

Article

AI-Based Classification of IT Support Requests in Enterprise Service Management Systems

Audrius Razma  and Robertas Jurkus * 

Faculty of Marine Technology and Natural Sciences, Klaipeda University, 92294 Klaipeda, Lithuania

* Correspondence: robertas.jurkus@ku.lt

Abstract

In modern organizations, IT Service Management (ITSM) relies on the efficient handling of large volumes of unstructured textual data, such as support tickets and incident reports. This study investigates the automated classification of IT support requests as a data-driven decision-support task within a real-world enterprise ITSM context, addressing challenges posed by multilingual content and severe class imbalance. We propose an applied machine-learning and natural language processing (NLP) pipeline combining text cleaning, stratified data splitting, and supervised model training under realistic evaluation conditions. Multiple classification models were evaluated on historical enterprise ticket data, including a Logistic Regression baseline and transformer-based architectures (multilingual BERT and XLM-RoBERTa). Model validation distinguishes between deployment-oriented evaluation on naturally imbalanced data and diagnostic analysis using training-time class balancing to examine minority-class behavior. Results indicate that Logistic Regression performs reliably for high-frequency, well-defined request categories, while transformer-based models achieve consistently higher macro-averaged F1-scores and improved recognition of semantically complex and underrepresented classes. Training-time oversampling increases sensitivity to minority request types without improving overall accuracy on unbalanced test data, highlighting the importance of metric selection in ITSM evaluation. The findings provide an applied empirical comparison of established text-classification models in ITSM, incorporating both predictive performance and computational efficiency considerations, and offer practical guidance for supporting IT support agents during ticket triage and automated request classification.

Keywords: BERT; logistic regression; natural language processing; text classification; IT service management; decision support systems



Academic Editor: Fernando De la Prieta Pintado

Received: 24 January 2026

Revised: 9 February 2026

Accepted: 12 February 2026

Published: 21 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

IT Service Management (ITSM) provides a structured framework for delivering and maintaining IT services in alignment with business objectives. Standards and best-practice frameworks such as ITIL 4, COBIT 2019, and ISO/IEC 20000 define governance models and process maturity levels that emphasize consistent service delivery, accountability, and continuous improvement [1–3]. Within these frameworks, automating key operational steps, such as incident classification and routing, enhances process maturity by reducing human dependency, enforcing standardized request handling, and accelerating service response. The proposed approach, therefore, aligns with the continuous service improvement principles central to ITSM governance, positioning intelligent ticket classification as a practical enabler of digital transformation in enterprise service environments.

Investments in automation and digitization enable organizations to adapt to demographic shifts, such as an aging workforce and shrinking talent pools, by boosting productivity, reducing dependence on manual labor, and preserving high service standards even as headcount fluctuates [4]. In the IT domain, these benefits are critical: modern enterprises generate thousands of unstructured support tickets each month, ranging from simple password resets and user onboarding requests to complex infrastructure changes and cybersecurity incidents. Each ticket must be accurately classified by type and routed to the appropriate resolution team; manual triage of free-text descriptions is time-consuming, inconsistent, and prone to human error. Delays in incident handling not only frustrate end users but also inflate operational costs and expose organizations to elevated risk. Automated ticket classification promises to streamline this process by rapidly interpreting textual content, enforcing standardized workflows, and freeing IT staff to focus on higher-value tasks.

In a typical enterprise IT environment, such as the company whose ITSM data were used in this study, end users submit support requests via email, telephone, or a local web portal. Figure 1 illustrates the actual operational workflow used by the company's IT department to process new requests. An IT help-desk agent manually determines the request type and assigns the responsible team based on the user's description. These steps are essential for ticket routing, but are also among the most time-consuming and error-prone. Our proposed approach automates these two stages to accelerate triage and improve consistency. An IT help-desk agent then performs the following steps to register and route each ticket:

- Step 1: Priority assignment (manual). The agent reads the user's description and selects a priority level (P1–P4) based on urgency and business impact.
- Step 2: Request type determination (manual). The agent chooses one of the predefined categories (Incident, Change Request, Access Management, Problem, Inquiry, etc.).
- Step 3: Responsible team assignment (manual). Based on the request type and content, the agent routes the ticket to an appropriate resolution team (e.g., Helpdesk, System Administrators, Network Security, Application Support).
- Step 4: Clarification loop (optional). When the initial description is incomplete or ambiguous, the agent may contact the requester for additional details before finalizing fields (2) and (3).
- Step 5: Ticket resolution workflow. Once the form is fully specified, the ticket enters the chosen team's queue for investigation and resolution.

Steps (2) and (3), highlighted in red in Figure 1, represent the most time-consuming and error-prone tasks. Our goal is to automate these two steps using machine learning and NLP models, thereby reducing triage time, minimizing misclassifications, and freeing help-desk staff to focus on problem resolution.

By automating request type determination and responsible team assignment, the proposed system can accelerate ticket routing and standardize triage procedures. This reduces the workload on help-desk staff and lowers the risk of human error. A final analyst review is still retained to address potential misclassifications specific to the organization's domain. All tickets remain stored in the company's on-premises SQL database, ensuring data security and controlled access. This database retains comprehensive metadata for each ticket, including submission date, description, priority, status, assigned team, and subsequent handling details.

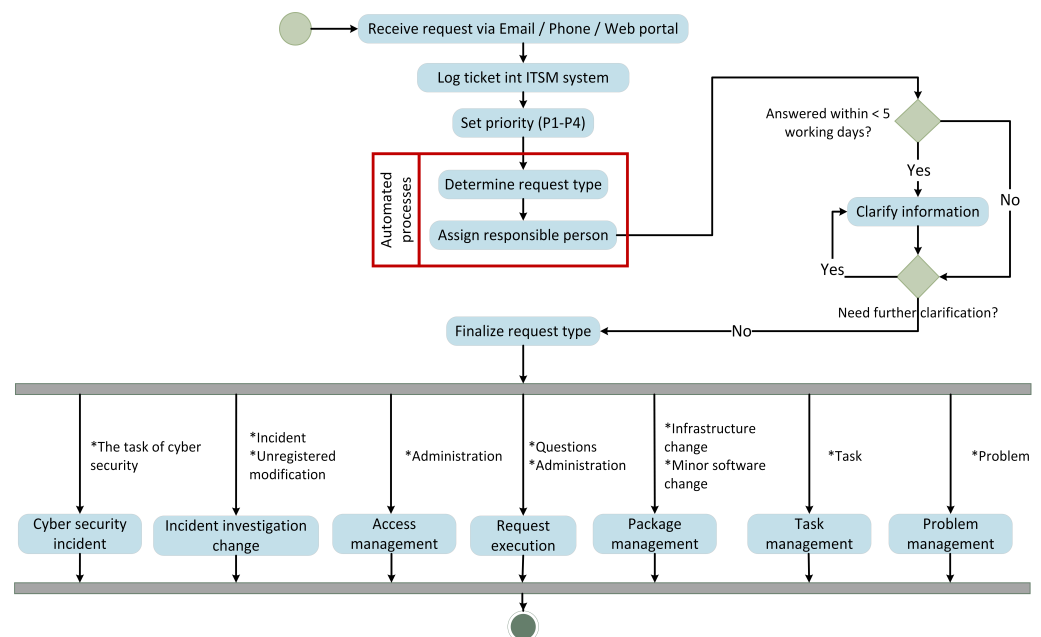


Figure 1. Activity diagram of IT support ticket handling process; the red-highlighted part indicates steps targeted for automation. The asterisk (*) denotes explanatory notes within the diagram.

Since support requests are written by humans in free-text form, they exhibit a wide range of writing styles, abbreviations, and domain-specific jargon. In the case of the studied company, the primary working language is Lithuanian, but since the company has departments across many EU countries, requests may also be submitted in English. This introduces an additional layer of complexity in natural language processing, as the system must handle multilingual or mixed-language input. Therefore, effective text analysis requires robust methods that can process both continuous and categorical data, while also accounting for linguistic and structural variability. Moreover, many support requests follow similar patterns or repeat over time. This repetition opens opportunities for automation by learning from previously resolved cases to assist with the classification and handling of new, similar requests. Traditional deep neural networks are mathematical functions designed to process large datasets of discrete inputs, but most real-world datasets combine the following two variable types:

- Continuous variables: Numerical features, such as response times or priority levels, which can take any value within a range;
- Categorical variables: Discrete labels or tokens used to partition data into classes, such as request types or team identifiers.

Recent research has shown that traditional machine-learning classifiers [5] remain effective on well-structured, moderate-sized text corpora. For example, Logistic Regression (LR) can achieve classification accuracies of approximately 94%, compared to around 92% for Random Forest and 89% for Naive Bayes on IT-support ticket datasets [6]. Other studies have explored ensemble methods such as Random Forest and XGBoost in this domain, reporting that XGBoost achieves a higher accuracy of around 90%, when combined with careful text pre-processing and hyperparameter optimization [7].

However, these traditional approaches often struggle with large-scale, unstructured data and severely imbalanced class distributions [8]. In this study, the dataset itself represents an enterprise-scale ITSM environment, containing over 87,000 multilingual support tickets collected from real operational systems. Natural language processing (NLP) techniques, encompassing text cleaning, vectorization [9], and semantic embedding, are therefore critical for robust performance. Classical feature-extraction methods, such as

TF-IDF (Term Frequency–Inverse Document Frequency), remain foundational for short texts, although embedding methods like Word2Vec can better capture semantic relationships in longer or more complex requests [10,11]. While LR scales linearly with input size and can be retrained efficiently on millions of records, transformer models such as BERT require greater computational resources during fine-tuning.

The advent of deep transformer models, notably BERT (Bidirectional Encoder Representations from Transformers), has further advanced the field. BERT's bidirectional attention mechanism enables the model to consider both preceding and succeeding context, which is particularly advantageous for parsing detailed IT problem descriptions [12]. Comparative studies indicate that transformer-based models such as BERT and RoBERTa consistently outperform traditional classifiers on text classification benchmarks—albeit at the cost of increased computational resources [13]. In this work, we leverage these advances to examine how class imbalance and training-time data augmentation affect both traditional and transformer-based models in the context of IT support ticket triage, with particular emphasis on realistic deployment conditions and minority-class behavior.

This publication consists of six sections. Section 2 provides a literature survey and reviews the applied methods for text classification, including both traditional ML and transformer-based architectures. Section 3 describes the experimental design, data challenges, and validation framework, including stratified data splitting and training-time class-balancing strategies. Section 4 presents the obtained results and performance comparisons under both diagnostic and deployment-oriented evaluation regimes. Section 5 discusses the implications of the findings for practical ITSM automation, including computational trade-offs. Finally, Section 6 concludes with recommendations and directions for future work.

2. Literature Review and Applied Classification Methods

In this section, we first survey related work on automated text classification in ITSM, highlighting both traditional machine-learning approaches and recent advances in deep NLP models. We then present the primary methods applied in our research: LR as a representative of classical classifiers, and BERT as a state-of-the-art transformer-based model. In addition, closely related transformer architectures are briefly discussed where relevant to contextualize the experimental comparisons. By grounding our work in both established and modern approaches, we aim to demonstrate how classical classifiers and transformer-based models can be effectively applied and compared within a unified experimental framework for ITSM request classification.

2.1. Related Studies

Automated text classification has been extensively studied across various domains, including cybersecurity, healthcare, social media analysis, and email filtering. Early work often combined traditional feature-extraction methods (TF-IDF, bag-of-words, Word2Vec) with classical classifiers (Logistic Regression, Random Forest, Naive Bayes), while more recent research leverages transformer-based architectures (BERT, RoBERTa) for a deeper semantic representation. Table 1 summarizes representative prior studies on automated text classification relevant to ITSM and enterprise service environments.

Table 1. Summary of key text classification methods for imbalanced data, grouped by thematic focus.

Theme	Reference	Title	Dataset	Methods	Results/Notes
Imbalance handling	[14]	Assessing the impact on prediction quality and inference from balancing in multilevel Logistic Regression	Simulated GDM Medicaid (Indiana)	Undersampling (General, Rank, Naïve) + 2-level LR	Balanced data trains 85% faster and reduces bias under mis-specification; recommends broader balancing strategies.
	[15]	Deep generative approaches for oversampling in imbalanced data classification problems: A comprehensive review and comparative analysis	Various image, medical, and fraud datasets	Review of SMOTE, ADASYN, GAN, and VAE variants	Highlights fairness and transparency issues; urges ethical constraints in synthetic oversampling.
	[16]	A robust ensemble classifier for imbalanced data via adaptive variety oversampling and embedded sampling rate	14 benchmark datasets	VAO partial oversampling + strengthened AdaBoost	Achieves top rank (F1, G-Mean, AUC); theoretical bounds for sampling rate optimization remain open.
Transformer-based models	[17]	Email spam detection by deep learning models using a novel feature selection technique and BERT	LingSpam corpus	GWO feature selection + BERT embeddings → CNN/BiLSTM/LSTM	99.14% accuracy; BERT reduces training time to 18.8 s, outperforming traditional tokenization.
	[18]	Advanced BERT and CNN-based computational model for phishing detection in enterprise systems	Enterprise email dataset	BERT feature extraction + CNN classifier	97.5% accuracy; runtime/memory footprint not evaluated—distilled variants recommended.
	[19]	Ransomware detection and family classification using fine-tuned BERT and RoBERTa	3300 API-call sequences from 10 ransomware families + 300 benign samples	Fine-tuned BERT & RoBERTa on dynamic API telemetry	BERT: 95.6% accuracy; RoBERTa: 94.4%; suggests broader real-world benchmarking.
	[20]	Comparative analysis of ML algorithms for email phishing detection using TF-IDF, Word2Vec, and BERT	Public phishing email corpus	TF-IDF, Word2Vec, BERT embeddings + LR/DT/RF/MLP	BERT clearly outperforms bag-of-words and static embeddings; calls for larger datasets and hybrid models.
Interpretability and/or multilingual NLP	[21]	Strategies for enhancing the performance of news article classification in Bangla: Handling imbalance and interpretation	Bangla news corpus (437,948 items)	RUS + SMOTE + BERT; LIME/SHAP interpretability	99.04% accuracy (balanced), 72.23% (imbalanced); calls for lightweight, interpretable models.
	[22]	From Zero to Hero: On the Limitations of Multilingual Transformers in Low-Resource Language Transfer	Low-resource European languages (cross-lingual transfer benchmarks)	Evaluation of multilingual transformer models under low-resource and domain-shift conditions	Limited robustness under low-resource and domain-shift conditions; motivates domain-specific evaluation
	[23]	Classification of mental illnesses from Reddit posts using Sentence-BERT embeddings and neural networks	Reddit posts (11 mental-health categories)	SBERT embeddings + LSTM/BiLSTM networks	Accuracy = 70.42%, F1 = 0.70; suggests richer datasets and transformer-only baselines.
Classical + hybrid ML	[24]	Spam email classification based on cybersecurity potential risk using natural language processing	INCIBE private + Bruce Guenter public corpora	56 NLP features + RF classifier/regressor	F1 = 0.914 (classification); alert calibration and deployment aspects remain untested.
	[25]	Efficient e-mail spam filtering approach combining Logistic Regression and Orthogonal Atomic Orbital Search	CSDMC2010, Enron	OAOS (AOS + orthogonal learning) + Logistic Regression	Strong F1 gains (95–96% on CSDMC2010; 74–78% on Enron); promotes multi-objective optimization.
	[26]	Efficient spam email classification via Logistic Regression trained by a modified Social Network Search algorithm	CSDMC2010 corpus	DOSNS (enhanced SNS) + Logistic Regression	Outperforms seven metaheuristic-LR baselines; metrics consistent with other heuristic-tuned LR models.

The reviewed studies can be grouped into four recurring themes: (i) imbalance-handling strategies, (ii) transformer-based models, (iii) interpretability and multilingual generalization, and (iv) classical or hybrid machine-learning approaches. While transformer architectures have achieved notable success across various domains, most prior work has focused on English-only or single-language corpora, with limited investigation of multilingual operational datasets. Similarly, few studies have compared model performance across both balanced and imbalanced scenarios using real enterprise data, where category distributions are naturally skewed. These gaps highlight the need for applied research that jointly examines data balancing and contextual understanding in multilingual, real-world IT service environments.

Tawil et al. [20] compared three feature pipelines—TF-IDF, Word2Vec, and BERT—across four classifiers for phishing email detection. Using TF-IDF, both LR and Random Forest achieved F1 scores of 0.98 and accuracies of 0.97; Word2Vec slightly trailed (e.g., LR F1 score = 0.96). Fine-tuned BERT outperformed all baselines (F1 = 0.99, accuracy = 0.99), demonstrating its strength in capturing bidirectional context. However, their evaluation was limited to a single public corpus, and they did not explore ensemble or zero-shot approaches.

In the healthcare IoT domain, Hussain et al. [19] created a custom ransomware dataset and fine-tuned BERT and RoBERTa on API call sequences. BERT attained an accuracy of 95.60% (loss = 0.1650), outperforming RoBERTa's 94.39% (loss = 0.1948). Their confusion-matrix analysis highlighted BERT's near-perfect detection of certain ransomware families but higher error rates on benign classes. Key limitations include the small dataset size and the lack of deployment benchmarks on resource-constrained devices.

On social media, Wagay and Jahiruddin [23] applied SBERT embeddings with LSTM/BiLSTM networks to classify mental-health conditions on Reddit (11 classes). The SBERT-BiLSTM model achieved an accuracy of 70.42% (F1 = 0.70), outperforming both a plain LSTM and other traditional classifiers. This work highlights the value of contextual embeddings but also reveals challenges related to class imbalance and platform-specific language.

Hybrid approaches have been proposed to enhance classical classifiers. Manita et al. [25] introduced OAOS-LR, which uses an improved metaheuristic (orthogonal-learning-enhanced AOS) to optimize LR for spam filtering, achieving up to 96.30% F1 on CSDMC2010. Gupta et al. [18] combined BERT embeddings with a CNN classifier for enterprise phishing detection, yielding an accuracy of 97.5% versus 95% for plain LR. Similarly, Nasreen et al. [17] proposed GWO-BERT using the Grey Wolf Optimizer for feature selection before BERT classification, reporting an accuracy of 99.14% on LingSpam.

Imbalance handling is another critical theme. Hasib et al. [21] showed that applying $RUS + SMOTE$ to a 437948 article Bangla news corpus boosted BERT's accuracy from 72.23% to 99.04%, while LR remained stable around 90%. Hayaeian Shirvan et al. [15] surveyed oversampling methods, contrasting SMOTE, ADASYN, and various GAN/VAE-based variants, and highlighted the need for reproducible benchmarks and domain-specific evaluation metrics.

Recent multilingual NLP studies further emphasize that contextual language models do not behave uniformly across languages and domains. Ulčar et al. [27] demonstrate that even state-of-the-art BERT and ELMo representations exhibit substantial performance variability depending on language and task, particularly for less-resourced languages. Similarly, Catelli et al. [28] show that cross-lingual transfer learning can significantly improve performance in low-resource settings, but it requires careful task-specific adaptation.

For morphologically rich and lower-resource European languages, prior peer-reviewed studies have shown that multilingual transformer models often struggle to preserve linguistic fluency and domain knowledge simultaneously when transferred across domains.

In particular, Lauscher et al. [22] demonstrate that multilingual transformers exhibit limited robustness in low-resource and domain-shifted settings, reinforcing the importance of applied, domain-specific evaluation rather than reliance on benchmark performance alone.

Despite these advances, existing studies largely focus on benchmark datasets or sentiment analysis tasks, with limited attention to multilingual, operational enterprise data characterized by severe class imbalance and heterogeneous request semantics, as encountered in real-world IT service management systems.

Overall, the methods summarized in Table 1 reveal two consistent insights from the text-classification literature: first, transformer models, such as BERT, generally outperform classical classifiers when trained on properly balanced datasets; second, simpler models such as LR remain competitive, especially in heavily imbalanced or resource-constrained scenarios. Both approaches have proven valuable across phishing, spam, ransomware, and social media classification tasks, yet few studies estimate them on large, multilingual, real-world enterprise ticket streams. Motivated by these findings, our work applies both LR and a fine-tuned BERT model to IT support ticket classification in a company's on-premises environment. Crucially, we incorporate a threshold-based oversampling augmentation, adapted from the reviewed imbalance-handling techniques, to ensure that minority ticket categories are adequately represented during training. By doing so, we place both classifiers on equal footing and can rigorously measure how class balancing influences performance in a practical, production-scale ITSM setting.

2.2. Logistic Regression

Logistic Regression (LR) was selected as a baseline classifier due to its widespread use in text classification tasks, interpretability, and computational efficiency. In IT Service Management contexts, LR remains attractive because it scales well to large, sparse feature spaces generated by TF-IDF representations and provides transparent decision boundaries that are easier to audit in operational settings. Prior studies have shown that LR can perform competitively on high-frequency and well-defined ticket categories, making it a strong reference point for evaluating the added value of contextual deep-learning models.

Logistic regression is a supervised learning algorithm used for binary and multiclass classification [14]. Despite its statistical origins, it remains a popular baseline for text classification due to its efficient training even on large datasets, its compatibility with sparse text vectorization such as TF-IDF or CountVectorizer, and its interpretability, as model coefficients directly indicate the influence of each feature (e.g., word) on the classification.

In the literature, LR often matches or exceeds the performance of more complex models on small to medium-sized, structured datasets, and its simplicity makes it robust when deep networks offer diminishing returns [15,20].

The objective function of LR is the regularized log loss:

$$J(\theta) = -\frac{1}{N} \sum_{i=1}^N \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right] + \frac{\lambda}{2} \|\theta\|^2 \quad (1)$$

where:

- N is the number of training examples;
- y_i is the observed class for the i -th example (0 or 1);
- p_i is the predicted probability that $y_i = 1$;
- θ is the model parameter (coefficient);
- λ is the regularization strength (L2 penalty);
- $\|\theta\|^2$ is the squared L2 norm of the parameter vector, which penalizes large weights to reduce over-fitting.

In this study, we implement LR as a baseline using a two-step pipeline. First, ticket texts are vectorized with TF-IDF over unigrams, restricting the vocabulary to the 10,000 most frequent terms to control dimensionality. Next, a SAGA-solver LR model is trained, which offers efficient convergence on sparse data and native support for L2 regularization. To address class imbalance, different strategies are considered depending on the evaluation regime. In diagnostic experiments, class weighting or training-time over-sampling is applied to analyze model sensitivity to minority classes, whereas deployment-oriented evaluation is performed without class re-weighting to preserve the natural data distribution. The maximum number of iterations is increased to 3000 to ensure convergence on the large dataset, and the regularization strength λ is selected via cross-validation. This configuration yields a fast and interpretable classifier that serves as a robust benchmark against transformer-based approaches.

2.3. BERT Model

BERT was selected to model contextual and semantic relationships in IT support tickets that are difficult to capture using bag-of-words representations. Transformer-based models have demonstrated superior performance in handling ambiguous, context-dependent, and multilingual text, which are common characteristics of real-world ITSM data. In contrast to classical models, BERT leverages bidirectional attention to incorporate both local and global linguistic context, making it particularly suitable for complex request descriptions and minority classes with limited training examples.

BERT is a deep neural network architecture built upon multiple stacked Transformer encoder layers (see Figure 2). Each encoder layer consists of a multi-head self-attention mechanism followed by position-wise feed-forward networks, with residual connections and layer normalization at each sub-layer. The key innovation of BERT is its bidirectional training objective: during pre-training, the model learns to predict masked tokens (Masked Language Modeling, MLM) and to determine sentence continuity (Next Sentence Prediction, NSP). These tasks force BERT to capture rich contextual representations of words, taking into account both their preceding and following tokens.

For text classification, a task-specific head is added on top of the pre-trained Transformer: the hidden state of the special [CLS] token is passed through a fully connected layer with softmax activation to produce class probabilities [29]. During fine-tuning, all model parameters, including those of the Transformer encoder blocks and classification layer, are updated on the labeled dataset.

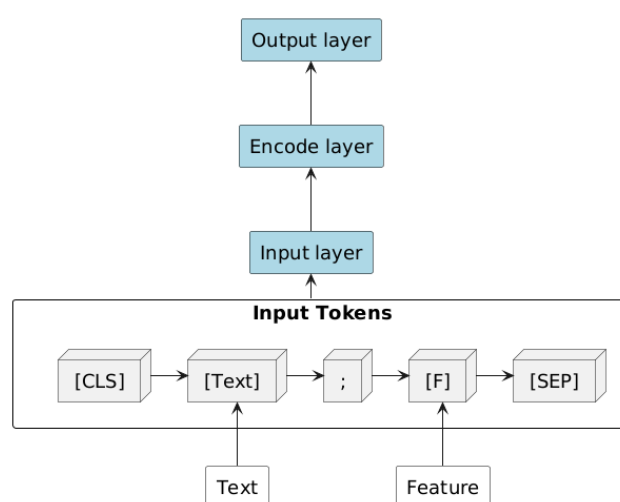


Figure 2. Basic structure of BERT [29].

In our study, we leverage transformer-based language models by fine-tuning multilingual BERT (mBERT) for ITSM request classification. Specifically, we use the BERT-base-multilingual-cased checkpoint and adapt it to the IT support domain. Ticket descriptions are tokenized into WordPiece subword units and padded or truncated to a maximum sequence length of 128 tokens. The input representation is formed by summing token, position, and segment embeddings, which are processed through 12 Transformer encoder layers to model contextual dependencies across the entire sequence. The contextualized representation corresponding to the [CLS] token is passed to a softmax classification head to produce class probabilities.

2.4. Model Evaluation

Evaluating machine-learning models is essential to assess their ability to generalize to unseen data and to ensure their practical suitability in operational settings. In text classification tasks with highly imbalanced class distributions, no single metric is sufficient; therefore, we compute a suite of complementary measures to obtain a comprehensive view of model behavior.

We report standard classification metrics as defined by Miller et al. [30], including overall accuracy (the proportion of correctly classified tickets), precision, recall, and the F1-score. In addition, confusion matrices are inspected to analyze class-specific prediction patterns and to identify systematic misclassifications. Given the strong class imbalance characteristic of ITSM data, macro-averaged and weighted variants of the F1-score are used to distinguish between balanced class performance and expected behavior under natural class distributions.

To account for class imbalance, several complementary evaluation measures have been proposed in the literature. For example, the Matthews correlation coefficient (MCC) summarizes all entries of the confusion matrix into a single correlation value between -1 and $+1$ and is often recommended for imbalanced classification problems [31]. Similarly, Cohen's Kappa has been widely used to quantify agreement between predicted and true labels beyond chance [32].

All metrics are computed on a held-out test set comprising 20% of the data. To ensure realistic performance assessment, this test set always preserves the naturally imbalanced class distribution of operational ITSM data. Class-balancing techniques, when applied, are restricted to the training data and are used exclusively for diagnostic analysis. Model hyperparameters are selected on the training split using cross-validation where applicable. After model development, the trained classifiers are further evaluated on a temporally disjoint dataset from the year 2025, which was not used during training or initial testing. This final evaluation step provides an additional assessment of generalization under real-world, temporally shifted conditions.

This multi-metric evaluation framework enables a fair comparison between LR and transformer-based models, highlighting their respective strengths and limitations across frequent and rare request categories, while also supporting interpretation of model behavior under both diagnostic and deployment-oriented evaluation regimes.

3. Materials and Methods

3.1. Overview of Experimental Workflow

The joint evaluation of LR and BERT enables an applied comparison between interpretable classical models and context-aware deep learning approaches under identical preprocessing and balancing conditions. In this section, we describe thoroughly the end-to-end preparation and training pipeline for our deep-learning models designed to automate IT support ticket classification. The workflow comprises the following five main stages:

Data Acquisition and Annotation. We begin with a raw dataset of 150,441 historical tickets, collected from 2024, each labeled with one of nine request types and one of eleven responsible teams. All records were exported from the on-premises ITSM SQL database, ensuring complete metadata integrity.

Text Cleaning and Deduplication. Raw ticket notes often contain HTML tags, special characters, and duplicate or near-duplicate entries, as the company system allows entering rich-text or submitting notes via email, which supports HTML formatting. We apply a cleaning procedure to remove unwanted symbols and markup, then use TF-IDF vectorization combined with cosine similarity filtering to identify and eliminate redundant records, reducing semantic noise.

Class Balancing via Oversampling. Initial analysis revealed a severe imbalance: Administration and Incident together account for nearly half of all tickets, while several categories fall below 1%. To address this, we classify each label as major ($>5\%$ of total) or minor ($\leq 5\%$), then randomly duplicate minor-class examples until they match the volume of major classes. This approach produces a training set with more uniform class representation while preserving realistic distribution. Balancing before splitting was a pragmatic choice due to low data volume in minority classes, to ensure every class was represented during training.

Dataset Splitting and Feature Extraction. The balanced data are shuffled and split into training (80%) and test (20%) sets using a stratified procedure. For traditional models, we extract TF-IDF features on unigrams. For the transformer model, text is tokenized with the multilingual BERT tokenizer (transformers v4.50.0) and packaged into PyTorch (v2.5.1) datasets for batch loading.

Model Implementation and Training. Logistic Regression is implemented in scikit-learn as a pipeline combining TF-IDF feature extraction with a SAGA solver classifier. Class weights are set to “balanced” to further mitigate residual imbalance.

BERT-based on the Bert-base-multilingual-cased architecture, extended with a classification head mapped via label2id/id2label. Training utilizes the Hugging Face Trainer with a learning rate of 2×10^{-5} , eight epochs, and a warmup schedule.

Each stage is crucial to ensure a fair and reproducible comparison between traditional and deep-learning approaches. In the subsections that follow, we provide a deeper dive into the data source and structure (Section 3.2), the cleaning and deduplication procedure (Section 3.3), the oversampling strategy (Section 3.4), feature encoding and dataset partitioning (Section 3.4), and, finally, model configurations and training protocols (Section 3.5).

3.2. Data Description and Distribution

The dataset comprises 150,441 historical IT support tickets extracted from the organization’s on-premises ITSM SQL system. Each record consists of:

- ID: Unique ticket identifier;
- Notes: Free-text description of the user’s request (usually in Lithuanian, or in another language if the user is from another country);
- Request Type: One of nine predefined categories (e.g., Incident, Administration, Cybersecurity Incident);
- Responsible Team: One of eleven resolution teams (e.g., IT Helpdesk, IT Systems Administrators, Cybersecurity Team).

Table 2 shows a sample of the raw data structure and a few request text fragments.

Figure 3c illustrates the marginal distribution of tickets by request type. Administration and Incident together account for over 50% of all records, whereas several categories, such as Infrastructure Change and Unregistered Change, make up less than 3% each, indicating a significant class imbalance.

Table 2. Sample record structure.

ID	Notes	Request Type	Responsible Team
1	The server has been down since 8 am...	Incident	IT Systems Administrators
2	Creating a new user in the system	Administration	IT Helpdesk
3	Notification of malicious activity	Cybersecurity Incident	Cybersecurity Team

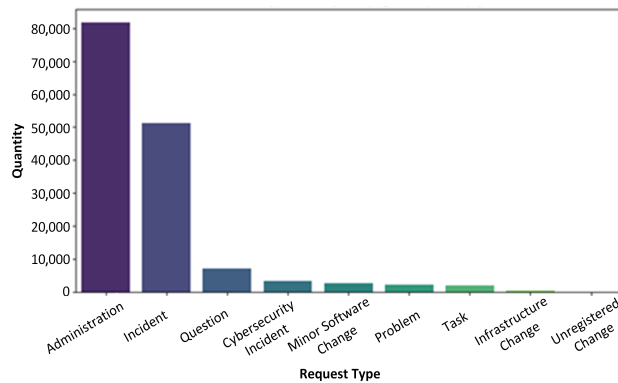
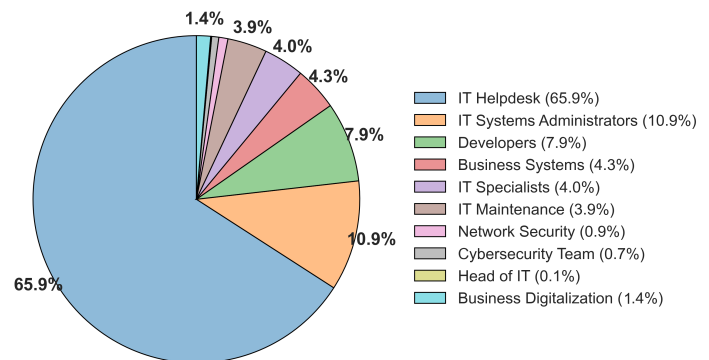
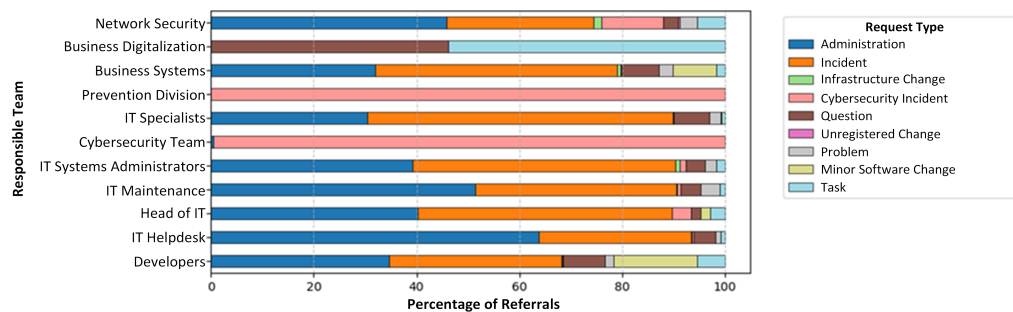
**(a)** Distribution of tickets by request type.**(b)** Distribution of tickets by the responsible team.**(c)** Distribution of referral types across the responsible team.**Figure 3.** Unprocessed data source presentation.

Figure 3b shows the breakdown by the responsible team. The IT Helpdesk team handles 65.9% of all tickets, followed by 10.9% of IT Systems Administrators and 7.9% of Developers. Eight other teams each contribute less than 5%, with two teams below 1%.

To examine specialization patterns, Figure 3a displays the conditional distribution of request types within each team. For instance, Cybersecurity and Prevention teams predominantly receive Cybersecurity Incident tickets, while the IT Helpdesk is dominated by Administration requests. Teams such as Business Systems and Developers show a higher share of Minor Software Change tickets.

These analyses confirm a pronounced imbalance, both marginally and conditionally, which can negatively impact model training, especially for rare classes. In Section 3.3, we describe our oversampling strategy to mitigate these imbalances and ensure robust learning across all categories.

3.3. Text Cleaning and Deduplication

Text cleaning is an essential preprocessing step to remove noise, standardize the corpus, and ensure that models learn from genuine semantic content rather than artefacts. Since many support tickets arrive via email and often include HTML formatting and embedded markup, we first strip all HTML tags and unwanted symbols (e.g., '<', '>', '/'), convert the text to lowercase, and normalize extra whitespace. Punctuation and diacritics are then either removed or mapped to their base characters. Earlier research has demonstrated that such rigorous cleaning can improve classification accuracy by up to 10% [33].

Since many tickets originated from templated system messages or automated alerts, the dataset contained numerous identical or near-duplicate entries. To identify and remove these redundancies, a TF-IDF vectorization approach [34] was applied:

1. Each cleaned ticket text was transformed into a TF-IDF vector over unigrams.
2. Cosine similarity was computed between all pairs of TF-IDF vectors.
3. Pairs exceeding a similarity threshold of 0.95 were deemed duplicates; one representative ticket from each pair was retained, and the remainder was removed.

The threshold value of 0.95 was selected based on exploratory analysis of the similarity distribution and qualitative inspection of matched ticket pairs. Lower thresholds (e.g., 0.90–0.93) were observed to remove semantically distinct user requests that shared domain-specific terminology, while higher thresholds (>0.97) failed to capture templated system-generated messages differing only in timestamps or identifiers. The chosen threshold, therefore, represents a compromise between effective noise reduction and preservation of semantic diversity and is consistent with commonly adopted practices in text deduplication tasks.

This deduplication process delivered two key advantages. First, it substantially reduced noise by removing boilerplate and repetitive system-generated messages, which often obscure meaningful linguistic patterns and hinder model learning. These duplicates were predominantly templated notifications or automatically generated emails, rather than semantically diverse user requests. Second, the process made the dataset more compact and efficient: the total number of tickets decreased from 150,441 to 87,859 (41.6% reduction), while preserving the semantic diversity of the data. Importantly, class-wise removal statistics were monitored during deduplication. Although high-frequency categories experienced the largest absolute reductions, minority classes were retained proportionally, ensuring that the overall class distribution and sample structure were not distorted. The resulting data were stored in CSV format for further processing. By integrating rigorous text cleaning with TF-IDF-based deduplication, we produced a dataset that is both more representative of genuine support requests and more manageable for training both classical and deep-learning classifiers.

The applied threshold-based oversampling moderately increased the dataset size, adding only linear computational overhead. Both models remained efficient, with only a minor increase in training time observed.

3.4. Class Balancing

The threshold of 5% used to distinguish major and minor classes was selected based on exploratory analysis of the class distribution and operational relevance within the enterprise ITSM dataset. Categories below this threshold exhibited insufficient representation for stable model training, while still corresponding to meaningful real-world request types handled by the organization.

Random oversampling was selected as a conservative balancing strategy to preserve the original semantic content of IT support tickets. In enterprise ITSM environments, artificially generated samples may introduce unrealistic or operationally invalid requests, potentially distorting decision-support outcomes. By duplicating existing minority-class samples, this approach increases class visibility during training while maintaining data authenticity.

An initial analysis of both Request Type (Stage 1) and Responsible Team (Stage 2) labels revealed severe class imbalance, with a small number of categories accounting for the majority of tickets and several classes represented by very few examples. In the diagnostic evaluation, class frequencies were first computed on the full dataset, and classes exceeding 5% of the total were designated as major classes, while those at or below 5% were treated as minor classes. Targeted oversampling was then applied to the minor classes by randomly duplicating their instances until a more even distribution was achieved

using the `sklearn.utils.resample` utility [35]. This diagnostic setup was used to analyze model sensitivity to extreme imbalance and to ensure sufficient exposure of low-frequency categories during training. Figure 4 illustrates the oversampling workflow.

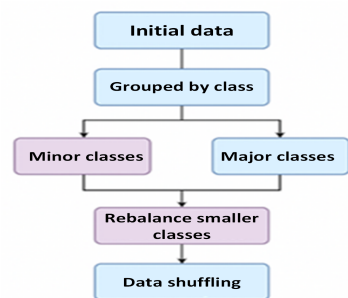


Figure 4. Class balancing process: split into major/minor, oversample minors, then merge and shuffle.

After training-time balancing, the distribution of request types (see Table 3) within the training data becomes more uniform, improving representation for minority categories such as Minor Software Change, Problem, and Task, while preserving a degree of imbalance to reflect realistic operational proportions. A similar effect is observed for responsible team labels (see Table 4), where intermediate and low-frequency teams gain sufficient representation for learning alongside dominant categories such as IT Helpdesk. Classes with extremely limited data are retained to preserve operational completeness; however, results for these categories should be interpreted with caution due to limited sample diversity.

Table 3. Request type distribution after balancing (unique 87,859).

Request Type	Count	Unique	%
Administration	47,600	47,600	54.18
Incident	31,281	31,281	35.60
Minor Software Change	27,622	1513	1.72
Cybersecurity Incident	20,373	1111	1.26
Problem	17,206	940	1.07
Task	13,591	745	0.85
Infrastructure Change	4428	249	0.28
Question	4415	4415	5.03
Unregistered Change	76	5	0.01

Table 4. Responsible team distribution after balancing (unique 87,859).

Team	Count	Unique	%
IT Specialists	61,433	4260	4.85
IT Helpdesk	60,409	60,409	68.76
IT Maintenance	55,587	3213	3.66
Network Security	32,652	575	0.65
Cybersecurity Team	14,979	819	0.93
Business Systems	13,017	3603	4.10
IT Systems Administrators	8405	8405	9.57
Developers	6513	6513	7.41
Head of IT	1472	58	0.07
Business Digitalization	159	4	0.00

For deployment-oriented evaluation, no balancing is applied beyond the stratified train–test split. The test set always retains the naturally imbalanced class distribution characteristic of real-world ITSM environments, ensuring that reported performance metrics reflect realistic operational behavior rather than artificially equalized conditions. Training-time balancing is therefore used exclusively as a diagnostic tool to analyze model sensitivity

to underrepresented classes, while final performance assessment is conducted on unmodified data.

Alternative imbalance-handling techniques, such as SMOTE or text augmentation via back-translation, were considered. However, SMOTE operates in continuous feature space and is less suitable for sparse, high-dimensional textual representations, while back-translation may distort technical terminology or introduce linguistic artefacts that do not reflect authentic ITSM communication. Given the operational nature of the dataset and the need to preserve semantic fidelity, random oversampling was adopted as a transparent and reproducible diagnostic strategy rather than a deployment-time optimization.

3.5. Data Splitting, Feature Extraction, and Model Configuration

To ensure objective evaluation and avoid over-fitting, the deduplicated and balanced dataset was split into two disjoint sets:

- Training Set: 80% of the data used for model fitting.
- Test Set: 20% of the data held out for final performance evaluation.

Feature Extraction. We convert each note to a sparse TF-IDF vector using scikit-learn `TfidfVectorizer` (vocabulary capped at 10,000 terms). By default, the vectorizer lowercases text and uses unigrams only (tokens of ≥ 2 alphanumeric characters); no stemming or stop-word list is applied. Each document vector is L2-normalized, and inverse-document frequency smoothing is enabled. These settings provide a compact representation while keeping preprocessing minimal and ensuring full reproducibility.

Logistic Regression Pipeline. We train a multiclass LR classifier on the TF-IDF features using scikit-learn. The model uses L2 regularization with $C = 1.0$ and the SAGA optimizer, with a limit of 10,000 iterations. Other parameters are presented in Table 5. To mitigate class imbalance, the class weight is set to “balanced”, which weights each class inversely to its frequency. In inference, we predict the class with the highest softmax probability. The TF-IDF step and classifier are combined in a single Pipeline to ensure identical preprocessing at train and test time.

BERT Fine-Tuning Configuration. We fine-tune a pretrained mBERT encoder end-to-end for single-label classification. Each ticket note is tokenized with WordPiece, padded/truncated to a fixed length, and passed through the Transformer stack. We take the pooled representation of the sequence (the [CLS] token) and pass it into a lightweight classification head (linear layer + softmax) to obtain per-class probabilities. The model is optimized with cross-entropy loss; class imbalance is handled upstream by our data-preparation pipeline (duplicate filtering and oversampling of minority classes), so the learner sees a more balanced label distribution. Training uses a standard schedule (mini-batch updates with linear warm-up then decay) and validation-set early selection; all optimization hyperparameters appear in the accompanying table. Because the checkpoint is multilingual, it handles mixed Lithuanian/English input without bespoke vocabulary adaptation. Hyperparameters were initially selected according to widely adopted best practices for transformer fine-tuning reported in prior studies [17,36–38], and were subsequently examined through targeted ablation experiments to assess their impact on classification performance and computational cost. These parameters were then refined through a limited grid-search optimization to confirm stability under our dataset. Learning rate was tested within the range $[1 \times 10^{-5}, 5 \times 10^{-5}]$, batch size between 8 and 32, and the number of epochs between 4 and 10. The final configuration (see Table 6) achieved the best trade-off between validation performance and training stability, confirming that the model was not overly sensitive to moderate changes in hyperparameters.

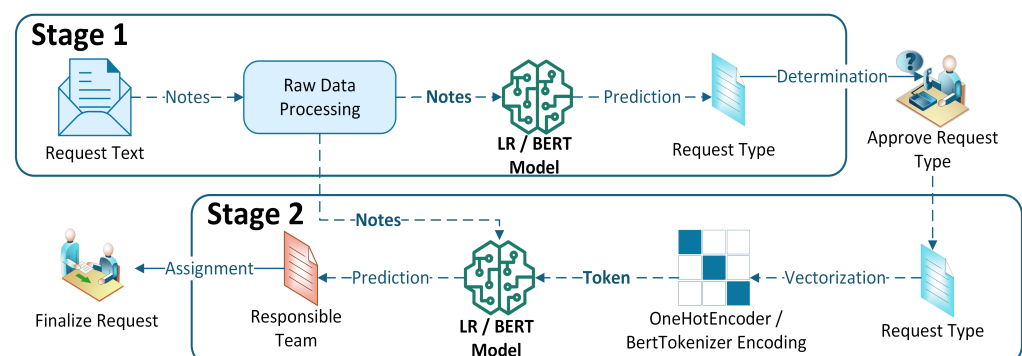
Table 5. Logistic regression hyperparameters.

Parameter	Description
max_iter = 3000	Maximum number of iterations for convergence.
solver = “saga”	Optimization algorithm supporting L1 and L2 regularization, suitable for large datasets.
class_weight = “balanced”	Automatically adjusts weights inversely proportional to class frequencies.

Table 6. BERT fine-tuning hyperparameters.

Parameter	Value and Purpose
learning_rate = 2×10^{-5}	Standard rate for stable fine-tuning.
warmup_ratio = 0.1	10% of training steps used to linearly increase the learning rate.
weight_decay = 0.01	L2 regularization to prevent over-fitting.
num_train_epochs = 8	Number of full passes over the training set.
per_device_train_batch_size = 16	Batch size per GPU/CPU during training.
per_device_eval_batch_size = 32	Batch size per GPU/CPU during evaluation.
gradient_accumulation_steps = 2	Accumulate gradients over two steps to simulate a larger batch size.
evaluation_strategy = “epoch”	Evaluate model at the end of each epoch.
save_strategy = “epoch”	Save checkpoint at the end of each epoch.
load_best_model_at_end = True	Automatically reload the best-performing checkpoint.

Our end-to-end workflow automates ticket routing in two coordinated stages (see Figure 5). Stage 1 ingests raw notes arriving via email into the ITSM system, applies text cleaning to produce model-ready input, and predicts the Request Type with a lightweight TF-IDF + logistic-regression classifier. This rapid baseline provides immediate feedback and automatically proposes a type based on the note text. In production, the suggestion can be confirmed or overridden by an agent; otherwise, it is accepted as the final label. Stage 2 predicts the Responsible Team. It consumes the same cleaned notes with a fine-tuned BERT classifier and, crucially, incorporates the Stage-1 request type as an auxiliary categorical feature (one-hot encoded) to provide additional context. Thus, the inputs are the request text and the request-type signal from Stage 1. The BERT model outputs the team prediction, which is used to assign the ticket and finalize the Request when no further clarification is needed. This two-stage design aligns with our setting of multilingual, unstructured IT tickets, where both the request type and the owning team must be inferred.

**Figure 5.** Two-stage ticket-routing pipeline.

In the training phase, both models were fitted using gold (real/true) labels already recorded in the company’s ITSM system, ensuring consistent supervision and avoiding label noise. Stage 1 predicted the request type, while Stage 2 predicted the responsible team using the same textual features together with the confirmed request type as an auxiliary

categorical input. This design mirrors the actual operational workflow in which a help-desk agent first validates the request type before assigning the ticket to a team, thus maintaining realistic sequential logic while preventing data leakage between stages.

4. Results

This section reports the experimental results in four stages. First, we present classification performance for the two-stage ITSM pipeline: (i) request type classification (Stage 1) and (ii) responsible team assignment (Stage 2). Second, we analyze the impact of class imbalance and training-time balancing procedures. Third, we examine model behavior using SHAP explanations. Finally, we summarize key findings relevant to ITSM decision support. The analysis flow is aligned with Figure 6.

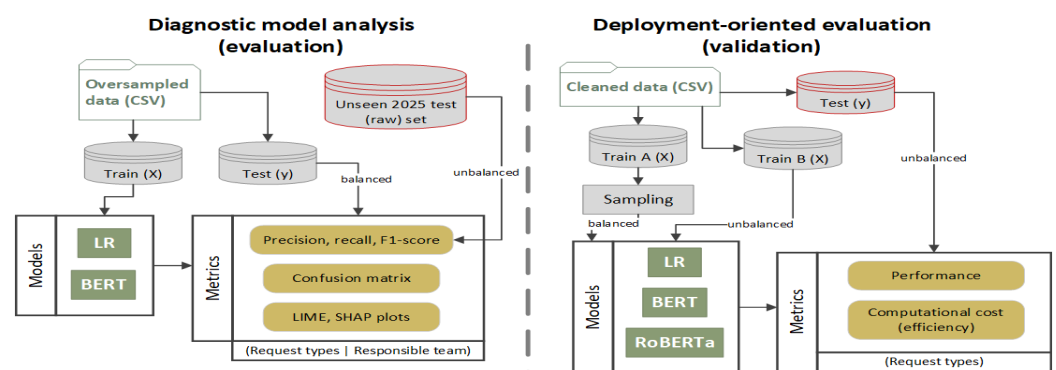


Figure 6. Flowchart of the testing evaluation process.

4.1. Model Training and Evaluation Setup

To evaluate automated IT support ticket classification, two learning algorithms were considered: a traditional LR model and mBERT. Both models were applied to the two-stage pipeline comprising request type classification (e.g., Incident, Administration, Problem) and responsible team assignment (e.g., IT Helpdesk, Developers, Cybersecurity). Model performance was assessed using precision, recall, and F1-score, with macro-averaged and weighted variants reported to account for class imbalance. Results are analyzed both at the class level and in aggregate to provide a comprehensive view of performance across frequent and rare categories. Such an evaluation method is called diagnostic model analysis.

Two complementary evaluation perspectives are reported in this section. First, we present the original pipeline results obtained under class-balancing conditions designed to mitigate extreme label skew and to ensure adequate exposure to low-frequency categories during training. Second, we introduce an extended validation of Stage 1 (request type classification) that distinguishes between deployment-oriented evaluation on naturally imbalanced data and diagnostic analysis with training-time oversampling. In this extended validation, the test set always preserves the original class distribution and is never resampled, preventing information leakage and ensuring that reported metrics reflect realistic operational conditions. Unless explicitly stated otherwise, the deployment-oriented regime is treated as the primary indicator of expected real-world performance.

4.2. Logistic Regression Classification Results

The LR model achieved relatively strong performance on frequent classes in Stage 1, as shown in Table 7. For example, in the Administration class, the model reached very high precision (0.92), but recall remained only 0.70, indicating that a portion of true cases was not recognized. The Minor Software Change and Problem classes were also classified fairly accurately, with F1-scores above 0.80, but only after class balancing was applied. Particularly great results were achieved for the Cybersecurity Incident class, where the

F1-score reached 0.97, which is highly relevant when evaluating the reliability of the model for security-related requests.

However, due to oversampling, some rare classes (e.g., Unregistered Change) reached perfect recall (1.00), i.e., many predictions were incorrectly assigned in the imbalanced dataset scenario.

The macro-average F1-score of 0.78 indicates that classification quality varies across categories; some classes are recognized very well, while others perform considerably worse. Meanwhile, the weighted-average F1-score of 0.81 reflects a high overall efficiency for the most common classes, whose numerical dominance strongly influences the dataset as a whole. In comparison, performance on the original imbalanced dataset was considerably weaker, with macro and weighted-average F1-scores of 0.42 and 0.56, respectively. Overall accuracy also improved substantially, from 0.54 on the imbalanced dataset to 0.81 after balancing across all available classes.

Table 7. Logistic Regression training results by request type.

Request Type	Unbalanced				Balanced			
	Precision	Recall	F1-Score	Support	Precision	Recall	F1-Score	Support
Administration	0.87	0.52	0.65	9494	0.92	0.70	0.80	9494
Incident	0.55	0.41	0.47	6144	0.79	0.63	0.70	6144
Infrastructure Change	0.14	0.79	0.24	921	0.50	0.99	0.66	921
Cybersecurity Incident	0.97	0.91	0.94	4045	0.97	0.99	0.98	4045
Question	0.16	0.44	0.23	931	0.25	0.54	0.35	931
Unregistered Change	0.01	1.00	0.02	17	1.00	1.00	1.00	17
Problem	0.00	0.00	0.00	3485	0.80	0.90	0.85	3485
Minor Software Change	0.72	0.72	0.72	5568	0.87	0.96	0.91	5568
Task	0.48	0.68	0.56	2714	0.76	0.89	0.82	2714
Macro average	0.43	0.61	0.42	33,319	0.76	0.84	0.78	33,319
Weighted average	0.63	0.54	0.56	33,319	0.84	0.81	0.81	33,319

The confusion matrix (see Figure 7) provides a more detailed view of how the model classifies individual categories and the frequency of misclassifications. It can be observed that the Administration and Incident classes are frequently confused with one another, and both are often misclassified as Question. By contrast, the Cybersecurity Incident class stands out with an exceptionally high level of accuracy, as 99.4% of cases, after balancing (compared to 91.0% before), were classified correctly, demonstrating the model's ability to reliably recognize this security-critical category. The Question and Problem classes are often assigned to Incident, but after balancing, more than half of the samples are correctly classified. These frequent confusions typically occur when ticket descriptions lack sufficient informative content. Nevertheless, the classification performance shown in Figure 7a (with class balancing) is consistently better than that shown in Figure 7c (without class balancing).

The confusion matrices presented in Figure 7b,d visually illustrate how the logistic regression model classified requests by the responsible team during validation. From the matrix, it can be observed that some teams were classified almost perfectly. For example, IT Division Manager, Cybersecurity, Network Security, and Business Digitalization Department (the latter with only a small number of examples) were always assigned correctly, which indicates that these classes have clearly expressed features that allow the model to distinguish them precisely from others.

In contrast, more noticeable classification errors were observed in other teams. The IT Helpdesk was frequently classified as IT Maintenance (6.9% of cases) or IT System Administrators (14.1%), which may be related to functional overlaps between these groups.

Similarly, IT System Administrators were often misclassified as IT Helpdesk (33.4% of cases), suggesting that the model struggles to separate their scopes of activity, most likely due to similar ticket content or the general overlap in IT infrastructure topics. Consistently, it was observed that balancing helped improve prediction results.

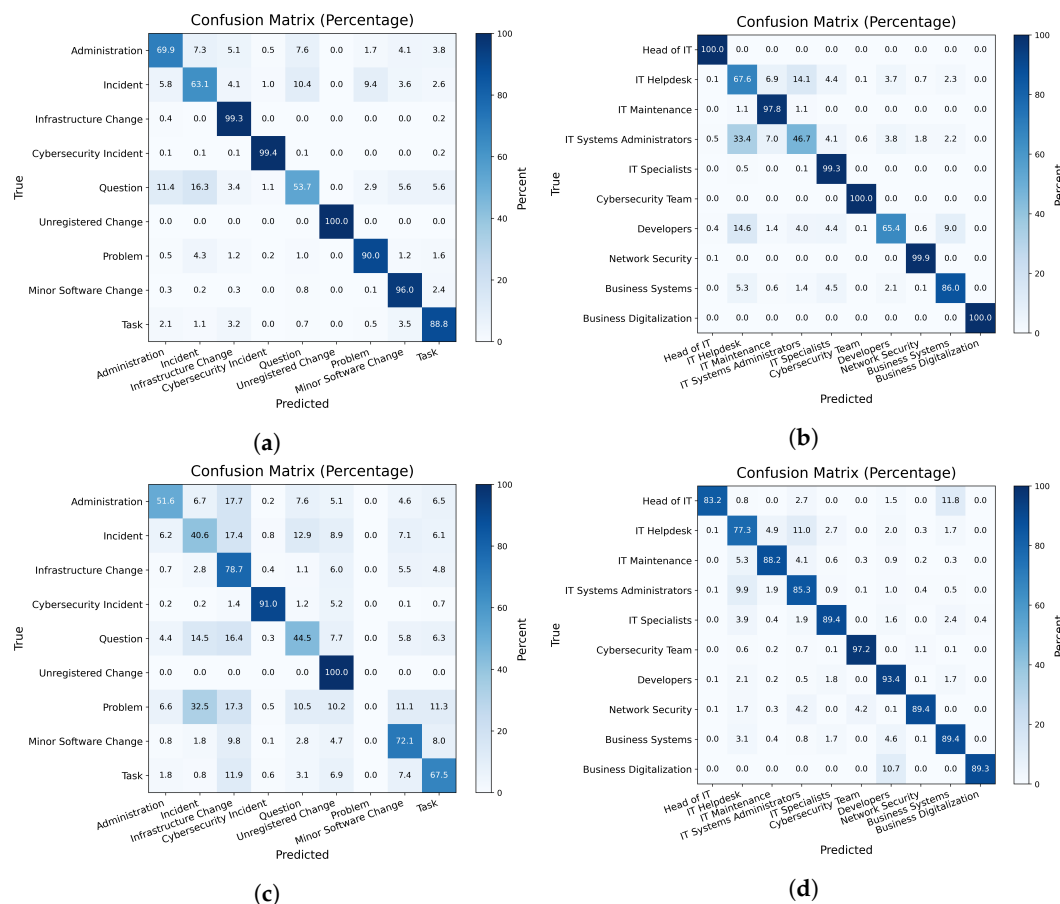


Figure 7. Logistic Regression model F1-score comparison on the test set. (a) Classification performance by request type with balanced data. (b) Classification performance by the responsible team with balanced data. (c) Classification performance by request type with unbalanced data. (d) Classification performance by the responsible team with unbalanced data.

Although such misclassifications could be reduced by additional data preprocessing or by employing more advanced semantic models, the confusion matrix shows that most teams were correctly identified by the model. Overall, Logistic Regression was able to classify the majority of frequent classes accurately, though testing highlighted its limited ability to handle rare or semantically overlapping cases. This positions the model as a potential baseline solution, but with restricted applicability in contexts that demand deeper semantic understanding.

The results in Table 8 for Stage 2 show that the model was able to classify the majority of teams with high accuracy. The Cybersecurity, Network Security, and Business Digitalization Department teams achieved particularly good results, with both precision and recall exceeding 0.98, and F1-scores approaching the maximum value. These outcomes suggest that the model is highly effective at recognizing teams with clearly defined, semantically distinctive characteristics. Similarly, the IT Specialists team achieved an impressive F1-score of 0.97.

By contrast, performance for the IT System Administrators and Developers teams declined by roughly 10 percentage points, indicating that although these teams were frequently predicted, a notable proportion of the predictions were incorrect. This can be

explained by the functional overlap of these classes with IT Helpdesk and IT Maintenance, making it more difficult for the model to reliably distinguish their features.

Table 8. Logistic Regression training results by the responsible team.

Responsible Type	Unbalanced				Balanced			
	Precision	Recall	F1-Score	Support	Precision	Recall	F1-Score	Support
Head of IT	0.94	0.83	0.88	262	0.89	1.00	0.94	262
IT Helpdesk	0.86	0.77	0.82	12,145	0.88	0.68	0.77	12,145
IT Maintenance	0.93	0.88	0.91	11,204	0.92	0.98	0.95	11,204
IT Systems Administrators	0.39	0.85	0.53	1739	0.30	0.47	0.36	1739
IT Specialists	0.96	0.89	0.92	12,327	0.94	0.99	0.97	12,327
Cybersecurity Team	0.90	0.97	0.94	2990	0.99	1.00	1.00	2990
Developers	0.63	0.93	0.76	1264	0.59	0.65	0.62	1264
Network Security	0.98	0.89	0.94	6436	0.98	1.00	0.99	6436
Business Systems	0.79	0.89	0.84	2531	0.83	0.86	0.85	2531
Business Digitalization	0.29	0.89	0.44	28	1.00	1.00	1.00	28
Macro average	0.77	0.88	0.80	50,926	0.83	0.86	0.84	50,926
Weighted average	0.89	0.87	0.87	50,926	0.89	0.88	0.88	50,926

Overall, the model achieved an F1-score of 0.88 on the balanced dataset and 0.87 on the imbalanced dataset, demonstrating robust classification performance across both frequent and less frequent teams. However, it is worth noting that these results were obtained using the validation set, which, due to oversampling, may have contained an increased number of highly similar records. As such, the reported metrics may be somewhat optimistic, and the true generalization capability of the model can only be reliably assessed on a completely isolated test set.

4.3. BERT Classification Results

Table 9 presents the BERT model classification results by request type evaluated on the test dataset. Although all metrics remained consistently high across training iterations, the most significant outcome was observed at the third epoch. At this point, the validation loss reached its lowest value (0.240), while the classification quality was particularly high, with precision = 0.920, recall = 0.922, and F1-score = 0.920. From the fourth epoch onward, a slight increase in validation loss was detected, suggesting the onset of overfitting. Therefore, the third epoch can be considered optimal, as the model achieved the best balance between training accuracy and generalization to unseen data.

Table 9. BERT fine-tuning results on the test set.

Epoch	By Request Type						By Team					
	Train	Val	Acc	Prec	Rec	F1 (Req)	Train	Val	Acc	Prec	Rec	F1 (Team)
1	0.312	0.368	0.880	0.875	0.880	0.872	0.253	0.300	0.899	0.902	0.899	0.895
2	0.276	0.252	0.917	0.911	0.917	0.912	0.187	0.189	0.938	0.938	0.938	0.937
3	0.217	0.240	0.922	0.920	0.922	0.920	0.145	0.171	0.953	0.952	0.953	0.952
4	0.210	0.247	0.923	0.919	0.923	0.919	0.120	0.185	0.952	0.950	0.952	0.951
5	0.156	0.254	0.928	0.924	0.928	0.925	0.105	0.202	0.948	0.947	0.948	0.948

Table 9 also shows the BERT results for team classification, highlighting rapid and stable learning within the first three epochs. Validation loss dropped sharply from 0.300 to 0.171, indicating that the model quickly learned to separate team features despite overlaps (e.g., IT Maintenance vs. IT Helpdesk). Optimal performance was achieved at the third

epoch, with precision = 0.952, recall = 0.953, and F1 = 0.952. While metrics remained stable in later epochs, the slight rise in validation loss (0.202) suggests early signs of overfitting. Compared to request-type classification, adaptation here was faster, likely because team categories are more distinct and semantically clearer.

The BERT model confusion matrices (see Figure 8) demonstrate consistently strong classification performance across both request types and responsible teams. In the case of request types (see Figure 8c), most categories achieve near-perfect separation, with classes such as Minor Software Change, Task, Cybersecurity Incident, Infrastructure Change, and Problem reaching almost 100% correct classification. The most frequent categories, Administration and Incident, also show very high accuracy (above 89% and 93% respectively), although partial misclassification between these two semantically related classes is visible. The greatest challenge arises with the Question class, where only around half of the cases (51.3%) are correctly identified, while a significant share is confused with Incident (32.7%) or Administration (11.8%), suggesting semantic overlap and ambiguity in user phrasing.

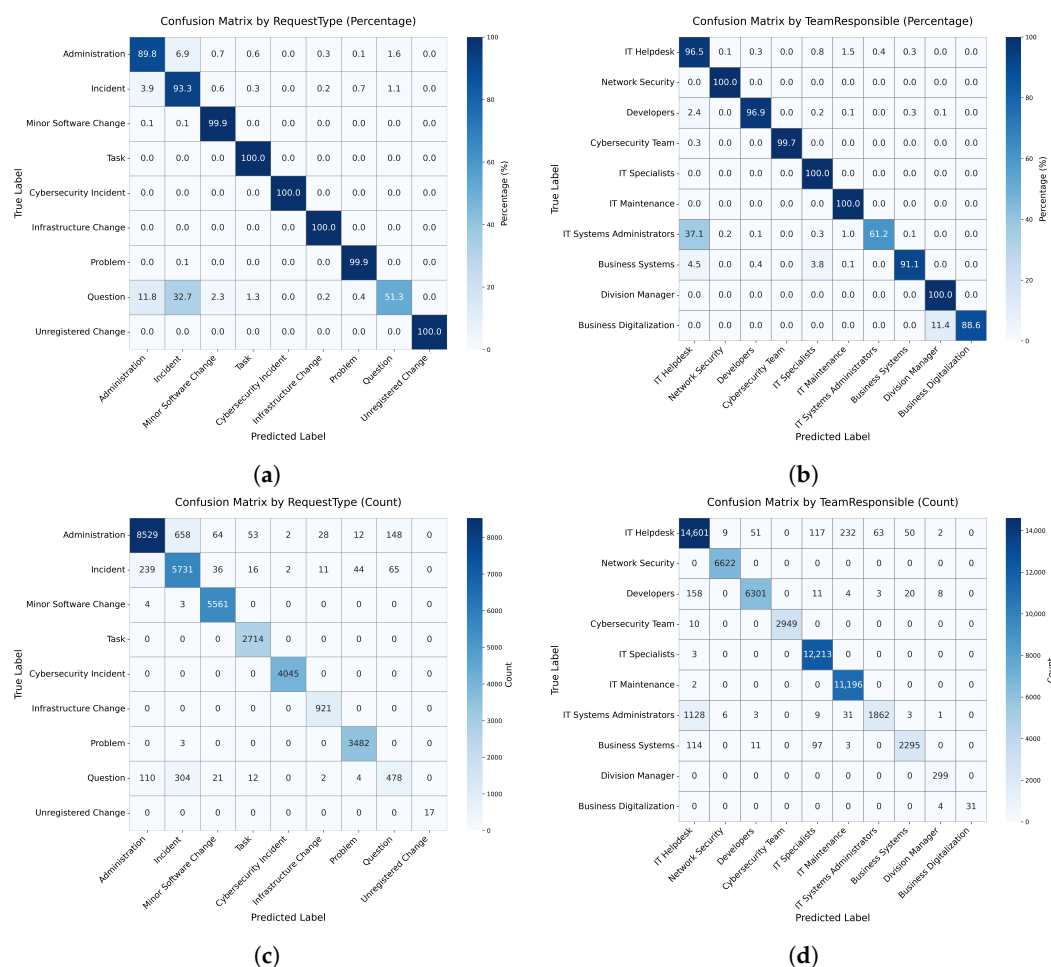


Figure 8. BERT model confusion matrix in percentages and counts by request type and responsible team. (a) Classification performance by request type with balanced data in percentage. (b) Classification performance by the responsible team with balanced data in percentage. (c) Classification performance by request type with balanced data count. (d) Classification performance by the responsible team with balanced data count.

Similarly, classification by the responsible team (see Figure 8d) confirms the robustness of the model. Teams with well-defined functions, such as Network Security, Cybersecurity Team, IT Specialists, and IT Maintenance, achieve almost flawless predictions ($\geq 99\%$). Broader or semantically overlapping roles, however, introduce more confusion: for example,

IT Systems Administrators are sometimes classified as IT Helpdesk (37.1%), indicating functional similarity in descriptions. Business Systems also show occasional confusion with Developers (4.5%), hinting at intersecting responsibilities. Despite these overlaps, the overall performance remains high-level across all categories, including low-frequency teams such as Division Manager (Head of IT) and Business Digitalization, which are identified with high confidence.

The verification with the fully isolated 2025 dataset provides a robust assessment of both models in handling previously unseen queries. The F1-score comparison (see Figures 9 and 10) highlights the consistent advantage of the BERT classifier over LR across most classes. In request type classification, BERT improved the F1-score by +17% for Administration, +27% for Minor Software Change, and +166% for Problem, demonstrating its ability to significantly enhance recognition of categories that were challenging for the baseline. However, both models struggled with Infrastructure Change and Unregistered Change, suggesting systematic limitations linked to data scarcity or unclear class definitions.

When evaluated by the responsible team, BERT again outperformed LR in nearly all cases. Notable improvements were achieved for Network Security (+62%) and IT Maintenance (+45%), while high absolute performance was observed for Developers (F1 = 0.89) and Cybersecurity Team (F1 = 0.89). The only negative outcome was for IT Systems Administrators, where BERT underperformed compared to LR (−20%), reflecting its sensitivity to semantic overlap with functionally similar teams such as IT Helpdesk.

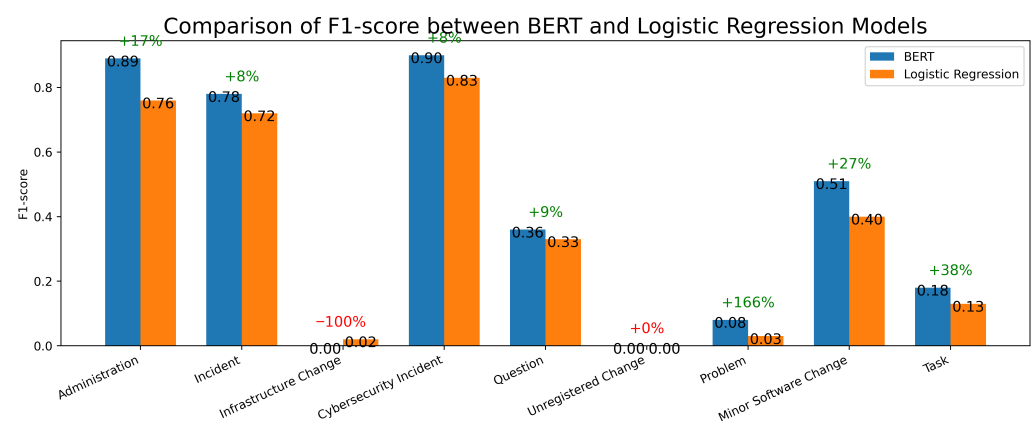


Figure 9. F1-score comparison for classification by request type.

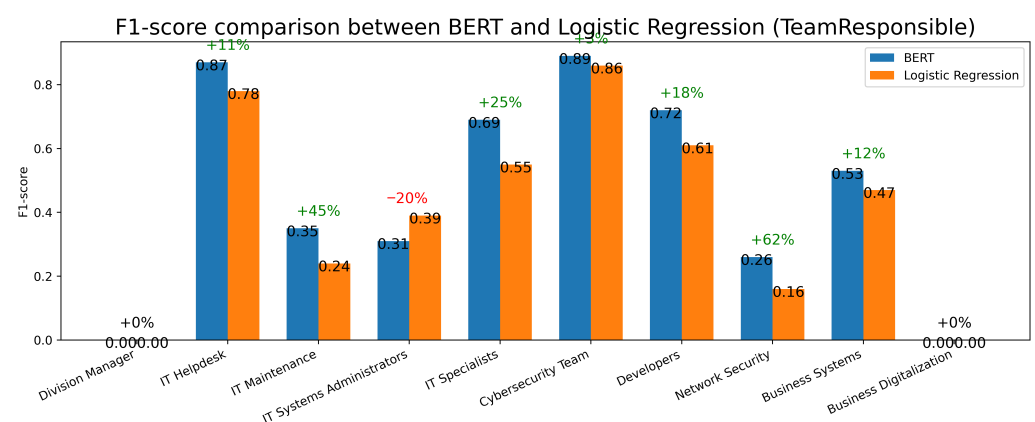


Figure 10. F1-score comparison for classification by the responsible team.

4.4. Model Interpretability Analysis (LIME, SHAP)

To enhance interpretability and transparency of the final models, we generated Local Interpretable Model-agnostic Explanations (LIME) for representative multilingual samples in both Lithuanian and English (Figure 11). The visualizations illustrate token-level contributions for the final, balanced models. In the LR baseline, term influence is relatively uniform and dominated by literal keyword frequency (e.g., “VPN”, “navision”, “neveikia”), whereas the BERT model exhibits stronger, semantically contextual weighting—emphasising failure- or connectivity-related expressions while suppressing stopwords and grammatical connectors. This comparison illustrates how transformer-based models capture more nuanced semantic relations across languages, thereby enhancing interpretability and trust in multilingual ITSM automation systems.

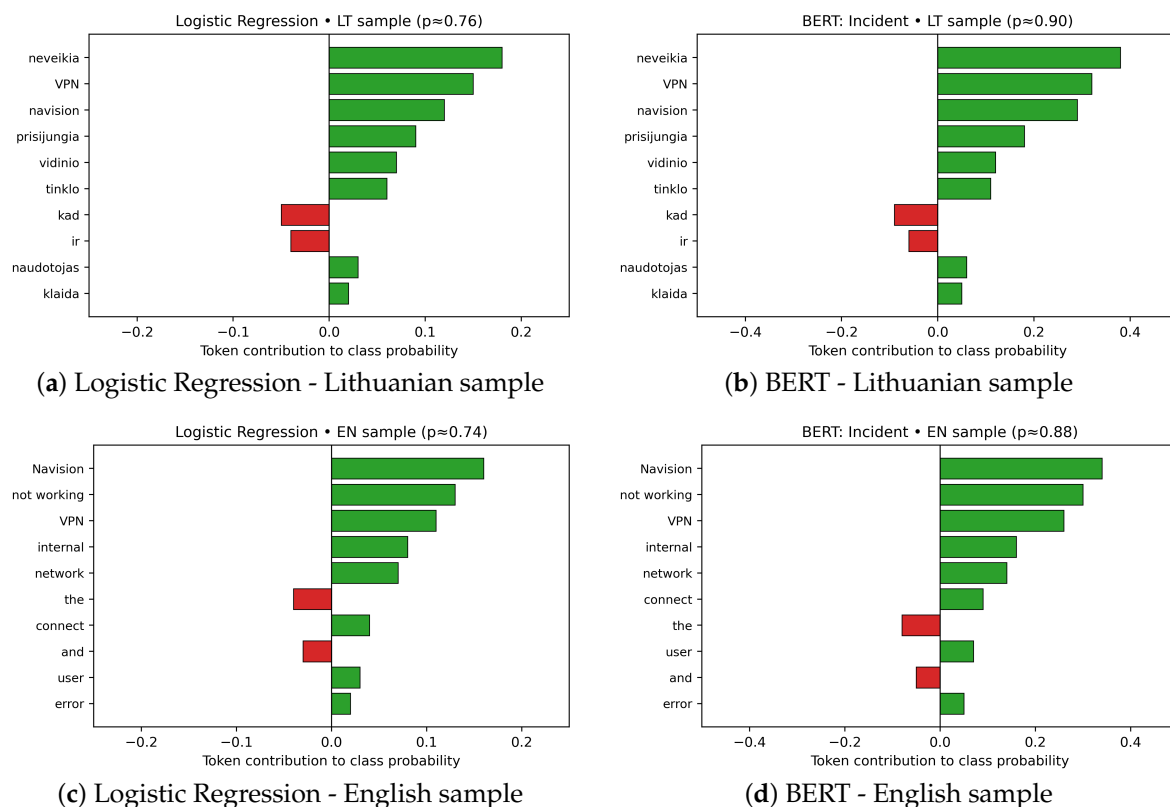


Figure 11. LIME token-level explanations for multilingual IT support ticket samples in Lithuanian and English. Green tokens increase the predicted probability of the shown class Incident, while red tokens decrease it. LR exhibits flatter, keyword-based contributions, whereas BERT assigns stronger contextual importance to domain-specific terms, thereby improving semantic interpretability.

Figure 12 presents SHAP summary plots illustrating the most influential textual features contributing to the predictions of the LR classifier for predicting the Incident and Problem classes. Features are ranked according to their mean absolute SHAP values, indicating their overall contribution to the model output. For the Incident class, terms such as “neveikia”, “reikia”, and “pakeisti” exhibit the strongest impact, reflecting language commonly associated with service disruptions and incident reports. In contrast, the Problem class is characterized by a broader set of issue-related keywords, including “problema” and system-specific terms, suggesting more general or less urgent technical difficulties.

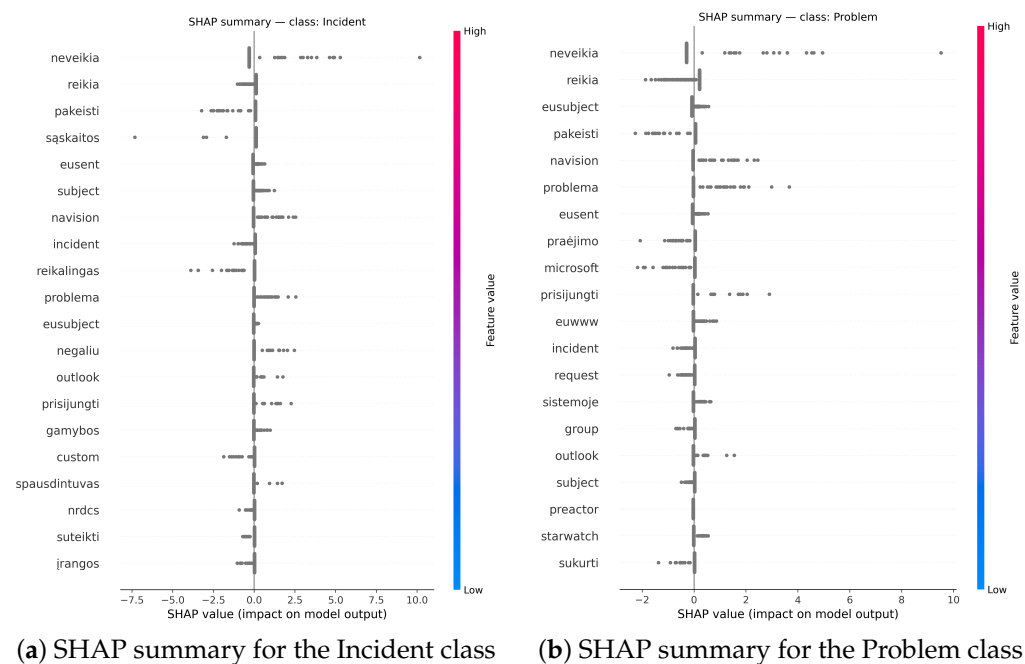


Figure 12. SHAP summary plots showing the most influential words for predicting the Incident and Problem classes, ranked by mean absolute SHAP value.

4.5. Validation of Results

The validation of the proposed ITSM request classification models was designed to ensure methodological rigor, realistic evaluation conditions, and transparent assessment of computational efficiency. In this study, validation is performed at the first stage of the ITSM automation pipeline, namely request type classification, which constitutes the foundation for subsequent downstream processes such as team assignment and workflow routing. A steady and reliable evaluation of this initial stage is, therefore, critical for assessing the practical applicability of the proposed approach.

To support this objective, the validation framework explicitly distinguishes between diagnostic model analysis and deployment-oriented evaluation, as illustrated in Figure 6. This separation enables controlled investigation of model behavior under alternative training conditions while preserving the natural characteristics of operational ITSM data during final performance assessment.

All experiments were conducted using a single, fixed stratified train–test split derived from the cleaned ITSM dataset. This split was preserved across all evaluated models, including LR, multilingual BERT (mBERT), and XLM-RoBERTa (XLM-R), to ensure direct and fair comparability of results. XLM-R was included as an additional strong baseline alongside multilingual BERT. The test set retained the original, highly imbalanced class distribution characteristic of real-world ITSM environments. Any class-balancing procedures, when applied, were restricted exclusively to the training data and were never used during test-time evaluation. This design prevents information leakage and ensures that reported performance metrics accurately reflect realistic operational conditions.

Two complementary evaluation regimes were considered. First, models were trained and evaluated without any class balancing, representing a deployment-oriented scenario in which models operate directly on naturally imbalanced ITSM data. Second, a diagnostic regime employing training-time random oversampling was used to analyze the impact of improved minority-class representation on model behavior. Results obtained under the latter regime are reported separately and interpreted as diagnostic evidence rather than as primary indicators of deployment performance.

In addition to predictive quality, validation also incorporates a quantitative assessment of computational efficiency, including training time, inference latency, model size, and GPU utilization. These measurements provide an explicit comparison of the operational cost associated with each modeling approach and support informed trade-off analysis between predictive performance and deployment feasibility in real-world ITSM settings. A quantitative comparison of these factors is provided in Table 10.

The results reveal substantial differences in computational complexity between traditional machine-learning and transformer-based approaches. The TF-IDF with LR baseline exhibits minimal training and inference cost, a compact model footprint, and negligible GPU utilization, making it highly efficient from an operational perspective. In contrast, transformer-based models require substantially higher computational resources due to their deep architectures and self-attention mechanisms, with XLM-R representing the most resource-intensive configuration in terms of training time, memory footprint, and energy consumption.

Computational efficiency metrics were obtained using a unified evaluation protocol. Training time was measured as the total time required to complete the full training procedure for each model under identical hardware conditions. Average GPU utilization, temperature, and power consumption were monitored throughout training using continuous system-level logging based on the nvidia-smi interface and aggregated over the complete training duration. Model size and parameter counts were determined from the stored model checkpoints on disk. Inference latency was evaluated on the unbalanced test set by measuring forward-pass execution time under fixed batch size and maximum sequence length, reporting both mean and 95th-percentile latency to capture typical and worst-case behavior.

Table 10. Computational efficiency and overhead comparison of LR and transformer models.

Model	Train Time (min)	Inference Mean (ms)	Inference p95 (ms)	Params	Model Size (MB)	Avg. GPU Usage (%)	Avg. GPU Temp. (°C)	Avg. GPU Power (W)
No oversampling								
LR TF-IDF	10	0.59	0.81	90,000	16	5.2	46.5	22.9
BERT	86	8.34	9.51	177,860,361	682	88.9	85.2	324.9
RoBERTa	93	8.46	9.66	278,050,569	1077	89.9	85.3	315.6
RandomOverSampler (train-only)								
LR TF-IDF	11	0.59	0.83	90,000	17	11	47	23.2
BERT	108	8.11	10.24	177,860,361	682	99.3	66	171.5
RoBERTa	117	16.47	32.73	278,050,569	1077	99.5	67	173.4

Despite the substantial increase in training costs, inference latency for all evaluated models remains within acceptable bounds for offline or near-real-time ITSM decision-support scenarios. Transformer-based models achieve inference times on the order of several milliseconds per request, which is compatible with batch-oriented processing and delayed-response workflows commonly employed in enterprise ITSM systems. Importantly, applying training-time oversampling increases training overhead but does not affect inference latency or model size, indicating that class-balancing strategies primarily affect model development costs rather than deployment-time efficiency.

Model performance was primarily assessed using the macro-averaged F1-score, which assigns equal importance to all request types and is therefore well-suited for highly imbalanced multi-class classification problems. Weighted F1-score and overall accuracy were reported as complementary metrics to reflect expected performance under the natural class distribution encountered in operational ITSM environments.

Figure 13 presents a comparative performance analysis of the evaluated models on the unbalanced test set. Transformer-based models consistently outperform the TF-IDF with LR baseline in terms of macro F1-score, confirming their superior ability to recognize under-represented request types. In particular, the macro F1-score increases from approximately 0.39 for the TF-IDF baseline to 0.49–0.51 for the transformer-based models, while accuracy and weighted F1-score remain within a narrow range (approximately 0.82–0.84 across all models). This indicates that improvements in minority-class performance are achieved without compromising overall classification reliability.

Additional diagnostic analysis using training-time random oversampling further illustrates this effect. Although oversampling does not lead to higher accuracy on the unbalanced test set, it consistently improves macro F1-score across all evaluated models. For example, macro F1-score increases from 0.39 to 0.41 for the TF-IDF baseline and from 0.49 to 0.51 for mBERT, reflecting enhanced sensitivity to rare request categories. Weighted F1-score remains largely unchanged, confirming that performance on dominant request types is preserved.

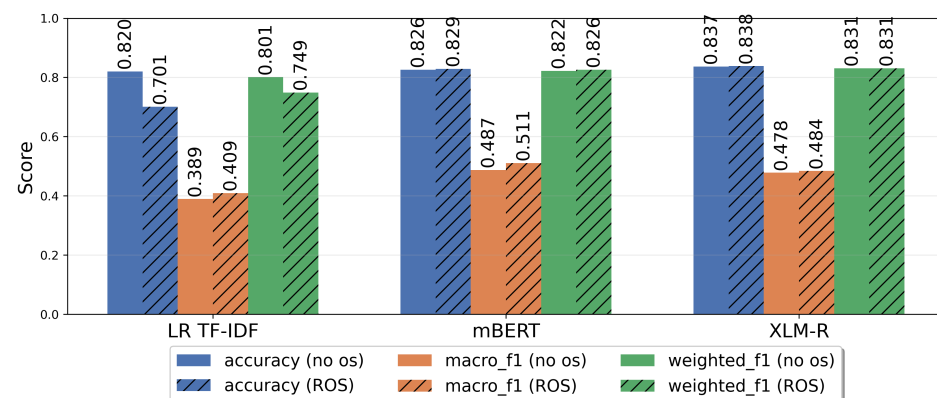


Figure 13. Performance comparison: No oversampling (no os) vs. RandomOverSampler (ROS).

Per-class evaluation provides further insight into these trends. For frequent request types such as Administration and Incident, all models achieve high and stable F1-scores regardless of the training regime. In contrast, low-frequency classes such as Question and Task exhibit notable gains in recall and F1-score under the oversampled regime, particularly for transformer-based models. These improvements explain the observed increase in macro F1-score and demonstrate that training-time oversampling primarily benefits underrepresented classes, which are often of practical importance in real-world ITSM workflows.

The combined evaluation of predictive performance and computational overhead demonstrates that transformer-based models provide a measurable improvement in classification quality for ITSM request types, particularly for rare but operationally important categories. This improvement is accompanied by increased computational cost, highlighting an inherent trade-off between model complexity and efficiency. Consequently, lightweight models remain suitable for resource-constrained or high-throughput environments, whereas transformer-based approaches are better aligned with scenarios prioritizing classification accuracy and decision-support effectiveness.

4.6. Summary of Findings

The conducted experiments provide a comprehensive evaluation of ITSM request classification models under both diagnostic and deployment-oriented validation regimes. Across all experimental settings, transformer-based models consistently demonstrate stronger generalization capabilities than the TF-IDF with LR baseline, particularly for

semantically diverse and low-frequency request types. This behavior is observed both in the original diagnostic experiments on unseen 2025 data and in the extended validation on a fixed, naturally imbalanced test set.

In addition, a targeted ablation experiment was conducted to examine the sensitivity of the proposed approach to key design choices. In particular, the effect of training-time oversampling and maximum input sequence length was evaluated for request-type classification. Removing oversampling reduced macro-averaged F1-score (e.g., from approximately 0.49 to 0.46 for mBERT), while overall accuracy remained stable at around 0.82, indicating that oversampling primarily affects minority-class recognition. Increasing the maximum sequence length from 128 to 256 tokens resulted in very similar performance (accuracy approx. 0.82, macro F1 approx. 0.46), while substantially increasing training and evaluation time. These results indicate diminishing returns for longer input contexts in ITSM ticket classification.

The results confirm that mBERT and XLM-R are more effective at capturing contextual information and recognizing underrepresented request categories, whereas LR remains competitive primarily for frequent classes characterized by strong lexical similarity. Training-time random oversampling improves sensitivity to minority classes and leads to consistent gains in macro-averaged F1-score, while overall accuracy and weighted F1-score remain largely unchanged. This outcome is expected in imbalanced multi-class settings and indicates that improvements in class-balanced performance are achieved without degrading reliability on dominant request types. Per-class analysis further shows that the most pronounced gains occur in rare categories, supporting the use of oversampling as a diagnostic tool rather than as a deployment-time strategy.

From a computational perspective, the results highlight a clear trade-off between efficiency and predictive capability. The TF-IDF baseline offers minimal training cost, compact model size, and extremely low inference latency, making it suitable for resource-constrained environments. Transformer-based models incur substantially higher training overhead and memory requirements, with XLM-R being the most resource-intensive; however, inference latency for all evaluated models remains within acceptable bounds for offline or near-real-time ITSM decision-support applications. Overall, the findings indicate that transformer-based models provide a more balanced and semantically robust solution for ITSM request classification in imbalanced settings, while LR remains a viable alternative when computational efficiency is the primary constraint.

5. Discussion

The preparatory analysis revealed a strong class imbalance in both request types and responsible teams. For instance, Administration requests accounted for 28.6% of all tickets, and Incidents accounted for 18.8%. In contrast, Infrastructure Change and Unregistered Change comprised only 2.7% and 0.05% respectively. A similar skew was observed at the team level, with IT Helpdesk handling 25.1% of all cases compared to only 0.05% for Business Digitalization. Such an imbalance risks producing models biased toward dominant classes while ignoring rare but practically important categories.

To mitigate this effect, we applied a threshold-based oversampling strategy that equalized major and minor classes. This balancing step proved crucial: without it, LR consistently favored high-frequency classes, and even BERT, despite its contextual learning capacity, underperformed on underrepresented categories. By exposing the models to a more uniform distribution, balancing enabled recognition of infrequent patterns and allowed BERT to fully exploit its semantic representations. In this study, oversampling was applied prior to splitting only to guarantee the presence of extremely low-frequency classes in the training data. Several categories contained so few samples that a standard

split would have eliminated them entirely from the training set, preventing any meaningful learning. To mitigate the risk of inflated metrics and validate generalization, we conducted a separate, independent evaluation on an isolated dataset from a subsequent year (2025) that was neither balanced nor preprocessed. This dataset reflects the true operational imbalance of the production ITSM environment.

The comparative evaluation (see Table 11) confirms that balancing combined with contextual modeling yields tangible gains. For request types, BERT achieved the strongest improvements in high-volume classes such as Administration (+17%) and Incident (+8%), and even mid-frequency categories such as Minor Software Change improved substantially (+27%). At the same time, performance remained limited to such semantically vague classes as Question and Problem, where boundaries are inherently blurred. This suggests that data quantity alone cannot resolve conceptual ambiguity and points to the need for refined annotation or additional rule-based constraints.

Table 11. Comparison of BERT and LR results on the isolated (unseen) 2025 test set.

Task	Model	Accuracy	Recall	F1-Score
Request Type	BERT	0.80	0.79	0.79
	Logistic Regression	0.81	0.62	0.69
Responsible Team	BERT	0.75	0.76	0.75
	Logistic Regression	0.71	0.66	0.68

Team-level classification showed a similar pattern. BERT achieved high accuracy for well-defined, frequent teams, with Developers and Cybersecurity each reaching F1-scores of 0.89, while Network Security gained +62% due to oversampling. However, categories with overlapping roles, such as IT Systems Administrators, saw a 21% decline compared to LR. This indicates that transformer-based models are highly effective when semantic boundaries are clear, but may become confused when team responsibilities are functionally similar.

Taken together, these findings demonstrate that class imbalance plays a critical role in the evaluation of ITSM request classification models and must be addressed carefully depending on the intended usage scenario. Training-time class balancing improves model sensitivity to rare request categories and enables more balanced classification behavior, particularly for transformer-based models with strong contextual representations. At the same time, the results confirm that such balancing primarily serves as a diagnostic mechanism rather than a direct means of improving deployment-time accuracy on naturally imbalanced data. Persistent weaknesses in semantically overlapping or weakly defined classes further underline the need for clearer labeling schemes and, potentially, hybrid solutions that combine contextual language models with explicit business rules. This aligns with ITSM principles, where process standardization and continuous improvement are central to operational governance frameworks such as ITIL 4 and COBIT.

From a computational perspective, the extended evaluation highlights a clear trade-off between efficiency and predictive capability. The applied oversampling strategy increases training-time overhead by enlarging the effective dataset size, but its computational cost grows linearly and remains feasible for real-world ITSM environments. LR with TF-IDF features remains highly efficient, offering minimal training cost, compact model size, and extremely low inference latency, which makes it suitable for rapid prototyping and resource-constrained deployments. Transformer-based models, while substantially more demanding in terms of training time, memory footprint, and energy consumption, provide improved recognition of underrepresented and semantically complex request types. Importantly,

inference latency for all evaluated models remains within acceptable bounds for offline or near-real-time decision-support scenarios commonly employed in enterprise ITSM systems.

From an ablation perspective, these findings clarify the role of individual design choices within the proposed pipeline. Training-time oversampling improves macro-averaged performance by increasing exposure to rare request categories, but does not significantly alter accuracy under naturally imbalanced test conditions. Similarly, extending the input sequence length beyond 128 tokens yields negligible gains, suggesting that ITSM tickets typically contain sufficient discriminative information in their initial segments. Together, these observations indicate that performance improvements stem primarily from class exposure and semantic modeling rather than from longer textual context.

In the conducted experiments, all models were trained offline, and inference performance was evaluated under realistic operational constraints. The measured computational metrics indicate that transformer-based models can be deployed in practice when training is performed infrequently on GPU-enabled infrastructure, while inference is executed as part of batch-oriented or delayed-response workflows. Although the present study does not explore performance scalability under substantially larger data volumes or real-time retraining conditions, such analysis represents a natural direction for future work aimed at assessing long-term robustness in evolving ITSM environments.

In practical terms, the trained models can be integrated directly into an ITSM platform as an automated triage and recommendation component, continuously classifying incoming tickets and supporting routing decisions made by human agents. The prototype developed in this study has already been validated on historical enterprise data and demonstrates full compatibility with existing help-desk workflows, confirming its readiness for operational use. While the focus of this work is on ITSM data, the proposed methodology and validation framework are applicable to other service-oriented and multilingual domains, such as healthcare, public administration, logistics, and financial support centers, where automated text classification plays a key role in maintaining service quality.

6. Conclusions

This study investigated automated ITSM request classification using real-world enterprise ticket data characterized by multilingual content and severe class imbalance. The proposed ITSM pipeline consists of request type classification (Stage 1) followed by responsible team assignment (Stage 2). In the diagnostic evaluation, a targeted class-balancing strategy based on minority-sample duplication was applied across both stages, demonstrating the strong influence of imbalance on model behavior: for request-type classification, LR accuracy increased from 0.54 on raw data to 0.81 after balancing, while responsible team assignment achieved 0.88 accuracy on the balanced dataset.

Building on these findings, the present work extends the analysis of Stage 1 through a deployment-oriented evaluation that preserves the naturally imbalanced class distribution of operational ITSM data. Within this setting, a TF-IDF with LR baseline was compared against transformer-based models, including mBERT and XLM-R, using a fixed stratified train-test split. This evaluation framework enables a realistic assessment of operational performance while maintaining direct comparability across models.

Across the held-out 2025 test set, the BERT-based model consistently outperformed the Logistic Regression baseline, particularly for semantically complex and underrepresented categories. The most pronounced improvements were observed in responsible team assignment for Network Security requests (+62% F1-score) and Administration-related tasks (+17%), with additional gains for mid-frequency classes such as IT Helpdesk (+11%) and Minor Software Change (+27%). These results indicate that contextual language models provide substantial benefits when class boundaries are less distinct or when request

descriptions require deeper semantic interpretation. LR remained competitive for frequent and structurally well-defined categories but showed clear limitations in cases involving overlapping semantics or sparse class representation.

From an operational perspective, the extended evaluation highlights a clear trade-off between computational efficiency and predictive capability at Stage 1 of the pipeline. The TF-IDF baseline exhibits minimal training cost and sub-millisecond inference latency, whereas transformer-based models incur substantially higher training overhead and memory requirements, with XLM-R being the most resource-intensive configuration. Nevertheless, inference latency for transformer-based models remains within 8–10 ms per request, which is acceptable for offline or near-real-time ITSM decision-support workflows. Given that request-type classification is typically performed prior to routing decisions and model training is conducted infrequently, these results support the practical feasibility of deploying transformer-based classifiers in enterprise ITSM systems.

The ablation analysis further demonstrates that the proposed approach is robust to moderate variations in design parameters. In particular, training-time oversampling primarily enhances minority-class recognition without substantially affecting deployment-time accuracy, while increasing the maximum sequence length beyond 128 tokens provides minimal benefit at significantly higher computational cost. These findings support the selected configuration as a balanced trade-off between performance, efficiency, and practical deployability in real-world ITSM environments.

Several limitations should be acknowledged. The extended validation focuses exclusively on request-type classification (Stage 1), while the responsible team assignment stage remains based on prior experimental work and was not re-evaluated under the proposed validation framework. In addition, the analysis relies on data from a single enterprise environment, which may limit generalization across organizations or language distributions. Although training-time oversampling improves minority-class visibility, it relies on sample duplication and may introduce overfitting effects. Future work should therefore extend the proposed validation framework to downstream pipeline stages, investigate alternative imbalance-handling strategies such as semantic-aware augmentation or cost-sensitive learning, and explore cross-organizational validation and human-in-the-loop decision-support pipelines to further strengthen the robustness and applicability of AI-assisted ITSM workflows.

Author Contributions: A.R.: Carried out the experiment and worked on the methodology; data presentation; investigation; article writing and editing. R.J.: Supervised the project and conceptualization; article reviewing and validating; article editor. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset was obtained from a Lithuanian-owned enterprise with over 30 years of operational history in the food manufacturing sector. The company operates internationally, supplying products to dozens of countries and employing more than 8000 staff across multiple facilities. The IT Service Management system supports internal operations across multilingual environments. Ticket texts were primarily written in Lithuanian (70%) and English (23%), and mixed or undefined languages (7%). This multilingual composition reflects realistic enterprise ITSM communication patterns and motivates the use of multilingual language models. The dataset used in this study originates from a company's internal ITSM system and contains confidential information. Due to company privacy policies, the full dataset cannot be made publicly available. A depersonalized sample and data structure example, together with the complete source

code and model training pipeline, are available in the public GitHub repository at <https://github.com/RazmaAudrius/BertTraining> (accessed on 6 August 2025).

Acknowledgments: This research was supported by the Lithuanian Research Council and the Ministry of Education, Science, and Sports of the Republic of Lithuania (Project No. S-A-UEI-23-9). The authors express their sincere gratitude to Zuzana Šiušaitė for professional English language editing and stylistic refinement of the manuscript. She is an experienced translator and editor of academic books and doctoral theses. The authors take full responsibility for the final content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
COBIT	Control Objectives for Information and Related Technologies
IT	Information Technology
ITIL	Information Technology Infrastructure Library
ITSM	IT Service Management
LR	Logistic Regression
NLP	Natural Language Processing
TF-IDF	Term Frequency–Inverse Document Frequency

References

1. Limited, A. *ITIL® Foundation, ITIL® 4 Edition*, 1st ed.; TSO (The Stationery Office): Norwich, UK, 2019; p. 212.
2. ISACA. *COBIT® 2019 Framework: Governance and Management Objectives*, 1st ed.; ISACA: Schaumburg, IL, USA, 2018; p. 302.
3. *ISO/IEC 20000-1*; 2018 Information Technology—Service Management. International Organization for Standardization: Geneva, Switzerland, 2018.
4. World Economic Forum. The Future of Jobs Report 2025. 2025. Available online: <https://www.weforum.org/publications/the-future-of-jobs-report-2025/in-full/> (accessed on 10 December 2025).
5. Sarker, I.H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Comput. Sci.* **2021**, *2*, 160. <https://doi.org/10.1007/s42979-021-00592-x>.
6. Wahyuningsih, T.; Manongga, D.; Sembiring, I.; Wijono, S. Comparison of Effectiveness of Logistic Regression, Naive Bayes, and Random Forest Algorithms in Predicting Student Arguments. *Procedia Comput. Sci.* **2024**, *234*, 349–356. <https://doi.org/10.1016/j.procs.2024.03.014>.
7. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, New York, NY, USA, 2016; pp. 785–794. <https://doi.org/10.1145/2939672.2939785>.
8. Wang, S.; Dai, Y.; Shen, J.; Xuan, J. Research on expansion and classification of imbalanced data based on SMOTE algorithm. *Sci. Rep.* **2021**, *11*, 24039. <https://doi.org/10.1038/s41598-021-03430-5>.
9. Yang, X.; Yang, K.; Cui, T.; Chen, M.; He, L. A Study of Text Vectorization Method Combining Topic Model and Transfer Learning. *Processes* **2022**, *10*, 350. <https://doi.org/10.3390/pr10020350>.
10. Revina, A.; Buza, K.; Meister, V.G. IT Ticket Classification: The Simpler, the Better. *IEEE Access* **2020**, *8*, 193380–193395. <https://doi.org/10.1109/ACCESS.2020.3032840>.
11. Cahyani, D.; Patasik, I. Performance comparison of TF-IDF and Word2Vec models for emotion text classification. *Bull. Electr. Eng. Inform.* **2021**, *10*, 2780–2788. <https://doi.org/10.11591/eei.v10i5.3157>.
12. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the North American Chapter of the Association for Computational Linguistics, Minneapolis, MN, USA, 2–7 June 2019.
13. Naseer, M.; Asvial, M.; Sari, R.F. An Empirical Comparison of BERT, RoBERTa, and Electra for Fact Verification. In Proceedings of the 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), Jeju Island, Republic of Korea, 13–16 April 2021; pp. 241–246. <https://doi.org/10.1109/ICAIIIC51459.2021.9415192>.

14. Gonzalez-Canas, C.; Valencia-Zapata, G.A.; Estrada Gomez, A.M.; Hass, Z. Assessing the impact on quality of prediction and inference from balancing in multilevel logistic regression. *Healthc. Anal.* **2024**, *6*, 100359. <https://doi.org/10.1016/j.health.2024.100359>.
15. Hayaean Shirvan, M.; Moattar, M.H.; Hosseinzadeh, M. Deep generative approaches for oversampling in imbalanced data classification problems: A comprehensive review and comparative analysis. *Appl. Soft Comput.* **2025**, *170*, 112677. <https://doi.org/10.1016/j.asoc.2024.112677>.
16. Dou, J.; Song, Y.; Wei, G.; Guo, X. A robust ensemble classifier for imbalanced data via adaptive variety oversampling and embedded sampling rate. *Appl. Soft Comput.* **2025**, *174*, 112922. <https://doi.org/10.1016/j.asoc.2025.112922>.
17. Nasreen, G.; Murad Khan, M.; Younus, M.; Zafar, B.; Kashif Hanif, M. Email spam detection by deep learning models using novel feature selection technique and BERT. *Egypt. Inform. J.* **2024**, *26*, 100473. <https://doi.org/10.1016/j.eij.2024.100473>.
18. Gupta, B.B.; Gaurav, A.; Arya, V.; Attar, R.W.; Bansal, S.; Alhomoud, A.; Chui, K.T. Advanced BERT and CNN-Based Computational Model for Phishing Detection in Enterprise Systems. *CMES-Comput. Model. Eng. Sci.* **2024**, *141*, 2165–2183. <https://doi.org/10.32604/cmes.2024.056473>.
19. Hussain, A.; Saadia, A.; Alserhani, F.M. Ransomware detection and family classification using fine-tuned BERT and RoBERTa models. *Egypt. Inform. J.* **2025**, *30*, 100645. <https://doi.org/10.1016/j.eij.2025.100645>.
20. Tawil, A.A.; Almazaydeh, L.; Qawasmeh, D.; Qawasmeh, B.; Alshinwan, M.; Elleithy, K. Comparative Analysis of Machine Learning Algorithms for Email Phishing Detection Using TF-IDF, Word2Vec, and BERT. *Comput. Mater. Contin.* **2024**, *81*, 3395–3412. <https://doi.org/10.32604/cmc.2024.057279>.
21. Hasib, K.M.; Towhid, N.A.; Faruk, K.O.; Al Mahmud, J.; Mridha, M. Strategies for enhancing the performance of news article classification in Bangla: Handling imbalance and interpretation. *Eng. Appl. Artif. Intell.* **2023**, *125*, 106688. <https://doi.org/10.1016/j.engappai.2023.106688>.
22. Lauscher, A.; Ravishankar, V.; Vulić, I.; Glavaš, G. From Zero to Hero: On the Limitations of Zero-Shot Language Transfer with Multilingual Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, 16–20 November 2020; Webber, B., Cohn, T., He, Y., Liu, Y., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 4483–4499. <https://doi.org/10.18653/v1/2020.emnlp-main.363>.
23. Wagay, F.A.; Jahiruddin. Classification of Mental Illnesses from Reddit Posts Using Sentence-BERT Embeddings and Neural Networks. *Procedia Comput. Sci.* **2025**, *258*, 1669–1676. <https://doi.org/10.1016/j.procs.2025.04.398>.
24. Jáñez-Martino, F.; Alaiz-Rodríguez, R.; González-Castro, V.; Fidalgo, E.; Alegre, E. Spam email classification based on cybersecurity potential risk using natural language processing. *Knowl.-Based Syst.* **2025**, *310*, 112939. <https://doi.org/10.1016/j.knosys.2024.112939>.
25. Manita, G.; Chhabra, A.; Korbaa, O. Efficient e-mail spam filtering approach combining Logistic Regression model and Orthogonal Atomic Orbital Search algorithm. *Appl. Soft Comput.* **2023**, *144*, 110478. <https://doi.org/10.1016/j.asoc.2023.110478>.
26. Radomirovic, B.; Petrovic, A.; Zivkovic, M.; Njegus, A.; Budimirovic, N.; Bacanin, N. 4—Efficient spam email classification logistic regression model trained by modified social network search algorithm. In *Computational Intelligence and Blockchain in Complex Systems*; Al-Turjman, F., Ed.; Advanced Studies in Complex Systems; Morgan Kaufmann: San Francisco, CA, USA, 2024; pp. 39–55. <https://doi.org/10.1016/B978-0-443-13268-1.00010-8>.
27. Ulčar, M.; Žagar, A.; Armendariz, C.S.; Repar, A.; Pollak, S.; Purver, M.; Robnik-Šikonja, M. Mono- and cross-lingual evaluation of representation language models on less-resourced languages. *Comput. Speech Lang.* **2026**, *95*, 101852. <https://doi.org/10.1016/j.csl.2025.101852>.
28. Catelli, R.; Bevilacqua, L.; Mariniello, N.; Scotto di Carlo, V.; Magaldi, M.; Fujita, H.; De Pietro, G.; Esposito, M. Cross lingual transfer learning for sentiment analysis of Italian TripAdvisor reviews. *Expert Systems with Applications* **2022**, *209*, 118246. <https://doi.org/10.1016/j.eswa.2022.118246>.
29. Hu, Y.; Ding, J.; Dou, Z.; Chang, H. Short-Text Classification Detector: A BERT-Based Mental Approach. *Comput. Intell. Neurosci.* **2022**, *2022*, 8660828. <https://doi.org/10.1155/2022/8660828>.
30. Miller, C.; Portlock, T.; Nyaga, D.M.; O'Sullivan, J.M. A review of model evaluation metrics for machine learning in genetics and genomics. *Front. Bioinform.* **2024**, *4*, 1457619. <https://doi.org/10.3389/fbinf.2024.1457619>.
31. Chicco, D.; Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genom.* **2020**, *21*, 6. <https://doi.org/10.1186/s12864-019-6413-7>.
32. Chicco, D.; Warrens, M.J.; Jurman, G. The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen's Kappa and Brier Score in Binary Classification Assessment. *IEEE Access* **2021**, *9*, 78368–78381. <https://doi.org/10.1109/ACCESS.2021.3084050>.
33. Uysal, A.K.; Gunal, S. The impact of preprocessing on text classification. *Inf. Process. Manag.* **2014**, *50*, 104–112. <https://doi.org/10.1016/j.ipm.2013.08.006>.

34. Albitar, S.; Fournier, S.; Espinasse, B. An Effective TF/IDF-Based Text-to-Text Semantic Similarity Measure for Text Classification. In *Web Information Systems Engineering—WISE, 15th International Conference, Thessaloniki, Greece, 12–14 October 2014*; Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y., Eds.; Springer: Cham, Switzerland, 2014; pp. 105–114.
35. Moreo, A.; Esuli, A.; Sebastiani, F. Distributional Random Oversampling for Imbalanced Text Classification. In *SIGIR '16: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, 17–21 July 2016*; Association for Computing Machinery: New York, NY, USA, 2016; pp. 805–808. <https://doi.org/10.1145/2911451.2914722>.
36. Mohammadi, S.; Chapon, M. Investigating the Performance of Fine-tuned Text Classification Models Based-on Bert. In *Proceedings of the 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Yanuca Island, Fiji, 14–16 December 2020*; pp. 1252–1257. <https://doi.org/10.1109/HPCC-SmartCity-DSS50907.2020.00162>.
37. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to Fine-Tune BERT for Text Classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, 18–20 October 2019*; Sun, M., Huang, X., Ji, H., Liu, Z., Liu, Y., Eds.; Springer: Cham, Switzerland, 2019; pp. 194–206. https://doi.org/10.1007/978-3-030-32381-3_16.
38. Eltahier, S.; Dawood, O.; Saeed, I. BERT Fine-Tuning for Software Requirement Classification: Impact of Model Components and Dataset Size. *Information* **2025**, *16*, 981. <https://doi.org/10.3390/info16110981>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.