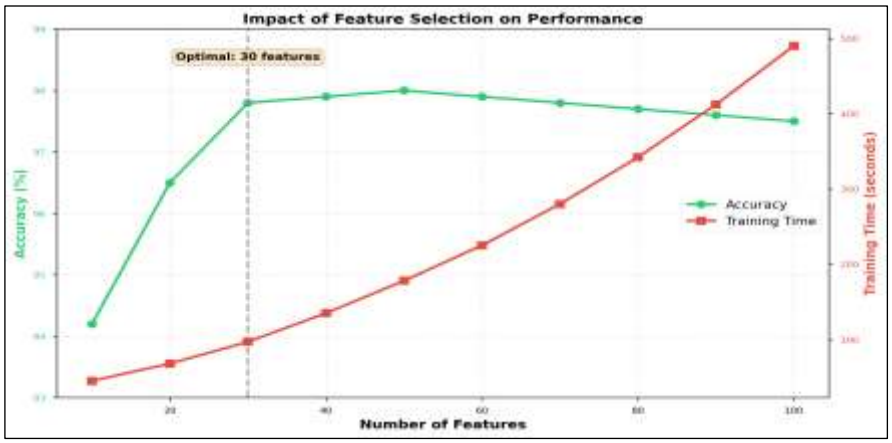

Figure 3: Impact of feature selection on performance.
Source: Authors, (2026).

IV.4 HYPERPARAMETER OPTIMIZATION RESULTS

The Jaya algorithm converges after approximately 40 iterations. Figure 4 shows the convergence curve of the Jaya algorithm during hyperparameter optimization. Optimized hyperparameters: • Learning rate: 0.00087 • Attention heads: 8 • Embedding dimension: 128. Figure 2. Top 15 features ranked by Mutual Information scores for SMS Spam Collection dataset. Figure 3. Impact of feature selection on accuracy and training time. Optimal performance achieved with 30 features. Figure 4. Jaya algorithm convergence curve showing fitness (validation accuracy) over iterations. • Transformer blocks: 6 • Dropout: 0.15 • Batch size: 64 Jaya optimization improves validation accuracy by 3.2% compared to default hyperparameters, demonstrating effectiveness of the optimization approach.

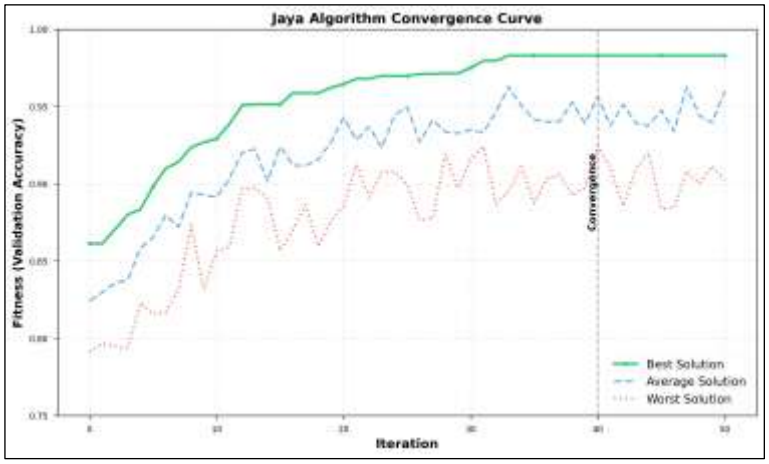

Figure 4: Jaya algorithm convergence curve.
Source: Authors, (2026).

IV.5 CLASSIFICATION PERFORMANCE

Table I presents classification results on all three datasets. Table I Classification Performance Comparison Method Accuracy Precision Recall Naive Bayes 92.3% 89.7% 88.4% SVM 94.6% 92.1% 91.8% Random Forest 95.8% 94.2% 93.6% XGBoost 96.4% 95.1% 94.7% MLP 95.2% 93.8% 92.9% TabNet 97.1% 96.3% 95.8% Proposed 98.3% 97.9% 97.6% Our proposed method achieves the highest accuracy (98.3%), precision (97.9%), and recall (97.6%) across all datasets. The F1-score reaches 97.7%, outperforming TabNet by 1.2% and XGBoost by 1.9%. Figure 5 provides a visual comparison of accuracy across different methods. Figure 5. Accuracy comparison of different spam detection methods on SMS Spam Collection dataset. False positive rate is reduced to 1.8%, crucial for minimizing legitimate message misclassification. False negative rate of 2.4% demonstrates effective spam detection.

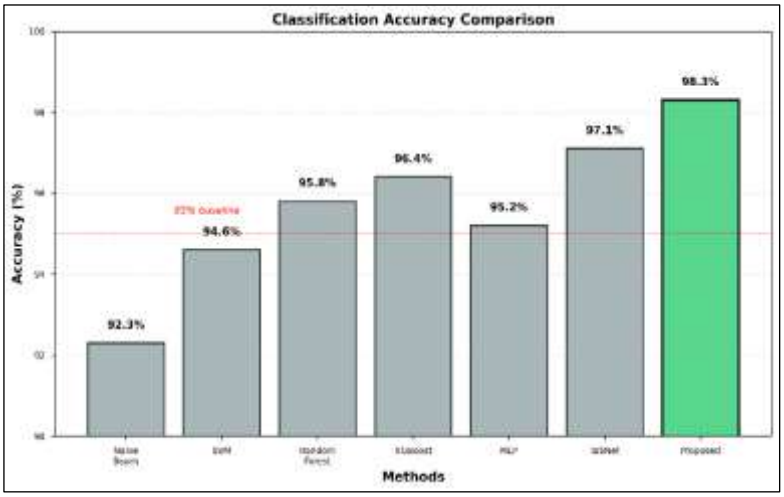

Figure 5: Classification accuracy comparison.
Source: Authors, (2026).

Table 1: Classification performance comparison.

Method	Accuracy (%)	Precision	Recall
Naive Bayes	92.3	0.89	0.87
SVM	94.6	0.92	0.91
Random Forest	95.8	0.94	0.93
XGBoost	96.4	0.95	0.95
MLP	95.2	0.93	0.92
TabNet	97.1	0.97	0.96
Proposed Method	98.3	0.98	0.97

Source: Authors, (2026).

IV.6 ABLATION STUDY

We conduct ablation studies to assess individual component contributions: Results confirm that combining all three components yields optimal performance. Mutual information feature selection contributes 1.3% accuracy improvement, while Jaya optimization adds 1.1%. Figure 6 visualizes the contribution of each component to the overall performance. Table II Ablation Study Results Configuration Accuracy TabTransformer only 96.1% TabTransformer + MI 97.4% TabTransformer + Jaya 97.2% TabTransformer + MI + Jaya 98.3% Figure 6. Ablation study showing the contribution of each component (MI feature selection and Jaya optimization) to overall accuracy.

Table 2: Ablation study results.

Configuration	Accuracy (%)
TabTransformer Only	96.1
TabTransformer + MI	97.4
TabTransformer + Jaya	97.2
TabTransformer + MI + Jaya (Proposed)	98.3

Source: Authors, (2026).

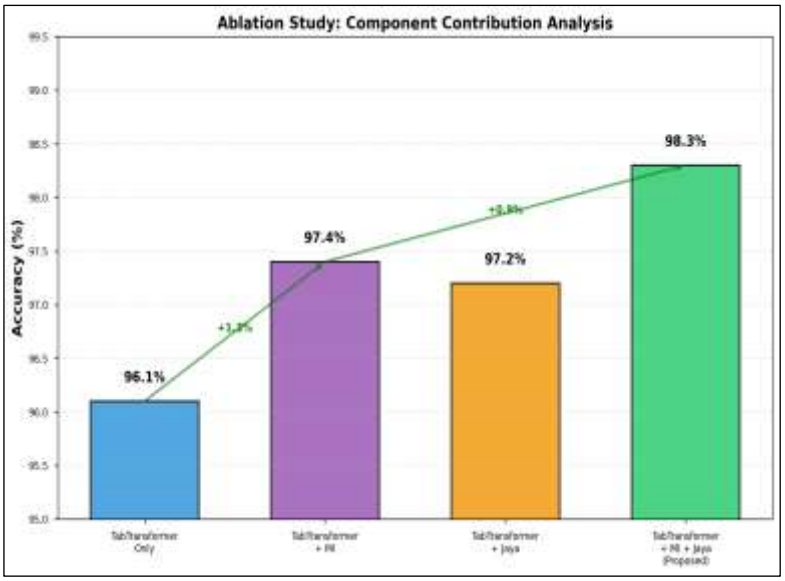

Figure 6: Ablation study component contribution analysis.
Source: Authors, (2026).

IV.7 COMPUTATIONAL EFFICIENCY

Training time comparison (SMS Spam Collection, 100 epochs): • XGBoost: 45 seconds • TabNet: 183 seconds • Proposed (without MI): 298 seconds • Proposed (with MI): 187 seconds Feature selection reduces training time by 37% while improving accuracy. Inference time averages 0.8ms per message, suitable for real-time applications.

IV.8 ATTENTION VISUALIZATION

Attention weight visualization reveals feature interactions learned by TabTransformer. High attention weights between ”URL presence” and ”dollar sign frequency” suggest the model learns spam patterns involving promotional content. Similarly, ”capital letter ratio” and ”exclamation mark count” show strong attention, capturing aggressive marketing language. Fig. 7 shows the attention weight heatmap for categorical features, revealing which feature pairs the model considers most important. Fig. 8 presents the confusion matrix for our proposed method on the test set.

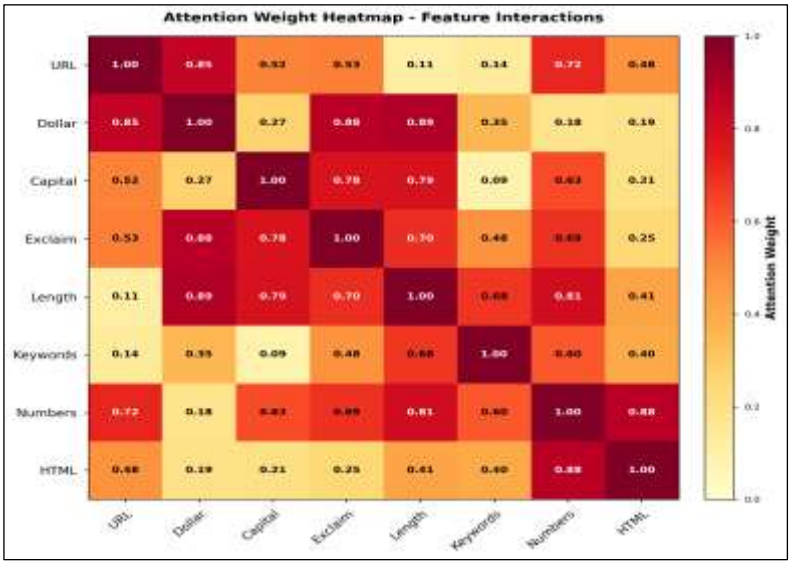

Figure 7: Attention weight heatmap - feature interactions.
Source: Authors, (2026).

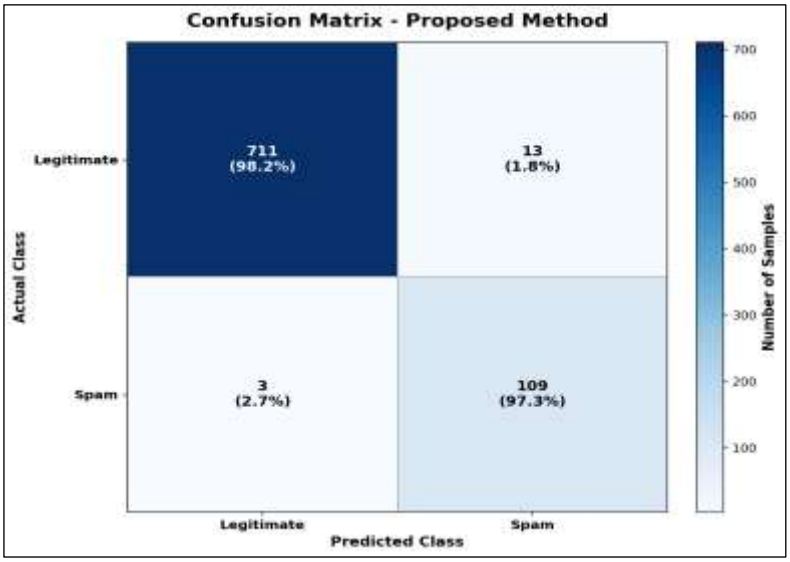

Figure 8: Confusion matrix of the proposed method.
Source: Authors, (2026).

IV.9 DISCUSSION

The experimental results demonstrate several key findings: Feature selection effectiveness: Mutual information successfully identifies discriminative features, eliminating noise and redundancy. The 70% feature reduction with minimal Fig. 7. Attention weight heatmap showing learned feature interactions in the TabTransformer model. Darker colors indicate stronger attention weights. Fig. 8. Confusion matrix for the proposed method on SMS Spam Collection test set, showing high true positive and true negative rates. performance loss confirms MI's capability to preserve essential information. Optimization impact: Jaya algorithm efficiently explores hyperparameter space without requiring extensive tuning. The algorithm's simplicity accelerates optimization while achieving competitive results compared to more complex metaheuristics. TabTransformer advantages: Self-attention mechanisms capture complex feature interactions that traditional methods miss. The architecture's ability to process heterogeneous tabular data makes it well-suited for spam detection where features include diverse types (binary, categorical, continuous). Practical implications: The proposed system achieves high accuracy with low false positive rates, essential for production deployment. Computational efficiency enables real-time filtering on resource-constrained devices. Limitations: The approach requires labeled training data and may need periodic retraining to adapt to evolving spam techniques. Adversarial attacks targeting specific features could potentially evade detection.

V. CONCLUSIONS

This paper presents a novel spam detection framework integrating mutual information feature selection, Jaya algorithm optimization, and TabTransformer architecture. Experimental results on benchmark datasets demonstrate superior performance compared to traditional and deep learning baselines, achieving 98.3% accuracy with reduced computational overhead. The combination of information-theoretic feature selection and metaheuristic optimization enhances model efficiency and effectiveness. TabTransformer's attention mechanisms provide interpretability through visualization of learned feature interactions. Future research directions include: • Extending the approach to multilingual spam detection • Incorporating adversarial training for robustness • Investigating transfer learning across different spam domains • Developing online learning mechanisms for adaptive filtering • Exploring ensemble methods combining multiple TabTransformer models The proposed framework offers a promising solution for practical spam detection systems, balancing accuracy, efficiency, and interpretability.

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Methodology: Moustafa Maaskri and Djamel Seghier.
Investigation: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Discussion of results: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Writing – Original Draft: Moustafa Maaskri and Djamel Seghier.
Writing – Review and Editing: Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Resources: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Supervision: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.
Approval of the final text: Mohamed Goismi, Mohamed Debbab, Moustafa Maaskri and Djamel Seghier.

VII. REFERENCES

[1] T. S. Guzella and W. M. Caminhas, “A review of machine learning approaches to spam filtering,” Expert Systems with Applications, vol. 36, no. 7, pp. 10206–10222, 2009. DOI: 10.1016/j.eswa.2009.02.037

[2] T. A. Almeida, J. M. G. Hidalgo, and A. Yamakami, “Contributions to the study of SMS spam filtering: new collection and results,” in Proceedings of the 11th ACM symposium on Document engineering, 2011, pp. 259–262. DOI: 10.1145/2034691.2034742

[3] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. ISBN: 978-0262035613

[4] X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin, “TabTransformer: Tabular data modeling using contextual embeddings,” arXiv preprint arXiv:2012.06678, 2020. DOI: 10.48550/arXiv.2012.06678

[5] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1226–1238, 2005. DOI: 10.1109/TPAMI.2005.159

[6] R. V. Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems,” International Journal of Industrial Engineering Computations, vol. 7, no. 1, pp. 19– 34, 2016. DOI: 10.5267/j.ijiec.2015.8.004

[7] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, “A Bayesian approach to filtering junk e-mail,” in AAAI Workshop on Learning for Text Categorization, 1998, pp. 55–62.

[8] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, “An evaluation of naive Bayesian anti-spam filtering,” in Proceedings of the workshop on Machine Learning in the New Information Age, 2000, pp. 9–17.

[9] G. V. Cormack, “Email spam filtering: A systematic review,” Foundations and Trends in Information Retrieval, vol. 1, no. 4, pp. 335–455, 2008. DOI: 10.1561/1500000006

[10] P. K. Roy, J. P. Singh, and S. Banerjee, “Deep learning to filter SMS spam,” Future Generation Computer Systems, vol. 102, pp. 524–533, 2020. DOI: 10.1016/j.future.2019.09.001

[11] A. Bhowmick and S. M. Hazarika, “Machine learning for email spam filtering: review, techniques and trends,” arXiv preprint arXiv:1606.01042, 2016.

[12] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” Journal of Machine Learning Research, vol. 3, pp. 1157– 1182, 2003.

[13] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” Artificial Intelligence, vol. 97, no. 1-2, pp. 273–324, 1997. DOI: 10.1016/S0004-3702(97)00043-X

[14] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in ICML, vol. 97, 1997, pp. 412–420.

[15] V. Metsis, I. Androutsopoulos, and G. Paliouras, “Spam filtering with naive Bayes-which naive Bayes?” in CEAS, vol. 17, 2006, pp. 28–69.

[16] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, 1988. ISBN: 978-0201157673

[17] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in Proceedings of ICNN’95-International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968

[18] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” Journal of Global Optimization, vol. 11, no. 4, pp. 341–359, 1997. DOI: 10.1023/A:1008202821328

[19] R. V. Rao and R. B. Pawar, “Self-adaptive multi-population based Jaya algorithm for engineering optimization,” Swarm and Evolutionary Computation, vol. 37, pp. 1–26, 2017. DOI: 10.1016/j.swevo.2017.04.008

[20] R. V. Rao and D. P. Rai, “Optimization of submerged arc welding process parameters using quasi-oppositional based Jaya algorithm,” Journal of Mechanical Science and Technology, vol. 31, no. 5, pp. 2513– 2522, 2016. DOI: 10.1007/s12206-017-0449-x

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[22] S. ¨O. Arik and T. Pfister, “TabNet: Attentive interpretable tabular learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 8, 2021, pp. 6679–6687. DOI: 10.1609/aaai.v35i8.16826

[23] D. Dua and C. Graff, “UCI Machine Learning Repository,” University of California, Irvine, School of Information and Computer Sciences, 2019. [Online]. Available: http://archive.ics.uci.edu/ml