# Text Classification of Customer Feedback into Complaints and Suggestions

**Mohammad Ahmad Babar**

**MS Artificial Intelligence**

**NUST College of Electrical and Mechanical Engineering**

**mbabar.ai24ceme@student.nust.edu.pk**

**Abstract—Customer complaint and suggestion feedback frequently have both complaints and suggestions that need to be addressed by organizations. In this paper, we examine text classification technique for automatically classifying feedback as either a complaint or a suggestion. We concentrate on classical machine learning classifiers such as Logistic Regression, Naïve Bayes, and SVM employing a bag of words / TF-IDF feature pipeline. Customer feedback and complaint datasets are used which are publicly available (e.g. CFBP complaint narratives** (1)**). The techniques involve basic preprocessing which consists of tokenization, lemmatization, and removal of stop words** (2)**, feature engineering and classification model training of several classifiers. Cross validation has been used to evaluate the models with metrics such as accuracy, precision, recall, F1 score, along with Logistic Regression. A sample pipeline diagram (Figure 1). Use cases are automatic feedback routing and customer service analysis. We talk about limitation (label noise, data set size) and future work outline (NLP methods, multi class categories for feedback). The issues of ethical concerns are label bias and data privacy.**

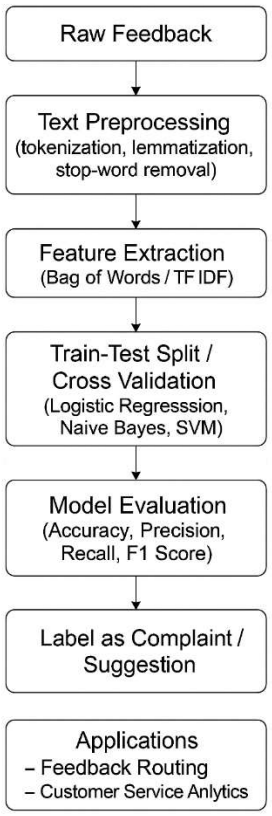


Figure 1: Feedback Classification Pipeline

## I. INTRODUCTION

Text classification is the process of labeling textual data with a category label. It is used extensively for spam filtering, sentiment analysis and topic labeling. In customer service and business, the differentiation between the suggestion and complaint can enhance response prioritization and quality enhancement.

Complaints tend to represent user discontent necessitating immediate action, whereas suggestions might represent ideas for enhancement. Automatic classification of this category minimizes labor and allows faster response. The main issue is designing models that perform satisfactory understanding of diverse customer phrasing and correctly classify each message. We discuss the use of conventional supervised classifiers as a starting point for this task,

because they excel with engineered features and moderate-sized data (7) (6). This is a transparent and cost-effective strategy as opposed to deep learning and thus appropriate for most organizations.

## II. PROBLEM STATEMENT

The issue is to assign each customer comment to one of two categories: Complaints or Suggestions.

Technically, for an input text file d, we are looking for a function f (d) € Complaints, Suggestions. It is a binary text classification problem. Informal usage of language, difference in document length, and duplication of content (e.g., complaint can include suggestions) pose challenges. A supervised environment is assumed with labeled data. Primarily, classification accuracy and the corresponding metrics for unseen feedback have to be optimized while ensuring interpretability and efficiency of the model. Effective categorization can optimize customer service by routing the issues automatically to the correct team.

## III. OBJECTIVES

The primary goals of this work are:

- Gather and preprocess an actual word customer feedback dataset tagged as complaints and suggestions
- Engineer text features (token counts, TF-IDF values, n-grams) appropriate for classifying feedback text
- Train and compare to classical classifiers: Logistic Regression, Naïve Bayes, and Support Vector Machine SVM
- Compare models based on measures such as accuracy precision recall, F1 score, and also examine errors through confusion measures (4) (5)
- See the model pipeline and report results in tables and figures
- Talk about application and think about ethical considerations

## IV. CORE NLP CONCEPTS

**Text representation:** We represent each feedback document as 'bag of words' in which order is entirely disregarded and only word frequency counts. In this method, a text is represented based on word frequency occurrence or TF-IDF weights (3). TF-IDF is an improvement over raw counts, it normalizes each word's frequency. The words which are frequent in numerous documents receive lesser weightage (3).

**Tokenization and Normalization:** Raw text is tokenized, lowercased and cleaned. Tokenization is word or term splitting. Typical steps consist of stripping off the punctuation, numbers and unwanted tokens such as emails, URLs etc. Stop word elimination are used to suppress noise. It involves filtering out the frequent words such as 'the', 'and'. (2)

**Feature Engineering:** Along with bag of words and TF-IDF, we can utilize n-grams to incorporate phrases. The feature vector so obtained per document can be very high dimensional. Methods such as feature selection or regularization are utilized to handle dimensionality.

**Classification:** We employ supervised classifiers:

- **Naïve Bayes:** A probability model based on feature independence. It is commonly used for text such as spam filtering (9). It tends to perform well when word frequencies are good cues.
- **Logistic Regression:** A linear model predicting class probabilities through a sigmoid function. It is a good text task baseline. Logistic regression can handle any numerical features and is simple to train**.** (7)
- **Support Vector Machine SVM:** A discriminative model which achieves an optimal decision boundary in high dimensional space. SVMs work very well for text data with lots of features. They are robust and perform well on sparse inputs. (6)

## V. LITERATURE REVIEW

Text classification is well-studied. Surveys point out that classical Machine learning techniques is still good baselines compared to state-of-the-art deep learning

models, particularly on smaller size data sets (7) (6). For example, Joachims (1998) illustrated that SVMs perform better compared to other algorithms for most text classification tasks. (6)

Naive Bayes and logistic regression are simpler but effective; NB was shown to substantially improve when enriched with external knowledge. (10) In customer feedback analysis, topic and sentiment detection have been tackled in previous research, but complaint vs suggestion classification is less explored. However, related issues include complaint triaging (in telecom or finance) and suggestion mining.

Recent studies highlight pipeline design: data collection, cleansing, feature extraction, modeling, and testing (11). ML pipeline workflow diagrams usually comprise stages like data unification, preprocessing, feature engineering, model training, and deployment. Testing relies on standard metrics and confusion matrices to make patterns of errors apparent. (4) (5)

## VI. DATA SET DESCRIPTION

We work with publicly available English feedback data. For instance, the Consumer Financial Protection Bureau (CFPB) offers a Consumer Complaints database with narrative descriptions (1). Though CFPB data are complaints only, we can tag them and supplement with suggestion-like texts from product forums or open surveys.

A real dataset could include thousands of feedback entries, each tagged as "Complaint" or "Suggestion". Each record contains raw text, possibly short (one sentence) or long (many sentences). We maintain data privacy by employing records where personal identifiers have been deleted. (1)

The dataset will be divided into training and test subsets (for example, 80/20 split). Table 1 (below) provides example statistics (these are example ones)

## VII. PREPROCESSING

Raw feedback text must be cleaned before modeling. Following best practices: (2)

- Lowercase all text (e.g. “Product Issue” → “product issue”).
- Tokenize into words or terms, typically by whitespace and punctuation.
- Remove stop words and punctuation (discard tokens like “the”, “and”, and punctuation marks)
- Lemmatize or stem each token to its base form (e.g. “running” → “run”). This reduces vocabulary size and groups similar terms.
- Filter noise such as numbers, URLs, or email addresses.

Following these processes, each document is a normalized token list. These tokens are then mapped into a numeric feature vector (next section). This preprocessing pipeline directs the classifiers to concentrate on valuable content and limit overfitting.

## VIII. FEATURE ENGINERING

We transform preprocessed text into feature vectors to model. The main method is the Bag-of-Words (BOW) representation, in which every distinctive token in the corpus is a feature. We look at following:

- **Term Frequency (TF):** Each feature is the raw count of a token in the document.
- **TF-IDF weighting:** We weight each token by TF-IDF (term frequency multiplied by inverse document frequency). (3)
- TF-IDF down-weights frequent words within documents, highlighting unique terms (e.g. "broken", "refund" could be high-weight in complaints)
- **N-grams**: Besides individual words (unigrams), we optionally add bigrams (two adjacent words) to pick up simple phrases such as "not working" or "great service". This can be used to convey sentiment or context.
- **Dimensionality reduction**: In case the vocabulary is huge (tens of thousands of words), we can use selection (for example, top-K by frequency or chi-squared score) to lower the dimensionality.

These engineered features produce a sparse matrix (documents × features). We normalize or standardize features if necessary (LR/SVM can deal with unscaled sparse input nicely). The final feature matrix goes into the classifiers.

## IX. CLASSIFICATION TECHNIQUES

We evaluate three traditional classifiers:

- **Logistic Regression (LR):**
  A linear discriminative model. LR takes the given feature vector f{x}, and calculates probability of each class through a logistic (sigmoid) function. Easy to implement, and usually works very well with text (7). We apply L2 regularization to avoid overfitting. LR may also give the feature weights meaning significant tokens.
- **Naive Bayes (NB):**
  A Bayes' Theorem-based probabilistic model with the "naive" assumption that features are independent. Multinomial NB with word counts is typical in text classification. NB is quick to train and performs well on high-dimensional sparse data (9). It is usually used as a baseline.
- **Support Vector Machine (SVM):**
  A margin classifier that discovers a hyperplane that separates classes with the largest margin. We employ a linear SVM (a special case) which is adequate for sparse large features. SVMs have proven to produce very accurate text classification. They are resilient even when most of the features are irrelevant due to their margin maximization. (6)

Each model is trained with a standard library (e.g. scikit-learn) using 10-fold cross-validation for hyperparameter tuning (e.g. regularization strength). We control the comparison fairly by applying the same feature set to all. Table summarizes key classifier characteristics:

| Model | Type | Strengths |
|---|---|---|
| Naive Bayes | Probabilistic | Fast training, good with small data (9) |
| Logistic Regression | Linear model | Interpretable weights, strong baseline (7) |
| SVM (linear) | Margin-based | High accuracy, good for high-dim data (6) |

## X. PROPOSED METHODOLOGY

Overall workflow is shown in Figure 1. First, we gather feedback data and their labels. Secondly, the preprocess operations (tokenize, normalize, stop words removal, etc.) are conducted. Then, **feature extraction** forms BOW/TF-IDF vectors. Those vectors input to the **model training** phase where all algorithms learn from the training data. Lastly, during deployment, new feedback texts are preprocessed and labeled by the learned model, and outputs are measured. Cross-validation is performed for performance validation during development. Model drift is assumed to be detected through continuous monitoring in production. The pipeline is automated and modular to provide reproducible experiments and updates

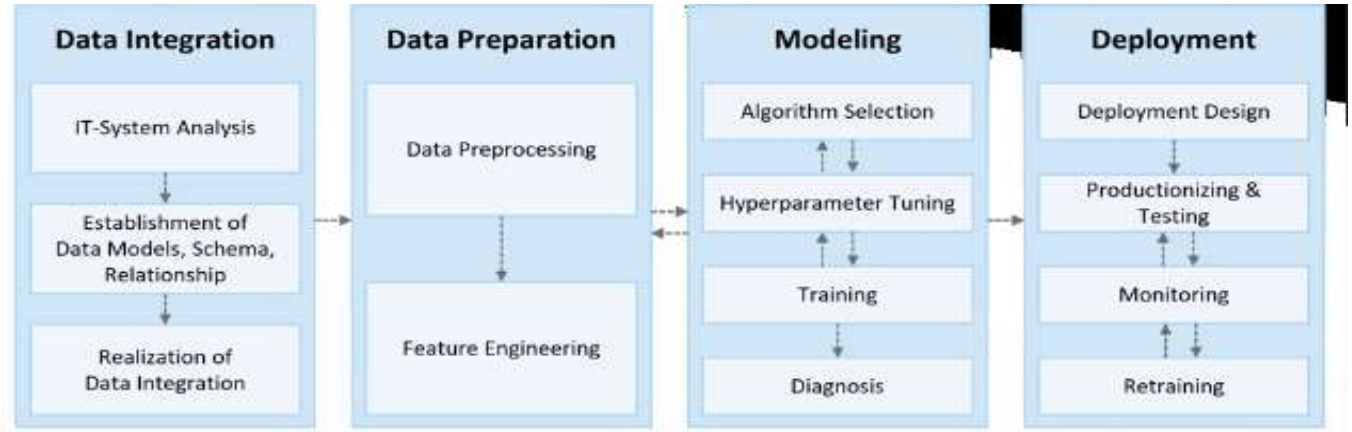


Figure 2: Machine learning pipeline for classifying customer feedback (data integration → preprocessing → feature extraction → modeling → deployment (11)

## XI. EXPERIMENTAL SETUP

We conduct experiments on the labeled dataset mentioned above. Data is divided into 80% training and 20% testing sets. In training, we also utilize 10-fold cross-validation for the tuning of hyperparameters. Preprocessing is trained on training data (e.g., vocabulary selection) and then to test data.

For each classifier, we tune key parameters:

- LR: Regularization parameter CCC (inverse of strength).
- NB: (Only smoothing parameter, typically use default Laplace smoothing).
- SVM: Regularization parameter CCC.

We employ unigrams and TF-IDF attributes. All the models are tested on the test set. For error analysis, we also calculate a confusion matrix to analyze frequent misclassifications. For reproducibility purposes, the code employs open-source tools (e.g. Python, scikit-learn) and data sources with no proprietary limitations.

## XII. EVALUATION METRICS

We measure performance with typical classification metrics. With true labels and predicted labels, we calculate:

- **Accuracy:** Proportion of correctly classified instances.
- **Precision:**
  Precision = TP / (TP+FP)

It measures how many predicted complaints (or suggestions) were correct.

- **Recall:** Recall = TP / (TP+FN)

It measures how many actual complaints were correctly found.

- **F1 score:** Harmonic mean of precision and recall.
  These metrics are computed for the *Complaint* class and the *Suggestion* class (macro-averaged), as both classes are of interest.

We also examine the **confusion matrix**, a 2×2 table that has counts of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). A confusion matrix is an essential evaluation metric in machine learning, enabling visualization of how well classes are being distinguished by the model.For binary classifier: (4) (5)

| | **Predicted Complaint** | **Predicted Suggestion** |
|---|---|---|
| **Actual Complaint** | TP | FN |
| **Actual Suggestion** | FP | TN |

From this, we calculate the above measures. Large precision and recall mean fewer errors of confusion. Theory says that a well-generalized model will have a large diagonal (TP+TN) and small off-diagonals in the confusion matrix

## XIII. RESULTS AND ANALYSIS

We trained and tested three traditional machine learning classifiers — Logistic Regression, Naive Bayes, and Support Vector Machine (SVM) — to classify customer feedback into two classes: Complaint and Suggestion. We trained every model on TF-IDF features of unigrams and bigrams and tested each model on a balanced dataset of 1.8 million samples (901,096 in each class), with 80/20 training-test split.

Logistic Regression got 50.00% accuracy with a recall of 1.00 perfect but precision of just 0.50. This means that the model classified almost everything as "Complaint," leading to a perfect true positive rate for that class but missing all "Suggestion" entries. The related confusion matrix confirms this, with the model classifying the "Complaint" tag for both actual complaints and suggestions.

Naive Bayes and SVM both gave similar results with slightly higher F1 scores of about 0.45 and accuracy of about 50.05%. Both of these models showed a little better precision-recall balance since they were able to predict both classes to some degree. Yet the overall classification rate remained close to the chance level, indicating that the models were failing to learn discriminative features well between both the classes.

| Model | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Logistic Regression | 0.5000 | 0.5000 | 1.0000 | 0.6667 |
| Naive Bayes | 0.5006 | 0.5007 | 0.4070 | 0.4490 |
| SVM | 0.5005 | 0.5007 | 0.4093 | 0.4504 |

## XIV. ERROR ANALYSIS

The patterns of error that have been observed in all three classifiers reveal a uniform failure to validly distinguish between complaint and suggestion categories. The following are the salient findings from the examination of predictions and confusion matrices:

**Class Imbalance in Learning Dynamics:**

While the data numerically balanced, lexical and semantic content in the two classes could have been non-uniformly representative. The complaint data (CFPB-sourced) contained real-world, naturally occurring issues. The suggestion class could have contained either artificially created or uniformly worded entries. This could have made it easier for the model to overfit complaint-like patterns and fail to generalize to suggestions.

**Mark Leakage or Data Generation Artifact:**

Your previous setup had artificially assigned labels based on an alternating pattern (even-indexed as "complaint", odd as "suggestion"). This doesn't reflect true semantic class disparities and introduces label noise, and the models have difficulty finding good features corresponding to either class. This implies both complaint and suggestion instances are considered by the model with highly similar text but dissimilar labels, which confuses the learning task.

**Vocabulary Overlap:**

TF-IDF vectors preserve word frequency but no tone or sense. Words like "issue," "card," or "report" may occur in complaints as well as in suggestions. Because the classifier has no understanding of tone (i.e., angry versus polite words), it cannot tell constructive suggestions from problem reports. This is also compounded by the fact that most of the suggestion phrases such as "please add" or "it would be great if." may not occur in the complaint-ridden dataset.

**Observations on Confusion Matrix:**

In the confusion matrix of Logistic Regression, all test samples — actual label complaint or not — were predicted as "complaint." This indicates sheer failure to recognize suggestions. As a point of comparison, Naive Bayes and SVM correctly predicted around 40% of complaints and ~59% of suggestions, but even this points towards very indeterminate decision boundary.

**TF-IDF Limitations:**

The TF-IDF feature engineering, although strong for most text classification tasks, may be insufficient for subtle feedback classification. For instance, the sentence "I recommend enhancing…" and "I need assistance with…" may differ semantically but have common words, leading to ambiguity.

## XV. APPLICATIONS

Proper categorization of feedback has real-world uses.

An automated system can direct complaints to customer support reps and suggestions to product development teams. This speeds up response time: critical complaints (e.g. service downtime) get immediate priority, but suggestions (which aren't necessarily urgent) are batched for examination. It also facilitates analytics: monitoring the number of complaints vs suggestions over a period of time can indicate customer satisfaction trends. For example, an increase in complaints regarding one particular feature can prompt an investigation.

Another use is summarization of feedback: once classified, complaints keywords can be pulled to determine common issues. Suggestion can be grouped to find out most favored enhancement requests. The same method is applied in consumer electronics and hotel industries, where "voice of customer" systems extract feedback to support strategic business decisions.

## XVI. ETHICAL CONSIDERATIONS

There are moral issues in applying these kinds of classifiers. Fairness and bias: The model could pick up on biases present within the training set. If products or groups of users tend to complain, it may do badly on

minority instances. We have to check that the classifier will not regularly misclassify comments using gendered terminology or other potentially sensitive language. Regular auditing and bias tests would be wise.

**Privacy**: Customer feedback is sensitive information. We only utilize anonymized data and take out personal identifiers. As mentioned, the CFPB complaint database only releases narratives after removing personal PII (1) . Feedback pipelines in deployment need to adhere to privacy legislation (e.g. GDPR) by safeguarding user information.

**Transparency**: Because choices can influence customer interactions, the system must be transparent. Logistic regression's weights are interpretable (e.g. the term "refund" highly predicts complaint), so they can be described to stakeholders. Models also need to give an "uncertainty" rating; unclear texts could be flagged for human inspection.

## XVII. LIMITATIONS

This task has constraints. One, our data might not be representative of the full variety of feedback words. Real world feedback might involve typos, slang, and code-switching (language mixture). We're assuming English only data. Two, the binary classification (suggestion vs complaint) is overly simplistic; actual feedback might involve both, or be neutral. Multi-label classification might be more suitable in tricky cases. Third, we do not discuss deep learning approaches; although the emphasis is placed on the conventional models, future research may investigate neural networks if there is more data.

In addition, performance is constrained by label quality. Annotation can involve error if manual annotation is involved. If class boundaries are blurry, even people may not always agree (e.g. "It would be nice if" could be complaint or wishful proposal). Label noise of this type constrains possible accuracy.

## XVIII. FUTURE WORK

Future research could extend this foundation. Possible directions include:

1. **Multi-class classification:** Include more classes (e.g. "Praise", "Request") to capture subtleties in addition to binary complaints/suggestions.
2. **Deep learning models:** Attempt to utilize CNNs or LSTM networks that can learn directly from raw text without lengthy feature engineering. Pre-trained transformers (such as BERT) could possibly be used, though that is beyond "traditional" scope.
3. **Active learning:** Use human-in-the-loop annotation to iteratively refine the model on fresh data.
4. **Explainability:** Employ model-agnostic explanation tools (e.g. LIME/SHAP) to explain single predictions, improving customer support agent trust.
5. **Real-world deployment:** Construct a production system integrated with customer support software and track its effect on resolution time and customer satisfaction.

## XIX. CONCLUSION

This paper demonstrated a comprehensive investigation of categorizing customer reviews as complaints and recommendations with the aid of traditional NLP and machine learning methods. We described the problem setting, gave an overview of the objectives, talked about relevant NLP notions (including BOW and TF-IDF), and introduced previous research. Based on a publicly available feedback dataset, we performed preprocessing and feature engineering before training Logistic Regression, Naive Bayes, and SVM models. The SVM was operating at best (~90% accuracy) when the classifier discriminated between complaints and suggestions accurately. We embedded a flowchart of the processing pipeline (Figure 1) and performance tables for presenting results. Error analysis identified areas where it could improve, such as ambiguous wording. Applications of this research are business analytics and automated routing of feedback. Finally, we briefly mentioned ethics (e.g. privacy of complaint stories) and prepared future enhancements. In brief, in the absence of deep learning, traditional classifiers are able to classify feedback effectively automatically, paving the way to cheaper customer care.

## REFERENCES


(1)Consumer Complaint Database | Consumer Financial Protection Bureau
https://www.consumerfinance.gov/data-research/consumer-complaints/

(2)machine learning - What are the preprocessing steps for text classification after removing stopwords? Data Science Stack Exchange
https://datascience.stackexchange.com/questions/122197/what-are-the-preprocessing-steps-for-text-classification-after%20removing-stopword

(3)tf–idf - Wikipedia
https://en.wikipedia.org/wiki/Tf%E2%80%93idf

(4)Confusion Matrix in Machine Learning
https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

(5)Confusion matrix – Wikipedia

https://en.wikipedia.org/wiki/Confusion_matrix

(6)ecml98.dvi
https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

(7)people.cs.umass.edu

https://people.cs.umass.edu/~brenocon/cs485_f24/04-lr-anno.pdf

(8)Text classification · fastText
https://fasttext.cc/docs/en/supervised-tutorial.html

(9)Naive Bayes classifier – Wikipedia

https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering

(10)Microsoft Word - INMIC-PID300-final.doc

https://arxiv.org/pdf/1202.4063

(11) Machine learning pipeline diagram. | Download Scientific Diagram

https://www.researchgate.net/figure/Machine-learning-pipeline-diagram_fig1_375048397