Email Scanner – A Machine Learning Approach to Staffing/Non-Staffing Automatic Classification

Sandra M. Aleixo[1,2(✉)], Cátia Dias[1], Artur Ferreira[1,3], Mariana Lopes[1], Inês Prazeres[4], and Ana Silveira[4]

[1] Department of Mathematics and Department of Computer Science and Engineering, ISEL – Instituto Superior de Engenharia de Lisboa, IPL – Instituto Politécnico de Lisboa, Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisbon, Portugal
{sandra.aleixo,catia.dias,artur.ferreira}@isel.pt, A49804@alunos.isel.pt
[2] CEAUL – Centro de Estatística e Aplicações da Universidade de Lisboa, Faculdade de Ciências da Universidade de Lisboa, Bloco C6, Piso 4, 1749-016 Lisbon, Portugal
[3] IDMEC, IST – Instituto Superior Técnico, Avenida Rovisco Pais, 1049-001 Lisbon, Portugal
[4] Independent Researcher, Lisbon, Portugal

Abstract. A company operates on a consulting basis, and one of the key tasks of the staffing teams is to manage project requests from project managers. This process has three stages: receiving, registering, and managing requests. Reception and registration rely on information sent by the company's Project Managers via email, describing client needs for specific roles. This communication is not standardized and often lacks project details. There is the need to process a vast set of incoming emails, classifying them as staffing if their contents is related to project-request emails, and as non-staffing otherwise. Currently, this classification is made through an empirical rule-based approach with unsatisfactory results. In this paper, we propose a solution to improve the tool for automatic data collection, performing email classification as staffing or non-staffing. Our approach enhances the previous system by improving classification performance using Machine Learning (ML) models while expanding the solution's functional scope. Since data vectorization is required to use ML techniques, several methods for this purpose are evaluated, selecting the most suitable one. Along with the best vectorization, some classifiers are trained and evaluated using appropriate metrics. For the best performing classifiers, their parameters are fine tuned, and the resulting metrics are analyzed, showing that ML models significantly improve the tool's classification performance. This work falls within the scope of industrial mathematics, as it solves a real-world problem through the joint work and collaboration between academia and industry.

Keywords: Artificial Intelligence · Big Data · Classification · Email Analysis · Industrial Mathematics · Machine Learning · Text Classification · Text Vectorization

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16759, pp. 144–162, 2027.
https://doi.org/10.1007/978-3-032-30518-3_10

1 Introduction

A leading multinational company has activity in the fields of management, consulting, information technology, and outsourcing services. The company operates globally in the development and implementation of innovative solutions combining strategy, technology, and digital transformation, supporting organizations across a wide range of industries. The company is on Fortune Global 500 ranking, being present worldwide, employing thousands of people. It partners with many of the world's largest organizations, delivering solutions in artificial intelligence, cloud computing, automation, data analytics, and cybersecurity. It is committed to innovation, sustainability, and inclusion.

In their daily operation, the company needs to deal with many staffing requests. These are sent by email by project managers and describe client needs in terms of specific roles, technical profiles, seniority levels, geographic location, and expected allocation duration. Traditionally, this process involved a substantial manual component, namely reading emails individually, identifying relevant information, and recording it in Excel files with the request date and the requester identifier. The high volume of incoming messages, combined with the variability of email content, makes this procedure time-consuming, prone to human error, and inefficient. This process is essential for planning and allocating human resources to ongoing technology projects.

In the literature, one may find some approaches to the email screening problem, with ML techniques. The EmFore solution [23] is an online learning system developed by Microsoft for automatically learning email folder interpretable symbolic classification rules. The system learns incrementally from user actions, updating rules in real time as new emails arrive. It introduces mechanisms for rule consistency, confidence-based suppression, and fast adaptation. Overall, it enables scalable and transparent email organization for modern email clients. In [9], we have an approach to classify customer support emails, from a telecom company, with word embeddings using the Support Vector Machines (SVM), Naïve Bayes, and Long-Short Term Memory (LSTM) networks. LSTM is found to achieve the best performance. Recently, an email sorting system based on DistilBERT, a lightweight transformer model was proposed [17]. It classifies emails into multiple categories and extracts key information such as deadlines and urgency. The system integrates with Gmail and prioritizes messages using context-aware Natural Language Processing (NLP) techniques.

The use of instruction-following Large Language Models (LLM) for enterprise email classification is approached by [22]. It addresses challenges such as class imbalance, domain-specific language, and limited labeled data. The proposed solution combines instruction-based classification with semantic enrichment techniques. The experimental results show that LLM provide robust contextual understanding across diverse email structures. They also improve handling of previously unseen technical terms and evolving vocabulary.

In [20], the authors address zero-shot spam email classification using pretrained LLM. It evaluates different models on the SpamAssassin dataset, with raw email content and with LLM-generated summaries. Results show strong

performance, with GPT-4 reaching 95% acceptable results without fine-tuning, at the expense of high computational and operational costs.

A comparison between Machine Learning (ML), Deep Learning (DL), and LLM for email classification is reported in [25]. Traditional ML models such as SVM, NB, SVM, and Random Forest (RF) [4] are found efficient requiring fewer resources. The DL models achieve higher accuracy by capturing complex patterns in email data. LLM models further improve performance by understanding context and semantics. The study concludes that hybrid approaches combining ML, DL, and LLM offer the best overall performance. For a recent survey on this topic, please see [1].

The previous email staffing requests classification method used by the company showed insufficient performance for real-world use, with a high rate of incorrect classifications and limited robustness to the linguistic, structural, and semantic diversity of the emails received daily. This limitation compromised the overall reliability of the tool, since the correct identification of staffing emails is a fundamental step in process automation. Incorrect message classification frequently required manual intervention, undermining the initial objective of reducing human effort and increasing operational efficiency. It became evident that the adopted approach to email classification needed to be reformulated by developing a more robust and effective classifier capable of reliably and consistently identifying emails related to staffing requests from those that fall outside this scope.

In this paper, we propose an approach for the unit responsible for the intake, registration, and management of staffing requests, within the company's technology department. A tool based on Artificial Intelligence (AI) [21] and Machine Learning (ML) techniques [2] techniques was developed to partially automate this process. The solution uses AI, namely Large Language Models (LLM) [6] to extract information from staffing emails, which was then stored into a database. This solution enables access to the mailbox, message reading, and extraction of some relevant data, which are then recorded in a database, representing a first step toward information centralization and historical request management. The proposed solution improves the existing one for the binary classification of emails into staffing and non-staffing categories.

The remainder of the paper is organized as follows. Section 2 summarizes the main methodological concepts outlining the materials and methods, considered in our work. Section 3 presents the detailed case study and the experimental evaluation with standard metrics. Section 4 concludes the paper, with the main findings, practical implications, and directions for future work.

2 Methodology

This section presents the methodology adopted for developing the automatic email classification system. It describes the main technical choices considered throughout the work, from text vector representations to the ML classification models employed, as well as the validation and optimization strategies adopted,

and concludes with the evaluation metrics to assess model performance. This structure provides a systematic rationale for the methodological choices that underpin the experimental analysis presented in the following section.

2.1 Vectorization

Text vectorization methods represent textual documents with numerical-valued vectors, suitable for the use of ML algorithms. For automatic text classification and information retrieval, the choice of an adequate vectorization method is a fundamental step, as it determines how textual information is presented to ML models. In the following, we briefly review some of these common representations [15,16].

The Bag-of-Words (BoW) model is one of the simplest techniques. A document is represented as a collection of words or terms, ignoring the order in which they occur while preserving their frequency. A vocabulary containing all distinct terms found in the corpus is then built. Each document is represented by a numerical vector holding the frequency of occurrence of each vocabulary term, yielding a fixed-dimensional vector representation. The conceptual simplicity of Bag-of-Words, combined with ease of implementation and computational efficiency, makes it suitable for applications with large volumes of text. Thus, it is widely used for document classification, clustering, and information retrieval. However, this simplicity entails important shortcomings. By ignoring word order and syntactic structure, the model loses relevant semantic information, being unable to capture relationships between words or distinguish contextual meanings. Moreover, synonyms are treated as distinct terms and lexical ambiguity is not addressed, limiting model performance for some tasks.

Consider a collection $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ of n documents and a vocabulary $V = \{t_1, t_2, \ldots, t_p\}$ with p distinct terms. Let $f(t, d)$ denote the number of occurrences of term t in document d. Each document d is represented by a vector $\mathbf{x}_d = (x_{d,1}, \ldots, x_{d,p}) \in \mathbb{R}^p$, where each component $x_{d,i}$ corresponds to the frequency of term t_i in document d, $x_{d,i} = f(t_i, d)$.

The Term Frequency–Inverse Document Frequency (TF-IDF) is one of the most widely used methods for vector representations of text in automatic classification tasks. It is simple, computationally efficient, and interpretable, and is frequently used as a baseline in text classification systems. However, like the BoW model, it ignores word order and context and does not capture semantic relationships between terms. It is a term-weighting scheme whose objective is to quantify the importance of each term in a document while accounting for both its local relevance (within the document) and its global relevance (across the document collection). The underlying idea of TF–IDF is that a term is informative if it occurs frequently in a given document but does not appear ubiquitously across the corpus. Thus, very common words tend to receive lower weights, whereas more specific and discriminative terms become more relevant in the vector representation.

Let $f(t, d)$ denote the number of occurrences of term t in document d, and let $df(t)$ denote the document frequency of term t, i.e., the number of documents

in $\mathcal{D}$ that contain t at least once. The final weight of term t in document d (TF-IDF) is given by the product of the Term Frequency (TF) and Inverse Document Frequency (IDF) components and is defined as $\text{tfidf}(t, d) = \text{tf}(t, d) \cdot \text{idf}(t)$. The TF component measures the importance of the term within the document. A common definition is the relative frequency $\text{tf}(t, d) = f(t, d)/p$. The IDF component penalizes terms that appear in many documents in the corpus, since such terms tend to have lower discriminative power, and is commonly defined as $\text{idf}(t) = \log(n/df(t))$. Thus, each document is represented by a TF-IDF vector $\mathbf{x}_d = (x_{d,1}, \ldots, x_{d,p}) \in \mathbb{R}^p$, where $x_{d,i} = \text{tfidf}(t_i, d)$. This representation is typically sparse, since only a small fraction of the vocabulary terms occurs in each document; the same usually happens with the BoW representation.

To reduce the influence of document length and enable more consistent comparisons between documents, it is common practice to normalize the vector (typically using the Euclidean norm). One direct application of this vector representation is the computation of similarity between documents, for which cosine similarity is the most common measure. In detail, it evaluates the cosine of the angle between the document vectors, quantifying their similarity in terms of content. Regardless of vector magnitude, cosine similarity enables comparison of documents with different lengths, assigning higher values to semantically closer documents and values near zero to weakly related documents.

Word embedding methods provide a recent approach to text vector representations, developed to overcome the main limitations of the previous models. Unlike them, embeddings aim to capture semantic and contextual information by representing each word with a vector. The central idea is that words with similar meanings tend to occur in similar contexts. By analyzing large volumes of text, these models learn vector representations in which geometric proximity reflects semantic and syntactic relationships between words. For example, synonymous or semantically related words are represented by vectors that are close in the embedding space. Despite their greater expressive power, word embeddings still have some limitations in text classification. Simple aggregation of word vectors may lead to loss of structural information and of the relative importance of terms. They are also more complex to train, requiring greater computational resources and, in some cases, large volumes of data to produce high-quality representations.

Each word (term) t is represented by a vector $\mathbf{v}_t \in \mathbb{R}^q$, with $q \ll p$, whose components capture semantic relationships between words. Once word vector representations are obtained, a document representation d can be constructed by aggregating the vectors of the words that compose it. A simple approach is to compute the mean of the word vectors in d, $\mathbf{x}_d = \frac{1}{p} \sum_{i=1}^{p} f(t_i, d)\mathbf{v}_{t_i}$. More details about text vectorization can be found in [15,16].

2.2 Machine-Learning Classifiers

In the context of this work, the problem under study falls within automatic binary text classification being similar to classic tasks such as email SPAM filtering. The objective is to automatically assign a category to each request based

on the content of the message. To this end, Natural Language Processing (NLP) models represent the text in a structured manner and to apply ML techniques. These techniques can be either supervised or unsupervised, depending on the availability of labeled data [2].

In this paper, a supervised learning approach is adopted, since a set of previously human labeled requests is available. The model is trained on these examples, to enable the automatic classification of new requests.

In summary, the main stages of the process adopted in this work for the automatic classification of requests are presented in Fig. 1. Based on this process, different supervised classification models (Näive Bayes, KNN, Logistic Regression, Decision Trees, Support Vector Machines, and Ensemble Methods), learned from several vectorization techniques, were considered in order to compare their performance and select the most effective combination.

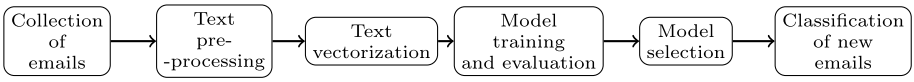

Fig. 1. Stages of the process for the automatic classification of requests.

The system thus follows a model-based approach, in which a classifier is trained on historical data and the resulting model is used to produce predictions for new incoming messages. This approach is appropriate for the problem under study, given that labeled data are available and the objective is the automatic classification of new requests [3,10,15,26].

Naive Bayes (NB) is a probabilistic classifier based on Bayes' theorem [26], and it produces predictions by computing simple and conditional probabilities. This classifier assumes that all predictor variables are conditionally independent of one another given the class label. This hypothesis implies that each variable contributes independently and with equal importance to the final outcome. Bayesian classification aims to estimate the probability that an instance is assigned to a class L (e.g., the category of a text), given a set of features. A decision between two possible classes can be made by computing the ratio between the conditional probabilities associated with each class.

The term *naive* stems from the strong independence assumption between features. Variants of this model include Bernoulli Näive Bayes, Multinomial Näive Bayes, and Complement Näive Bayes. Bernoulli Näive Bayes is particularly suitable for datasets with binary features. In this model, each attribute indicates only the presence or absence of a given feature, taking values in $\{0, 1\}$. Multinomial Näive Bayes is particularly suitable for datasets whose features represent counts or frequencies. In this model, each attribute takes non-negative integer values, corresponding, for example, to the number of occurrences of a word in a document. Complement Näive Bayes is a variant of Multinomial Näive Bayes in which the conditional probabilities of the attributes are estimated from the empirical distribution of the remaining classes, yielding more robust weights, especially

in settings with class imbalance and high dimensionality. Overall, despite their simplicity, Näive Bayes classifiers are often highly efficient, achieving satisfactory levels of accuracy and performance across a variety of classification tasks, particularly in text processing contexts, where they may even outperform more complex classifiers.

The K-Nearest Neighbours (KNN) classifier [26] assumes that similarity between instances is directly related to their proximity. Similar instances (labeled with the class) tend to be close in the feature space, in terms of some distance metric, which is typically computed using the Euclidean norm. The parameter K in the KNN algorithm represents the number of nearest neighbours considered in the decision process. Choosing this parameter is critical, as it defines the region of influence used to classify or estimate a new point. The optimal value of K can be determined empirically or through standard methods such as k-fold cross-validation. The procedure then proceeds with distance computation, identification of the nearest neighbours, and voting for the final class. One of the main advantages of KNN is its simplicity. There is no explicit training phase and no need to tune complex parameters. However, as the number of instances or features increases, the algorithm becomes computationally slower, which makes it less suitable for scenarios requiring fast labeling responses.

Logistic Regression (LR) [10] models are widely used in text classification tasks due to its simplicity, computational efficiency, and strong performance with high-dimensional vector representations such as Bag-of-Words or TF-IDF. This classifier assumes that the decision boundary can be obtained via a linear combination of features, learning a weight vector that maximizes class separation and directly modeling the probability that an instance belongs to a class through the logistic function.

Decision Trees (DT) are classification models based on a tree-like structure [18,19]. Classification consists of traversing the tree from the root to a leaf, following the branches defined by the conditions evaluated at each node. In this way, each root-to-leaf path corresponds to a decision rule used to classify the data. The leaf nodes hold the class labels. For the DT learning process, a function is used to compute the cost associated with each branch, and the branch with the lowest cost is selected. This classifier is sensitive to small variations in the training data and their presence may yield completely different trees. Moreover, its greedy nature does not always guarantee a globally optimal tree and, in the presence of dominant classes, biased trees may emerge.

Support Vector Machines (SVM) [7,24] are among the most popular and well-established classifiers for ML applications. They are versatile models capable of performing linear and nonlinear classification. The main objective of SVM is to identify, in the multidimensional feature space, a hyperplane that correctly separates the data while maximizing the margin, i.e., the distance between the hyperplane and the closest points from each class, called support vectors. Maximizing the margin enables more confident classification of new data, reducing the probability of classification errors. When the data are not linearly separable, SVM can use kernel functions to project the data into a higher-dimensional

space where linear separation becomes possible. Due to its robustness and strong generalization ability, it is frequently used in high-dimensional problems such as text classification [11].

Ensemble methods [27] combine multiple base models to produce a more robust classifier with improved generalization ability. In text classification tasks, approaches such as Bagging, Random Forest, Extremely Randomized Trees, AdaBoost, and Gradient Boosting build several classifiers (typically DT) and aggregate their predictions via voting or weighted combination [4]. These methods can improve performance as compared to individual models and are particularly useful when the goal is to capture complex patterns in textual data.

Bagging [5] reduces variance by averaging models trained on random subsets of the data. No differentiated weights are assigned to individual classifiers, which contribute equally to the final decision. Random Forest and Extremely Randomized Trees rely on an ensemble of trees. Each tree produces a prediction, and the final class is obtained by aggregating individual predictions, typically via majority voting. The main difference between these methods lies in the degree of randomness introduced during tree construction. While Random Forest randomly selects a subset of features and searches for the best split at each node, Extra Trees introduces additional randomness by selecting split points at random for each input feature. Boosting methods build models sequentially, giving greater weight to misclassified examples. AdaBoost (Adaptive Boosting) [8] iteratively combines weak classifiers by adjusting the weights of misclassified observations and training based on errors attained by previous models. In Gradient Boosting, each new model is trained to correct the errors made by the ensemble built so far, explicitly optimizing a loss function via gradient descent and fitting each new model to the residuals of the previous model.

2.3 Evaluation Metrics for Classifier Performance

Evaluating the performance of a ML model is a fundamental step. Evaluation metrics quantify model effectiveness and enable objective comparison of different approaches. Depending on the nature of the problem, selecting an appropriate metric is crucial for model selection [15,26]. In the following, we describe the key evaluation metrics for classifier performance assessment.

The confusion matrix is a fundamental evaluation tool for classifier performance. It compares true classes with predicted classes and forms the basis for computing several evaluation metrics, such as Accuracy, Precision, Recall, and F_1-Score. In binary classification, the confusion matrix has dimensions 2×2 and consists of four fundamental elements: True Positives (TP) - positive cases correctly classified; True Negatives (TN) - negative cases correctly classified; False Positives (FP) - negative cases incorrectly classified as positive, and False Negatives (FN) - positive cases incorrectly classified as negative. Accuracy measures the proportion of correct predictions made by the model among all predictions,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

Although it provides a quick overall view of model performance, Accuracy can be misleading on imbalanced datasets, as it may convey a false impression of good performance when the misclassification rate for minority classes is high.

Precision assesses the proportion of positive predictions made by the model that are actually correct,

$$\text{Precision} = \frac{TP}{TP + FP}.$$

This metric is particularly relevant in scenarios where the cost associated with false positives is high. Precision helps ensure that, when the model predicts a positive outcome, that prediction has a high probability of being correct.

Recall, also known as Sensitivity, measures the proportion of actual positive cases that are correctly identified by the model,

$$\text{Recall} = \frac{TP}{TP + FN}.$$

This metric is particularly important in scenarios where failing to detect a positive case (a false negative) has a higher cost than a false positive.

The F_1-Score corresponds to the harmonic mean of Precision and Recall,

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}},$$

and is particularly useful when both must be balanced into a single measure. A high F_1-Score indicates that the model performs well simultaneously in terms of both Precision and Recall.

The F_β-Score is a generalization of the F_1-Score that allows adjusting the relative importance of Precision and Recall through a parameter β,

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}},$$

thereby allowing one metric to be prioritized depending on the objectives of the problem under analysis. In particular, when $\beta > 1$ greater emphasis is placed on Recall, resulting in a stronger penalty for false negatives. Conversely, when $0 < \beta < 1$, greater emphasis is placed on Precision, resulting in a stronger penalty for false positives.

The Receiver Operating Characteristic (ROC) curve is a graphical representation used in binary classification assessment problems that shows the relationship between the True Positive Rate (TPR), which coincides with Recall, and the False Positive Rate (FPR) given by $FPR = FP/(FP + TN)$, for different threshold values. This curve allows analysis of the trade-off between the ability to detect positive instances and the number of false alarms. The Area Under the Curve (AUC) metric corresponds to the area under the ROC curve and quantifies the model's overall ability to distinguish between classes. AUC takes values in the range $[0, 1]$, with higher values indicating better performance. More details about these evaluation metrics can be found in [26].

2.4 Classifier Assessment and Fine Tuning

To ensure a model's ability to generalize to unseen data, validation and optimization strategies are adopted to robustly estimate classifier performance and to select suitable hyperparameter configurations. A hyperparameter is defined as a setting chosen before or during training validation that controls how the model learns or its ability, and is not directly estimated from the data as it happens for model parameters [3,26].

Cross-Validation (CV) is a statistical technique widely used in ML to assess the generalization ability of a predictive model. Its main objective is to estimate model performance on unseen data with different random samples from the input datasets (data folds) mitigating overfitting issues associated with evaluations performed on a single training/test split. In classification, overfitting occurs when a classifier achieves very high performance on the training set, but performance drops sharply on the validation/test set, indicating that the model memorised training-specific patterns/noise and was not able to generalize to new examples.

The most common CV method is k-fold cross-validation, in which the dataset is divided into k subsets or folds. In each iteration, the model is trained using $k-1$ folds and evaluated on the remaining fold, which acts as the test set. This process is repeated k times, ensuring that each subset is used exactly once for testing. Final model performance is obtained by averaging the results across all iterations, providing a more stable and robust estimate of performance. In imbalanced-data settings, i.e., with much more instances from one class than the other, stratified k-fold cross-validation is commonly used to preserve class proportions within each fold.

One of the main advantages of cross-validation is its ability to reduce the variance of the model performance estimate by evaluating classifier behaviour across multiple data partitions. However, it increases computational cost, since the model is trained multiple times, which can be limiting for large datasets or complex models. It is frequently used together with hyperparameter tuning procedures such as Grid Search, enabling systematic evaluation of different parameter configurations based on robust validation.

Grid Search is a systematic approach to hyperparameter optimization in which predefined combinations of values for a set of hyperparameters are exhaustively evaluated. The objective is to identify the configuration that maximizes (or minimizes) a selected performance metric, yielding a model with improved generalization ability. Despite its simplicity and interpretability, Grid Search can become computationally expensive, since the total number of trained models grows with the product of the number of values considered for each hyperparameter and the number of folds.

Stochastic Gradient Descent (SGD) is an iterative optimization method widely used in supervised learning models, particularly in settings with large datasets or high dimensionality. Unlike standard Gradient Descent, which computes the gradient of the cost function using the entire training set, SGD updates model parameters progressively using individual examples or small batches, resulting in a significant reduction in computational cost. The Log Loss function,

combined with SGD, evaluates whether the probabilities predicted by the model match the true classes, penalizing incorrect predictions with high probability.

3 Case Study: Staffing/Non-Staffing Classification

The process of classifying emails as Staffing/Non-Staffing is essential for planning and resource management at the company. A staffing request is defined as a formal request for allocating human resources to a given project or set of projects. These requests are typically made by project managers or team leads, who identify the need to reinforce resources to meet operational requirements and deadlines agreed with the client. Each staffing request specifies a set of requirements associated with the desired profile, is usually communicated via email, and exhibits high variability in structure and level of detail, since it does not follow a rigidly standardized format or taxonomy. This heterogeneity makes manual process management even more complex, particularly in high-volume settings, reinforcing the importance of automated mechanisms that can correctly identify, classify, and handle requests.

3.1 Data Description

The data consists of real corporate emails from an internal mailbox, covering communications recorded since 2023. Emails are written in multiple languages, namely Portuguese, English, Spanish, and, to a lesser extent, French, reflecting the linguistic diversity of the organizational context. To increase the dataset size and improve model generalization, additional emails had to be generated using AI techniques, based on existing emails.

The database initially comprised 1132 real emails and was used only to evaluate the existing empirical classifier. It was subsequently augmented to 1672 real emails and used for vectorization experiments and in an initial classification phase, that produced some interesting performance results. To analyze classifier behaviour on a larger and more balanced dataset, company-approved AI-generated emails were incorporated, yielding a total of 3263 emails. Emails were manually labeled into two classes, staffing and non-staffing, and the labeling was validated by the company. The database includes the following attributes/variables: a unique email identifier (id), the email subject (subject), the email body (body), an indicator identifying whether the email is part of a reply chain (email_chain), and the corresponding label (classification). The subject and body variables constitute the primary sources of textual information used by the classification models.

3.2 Data Pre-processing and Preparation

An initial pre-processing phase was performed on the textual data to reduce noise, standardize text representation, and improve the quality of extracted features. Pre-processing included normalization of the text in the subject and body

fields, encompassing conversion of all characters to lowercase, removal of accents, and elimination of redundant whitespace. This reduces vocabulary size and prevents superficial writing variations from being treated as distinct terms by the models. The subject and body fields were concatenated, resulting in a single textual field per email.

3.3 Pre-existing Classification Rule

The existing tool under development, used to automate email classification, incorporated a model based on a set of empirical rules and keywords, without the use of ML techniques. The decision logic relied exclusively on the presence and frequency of predefined keywords, applying deterministic rules to email textual content. In detail, an email was classified by an empirical rule, as *staffing* if the text did not contain the word "férias" (the Portuguese word for holidays) and contained at least a minimum number of keywords.

The performance of this empirical classifier was evaluated on an initial database consisting of 1132 real emails. The obtained predictions were compared with the true labels, yielding TP=140, TN=522, FP=428 and FN=42, hence Accuracy=0.5848. Precision, Recall, and F_1-Score values are reported in Table 1.

Table 1. Performance of the initial rule-based approach.

	Staff	No-Staff
Precision	0.2465	0.9255
Recall	0.7692	0.5495
F_1-Score	0.3733	0.6896

These results indicate limited overall performance. Although the classifier achieves relatively high Recall for the staffing class, its Precision for that class is very low, indicating that while it identifies most staffing emails, it incorrectly classifies a large number of non-staffing emails as staffing. For the non-staffing class, the opposite behaviour is observed: high Precision but moderate Recall, resulting in several emails that in fact belong to this class being missed. Overall, the classifier shows a clear tendency to overclassify emails as staffing.

This classifier depends entirely on a manually defined keyword list. Small linguistic variations, differences in request phrasing, or the introduction of new terms not covered by the rules frequently lead to incorrect classifications. This rigidity makes system maintenance difficult and compromises its ability to generalize to new data. Therefore, adopting a ML-based approach becomes necessary, as it can exploit statistical and semantic patterns present in the texts.

3.4 Vectorization Experiments

To enable the use of ML classifiers, vector representations are required. Accordingly, the methods presented in Sect. 2.1 were evaluated to assess the impact of text representation on the performance of some of the classifiers presented in Sect. 2.2. The selected classifiers were Logistic Regression (LogReg), SVM (LinearSVC), and Random Forest (RandomForest), as they are widely used in text classification tasks and perform well in high-dimensional spaces. The BoW and TF-IDF vectorization methods were tested with and without stopwords (BoW-SW, BoW, TFIDF-SW, and TFIDF). Stopwords are highly frequent words in a language (e.g., articles, prepositions, and conjunctions) that carry limited semantic content and tend to introduce noise while contributing little relevant information for class discrimination. Word embeddings (WEMB) were tested only without stopwords, since pre-trained embeddings already provide vectors that capture context, and the presence of stopwords may be useful because it helps the model capture grammar and relationships between words. For each vectorization method, stratified five-fold cross-validation was applied, allowing identification of the best mean F_1-Score for the staffing class, as reported in Table 2.

Table 2. Mean F_1-Score for the different classifier/vectorization combinations. The best result is in boldface.

	BoW-SW	BoW	TFIDF-SW	TFIDF	WEMB
LogReg	**0.8572**	**0.8530**	0.8321	0.8257	0.7619
LinearSVC	0.8526	0.8398	**0.8390**	**0.8522**	**0.7770**
RandomForest	0.8278	0.8176	0.8101	0.8168	0.7451

For each vectorization method, the classifier with the highest mean F_1-Score was selected, and Grid Search was applied to optimize its hyperparameters. The selected classifier was then trained on the dataset with 1672 emails using a test split consisting of 75% of the emails, obtaining the metrics reported in Table 3.

Table 3. Metrics for the best classifier/vectorization combination. The best result is in boldface.

	Accuracy	Precision		Recall		F_1-Score	
		Staff	No-Staff	Staff	No-Staff	Staff	No-Staff
LogReg + BoW-SW	0.8373	0.8063	0.8634	0.8324	0.8412	0.8191	0.8522
LogReg + BoW	0.8373	0.8063	0.8634	0.8324	0.8412	0.8191	0.8522
LinearSVC + TFIDF-SW	0.8421	0.7960	**0.8848**	**0.8649**	0.8240	0.8290	0.8533
LinearSVC + TFIDF	**0.8541**	**0.8196**	0.8839	0.8595	**0.8498**	**0.8391**	**0.8665**
WEMB	0.7990	0.7306	0.8744	**0.8649**	0.7468	0.7921	0.8056

In this setting, for BoW the stopword removal does not affect the metrics. For Accuracy and F_1-Score, the highest value is obtained with TF-IDF vectorization without stopwords. However, in the context of this work, evaluation should be primarily driven by Recall for the staffing class, since the system's main objective is to minimize missed staffing requests, i.e., to reduce the number of false negatives. Both TF-IDF with stopwords and word embeddings achieve the highest value for this metric. However, since TF-IDF with stopwords yields higher values for the remaining metrics, it was selected as the vectorization method to be used for automatic classification.

3.5 Classifier Selection

After selecting TF-IDF vectorization with stopwords, several classifiers were trained and evaluated on the final dataset, which comprised 3263 emails: 1737 staffing emails and 1526 non-staffing emails, corresponding to 53.23% and 46.77% of the total, respectively. Thus, the dataset is reasonably balanced and suitable for subsequent stages. The dataset was split into training and test subsets using a stratified train–test split strategy, ensuring preservation of class proportions in both subsets. The adopted split used 75% of the data for training and 25% for testing, and it was performed before any numerical transformation of the text in order to avoid data leakage, i.e., to prevent information from the test set from influencing the training process. If vectorization were performed prior to splitting, the statistics used would be computed on the full dataset, including test emails, giving the classifier indirect access to information that should remain unknown during training.

In parallel, an analysis of the vocabulary most characteristic of emails labeled as staffing was conducted based on mean TF-IDF weights computed exclusively for this class. This analysis showed that the highest-weight terms are strongly associated with the context of resource allocation and project management.

The classifiers presented in Sect. 2.2 were considered, as well as a linear classifier optimized with SGD using Log Loss. Evaluation was performed using five-fold cross-validation, preserving the class distribution in each fold. The following metrics were considered for the staffing class: Accuracy, Precision, Recall, F_1-Score, F_2-Score, and AUC, as reported in Table 4.

For selecting the classifier to be used in the tool, the F_2-Score metric was defined as the priority, since it emphasizes Recall for the staffing class. For the company, it is more important that a staffing email is not classified as non-staffing rather than a non-staffing email is classified as staffing (no false negatives). The results indicate that the SVM classifier achieves the best overall performance, reflecting a strong ability to detect staffing emails, combined with a balanced trade-off with Precision. However, the SGD classifier with Log Loss achieves a very similar value, with slightly higher Accuracy and AUC than SVM.

Based on this analysis, the hyperparameters of both classifiers were optimized using Grid Search to obtain the final evaluation. After applying Grid Search, the F_2-Score values were 0.9315 for SVM and 0.9244 for SGD with Log Loss. The improvement achieved by SVM is larger, increasing the gap between the

Table 4. Mean metrics for the classifiers. The best result is in boldface.

Classifier	Accuracy	Precision	Recall	F_1-Score	F_2-Score	AUC
BernoulliNB	0.8789	0.8759	0.9035	0.8894	0.8978	0.9504
MultinomialNB	0.8896	0.8932	0.9021	0.8976	0.9002	0.9644
KNN	0.8799	0.9084	0.8589	0.8827	0.8681	0.9493
LogReg	0.8730	0.8628	0.9115	0.8863	0.9011	0.9476
DecisionTree	0.7824	0.7668	0.8653	0.8127	0.8434	0.8126
SVM	0.9104	0.9080	0.9280	0.9177	**0.9238**	0.9707
Bagging	0.8792	0.8875	0.8870	0.8870	0.8869	0.9538
RandomForest	0.8774	0.8675	0.9136	0.8899	0.9039	0.9622
ExtRandTrees	0.8874	0.8801	0.9172	0.8981	0.9094	0.9667
AdaBoost	0.8327	0.8074	0.9143	0.8573	0.8905	0.9241
GradientBoosting	0.8490	0.8184	**0.9338**	0.8722	0.9081	0.9434
SGDLogLoss	**0.9116**	**0.9105**	0.9273	**0.9187**	**0.9238**	**0.9744**

two models. An increase in AUC to 0.9740 is also observed, indicating excellent separability between the staffing and non-staffing classes. The ROC curve shape, depicted in Fig. 2, positioned mostly above the diagonal, further supports the conclusion that the model performs substantially better than chance.

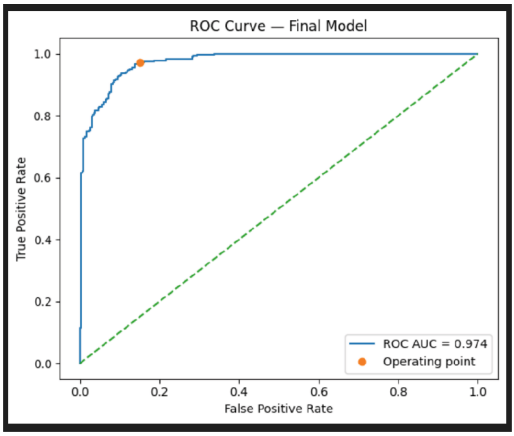

Fig. 2. ROC Curve and AUC.

In addition, the classifier achieves an Accuracy of 0.9142, indicating strong overall classification performance. For the staffing class, Precision and Recall are 0.8802 and 0.9713, respectively, showing effective identification of most true staffing requests, in line with the objective of prioritizing Recall for this class. For

the non-staffing class, Precision and Recall are 0.9628 and 0.8492, respectively. These results confirm the suitability of the classifier for the problem under study, demonstrating consistent discrimination between the two classes.

After selecting the classifier, the decision threshold was further adjusted to prioritize minimizing false negatives for the staffing class. The threshold was selected by maximizing the F_2-Score on a validation set (a subset of the training set) to avoid overfitting. On the test set, a Recall of 0.9770 was obtained for the staffing class. Despite a slight reduction in Precision to 0.8416, the F_2-Score increased to 0.9465, confirming that threshold tuning aligned the classifier with our objectives. Using the adjusted threshold, the classifier achieves an Accuracy of 0.9465, indicating strong overall classification performance. For the staffing class, Precision and Recall are 0.8416 and 0.9770, respectively, and for the non-staffing class, Precision and Recall are 0.9679 and 0.7902, respectively.

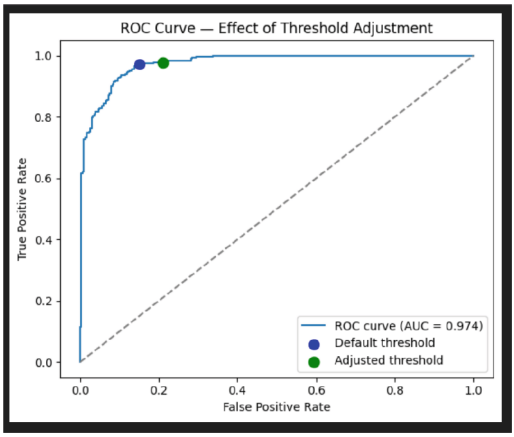

Fig. 3. ROC curve with default and adjusted thresholds.

The ROC curve of the selected classifier in Fig. 3 shows the two operating points corresponding to the default and the adjusted threshold The adjusted threshold shifts the operating point along the curve, increasing TPR at the expense of a slight increase in FPR. This behaviour confirms that threshold optimization allows prioritizing the detection of staffing emails, aligning the model with our objectives without compromising its overall discriminative capability.

4 Conclusions

The company faces a daily challenge within their technology department responsible for managing staffing requests. These requests, sent by project managers via email, include details such as technical profiles, seniority levels, location, and allocation duration. The process is largely manual (emails are read individually

and relevant information is recorded in Excel files). The company faces a big data problem (in both volume and variability) since the current approach is time-consuming, inefficient, and prone to human error. The first attempt of the company to screen out these emails was to devise an empirical classification rule. However, this lead to unsatisfactory results, since it provides a large number of false negatives (true staffing emails wrongly classified as non-staffing).

Our approach aims to provide a new solution for the email staffing/non-staffing classification, grounded on ML techniques. The goal was to improve the classification results for staffing and non-staffing emails, integrated into an AI-based system to support the management of project requests within Staffing teams. The previously developed empirical, rule-based solution was evaluated, leading to the conclusion that improvements could be implemented at the classification level, by devising a different approach from a labeled dataset.

Several vectorization methods were evaluated. TF-IDF with stopwords proved to be the most effective and was subsequently used to train and evaluate different ML classifiers. For the two best-performing models, SVM and SGD with Log Loss, hyperparameter optimization was performed using Grid Search to select the classifier to be integrated into the system. The proposed solution significantly improves the previous results attained by the empirical rule approach.

Overall, this entire study and the experimental evaluation demonstrate that an approach based on Natural Language Processing and Machine Learning techniques enables more robust and consistent email classification, reducing manual intervention and increasing process efficiency. The developed system therefore contributes to more structured information management, facilitating the logging, tracking, and historical analysis of project requests.

As future work, we aim to explore dimensionality reduction with feature selection techniques. The reduction of data dimensionality will identify the relevant, irrelevant, and redundant features of our dataset. With lower dimensionality of the data, we expect to further improve the classification accuracy and further reduce the false negatives, achieving the maximum Recall.

Acknowledgments. This work is funded by national funds through FCT Fundação para a Ciência e a Tecnologia, I.P., under CEAUL Research Unit, UID/00006/2025, DOI: https://doi.org/10.54499/UID/00006/2025, and by the European Union Next GenerationEU through the project UID/PRR/00006/2025, DOI: https://doi.org/10.54 499/UID/PRR/00006/2025 and ISEL. The authors acknowledge also FCT for its financial support via LAETA (project https://doi.org/10.54499/UID/50022/2025).

Disclosure of Interests. The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this article. All analyses were conducted independently, and no external funding sources affected the study design, data collection, or interpretation of results. During the preparation of this manuscript/study, the authors used ChatGPT-5 (OpenAI) for the purposes of improving the clarity and readability of the manuscript. The

authors have reviewed and edited the output and take full responsibility for the content of this publication.

References

1. Alshaikh, M., Alrajeh, Y., Alamri, S., Melhem, S., Abu-Khadrah, A.: Supervised methods of machine learning for email classification: a literature survey. Syst. Sci. Control Eng. **13**(1), 2474450 (2025)
2. Alpaydin, E.: Introduction to Machine Learning, 2nd edn. MIT Press, Cambridge (2010)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Hiedelberg (2009)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
6. Brown, T., et al., : Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
8. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proceedings of the Int. Conference on Machine Learning (ICML), pp. 148–156 (1996)
9. García, V., Sánchez, J.S., Mollineda, R.A.: E-mail classification with machine learning and word embeddings for improved customer support. Neural Comput. Appl. **31**, 3891–3903 (2019)
10. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Heidelberg (2017)
11. Hsu, C., Lin, C.: A comparison of methods for multi-class support vector machines. IEEE Trans. Neural Netw. **13**(2), 415–425 (2002)
12. Jebara, T.: Machine Learning: Discriminative and Generative. Springer, Heidelberg (2004)
13. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers (1998)
14. Mackay, D.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Cambridge (2003)
15. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2009)
16. Patil, R., Boit, S., Gudivada, V., Nandigam, J.: A survey of text representation and embedding techniques in NLP. IEEE Access **11**, 36120–36146 (2023)
17. Priya, S.Y., Reddy, B.I.S., Malleswar, R.U., Kumar, B.V., Jeevana Priya, C.: Smart email sorting: a machine learning approach for enhanced productivity. Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET) (2026). https://doi.org/10.22214/ijraset.2026.78656
18. Quinlan, J.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
19. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers - a survey. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **35**(4), 476–487 (2005)
20. Rojas-Galeano, S.: Zero-shot spam email classification using pre-trained large language models. arXiv preprint arXiv:2405.15936 (2024)
21. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson Education, England (2016)

22. Sarıyıldız, A.Ç., Durukan-Odabaşı, Ş: Enterprise e-mail classification using instruction-following large language models. Appl. Sci. **16**(5), 2173 (2026)
23. Singh, M., Cambronero, J., Gulwani, S., Le, V., Verbruggen, G.: EmFore: online learning of email folder classification rules. In: Proceedings of Int. Conference on Information and Knowledge Management (CIKM 2023), pp. 2280–2290 (2023)
24. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
25. Wang, X.: Emails classification: comparing machine learning, deep learning and large language models. In: Proceedings of the International Conference on Data Analysis and Machine Learning (DAML 2024), pp. 451–458 (2024)
26. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., Foulds, J.: Data mining : practical machine learning tools and techniques. 5th Edition, Morgan Kaufmann (2025)
27. Zhang, C., Ma, Y.: Ensemble Machine Learning: Methods and Applications. Springer, Heidelberg (2012)