Computing (2026) 108:112
https://doi.org/10.1007/s00607-026-01706-0

REGULAR PAPER

Beyond accuracy: a multi-dimensional green AI framework for sustainable machine learning—energy, carbon, and performance trade-offs in SMS spam detection

Mustafa Aksu[1]

Received: 17 March 2026 / Accepted: 18 June 2026 / Published online: 29 June 2026

Abstract

Until recently, machine learning research has primarily focused on prediction accuracy, often neglecting computational efficiency and environmental impact. In this study, 10 models encompassing classical machine learning, ensemble learning, and deep learning methods were evaluated on the SMS Spam Collection dataset (5,169 messages) using the proposed Multidimensional Green AI Framework. This framework considers models in three dimensions: (i) classification performance (MCC and F1-score), (ii) operational efficiency (p95 inference latency, RAM usage, and model size), and (iii) environmental sustainability (energy consumption in Wh and carbon footprint in kg CO_2). The results reveal a clear accuracy–sustainability trade-off. Although DistilBERT achieved the highest performance (99.13% accuracy, 0.9603 MCC), its marginal gains over simpler models come at a substantial environmental and computational cost. Total pipeline time is approximately 1,720 times longer than Naive Bayes, and training carbon emissions are approximately 1,000 times higher than the most efficient models. Furthermore, CO_2 per one million inferences is 88 times greater than Logistic Regression. In contrast, classical models such as Naive Bayes and Logistic Regression demonstrated competitive performance with significantly lower resource consumption, while ensemble methods, particularly XGBoost, offered a balanced trade-off between accuracy and efficiency. These findings highlight that model selection should not rely solely on accuracy, but must also consider efficiency and environmental impact. Accordingly, this study proposes a practical and environmentally aware Green AI framework for use-specific model selection, supporting classical models for mobile/IoT environments, ensemble methods for edge computing, and deep learning approaches for cloud-based systems where higher resource consumption is acceptable.

Keywords Green AI · Energy Consumption · Sustainable Machine Learning · Carbon Footprint · Deep Learning · SMS Spam Detection

Extended author information available on the last page of the article

1 Introduction

Advancements in the field of artificial intelligence, particularly deep learning and transformer-based models, have enabled achieving accuracy rates of over 99% in SMS spam detection [1]. However, this performance increase comes at a cost known as "Red AI," which requires massive computational resources, high energy consumption, and carbon emissions [2]. The ability of a large-scale NLP model to generate emissions equivalent to the lifetime carbon output of five cars during its training process has opened up discussions on the impact of technological development on environmental sustainability [3–5]. Unlike tasks requiring occasional processing, SMS filtering is a high-frequency, always-on NLP task where the primary environmental burden accumulates during billions of daily real-time inferences. Considering today's energy bottleneck and environmental issues, it has become increasingly important to evaluate models multidimensionally, taking into account their classification success, energy efficiency, and carbon footprint. In this context, the "Green AI" approach has gained strategic importance for the future [6, 7].

Short Message Service (SMS) has been intensively used for communication and information purposes in many areas, both personally and institutionally, since its initial use [8, 9]. In recent years, while internet-based instant messaging platforms have increased in personal use, their intensive use for notifications, communication, advertising, and information purposes by institutions and organizations continues [10–13]. According to the International Telecommunication Union's 2025 report, the total number of mobile cellular subscriptions worldwide is approximately 9.2 billion [14]. Additionally, according to 2025 data, there are approximately 4.2 billion active SMS users worldwide, and daily SMS traffic has reached 22 billion messages [15]. Statistics show that approximately 80% of emails are not read, while SMS messages are read daily at a rate of 98% [16]. This widespread usage rate has made SMS attractive for commercial purposes, cyber attacks, and spammers. As a result, there has been a significant increase in abuses such as fraud, phishing (smishing), and the spread of malware [17–19].

Spam SMS messages can be bothersome in content, but they can also cause financial losses and security breaches. Rule-based filtering methods and blacklisting techniques have been used to eliminate these threats. These traditional methods are inadequate against wordplay, abbreviations, and contextual manipulations used in spam messages [20, 21]. This situation has necessitated the development of machine learning (ML) and deep learning (DL) approaches that can recognize more complex patterns in text analysis [11, 22]. Zapata-Cortes et al. [23], in their literature review on fraud detection, categorized the models into 'Single Base Models' (SBM) and 'Stacked Ensemble Models' (SEM) for examination. As a result of this review, they noted that community methods exhibited superior performance with an accuracy of over 90%.

In this study, the SMS Spam Collection dataset consisting of 5572 messages (5169 unique messages, 12.6% spam) was used [24, 25]. On this dataset, 10 different models (Logistic Regression, Naive Bayes, SVM, Random Forest, XGBoost, LightGBM, LSTM, GRU, CNN, DistilBERT) from the categories of Classical Machine Learning, Ensemble Learning, and Deep Learning were compared. The comparison of the

models was not limited to classical metrics such as accuracy but was conducted with a comprehensive approach on the accuracy-sustainability axis using a Multi-Dimensional Green AI Framework. Model evaluations were conducted under headings such as classification success, computational efficiency, environmental sustainability, and statistical rigor. In addition to these, threshold optimization for a better Matthews Correlation Coefficient (MCC) value and CO_2 emissions measured using the CodeCarbon module are also among the unique features of the study. Additionally, the contributions of energy-efficient models like Logistic Regression and XGBoost, which provide very little accuracy loss, and the usage costs of high-accuracy models like DistilBERT have been revealed. Based on these results, the aim has been to determine the accuracy-sustainability paradox and technically appropriate strategies for model selection in real applications.

2 Related work

Until a few years ago, studies in the field of artificial intelligence traditionally focused on the classification accuracy of models. For the past few years, the high hardware requirements, energy consumption, computational costs, and environmental sustainability aspects of artificial intelligence systems, especially those based on large language models (LLMs), have been discussed. This situation indicates a trend from a resource-intensive approach prioritizing accuracy (Red AI) to an approach that also prioritizes efficiency (Green AI) [2, 6]. Especially on platforms with limited hardware, such as edge/mobile devices, the model size, RAM usage, and latency have become critical success metrics as important as the accuracy rate [20, 26, 27]. This cost increase can also be caused by factors such as data structures, algorithmic complexity, and profit orientation [2]. Şanlıalp et al. [28], in their research focusing on the effects of code optimization on energy consumption, emphasized the importance of optimization in devices powered by batteries, such as mobile phones and laptops.

2.1 Classical models and computational economy

Classical machine learning algorithms remain the most feasible options for resource-constrained edge devices because of their minimal computing expenses [17]. Models such as NB and LR have exceptionally low latency and reduced memory demands [16]. Nuruzzaman et al. [27] shown in their research on an independent Android smartphone that the word frequency-based framework of Naive Bayes offers minimal storage requirements and rapid processing times, eliminating the necessity for intricate vector tables [27]. The experimental results obtained from this study confirm this efficiency; the Naive Bayes model is the fastest model with a total processing time of 0.56 s, generating 0.0036 kg of CO_2 for every million predictions, thus exhibiting a low environmental impact. Logistic Regression, on the other hand, offers a structure that can be easily deployed on mobile devices with a model size of 0.44 MB and the lowest CO_2/1 M prediction ratio (0.0032 kg) among all models.

2.2 Ensemble learning: performance and energy trade-off

Due to their success in balancing accuracy and generalization capacity, ensemble learning models are frequently preferred [29, 30]. Abid et al. [29] demonstrated in their study that the Random Forest model achieved a 99% accuracy rate, showcasing good performance. Sjarif et al. [31] proved in their research that the Random Forest algorithm achieved a 97.5% accuracy rate, surpassing traditional models such as Naive Bayes and SVM. However, the high accuracy rates of up to 99% offered by models like Random Forest (RF) and XGBoost [29] increase computational costs with the growing model depth and number of trees. Fatima et al. [32] focused on hyperparameter optimization in their study to reduce these costs and increase model efficiency. In this context, they have demonstrated that the Stochastic Gradient Descent (SGD) model, supported by genetic algorithms and grid search techniques, achieved an accuracy rate of 98.12% on the SMS Spam dataset, surpassing ensemble models. Similarly, Kumar and Gupta [30] highlighted the necessity of feature pruning techniques by stating that training high-dimensional feature vectors is costly and can lead to memory issues on mobile devices. In this study, it was observed that Random Forest was the most stable model with a generalization difference of 0.0048, while XGBoost was found to offer an "optimal balance" between high performance and low resource usage with an astonishingly small model size of 0.13 MB and an average latency of 0.78 ms.

2.3 Meta-heuristic optimization and resource efficiency

Achieving a balance between accuracy and computational cost in artificial intelligence systems is one of the important research topics from the Green AI perspective. In this context, nature-inspired meta-heuristic algorithms are increasingly attracting attention, particularly for their potential to reduce computational complexity and energy consumption in model hyperparameter tuning and architecture optimization processes. Genetic Algorithms (GA), Simulated Annealing (SA), and Bayesian Optimization can be considered as established approaches in this field; meanwhile, swarm intelligence and evolutionary methods such as Particle Swarm Optimization (PSO), Differential Evolution (DE), Grey Wolf Optimizer (GWO), Firefly, and Cricket have also demonstrated their success in enhancing resource efficiency without compromising model accuracy [32–34].

Alhussan et al. [35] proposed a hybrid BERPO approach combining Al-Biruni and Puma Optimization for quality of service classification in 5G networks. This method achieved 99.6% accuracy by optimizing feature selection and data balancing processes. El-Kenawy et al. [36] developed the Glider Snake Optimizer (GSO) algorithm for high-dimensional optimization problems. GSO provided up to 90% error reduction compared to existing algorithms. El-Kenawy et al. [37] presented a hybrid ARIMA model supported by Greylag Goose Optimization for electricity demand forecasting in smart cities. The model minimized computational load while achieving a high forecasting accuracy of R^2=0.99995. Radwan et al. [38] significantly reduced the computational cost of hyperparameter tuning and feature selection by combining the binary iHOW optimization algorithm with a Multi-Scale Attention Network.

The common finding of these studies shows that nature-inspired optimization techniques can reduce resource consumption while maintaining model performance. These findings align with the proposed Multi-Dimensional Green AI Framework in the study and lay the groundwork for the integration of hyperparameter optimization and model compression techniques in future SMS spam detection systems.

2.4 Deep learning: carbon footprint, energy and memory

Deep learning (DL) architectures are known for their ability to capture temporal and spatial dependencies in text, but high RAM usage makes it difficult for these models to run on low-configuration devices [11, 21]. Recurrent neural networks (RNN) such as LSTM and GRU are not suitable for parallel processing due to their sequential computation structures, which extends training times and increases energy consumption [2, 39]. Altunay and Albayrak [11] have noted that despite the accuracy advantage of hybrid CNN + GRU models, the requirement for memory capacity is a significant constraint.

To reduce these costs, techniques such as model quantization and pruning are widely discussed in the literature with the aim of reducing the model's size without significantly affecting its accuracy [2, 7]. Paula et al. [40] empirically compared multiple transformer variants (BERT, DistilBERT, ELECTRA, TinyBERT, MobileBERT) in terms of energy consumption and carbon emissions using CodeCarbon with hardware-level CPU and GPU monitoring. Notably, DistilBERT consumed 53% less energy than BERT (3.364 kWh vs. 7.197 kWh) with only a marginal 0.27% accuracy difference (95.88% vs. 96.15%), demonstrating that compression techniques can reduce energy consumption by up to 32% while maintaining accuracy above 95%. Savci and Daş [41] compared 12 different pre-trained transformer models (BERTurk, ConvBERTurk, ElecTRa, DistilBERTurk) in a Turkish multi-class text classification task with 250,000 samples. They found that CO_2 emissions differed by up to 317 times among models with similar accuracy levels (0.61 vs. 193.80). This result shows that model selection has serious consequences not only in terms of performance but also in terms of the environment. Yokoyama et al. [42], on the other hand, examined the energy consumption of classical machine learning algorithms such as XGBoost and Random Forest in different hardware configurations. It was observed that hardware architecture and energy source could significantly change the carbon footprint of the same model.

The experimental data obtained in this study corroborate these findings: deep learning models (CNN, LSTM, GRU, DistilBERT) require between 97 and 291 MB of RAM during inference and CO_2 emissions per one million predictions are approximately 53 times higher compared to classical models (0.2015 kg vs. 0.0038 kg), while training emissions are on average 137 times greater (0.000593 kg vs. 0.000004 kg) — confirming that empirical energy tracking remains absent from SMS spam detection literature prior to this work, a gap directly addressed by the present study.

2.5 Transformer models and "Red AI" cost

Building on the empirical energy comparisons presented in Sect. 2.4, transformer-based models represent the most resource-intensive end of the computational spectrum in NLP tasks.

Transformer-based models (BERT, DistilBERT) are leading in SMS spam detection with over 99% accuracy performance [1, 39]. However, the massive number of parameters and "Attention" mechanisms of these models make them high-cost examples of "Red AI" [7]. The training of a single large language model (LLM) can sometimes produce emissions equivalent to five times the amount of carbon emitted by a car over its lifetime [4, 7]. Although the DistilBERT model used in this study in this study achieved the highest classification success with a 0.9603 MCC score, it requires 1720 times longer than classical models, specifically Naive Bayes, in terms of total development time (963.15 s vs. 0.56 s), due to its 255.54 MB model size. This situation indicates that the approximately 3.5% MCC improvement offered by DistilBERT is not operationally sustainable on low-configuration mobile devices or systems with limited energy budgets.

The comparative literature review in Table 1 shows that various models are successful in SMS spam detection with an accuracy range of 97–99%. However, none of the studies reviewed provided empirical measurements for energy consumption or carbon emissions. Studies using transformer-based models such as BERT and DistilBERT [1, 39] theoretically acknowledge high operational costs. In contrast, classical and ensemble model studies [17, 29, 31] focused only on classification accuracy. This gap in the literature formed the main motivation for the empirical Green AI framework proposed in this study. The study presents the first quantitative energy and carbon comparison on 10 models in the field of SMS spam detection, and the findings are given in Sects. 4–5.

3 Materials and methods

This study is based on a multidimensional methodology that considers not only high accuracy performance of models in SMS spam detection but also energy efficiency, computational costs, and real-time deployment requirements (Fig. 1). The process consists of the stages of data set analysis, advanced preprocessing techniques, feature extraction, and modeling.

3.1 Dataset and exploratory data analysis

In this study, the "SMS Spam Collection" dataset, considered important in the literature for SMS spam detection and located in the UCI Machine Learning Repository, was used [24, 25]. The original dataset consisted of 5572 messages. In the preprocessing stage, performed to improve data quality, tag conflicts were checked and 403 duplicate records were removed. This left 5169 unique messages. The cleaned dataset contained 4516 ham (87.37%) and 653 spam (12.63%) messages. Statistical analyses of the structural features of the messages revealed significant differences between raw

Table 1 Performance and Sustainability-Focused Comparison of SMS Spam Detection Approaches

Study	Core models and approach	Performance (accuracy/MCC)	Efficiency and sustainability note
Results of This Study	LR, NB, SVM, RF, XGBoost, LGBM, LSTM, CNN, GRU, DistilBERT	%99.13 Accuracy/0.9603 MCC (DistilBERT)	Empirical measurement of CO_2 emissions (e.g., 0.002240 kg for DistilBERT) and energy (e.g., 0.004822 kWh for DistilBERT) via CodeCarbon; precise tracking of RAM and p95 inference latency (0.31–86.26 ms) across 10 models.
Airlangga, G. [26]	CNN-LSTM, CNN-GRU, ResNet	%99.08 Accuracy (ResNet)	Analyzed model parameter counts and algorithmic complexity, but lacked empirical measurements of physical energy consumption (Wh) or carbon footprint.
Ghourabi, A., & Alohaly, M. [1]	BERT & GPT-3 Embedding + Ensemble Learning	%99.91 Accuracy (Weighted Voting)	Discussed the high operational costs of transformer models theoretically, but did not provide empirical tracking of hardware energy consumption or inference latency.
Sjarif, N. N. A. et al. [31]	TF-IDF + Random Forest (RF)	%97.5 Accuracy (RF)	Evaluated algorithms solely based on classification accuracy; operational efficiency (inference speed, training time) and sustainability metrics were excluded.
Abid, M. A. et al. [29]	TF-IDF + RF + SMOTE	%99.0 Accuracy (RF)	Achieved high accuracy on imbalanced data, but the computational overhead (CPU/memory cost) resulting from increased tree depth in the Random Forest model was not analyzed.
Pudasaini, S. et al. [17]	RVM, SVM, Naive Bayes, LSTM	%98.84 Accuracy (RVM)	Noted that RVM inference time is comparable to SVM, though its training time is longer due to "prior information" computations; specific energy or carbon metrics were not provided.
Salman, M. et al. [18]	TCSVM, Word2Vec, GloVe, fastText	%97.8 Accuracy (TCSVM)	Focused on model robustness against evasion (e.g., "Charswap"); however, the impact of these defense mechanisms on real-time filtering latency and energy load was not reported.
Zawaideh, F. et al. [13]	SVC, Naive Bayes, AdaBoost	%98.02 Accuracy (SVC)	Maximized accuracy through extensive hyperparameter optimization (BoW/TF-IDF), but the resulting hardware resource consumption (RAM/Energy) was not quantified.

and spam messages. In terms of character length, spam messages reach an average of 137.89 characters, nearly double that of raw messages (70.46 characters). In terms of word count, spam messages contain an average of 23.68 words, while raw messages are shorter with 14.13 words. Word frequency analysis has shown that spam content is concentrated with commercial and urgency-indicating keywords such as "free," "call," "text," and "win," while raw messages contain a more general and personal language. These observations demonstrate that word frequency and text length will be strong distinguishing parameters for text classification algorithms [17, 43].

3.2 Data preprocessing

Training Natural Language Processing (NLP) models with noise-free, clean data not only enhances classification performance but also minimizes computational costs by eliminating unnecessary features [7]. In this context, a preprocessing strategy specific

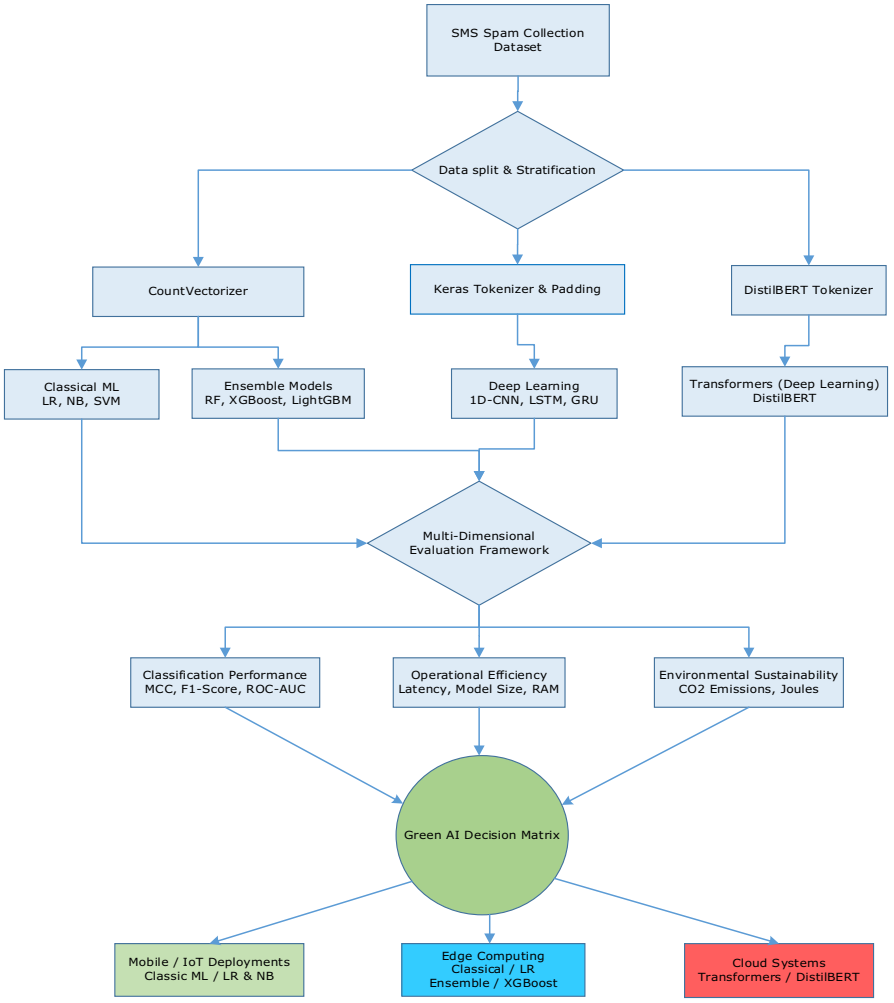

Fig. 1 Multi-dimensional model evaluation and Green AI-based decision framework methodology for SMS spam detection

to the model types was applied in this study using the Python, NLTK, and Beautiful-Soup libraries.

3.2.1 Basic preprocessing (for all models)

Some data preprocessing steps were applied uniformly to all models and include the following steps:

Text cleaning: HTML tags and special characters were removed, and Unicode normalization (NFKC) was applied.

Semantic masking: To enhance the generalization ability of the models, phone numbers were masked with the `< PHONE> ` label, email addresses with the `<

EMAIL> ` label, URLs with the `< URL> ` label, and numerical values with the `< NUM> ` label.

Emoji conversion: To prevent semantic loss in messages, emojis were converted into textual expressions (like → “smiling_face”).

Character normalization: The entire texts were converted to lowercase, and repeated characters were normalized (e.g., “Goooood” → “good”).

These preprocessing preferences aim to reduce feature space complexity. Semantic Masking replaces thousands of unique tokens (such as specific URLs or phone numbers) with a single tag. This significantly reduces input vocabulary size and computational load during training. Emoji Conversion preserves the emotional context critical to spam, and does so without requiring a separate emoji-supported embedding layer, thus maintaining model lightness.

3.2.2 Extended preprocessing for classical machine learning

In addition to the basic preprocessing for classical machine learning models, the following steps have been applied:

Abbreviation (chat) transformation: 93 slang/abbreviations commonly found in SMS, such as "u" (you), "r" (are), "ASAP" (As Soon As Possible), have been replaced with their standard forms to ensure semantic coherence in the text.

Filtering out stop words: English "stop-words" (the, a, is, etc.) that do not have a direct impact on classification have been removed.

Morphological analysis: WordNetLemmatizer has been used to obtain the contextual roots of words ("running" → "run").

Lemmatization was specifically applied to classical models to normalize morphological variations into a single root. While this adds a one-time computational overhead during preprocessing, it substantially reduces the dimensionality of the CountVectorizer matrix, leading to faster inference and lower RAM usage on mobile devices.

3.2.3 Minimal preprocessing for deep learning

Since deep learning models such as LSTM, GRU, and CNN can learn word representations through their own embedding layers, only basic preprocessing (Section Basic preprocessing (for all models)) has been applied to these models. Operations such as chat transformation, stop-word filtering, and lemmatization have not been used because they could limit the learning capacity of deep learning models.

3.3 Feature extraction and vectorization

Two different vectorization strategies specific to model types have been adopted to make textual data suitable for algorithmic processing:

Count vectorization (for classical machine learning models): CountVectorizer has been used to create word frequency-based feature matrices for Naive Bayes, Logistic Regression, Support Vector Machines, and ensemble learning models. This

method generates a sparse matrix structure by counting how many times words appear in documents and increases computational efficiency [2].

The vectorizer configuration is as follows:

N-gram range: (1,3) - from single words to three-word groups

min_df: 1 (include all terms that appear in at least 1 document)

max_df: 1.0 (include all terms)

binary: False (preserve frequency information)

With this configuration, the vocabulary size extracted from the training set was determined to be 57376 features, and the training matrix was created with dimensions (4135 × 57376).

Tokenization and sequence padding (for deep learning models): Texts have been converted into integer sequences using the Keras Tokenizer library for deep learning models such as LSTM, GRU, and CNN. In this process, the following were done:

Vocabulary size: 10,000 words (most frequently used words)

Out-of-vocabulary (OOV) token: "<OOV>" (special marker for words outside the vocabulary)

Sequence length: 100 tokens (standardization with post-padding and post-truncating)

Padding strategy: Adding zeros to the end of short messages, truncating long messages from the end.

The tokenizer was fitted only on the training set to prevent data leakage, and the same vocabulary was used for the transformation on the test set. The actual vocabulary size obtained from the training set was 7816 unique tokens, and the training matrix was created with dimensions (4135 × 100).

3.4 Model training, hyperparameter optimization, and environmental monitoring

Considering the imbalanced dataset and algorithmic diversity for the spam detection problem, 10 algorithms proven effective in the field of text classification in the literature have been compared in three categories:

- Classical Machine Learning (3): Logistic Regression (LR), Naive Bayes (NB), Support Vector Machines (SVM)

- Ensemble Learning (3): Random Forest (RF), XGBoost (XGB), LightGBM (LGBM)

- Deep Learning (4): LSTM, GRU, CNN, DistilBERT (Transformer-based)

Performance metrics: The success of the models has been evaluated using the following metrics (Eq. 1, 2, 3, 4, 5, 6):

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \tag{1}$$

$$precision = TP / (TP + FP) \tag{2}$$

$$Recall\ (Sensitivity) = TP / (TP + FN) \tag{3}$$

$$F1 - Score \;=\; 2 \;\times\; (Precision \;\times\; Recall) \;/\; (Precision \;+\; Recall) \quad (4)$$

$$Specificity \;=\; TN \;/\; (TN \;+\; FP) \quad (5)$$

$$ROC - AUC \;=\; \int_0^1 TPR\,(FPR)\,dFPR \quad (6)$$

where

TP, TN, FP, FN: Confusion matrix elements

TPR= TP/(TP+FN), FPR = FP/(FP+TN)

Hyperparameter optimization: In classical and ensemble models, 5-fold stratified CV (scoring: PR-AUC) was applied using GridSearchCV. In deep learning models, a manual 5-fold CV loop was used; the LSTM/GRU bidirectional architecture (64→32 units, dropout=0.5, embedding_dim=128, vocabulary=10,000), the CNN convolutional structure (128→64 filters, kernel_size=5→3), and DistilBERT as the pretrained model with 66M parameters (max_length=128, learning_rate=2e-5, epochs=3, batch_size=16) were configured. For deep learning models (LSTM, GRU, CNN), no exhaustive grid search was conducted due to prohibitive computational costs — consistent with Green AI principles. Architecture configurations were manually evaluated across 5-fold CV with a maximum of 15 epochs per fold (batch_size=32), resulting in up to 75 total training passes per model. EarlyStopping (patience=3, monitor=val_loss) and ReduceLROnPlateau (factor=0.5, patience=2, min_lr=1e-6) callbacks were applied throughout both CV and final training, reducing actual training to an average of 8–12 epochs per fold and yielding an estimated 20–40% reduction in energy consumption compared to fixed-epoch training. DistilBERT was fine-tuned for a fixed 3 epochs with EarlyStopping (patience=1) as an energy-saving safeguard, as convergence was verified during CV runs.

Threshold optimization: The decision threshold (Eq. 7) for all models has been optimized to maximize the Matthews Correlation Coefficient (MCC) (Eq. 8):

Optimal Threshold Determination:

$$t^* \;=\; \arg\max\; MCC\,(t)\,,\; t \in [0.01,\; 0.99] \quad (7)$$

Matthews Correlation Coefficient:

$$MCC = \frac{TP \times TN \;-\; FP \times FN}{\sqrt{[(TP+FP)\,(TP+FN)\,(TN+FP)\,(TN+FN)]}} \quad (8)$$

where

t*: Optimal decision threshold

t ∈ [0.01, 0.99]: Search range (evaluated at 100 steps)

MCC ∈ [-1, 1]: -1 (perfect disagreement), 0 (random), +1 (perfect)

This process was performed using 5-fold cross-validation estimations to reduce the risk of overfitting. To ensure validity, all feature extraction steps (CountVectorizer and Tokenizer) were applied only to the training folds. This prevented any data leakage from the test set to the thresholding process. Thanks to this approach, the

vocabulary of the test set remains unseen during the feature extraction phase, allowing for a more realistic evaluation of the models' performance on new data.

Environmental monitoring: In line with "Green AI" goals, energy consumption and carbon emissions have been measured throughout all training and inference processes. CO_2 emissions have been calculated using the CodeCarbon library through the following formula (Eq. 9):

$$CO_2\,(kg) \;=\; P\,(kW) \;\times\; t\,(h) \;\times\; CEF\,(kgCO_2/kWh) \tag{9}$$

where:

P: Average power consumption (CPU/GPU)

t: Total runtime (training + inference)

CEF: Regional carbon emission factor

The Carbon Emission Factor (CEF) was dynamically determined by CodeCarbon according to the regional energy grid (Turkey) [45] and the average value was measured as 0.465 kg CO_2/kWh. This value reflects a grid mix based predominantly on fossil fuels. For comparison, if the same calculation scheme were used in a region with high renewable energy intensity (e.g., Sweden, CEF~0.012), the absolute CO_2 emissions would decrease by approximately 97% without changing the energy consumption (Wh) of the models.

Additionally, RAM usage (MB), inference latency (mean, p95, p99), model size (MB), and generalization capacity (train-test accuracy gap) have been systematically recorded using the CodeCarbon and psutil libraries.

All models were trained and evaluated on the same train-test split (80%-20%, stratified, random_state=42).

3.5 Experimental setup and environment

The model trainings, hyperparameter optimizations, and duration measurements in this study were fairly compared by being conducted on a device with the same hardware. The experiments were conducted on a device with an Intel Core i7-10750H 2.60 GHz processor, 16 GB DDR4 memory, NVIDIA GeForce RTX 2060 6GB graphics card, and Windows 11 64-bit operating system. Python 3.10 was used in the Jupyter Notebook development environment for the software. For classical models, the scikit-learn (v1.6.1) library was used, and for deep learning models, the TensorFlow/Keras (v2.10.0) libraries were used. For environmental impact (CO2) and energy consumption calculations, the CodeCarbon (v3.2.5) library was used.

4 Experimental results

In this section, 10 different models trained on the SMS Spam Collection dataset were examined in terms of classification success, computational efficiency, environmental sustainability, and statistical rigor. The findings reveal not only the prediction success of the models but also their operational costs in terms of "Green AI" [2, 6].

4.1 Classification performance and model metrics

The dataset used has an imbalanced distribution (Sect. 3.1). Therefore, in the evaluation of the models' success, the Matthews Correlation Coefficient (MCC), a more reliable metric for imbalanced datasets, has been used instead of Accuracy [17, 44].

In Table 2, the performance metrics of the models sorted by their MCC scores are shown. According to these results, the most successful model of the study was DistilBERT, with an accuracy of 99.13% and an MCC score of 0.9603. Especially the 100% precision rate indicates that DistilBERT did not mistakenly label any normal messages as spam.

When examining the average MCC performance of the models by category:

- Deep Learning: MCC 0.9354 (highest).
- Classical ML: MCC 0.9203.
- Ensemble: MCC 0.9046.

These results indicate that deep learning models are generally better in classification performance [26]. However, the rise of Naive Bayes to the 4th position in the overall rankings (MCC: 0.9250) indicates that probabilistic models are still competitive in text classification.

4.2 Computational efficiency and mobile device compatibility

4.2.1 Inference latency and model size

In real-time SMS filtering systems, the prediction speed (latency) and size of the models are crucial for operational sustainability. Table 3 presents the mobile deployment metrics of the models.

The operability of the models on low-configuration devices (mobile, edge devices, etc.) has been evaluated based on model size, RAM usage, and inference latency [20, 27]. Logistic Regression stands out as the most suitable model for mobile platforms with a 0.31 ms p95 latency and a size of 0.44 MB. DistilBERT, with a size of 255.5

Table 2 Comparison of Models' Classification Performance in SMS Spam Detection (Sorted by MCC Score)

	Model	Category	Accuracy	Precision	Recall	F1-Score	MCC	ROC-AUC
1	DistilBERT	Deep Learning	0.9913	1.0000	0.9313	0.9644	0.9603	0.9983
2	GRU	Deep Learning	0.9865	1.0000	0.8931	0.9435	0.9378	0.9897
3	LSTM	Deep Learning	0.9865	0.9916	0.9008	0.9440	0.9377	0.9871
4	NaiveBayes	Classical	0.9836	0.9453	0.9237	0.9344	0.9250	0.9841
5	LR	Classical	0.9826	0.9248	0.9389	0.9318	0.9219	0.9911
6	SVM	Classical	0.9807	0.9111	0.9389	0.9248	0.9139	0.9935
7	XGBoost	Ensemble	0.9807	0.9302	0.9160	0.9231	0.9121	0.9849
8	CNN	Deep Learning	0.9797	0.9583	0.8779	0.9163	0.9059	0.9792
9	LightGBM	Ensemble	0.9787	0.9160	0.9160	0.9160	0.9038	0.9873
10	RandomForest	Ensemble	0.9778	0.9286	0.8931	0.9105	0.8980	0.9887

Table 3 Comparison of System Resource Consumption and Inference Efficiency of Evaluated Models (Sorted by p95 Latency)

Rank	Model	Category	Model Size (MB)	RAM Inference (MB)	Latency p95 (ms)	Latency p99 (ms)
1	LR	Classical	0.4386	0.0117	0.3052	0.4881
2	NaiveBayes	Classical	1.7517	0.0117	0.4797	0.6272
3	XGBoost	Ensemble	0.1324	0.0117	0.8091	0.9589
4	SVM	Classical	0.3440	0.0273	0.8797	0.9696
5	RandomForest	Ensemble	0.8720	0.0234	10.9491	12.2899
6	CNN	Deep Learning	4.2566	97.4648	44.2819	48.1349
7	LSTM	Deep Learning	4.3969	256.1641	59.3449	72.1265
8	GRU	Deep Learning	4.2660	233.3242	60.6467	74.4037
9	DistilBERT	Deep Learning	255.5393	291.2266	86.2640	91.8661
10	LightGBM	Ensemble	1.1108	0.0117	204.1804	214.4331

Note: The RAM inference values (0.0117 MB) for classical and ensemble models reflect the minimum measurable resolution of the psutil library, likely corresponding to standard OS memory page sizes (3×4 KB ≈ 12 KB), indicating that their actual memory consumption during inference is negligible.

MB, requires 583 times more disk space than Logistic Regression and creates difficulty in real-time mobile applications with a p95 latency of 86.26 ms.

In terms of RAM usage, classical ML and ensemble models consume negligible memory during inference, while deep learning models average 219.5 MB (excluding DistilBERT: 195.7 MB). XGBoost offers the most compact solution in terms of storage efficiency with a model size of 0.13 MB.

Average latency times (p95) by category:

- Classic ML: 0.55 ms (mobile compatible).
- Deep Learning: 62.63 ms (cloud recommended).
- Ensemble: 71.98 ms (edge computing compatible).

These findings indicate that the preference for classic ML models in resource-constrained environments (mobile devices, IoT) is essential in terms of both latency and model size [7]. When examining the visual in Fig. 2, the high p95 latency value of 204.18 ms for LightGBM indicates that ensemble methods can show unexpected increases in prediction time when processing sparse high-dimensional feature vectors.

4.2.2 Training and development durations

In real-time SMS filtering systems, the prediction speed of models and total development times are of great importance in terms of operational continuity [44]. Table 4 presents the time performance of the models in the cross-validation, training and inference processes in detail.

When Table 4; Fig. 3 are examined, Naive Bayes was the fastest model in terms of development with a total time of 0.56 s. Logistic Regression provided the highest efficiency in real-time applications with an average delay of 0.29 ms per sample. In contrast, DistilBERT was recorded as the slowest model with a total time of 963.15 s (approximately 16 min) due to its complex transformer architecture.

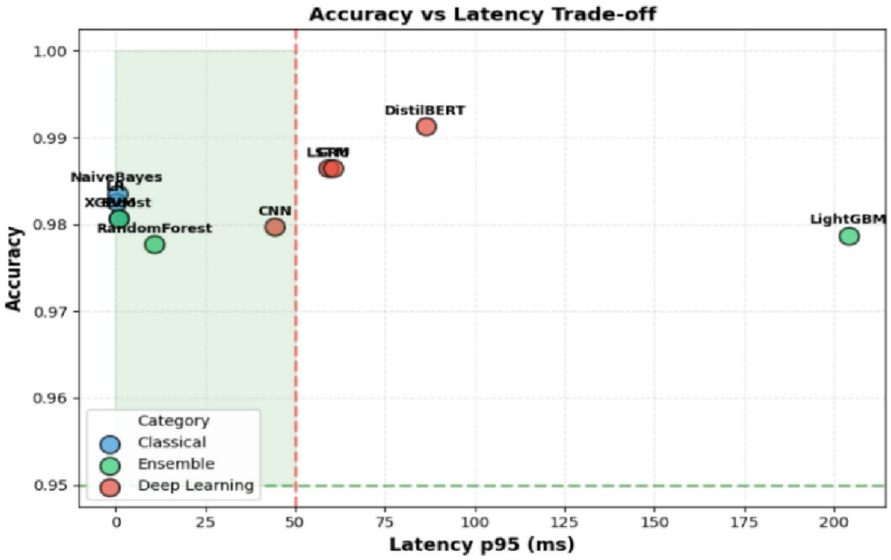

Fig. 2 Trade-off Analysis of Classification Accuracy and Inference Latency of Models

Table 4 Comprehensive Time Cost and Computational Efficiency Analysis of Evaluated Models (Sorted by Total Development Time)

Model	Category	CV (s)	Training (s)	Inference (s)	Total (s)	Latency Mean (ms)
NaiveBayes	Classical	0.1776	0.0029	0.3796	0.5601	0.3671
LR	Classical	1.3220	0.0648	0.2980	1.6848	0.2882
SVM	Classical	19.2389	2.0084	0.8034	22.0507	0.7770
RandomForest	Ensemble	36.2483	0.6655	9.9246	46.8384	9.5983
XGBoost	Ensemble	58.0378	1.3179	0.8032	60.1590	0.7768
CNN	Deep Learning	28.4088	6.8563	42.4287	77.6937	41.0335
LightGBM	Ensemble	29.9633	0.1782	104.9502	135.0918	101.4993
GRU	Deep Learning	108.1223	22.7599	56.4830	187.3652	54.6257
LSTM	Deep Learning	119.9789	29.8895	56.0822	205.9506	54.2381
DistilBERT	Deep Learning	721.0333	171.3013	70.8128	963.1474	68.4844

Significant differences have been observed in hyperparameter optimization (CV) times:

- DistilBERT: 721.03 s (slowest CV - manual 5-fold loop).
- LSTM: 119.98 s, GRU: 108.12 s (deep learning).
- SVM: 19.24 s (kernel computation cost).
- Naive Bayes: 0.18 s (fastest CV - probabilistic simplicity).

The differences are quite striking when looking at inference times on a model-by-model basis. Logistic Regression was the fastest model with an inference time of 0.30 s across the entire test set. LightGBM, on the other hand, was the slowest at 104.95

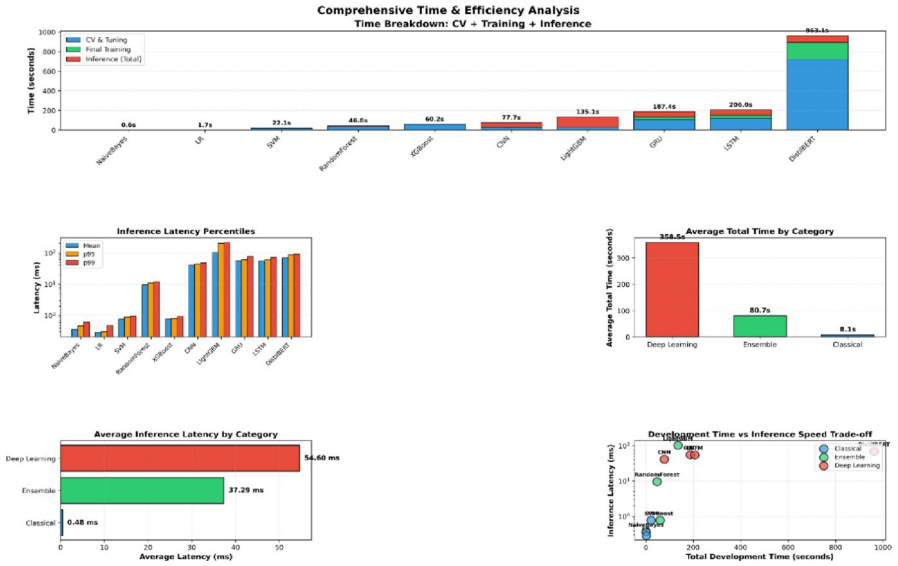

Fig. 3 Panel of Model Time Cost and Computational Efficiency: Training Processes, Delay Distributions, and Category-Based Performance Analysis

s. This difference suggests that LightGBM may encounter unexpected computational overhead in high-dimensional sparse feature spaces.

Average inference latencies (latency mean) by model category:

- Classical ML: 0.48 ms (ultra-fast).
- Deep Learning: 54.60 ms (moderate).
- Ensemble: 37.30 ms (moderate-to-high).

When examining the average model development times (CV + Training + Inference), it is observed that classical models require 8.10 s, ensemble models require 80.70 s, and deep learning models require 358.54 s. These results indicate that classical models provide time efficiency in processes such as prototype development and iterative model improvement. Additionally, these findings indicate that the preference for classical models in edge/mobile devices with limited resources is necessary in terms of both development speed and inference performance [7, 27].

4.3 Environmental sustainability and energy consumption

The environmental metrics measured with the CodeCarbon module reveal the carbon emissions and energy consumption of the models from a "Green AI" perspective [6, 7]. The CO_2 emissions, energy consumption, and energy costs per unit prediction for all models are provided in Table 5.

According to the findings in Table 5 and the visual in Fig. 4, Logistic Regression is the most environmentally friendly model with an inference CO_2 emission of 0.000003 kg, while DistilBERT causes approximately 373 times more carbon emis-

Table 5 Analysis of Environmental Sustainability and Energy Consumption of Models (Sorted by Inference CO_2 Emission)

Model	Training CO2 (kg)	Training Energy (kWh)	Energy/1000 train (Wh)	Inference CO2 (kg)	Inference Energy (kWh)	Energy/1000 pred (Wh)	CO2/1 M pred (kg)
LR	0.000003	0.000006	0.001355	0.000003	0.000007	0.006879	0.003196
NB	0.000002	0.000005	0.001256	0.000004	0.000008	0.007655	0.003557
SVM	0.000008	0.000018	0.004392	0.000005	0.000010	0.009962	0.004628
XGBoost	0.000010	0.000022	0.005292	0.000005	0.000012	0.011448	0.005319
RF	0.000004	0.000009	0.002265	0.000033	0.000071	0.068309	0.031735
CNN	0.000054	0.000115	0.027871	0.000142	0.000305	0.295173	0.137133
LSTM	0.000224	0.000483	0.116711	0.000200	0.000430	0.415943	0.193241
GRU	0.000143	0.000309	0.074666	0.000201	0.000432	0.418098	0.194243
DistilBERT	0.001949	0.004195	1.014400	0.000291	0.000627	0.606027	0.281552
LightGBM	0.000003	0.000006	0.001536	0.000344	0.000740	0.715450	0.332388

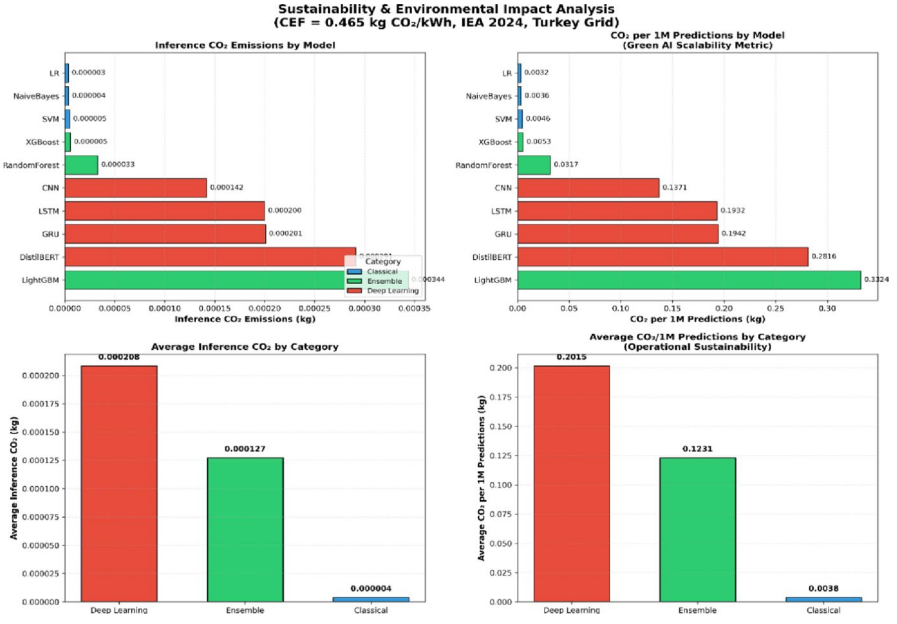

Fig. 4 Environmental Impact and Sustainability Panel of Models: Categorical Distribution of Carbon Emission (CO2) and Energy Consumption

sions than Naive Bayes with a total CO_2 of 0.002240 kg. This situation indicates that the environmental costs of deep learning models increase rapidly compared to classical models due to their high computational requirements.

Energy consumption of models for 1000 predictions:

- Logistic Regression: 0.0069 Wh (ultra-efficient).
- Naive Bayes: 0.0077 Wh (ultra-efficient).
- XGBoost: 0.0114 Wh (efficient).

- CNN: 0.2952 Wh (moderate).
- GRU/LSTM: ~0.417 Wh (high).
- DistilBERT: 0.6060 Wh (resource-intensive).

These findings show that achieving a 3.84% improvement in the MCC metric (LR: 0.9219 → DistilBERT: 0.9603) requires 88 times more carbon footprint in terms of environmental cost. In large-scale deployment scenarios where millions of predictions are made daily, this difference translates to significant energy costs and carbon emissions.

For example, in a system making 1 million predictions per day:

- Logistic Regression: 0.0069 kWh/day (~ 1.17 kg CO_2/year).
- DistilBERT: 0.6060 kWh/day (~ 102.86 kg CO_2/year).

This comparison clearly demonstrates that model selection should be evaluated not only for accuracy but also for operational sustainability in line with "Green AI" principles [6].

4.4 Model generalization and statistical stability

The consistency of the models on the training and test data was examined through the accuracy-based generalization difference [44]. Table 6 presents the training-test accuracies, generalization differences, and cross-validation stability metrics of all models.

Random Forest was the most robust model against changes in the dataset with an accuracy-based generalization difference of 0.0048. This low difference value confirms that ensemble methods are inherently strong against overfitting. DistilBERT, despite its high accuracy, exhibited acceptable stability with a difference of 0.0056.

Although the LR and SVM models achieved 100% accuracy in the training set, it is noteworthy that their accuracy-based generalization differences remained below 0.02. This indicates that the applied methodology (5-fold stratified CV, MCC-based threshold optimization, comprehensive preprocessing) successfully controlled overlearning.

Table 6 Generalization Gap and Stability Analysis Between Training and Test Performance of the Models

Model	Category	Train Accuracy	Test Accuracy	Gen. Gap (Acc)	Gen. Gap (MCC)	CV Acc. Mean	CV Acc. Std	Risk Level
RF	Ensemble	0.9826	0.9778	0.0048	0.0216	0.9799	0.0041	Low
DistilBERT	Deep Learning	0.9969	0.9913	0.0056	0.0254	0.9889	0.0034	Low
GRU	Deep Learning	0.9971	0.9865	0.0106	0.0490	0.9780	0.0037	Low
XGBoost	Ensemble	0.9918	0.9807	0.0111	0.0504	0.9780	0.0027	Low
LSTM	Deep Learning	0.9993	0.9865	0.0128	0.0590	0.9843	0.0032	Low
LightGBM	Ensemble	0.9940	0.9787	0.0152	0.0685	0.9802	0.0039	Low
NB	Classical	0.9993	0.9836	0.0157	0.0717	0.9574	0.0071	Low
LR	Classical	1.0000	0.9826	0.0174	0.0781	0.9821	0.0032	Low
CNN	Deep Learning	0.9973	0.9797	0.0176	0.0820	0.9741	0.0044	Low
SVM	Classical	1.0000	0.9807	0.0193	0.0861	0.9826	0.0035	Low

Risk level status (gap):

- High Risk (gap > 0.10): 0 models.
- Medium Risk (0.05 < gap ≤ 0.10): 0 models.
- Low Risk (gap ≤ 0.05): 10 models (all models).

CV stability status (CV Accuracy Std):

- Most stable: XGBoost (std: 0.0027), LR, LSTM (std: 0.0032).
- Most variable: NB (std: 0.0071), CNN (std: 0.0044).

The training-test accuracy comparison, gap risk analysis, stability of classical and ensemble models, and robustness of deep learning models are detailed in Fig. 5.

The fact that the accuracy-based generalization difference of all models remains below 0.02 indicates that the data splitting strategy (stratified splitting), cross-validation protocol, and threshold optimization processes applied in the study are reliable. These results provide strong evidence that the models can exhibit similar performance in real-world applications.

4.5 Multidimensional model evaluation and usage recommendations

When the findings of this study are evaluated from multiple perspectives, it shows that there is no single "best model"; rather, the choice should be made according to the use case, system requirements, and organizational priorities. Table 7 shows

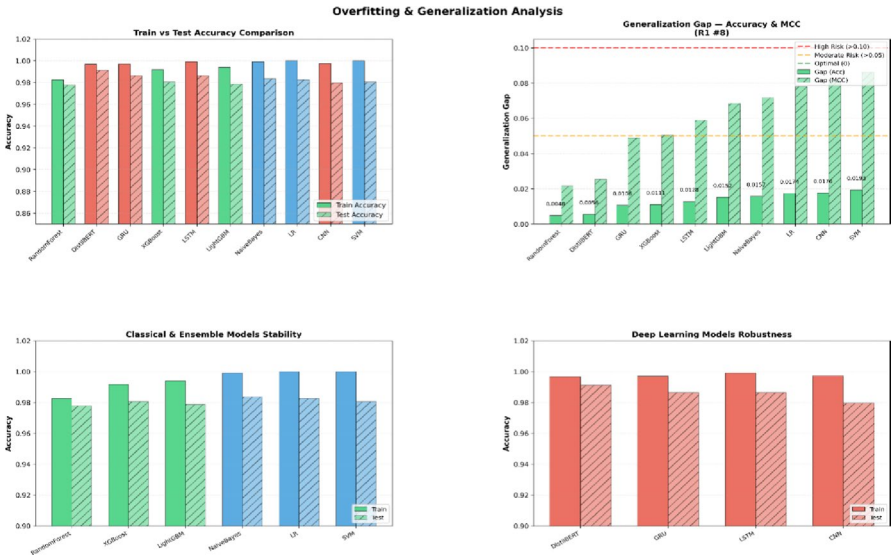

Fig. 5 Model Generalization Capacity and Overfitting Analysis: Differences Between Training-Test Performance (Generalization Gap) and Risk Assessment

Table 7 Model Selection Matrix According to Use Cases: Performance, Sustainability, and Operational Efficiency Focused Decision Analysis

Scenario	Model	Accuracy	MCC	Latency p95	CO_2/1 M pred (kg)	Model Size	Reason
Mobile (On-Device)	LR/NB	98.26%/98.36%	0.9219/0.9250	0.31/0.48 ms	0.0032/0.0036	0.44/1.75 MB	Ultra-low latency, minimal footprint, lowest CO_2/1 M among classical models
Real-Time Filtering	NB	98.36%	0.9250	0.48 ms	0.0036	1.75 MB	Fast inference (0.48ms), minimal energy
Edge Computing (IoT)	LR/XG-Boost	98.26%/98.07%	0.9219/0.9121	0.31/0.81 ms	0.0032/0.0053	0.44/0.13 MB	LR: lowest CO_2 and small footprint; XG-Boost: most compact model
Cloud (High-Accuracy)	Distil-BERT	99.13%	0.9603	86.26 ms	0.2816	255.54 MB	Highest accuracy and MCC, no resource constraints
Green AI Priority	LR	98.26%	0.9219	0.31 ms	0.0032	0.44 MB	88x less CO_2 than Distil-BERT
Balanced	GRU	98.65%	0.9378	60.65 ms	0.1942	4.27 MB	High accuracy + acceptable latency and cost

the recommended models and performance-efficiency trade-offs for different deployment scenarios.

Accuracy-Latency-Carbon triangle:

- Classic ML: ~98% accuracy, < 1 ms latency, minimal carbon → Mobile, IoT.
- Ensemble: ~98% accuracy, 0.81–204.18 ms latency, low carbon → Edge computing.

- Deep Learning: ~98–99% accuracy, 44–86 ms latency, medium-high carbon → Cloud.

Example: In a system processing 10 million SMS per day, Naive Bayes emits 0.036 kg CO_2/day, while DistilBERT emits 2.82 kg CO_2/day.

This trade-off in Table 7 clearly demonstrates the necessity of making a conscious model selection based on system requirements. Especially institutions/organizations with "Green AI" goals should not forget that a less than 1% increase in accuracy comes with 79 times more carbon cost [6].

5 Discussion

This study has shown that models should be evaluated as a whole not only based on performance success but also considering computational cost and environmental sustainability, specifically on the "SMS Spam Collection" dataset. Overall, it highlights the sharp distinction between the "Red AI" and "Green AI" paradigms [2, 6] frequently discussed in the current literature, using data to illustrate this difference. This distinction shows that evaluations in many areas where machine learning models are used should be considered as a whole.

5.1 Performance and accuracy perspective

DistilBERT yielded the best results in terms of classification performance; with an MCC of 0.9603 and accuracy of 99.13%. It is noteworthy that this model achieved 100% Precision because no genuine messages were mistakenly marked as spam, which is quite important in terms of user trust [1]. Among deep learning models, GRU showed strong performance with an MCC of 0.9378. Classical models also exceeded expectations; Naive Bayes achieved an MCC of 0.9250, and Logistic Regression achieved an MCC of 0.9219. Ghourabi and Alohaly [1] had previously demonstrated the superiority of Transformer models in the literature, and this study supports the same finding. However, high operational costs make these models less preferable in real-world applications.

5.2 Environmental sustainability and the "Green AI" paradigm

The most striking results in this study come from the environmental sustainability dimension. This dimension should not be confused with computational speed. For stakeholders managing large data centers and cloud infrastructures, and for organizations with strict ESG targets, the real determinants are energy consumption (Wh) and carbon footprint (kg CO_2).

Looking at the numbers, the picture is quite clear. DistilBERT produces 0.001949 kg CO_2 in training; this is 974 times more than Naive Bayes. In inference, it is 0.000291 kg, or 73 times more. In total, DistilBERT produces 0.002240 kg while Naive Bayes produces only 0.000006 kg. The difference is 373 times. Alzoubi and Mishra [7] say that the environmental costs in AI projects are mostly accumulating

unnoticed. Increasing carbon emissions hundreds of times for only a 3.53% increase from 0.9250 to 0.9603 in MCC is not sustainable for continuously operating systems. Naive Bayes and XGBoost stand out as "Green AI" standards for stakeholders who care about ecological impact [6].

5.3 Operational efficiency and edge/mobile applicability

Unlike environmental emissions, operational efficiency focuses on instantaneous system resource consumption. Specifically, p95 inference latency, RAM usage, and model size (MB) are evaluated within this scope. These metrics are critical for a different stakeholder group: mobile end-users, IoT developers, and edge computing architects operating under severe hardware constraints.

Model size and latency create critical bottlenecks in deployment to low-configuration mobile devices [27]. DistilBERT, with its size of 255.5 MB and p95 latency of 86.26 ms, is completely unsuitable for such environments. XGBoost, on the other hand, offers an extremely small size of only 0.13 MB. Logistic Regression stands out for real-time SMS filtering with a p95 latency of 0.31 ms. The independent mobile filtering approach proposed by Nuruzzaman et al. [27] achieved the highest technical feasibility in this study through classical machine learning and ensemble models.

5.4 Statistical stability and generalization ability

The models' resistance to real-world data variations has been measured by the generalization gap. It has been determined that the Random Forest model is the most stable model with a generalization gap of **0.0048**. This finding suggests that ensemble learning approaches may yield more robust results across different SMS languages and user habits [44]. Deep learning models (DistilBERT, GRU, LSTM, CNN), despite requiring high RAM usage, have not demonstrated a significant advantage over classical models in terms of generalization.

5.5 Synthesis and practitioner recommendations

In conclusion, the design of SMS spam detection systems should avoid the notion of a "one-size-fits-all best model." The following threefold strategy is recommended to practitioners:

- **Maximum Accuracy Priority: DistilBERT** should be preferred for server-side systems without energy and resource constraints.
- **Mobile/Edge Deployment Priority**: In applications with high RAM and latency sensitivity, **Logistic Regression (MCC: 0.9219)**, **Naive Bayes (MCC: 0.9250)**, or **XGBoost (MCC: 0.9121)** is the most rational solution.
- **Sustainability Priority**: For institutions and organizations aiming to minimize their environmental carbon footprint (Green AI), **Logistic Regression** or **Naive Bayes** is the suitable choice.

This evaluation proposal will enable practitioners to make informed decisions by complementing the missing efficiency and sustainability dimensions in the literature, which has focused solely on accuracy-oriented research [17, 22].

6 Conclusion and recommendations

This study has holistically evaluated the dimensions of performance, efficiency, and sustainability across 10 different models in SMS spam detection. While DistilBERT exhibited the highest performance with 99.13% accuracy, Logistic Regression was the optimal choice for environmental sustainability with 88 times lower inference carbon emissions per million predictions than DistilBERT. The findings emphasize the importance of choosing a "domain-specific model" instead of the "best model" (Sect. 5.5).

6.1 Limitations

This study has some limitations, which offer specific avenues for future robustness testing:

- Only a single dataset (SMS Spam Collection) and English texts were analyzed. To better test the linguistic robustness of the models, evaluation on multilingual datasets and different messaging platforms (such as WhatsApp, Telegram) is necessary.
- In addition, robustness testing against adversarial spam attacks designed to evade detection (such as intentional character manipulations, "Charswap" techniques) was not performed in this study.
- CodeCarbon measurements reflect the hardware configuration used in this study. Absolute energy values may vary in different computational infrastructures, but the relative efficiency differences between the models will remain consistent.

The main aim of this study is not to solve the SMS spam problem, but to propose a multidimensional evaluation framework in the categories of performance, efficiency, and sustainability. Therefore, these limitations have been applied consciously.

6.2 Suggestions for future studies

SMS and email spam detection is undergoing dynamic change due to the constantly evolving techniques of spammers and the increasing computational costs of artificial intelligence models. Based on the results obtained from this study and current trends in the literature, four main areas of development have been identified for future research.

6.2.1 Model architecture and hyperparameter optimization

DistilBERT gave the best MCC value in this study. But relying solely on manual hyperparameter tuning is not enough for Green AI. As accuracy increases in deep learning, computational load and energy consumption also increase. This is a serious problem in intensive systems such as SMS spam filtering.

Future studies should focus on meta-heuristic methods that can improve model performance and computational efficiency simultaneously. Recent research on nature-inspired algorithms [35–38] has shown that they can significantly reduce the training load. Adding such lightweight methods to the hyperparameter tuning of CNN + GRU hybrid structures both improves performance and conserves energy [11].

6.2.2 Green AI and model compression techniques

In this study, DistilBERT's development time, 1720 times longer than Naive Bayes, is a concrete example of the costs of "Red AI". To reduce these costs, focus should be placed on model quantization and pruning techniques [7]. Knowledge Distillation methods can reduce model size without sacrificing performance, thus enabling high-accuracy models to run on low-spec mobile devices [2, 7]. In terms of energy consumption, "Green Algorithms" are of strategic importance. Using a sparse attention mechanism in Transformers or implementing Low-Rank Adaptations (LoRA) in training can be considered in this category. Adopting these techniques in future deployment processes can significantly reduce FLOPs and energy usage while maintaining prediction accuracy.

6.2.3 Data diversity and multilingual support

This study is based primarily on an English dataset. However, SMS language can also include regional dialects, abbreviations, and slang expressions [25]. Future studies should develop multilingual models that include agglutinative languages such as Turkish. Datasets should be expanded to include image-based spam or voice message analyses [1, 11]. The SMS spam axis can be expanded to include internet-based instant messaging applications. Federated Learning approaches, where data is trained without being collected on a central server to protect user privacy, are emerging as a new research topic in the field of cybersecurity [7, 25].

6.2.4 Real-time and adaptive systems

Future research should aim at real-time updating of models (incremental learning) against constantly changing spam tactics [44]. High-performance models can be run on limited hardware using edge AI tools such as TensorFlow Lite/ONNX and model quantization (INT8/FP16) [2, 7].

Defense mechanisms should be developed by performing robustness tests on adversarial attack scenarios against spammers' attempts to manipulate models [18]. These approaches will increase the reliability and long-term sustainability of SMS spam detection systems.

Author contributions M.A. designed the study, prepared the dataset, conducted the experiments and analyses, and wrote the manuscript.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). Not applicable.

Data availability The dataset used in this study is publicly available. The source code, including data preprocessing, model training procedures, and evaluation scripts, is available at: https://github.com/mustafaaksu46/sustainable-spam-detection.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical approval Not applicable.

Consent for publication Not applicable.

References

1. Ghourabi A, Alohaly M (2023) Enhancing spam message classification and detection using transformer-based embedding and ensemble learning. Sensors 23(8):3861. https://doi.org/10.3390/s23083861
2. Barbierato E, Gatti A (2024) Toward green AI: A methodological survey of the scientific literature. IEEE Access 12:3360705. https://doi.org/10.1109/ACCESS.2024.3360705
3. Strubell E, Ganesh A, McCallum A (2019) Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019)* (pp. 3645–3650). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1355
4. Hao K (2019), June 6 Training a single AI model can emit as much carbon as five cars in their lifetimes. *MIT Technology Review*. https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/
5. Nishant R, Kennedy M, Corbett J (2020) Artificial intelligence for sustainability: Challenges, opportunities, and a research agenda. Int J Inf Manag 53:102104. https://doi.org/10.1016/j.ijinfomgt.2020.102104
6. Schwartz R, Dodge J, Smith NA, Etzioni O (2020) Green AI. Commun ACM 63(12):54–63. https://doi.org/10.1145/3381831
7. Alzoubi YI, Mishra A (2024) Green artificial intelligence initiatives: Potentials and challenges. J Clean Prod 468:143090. https://doi.org/10.1016/j.jclepro.2024.143090
8. Saeed VA (2023) A method for SMS spam message detection using machine learning. Artif Intell Rob Dev J 3(1):214–228. https://doi.org/10.52098/airdj.202366
9. Uysal AK, Gunal S, Ergin S, Sora Gunal E (2013) The impact of feature extraction and selection on SMS spam filtering. Elektronika ir Elektrotechnika 19(5):67–72. https://doi.org/10.5755/j01.eee.19.5.1829

10. Abayomi-Alli O, Misra S, Abayomi-Alli A (2022) A deep learning method for automatic SMS spam classification: Performance of learning algorithms on indigenous dataset. Concurrency Computation: Pract Experience 34(17):e6989. https://doi.org/10.1002/cpe.6989
11. Altunay HC, Albayrak Z (2024) SMS spam detection system based on deep learning architectures for Turkish and English messages. Appl Sci 14(24):11804. https://doi.org/10.3390/app142411804
12. Ghourabi A, Mahmood MA, Alzubi QM (2020) A hybrid CNN-LSTM model for SMS spam detection in Arabic and English messages. Future Internet 12(9):156. https://doi.org/10.3390/fi12090156
13. Zawaideh F, Bsoul Q, Alzoubi A, Botros NT, Fawzy MT, AbdElminaam DS, Mostafa N (2025) Optimized machine learning framework for SMS spam detection and classification: A comparative evaluation. Fusion: Pract Appl 18(1):145–182. https://doi.org/10.54216/FPA.180112
14. ITU (International Telecommunication Union) (2025) *Facts and Figs. 2025*. https://www.itu.int/itu-d/reports/statistics/2025/10/15/ff25-subscriptions/
15. Texting.io (2025) Synthesis, characterization and thermal decomposition of new complexes of *p*-methyl-, *p*-trifluoromethyl- and *p*-bromo-phenylalanine with copper. J Therm Anal Calorim. https://doi.org/10.1007/s10973-005-0589-6
16. Maqsood U, Rehman SU, Ali T, Mahmood K, Alsaedi T, Kundi M (2023) An intelligent framework based on deep learning for SMS and e-mail spam detection. Applied Computational Intelligence and Soft Computing 2023:6648970. https://doi.org/10.1155/2023/6648970
17. Pudasaini S, Shakya A, Pandey SP, Paudel P, Ghimire S, Ale P (2024) SMS spam detection using relevance vector machine. Procedia Comput Sci 230:337–346. https://doi.org/10.1016/j.procs.2023.12.089
18. Salman M, Ikram M, Kaafar MA (2024) Investigating evasive techniques in SMS spam filtering: A comparative analysis of machine learning models. IEEE Access 12:23812–23832. https://doi.org/10.1109/ACCESS.2024.3364671
19. Al Saidat MR, Yerima SY, Shaalan K (2024) Arabic SMS spam detection using CNN and Bi-LSTM hybrid model. Procedia Comput Sci 244:260–267. https://doi.org/10.1016/j.procs.2024.10.199
20. Al-Kaabi HA, Darroudi AD, Jasim AK (2024) Survey of SMS spam detection techniques: A taxonomy. AlKadhim J Comput Sci 2(4):23–34. https://doi.org/10.61710/kjcs.v2i4.88
21. Das L, Ahuja L, Pandey A (2024) A novel deep learning model-based optimization algorithm for text message spam detection. J Supercomputing 80:17823–17848. https://doi.org/10.1007/s11227-024-06148-z
22. Roy PK, Singh JP, Banerjee S (2020) Deep learning to filter SMS spam. Future Generation Computer Systems 102:524–533. https://doi.org/10.1016/j.future.2019.09.001
23. Zapata-Cortes O, Arango-Serna MD, Zapata-Cortes JA, Restrepo-Carmona JA (2024) Machine learning models and applications for early detection. Sensors 24(14):4678. https://doi.org/10.3390/s24144678
24. Almeida TA, Hidalgo JMG, Yamakami A (2011) Contributions to the study of SMS spam filtering: New collection and results. In *Proceedings of the 11th ACM Symposium on Document Engineering* (pp. 259–262). https://doi.org/10.1145/2034691.2034742
25. Johari MF, Chiew KL, Hosen AR, Yong KSC, Khan AS, Abbasi IA, Grzonka D (2025) Key insights into recommended SMS spam detection datasets. Sci Rep 15:92223. https://doi.org/10.1038/s41598-025-92223-1
26. Airlangga G (2024) A comparative analysis of deep learning models for SMS spam detection: CNN-LSTM, CNN-GRU, and ResNet approaches. J Comput Networks Archit High Perform Comput 6(4):1942–1951. https://doi.org/10.47709/cnahpc.v6i4.4827
27. Nuruzzaman MT, Lee C, bin Abdullah MFA, Choi D (2012) Simple SMS spam filtering on independent mobile phone. Secur Commun Netw 5(2):120–129. https://doi.org/10.1002/sec.577
28. Şanlıalp İ, Öztürk MM, Yiğit T (2022) Energy efficiency analysis of code refactoring techniques for green and sustainable software in portable devices. Electronics 11(3):442. https://doi.org/10.3390/electronics11030442
29. Abid MA, Ullah S, Siddique MA, Mushtaq MF, Aljedaani W, Rustam F (2022) Spam SMS filtering based on text features and supervised machine learning techniques. Multimedia Tools Appl 81(28):39853–39871. https://doi.org/10.1007/s11042-022-12991-0
30. Kumar S, Gupta S (2024) Legitimate and spam SMS classification employing novel ensemble feature selection algorithm. Multimedia Tools Appl 83:19897–19927. https://doi.org/10.1007/s11042-023-16327-4
31. Sjarif NNA, Azmi NFM, Chuprat S, Sarkan HM, Yahya Y, Sam SM (2019) SMS spam message detection using term frequency–inverse document frequency and random forest algorithm. Procedia Comput Sci 161:509–515. https://doi.org/10.1016/j.procs.2019.11.150

32. Fatima R, Fareed MMS, Ullah S, Ahmad G, Mahmood S (2024) An optimized approach for detection and classification of spam emails using ensemble methods. Wireless Pers Commun 139:347–373. https://doi.org/10.1007/s11277-024-11628-9
33. Al-Zebari A, Barwary M, Omar N, Zebari NA, Zebari DA (2025) Deep learning hybrid approach for accurate SMS spam identification. J Inform Syst Eng Manage 10(10s):619–635. https://doi.org/10.52783/jisem.v10i10s.1426
34. Gibson S, Issac B, Zhang L, Jacob SM (2020) Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms. IEEE Access 8:187914–187932. https://doi.org/10.1109/ACCESS.2020.3030751
35. Alhussan AA, El-Kenawy E-SM, Eid MM, Khodadadi N (2026) Hybrid Al-Biruni and Puma Optimization (BERPO) for boosting the classification of Quality-of-Service (QoS) in 5G networks. J Netw Comput Appl 250:104462. https://doi.org/10.1016/j.jnca.2026.104462
36. El-Kenawy E-SM, Khodadadi N, Mirjalili S, Zaki AM, Ibrahim A, Alhussan AA, Khafaga DS, Eid MM (2026) Glider snake optimizer (GSO): A nature-inspired metaheuristic algorithm for global and engineering optimization problems. Artif Intell Rev 59:91. https://doi.org/10.1007/s10462-026-11504-x
37. El-Kenawy ESM, Ibrahim A, Alhussan AA et al (2026) Smart City Electricity Load Forecasting Using Greylag Goose Optimization-Enhanced Time Series Analysis. Arab J Sci Eng 51:8359–8377. https://doi.org/10.1007/s13369-025-10647-3
38. Radwan M, Ibrahim A, Abdelsalam MM, Alhussan AA, Mattar EA, El-Kenawy E-SM (2026) Optimizing solar and wind forecasting with iHow optimization algorithm and multi-scale attention networks. Sci Rep 16(1):8597. https://doi.org/10.1038/s41598-026-39632-y
39. Liu X, Lu H, Nayak A (2021) A spam transformer model for SMS spam detection. IEEE Access 9:80253–80263. https://doi.org/10.1109/ACCESS.2021.3081479
40. Paula E, Soni J, Upadhyay H, Lagos L (2025) Comparative analysis of model compression techniques for achieving carbon efficient AI. Sci Rep 15:23461. https://doi.org/10.1038/s41598-025-07821-w
41. Savci P, Das B (2023) Comparison of pre-trained language models in terms of carbon emissions, time and accuracy in multi-label text classification using AutoML. Heliyon 9(4):e15670. https://doi.org/10.1016/j.heliyon.2023.e15670
42. Yokoyama AM, Ferro M, de Paula FB, Vieira VG, Schulze B (2023) Investigating hardware and software aspects in the energy consumption of machine learning: A green AI-centric analysis. Concurrency Computation: Pract Experience 35(17):e7825. https://doi.org/10.1002/cpe.7825
43. Dharani V, Hegde D, Mohana (2023) Spam SMS (or) email detection and classification using machine learning. In *Proceedings of the 5th International Conference on Smart Systems and Inventive Technology (ICSSIT 2023)* (pp. 1104–1108). IEEE. https://doi.org/10.1109/ICSSIT55814.2023.10060908
44. Zhang C, Jia D, Wang L, Wang W, Liu F, Yang A (2022) Comparative research on network intrusion detection methods based on machine learning. Computers Secur 121:102861. https://doi.org/10.1016/j.cose.2022.102861
45. International Energy Agency (2024) *Emissions factors 2024*. IEA. https://www.iea.org/data-and-statistics/data-product/emissions-factors-2024

Authors and Affiliations

Mustafa Aksu[1]

✉ Mustafa Aksu
mustafa.aksu@ahievran.edu.tr

[1] Faculty of Engineering and Architecture, Department of Computer Engineering, Kırşehir Ahi Evran University, Kırşehir, Turkey