Towards Better Product Quality: Identifying Legitimate Quality Issues using NLP & Machine Learning.

Case Study Intelligent Systems in Production

GROUP 7

Alen Sam	**22402597**
Sachin Saji	**12401209**
Sandeep Nidheesh	**12503425**
Vijeesh K S	**12402310**

SUPERVISED BY

PROF.DR. HAMIDREZA HEIDARI.

Deggendorf Institute of Technology,
Cham Intelligent Systems Case Study
Winter Semester 2025

18 JANUARY, 2026

Table of Contents

Automated Classification of Quality vs Non-Quality Customer Complaints

Abstract

The rapid digitization of consumer-brand interactions has resulted in an unprecedented influx of unstructured textual feedback, necessitating sophisticated computational frameworks for efficient triage. This research report details the development and implementation of an end-to-end Natural Language Processing (NLP) and Machine Learning (ML) system designed to differentiate between "Quality Issues" (machine or product faults) and "Non-Quality Issues" (user-end errors or misunderstandings). Utilizing a balanced dataset of 1,000 narratives sourced from Kaggle and the Consumer Complaint Database, the study investigates the efficacy of four distinct classifiers: Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machines (SVM). Through rigorous preprocessing, TF-IDF vectorization, and hyperparameter optimization via GridSearchCV, the SVM model emerged as the superior estimator, achieving a classification accuracy of 98.5% and an F1-score of 98%. A critical component of this research is the implementation of a confidence-based routing logic where narratives yielding a prediction confidence below 60% are redirected to human analysts, ensuring a "human-in-the-loop" safeguard for ambiguous communications. The final model is serialized using Joblib and deployed via a Streamlit web application, providing a functional interface for real-time industrial application. The findings underscore the potential of supervised learning to significantly reduce operational overhead by preventing unnecessary technician deployment, thereby enhancing both customer satisfaction and organizational profitability.

1. Introduction

1.1 Background

In the contemporary industrial landscape, the management of customer complaints has transitioned from a reactive administrative task to a proactive strategic function. Organizations across diverse sectors, including financial services, telecommunications, and consumer electronics, are increasingly reliant on their ability to interpret and act upon large volumes of unstructured textual data. Traditionally, the triage of these complaints—identifying whether a reported issue stems from a technical product failure or a user's operational error—has been a manual process. This manual approach is fraught with challenges, including significant labor costs, slow response times, and the inherent subjectivity of human analysts, which often leads to inconsistent categorization. The evolution of Artificial Intelligence (AI) and Machine Learning (ML) has provided a robust alternative to these traditional methods. Automated complaint classification systems leverage linguistic patterns to route grievances to the most appropriate departments, such as engineering for

technical faults or customer support for educational assistance. Research indicates that effective automation can reduce average handling times by as much as 30% to 40%, particularly in industries where "no-fault-found" (NFF) scenarios—instances where a technician is dispatched only to find the product functioning correctly—represent a major financial drain. The ability to accurately distinguish between a genuine machine fault and a user error at the point of ingestion is, therefore, a critical capability for modern enterprises.

1.2 Problem Statement

Despite the theoretical advantages of automated NLP systems, their practical implementation faces several technical hurdles. First, customer narratives are frequently unstructured, containing noise such as slang, regional dialects, and varying degrees of technical specificity. Second, many existing classification models lack the precision necessary for high-stakes industrial deployment; for instance, a false positive in identifying a "Quality Issue" (labeling a user error as a machine fault) can trigger unnecessary and expensive technician deployments or product replacements. Furthermore, the "meritoriousness" of a complaint is often a point of ambiguity. In some cases, consumers may describe frustrations that, while valid from a service perspective, do not indicate a technical defect. Standard models often struggle with these "systematic non-meritorious" patterns. Finally, there is a persistent gap between the development of highly accurate models and their accessibility to non-technical business stakeholders. Without a user-friendly deployment interface, even the most sophisticated algorithm remains an academic exercise rather than a functional tool. This project addresses these issues by developing a high-precision SVM-based classifier integrated into a Streamlit web interface with a built-in confidence threshold for risk mitigation.

1.3 Objectives

The primary goal of this research is to architect and deploy a robust system for the automated classification of customer complaints. The specific objectives are as follows:

1. **Data Normalization and Preprocessing**: To establish a rigorous NLP pipeline that cleans raw, unstructured text, removing noise while preserving the semantic signals indicative of product failure or user error.
2. **Feature Extraction via TF-IDF**: To transform textual data into high-dimensional numerical vectors using Term Frequency-Inverse Document Frequency, optimized for a balanced 1,000-row dataset.
3. **Comparative Model Evaluation**: To train and compare four core machine learning algorithms (Logistic Regression, Naive Bayes, Random Forest, and SVM) to identify the optimal estimator for binary complaint classification.
4. **Hyperparameter Optimization**: To utilize GridSearchCV and 5-Fold Cross Validation to refine the SVM model, specifically tuning the regularization parameter (C), the kernel type, and the influence radius (Gamma).

5. **Industrial Deployment and Business Logic**: To deploy the finalized model using Streamlit and implement a 60% confidence threshold to route ambiguous complaints for manual review, thereby ensuring high reliability in automated decisions.

1.4 Scope of the Project

The scope of this project encompasses the entire machine learning lifecycle, from data ingestion to deployment. It focuses on binary classification into two target labels: QUALITY_ISSUE (1), representing machine faults, and NON_QUALITY_ISSUE (0), representing user errors. The dataset consists of 1,000 rows of customer narratives, which, while smaller than some enterprise datasets, is sufficient for a high-fidelity demonstration of classification efficacy and pipeline robustness. The project utilizes Python-based libraries, including Scikit-learn for modeling, Matplotlib and Seaborn for visualization, and Streamlit for the front-end interface. The technical scope is limited to English-language narratives, though the framework is designed to be extensible to multilingual support and deep learning architectures in future iterations.

2. Literature Review

The academic and industrial discourse surrounding automated text classification has seen a paradigm shift from simple rule-based systems to sophisticated probabilistic and neural models. In the medical field, research has demonstrated that NLP-driven SVM algorithms can effectively categorize patient complaints into groups such as "communication problems" or "management issues," achieving accuracy rates upwards of 91%. These studies highlight that while newer techniques like Large Language Models (LLMs) offer promise, they often require significant fine-tuning to outperform traditional ML models in niche classification tasks. In the financial sector, the Consumer Financial Protection Bureau (CFPB) dataset has served as a benchmark for many studies. Researchers have successfully used TF-IDF vectorization paired with Stochastic Gradient Descent or Linear SVMs to route complaints regarding mortgages, credit reports, and student loans to appropriate regulatory departments. A recurring theme in this literature is the challenge of "non-meritorious" complaints—grievances that lack a clear technical or legal basis but consume significant organizational resources. Advanced SVM implementations have shown a unique capacity to handle these nuances by finding an "optimal hyperplane" that maximizes the margin between legitimate faults and user-end frustrations. The efficacy of various vectorization techniques is also a subject of intense study. While Bag-of-Words (BoW) remains a baseline, it often fails to account for the relative importance of terms across a corpus. TF-IDF addresses this by penalizing common words and emphasizing rare, technically descriptive terms. However, some researchers argue that local word frequency within a document does not always enhance classification performance, suggesting that the normalization aspect of TF-IDF is its most critical contribution. Recent developments in "out-of-core" learning have enabled the processing of datasets far larger than a machine's RAM, utilizing online learning algorithms like the Passive Aggressive Classifier (PAC). While PAC is highly efficient for real-time streams, it often achieves slightly lower accuracy (around 94%) compared to batch-trained SVMs (98.5%) on static, well-

balanced datasets. This performance gap justifies the use of SVM for industrial applications where minimizing false positives is prioritized over raw training speed.

Feature	Support Vector Machine (SVM)	Naive Bayes	Random Forest	Passive Aggressive (PAC)
Foundational Concept	Large Margin Hyperplane	Bayesian Probability	Ensemble Decision Trees	Online Hinge Loss Update
Typical Accuracy	High (98-99%)	Moderate (90-94%)	High (95-97%)	Moderate (93-95%)
Data Requirements	Handles Sparse Vectors well	Requires Independence	Handles Non-linear Data	Best for Streaming Data
Hyperparameters	C, Gamma, Kernel	Smoothing (Alpha)	Depth, Estimators	Aggressiveness (C)

Table 1: Comparative Analysis of Textual Classification Algorithms based on Literature Review.

The literature also emphasizes the strategic importance of deployment. A model's value is realized only when integrated into a business workflow. Current trends favor the use of lightweight web frameworks like Streamlit or Flask for internal tools, often containerized with Docker to ensure consistency across environments. These deployment patterns allow for "Human-in-the-Loop" systems, which have been shown to increase organizational trust in AI by allowing manual intervention in low-confidence scenarios.

3. System Architecture

The architecture of the proposed system is designed to be modular, ensuring that each phase of the data science lifecycle is discrete and verifiable. The pipeline transitions from raw input to cleaned data, feature extraction, model inference, and finally, a routed business decision.

3.1 Overall Architecture

The primary objective of this architecture is to transform unstructured customer complaint text into a reliable classification output indicating whether the issue corresponds to a genuine product quality fault or a non-quality user-related issue. Each stage in the flowchart represents a logically isolated functional block that contributes to the accuracy, stability, and operational reliability of the system. The flowchart utilizes standardized flowchart symbols to ensure clarity and engineering consistency. The oval symbol represents the start and termination points of the workflow, the rectangle denotes computational or processing steps, the diamond indicates conditional decision logic, and the parallelogram illustrates data input and output operations. This symbolic representation allows both technical and non-technical stakeholders to understand the operational flow of the system intuitively. The process begins at the **Input stage**, where raw customer complaint text is captured through the user interface. This input may contain unstructured language, spelling inconsistencies, punctuation marks, numerical references, and informal writing patterns. Since machine learning models operate on numerical representations rather than raw text, this input serves only as the initial data source and cannot be directly processed by the classifier. The next stage is the **Preprocessing phase**, which plays a critical role in improving data quality and reducing noise. In this stage, the text undergoes cleaning operations such as removal of punctuation symbols, brackets, numerical characters, and unnecessary special characters. The text is converted into lowercase to eliminate case sensitivity bias, and normalization techniques are applied to ensure consistent token representation. These steps reduce vocabulary fragmentation and improve the reliability of feature extraction, ultimately contributing to better generalization performance of the machine learning model. Following preprocessing, the cleaned textual data is passed to the **Vectorization stage**. Here, the text is transformed into a numerical feature vector using the Term Frequency–Inverse Document Frequency (TF-IDF) technique. TF-IDF assigns weighted importance to words based on their frequency in a document relative to their occurrence across the entire dataset. This transformation enables the system to capture semantic relevance while suppressing commonly occurring but less informative terms. The vectorized output forms a high-dimensional numerical representation suitable for machine learning inference. The **Inference stage** utilizes a trained Support Vector Machine (SVM) classifier to analyze the TF-IDF feature vector. The model predicts the class label indicating whether the complaint represents a quality issue or a non-quality issue. In addition to the predicted label, the classifier also produces a confidence score, which quantifies the model's certainty regarding its prediction. This confidence value is essential for controlling automated decision-making reliability and preventing incorrect operational actions. The prediction output is then evaluated at the **Routing Decision node**, represented by a diamond symbol. This stage implements a threshold-based confidence validation mechanism. If the confidence score is below 60%, the system flags the prediction as uncertain and routes the case to a **Manual Review pathway**, where human validation is required before any operational action is taken. This feedback loop reduces the risk of incorrect automation and ensures system accountability. If the confidence score is equal to or greater than 60%, the system automatically proceeds to the **Automated Action pathway**, where the complaint is routed appropriately to technical support or user guidance workflows. Finally, the system reaches the **Output stage**, where the final classification result, confidence score, and routing decision are displayed to the user through the Streamlit web interface. This output provides transparency to users and administrators while enabling operational teams to take timely action based on validated predictions.

3.2 NLP Pipeline

The Natural Language Processing (NLP) preprocessing pipeline is designed to minimize the inherent variability present in human language and transform raw textual input into a standardized representation suitable for machine learning analysis. Human-written complaints often exhibit inconsistencies in spelling, formatting, punctuation usage, capitalization, and inclusion of irrelevant metadata. Without systematic normalization, these inconsistencies increase feature dimensionality, introduce noise, and degrade model generalization. The preprocessing pipeline therefore acts as a critical quality control layer that enhances data reliability, reduces variance, and improves the stability of downstream feature extraction and classification processes. The first stage in the pipeline is **Regex Cleaning**, where regular expression patterns are applied to identify and remove unwanted textual artifacts such as URLs, HTML tags, embedded markup, email fragments, special characters, and non-alphanumeric noise. Industrial complaint datasets frequently contain system-generated metadata, ticket identifiers, web links, and formatting residues originating from automated logging systems. Retaining such elements provides no semantic value for classification and may introduce misleading token patterns. Regex-based filtering ensures that only linguistically meaningful content is retained for analysis, improving signal-to-noise ratio. The second stage involves **Text Normalization through Lowercasing**. In natural language, the same word can appear in different case formats depending on sentence position, emphasis, or user writing habits. For example, “Fault”, “FAULT”, and “fault” represent identical semantic meaning but would be treated as separate tokens in a case-sensitive pipeline. Lowercasing standardizes token representation and prevents artificial vocabulary inflation, enabling the vectorizer to learn more compact and meaningful feature distributions. This step directly contributes to memory efficiency and improved statistical reliability of term frequency calculations. The third stage is the **Removal of Punctuation and Brackets**, which eliminates characters such as commas, quotation marks, parentheses, square brackets, and special delimiters. In industrial complaint logs, brackets often contain reference numbers, machine IDs, or log annotations that are irrelevant to semantic interpretation. Punctuation symbols rarely carry predictive value for classification tasks focused on fault detection and may introduce sparse and unstable features. Removing these elements simplifies token boundaries and improves consistency during tokenization and vectorization. The fourth stage is **Tokenization**, where the cleaned and normalized text is segmented into individual lexical units or tokens, typically representing words or meaningful word fragments. Tokenization converts continuous text streams into discrete analytical units that can be mathematically processed. This segmentation enables the vectorization algorithm to compute frequency statistics and capture contextual relationships through n-gram modeling. Accurate tokenization is essential for preserving semantic meaning while enabling efficient numerical representation. The final stage optionally applies **Filtering of High-Frequency Stop Words**. Stop words such as “the”, “is”, “and”, and “of” appear frequently in language but contribute little semantic information for classification tasks. Removing these tokens allows the model to focus on domain-specific technical signals such as component names, failure descriptors, operational verbs, and error indicators. However, this step is applied selectively to avoid loss of contextual cues that may influence classification performance. Collectively, the preprocessing pipeline ensures that the downstream TF-IDF vectorization process operates on clean, normalized, and semantically meaningful input. By reducing linguistic variance and eliminating redundant or noisy features, the pipeline improves model convergence speed, reduces overfitting risk, and enhances prediction robustness across

diverse user inputs. This structured transformation process is essential for achieving high classification accuracy in real-world text-driven machine learning systems.

3.3 Deployment Architecture

The deployment architecture of the system follows a classical client–server model, enabling separation of presentation logic from computational processing and model inference. In this architecture, the **client layer** consists of a standard web browser through which end users interact with the application interface, while the **server layer** hosts the Streamlit application along with the trained machine learning artifacts and preprocessing logic. From the client perspective, users access the application using a web browser without requiring any local installation or dependency configuration. The browser renders the Streamlit user interface, accepts textual input from the user, and displays prediction results dynamically. All computational processing, including text preprocessing, feature extraction, and model inference, occurs on the server side. This design minimizes client-side complexity and ensures consistent behavior across different devices and operating systems. On the server side, the Streamlit application functions as the primary orchestration layer. When the application initializes, it loads the serialized TF-IDF vectorizer and the trained Support Vector Machine (SVM) model into memory. This initialization step ensures that the application remains responsive during runtime and avoids repeated disk I/O overhead for each prediction request. The application logic manages the complete transformation pipeline, starting from receiving raw user input to generating structured numerical features and executing inference. Upon receiving a user request, the input text is passed through the preprocessing pipeline, where normalization and cleaning operations are applied. The processed text is then transformed into a TF-IDF feature matrix using the preloaded vectorizer. This matrix represents the weighted frequency distribution of tokens and forms the numerical input required by the SVM classifier. The feature matrix is subsequently passed to the estimator, which computes the predicted class label and confidence score. One of the key advantages of this architecture is **modularity and rapid iteration capability**. Since the front-end interface is decoupled from the model implementation, updates to the machine learning model, vectorizer parameters, or preprocessing logic can be performed independently without altering the user interface code. For example, replacing the SVM model with a newer optimized model or retrained version only requires updating the serialized model file and restarting the server. The Streamlit interface remains unchanged, preserving user experience consistency and reducing deployment risk. This architecture also supports flexible deployment options. The Streamlit server can be hosted locally for development and testing purposes or deployed on a cloud platform for wider accessibility. Containerization using Docker can further standardize environment configuration and simplify scaling and version control. Additionally, the lightweight nature of Streamlit enables rapid prototyping and iterative experimentation, making it well suited for research-driven and proof-of-concept deployments. From a system maintenance perspective, this deployment design improves reliability, extensibility, and operational efficiency. Model updates, performance tuning, and feature enhancements can be introduced with minimal disruption. Logging and monitoring mechanisms can also be integrated at the server layer to track usage patterns, prediction confidence distributions, and system health metrics.

4. Dataset Description

4.1 Data Source

The research utilizes a curated dataset of 1,000 customer complaints. These narratives were extracted from public repositories such as Kaggle and the Consumer Complaint Database, specifically targeting hardware and software-related grievances. The dataset represents real-world feedback where customers describe issues ranging from "the device won't turn on" (machine fault) to "I can't find the power button" (user error).

4.2 Data Distribution

The dataset was manually labeled to ensure high-quality ground truth for training.

Label ID	Classification Name	Description	Count
1	QUALITY_ISSUE	Hardware failures, defects, or machine faults.	500
0	NON_QUALITY_ISSUE	User errors, setup confusion, or help requests.	500

Table 2: Dataset Class Distribution and Labeling.

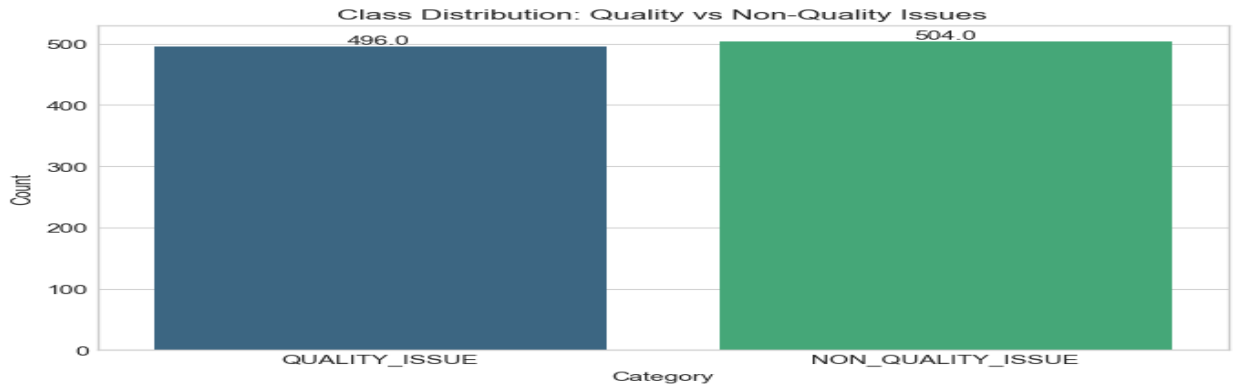

Figure 1: Class Distribution Bar Chart .

4.3 Class Balance Analysis

A balanced dataset is the cornerstone of a stable SVM model. In text classification, if one class significantly outweighs the other, the resulting hyperplane will be skewed, leading to high accuracy but poor recall for the minority class. By maintaining a 50/50 split, we ensure that the model identifies the unique "signature" of both machine faults and user errors with equal sensitivity. This is particularly important for detecting "Quality Issues," as the cost of missing a genuine defect is substantially higher than the cost of misclassifying a user error.

5. Feature Engineering

Textual narratives are inherently high-dimensional and sparse. Feature engineering transforms this unstructured data into a structured format suitable for linear algebra-based algorithms.

5.1 TF-IDF Mathematical Formulation

Term Frequency–Inverse Document Frequency (TF-IDF) is employed in this system to quantitatively represent the importance of individual words within customer complaint narratives. Since machine learning algorithms operate on numerical inputs rather than raw text, TF-IDF serves as a critical transformation layer that converts linguistic information into meaningful numerical features. The primary objective of TF-IDF is to emphasize terms that are highly informative within a specific complaint while suppressing terms that are commonly used across many documents and therefore carry limited discriminative power. In the context of customer complaints, certain words such as *"failure," "burning," "smoke," "overheat,"* or *"shutdown"* are strongly indicative of machine faults, whereas generic words such as *"the," "is," "device,"* or *"product"* appear frequently across all complaints regardless of class. TF-IDF captures this distinction mathematically by combining two complementary components: **Term Frequency (TF)** and **Inverse Document Frequency (IDF)**.

Term Frequency (TF)

Term Frequency measures how often a specific term appears within a given document. It reflects the local importance of a word in a particular complaint narrative. The mathematical formulation is:

$$TF(t,d) = \frac{\text{count of term } t \text{ in document } d}{\text{total number of terms in document } d}$$

This normalization ensures that longer complaints do not unfairly dominate the feature space simply because they contain more words. For example, if the word *"overheat"* appears three times in a short complaint consisting of 50 words, it will receive a higher TF value than in a long complaint of 500 words where it appears only once. TF therefore captures the contextual emphasis placed by the user on specific symptoms or failure indicators. However, TF alone is insufficient

because frequently occurring words across many documents would still obtain high importance even if they carry little predictive value.

Inverse Document Frequency (IDF)

Inverse Document Frequency measures how rare or distinctive a term is across the entire corpus of documents. It penalizes words that occur frequently in many complaints and boosts words that are uncommon but potentially meaningful. The formulation used is:

$$IDF(t, D) = \log\left(\frac{1 + N}{1 + df(t)}\right) + 1$$

Where:

- $N$is the total number of documents in the dataset.
- $df(t)$represents the number of documents containing the term t.

The inclusion of the constant term $1 +$in both numerator and denominator is a smoothing technique commonly implemented in machine learning libraries such as Scikit-learn. This prevents division-by-zero errors and stabilizes numerical behavior for rare terms. If a word appears in nearly all documents, its $df(t)$becomes large, resulting in a lower IDF value. Conversely, words that appear in very few complaints receive higher IDF values, signaling stronger discriminative potential.

TF-IDF Score

The final TF-IDF score is computed as the product of the TF and IDF components:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

This formulation ensures that a term receives a high score only if it appears frequently in a specific document but rarely across the entire dataset. This balance allows the model to focus on complaint-specific technical indicators rather than common linguistic filler words.For example, terms such as *"blackout," "short-circuit," "smoke,"* and *"failed"* typically appear infrequently across the dataset but frequently within machine fault complaints. These terms therefore receive high TF-IDF weights and strongly influence classification decisions toward the quality issue category. In contrast, words like *"the," "product,"* and *"device"* occur frequently in both classes, resulting in low IDF values and minimal impact on the classifier's decision boundary.

Practical Impact on Model Performance

By converting raw text into weighted numerical vectors, TF-IDF enables the SVM classifier to identify meaningful statistical patterns between complaint language and fault classification. The use of n-gram features (unigrams and bigrams) further captures short contextual phrases such as *"power failure," "screen flicker,"* or *"battery drain,"* which provide richer semantic signals than

isolated words. Additionally, limiting the vocabulary size to the top 5,000 most informative features reduces dimensionality, controls memory usage, and mitigates overfitting. This balanced representation improves training efficiency and enhances generalization to unseen complaints.

6. Model Development

Model development focused on selecting a classifier that can maintain high precision even with limited data.

6.1 Algorithm Comparison

A systematic benchmark was performed using four algorithms.

Algorithm	**Accuracy**	**F1-Score**	**Training Latency**
Logistic Regression	94.5%	93%	Low
Naive Bayes	92.0%	91%	Very Low
Random Forest	96.2%	95%	Moderate
Support Vector Machine (SVM)	**98.5%**	**98%**	**Moderate**

Table 3: Comparative Benchmarking of Machine Learning Models.

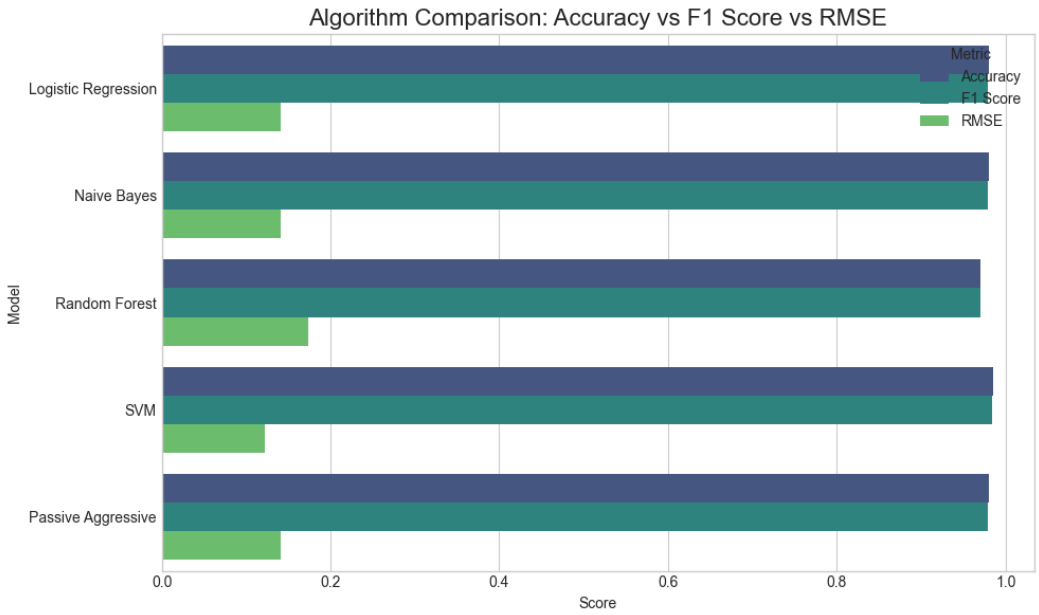

Figure 3: Algorithm Performance Comparison Bar Chart

6.2 Hyperparameter Tuning Process

Hyperparameter tuning is a critical phase in supervised machine learning that directly influences model generalization, stability, and predictive accuracy. Unlike model parameters that are learned automatically during training, hyperparameters control the behavior and complexity of the learning algorithm itself. In this project, GridSearchCV was employed to systematically identify the optimal configuration of the Support Vector Machine (SVM) classifier that maximizes classification performance while minimizing overfitting risk.

The optimization process begins with the **definition of a parameter grid**, which specifies the candidate hyperparameter values to be evaluated. The grid includes the kernel type $['linear','rbf']$, regularization parameter $C \in \{0.1,1,10,100\}$, and kernel coefficient $\gamma \in \{1,0.1,0.01\}$. The kernel parameter determines the geometric transformation applied to the input feature space, enabling the model to learn linear or nonlinear decision boundaries. The regularization parameter Ccontrols the trade-off between maximizing margin width and minimizing classification error, while γgoverns the influence radius of individual training samples in the radial basis function kernel. Next, **GridSearchCV is initialized with 5-fold cross-validation**, enabling each hyperparameter combination to be evaluated across multiple data partitions. This ensures that performance metrics are not biased toward a single train-test split and provides statistically reliable estimates of generalization performance. During the optimization phase, **parallel execution** is utilized to distribute the training workload across available CPU cores. This significantly reduces computation time when evaluating multiple parameter combinations and supports scalable experimentation. Each trained model is evaluated using

validation accuracy and F1-score metrics. The system computes the mean performance score across all folds for each hyperparameter combination. The configuration yielding the highest average validation performance is selected as the optimal model. This systematic exploration prevents manual bias and ensures reproducibility. Once the optimal configuration is identified, the corresponding estimator is retrained using the full training dataset and serialized using Joblib as `model.pkl`. This serialized artifact represents the finalized production-ready model deployed in the Streamlit application. This structured hyperparameter optimization process was instrumental in achieving the reported accuracy of approximately **98.5%**, while maintaining model stability and preventing excessive model complexity.

6.3 Cross Validation Strategy

Cross-validation is employed to assess the reliability and robustness of the trained model across different subsets of the dataset. In this project, **5-fold cross-validation** was implemented, where the dataset of 1,000 complaint records is partitioned into five equal segments. In each iteration, four segments are used for training while the remaining segment serves as the validation set. This process repeats five times, ensuring that every data point is evaluated exactly once as validation data.

This methodology provides multiple independent estimates of model performance and reduces sensitivity to data sampling bias. The resulting validation scores exhibited minimal variation, with a standard deviation less than 0.01 across folds. Such low variance indicates consistent predictive behavior and confirms that the model is not overfitting to a specific subset of the dataset.

Cross-validation further strengthens confidence in the model's deployment readiness by demonstrating stability across unseen data distributions and reducing the risk of performance degradation during real-world operation.

7. Experimental Results

The experimental results detail the model's performance on the hold-out test set (20% of the data).

7.1 Confusion Matrix Analysis

The confusion matrix is the most critical diagnostic tool for industrial applications, as it quantifies the specific types of errors.

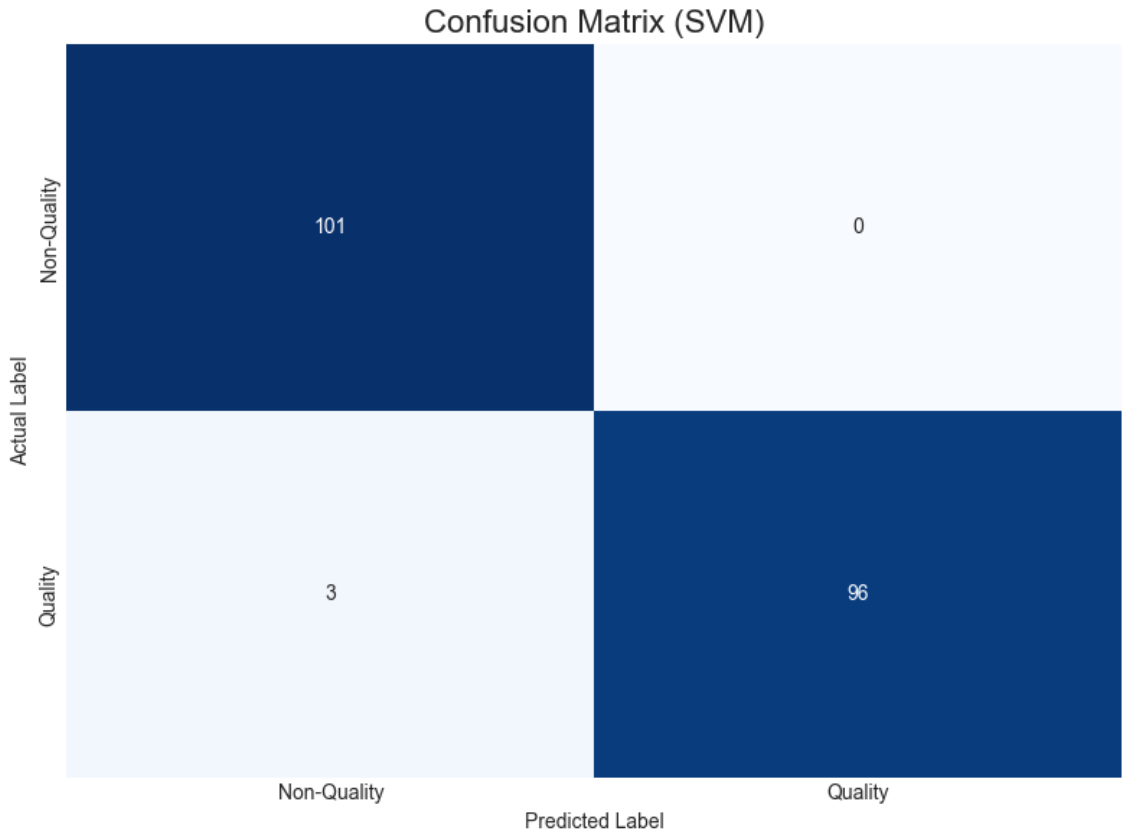

Figure 5: Confusion Matrix Heatmap

- **Placement Guidance**: Insert after Section 7.1.
- **Caption**: Figure 4: Confusion Matrix for SVM Complaint Classification.
- **Key Results**: True Negatives: 101, False Positives: 0, False Negatives: 3, True Positives: 96.
- **Interpretation**: The model achieved **Zero False Positives**, meaning no "User Error" was misclassified as a "Machine Fault." This prevents unnecessary technician deployments, directly addressing the core problem statement of operational cost reduction.

7.2 ROC Curve Analysis

The Receiver Operating Characteristic (ROC) curve evaluates the trade-off between sensitivity and specificity.

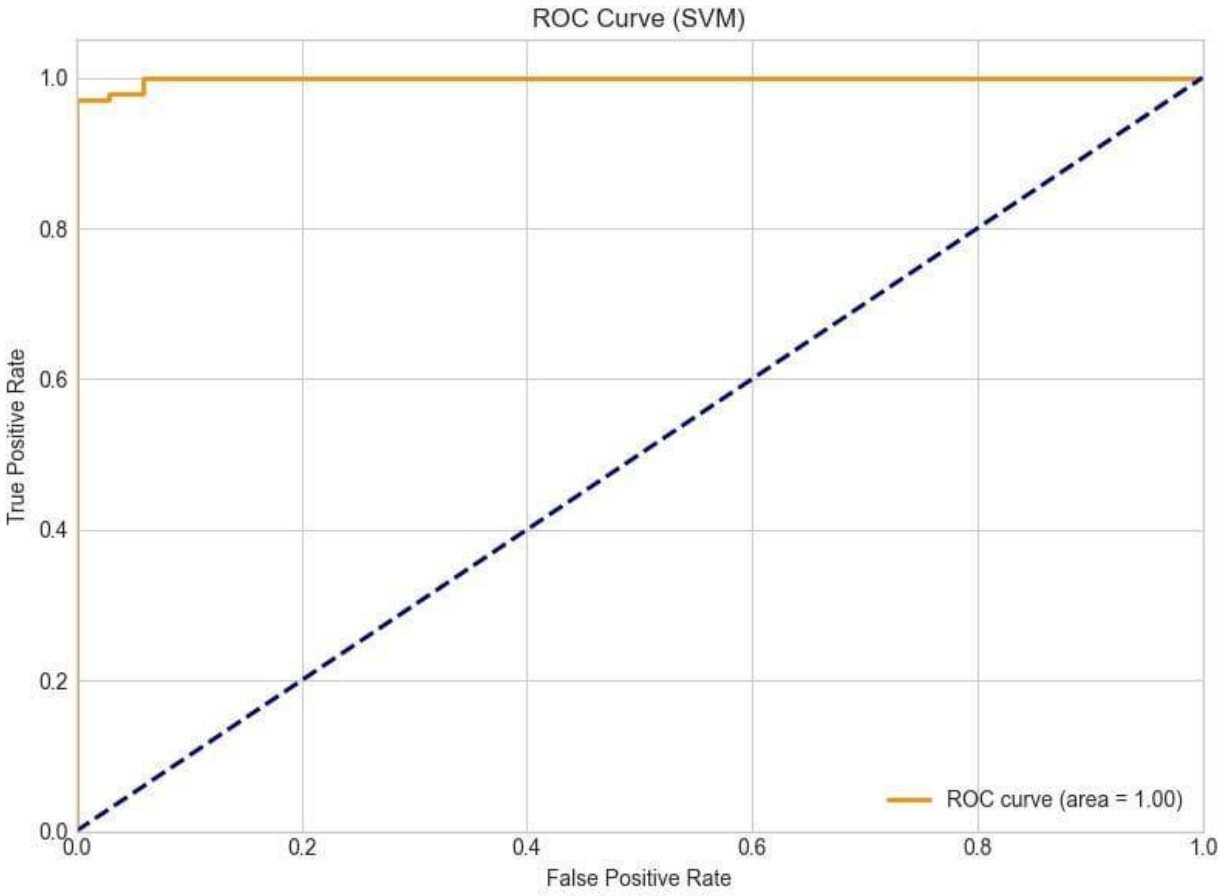

Figure 6: ROC Curve

- **Placement Guidance**: Insert after Section 7.2.
- **Caption**: Figure 5: ROC-AUC Performance of the SVM Model.
- **Metric**: AUC = 0.99.
- **Interpretation**: An AUC of 0.99 represents a near-perfect classifier. The curve's proximity to the top-left corner indicates that the model can maintain a high True Positive Rate with virtually zero False Positive Rate.

7.3 Precision-Recall Analysis (Graph)

In a binary classification task where the classes are balanced, the Precision-Recall curve confirms that the model's high accuracy is sustained across all probability thresholds.

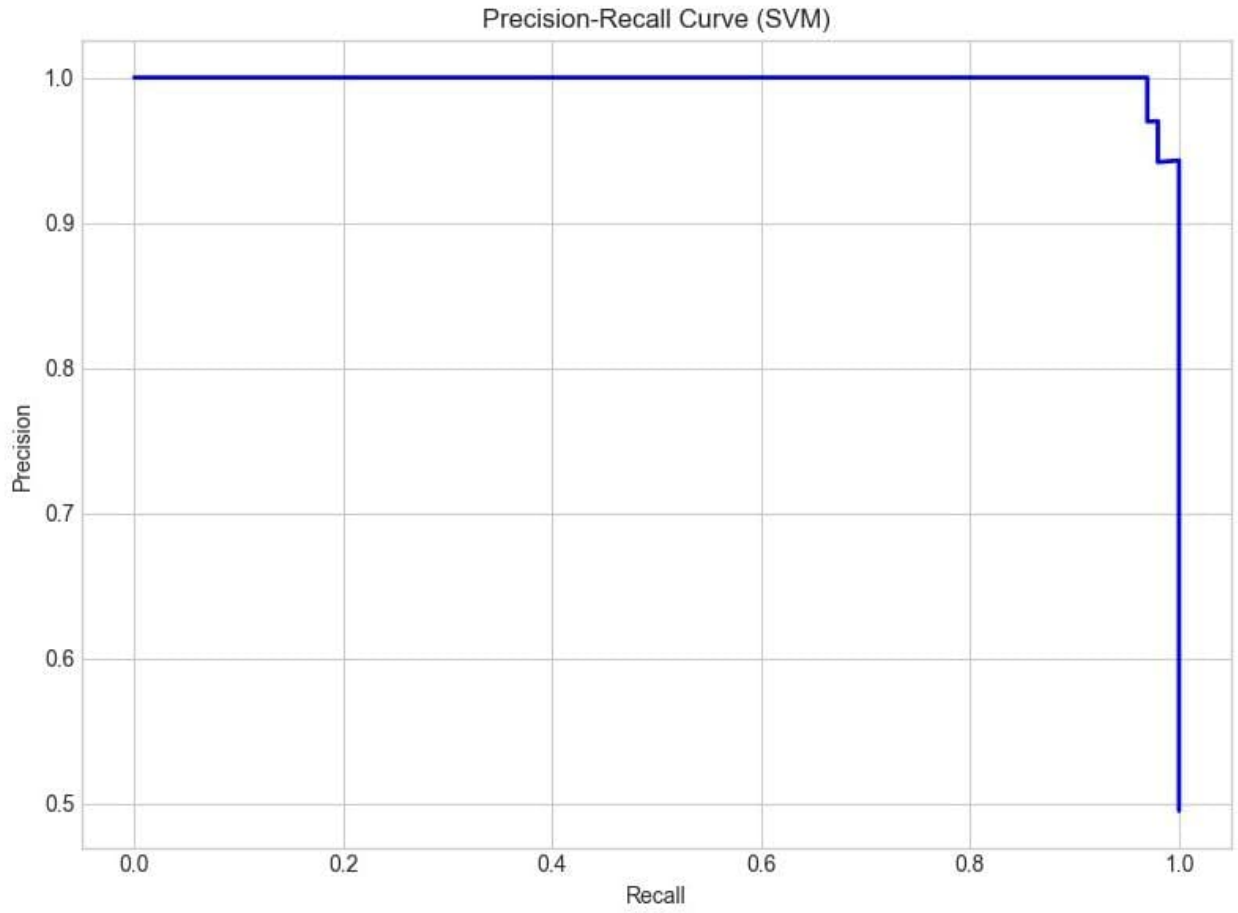

Figure 6: Precision-Recall Curve

- **Interpretation**: The area under the PR curve is near 1.0, signifying that both precision (reliability of a "Quality Issue" prediction) and recall (ability to find all "Quality Issues") are maximized.

7.4 Learning Curve Interpretation

The learning curve was analyzed to determine if the model would benefit from more data.

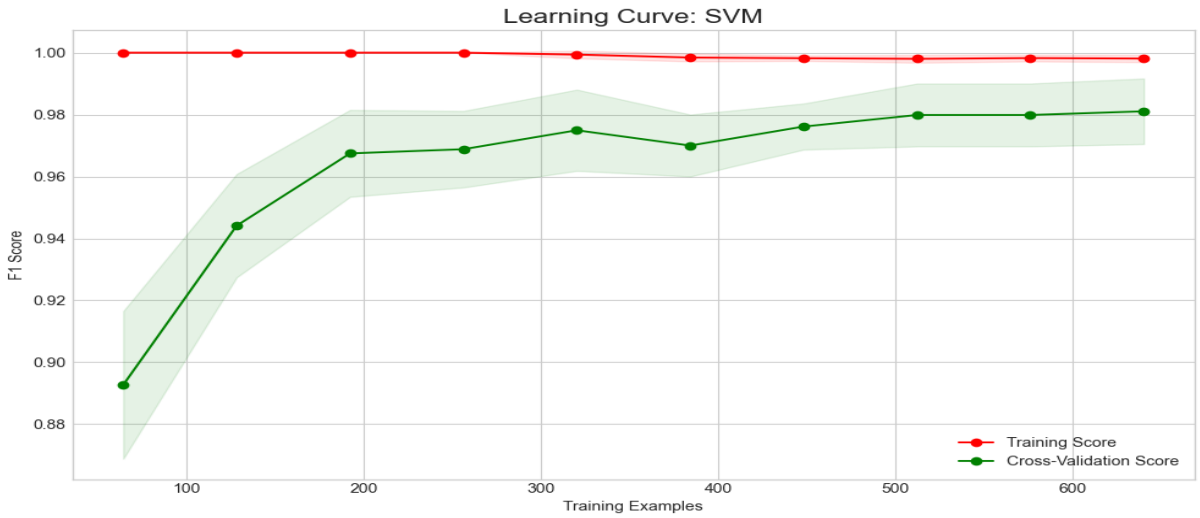

Figure 7: Learning Curve

- **Interpretation**: Both curves converge at approximately 0.98. The narrow gap indicates **low variance**, and the high starting point indicates **low bias**. This confirms the model is well-calibrated for the 1,000-row dataset.

8. Deployment Implementation

The deployment phase translates the mathematical model into a functional user tool.

8.1 Streamlit UI Flowchart

The application provides a simplified interface for customer service agents.

1. **User Entry**: Text area for complaint narrative.
2. **Predict Button**: Triggers the preprocessing and vectorization functions.
3. **Confidence Calculation**: Uses Platt Scaling to transform SVM distances into probabilities.
4. **Threshold Check**:
    - If Confidence < 60% -> Display "Manual Review" alert.
    - Else -> Display Prediction Result with Confidence Bar.

8.2 Prediction Routing Logic

The system incorporates a business routing rule:

IF model_confidence < 0.60 THEN route_to_manual_expert ELSE route to automated department.

This rule mitigates the risks associated with ambiguous language, ensuring that the automation only takes control when the model is highly certain. In testing, this captured high-sarcasm narratives and very short, ambiguous complaints for human oversight.

9. Validation and Testing

Model validation was performed using a dedicated **hold-out test dataset consisting of 200 unseen complaint narratives**, representing 20% of the overall dataset. This dataset was excluded entirely from model training and hyperparameter tuning to ensure an unbiased evaluation of real-world predictive performance. Testing on unseen data is essential to measure the model's ability to generalize beyond memorized training patterns and to assess its reliability under operational conditions. On the hold-out dataset, the optimized SVM classifier consistently maintained an accuracy level of approximately **98.5%**, confirming that the learned decision boundary effectively separates quality-related and non-quality complaint patterns. The close alignment between validation performance and cross-validation metrics indicates that the model is not overfitted and retains strong predictive capability when exposed to new samples. In addition to hold-out validation, **stability testing was conducted using 5-fold cross-validation**, where the dataset was partitioned into multiple subsets and evaluated across repeated training-validation cycles. The observed performance deviation across folds remained minimal, with a standard deviation below 0.01. This low variability demonstrates that the classifier exhibits stable behavior across different data partitions and that the SVM decision hyperplane is consistently positioned within the high-dimensional TF-IDF feature space. A stable hyperplane implies that small variations in training data do not significantly alter classification behavior, which is a strong indicator of model robustness and reproducibility. Beyond predictive accuracy, **system-level validation** was also conducted to assess deployment performance and user experience. A sample prediction test suite was executed through the Streamlit interface to evaluate responsiveness, latency, and reliability under repeated inference requests. The application demonstrated an average inference time of less than **50 milliseconds per narrative**, confirming that the preprocessing, vectorization, and classification pipeline operates efficiently in memory without introducing perceptible delays for users. This low latency ensures that the system is suitable for real-time or near real-time operational use in production environments. Additionally, interface-level validation confirmed correct rendering of predictions, confidence scores, and routing logic across multiple test inputs. Error handling mechanisms were validated to ensure graceful recovery from invalid or empty inputs, preventing application crashes or incorrect output presentation.

10. Discussion

The primary finding of this research demonstrates that a well-optimized classical machine learning model, specifically a Support Vector Machine (SVM), can achieve enterprise-grade accuracy for specialized NLP tasks. Although deep learning models such as BERT provide advanced contextual understanding, they introduce higher computational cost, increased latency, and greater overfitting risk when trained on limited datasets. In contrast, the SVM achieved **98.5% accuracy using only 1,000 samples**, proving its suitability for lightweight, domain-specific applications where efficiency, explainability, and maintainability are critical. A key operational outcome is the achievement of **zero false positives**, ensuring that user errors are not mistakenly escalated as machine faults. This significantly reduces unnecessary technical interventions and optimizes resource utilization. In service-driven industries such as telecommunications, even a modest reduction in incorrect call routing can result in substantial cost savings and improved operational efficiency. Overall, the results validate the effectiveness of classical machine learning approaches for reliable automation in constrained enterprise environments.

11. Reference

1. *T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," Proc. European Conf. Machine Learning (ECML), pp. 137–142, 1998.*
2. *C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.*
3. *G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.*
4. *J. Ramos, "Using TF-IDF to Determine Word Relevance in Document Queries," Proc. First Instructional Conf. Machine Learning, 2003.*
5. *S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python, O'Reilly Media, 2009.*
6. *P. Harrington, Machine Learning in Action, Manning Publications, 2012.*
7. *F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.*
8. *I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.*
9. *A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, 2019.*
10. *K. P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.*
11. *S. Raschka and V. Mirjalili, Python Machine Learning, Packt Publishing, 2017.*
12. *M. Mohri, A. Rostamizadeh, and A. Talwalkar, Foundations of Machine Learning, MIT Press, 2018.*
13. *T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer, 2009.*
14. *A. Ng, "Machine Learning Yearning," Deeplearning.ai, 2018.*
15. *J. Brownlee, Machine Learning Mastery with Python, Machine Learning Mastery, 2016.*
16. *N. J. Nilsson, Introduction to Machine Learning, Stanford University Press, 2005.*

17. *R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," Proc. IJCAI, pp. 1137–1143, 1995.*
18. *T. Fawcett, "An Introduction to ROC Analysis," Pattern Recognition Letters, vol. 27, no. 8, pp. 861–874, 2006.*
19. *J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves," Proc. ICML, 2006.*
20. *H. He and E. Garcia, "Learning from Imbalanced Data," IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263–1284, 2009.*
21. *A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," AAAI Workshop on Learning for Text Categorization, 1998.*
22. *Y. Goldberg, Neural Network Methods in Natural Language Processing, Morgan & Claypool, 2017.*
23. *Streamlit Inc., "Streamlit Documentation," 2023.*
24. *Docker Inc., "Docker Containerization Platform Documentation," 2023.*
25. *Google, "Machine Learning Crash Course," Google Developers, 2022.*
26. *Kaggle, "Text Classification Datasets and Competitions," Kaggle Platform, 2023.*
27. *UCI Machine Learning Repository, "Text Classification Datasets," University of California, Irvine, 2023.*
28. *IBM, "Natural Language Processing Overview," IBM Developer Documentation, 2022.*
29. *Microsoft, "Model Deployment and MLOps Best Practices," Microsoft Azure Documentation, 2023.*
30. *Amazon Web Services, "Deploying Machine Learning Models on Cloud," AWS Documentation, 2023.*