Electronic Journal of Applied Statistical Analysis
EJASA, Electron. J. App. Stat. Anal.
`http://siba-ese.unisalento.it/index.php/ejasa/index`
e-ISSN: 2070-5948
DOI: 10.1285/i20705948v18n2p238

Balancing performance and environmental efficiency: a multiclass classification study of textual data

By Priola, Romano

15 October 2025

Electronic Journal of Applied Statistical Analysis
Vol. 18, Issue 02, October 2025, 238-263
DOI: 10.1285/i20705948v18n2p238

Balancing performance and environmental efficiency: a multiclass classification study of textual data

Maria Paola Priola[a] and Maurizio Romano*[a]

[a]*Department of Economics and Business Sciences, University of Cagliari, Viale Fra Ignazio 17, Cagliari 09123, Italy*

15 October 2025

MultiClass Classification (MCC) is a foundational task in machine learning, especially within high-dimensional domains like text classification. While most studies focus on predictive accuracy, the growing demands of large-scale models have raised urgent questions about their environmental cost. In line with the Green AI paradigm, this work examines not only the performance but also the carbon and energy efficiency of various classifier–strategy combinations for MCC. Using two real-world textual datasets we systematically evaluate strategies such as One-Vs-Rest (OVA), One-Vs-One (OVO), Best-of-Best (BOB), and Error-Correcting Output Codes (ECOC), across classifiers ranging from simple Naïve Bayes to complex Artificial Neural Networks. We introduce emissions-per-accuracy metrics to measure the environmental efficiency of each configuration. Our findings show that while models like Random Forest incur high computational and ecological costs, simpler classifiers such as Logistic Regression and Naïve Bayes achieve comparable performance with drastically lower emissions. OVA consistently offers the best trade-off between speed and accuracy, while OVO and BOB prove more robust to class imbalance. Notably, Threshold-based Naïve Bayes paired with OVO demonstrates strong performance and sustainability. By integrating environmental considerations into MCC evaluation, this study highlights the importance of choosing classifier–strategy pairs that are not only effective but also computationally and ecologically responsible.

Keywords: Multiclass Classification, Green AI, Classification Strategies, Carbon Emissions, Text Classification, Supervised Learning

*Corresponding author: romano.maurizio@unica.it.

1 Introduction

MultiClass Classification (MCC) is a crucial and complex task that has gained increasing importance due to its widespread application across various domains, particularly in the context of textual data analysis. Selecting the appropriate classification strategy is vital, as it requires balancing model complexity, computational time, and environmental impact. The growing concern about climate change and the computational cost of large-scale models further underscores the importance of this balance.

MCC involves assigning a single class label to each instance from a set of more than two possible classes. This task differs from MultiLabel Classification (MLC), where multiple labels can be assigned to each instance (Sharma and Mehrotra 2018). For instance, a news article may be tagged with multiple topics like "sports", "politics", and "entertainment" in MLC, whereas MCC would assign a single label to each instance. The distinction between these two classification paradigms is fundamental to understanding the scope and challenges of MCC.

One of the primary approaches to handle MCC problems is the binarization technique, which decomposes the multiclass problem into multiple binary classification tasks. Techniques such as One-Vs-Rest (OVA) and One-Vs-One (OVO) have been widely employed (Galar et al. 2011). In OVA, a separate binary classifier is trained for each class against all others, while in OVO, classifiers are trained for every possible pair of classes. While OVA is computationally efficient, OVO can provide more robust predictions by focusing on specific class pairs.

Another significant strategy in MCC is the Best-Of-Best (BOB) approach (Conversano 2011). BOB refines the OVO technique by applying the classification in a two-stage process. Initially, OVO is applied, and subsequently, a final classifier is trained using only the top predicted class pairs to make the ultimate prediction.

Additionally, Error Correcting Output Codes (ECOC) provide an alternative method for handling multiclass problems. This strategy encodes classes into binary strings and employs error correction mechanisms to improve classification robustness (Dietterich and Bakiri 1994). This technique leverages redundancy in binary codes to mitigate classification errors, particularly in noisy datasets.

While MCC strategies are primarily assessed for predictive accuracy, recent discussions have shifted attention toward computational efficiency and environmental impact. The rise of resource-hungry AI models has given birth to the Green AI paradigm, which promotes sustainable practices in model training and deployment (Schwartz et al. 2020). Researchers now emphasize the importance of balancing performance with energy consumption to reduce the carbon footprint of large-scale AI systems (Strubell et al. 2019). To promote transparency, initiatives for standardized reporting of training time and energy usage have been introduced (Strubell et al. 2019). Furthermore, practical recommendations, such as optimizing hyperparameters, using sustainable cloud services, and selecting energy-efficient hardware, offer concrete steps towards more environmentally friendly AI solutions (Lacoste et al. 2019).

This study integrates environmental impact assessment into the comparative analysis of MCC strategies, evaluating not only predictive performance but also energy consumption and carbon footprint. This comprehensive evaluation includes statistical classifiers such as Naíve Bayes and it Threshold-Based variant (Romano et al. 2023), Random Forest, Logistic Regression, and Artificial Neural Networks, employing two distinct datasets: the widely recognized 20NewsGroups dataset from sklearn[1], and a dataset comprising unstructured Italian-language health-related news articles. This corpus, compiled from diverse media sources, offers contextual richness and supports natural language querying of health-related content, aligning with the study's broader objective of evaluating MCC in text data.

This study addresses a critical gap at the intersection of MCC and Green AI by systematically evaluating how strategy-classifier combinations impact both predictive performance and environmental efficiency in text data analysis. Using a comprehensive experimental framework, the study examines the trade-offs between accuracy, training time, and resource consumption across diverse classifier-strategy pairings. Specifically, the contributions of this work are threefold: (i) the development of a robust framework to evaluate multiple MCC strategies applied across heterogeneous text datasets, (ii) the use of computational efficiency metrics to assess the cost-effectiveness of each approach, and (iii) the integration of environmental assessment metrics that quantify energy consumption and CO_2 emissions using standardized carbon tracking tools. This structured evaluation not only reveals the computational and environmental costs of various MCC approaches but also provides actionable insights into optimizing classification workflows in line with Green AI principles. The study ultimately aims to inform practitioners about the practical implications of strategy and classifier selection, emphasizing the importance of balancing accuracy, efficiency, and sustainability in real-world applications.

Results show that classifier choice has a greater impact than strategy on both accuracy and environmental efficiency. Simpler models offer strong performance with low emissions, while complex models incur higher costs with limited accuracy gains. However, the combination of classifier and strategy proves crucial, as certain pairings significantly outperform others. Among strategies, OVA provides the best overall trade-off in most cases, underscoring the importance of aligning predictive goals with sustainability.

The remainder of the paper is structured as follows. Section 2 outlines the related works, Section 3 presents the data and its required processing, Section 4 reviews the methodology, Section 5 provides the results and Section 6 concludes.

2 Related Works

The literature on MCC strategies is extensive, encompassing a wide array of approaches aimed at enhancing classification accuracy. Despite substantial research, there remains no consensus on the optimal MCC strategy, as the effectiveness of these methods varies across datasets and application domains. Concurrently, the growing computational burden associated with large-scale MCC tasks has led to a surge in studies addressing the environmental impact of model training, aligning with the principles of Green AI. This

[1] https://scikit-learn.org/stable/

section reviews key contributions in both MCC strategies and Green AI frameworks, providing a comprehensive overview of recent advancements in these domains.

2.1 MultiClass Classification Strategies

Base classifiers are employed in MCC tasks, and numerous algorithms exist in the literature for decomposing the problem into binary classification subproblems, which is a trend in literature towards the search for techniques to increase classification accuracy. The literature on the use of MCC strategies is extensive, but there are no studies that unequivocally demonstrate the superiority of a specific strategy over others. One of the foundational works in this field is by Galar et al. (2011), who develop a comprehensive study of several established algorithms, including Support Vector Machines (SVM), Decision Trees, and Instance Based Learning with both OVO and OVA. They compare the performances of these algorithms using nineteen datasets from the widely recognized UCI repository. Their findings highlight the superiority of OVO over OVA in terms of prediction accuracy. In contrast, Rifkin and Klautau (2004) conduct a comparative analysis of MCC schemes, also focusing on OVA and OVO, emphasizing that the choice of binary classifier plays a more critical role than the choice of strategy. Their results suggest that once an effective base classifier is selected, the difference in performance between OVO and OVA diminishes. As such, they recommend the simpler OVA strategy in most cases, although they acknowledge that OVO can yield better results on smaller datasets.
MCC challenges are prevalent across a wide range of research domains, and both OVO and OVA strategies have been widely applied to address them. For example, Student and Fujarewicz (2012) propose a classification algorithm tailored for multiclass microarray data and evaluated its performance using both OVA and OVO strategies, comparing the results with standard models such as SVM. Their findings suggest that OVA performs well in high-dimensional settings and scales effectively to large datasets. Similarly, Zdrojewska et al. (2019) assess the performance of an OVO-based Multinomial Naïve Bayes classifier combined with ensemble techniques for classifying the widely used Reuters-21578 corpus. Their objective is to rank classifiers by accuracy and evaluate the improvements brought by boosting algorithms relative to traditional models like SVM and Random Forest. They conclude that ensemble methods consistently outperformed standalone classifiers. In another application, Delachaux et al. (2013) employ OVA neural network classifiers to enhance the reliability of indoor human activity recognition by integrating data from wearable and depth sensors.

Other studies have adapted OVO and OVA strategies to specific classifiers, often proposing sub-strategies to enhance performance. For example, Hong et al. (2008) apply the OVA approach to evaluate models such as SVM, Naïve Bayes, and hybrid combinations for improving fingerprint classification accuracy. Kumar and Gopal (2011) introduce a Reduced-OVA (ROVA) method for MCC-SVM tasks. Their approach defines a k-region around each class's decision boundary, selecting only the most representative data points for training. By excluding data outside this region, the method reduces dataset size and significantly improves training efficiency, achieving similar accuracy to standard OVA while reducing training time by approximately 50%. Eichelberger and Sheng (2013) examine OVA and OVO using Naïve Bayes and Logistic Regression across UCI datasets. While these strategies are commonly assumed to enhance classification, their findings indicate that OVA often degrades performance, and although OVO performs slightly better, it still falls short compared to ensemble methods like bagging. The authors argue that for algorithms inherently capable of multiclass classification, OVA and OVO should be avoided in favor of bagging. Lastly, Sáez et al. (2014) explore the robustness of MCC strategies in noisy environments. Their results show that OVO tends to yield more reliable classifiers under noisy conditions, making it a preferable choice when data quality is a concern.

More specifically, and in line with the objectives of this study, several works have applied MCC strategies to textual data. A significant number of studies have used the 20NewsGroups dataset for text mining and classification. For example, Albishre et al. (2015) evaluate the role of data cleaning, finding that although performance gains were modest, corpus size was reduced by 60%, improving efficiency and memory usage. Rennie and Rifkin (2001) compare Naïve Bayes and SVMs for MCC tasks, showing SVMs achieve substantially lower error rates, especially when used within an ECOC framework. Similarly, Adi and Çelebi (2014) address numerical instability in Naïve Bayes with a logarithm-based correction, achieving 86% accuracy—outperforming conventional techniques like Icsiboost-bigram and the Expected Maximum algorithm.

Beyond 20NewsGroups, other works have explored MCC across various textual domains. Dogan and Uysal (2020) introduce a new weighting scheme that improved classification across multiple datasets, while Han et al. (2021) applied meta-learning to few-shot text classification, improving accuracy notably for 1-shot and 5-shot scenarios. Li et al. (2006) examine discriminant analysis as an alternative to SVMs for text categorization, finding it competitive in multiclass contexts. In another practical application, D'Andrea et al. (2015) use SVMs to classify tweets related to traffic events, achieving a real-time detection system with 95% accuracy.

Our study directly addresses the open question of how different MCC decomposition strategies perform when paired with diverse classifiers on real-world text data. We conduct a systematic comparative analysis of multiple classifier–strategy combinations, ranging from classical statistical models to neural networks, evaluating not only their predictive accuracy but also their computational cost with two different datasets. This approach highlights which pairings offer the best trade-off between performance and efficiency on heterogeneous corpora.

2.2 Towards Sustainable AI: Energy and Carbon Tracking

Training large-scale MCC models can incur substantial energy consumption and carbon emissions. Schwartz et al. (2020) were among the first to quantify these environmental and economic costs, coining the shift from "Red AI" to "Green AI" to promote efficiency alongside accuracy. Expanding on this, Strubell et al. (2019) advocate for standardized reporting of training time, hardware use, and hyperparameter sensitivity to facilitate transparent comparisons. Similarly, Lacoste et al. (2019) introduce a Machine Learning

Emission Calculator that estimates carbon output based on training duration, hardware type, and data-center location, while also recommending practices such as using renewable-powered cloud platforms and lightweight hyperparameter searches.

To operationalize Green AI principles, several open-source toolkits have been developed to directly measure energy use and emissions. For example, Budennyy et al. (2022) present eco2AI, which tracks power consumption and regional carbon intensity; Anthony et al. (2020) propose Carbontracker for real-time monitoring and predictive energy use analysis; and Lottick et al. (2019) develop `CodeCarbon`, a plug-in for ML frameworks that estimates training emissions and generates standardized energy usage reports.

Beyond point tools, more comprehensive platforms support full-lifecycle impact assessments. Henderson et al. (2020) introduce Experiment Impact Tracker, offering region-specific carbon accounting, automated appendices, and a leaderboard for energy-efficient models. Lannelongue et al. (2021) propose Green Algorithms, an accessible online tool that contextualizes carbon impact against real-world benchmarks and provides actionable reduction recommendations.

Recent studies have built on these tools to develop structured evaluation frameworks. For instance, Hrib et al. (2024) propose a unified approach for collecting CO_2 and energy data using `eco2AI` and `CodeCarbon`, enabling comparison across models and hardware. Verma et al. (2024) assess deep learning models using multiple trackers to jointly evaluate accuracy and emissions. Bouza et al. (2023) provide a comparative review of `CodeCarbon`, `eco2AI`, and Experiment Impact Tracker, validating each tool against wattmeter data and offering practical usage guidelines.

Despite this growing ecosystem of tools and research, no prior work has systematically examined how MCC decomposition strategies affect energy consumption and carbon emissions. This study addresses that gap by evaluating a range of classifier–strategy combinations across heterogeneous text corpora, measuring not only predictive accuracy but also runtime, power draw, and CO_2 emissions. In doing so, we expose the performance–sustainability trade-offs inherent in MCC design choices.

3 Data and Preprocessing

Our study builds on the above-mentioned literature by focusing on two datasets. The the 20NewsGroup Dataset (Lang 1995), widely employed in the field of text classification and NLP for research purposes, and the Health News on Media Coverage Dataset, specifically developed for this study. Both datasets have been subjected to the same preprocessing pipeline to maintain consistency and comparability in analysis.

The 20NewsGroup dataset (20NG)[2], a well-known dataset of news articles available from the Sklearn library, contains 18,846 unique news articles categorized into 20 distinct topics. This dataset is used for text classification, topic modeling, and other machine learning tasks due to its diverse topics, which provide a robust testing ground for various algorithms. In contrast, the Health News Dataset on Media Coverage (HNMC) comprises 5,000 unique articles categorized into 5 distinct topics. This dataset serves

[2]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html

to complement the variability provided by the 20NG by introducing a set of categories with more structured topical segmentation and a multilingual component, as entirely in Italian. This aspect introduces unique challenges and opportunities for text processing, classification, and topic modeling in a non-English context.

Both datasets have been employed with the exclusion of headers, footers, and quotes, as these segments often contain high-specific or irrelevant information (Sarkar 2019). The preprocessing pipeline applied to both datasets is based on the text-corpus methodology outlined by Grün and Hornik (2011). It involves the following steps: removal of numbers, conversion to lowercase, deletion of extra spaces and punctuations, and elimination of standard stopwords using the `nltk` package (Bird et al. 2009). Subsequently, words occurring in fewer than 10 documents, as well as suffixes and prefixes, have been removed through lemmatization and stemming techniques. The lemmatization step employs a dictionary-based morphological analysis, converting words like "saw" to "see," while the stemming step implements Porter's algorithm, reducing words like "caresses" to "caress" and "economies" to "economi" (Porter 1980).

The final objective of the preprocessing stage is to generate tokens readable by machine learning models. In both datasets, unigrams (n-grams with a single word) are employed to construct document-term matrices, where each row represents a document and each column represents a feature (word) that occurs in at least one document. The matrices are populated using the term-frequency inverse-document-frequency (tf-idf) scheme, as defined by Baeza-Yates and Ribeiro-Neto (1999). The tf-idf function is formalized as follows:

$$\text{tf-idf}_{i,\omega} = \frac{\tilde{n}_{i,\omega}}{|d_\omega|} \cdot \log\left(\frac{|D|}{|\{d_\omega : i \in d_\omega\}|}\right) \tag{1}$$

where $\frac{\tilde{n}_{i,\omega}}{|d_\omega|}$ represents the *term frequency* (tf) of word i in text snippet ω; that is, $\tilde{n}_{i,\omega}$ is the number of occurrences of term i in snippet ω, and $|d_\omega|$ is the total number of terms in ω. The component $\log\left(\frac{|D|}{|\{d_\omega : i \in d_\omega\}|}\right)$ corresponds to the *inverse document frequency* (idf) of term i in the document collection D, where $|D|$ is the total number of documents and $|\{d_\omega : i \in d_\omega\}|$ is the number of documents containing term i. The tf-idf score thus assigns higher weights to terms that are frequent in a specific document but rare across the entire dataset, helping to identify distinctive features in the corpus.

At the end of data processing, the 20NG ends up containing 18,287 unique news articles. Each news varies from a minimum of 1 to a maximum of 5761 words, with an average of 85, a median of 38, and a standard deviation of 242. The vocabulary size consists of 128,204 tokens. According to Adi and Çelebi (2014), to conduct our investigation response classes of the 20NewsGroup Dataset are merged into 6 main topics: Information Technology (27.07%), miscellaneous (5.40%), politics (10.24%), recreation (22.02%), religion (13.41%) and science (21.87%). A brief summary of this composition is reported both in Figure 1 and Table 1.

The HNMC dataset exhibits an average document length of 380 words with a median length of 305 words. The standard deviation is 297, with document lengths ranging

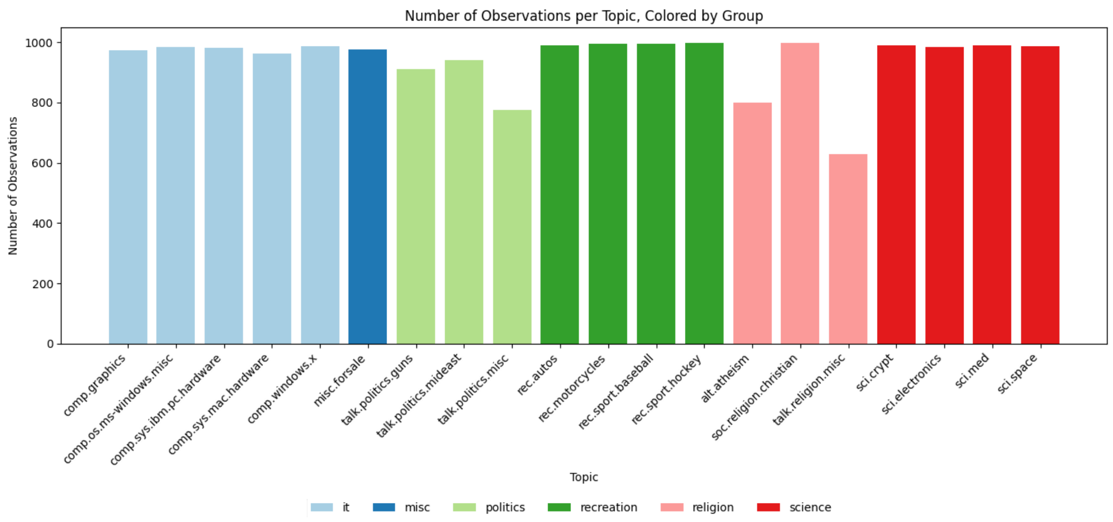

Figure 1: 20NewsGroups Dataset Topics

Table 1: 20NewsGroups dataset: Topic-Group Mapping

Topic	**Groups**
Information Technology	comp.graphics, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x
Miscellaneous	misc.forsale
Recreation	rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey
Politics	talk.politics.guns, talk.politics.mideast, talk.politics.misc
Religion	alt.atheism, soc.religion.christian, talk.religion.misc
Science	sci.crypt, sci.electronics, sci.med, sci.space

from a minimum of 15 words to a maximum of 3,638 words. The vocabulary size is 80,720 unique words, offering a significant breadth for feature extraction and model training. In Figure 2 the distribution of the 5 distinct topics: science-and-pharmaceuticals (20.86%), government-and-parliament (20.56%), work-and-professions (20.38%), letters-to-the-editors (19.32%), and studies-and-analysis (18.88%).

4 Methodology

4.1 Performance Metrics

We compare the performance of the models employing four MCC performance metrics: accuracy, sensitivity, precision, and F1-Score (see Taghavinejad et al. (2020)). We consider a classifier f that maps a dataset D of n instances into a set G composed of n_j

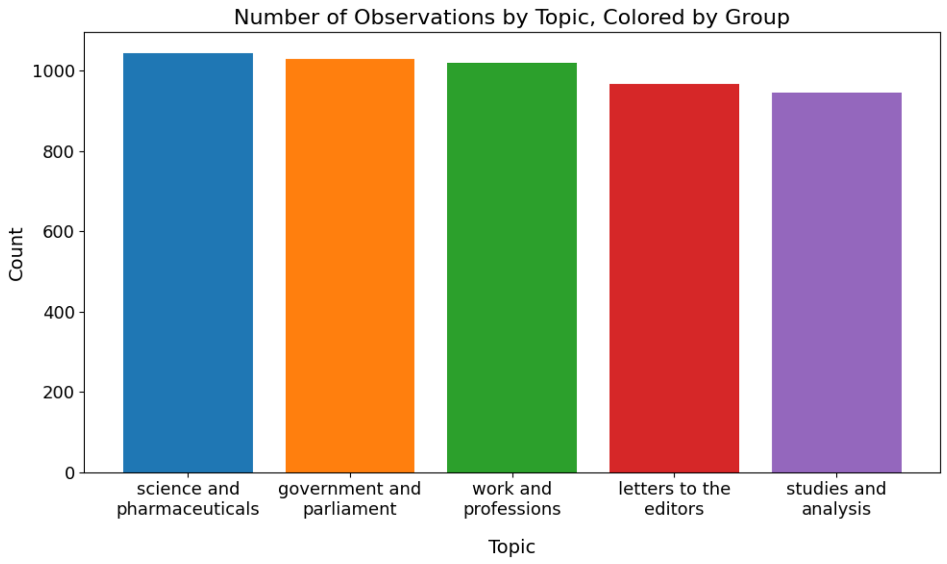

Figure 2: Health News on Media Coverage Topics

classes ($n_j >> 2$), that is $f : D \rightarrow G = \{g_1, \ldots, g_j, \ldots, g_{n_j}\}$.

Accuracy of a classifier f, denoted as A_f, is the ratio of correctly predicted instances to the total instances:

$$A_f = \sum_{j=1}^{n_j} \frac{\#(\hat{f}_{i,g_j} = f_{i,g_j})}{n} \tag{2}$$

where $\hat{f}_{i,g_j}$ denotes the predicted class for the i-th instance and f_{i,g_j} denotes the observed class. For a given class g_j, Precision (P_{f,g_j}) expresses the proportion of instances correctly classified in class g_j with respect to the total number of instances classified in class g_j. Recall (R_{f,g_j}), also known as sensitivity, expresses the proportion of instances correctly classified in class g_j with respect to the total number of instances in class g_j. F1-Score is computed as the harmonic mean of P_{f,g_j} and R_{f,g_j}. In MCC, Precision and Recall correspond, respectively, to the average precision and recall computed over the entire set of G classes ($g_j \in G$), namely:

$$P_f = \frac{1}{n_j} \sum_{j=1}^{n_j} \frac{\#(\hat{f}_{i,g_j} = f_{i,g_j})}{\hat{n}_{g_j}} \tag{3}$$

$$R_f = \frac{1}{n_j} \sum_{j=1}^{n_j} \frac{\#(\hat{f}_{i,n_j} = f_{i,n_j})}{n_{g_j}} \tag{4}$$

where n_{g_j} and $\hat{n}_{g_j}$ represent, respectively, the number of instances in class g_j and the number of instances classified in class g_j. Notably, F1-Score is the recommended measure in MCC as it accounts for possible unbalancing among classes (see Opitz and Burst (2021), among others). It is computed as:

$$F1_f = \frac{1}{n_j} \sum_{j=1}^{n_j} \frac{2 \cdot P_{f,g_j} \cdot R_{f,g_j}}{P_{f,g_j} + R_{f,g_j}} \tag{5}$$

4.2 Emissions Tracking

`CodeCarbon` (Lottick et al. 2019) and `eco2AI` (Budennyy et al. 2022) estimate the carbon footprint associated with computational tasks by monitoring the energy consumption of CPUs, GPUs, and RAM. The calculation of CO_2 emissions is based on the carbon intensity of electricity in the specific geographical location where the computation occurs. Both tools share a foundational approach to emissions estimation, with energy consumption calculated using the standard formula:

$$E = \sum_{u=1}^{n} PC_u \cdot T_u \tag{6}$$

where E denotes the total energy consumption in kilowatt-hours (kWh), PC_u represents the power consumption of component u in kilowatts (kW), and T_u is the duration of operation of component u in hours. The index u iterates over the set of monitored components, specifically CPU, GPU, and RAM. Following the determination of energy consumption, the regional carbon intensity is applied to estimate CO_2 emissions:

$$C_{CO_2} = E \times \gamma \tag{7}$$

where C_{CO_2} is the quantity of CO_2 emissions in kilograms (kg) and γ is the carbon intensity factor expressed in kilograms of CO_2 per kilowatt-hour (kg CO_2/kWh).

Despite the similarity in their core methodology, `CodeCarbon` and `eco2AI` differ in their implementation and parameterization. The former relies primarily on system interfaces such as Intel's RAPL[3] for CPUs and NVIDIA Management Library[4] for GPUs to obtain real-time power consumption data. When direct measurement tools are unavailable, it estimates energy consumption based on the hardware's Thermal Design Power and system utilization metrics. For RAM, the tool approximates power consumption based on the number of RAM slots utilized, assigning a default power consumption of 5 Watts per slot (Lottick et al. 2019).

In contrast, `eco2AI` employs a more detailed and flexible approach to energy estimation. It leverages the Pynvml[5] wrapper for GPU monitoring, utilizes a comprehensive database of over 3,000 CPU models to estimate CPU power usage, and considers specific energy consumption rates for DDR-n RAM modules. In the context of RAM, `eco2AI` assumes a power consumption of 0.375 W per GB of allocated memory (Budennyy et al. 2022). Additionally, this tool introduces the Power Usage Effectiveness (PUE) parameter, accounting for data center overhead, which is particularly relevant for cloud-based

[3]https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html

[4]https://developer.nvidia.com/management-library-nvml

[5]https://pypi.org/project/pynvml/

computations. The inclusion of the PUE factor modifies the carbon footprint calculation as follows:

$$CF = \gamma \times PUE \times (E_{CPU} + E_{GPU} + E_{RAM}) \tag{8}$$

where CF denotes the total carbon footprint in kilograms of CO_2. The parameter PUE is a dimensionless factor, set to a default value of 1 but adjustable based on data center efficiency. The energy consumption components E_{CPU}, E_{GPU}, and E_{RAM} represent the energy consumed by the CPU, GPU, and RAM respectively, measured in kilowatt-hours.

While both tools provide robust frameworks for emissions tracking, `eco2AI`'s incorporation of the PUE factor and more granular component-level monitoring offers a more comprehensive estimation, particularly when evaluating cloud-based computations.

4.3 Strategies and Classifiers

This study evaluates the performance of multiple statistical classifiers across four MCC strategies, denoted by the set $\mathcal{S}$: OVA, OVO, BOB, and ECOC. The classifiers considered form the set $\mathcal{F}$, which includes Artificial Neural Network (ANN), Linear Discriminant Analysis (LDA), Logistic Regression (LR), Naïve Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), and Threshold-Based Naïve Bayes (TB-NB). Model performance is assessed using a 10-fold cross-validation procedure, as detailed in Algorithm 1.

In OVA, the learning set D^L is divided into k disjoint folds. Each fold serves as the test set, while the remaining $k-1$ folds are used to train $n_j - 1$ binary classifiers, each contrasting class j against the other $n_j - 1$ classes. Final predictions are obtained via majority voting.

In OVO, a classifier is trained for each pair of classes within each fold, resulting in a total of $\binom{n_j}{2} \cdot k$ classifiers. The class receiving the highest number of votes across all pairwise comparisons is selected as the final prediction.

BOB extends the OVO strategy by focusing, for each test instance, on the top two classes receiving the most votes. A final binary classifier is trained using only the data points corresponding to these two classes, and its prediction determines the final label.

In ECOC, each class is represented by a unique binary code. Binary classifiers are trained to predict each bit in the code, and final predictions are made by assigning the test instance to the class whose code is closest in Hamming distance to the predicted bit string.

The classifiers used in this study are selected for their diverse modeling assumptions and methodological foundations. ANN is inspired by biological neural networks and was introduced as the perceptron by Rosenblatt (1958). It consists of input, hidden, and output layers, with interconnected units that apply mathematical functions to model complex patterns. ANNs are capable of learning tasks such as classification and regression without explicit task-specific rules. LDA, introduced by Fisher (1936), is a statistical method for classification and dimensionality reduction. It identifies a linear combination of features that maximizes between-class variance while minimizing within-class variance, enhancing class separability. Its computational efficiency stems from solving a generalized eigenvalue problem (Xanthopoulos et al. 2013). LR (Hosmer and Lemeshow 2000) models the probability of a binary outcome using the logistic function, which bounds output between 0 and 1. For MCC, LR is extended through OVA or OVO schemes to handle multiple classes. RF, proposed by Breiman (2001), is an ensemble-based classifier that constructs a collection of decision trees during training. It predicts classes by aggregating votes from individual trees, combining robustness and interpretability. SVM introduced by Cortes and Vapnik (1995), finds the hyperplane that maximally separates data classes. It supports both linear and nonlinear classification using kernel functions (e.g., linear, polynomial, RBF). SVMs perform well in high-dimensional spaces and are valued for their ability to model complex, nonlinear relationships (Zhang and Wang 2011). NB is a generative probabilistic classifier based on the assumption of feature independence (Murty and Devi 2011). For a text snippet ω_t of length d_{ω_t}, the probability of it belonging to class j is given by:

$$\pi(j \mid i) \propto \pi(j) \prod_{i=1}^{d_{\omega_t}} \pi\left(i \mid j\right) \tag{9}$$

where $\pi(i|j)$ is the conditional probability of word i appearing in class j, and $\pi(j)$ is the prior probability of class j. The classification decision is made by selecting the class with the highest posterior probability under this assumption.

Given this in mind, Romano et al. (2023) introduced the TB-NB classifier tailored for the binary output case, which has been fruitfully used in sentiment analysis. It is based on the computation of a scoring function that considers the log-odds ratio of the probability of a text snippet belonging to one of the two competing classes. Next, TB-NB utilizes the previously defined computed score while introducing a threshold to define the final decision rule. This threshold is used to classify a text snippet into one of the two classes.

The scoring function $\Lambda(\cdot)$ is computed for all the terms composing each individual text snippet included in the learning set based on the probability function $\pi(\cdot)$ and the Bayes' rule in order to predict, as accurately as possible, if a text snippet ω_t belongs to the class j or j'. Notationally, for a text snippet ω_t and two classes j and j', the scoring function $\Lambda(\omega_t|i)$ corresponds to the log-odds ratio of the probability that a text snippet ω_t belongs to class j given that it includes a certain word i, that is $\pi(\omega_t^j|i)$ over the probability that ω_t belongs to class j' given that it includes the word i, denoted as $\pi(\omega_t^{j'}|i)$. Thus, the log-odds ratio Λ for a text snippet ω_t that contains a word i is:

$$
\begin{aligned}
\Lambda\left(\omega_t|i\right) &= \log\left[\frac{\pi(\omega_t^j|i)}{\pi(\omega_t^{j'}|i)}\right] \\
&= \log\left[\frac{\pi(i|\omega_t^j)}{\pi(i|\omega_t^{j'})}\cdot\frac{\pi(\bar{i}|\omega_t^j)}{\pi(\bar{i}|\omega_t^{j'})}\cdot\frac{\pi(\omega_t^j)}{\pi(\omega_t^{j'})}\right] \\
&= \underbrace{\left[\log\pi(i|\omega_t^j)-\log\pi(i|\omega_t^{j'})\right]}_{\mathcal{L}(i)} + \underbrace{\left[\log\pi(\bar{i}|\omega_t^j)-\log\pi(\bar{i}|\omega_t^{j'})\right]}_{\mathcal{L}(\bar{i})} \\
&\quad + \left[\log\pi(\omega_t^j)-\log\pi(\omega_t^{j'})\right] \\
&\approx \mathcal{L}(i)+\mathcal{L}(\bar{i})
\end{aligned} \tag{10}
$$

In Eq. 10, $\mathcal{L}(i)$ measures how likely a specific word i is present in a text snippet, whilst $\mathcal{L}(\bar{i})$ measures how likely i is not present in the same one. Those two functions derive from the log-likelihood ratio of the event $(i \in \omega_t)$ and $(i \notin \omega_t)$, respectively. The term $\left[\log\pi(\omega_t^j)-\log\pi(\omega_t^{j'})\right]$ is discarded as it is constant for all the words i composing ω_t. It corresponds to the proportions of observed text snippets in class j (j') in the set of text snippets included in the collection D.
Therefore, Eq. 10 measures the likelihood of a document ω_t to belong to class j or j' given that it includes a certain word i. Extending Eq. 10 to all the d_ω words composing ω_t allows us to compute the scoring function $\Lambda(\omega_t)$ for the entire text snippet, that is, for all the d_ω terms composing it:

$$
\Lambda\left(\omega_t\right) = \sum_{r=1}^{d_\omega}\Lambda\left(\omega_t|i_r\right) = \sum_{r=1}^{d_\omega}\mathcal{L}(i_r)+\mathcal{L}(\bar{i}_r) \tag{11}
$$

with $(i_r,\ldots,i_r,\ldots i_{d_\omega}) \in \omega_t$.
Once the set of scores $\Lambda\left(\omega_t\right)$ is computed for all the available text snippets in the learning set, the decision rule $\mathcal{D}$ used to classify a test set instance ω_t^* is defined based on the estimated value of a threshold parameter τ, which is the unique parameter to be estimated when using TB-NB. It corresponds to a specific value of the log-odds ratio $\Lambda\left(\omega_t\right)$ computed on the learning set instances. For the two class case, the decision rule $\mathcal{D}_{\omega_t^*}$ that allows us to classify a text snippet d_j^* in class j or j' is:

$$
\mathcal{D}_{\omega_t^*}:\begin{cases}\Lambda\left(\omega_t^*\right) > \hat{\tau} & \rightarrow \omega_t^* = j\\ \Lambda\left(\omega_t^*\right) \leq \hat{\tau} & \rightarrow \omega_t^* = j'\end{cases} \tag{12}
$$

In the implemented horse race $\hat{\tau}$ corresponds to the value of $\Lambda\left(\omega_t\right)$ minimizing both Type I and Type II errors and is estimated by k-fold cross-validation.

4.4 Algorithm for Performance, Time and Emissions Tracking

The framework for implementing MCC strategies is presented in Algorithm 1. The algorithm accepts a dataset D consisting of n labeled text instances, a set of M classifiers $\mathcal{F}$, and a set of H MCC strategies $\mathcal{S}$. Additionally, it incorporates the estimation of energy consumption and CO_2 emissions, as previously defined.

The dataset D is partitioned into a learning set D^L of $\alpha \times n$ instances and a test set D^T of $(1-\alpha) \times n$ instances, such that $D^L \cap D^T = \emptyset$ and $0 < \alpha < 1$. For each classifier $f_m \in \mathcal{F}$ and strategy $s_h \in \mathcal{S}$, the framework performs training, prediction, performance metrics, energy tracking, and emissions estimation.

Algorithm 1 Framework for MCC Strategies with Energy and Emissions Tracking

```
Input:
    $D$ - Dataset
    $\mathcal{F} = \{f_1, \dots, f_M\}$ classifiers
    $\mathcal{S} = \{s_1, \dots, s_H\}$ strategies
    $\alpha = 0.7$ train-test split ratio
    $k = 10$ folds used in cross-validation
    $\gamma$ - Carbon intensity (kg $CO_2$/kWh)
    $PUE$ - Power Usage Effectiveness (default: 1)

Output:
    $A_{m,h}$, $P_{m,h}$, $R_{m,h}$, $F1_{m,h}$ - Performance metrics
    $\mathcal{C}_{m,h}$ - Energy consumption (kWh)
    $\mathcal{E}_{m,h}$ - $CO_2$ emissions (kg)

Procedure:
 1: Partition $D$ into $D^L$ and $D^T$ using $\alpha$
 2: for $h = 1$ to $H$ do
 3:     for $m = 1$ to $M$ do
 4:         Init $\mathcal{C}_{m,h} = 0$, $\mathcal{E}_{m,h} = 0$, and performance metrics to zero
 5:         Init $\phi$: training, $\psi$: inference, $V_k$: $k$-fold cross-validation
 6:         $\phi(f_{m,h}(D^L)) = \hat{f}_{m,h}(D^L)$
 7:         $V_k(\hat{f}_{m,h}(D^L)) = \hat{f}_{m,h,k}(D^L)$
 8:         $\psi(f_{m,h}(D^T)) = \hat{f}_{m,h}(D^T)$
 9:         Compute $A_{m,h}, P_{m,h}, R_{m,h}, F1_{m,h}$ on $D^L$ and $D^T$
10:         $\mathcal{C}_{m,h} = \sum_{u \in \{CPU,GPU,RAM\}} \sum_{i=1}^{n} PC_{i,u} \cdot T_{i,u}$
```

$$\mathcal{E}_{m,h} = \begin{cases} \gamma \cdot \mathcal{C}_{m,h}, & \text{if } \texttt{CodeCarbon} \\ \gamma \cdot PUE \cdot \mathcal{C}_{m,h}, & \text{if } \texttt{eco2AI} \end{cases} \tag{13}$$

```
11:     end for
12: end for
```

5 Results

In this section, we present the results obtained by implementing Algorithm 1. All experiments were conducted on a 12th Gen Intel(R) Core(TM) i7-12700H CPU 2.70 GHz, with 14 cores, 20 logical processors, and 16GB of DDR4 RAM.

We begin with a classifier–strategy analysis, evaluating the performance of each classifier across all MCC strategies using the metrics described in Section 4.1. Conversely, we assess the performance of each MCC strategy across all classifiers to identify which strategies are best suited to specific tasks. Subsequently, we examine the environmental and computational efficiency of each classifier, focusing on training time, energy consumption, and estimated CO_2 emissions.

5.1 Classifier-Strategy pair Analysis

In terms of classifiers achievements, results of both training and test performance are reported in Figures 3-4. Rankings for both classifier–strategy and strategy–classifier pairs are presented in Tables 2–5.

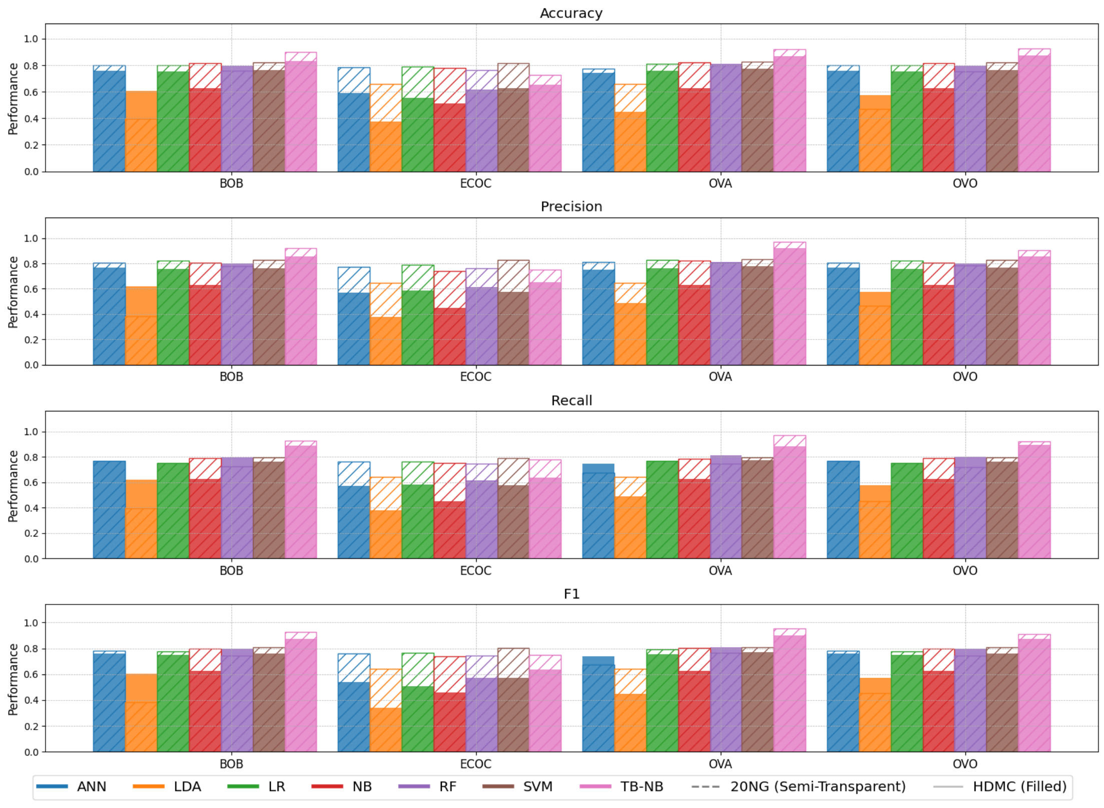

Figure 3: Performance Metrics for Learning Set (both datasets)

Table 2: Accuracy Classifier-Strategy and Strategy-Classifier Ranking Pairs

Classifier	**BOB**		**ECOC**		**OVA**		**OVO**			
	HNMC	20NG	HNMC	20NG	HNMC	20NG	HNMC	20NG	**avg**	**med**
ANN	4 (1)	5 (1)	4 (4)	3 (3)	4 (3)	6 (4)	4 (2)	4 (2)	4.38	4
LR	5 (2)	3 (2)	5 (4)	2 (4)	3 (1)	3 (1)	5 (2)	5 (2)	3.75	3.5
NB	6 (2)	2 (2)	6 (4)	4 (4)	6 (1)	4 (1)	6 (2)	6 (2)	4.63	5
RF	2 (3)	6 (3)	3 (4)	5 (2)	1 (1)	5 (1)	2 (2)	2 (2)	3.75	4
TB-NB	1 (2)	1 (2)	1 (4)	7 (4)	5 (3)	1 (3)	1 (1)	1 (1)	**2.25**	**1**
LDA	7 (1)	7 (3)	7 (4)	6 (1)	7 (3)	7 (2)	7 (2)	7 (2)	6.88	7
SVM	3 (2)	4 (4)	2 (4)	1 (3)	2 (1)	2 (1)	3 (3)	3 (3)	2.38	2
avg	2.15		3.50		**1.90**		2.30			
med	2		4		**1**		2			

Note: Rankings indicate classifier-strategy pairs (strategy-classifier pairs in parentheses) for each dataset. 'avg' and 'med' refer to the average and median rankings across datasets, respectively. Best overall performers are shown in bold.

Table 3: Precision Classifier-Strategy and Strategy-Classifier Ranking Pairs

Classifier	**BOB**		**ECOC**		**OVA**		**OVO**			
	HNMC	20NG	HNMC	20NG	HNMC	20NG	HNMC	20NG	**avg**	**med**
ANN	4 (1)	5 (2)	5 (4)	4 (4)	5 (3)	6 (3)	4 (2)	4 (2)	4.63	4.5
LR	5 (3)	3 (2)	3 (4)	2 (4)	4 (1)	2 (1)	5 (2)	5 (2)	3.38	3
NB	6 (2)	4 (2)	6 (4)	5 (4)	6 (1)	3 (1)	6 (2)	6 (2)	5.13	5.5
RF	2 (3)	6 (3)	2 (4)	3 (4)	2 (1)	5 (1)	2 (2)	2 (2)	3.50	2.5
TB-NB	1 (1)	1 (2)	1 (4)	6 (4)	1 (3)	1 (1)	1 (2)	1 (2)	**1.63**	**1**
LDA	7 (1)	7 (3)	7 (4)	7 (2)	7 (3)	7 (1)	7 (2)	7 (2)	7.00	7
SVM	3 (2)	2 (2)	4 (4)	1 (3)	3 (1)	4 (4)	3 (3)	3 (3)	2.75	3
avg	2.10		3.80		**1.80**		2.20			
med	2		4		**1**		2			

Note: Rankings indicate classifier-strategy pairs (strategy-classifier pairs in parentheses) for each dataset. 'avg' and 'med' refer to the average and median rankings across datasets, respectively. Best overall performers are shown in bold.

Table 4: Recall Classifier-Strategy and Strategy-Classifier Ranking Pairs

Classifier	**BOB**		**ECOC**		**OVA**		**OVO**			
	HNMC	20NG	HNMC	20NG	HNMC	20NG	HNMC	20NG	**avg**	**med**
ANN	4 (1)	4 (1)	5 (4)	3 (3)	5 (3)	7 (4)	4 (2)	4 (2)	4.50	4
LR	5 (3)	5 (2)	3 (4)	2 (4)	4 (1)	4 (1)	5 (2)	5 (2)	4.13	4.5
NB	6 (2)	3 (1)	6 (4)	5 (4)	6 (1)	3 (3)	6 (2)	6 (2)	4.75	5.5
RF	2 (3)	6 (3)	2 (4)	4 (1)	1 (1)	5 (2)	2 (2)	2 (2)	3.50	3
TB-NB	1 (2)	1 (1)	1 (4)	7 (4)	2 (3)	1 (3)	1 (1)	1 (1)	**1.88**	**1**
LDA	7 (1)	7 (3)	7 (4)	6 (1)	7 (3)	6 (2)	7 (2)	7 (2)	6.75	7
SVM	3 (2)	2 (1)	4 (4)	1 (3)	3 (1)	2 (4)	3 (3)	3 (3)	2.50	2.5
avg	**1.85**		3.45		2.30		2.30			
med	**2**		4		3		**2**			

Note: Rankings indicate classifier-strategy pairs (strategy-classifier pairs in parentheses) for each dataset. ‘avg’ and ‘med’ refer to the average and median rankings across datasets, respectively. Best overall performers are shown in bold.

Table 5: F1 Classifier-Strategy and Strategy-Classifier Ranking Pairs

Classifier	**BOB**		**ECOC**		**OVA**		**OVO**			
	HNMC	20NG	HNMC	20NG	HNMC	20NG	HNMC	20NG	**avg**	**med**
ANN	4 (1)	4 (1)	4 (4)	3 (3)	5 (3)	7 (4)	4 (2)	4 (2)	4.38	4
LR	5 (2)	5 (2)	5 (4)	2 (4)	4 (1)	4 (1)	5 (3)	5 (3)	4.38	5
NB	6 (2)	3 (2)	6 (4)	5 (4)	6 (1)	3 (1)	6 (2)	6 (2)	4.75	5.5
RF	2 (3)	6 (2)	2 (4)	4 (4)	2 (1)	5 (1)	2 (2)	2 (2)	3.63	3
TB-NB	1 (2)	1 (1)	1 (4)	7 (4)	1 (3)	1 (2)	1 (1)	1 (1)	**1.75**	**1**
LDA	7 (1)	7 (3)	7 (4)	6 (1)	7 (3)	6 (2)	7 (2)	7 (2)	6.75	7
SVM	3 (2)	2 (2)	3 (4)	1 (3)	3 (1)	2 (4)	3 (3)	3 (3)	2.38	2.5
avg	**1.90**		3.65		2.00		2.35			
med	**2**		4		**2**		**2**			

Note: Rankings indicate classifier-strategy pairs (strategy-classifier pairs in parentheses) for each dataset. ‘avg’ and ‘med’ refer to the average and median rankings across datasets, respectively. Best overall performers are shown in bold.

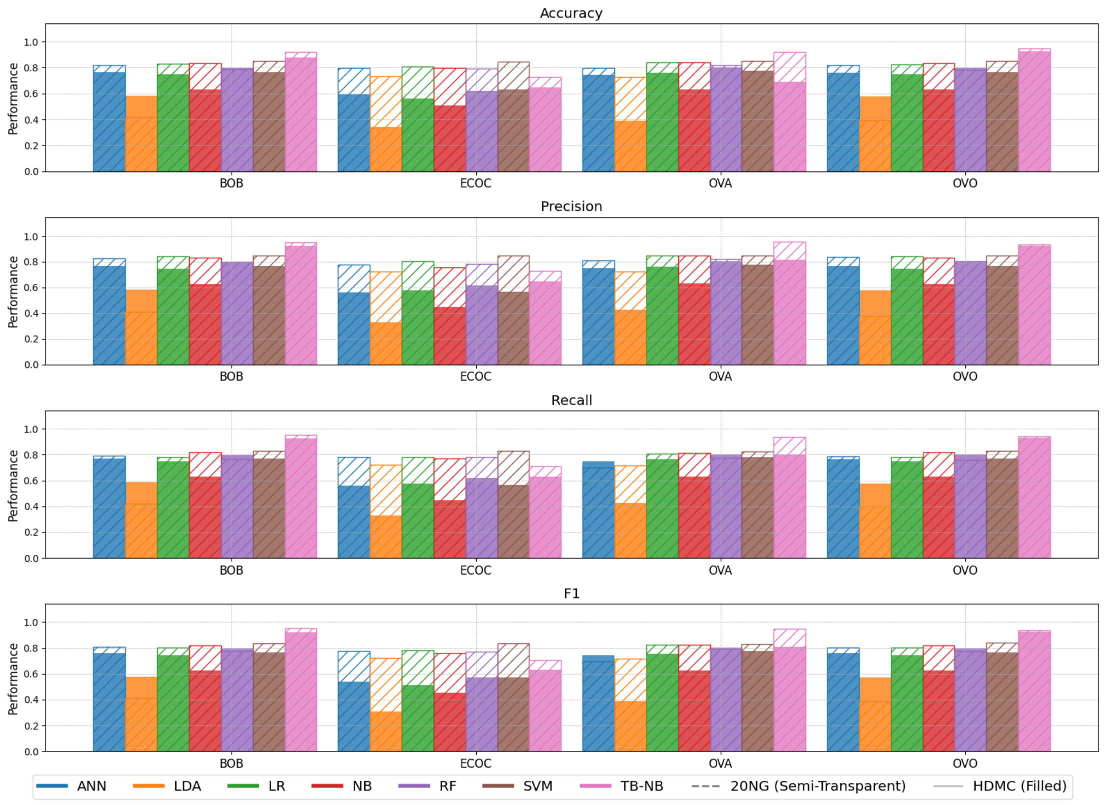

Figure 4: Performance Metrics for Testing Set (both datasets)

Looking across all classifiers, TB-NB emerges as the top performer, achieving the highest median and average ranks across evaluation metrics. Its consistent performance across strategies highlights its robustness and adaptability. SVM ranks second overall, delivering strong results under most strategies. Notably, it remains relatively stable even in contexts where ECOC performs poorly. LR also secures a solid overall position, particularly when paired with OVA, offering a dependable balance between accuracy and generalization. In contrast, LDA consistently underperforms across strategies and datasets, confirming its limited adaptability and overall weakness as a classifier.

Among the evaluated strategies, OVA proves to be the most effective overall, achieving the highest average and median rankings in both accuracy and precision. This underscores its reliability as a balanced and general-purpose MCC approach. BOB and OVO achieve the best results in recall and F1-score, making them the most effective options for recall-sensitive tasks where minimizing false negatives is critical. In contrast, ECOC ranks lowest overall, exhibiting inconsistent performance across classifiers. As said, a notable exception is observed when ECOC is paired with SVM, particularly on the 20NG dataset, where a localized performance improvement is evident. However, this isolated success does not compensate for ECOC's broader instability.

Looking across all classifier–strategy combinations, `TB-NB` stands out due to its strong compatibility with both `OVA` and `OVO`, which contributes significantly to its leading position in the aggregated rankings. Additionally, `LR` demonstrates strong performance when combined with `OVA`, reinforcing its position as the most balanced and dependable strategy overall. In contrast, the combination of `ECOC` and `LDA` performs the worst across all metrics and classifiers, highlighting a clear lack of adaptability and overall effectiveness.

5.2 Environmental Impact and Efficiency

To assess the environmental and computational cost of each classifier-strategy combination, we analyze the trade-off between generalization performance and training time.

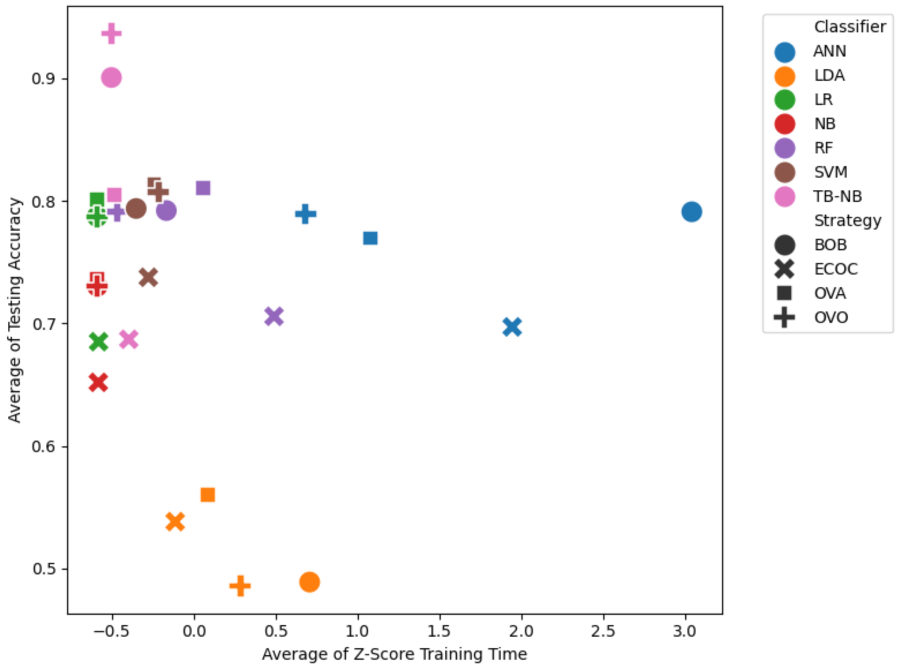

Figure 5: Trade-off between Training Time and Testing Accuracy

In Figure 5, the optimal classifier-strategy combinations appear in the upper-left quadrant—indicating high accuracy paired with low training time. This region represents the best trade-off between performance and computational efficiency.

Accuracy values are used without modification, as they are already normalized between 0 and 1. Training times, initially recorded in *hh:mm:ss.fff* format, are converted to total seconds to ensure consistency in computational comparisons. To allow fair comparisons across datasets of different sizes, training times are standardized using z-score normalization within each dataset. We then aggregate results by averaging the per-dataset z-scores and the corresponding mean accuracies for each classifier-strategy pair. This equal-weight approach ensures that neither dataset disproportionately influences the outcome, enabling balanced and interpretable comparisons.

Based on this analysis, several trends emerge. NB and LR consistently show the shortest training times across both datasets. Combined with stable accuracy across strategies, they stand out as efficient, low-cost classifiers. ANN and RF incur significantly higher training costs. Although they are capable of strong predictive performance, their computational demands reduce their efficiency in time-constrained or resource-limited environments.

LDA offers the weakest trade-off, combining low accuracy with inefficient training performance, confirming its limited suitability for MCC tasks. TB-NB achieves moderate training times—typically between 7 and 10 minutes—and performs most efficiently under the OVO and BOB strategies. It emerges as the overall best choice when considering the combined trade-off between training time and accuracy, in line with its strong overall ranking. Notably, SVM exhibits low variability in both accuracy and training time across strategies, reinforcing its reliability as the most consistent performer.

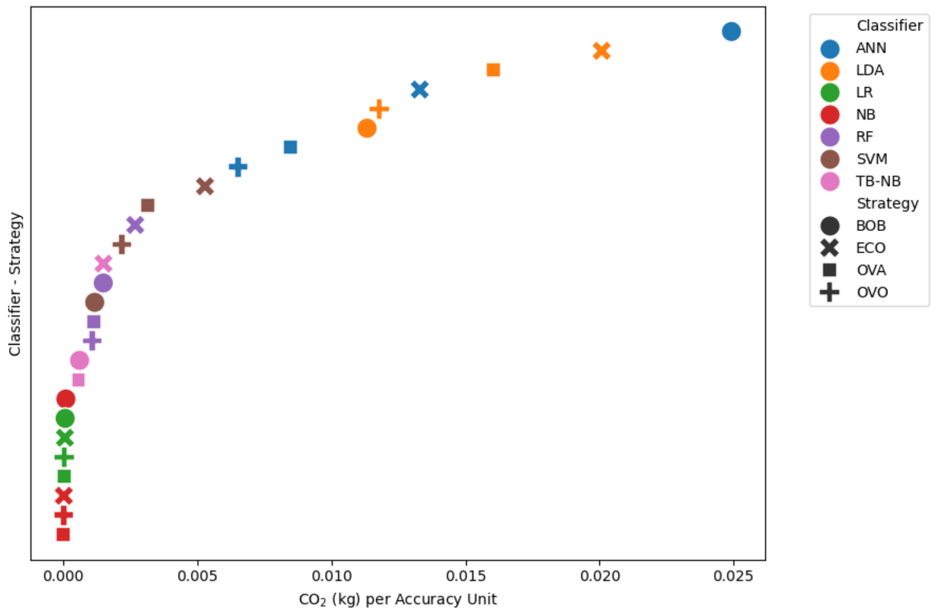

Figure 6: CO_2 Efficiency by Classifier and Strategy

To assess the environmental impact of the evaluated classifiers, we collect carbon and energy emission data as explained in Section 4.2. We average the emissions reported by both tools for each dataset to produce a unified estimate. Since the datasets differ in size, direct comparison of raw emission values would be misleading. To normalize for dataset variability and model performance, we compute two efficiency ratios: CO_2 per accuracy unit (in kilograms) and energy per accuracy unit (in kilowatt-hours). These metrics represent the environmental cost required to achieve a unit of predictive accuracy.

The resulting values are visualized in Figures 6 and 7. The analysis shows that the ANN and LDA classifiers perform worst in terms of environmental efficiency, with ANN reaching up to 0.025 kg CO_2 and 0.074 kWh per accuracy point under the BOB strategy. Similarly,

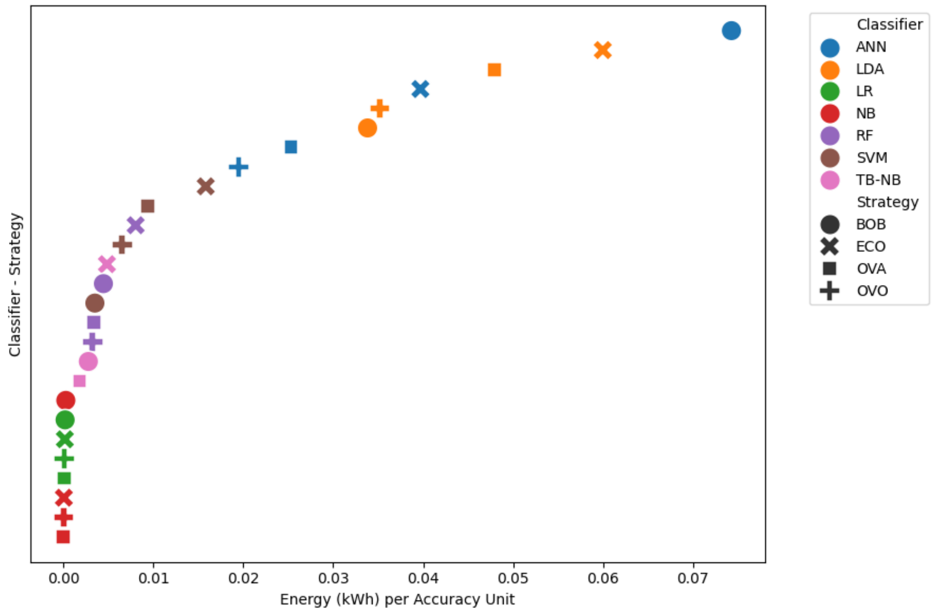

Figure 7: Energy Efficiency by Classifier and Strategy

LDA shows consistently high emission values across strategies, with the highest reaching 0.020 kg CO_2 and 0.060 kWh per accuracy point under ECO. In contrast, NB and LR yield the most favorable emission-to-performance ratios, with values as low as 1.06×10^{-5} kg CO_2 and 3.88×10^{-5} kWh per accuracy unit for NB under OVA.

TB-NB positions itself close to SVM, forming a cluster of models that balance higher predictive performance with reasonable environmental efficiency. This confirms that TB-NB represents a promising trade-off when prioritizing both accuracy and sustainability.

6 Conclusions

In MultiClass Classification, the choice of both classifier and strategy is critical, as it directly affects not only predictive performance but also training efficiency and environmental impact. The latter is particularly relevant given the growing awareness of machine learning's carbon footprint in the context of climate change. Classifiers vary in complexity, interpretability, and suitability for different data distributions, with more complex models often requiring significantly greater computational resources—potentially increasing the environmental cost of deployment.

Our analysis shows that classifier choice has the greatest influence on overall performance and efficiency. TB-NB emerges as the top performer, achieving the best overall ranking across metrics and maintaining strong compatibility with multiple MCC strategies. SVM ranks second, offering a compelling balance between accuracy, training time, and emissions, particularly due to its consistent behavior across strategies. While LR and NB do not lead in accuracy, they demonstrate exceptional training efficiency and low emission-to-performance ratios, making them ideal for sustainability-oriented applications. In contrast, LDA underperforms in both accuracy and environmental efficiency, and ANN and RF, despite their modeling capacity, incur high computational and energy costs without commensurate performance gains.

To complement performance evaluations, we assess environmental efficiency by computing CO_2 and energy consumption per unit of test accuracy. This metric enables fairer comparisons by incorporating both predictive effectiveness and resource usage. LR and NB emerge as the most environmentally efficient classifiers, while ANN and RF exhibit substantial training times and computational demands. Although these models can effectively capture complex relationships, their extended training durations do not correspond to meaningful improvements in accuracy relative to other classifiers. This discrepancy raises concerns about their environmental viability, particularly in contexts emphasizing sustainability.

The role of MCC strategies, while secondary to classifier choice, is nonetheless important. OVA provides the best overall balance between performance and training efficiency, particularly when paired with efficient classifiers such as LR, NB, or TB-NB. In contrast, ECOC ranks lowest overall and proves incompatible with several classifiers. However, it demonstrates localized success when paired with SVM. BOB and OVO yield the highest scores in recall and F1-score, making them suitable for recall-sensitive tasks.

From a classifier–strategy pairing perspective, TB-NB combined with OVO offers the best performance-to-emission ratio, making it the leading choice for sustainable yet high-performing classification. We also recommend LR and NB paired with OVA for contexts requiring minimal resource use and dependable accuracy.

Our findings advocate for a more holistic and sustainability-aware approach to classifier selection. The environmental implications of model and strategy selection should not be overlooked. As machine learning becomes more integral to real-world systems, integrating sustainability considerations into model development will be increasingly essential. Researchers and practitioners are encouraged to jointly evaluate classifiers and strategies, rather than optimizing accuracy in isolation.

Future work should aim to refine environmental impact assessments by incorporating factors such as hyperparameter tuning and model search, which often involve substantial computational effort. More comprehensive evaluation frameworks will be critical in supporting the development of sustainable and responsible AI systems.

Acknowledgements

The acknowledgments are detailed by the author.

Maria Paola Priola. This publication was produced while attending the PhD program in Economics and Business at the University of Cagliari, Cycle 38, with the support of a scholarship financed by the Ministerial Decree no. 351 of 9th April 2022, based on the NRRP - funded by the European Union - NextGenerationEU - Mission 4 “Education and Research", Component 1 "Enhancement of the offer of educational services: from nurseries to universities" - Investment 3.4 "Advanced teaching and university skills".

Maurizio Romano acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No.3277 published on December 30, 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union – Next Generation EU. Project Code ECS0000038 – Project Title eINS Ecosystem of Innovation for Next Generation Sardinia – CUP F53C22000430001- Grant Assignment Decree No. 1056 adopted on June 23, 2022 by the MUR.

References

Adi, A. O. and Çelebi, E. (2014). Classification of 20 news group with naive bayes classifier. In *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, pages 2150–2153. IEEE.

Albishre, K., Albathan, M., and Li, Y. (2015). Effective 20 newsgroups dataset cleaning. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 3, pages 98–101. IEEE.

Anthony, L. F. W., Kanding, B., and Selvan, R. (2020). Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051.*

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Bird, S., Klein, E., and Edward, L. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Bouza, L., Bugeau, A., and Lannelongue, L. (2023). How to estimate carbon footprint when training deep learning models? a guide and review. *Environmental Research Communications*, 5(11):115014.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Budennyy, S., Lazarev, V., Zakharenko, N., Korovin, A., Plosskaya, O., Dimitrov, D., Akhripkin, V., Pavlov, I., Oseledets, I., Barsola, I., Egorov, I., Kosterina, A., and Zhukov, L. (2022). eco2ai: Carbon emissions tracking of machine learning models as the first step towards sustainable ai. *Doklady Mathematics*, 106(Suppl 1):S118–S128.

Conversano, C. (2011). Interactive visualization in multiclass learning: integrating the sassc algorithm with klimt. *Computational Statistics*, 26:711–731.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20:273–297.

D'Andrea, E., Ducange, P., Lazzerini, B., and Marcelloni, F. (2015). Real-time detection of traffic from twitter stream analysis. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2269–2283.

Delachaux, B., Rebetez, J., Perez-Uribe, A., and Satizabal Mejia, H. F. (2013). Indoor activity recognition by combining one-vs.-all neural network classifiers exploiting wearable and depth sensors. In *Advances in Computational Intelligence: 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part II 12*, pages 216–223. Springer.

Dietterich, T. G. and Bakiri, G. (1994). Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, 2:263–286.

Dogan, T. and Uysal, A. K. (2020). A novel term weighting scheme for text classification: Tf-mono. *Journal of Informetrics*, 14(4):101076.

Eichelberger, R. K. and Sheng, V. S. (2013). Does one-against-all or one-against-one improve the performance of multiclass classifications? *Proceedings of the AAAI Conference on Artificial Intelligence*, page n. pag.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, 44(8):1761–1776.

Grün, B. and Hornik, K. (2011). topicmodels: An r package for fitting topic models. *Journal of statistical software*, 40(1):1–30.

Han, C., Fan, Z., Zhang, D., Qiu, M., Gao, M., and Zhou, A. (2021). Meta-learning adversarial domain adaptation network for few-shot text classification. *arXiv preprint arXiv:2107.12262*.

Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., and Pineau, J. (2020). Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43.

Hong, J.-H., Min, J.-K., Cho, U.-K., and Cho, S.-B. (2008). Fingerprint classification using one-vs-all support vector machines dynamically ordered with naive bayes classifiers. *Pattern Recognition*, 41(2):662–671.

Hosmer, D. W. and Lemeshow, S. (2000). *Applied Logistic Regression.* John Wiley & Sons, Ltd.

Hrib, I., Topal, O., Šturm, J., and Škrjanc, M. (2024). Measuring and modeling co2 emissions in machine learning processes. In *Information Society 2024, Proceedings of the 27th International Multiconference*, pages 7–11, Ljubljana, Slovenia. Jožef Stefan Institute.

Kumar, M. A. and Gopal, M. (2011). Reduced one-against-all method for multiclass svm classification. *Expert Systems with Applications*, 38(11):14238–14248.

Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning. arxiv 2019. *arXiv preprint arXiv:1910.09700.*

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Machine learning proceed-*

ings 1995, pages 331–339. Elsevier.

Lannelongue, L., Grealey, J., and Inouye, M. (2021). Green algorithms: quantifying the carbon footprint of computation. *Advanced science*, 8(12):2100707.

Li, T., Zhu, S., and Ogihara, M. (2006). Using discriminant analysis for multi-class classification: an experimental investigation. *Knowledge and information systems*, 10:453–472.

Lottick, K., Susai, S., Friedler, S. A., and Wilson, J. P. (2019). Energy usage reports: Environmental awareness as part of algorithmic accountability. *arXiv preprint arXiv:1911.08354*.

Murty, M. N. and Devi, V. S. (2011). *Bayes Classifier*, pages 86–102. Springer London, London.

Opitz, J. and Burst, S. (2021). Macro f1 and macro f1.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 40:211–218.

Rennie, J. D. and Rifkin, R. (2001). Improving multiclass text classification with the support vector machine. Technical report, Massachusetts Institute of Technology AI Memo 2001–026.

Rifkin, R. M. and Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141.

Romano, M., Contu, G., Mola, F., and Conversano, C. (2023). Threshold-based naïve bayes classifier. *Advances in Data Analysis and Classification*, pages 1–37.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sáez, J. A., Galar, M., Luengo, J., and Herrera, F. (2014). Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, 38:179–206.

Sarkar, D. (2019). Text classification. In *Text Analytics with Python: A Practitioners ́ Guide to Natural Language Processing*, pages 275–342. Springer.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green ai. *Communications of the ACM*, 63(12):54–63.

Sharma, S. and Mehrotra, D. (2018). Comparative analysis of multi-label classification algorithms. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 35–38. IEEE.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Student, S. and Fujarewicz, K. (2012). Stable feature selection and classification algorithms for multiclass microarray data. *Biology Direct*, 7.

Taghavinejad, S. M., Taghavinejad, M., Shahmiri, L., Zavvar, M., and Zavvar, M. H. (2020). Intrusion detection in iot-based smart grid using hybrid decision tree. In *2020 6th International Conference on Web Research (ICWR)*, pages 152–156. IEEE.

Verma, A., Singh, S. K., Sah, R. K., Misra, R., and Singh, T. (2024). Performance comparison of deep learning models for co2 prediction: Analyzing carbon footprint with advanced trackers. In *2024 IEEE International Conference on Big Data (BigData)*, pages 4429–4437. IEEE.

Xanthopoulos, P., Pardalos, P. M., Trafalis, T. B., Xanthopoulos, P., Pardalos, P. M., and Trafalis, T. B. (2013). Linear discriminant analysis. *Robust data mining*, pages 27–33.

Zdrojewska, A., Dutkiewicz, J., Jedrzejek, C., and Olejnik, M. (2019). Comparison of the novel classification methods on the reuters-21578 corpus. *Advances in Intelligent Systems and Computing*, 833:290 – 299.

Zhang, R. and Wang, W. (2011). Facilitating the applications of support vector machine by using a new kernel. *Expert Systems with Applications*, 38(11):14225–14230.