

Comprehensive analysis of machine learning classification metrics, algorithmic models, and the nuances of the neutral class

Oleksandr Tkachenko^{1,*†}, Anna Ilyenko^{2†}, Yelyzaveta Meleshko^{1†} and Oleksandr Ulichev^{1†}

¹Central Ukrainian National Technical University, Universytetskyi Ave. 8, Kropyvnytskyi, 25000, Ukraine

²Scientific Cyber Security Association of Ukraine, Mykhaila Donsia Str., 2A, Kyiv, 03161, Ukraine

Abstract

The rapid proliferation of social media networks has fundamentally transformed the global information ecosystem, creating a critical tension between the preservation of freedom of speech and the mitigation of systemic risks to public safety and national security. This paper establishes a novel theoretical mathematical model designed to simulate and optimize the legal regulation of information flows within digital platforms. By treating legal and structural regulation as an explicit control variable, the proposed framework quantifies its cascading effects on core societal performance indicators. The system dynamics incorporate a multi-layered matrix of contextual risk factors, including real-time disinformation indices, historical user behavioral patterns, spatial-temporal constraints, and geopolitical event proximity. Through simulation-based optimization, the model demonstrates how optimal regulatory thresholds can be algorithmically derived to prevent the dual failure modes of authoritarian over-censorship and unmitigated information disorder. While the framework provides a robust conceptual architecture for algorithmic policymaking, it is presently bounded by its theoretical nature; validation has been executed strictly through synthetic scenarios and simulated environments rather than empirical production data. Consequently, this study lays the formal mathematical groundwork for adaptive, risk-aware digital governance tools, outlining critical pathways for future empirical validation across diverse socio-cultural and international legal frameworks.

Keywords

social networks, legal informatics, information flows, disinformation models, freedom of expression, algorithmic regulation, public safety, cybersecurity governance, platform content moderation, risk-aware optimization

1. Introduction

The contemporary digital landscape is defined by the absolute centralization of the public square within commercial social media networks. These hyper-connected environments have transcended their original status as communication utilities to become the primary cognitive engines shaping public opinion, socio-political movements, and the geopolitical security paradigms of modern nation-states. By democratization of the information space, these platforms dramatically lower the barrier to entry for global discourse, empowering individuals and fostering unprecedented collective action. However, this architectural openness simultaneously exposes the socio-political fabric to severe structural vulnerabilities. The exact same algorithmic optimization loops that maximize user engagement inadvertently serve as highly efficient vectors for the distribution of targeted disinformation campaigns, malignant foreign propaganda, and radicalizing content that directly undermines public safety and threatens institutional stability.

This fundamental duality poses a profound dilemma for modern legal philosophy and constitutional jurisprudence. Traditional, static legislative frameworks are fundamentally unequipped to handle the

CMiGIN'26: Fifth International Workshop on Conflict Management in Global Information Networks, June 26–27, 2026, Krakow, Poland

*Corresponding author.

†These authors contributed equally.

✉ alexsunnik@gmail.com (O. Tkachenko); anna.ilyenko@npp.nau.edu.ua (A. Ilyenko); elismeleshko@gmail.com (Y. Meleshko); askin79@gmail.com (O. Ulichev)

ORCID 0009-0008-1721-3455 (O. Tkachenko); 0000-0001-8565-1117 (A. Ilyenko); 0000-0001-8791-0063 (Y. Meleshko); 0000-0003-3736-9613 (O. Ulichev)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

velocity, volume, and decentralized mutability of digitally mediated information warfare. Consequently, there is an urgent societal imperative to pioneer dynamic legal mechanisms capable of resolving this systemic tension. The core challenge lies in avoiding two equally catastrophic failure states: state-driven over-censorship that results in the chilling of legitimate democratic expression, and algorithmic laissez-faire that allows weaponized information operations to destabilize democratic processes. Achieving a sustainable equilibrium demands moving beyond binary qualitative debates toward quantitative, risk-aware computational models.

To address these gaps, this paper introduces a comprehensive mathematical framework that conceptualizes legal regulation as a dynamic control parameter. By establishing formal functional dependencies between regulatory intervention intensities and key societal metrics – specifically freedom of speech, public safety vectors, and national security indicators – the proposed model simulates the continuous interplay between state intervention, platform enforcement, and hostile information flows. Through the integration of complex contextual risk variables, this research aims to provide algorithmic decision-support tools for legislative bodies and platform architects. Ultimately, this study offers a rigorous, mathematically grounded methodology to optimize governance strategies in the digital era, ensuring robust national security posture without violating the foundational tenets of human rights.

2. Background

The contemporary strategic landscape of information warfare is characterized by the systematic weaponization of social media environments through automated and semi-automated coordination vectors [1]. In conditions of intense information confrontation, state and non-state actors deploy multi-layered malicious campaigns where hostile information influences are obfuscated within massive streams of benign, daily user-generated traffic [2]. Dynamically mitigating these algorithmic threats while strictly preserving the constitutional protections of legitimate democratic discourse requires national cybersecurity frameworks to transition from static, signature-based keyword filtering to automated, risk-aware machine learning engines operating in near real-time [3].

At the core of these defensive cognitive utilities is the application of multi-class machine learning classification models. Within highly complex, interconnected digital platforms, these classification algorithms act as automated decision-making sentinels tasked with parsing vast, unstructured environmental data fields – such as textual micro-blogs, metadata logs, and user interaction graphs – and converting them into actionable indicators of threat infrastructures. For instance, sophisticated smart-system architectures utilize automated classifiers to continuously assess competence, behavioral, and structural profile vectors to distinguish organic social movements from orchestrated foreign manipulation pipelines.

However, the operational environment of social networks presents extreme mathematical challenges for standard statistical estimators. Real-world datasets derived from active security monitoring are profoundly imbalanced; authentic instances of coordinated psychological operations or targeted disinformation constitute a minute fraction of a percent relative to standard social discourse [4]. In such heavily asymmetrical topologies, the traditional reliance on global evaluation parameters can be analytically disastrous. A naive optimization process focused strictly on maximizing crude evaluation metrics will inherently cause models to completely submerge critical minority threat vectors underneath the statistical dominance of majority classes, yielding a false sense of institutional security while failing catastrophically on novel, unseen vectors.

Furthermore, the continuous mapping of fluid human language into rigid categorical structures introduces profound semantic entropy. The ultimate benchmark testing for any automated content-moderation paradigm lies in its mathematical capacity to resolve highly ambiguous inputs, a challenge perfectly encapsulated by the intractability of the 'Neutral' class and complex mixed-polarity texts in sentiment analysis [5]. Objective, emotionless texts, defensive sarcasm, and relative entity-specific sentiment naturally gravitate extremely close to the decision hyperplanes of standard linear estimators [6]. Mismanaging this neutral territory either triggers authoritarian over-censorship due to false

positives or accelerates information disorder through systemic false negatives [7, 8].

Consequently, ensuring the scientific integrity and deployment viability of defensive information systems demands a rigorous, multi-dimensional audit of available computational architectures. This report systematically dissects the theoretical mechanics, optimization solvers, and algorithmic latency profiles of sixteen foundational classification estimators implemented within modern computational libraries [9]. By establishing an exact mathematical taxonomy of averaging methodologies and evaluating model behaviors under extreme class imbalances, this analysis provides the technical foundation necessary to optimize robust, context-aware automated governance strategies capable of securing the digital state square.

3. Foundational evaluation parameters in classification modeling

The empirical evaluation of a classification model requires an exact, mathematically rigorous understanding of how its algorithmic predictions align with objective reality, commonly referred to as the ground truth. This mathematical alignment is traditionally quantified through the generation of a confusion matrix, which cross-references predicted labels against true labels. At the core of all classification metrics lie four fundamental atomic units of measurement:

- **True Positives (TP)**, where the model correctly identifies the presence of a class;
- **True Negatives (TN)**, where the model correctly identifies the absence of a class;
- **False Positives (FP)**, denoting a Type I error where the model incorrectly asserts the presence of a class;
- **False Negatives (FN)**, denoting a Type II error where the model fails to detect a class that is genuinely present [10, 11].

In complex multi-class scenarios, these four parameters are isolated and calculated independently for each individual class before broader aggregation techniques are applied [10].

3.1. The imbalance paradox and the limitations of accuracy

Accuracy is universally recognized as the most intuitive and frequently utilized metric for initial model evaluation. Mathematically defined as the ratio of all correct predictions (both True Positives and True Negatives) divided by the total number of evaluated instances:

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

Accuracy offers a highly comprehensible snapshot of global performance. In a perfectly balanced dataset—where every class is represented by an equal number of samples—accuracy operates flawlessly. If an analyst desires an easily digestible metric for overall performance regardless of individual class dynamics in a balanced environment, accuracy is the premier choice [11]. Furthermore, when every data point in a set is assigned to one and only one class, the mathematical calculation for accuracy is strictly identical to the calculations for micro-averaged precision and micro-averaged recall [10].

However, the analytical utility of accuracy collapses entirely when subjected to heavily imbalanced datasets, a phenomenon highly prevalent in real-world scenarios such as medical diagnostics, spam filtration networks, and financial fraud detection systems [11, 4]. Consider a well-documented benchmark scenario involving a profoundly unbalanced dataset containing 28,481 total examples. Within this dataset, 28,432 examples belong to Class 0, meaning that Class 0 constitutes an overwhelming 99.8% of the entire data population, leaving only 49 examples (0.2%) belonging to Class 1 [4].

If a poorly configured algorithmic predictor simply learns to guess Class 0 for every single sample it encounters, it will achieve an astronomical accuracy score of 99.8% [11, 4]. To an untrained observer, the model appears to be functioning with near-perfect predictive capability. In reality, the algorithm has failed completely. It possesses zero predictive power for Class 1, entirely missing the anomaly it

was likely designed to detect. In practical deployment – where the cost of a false prediction for a rare disease or a fraudulent financial transaction is catastrophic – a predictor that continuously guesses the majority class provides negative operational value despite its mathematically high accuracy [4]. Therefore, while accuracy is often the primary metric observed, it is inherently misleading and highly dangerous when utilized in isolation within imbalanced environments [11].

3.2. Precision, recall, and the synthesized F1-score

To counteract the dangerous blind spots created by raw accuracy, data scientists rely on precision and recall to provide targeted, granular insights into a model's specific operational characteristics.

Precision functions as a measure of exactness or algorithmic caution. It answers a vital layman's question: "Of all the positive predictions my model made, how many of them were actually, truly positive?" [11]. Mathematically calculated as the number of True Positives divided by the sum of True Positives and False Positives, precision heavily penalizes the model for crying wolf [10, 11]:

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

Consequently, precision is the absolute metric of choice for problem spaces where False Positives are highly critical or incredibly expensive [11]. For example, in automated spam detection, a False Positive means routing a legitimate, highly important email into a junk folder, an action that must be minimized at all costs.

Conversely, recall functions as a measure of completeness or algorithmic sensitivity. It calculates the proportion of actual positive instances that the model successfully identified. Expressed mathematically as the number of True Positives divided by the sum of True Positives and False Negatives, recall heavily penalizes the model for missing critical events [10]:

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

Recall is the prioritized metric for problem spaces where False Negatives are critical [11]. In tumor detection algorithms, a False Negative means failing to identify a malignant growth. In this scenario, the system prioritizes extreme sensitivity, willingly accepting a higher rate of False Positives (which can be cleared by subsequent human review) to ensure that no true positive cases are ever missed.

Because a natural, inescapable mathematical trade-off exists between precision and recall – aggressively optimizing for precision will inherently degrade recall, and vice versa – the F1-Score is deployed as a sophisticated balancing mechanism. The F1-Score is defined as the harmonic mean of precision and recall [12, 11]:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

By utilizing the harmonic mean rather than a simple arithmetic average, the F1-Score severely punishes models that exhibit massive disparities between their precision and recall values. It collapses these two divergent dimensions into a single, comprehensive scalar number, making it exceptionally useful when precision and recall must be evaluated equally or when an optimal equilibrium is required to satisfy conflicting business objectives [11].

3.3. Multi-class averaging methodologies: macro, micro, and weighted

When a classification paradigm extends beyond a simple binary choice into a multi-class environment (e.g., classifying images into airplanes, boats, and cars), evaluating global performance requires aggregating the precision, recall, and F1-scores across all distinct categories. The specific methodology chosen to compute these averages fundamentally alters the interpretation of the model's success and must be aligned with the underlying data distribution [11].

Table 1
Comparison of Multi-Class Averaging Methodologies

Methodology	Computational Mechanism	Primary Analytical Implication	Optimal Deployment Scenario
Micro-Averaging	Aggregates the global, absolute counts of <i>TP</i> , <i>FP</i> , and <i>FN</i> across every single class before computing metrics [10].	Bestows strictly equal weight to every single instance, entirely ignoring the overarching class label [10].	Perfectly balanced datasets, or maximizing total correct predictions [10, 11].
Macro-Averaging	Calculates the performance metric independently for each class, then computes the unweighted arithmetic mean [10].	Bestows strictly equal weight to every single class, regardless of how many instances belong to it [10].	Environments where minority classes are deemed equally as important as majority classes [10].
Weighted-Averaging	Computes the metric for each class, but aggregates them using a mean weighted by class support [10, 11].	Operates similarly to macro-averaging, but ensures heavily populated classes dictate the final value [10, 11].	Highly imbalanced datasets where the metric should reflect natural class distribution [10, 11].

To truly understand the profound difference between these averaging methodologies, we return to the previously established imbalanced dataset where Class 0 holds 99.8% of the data and Class 1 holds 0.2% [4]. If the predictor blindly guesses Class 0, it achieves a high score for Class 0 and a near-zero score for Class 1.

If we utilize weighted averaging, the final metric is calculated as:

$$\text{Metric}_{\text{weighted}} = 0.998 \cdot \text{Score}_{\text{class 0}} + 0.002 \cdot \text{Score}_{\text{class 1}}$$

Because Class 1’s terrible score is multiplied by 0.002, its failure is almost completely masked, resulting in an artificially brilliant final score that turns out to be roughly 1.0 due to the extreme class imbalances [4].

However, if we deploy macro-averaging, the formula ignores the 99.8% population size. The calculation becomes a straightforward:

$$\text{Metric}_{\text{macro}} = 0.5 \cdot \text{Score}_{\text{class 0}} + 0.5 \cdot \text{Score}_{\text{class 1}}$$

Because the model’s score for Class 1 is virtually zero, the macro-average plummets to roughly 0.5, accurately exposing the model’s underlying predictive failure [4].

The Macro F1 score functions explicitly as the harmonic mean of macro-precision and macro-recall [12]. Theoretical explorations of these multi-class metrics reveal subtle complexities regarding chance baseline correction. While virtually all evaluation metrics (with the explicit exceptions of raw Accuracy and Weighted F1) demonstrate some degree of chance baseline correction, strict chance baseline correction is not an inherent architectural feature of the Macro F1 score [12]. Complete, class-count independent chance correction is truly only achieved by highly specialized statistical functions like Matthew’s Correlation Coefficient (*MCC*) and Cohen’s Kappa [12].

Interestingly, analytical literature suggests that Macro Recall and Accuracy operate as highly complementary forces; Macro Recall can be conceptually understood and utilized as a strict prevalence-calibrated version of Accuracy, allowing for clear interpretability while accounting for shifting baseline distributions [12].

3.4. Probabilistic predictions and strictly consistent scoring rules

While metrics like precision and recall evaluate absolute, discrete point predictions (e.g., forcing the model to issue a definitive “Yes” or “No”), advanced machine learning often requires predicting a complete probability distribution [13]. In reality, there is rarely a perfect, deterministic function $Y = g(X)$ mapping features to outcomes; instead, there is a probability distribution F of the random variable Y [13].

When a classifier is deployed for probabilistic prediction, the focus shifts to evaluating the chosen functional of that distribution, such as the expected value (mean), median, or specific quantiles conditionally based on the features X [13]. Once the target functional is established, evaluation must rely on a strictly consistent scoring function [13]. A strictly consistent scoring function guarantees that the metric is mathematically aligned with measuring the true probabilistic distance between the predictions (y_{pred}) and the true target functional observed in Y (y_{true}) [13].

In the context of classification, these strictly proper scoring rules coincide perfectly with strictly consistent scoring functions [13]. These functions act as an algorithmic “truth serum”; they mathematically guarantee that “truth telling” – meaning the model outputting its true learned probability rather than gaming the metric – is the optimal strategy in expectation [13]. Consequently, it is an industry best practice to utilize the identical strictly consistent scoring function for both the loss function during model training and the final metric during model comparison [13].

To visually evaluate the efficacy of these probabilistic classifiers, researchers utilize calibration curves, also known as reliability diagrams [14]. These diagrams are generated by dividing the model’s predicted probabilities into specific bins, and then plotting the mean predicted probability within each bin directly against the actual, observed frequency of positive occurrences in the test dataset [14]. If a model states that an event has an 80% chance of occurring, it should ideally occur exactly 80% of the time. Calibration curves immediately reveal if a multi-class model is suffering from systemic overconfidence or underconfidence, allowing for subsequent threshold tuning for class prediction. This is critical for downstream decision-making, where the continuous output of `predict_proba` must be discretized into a single binary action, such as utilizing a weather model’s probability distribution to decide whether or not to carry an umbrella [13].

4. Detailed taxonomy of 16 classification models

The modern data scientist is confronted with a staggering array of algorithmic estimators, making the selection of the correct model the most challenging aspect of solving a machine learning problem [15]. Choosing the correct algorithm requires carefully navigating the problem statement, saving both computational time and financial resources [15]. This decision is often guided by a heuristic flowchart: determining if the dataset contains more than 50 samples (the minimum threshold for algorithmic validity), identifying if the goal is categorizing labeled data, and querying if the dataset exceeds massive thresholds like 100,000 samples [15].

The following section provides an exhaustive, theoretical analysis of sixteen distinct classification algorithms frequently deployed via modern computational libraries such as scikit-learn. These models range from elementary probabilistic estimators to vast, ensemble-based neural architectures, each characterized by unique decision boundaries spanning piece-wise constant lines, linear separations, quadratic curves, and Voronoi tessellations [9, 16].

4.1. Logistic regression

Logistic Regression is the absolute foundational bedrock of probabilistic classification. It is a generalized linear classifier that maps any real-valued input vector into a highly bounded probability ranging exactly between 0 and 1 by utilizing the standard sigmoid hypothesis [6]. During training, it minimizes the log-probability (log-loss) across the dataset [6, 17]. For instance, it can successfully determine a

university applicant's chance of admission based on an array of historical exam scores by plotting a clean probability curve [6].

The profound power of Logistic Regression within environments like scikit-learn stems from its access to a diverse suite of specialized mathematical solvers. The algorithm can be optimized using solvers such as 'newton-cg', 'lbfgs', 'liblinear', 'sag', and 'saga' [17]. The 'sag' solver, which stands for Stochastic Average Gradient descent, is particularly notable for its speed on large datasets [17]. These solvers dictate precisely how the gradient descent is computed and enable various forms of L_1 and L_2 regularization to prevent overfitting [17].

4.2. Stochastic gradient descent (SGD) classifier

The SGD Classifier is fundamentally distinct; it is not a standalone mathematical algorithm, but rather a generalized, highly efficient learning routine that utilizes stochastic gradient descent to iteratively fit linear classifiers [15, 17]. It operates by minimizing various convex loss functions [15]. If a developer explicitly sets the loss parameter to 'log', the SGD Classifier mechanically transforms into a Logistic Regression model, utilizing identical loss functions but a fundamentally different iterative solver [17]. If the loss is set to 'hinge', it mimics a linear Support Vector Machine [15].

The defining technological advantage of the SGD Classifier is its ability to bypass standard batch loading. It can be trained continuously via the `partial_fit()` method, making it the undisputed standard for online training, active training, or processing big data that exceeds system RAM limits [17]. Because it updates parameters using single points (or mini-batches) rather than evaluating the entire dataset simultaneously, it acts as an approximation algorithm that gradually converges on the optimal solution as the number of data points increases [17]. By allowing developers to tune hyperparameters like learning rates across multiple iterations, training an SGD Classifier heavily resembles the intricate epoch-based training loops of deep neural networks [17].

4.3. Ridge classifier

The Ridge Classifier elegantly adapts the mechanics of Ridge Regression to solve classification problems. It achieves this by mathematically converting binary target labels into continuous values of -1 and 1, and subsequently solving the problem using linear regression techniques equipped with L_2 regularization [18]. The model forms its decision boundary based on whether the continuous prediction falls above or below zero.

The primary allure of the Ridge Classifier is its breathtaking computational efficiency in specific architectural scenarios. When deployed against a multi-class problem with a massive number of distinct classes, it can compute the highly complex inverse projection matrix $(X^T X)^{-1} X^T$ exactly once [18]. This caching of the projection matrix allows the Ridge Classifier to operate significantly faster than standard Logistic Regression, particularly when dealing with the classification of massive text documents utilizing sparse feature arrays [18].

4.4. Least squares support vector machine (LS-SVM)

The Least Squares Support Vector Machine is an algorithmic variant closely related to the Ridge Classifier. Traditional SVMs rely on solving complex, resource-intensive convex quadratic programming problems to identify the optimal classifying hyperplane. The LS-SVM drastically simplifies this constraint by reforming the optimization objective into a straightforward set of linear equations [18]. Within standard computational frameworks, the Ridge Classifier is mathematically recognized and referred to as an LS-SVM operating strictly with a linear kernel [18].

4.5. Linear support vector classification (linear SVM)

A standard Linear SVM seeks to identify the optimal hyperplane that perfectly separates distinct classes while mathematically maximizing the spatial margin between the nearest data points of those classes

(the support vectors) [9]. As previously established, this exact linear decision boundary can be replicated by an SGD Classifier utilizing the hinge loss function [15].

While linear models seem overly simplistic, they are theoretically paramount in high-dimensional feature spaces. In high-dimensional hyperspace, data points are significantly easier to separate linearly; thus, the structural simplicity of a Linear SVM prevents the model from mapping noise, frequently leading to substantially better generalization on unseen testing data compared to wildly complex, non-linear classifiers [9].

4.6. Radial Basis Function (RBF) Support Vector Machine

When data exists in a configuration that is mathematically impossible to separate with a straight line (such as concentric circles or intertwined moons), the RBF SVM is deployed [9]. This algorithm utilizes the ‘kernel trick’ via the Radial Basis Function, mathematically projecting the low-dimensional data into a theoretical infinite-dimensional space where linear separation becomes possible. When projected back down to the original dimensions, the decision boundary appears as highly non-linear, contoured regions tightly encapsulating the data classes [9].

4.7. Kernel approximation classifier

Despite the theoretical brilliance of RBF SVMs, calculating exact kernel matrices becomes catastrophically slow and memory-intensive as datasets grow. For massive datasets exceeding sample sizes of 100,000 instances, standard kernelized algorithms fail [15]. To solve this, developers employ Kernel Approximation. This technique utilizes mathematical functions to approximate the feature mappings of complex kernels. By transforming the data into this approximated higher-dimensional space, developers can then apply lightning-fast linear models (like the SGD Classifier) to the transformed data, yielding the benefits of non-linear decision boundaries without the crippling computational overhead [15].

4.8. K-nearest neighbors (KNeighborsClassifier)

The K-Nearest Neighbors (KNN) algorithm abandons mathematical equations in favor of pure proximity mapping [9, 16]. KNN is a non-parametric, lazy-learning classifier; it performs zero computation during the training phase, opting instead to simply memorize the entirety of the training dataset. When asked to classify a novel data point, it scans the entire memorized dataset, calculates the absolute spatial distance to every single point, identifies the k closest points, and assigns a label based on a majority vote of those neighbors. While training is instantaneous, the inference time required to calculate distances for every new prediction can become a paralyzing bottleneck in production environments [16].

4.9. Gaussian naive Bayes (GaussianNB)

Gaussian Naive Bayes is a lightning-fast probabilistic classifier anchored entirely in Bayes’ Theorem [9]. It makes the “naive” assumption that every single feature in the dataset is strictly, conditionally independent of every other feature – an assumption that is almost universally false in real-world data. Despite this mathematical flaw, GaussianNB forces continuous data to fit into standard Gaussian (normal distributions), allowing it to rapidly calculate probabilities. Because it requires very little training data to estimate the necessary means and variances, it serves as a highly robust, easily deployable baseline classifier, particularly in high-dimensional text classification spaces [9].

4.10. Gaussian process classifier

Contrasting sharply with rigid linear models, the Gaussian Process Classifier is a non-parametric, deeply Bayesian methodology [9]. Rather than calculating a single optimal set of weights to form a decision boundary, it assumes a prior probability distribution over an infinite number of possible functions that could fit the data. Utilizing complex kernels (such as the RBF kernel), it evaluates the

training data to calculate a posterior distribution [9]. The output is not merely a rigid class label, but a highly sophisticated, mathematically sound probabilistic forecast accompanied by absolute measures of algorithmic uncertainty [9].

4.11. Discriminant analysis (general)

Standard Discriminant Analysis is a generative classification model. It operates under the fundamental assumption that the data for different classes are generated by fundamentally different Gaussian distributions [16]. During training, it estimates the mean and covariance of each class's distribution. To classify a new instance, it calculates the probability that the instance was generated by each specific distribution, assigning it to the most probable origin. Depending on the covariance structures, this model generates highly effective piece-wise constant or linear decision surfaces [16].

4.12. Quadratic discriminant analysis (QDA)

Quadratic Discriminant Analysis is a powerful structural extension of standard linear Discriminant Analysis. Linear models assume that all classes in the dataset share the exact same covariance matrix, severely limiting the shapes of the decision boundaries [9]. QDA completely abandons this constraint, mathematically allowing each individual class to possess its own distinct, independent covariance matrix [9]. This extreme flexibility permits the generation of complex, quadratic decision boundaries capable of wrapping around highly irregular data clusters. However, calculating independent covariance matrices requires a massively increased number of parameters, making QDA highly susceptible to overfitting if the training dataset is not sufficiently large [9].

4.13. Decision tree classifier

The Decision Tree Classifier models human-like logic by recursively partitioning the feature space into distinct Voronoi-like tessellations via a series of strictly binary algorithmic splits [9]. Each node in the tree represents a mathematical condition based on a single feature, creating a heavily segmented, piece-wise constant decision boundary [16]. While Decision Trees are universally beloved for their absolute interpretability and lack of required data preprocessing, deep, unconstrained trees are notorious for suffering from extreme variance, routinely memorizing the training data noise and failing entirely to generalize to test data [9].

4.14. Random forest classifier

To surgically eradicate the high variance inherent in solitary decision trees, the Random Forest Classifier utilizes a massive ensemble bagging methodology [9]. During the training phase, it constructs hundreds or thousands of independent decision trees. Crucially, each individual tree is trained on a completely random subset of the training data, and is only permitted to split nodes utilizing a random subset of the available features. During inference, every tree generates a prediction, and the forest outputs the majority vote. This randomization inherently decorrelates the trees, creating an incredibly robust classifier highly resistant to background noise and overfitting [9].

4.15. AdaBoost classifier

Adaptive Boosting (AdaBoost) represents a fundamentally different philosophical approach to ensemble learning [9]. Instead of building independent, highly complex trees simultaneously, AdaBoost builds incredibly weak, shallow classifiers (often decision “stumps” with only one split) sequentially. Each subsequent model built evaluates the errors of the preceding model, actively assigning massively higher mathematical weights to the specific instances that were previously misclassified [9]. This iterative, self-correcting weighting process forces the algorithmic boundary to continuously adapt until optimal spatial separation is achieved across the entire dataset [9].

4.16. Multi-layer perceptron (MLPClassifier)

The Multi-Layer Perceptron is a foundational feedforward artificial neural network architecture deployed for classification [9]. It consists of an input layer, one or more hidden layers equipped with non-linear activation functions (such as ReLU or Tanh), and a final output layer. While a neural network severely restricted to a single layer and a single neuron using logistic loss devolves exactly into a standard logistic regression algorithm [17], the addition of deep hidden layers allows the MLP to act as a universal function approximator, capable of learning and drawing exceptionally complex, highly non-linear decision boundaries across virtually any data topology [9].

4.17. Auxiliary classification mechanism: thresholded linear regression

While intrinsically designed as a continuous regression algorithm, standard Linear Regression can be mechanically mutilated to serve as a rudimentary binary classifier [16]. By fitting the continuous data points and manually establishing a rigid mathematical threshold (for example, arbitrarily declaring that any continuous output greater than 0.0 is a positive class prediction, and anything less is negative), the regressor is forced to act as a classification boundary [16]. However, because Linear Regression fundamentally optimizes its weights to minimize the mean squared error rather than optimizing a probabilistic classification boundary, it is notoriously sensitive to spatial outliers and is generally considered vastly inferior to dedicated classifiers like Logistic Regression in almost all applications.

5. System architecture: parallelization, inference latency, and deployment frameworks

The theoretical brilliance of these sixteen mathematical models is frequently bottlenecked by the harsh computational realities of their physical implementation. When deploying a machine learning model, researchers must balance two primary characteristics: accuracy (how mathematically proficient the model is on test data) and satisficing (the absolute time required by the model to process an input and generate a prediction) [19].

In modern deployment environments, inference time – often defined as algorithmic latency – is paramount. For instance, consider a quantitative firm executing high-frequency algorithmic trading, where a model must predict a buy or sell action in mere nanoseconds to capitalize on transient market phenomena [19]. Similarly, Smart Home platforms require systems that can instantly recognize changes in indicator data and rapidly alter complex control strategies to neutralize internal or external disruptive actions, managing electrical systems via interconnected IoT devices and automated Telegram bots.

A critical architectural revelation concerning popular machine learning libraries such as scikit-learn, Keras, PyTorch, and LightGBM is that they are fundamentally designed and highly optimized to execute batch predictions [19]. Leveraging advanced linear algebra processing architectures, these libraries are exponentially faster at processing 1000 data records simultaneously than they are at processing a single record 1000 consecutive times [19]. Consequently, deploying a standard `predict()` function within an active, real-time API loop – where requests arrive individually rather than in batches – incurs a catastrophic latency overhead, rendering the models too slow for high-frequency applications [19]. To optimize inference times for individual predictions, developers must actively bypass the library's batch-processing architecture, often writing highly customized classes equipped with dedicated `predict_single` methods that strip away the linear algebra overhead [19].

6. Benchmark performance and the intractable challenge of the 'neutral' class

While evaluating algorithmic latency and mapping the mathematical contours of decision boundaries provide a strict structural understanding of classification modeling, the ultimate benchmark of any

system lies in its ability to parse highly ambiguous, unstructured human data. This challenge is perfectly encapsulated in the field of Natural Language Processing, specifically within Sentiment Analysis, where assigning polarity to human text reveals the absolute functional limits of rigid algorithmic classification systems [5].

6.1. The mathematical flaws of binary and multi-class sentiment taxonomies

Historically, sentiment analysis architectures forcefully binned the entirety of human linguistic expression into a rigid binary taxonomy: Positive and Negative [20]. This binary constraint is mathematically violent; it completely removes any spatial volume for neutrality, forcing algorithms to classify utterly objective, emotionless features into highly polarized categories [20]. This binary restriction systematically triggers catastrophic algorithmic overfitting [20]. If a purely neutral word happens to arbitrarily appear more frequently alongside negative tokens within a specific randomized training batch, the algorithm has no choice but to blindly assign a heavy negative mathematical weight to that neutral word, permanently corrupting its decision matrix [20].

To circumvent this mathematical collapse, researchers typically expand the model into a multi-class classification system by introducing a 'Neutral' class [8, 5]. Empirical benchmarks demonstrate that factoring neutral polarity into the training corpus noticeably improves system stability. In benchmark analyses assessing absolute algorithmic failure rates, researchers isolated 48 specifically misclassified phrases; of those 48 errors, a staggering 37 were strictly neutral in nature [21]. This massive error rate was directly attributed to the failure of the engineers to utilize a "neutral corpus" when training the classifier [21]. Academic titans such as Professors Koppel and Schler definitively proved that incorporating neutral examples is paramount for learning accurate sentiment boundaries, providing empirical evidence that algorithms like Support Vector Machines and Maximum Entropy frameworks heavily benefit from the inclusion of the third class [20, 8].

6.2. The illusion of multi-class nuance and semantic entropy

However, despite these mathematical improvements, simply appending a third 'Neutral' class proves woefully insufficient for capturing the erratic, highly fluid nuance of human emotion [5]. The multi-class paradigm fundamentally fractures when confronted with real-world linguistic complexities, demonstrating why viewing sentiment analysis as a classification problem is inherently flawed [5].

6.2.1. The conflation of "no sentiment" and "mixed sentiment"

A fatal architectural flaw in modern sentiment modeling is the algorithmic conflation of absolute neutrality with conflicted sentiment [8]. Consider a ubiquitous review paradigm: "The service was good, but the food was terrible!" [5]. This sentence does not belong in the Positive class, nor does it belong in the Negative class. Crucially, it does not belong in the Neutral class either. True neutrality mathematically implies a completely objective statement utterly devoid of emotional polarity [8]. The example phrase contains two extreme, diametrically opposed emotional polarities bound within the same text. If researchers blindly assign mixed sentiments to the 'Neutral' class, it injects massive entropy into the training data, heavily degrading the algorithm's overall accuracy [8]. Properly resolving mixed sentiment requires abandoning unified multi-class boundaries entirely, relying instead on deploying an array of highly specific, parallel binary classifiers or migrating to deeply complex aspect-oriented sentiment analysis, which isolates specific entities and maps independent vectors to each [8, 5].

6.2.2. Relative polarity and the sarcasm barrier

Human sentiment is rarely applied globally across a text string; it is intensely relative to specific entities [5]. A single text array may exhibit fierce, glowing positivity toward one subject while simultaneously expressing deep hostility toward another entity mentioned mere tokens away [5]. Furthermore, sarcasm acts as a cryptographic barrier to machine learning [5]. Sarcastic structures actively utilize highly

positive linguistic tokens to convey aggressive negative intent. Standard algorithmic token-counting methodologies – which operate under the naive assumption that a preponderance of positive words inherently equals a positive classification – are rendered completely useless and obsolete in the face of sarcastic syntax [22].

6.2.3. Inter-annotator collapse and ground truth instability

Supervised machine learning classification relies entirely on absolute Ground Truth labels. However, attempting to objectively define emotion introduces profound, systemic human error. Multiple human annotators frequently disagree entirely on the proper label for a specific text, destroying the integrity of the ground truth [5]. Even more problematic is the documented phenomenon of self-agreement failure; an individual annotator’s subjective perception of a text’s sentiment may shift drastically over time, leading them to actively contradict their own previous annotations [5]. As the taxonomy of classes increases from a simple binary setup to a multi-class environment, this human-in-the-loop entropy scales exponentially, injecting a permanent layer of noise directly into the foundational training data, making perfect classification a mathematical impossibility [5]. Indeed, empirical analyses of hand-labeled Twitter data sets revealed that roughly 60% of all generated data contained no definitive sentiment, completely overwhelming the polarized classes [8].

6.3. Boundary manipulation and artificial accuracy inflation

Because mathematically neutral texts exist extremely close to the hyperplane boundaries of standard binary classifiers, researchers frequently ignore them, operating under the highly flawed assumption that there is less statistical value to learn from them [20]. To process these texts, advanced pipelines occasionally deploy hierarchical classification, utilizing a primary classifier to first determine if a text is subjective or purely objective (neutral), and deploying a secondary classifier exclusively to determine the polarity of the subjective texts [7].

An intriguing, highly controversial anomaly regarding benchmark performance frequently emerges during the testing phases of these models. If an engineering team definitively knows beforehand that absolutely no neutral documents exist within their unseen testing dataset, they can employ boundary manipulation. They can arbitrarily force the algorithm to map all internal predictions generated as ‘Neutral’ directly into either the Positive or Negative class [7]. Because a highly consistent, baseline percentage of documents (roughly 1.5% to 3%, depending on the specific dataset) are routinely falsely classified as neutral by the algorithm, forcefully remapping these predictions to the binary categories can artificially inflate the model’s overarching accuracy by an additional 1% to 1.5% [7]. However, this is largely an academic illusion; in actual real-world deployments where incoming text streams cannot be pre-screened for neutrality, forcing texts strictly into binary boundaries incurs a massive, permanent cost to overall operational accuracy [8].

Ultimately, analyzing human emotion via text is highly erratic, and attempting to map it perfectly to mathematical classifications is fundamentally flawed [22, 5]. Sentiment is rarely as useful as contextual, aspect-oriented analysis, which abandons the idea of overall polarity in favor of identifying precise interactions between subjects and predicates [22, 5].

7. Empirical benchmark analysis: a practical case study

An empirical evaluation of the sixteen classification models across a dataset of 2,400 instances explicitly exposes the theoretical vulnerabilities previously discussed. The test set distribution is notably imbalanced, comprising 946 Negative, 433 Neutral, and 1,021 Positive instances.

Reviewing the classification reports, the most glaring algorithmic failure across almost all models is the catastrophic inability to predict the Neutral class. Foundational models such as Logistic Regression, the Ridge Classifier, AdaBoost, and Linear SVM recorded an F1-score of exactly 0.00 for the Neutral class. This confirms the hypothesis that neutral sentiment is the primary source of classification difficulty.

Because neutral text frequently lacks polarizing keywords or contains conflicting mixed sentiments, it sits dangerously close to the hyperplanes of binary classifiers, forcing the algorithms to erroneously assign them to the more dominant Positive or Negative classes.

When evaluating computational efficiency against predictive power, the Ridge Classifier demonstrated extreme speed, requiring a mere 0.0208 seconds to train. This exceptional speed is mathematically justified, as the algorithm computes the complex inverse projection matrix exactly once for multi-class problems, allowing it to bypass the iterative overhead of standard gradient descent. Conversely, the Support Vector Machine utilizing the RBF Kernel achieved the highest absolute Accuracy (0.4229) but incurred a massive computational penalty, requiring over 38 seconds for training and 2.6 seconds for inference. This demonstrates the steep latency costs associated with calculating complex non-linear kernel matrices, which can be detrimental in real-time inference environments.

Interestingly, the Decision Tree Classifier achieved the highest Macro F1-Score (0.3233) despite possessing one of the lowest overall Accuracy scores (0.3567). Because it managed to mathematically isolate a small fraction of the Neutral class (achieving a recall of 0.18, compared to 0.00 for most linear models), its unweighted macro-average was elevated.

However, the overarching reality that all evaluated models hovered around a Macro F1-Score of 0.30 indicates a severe algorithmic underfitting problem. It proves that simple feature mapping without deep contextual understanding or specialized handling of the neutral corpus is entirely insufficient for robust sentiment analysis.

8. Conclusions

The rigorous architecture of machine learning classification is governed by a highly delicate, mathematically tense interplay between strict evaluation metrics, algorithmic processing efficiency, and the inherent, chaotic complexity of the real-world data being mapped. While foundational metrics such as accuracy provide a highly comprehensible surface-level understanding, their deployment in imbalanced environments is analytically disastrous. Such datasets necessitate the targeted deployment of precision and recall parameters, intricately bound together by meticulously calculated macro and weighted F1-scores to mathematically guarantee that critical minority classes are not subsumed by the statistical dominance of majority classes.

The comprehensive evaluation of the sixteen primary algorithmic classifiers unequivocally highlights that there is no universal, silver-bullet mathematical solution. The algorithms range dynamically from the highly interpretable, memory-intensive mapping of K-Nearest Neighbors to the incredibly scalable, mathematically elegant gradient-based optimizations of the Stochastic Gradient Descent Classifier. Modern system architects must continuously weigh the immense predictive benefits of probabilistic forecasts and non-linear RBF kernels against the strict, unforgiving constraints of computational inference times. To maintain viability in highly responsive environments like IoT infrastructures and high-frequency trading platforms, engineers must actively dismantle standard batch-processing paradigms and aggressively leverage complex Python multiprocessing architectures to achieve true mathematical parallelization.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] O. Tkachenko, A. Iliencko, O. Ulichev, Y. Meleshko, L. Halata, Modern studies of information influence in social networks, *Cybersecurity: Education, Science, Technique* 3 (2025) 120–140. URL: <https://doi.org/10.28925/2663-4023.2025.27.716>. doi:10.28925/2663-4023.2025.27.716.

- [2] O. S. Ulichev, Model and Methods of Spreading Information Influences in Social Networks in Conditions of Information Confrontation, Candidate of technical sciences dissertation, National Aviation University, Kyiv, 2021. Specialty 21.05.01 "Information Security of the State", 170 p.
- [3] O. Tkachenko, A. Iliencko, O. Ulichev, Y. Meleshko, O. Smirnov, Legal foundations of information influence dissemination in social networks, *Cybersecurity: Education, Science, Technique* 2 (2024) 170–188. URL: <https://doi.org/10.28925/2663-4023.2024.26.685>. doi:10.28925/2663-4023.2024.26.685.
- [4] Stack Exchange, Macro average and weighted average meaning in classification_report, 2026. URL: <https://datascience.stackexchange.com/questions/65839/macro-average-and-weighted-average-meaning-in-classification-report>.
- [5] Quora, How do you accurately classify neutral examples in sentiment analysis, multi-class classification or by using confidence of binary classification, 2026. URL: <https://www.quora.com/>.
- [6] S. Sawtelle, Logisticregression vs. sgdcclassifier, 2026. URL: <https://sdsawtelle.github.io/blog/output/week3-andrew-ng-machine-learning-with-python.html>.
- [7] Datumbox, The importance of neutral class in sentiment analysis, 2026. URL: <https://blog.datumbox.com/the-importance-of-neutral-class-in-sentiment-analysis/>.
- [8] A. Martin, et al., Sentiment analysis and the neutral class, 2012. *Sentiment Analysis Research Review*.
- [9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, *Journal of Machine Learning Research* 12 (2011) 2825–2830.
- [10] Evidently AI, Accuracy, precision, and recall in multi-class classification, 2026. URL: <https://www.evidentlyai.com/classification-metrics/multi-class-metrics>.
- [11] P. Kashyap, Understanding precision, recall, and f1 score metrics, 2026. URL: <https://medium.com/@piyushkashyap045/understanding-precision-recall-and-f1-score-metrics-ea219b908093>.
- [12] J. Opitz, A closer look at classification evaluation metrics and a critical reflection of common evaluation practice, *arXiv preprint arXiv:2404.16958* (2024). URL: <https://arxiv.org/abs/2404.16958v2>.
- [13] Scikit-learn, Probability calibration, 2026. URL: <https://scikit-learn.org/stable/modules/calibration.html>.
- [14] Scikit-learn, Comparison of calibration of classifiers, 2026. URL: https://scikit-learn.org/stable/auto_examples/calibration/plot_compare_calibration.html.
- [15] S. Mohelay, How to choose the right machine learning algorithm, 2026. URL: <https://samyakmohelay.medium.com/how-to-choose-the-right-machine-learning-algorithm-7adbe8238638>.
- [16] Scikit-learn, Supervised learning, 2026. URL: https://scikit-learn.org/stable/supervised_learning.html.
- [17] Stack Overflow, Sgdcclassifier vs logisticregression with sgdc solver in scikit-learn library, 2026. URL: <https://stackoverflow.com/questions/43961225/sgdcclassifier-vs-logisticregression-with-sgd-solver-in-scikit-learn-library>.
- [18] Scikit-learn, Linear models, 2026. URL: https://scikit-learn.org/stable/modules/linear_model.html, scikit-learn 1.8.0 Documentation.
- [19] H. S. Bhatti, Speeding up inference time of scikit-learn models, 2026. URL: <https://hamxahbhatti.medium.com/speeding-up-inference-time-of-scikit-learn-models-728a88f10b5>.
- [20] M. Koppel, J. Schler, The importance of neutral examples for learning sentiment, *Computational Intelligence* 22 (2006) 100–109.
- [21] J. Maguire, The difficulties of sentiment analysis and what are the solutions?, 2016. URL: <https://jamiemaguire.net/index.php/2016/05/24/the-difficulties-of-sentiment-analysis-and-what-are-the-solutions/>.
- [22] Making Noise and Hearing Things, The trouble with sentiment analysis, 2022. URL: <https://makingnoiseandhearingthings.com/2022/04/19/the-trouble-with-sentiment-analysis/>.