Project Report

On

SPAM CLASSIFICATION USING NAÏVE BAYES AND CENTROID COSINE: A COMPARATIVE PERSPECTIVE

Submitted

in partial fulfilment of the requirements for the degree of

BACHELOR OF SCIENCE

in

MATHEMATICS

by

DEVIKA VIJESH

(Register No. AB23BMAT015)

Under the Supervision of

MS. SREELAKSHMI K ANIL



DEPARTMENT OF MATHEMATICS

ST. TERESA'S COLLEGE (AUTONOMOUS)

ERNAKULAM, KOCHI - 682011

APRIL 2026

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

CERTIFICATE

This is to certify that the dissertation entitled, **SPAM CLASSIFICATION USING NAÏVE BAYES AND CENTROID COSINE: A COMPARATIVE PERSPECTIVE** is a bonafide record of the work done by Ms. **DEVIKA VIJESH** under my guidance as partial fulfillment of the award of the degree of **Bachelor of Science in Mathematics** at St. Teresa's College (Autonomous), Ernakulam affiliated to Mahatma Gandhi University, Kottayam. No part of this work has been submitted for any other degree elsewhere.

Date:
Place: Ernakulam

Ms. Sreelakshmi K Anil
Assistant Professor,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

ST. TERESA'S COLLEGE (AUTONOMOUS) DEPARTMENT OF MATHEMATICS AND STATISTICS ERNAKULAM

Dr. Elizabeth Reshma M.T
Assistant Professor and Head,
Department of Mathematics,
St. Teresa's College(Autonomous),
Ernakulam.

28/01/26

External Examiners

1:.....

2: .....

Dr. SREEJA K.U. (PEN:716628)
Assistant Professor
Department of Mathematics
Maharaja's College
Ernakulam - 682 011

DECLARATION

I hereby declare that the work presented in this project is based on the original work done by me under the guidance of Ms. Sreelakshmi K Anil, Assistant Professor, Department of Mathematics, St. Teresa's College(Autonomous), Ernakulam and has not been included in any other project submitted previously for the award of any degree.

Place: Ernakulam

Date: 24.11.25

DEVIKA VIJESH

AB23BMAT015

ACKNOWLEDGEMENT

I would like to express my profound gratitude towards Ms. Sreelakshmi K Anil, our research guide, for their unwavering encouragement, wise counsel and strong support during this project. I truly appreciate the opportunity to learn from their instructions, and their expertise and patience helped me complete this assignment from start to finish. Additionally, I would like to express my gratitude to the entire staff of Department of Mathematics at St. Teresa's College for fostering a wonderful academic environment and providing me all the resources I required for my study.

Place: Ernakulam

Date: 24-11-2025

DEVIKA VIJESH

AB23BMAT015

Contents

CERTIFICATE ii
DECLARATION iii
ACKNOWLEDGEMENT iv
CONTENT v

1 INTRODUCTION 1
1.1 Introduction to Spam Filtering 2
1.2 Initial Approaches to Spam Classification 2
1.3 Challenges in Spam Filtering 3
1.4 Overview of Text Classification Techniques 4
1.5 Prior studies on Naïve Bayes and Centroid Cosine Method 6
1.5.1 Naïve Bayes in Spam Filtering 6
1.5.2 Centroid Cosine in Spam Filtering 7

2 NAÏVE BAYES METHOD 8
2.1 Fundamentals of Bayesian Probability Theory 8
2.2 Illustration 13
2.3 Implementation of algorithm 15
2.4 Advantages and Disadvantages 17

3 CENTROID COSINE METHOD 19
3.1 Theoretical Framework 19
3.1.1 Vectorization 20
3.1.2 Concept of Centroid 21
3.1.3 Cosine Similarity 23
3.1.4 Application of Centroid Cosine Method in Spam Filtering 24

3.2 Illustration 26
3.3 Implementation 27
3.4 Advantages and Disadvantages of the Centroid Cosine Method in Spam Filtering 28

4 RESULT AND ANALYSIS **31**
4.1 Methodology 31
4.2 Evaluation Matrix 33
4.3 Comparative Analysis 35
4.4 Observations 37
4.5 Conclusion 39
REFERENCES 40

Chapter 1

INTRODUCTION

Spam Filtering is a necessity for the modern communication systems, aiming to detect and eliminate unsolicited or harmful messages. This posses significant challenges to both individuals and organizations.

The objective of this project is to study the mathematical foundation of both Naïve Bayes and Centroid Cosine methods, to implement them in a spam classification task, and to compare their performance using standard evaluation measures such as accuracy, precision, and recall. The study analyzes the overall effectiveness of each method in distinguishing spam, also it seeks to answer the important questions such as: Which method provides higher precision? Which method can capture a larger share of all spam messages present? And finally, how do the mathematical assumptions of each method influence their practical performance in real data?

The project is organized in a structured way. The first chapter, which is this introduction, provides the background, importance, and reviews the existing literature, including earlier works on Naïve Bayes, cosine similarity and comparative studies in spam filtering. The second and third chapters include the mathematical foundation and implementation of Naïve Bayes and Centroid Cosine methods respectively, in spam classification. The fourth chapter explains the methodology used in this study, the results and discussion, comparing the performance of both the classifiers. Finally it concludes by summarizing the key findings, discussing the limitations and suggesting directions for future research.

Here, this chapter introduces the concept of spam filtering, outlines the challenges involved and also presents an overview of text classification techniques, including focus on the Centroid Cosine Method and Naïve Bayes classifiers.

1.1 Introduction to Spam Filtering

With the rapid growth of instant messaging and social media platforms, the number of unwanted and irrelevant messages, referred to as Spam messages, has also exponentially increased in digital communication. This may include advertising content, fraudulent schemes or malicious links designed to hack or to dox user's personal information. Apart from being a nuisance, contents can also have a dangerous side leading to identity theft, phishing attacks or financial loss. Reports from email service providers and internet security companies show that a large percentage of daily email traffic worldwide still consists of spam. Hence the importance of efficient spam detection methods arises as a problem in the field of information security.

1.2 Initial Approaches to Spam Classification

In the beginning, spam filtering relied on simple techniques like keyword rules or lists of known spam email addresses. These methods used fixed rules or patterns based on common features of spam. The rules could look for specific words, phrases or patterns in the email subject, body or sender details. If an email matched any of these rules, it would be marked as spam and is either deleted or moved to a spam folder [3]. But as spammers started changing their tactics to avoid these filters, traditional methods became less effective. This led to the need for more advanced solutions. Machine learning is one such solution. It uses mathematical models to analyze existing data and make predictions or classifications for new data. In spam detection, machine learning adapts to new spam patterns and can classify emails as spam or not spam. This method is more flexible than fixed rules because it can recognize complex patterns in language and word usage.

The problem that this project addresses is clear: Spam messages are a growing threat that cannot be reliably handled by traditional rule-based filters. Machine learning methods are necessary to provide adaptable and accurate solutions. This project is significant because spam detection has wide applications in everyday life. Email providers need strong spam filters to protect users from fraud and phishing. Mobile phone companies require SMS filters to block promotional and scam messages. Banks and e-commerce companies rely on spam detection to prevent fraudulent communication and protect customers. Therefore, by focusing on mathematical methods such as Naïve Bayes and Centroid Cosine, this project shows how abstract concepts from probability, statistics and vector space geometry are directly applied to solve a problem that affects millions of people worldwide.

1.3 Challenges in Spam Filtering

There are a range of dynamic and technical challenges faced by spam classification systems that complicate accurate classification. One of the main issues is evolving spam tactics, where the spammers continuously modify its contents, to avoid the detection mechanisms. Another critical challenge is false positives and false negatives- misclassifying legitimate emails as spam can disrupt communication, while failing to block actual spam exposes users to security risks.

Apart from the common issues, spam filters also face problems with spam that comes in the form of images or other media, instead of plain text, as it is harder to read and detect. Spammers also try to trick the system by changing words with symbols or random letters, so the filter cannot easily catch them. Another challenge is phishing mails that look very real and are often made for a particular person, so they are difficult to identify. Also, what one person sees as spam may be useful for another, so it is not easy to set one fixed rule for everyone. On top of that, scanning mails for spam can sometimes create privacy concerns, and using advanced filtering methods can be costly and require more computer resources.

Another thing to be considered is the scalability issue, as modern email servers must process millions of messages daily without compromising performance. Additionally, language and contextual diversity complicate filtering, since spam content varies across regions, cultures, and linguistic structures. Finally, real-time processing is essential for user experience and system integrity; filters must operate with minimal latency to ensure timely delivery and protection.

1.4 Overview of Text Classification Techniques

Text classification is now a key area in natural language processing and machine learning. It means putting text into certain groups based on what the text says. In the past, many different methods have been developed and used, starting from older statistical approaches up to more modern machine learning and deep learning techniques.

In the past, people mainly used manual rules and looked for specific keywords to classify text. But as technology evolved, machine learning became a popular choice. This led to the development and study of several algorithms, including Naive Bayes, k-Nearest Neighbor (k-NN), Support Vector Machines (SVM), Decision Trees, and Neural Networks. Each of these methods has its own way of dealing with features, uses different amounts of computing power, and works differently when it comes to scaling up to handle large amounts of data.

For example, the Naïve Bayes algorithm makes a key assumption that each feature in the data is independent of the others, and it uses probability calculations to make predictions. This approach is known for being quick and efficient, especially when dealing with big sets of data. In contrast, similarity-based methods like the 'Centroid Cosine method' focus on how close different document vectors are to each other in a multi-dimensional space. These methods are especially helpful when organizing text into groups or categories. In recent years, more advanced approaches such as ensemble methods and deep learning models, including Convolutional Neural Networks (CNNs) and Transformer-based models like BERT and GPT, have been used for text classification tasks.

These modern techniques often provide better results than traditional ones. However, older models like Naïve Bayes and the Centroid Cosine method are still widely used because they are easy to understand, require less computational power, and are straightforward to implement.

Among the various machine learning methods, Naïve Bayes and Centroid Cosine are two algorithms that are mathematically simple but effective in practice. Naïve Bayes is a probabilistic classifier based on Bayes' theorem. It assumes that all features (in this case, words in a message) are independent of each other. While this assumption is not entirely true in natural language, the method still performs surprisingly well in spam filtering. It calculates the probability of a message being spam, given the words it contains and classifies it accordingly. Due to its simplicity, swiftness and less requirement of training data, it has been used in many real-world spam filters. On the other hand, the Centroid Cosine method is based on vector space models of text. In this approach, every message is represented as a vector of word frequencies, and then the two centroids are computed; one for spam and one for non-spam messages. When a new message arrives, its vector is compared to the centroids using cosine similarity, and the message is classified in accordance with the closer centroid. This method is easy to implement, works directly with text features, and provides an intuitive geometric view of classification.

Although both Naïve Bayes and Centroid Cosine have been individually studied in the past, direct comparisons between the two are less common. Much of the existing research focuses on Naïve Bayes in detail or compares Naïve Bayes with more complex algorithms such as Support Vector Machines or Neural Networks. Similarly, cosine similarity is often studied as a tool for information retrieval or clustering, but not always in direct comparison with probabilistic classifiers. This research gap motivated for the present study to compare these two methods more closely. By doing so, we can highlight not only their strengths but also their weaknesses and determine which method is preferable under each circumstances.

1.5 Prior studies on Naïve Bayes and Centroid Cosine Method

1.5.1 Naïve Bayes in Spam Filtering

Among text classification applications, spam filtering has been a dominant area of research. Naïve Bayes classifiers have been extensively applied in detecting unsolicited emails due to their ability to handle high-dimensional datasets effectively. The algorithm works on probability, estimating whether an email belongs to spam or non-spam based on the frequency of words. Early studies, such as Sahami et al. [3], showed that Naïve Bayes could outperform rule-based spam filters by adapting to new spam trends automatically.

Later works improved the model by adding features such as n-grams, header analysis, and sender reputation [4]. Researchers also tested different versions like Multinomial and Bernoulli Naïve Bayes, where the multinomial model often showed better results in textrich datasets [5]. Further enhancements included feature selection methods such as Information Gain or Chi square to remove unnecessary words and improve accuracy [6]. Comparative studies with Decision Trees, SVMs, and hybrid models confirmed that Naïve Bayes, despite its simplifying independence assumption, continues to deliver competitive accuracy with very low computational cost [7]. Another advantage is its adaptability to concept drift in spam by incremental learning updates [8], and its successful application to multi SMS spam; even social media spam [9] and lingual spam. These qualities have made Naïve Bayes a benchmark method for spam detection and other text classification tasks, though its main limitation remains handling context where the same word may carry different meanings.

Results consistently showed that Naïve Bayes, despite its simplifying assumption of independence, provided competitive accuracy with very low computational cost. This balance of efficiency and performance has made it a benchmark method in spam filtering and broader text classification tasks.

1.5.2 Centroid Cosine in Spam Filtering

In contrast, the Centroid Cosine method takes a similarity-based approach, where each class (spam or non-spam) is represented by a centroid vector computed as the average of its feature vectors. New documents are classified by calculating the cosine similarity between their vector and each class centroid, with the highest similarity determining the label.

Previous research showed that this approach was effective in document clustering, categorization and information retrieval [10], particularly when combined with TF-IDF weighting schemes [11]. Later studies applied Centroid Cosine to spam filtering, demonstrating strong recall performance by catching more spam messages, although precision was sometimes lower than Naïve Bayes. Comparative analyses with k-NN confirmed that Centroid Cosine is less computationally heavy, since it avoids storing all training instances, making it more scalable for large datasets [12].

Researchers further improved this method by integrating dimensionality reduction techniques such as Latent Semantic Analysis [13] and PCA to reduce noise, as well as experimenting with advanced weighting schemes like BM25 [14]. Hybrid models combining cosine similarity with clustering algorithms such as k-means have also been explored to enhance scalability and robustness [15]. While Centroid Cosine may show slightly lower precision and perform less effectively on very short texts like SMS, its simplicity, scalability and robustness in high-dimensional datasets explain its continued relevance in spam detection research.

Chapter 2

NAÏVE BAYES METHOD

Naïve Bayes is fundamentally rooted in probability theory, particularly Bayes' Theorem, which provides a mathematical framework for updating the probability estimate for a hypothesis as more evidence becomes available. By modelling the likelihood of certain words or features occurring in spam versus legitimate (ham) emails, Naïve Bayes can effectively classify new emails based on learned probabilities. Despite its simplifying assumption that all features are independent—a condition rarely met in natural language—the algorithm often yields remarkably accurate results in practice. This chapter explores the theory behind Naïve Bayes, its application in spam detection, and its advantages and limitations in addressing one of the most critical cybersecurity challenges of the digital age.

2.1 Fundamentals of Bayesian Probability Theory

Bayes' Theorem

Bayes' theorem provides the mathematical foundation for updating probabilities based on new evidence. The theorem can be expressed as:

$$\mathbf{P(H|E)} = \frac{\mathbf{P(E|H)}P(H)}{P(E)}$$

Where:

- P(H|E) represents the posterior probability of hypothesis H given

evidence E

- **P(E|H) represents the likelihood of observing evidence E given hypothesis H**
- **P(H) represents the prior probability of hypothesis H**
- **P(E) represents the marginal probability of evidence E**

This elegant formulation encapsulates the essence of Bayesian reasoning: combining prior beliefs with observed evidence to obtain updated posterior beliefs.

Prior Probabilities

Prior probabilities, denoted as P(H), represent our initial beliefs about the probability of a hypothesis before observing any evidence. The selection of appropriate priors is crucial in Bayesian analysis and can significantly influence results, particularly when data is limited. Priors can be categorized into several types:

Informative priors incorporate substantial domain knowledge and reflect strong beliefs about parameter values. These priors are typically based on historical data, expert knowledge, or theoretical considerations. For example, in medical diagnosis, prior probabilities might be based on disease prevalence rates in the general population.

Non-informative Priors attempt to represent ignorance or minimal prior knowledge. These include uniform priors, which assign equal probability to all possible parameter values within a specified range, and Jeffreys priors, which remain invariant under parameter transformations.

Conjugate Priors are mathematically convenient choices that, when combined with certain likelihood functions, yield posterior distributions of the same functional form as the prior. This property simplifies computational procedures and provides analytical solutions in many cases.

The choice of prior distributions reflects both philosophical considerations about the nature of probability and practical concerns about computational tractability and interpretability of results.

Likelihood Functions

The likelihood function, P(E|H), measures the probability of seeing the evidence if a certain hypothesis is true. Unlike probability distributions, likelihoods don't have to add up to one across all possible evidence values. Instead, they show how well different hypotheses fit the observed data.

In parameter estimation, the likelihood function helps find the parameter values that best match the data. Maximum likelihood estimation picks the values that make the likelihood as large as possible. Bayesian methods combine the likelihood with prior knowledge to update beliefs and form posterior distributions.

For discrete data, calculating the likelihood is simple. But for continuous data, you need to use probability density functions and be careful about their properties. Log-likelihood is often used because it makes calculations easier and more stable.

Posterior Probabilities

Posterior probabilities, P(H|E), represent our updated beliefs about hypotheses after observing evidence. These probabilities synthesize prior knowledge with observed data, providing a complete probabilistic description of our uncertainty about hypotheses given the available information.

The computation of posterior probabilities often requires normalization by the marginal probability P(E), which can be computed as the sum (for discrete cases) or integral (for continuous cases) of the numerator $P(E|H) \times P(H)$ over all possible hypotheses. This normalization ensures that posterior probabilities sum to unity across all competing hypotheses.

Posterior probabilities serve multiple purposes in Bayesian analysis. They provide point estimates through measures such as the posterior mean or mode, interval estimates through credible intervals, and complete uncertainty quantification through the full posterior distribution. Sequential updating allows posterior probabilities from one analysis to

serve as priors for subsequent analyses as new data becomes available.

Naïve Bayes Classifier: Assumptions and Methodology

The Conditional Independence Assumption

The Naïve Bayes classifier derives its name from the "Naïve" assumption of conditional independence among features given the class label. Formally, this assumption states that for features $X_1, X_2, ..., X_n$ and class variable C:

P $(X_1, X_2, ..., X_n|\mathbf{C}) = \mathbf{P}(X_1|\mathbf{C}) \times \mathbf{P}(X_2|\mathbf{C}) \times ... \times \mathbf{P}(X_n|\mathbf{C})$

This assumption dramatically simplifies the computation of class probabilities by decomposing a high-dimensional joint probability into a product of simpler marginal probabilities. Without this assumption, estimating the joint probability P $(X_1, X_2, ..., X_n|\mathbf{C})$ **would require exponentially large amounts of data and computational resources.**

The mathematical elegance of this assumption enables efficient parameter estimation and classification. For a classification problem with k classes and n features, the conditional independence assumption reduces the number of parameters to estimate from O(k $\times$ 2^n**) to O(k** $\times$ **n), making the algorithm tractable for high-dimensional data.**

Despite being termed "Naïve," this assumption often performs remarkably well in practice, particularly for text classification tasks. The robustness of Naïve Bayes to violations of the independence assumption has been attributed to several factors, including the discriminative nature of the classification task and the averaging effects of combining many weakly dependent features.

Classification Mechanism

The Naïve Bayes classifier assigns new instances to the class with the highest posterior probability. For a new instance with features $x_1, x_2, ..., x_n$, the predicted class ĉ is:

$\hat{\mathbf{c}} = \arg\max \mathbf{P}(\mathbf{C} = \mathbf{c}|x_1, x_2, ..., x_n)$

Using Bayes' theorem and the conditional independence assumption:

$\hat{\mathbf{c}} = \arg\max \mathbf{P}(\mathbf{C} = \mathbf{c}) \times \prod_{i=1}^{n} \mathbf{P}(x_n|\mathbf{C} = \mathbf{c})$

The marginal probability $P(x_1, x_2, ..., x_n)$ is omitted from the calculation as it remains constant across all classes and does not affect the relative ranking of posterior probabilities.

Parameter estimation in Naïve Bayes typically employs maximum likelihood estimation. For categorical features, this involves computing relative frequencies, while for continuous features, parameters of assumed probability distributions (such as Gaussian parameters) are estimated from training data.

Applications and Limitations in Real-World Text Data

Text Classification Applications

Text classification represents one of the most successful application domains for Naïve Bayes classifiers. The bag-of-words representation commonly used in text processing naturally aligns with the conditional independence assumption, as word occurrences are treated as independent features given the document class.

Spam Detection was among the early successful applications of Naïve Bayes in text processing. Email messages are represented as vectors of word frequencies, and the classifier learns to distinguish between spam and legitimate messages based on the probability distributions of words within each class. The effectiveness of Naïve Bayes in this domain stems from the discriminative power of individual words and phrases characteristic of spam messages.

Sentiment Analysis leverages Naïve Bayes to classify text documents according to their emotional tone or opinion polarity. Features may include individual words, n-grams, or more sophisticated linguistic features. The assumption that individual words contribute independently to the overall sentiment often provides reasonable approximations, despite obvious violations such as negation and context-dependent sentiment.

Topic Classification applies Naïve Bayes to categorize documents into predefined topics or subjects. News articles, academic papers, and web pages can be classified using word frequency patterns characteristic

of different domains. The success of this approach demonstrates the robustness of Naïve Bayes to the violation of independence assumptions that inevitably occur in natural language.

Mitigation Strategies and Improvements

Several methods have been created to overcome the limitations of Naïve Bayes while keeping it fast and easy to understand. One way is feature engineering, which creates new features like n-grams, parts of speech, and grammar patterns to show some relationships between words. This can help the model perform better, but it might also make the model more complex and slower.

Another method is smoothing, which helps when there's not enough data. Techniques like Laplace smoothing add a small value to all counts to avoid zero probabilities, while methods like Good-Turing smoothing offer better ways to estimate probabilities when data is sparse.

Ensemble methods combine Naïve Bayes with other models to use their strengths. Techniques like voting, bagging, and boosting can improve the model's accuracy while keeping the benefits of Naïve Bayes, like being easy to explain.

Preprocessing steps can also help by reducing the effect of word independence. These include cleaning the text, removing common words, shortening words, and selecting the most useful features. These steps aim to keep the important parts of the data while removing unnecessary or misleading information.

2.2 Illustration

This chapter is meant to illustrate on how the naïve bayes method is used to classify spam messages.

Training data:

Suppose we have a small dataset of three messages such as Spam: "win money", Spam: "free money", Ham: "see you tomorrow"

Step 1: Vocabulary

Vocabulary= {win, money, free, see, you, tomorrow} This is a total of 6 words.

Step 2: Word count by class

spam messages: win (1), free (1), money (2), total spam words= 4
ham messages: see (1), you (1), tomorrow (1), total ham words= 3

Step 3: Applying Laplace smoothing

P(w | class) = [count(w,class)+1]/[total words in class + |V|]
Estimating word likelihoods with Laplace smoothing(α=1).
For Spam: denominator =no. of words + V(total count) = 4 + 6 = 10

- **P(win | spam) = (1+1)/10 = 0.2**
- **P(free | spam) = (1+1)/10 = 0.2**
- **P(money | spam) = (2+1)/10 = 0.3**
- **P(see | spam) = (0+1)/10 = 0.1**
- **P(you | spam) = 0.1**
- **P(tomorrow | spam) =0.1**

For Ham: denominator = 3 + 6 = 9

- **P(win | ham) = (0+1)/9 = 0.111..**
- **P(free | ham) = 0.111..**
- **P(money | ham) = 0.111..**
- **P(see | ham) = (1+1)/9 = 0.222..**
- **P(you | ham) = 0.222..**
- **P(tomorrow | ham) =0.222..**

Classifying new message:

Let the message be "win free prize". Tokens = win, free, prize Since prize is an unseen word, it gets smoothing probability.

- **Spam: denominator = 10; unseen word = 1/10=0.1**
- **Ham: denominator = 9; unseen word = 1/9=0.111..**

Setting Priors: Assume P(Spam) = P(Ham) = 0.5

Step 4: Computing Likelihood for each class

Priors are multiplied with likelihoods for each word in new message.
Spam likelihood: P(Spam) x P (win | spam) x P(free | spam) x P(prize | spam) = 0.5 x 0.2 x 0.2 x 0.1 = 0.002
Ham likelihood: P(Ham) x P (win | Ham) x P(free | Ham) x P(prize | Ham) = 0.5 x 0.111 x 0.111 x 0.111 = 0.00068

Step 5: Decision

The class with higher posterior score is chosen. $0.002 > 0.00068$
Hence the new message is classified as Spam.

From this example it is clear that Naïve Bayes uses probabilities for classification, handles unseen words with Laplace Smoothing, and compares likelihoods across classes for spam filtering.

2.3 Implementation of algorithm

The implementation consists of two stages ;

- **Training stage: In the training stage, the algorithm learns from a labelled dataset. It calculates the prior probability of each class (such as spam or ham) and the likelihood of each word appearing in those classes. These probabilities are stored for future use. Text preprocessing steps such as tokenization and removal of stop words are also applied during this stage.**

- **Prediction stage: In the prediction stage, the trained model is used to classify new, unseen messages. The email is first pre-processed in the same way as the training data. Then, the model uses Bayes' Theorem to compute the probability of the email belonging to each class, based on the words it contains. The class with the highest probability is selected as the final classification.**

```python
def train_naive_bayes(dataset):
    """Train a simple multinomial NB with Laplace smoothing"""
    class_counts = Counter()
    word_counts = defaultdict(Counter)
    vocab = set()
    total_docs = 0

    for label, text in dataset:
        total_docs += 1
        class_counts[label] += 1
        words = tokenize(text)
        for w in words:
            word_counts[label][w] += 1
            vocab.add(w)

    # compute priors and likelihoods (log-space)
    model = {
        'priors': {},
        'log_likelihoods': {},
        'vocab_size': len(vocab),
        'vocab': vocab
    }
    for c in class_counts:
        model['priors'][c] = math.log(class_counts[c] / total_docs)
        # denom for Laplace: total words in class + V
        total_words = sum(word_counts[c].values())
        denom = total_words + model['vocab_size']  # Laplace
        model['log_likelihoods'][c] = {}
        for w in vocab:
            count = word_counts[c].get(w, 0)
            model['log_likelihoods'][c][w] = math.log((count + 1) / denom)
    return model
```

```python
    log_probs = {}
    for c in model['priors']:
        logp = model['priors'][c]
        for w in words:
            # if word unseen in vocab, use uniform smoothing probability
            if w in model['vocab']:
                logp += model['log_likelihoods'][c].get(w, math.log(1 / (model['vocab_size'])))
            else:
                # unseen word: approximate with Laplace on zero count (1/(total+V)) ~ small
                logp += math.log(1 / (model['vocab_size'] + 1))
        log_probs[c] = logp

    # convert log probs to normalized probability using log-sum-exp
    max_log = max(log_probs.values())
    sum_exp = sum(math.exp(lp - max_log) for lp in log_probs.values())
    probs = {c: math.exp(lp - max_log) / sum_exp for c, lp in log_probs.items()}

    # Compute top contributing tokens (difference in log-likelihoods spam - ham)
    contributors = []
    if 'spam' in model['log_likelihoods'] and 'ham' in model['log_likelihoods']:
        token_scores = {}
        for w in set(words):
            score = model['log_likelihoods']['spam'].get(w, 0) - model['log_likelihoods']['ham'].get(w, 0)
            token_scores[w] = score
        # sort by absolute impact
        sorted_tokens = sorted(token_scores.items(), key=lambda kv: abs(kv[1]), reverse=True)
        contributors = [t for t, s in sorted_tokens[:6]]

    # decide label based on higher prob
    label = "Likely Spam" if probs.get('spam', 0) > probs.get('ham', 0) else "Likely Ham (Not Spam)"
    # Use spam probability if available else 0.0
    spam_prob = probs.get('spam', 0.0)
    return label, spam_prob, contributors
```

This implementation produces a functional model capable of distinguishing between spam and ham.

2.4 Advantages and Disadvantages

Advantages

One of the main benefits of Naïve Bayes is how fast it can work. The algorithm runs quickly because it has a simple time requirement, which makes it great for real-time filtering of spam in big email systems. Its use of probabilities lets it classify emails quickly by looking at how often certain words appear, which works well when dealing with a lot of different information like email text.

Naïve Bayes is also easy to set up and doesn't need much fine-tuning. It can do a good job even when it has only a small amount of training data, and it doesn't get too affected by extra or unnecessary information. This is because it makes a simple assumption that features are independent, which helps it work well with different types of emails and content.

Another good thing is that Naïve Bayes can learn as it goes. It can keep updating its knowledge as new data comes in, which helps spam filters stay up-to-date with new spam tricks. Research has also shown that Naïve Bayes often works as well as more complicated methods, making it a trusted starting point in studies about detecting spam.

Disadvantages

Despite its practical success, the application of Naïve Bayes to real-world text data faces several significant limitations stemming from violations of its fundamental assumptions.

Linguistic Dependencies represent the most obvious violation of the conditional independence assumption in text data. Natural language exhibits complex syntactic and semantic dependencies that cannot be captured by treating words as independent features. For example, the phrase "not good" conveys a different meaning than would be predicted by considering "not" and "good" independently. Similarly, idiomatic expressions, compound terms, and contextual word meanings all violate the independence assumption.

Feature Correlation in text data extends beyond simple linguistic dependencies. Topical coherence within documents creates correlations among semantically related terms. Documents about sports are likely to contain multiple sports-related terms, while technical documents contain clusters of domain-specific vocabulary. These correlations can lead to overconfident predictions and suboptimal classification performance.

Data Sparsity poses computational and statistical challenges in text classification applications. The high dimensionality of text feature spaces, combined with the zipfian distribution of word frequencies, results in sparse feature matrices where most word-document combinations have zero frequency. This sparsity complicates parameter estimation and necessitates smoothing techniques to handle unseen word-class combinations.

Context Sensitivity limitations become apparent when word meanings depend heavily on surrounding context. Polysemous words, sarcasm, and domain-specific terminology all challenge the assumption that word probabilities depend only on the document class and not on other words present in the document.

Chapter 3

CENTROID COSINE METHOD

3.1 Theoretical Framework

The Centroid Cosine Method (CCM) represents a similarity-driven technique for classifying text documents through comparison with representative centroids from established categories. This approach builds upon the Vector Space Model (VSM), which transforms documents into vectors within a multi-dimensional space. Every dimension relates to a distinct term from the vocabulary, with term weights commonly determined through Term Frequency–Inverse Document Frequency (TF-IDF) calculations. This framework encapsulates both how frequently terms appear in documents and their ability to distinguish between different categories in the collection.

Within this framework, each category receives its own centroid through averaging the vectors from all training documents in that specific group. Taking spam detection as an example, the system creates two centroids: one representing spam content and another for genuine (ham) correspondence. These centroids function as "templates" for their corresponding categories, offering single comparison points instead of maintaining entire training document collections. Upon encountering a fresh document, the system transforms it into vector format using identical preprocessing and weighting techniques. Similarity calculations between this document vector and every category centroid employ cosine similarity measurements, which determine the cosine value of angles

between vector pairs. Smaller angles between document vectors and category centroids indicate stronger similarity relationships. The system ultimately places documents into categories showing the strongest similarity measurements.

3.1.1 Vectorization

Textual data such as email messages cannot be directly processed by mathematical algorithms; instead, they must be converted into numerical form. This transformation process, known as vectorization, maps each email into a point in a high-dimensional feature space. In spam filtering, vectorization serves as the foundation for all subsequent steps, including centroid computation and similarity measurement.

Preprocessing and Tokenization

The first step involves standard preprocessing: case-folding, punctuation removal, tokenization, and optional stemming or lemmatization. Frequently occurring but semantically weak words (e.g., "the," "and") may be removed as stop words to reduce noise. Tokenization choices affect the granularity of features: unigram tokens capture single words, while bigrams or trigrams capture short phrases such as "free offer" or "click link," which are highly indicative of spam.

Feature Representation

Once tokens are identified, they are mapped into vectors using one of several schemes:

1. Bag-of-Words (BoW): represents the frequency of each word in a message.
2. Term Frequency–Inverse Document Frequency (TF–IDF): scales raw counts by inverse document frequency, giving higher weight to discriminative terms such as "viagra" and lower weight to common words such as "email" [20, 21]
3. Character n-grams: effective for detecting obfuscated spam words (e.g., "fr33" instead of "free").

4. High-Dimensional and Sparse Representations

The resulting feature space is typically very high-dimensional, with tens of thousands of unique terms. Most individual documents, however, use only a small fraction of these, resulting in sparse vectors. Efficient data structures (e.g., sparse matrices) are therefore employed for storage and computation [17]

Normalization and Feature Selection

Since document lengths vary widely, normalization is essential before computing similarities. L2 normalization scales each vector to unit length, ensuring that cosine similarity depends only on the orientation of vectors, not their magnitude. This makes comparison between short and long messages meaningful.

To reduce computational overhead and enhance discriminative power, dimensionality reduction or feature selection techniques may be applied. Common approaches include vocabulary trimming based on minimum document frequency or statistical criteria such as chi-square or mutual information [22]

In summary, vectorization provides the mathematical representation that enables the computation of centroids and cosine similarity. The choice of vectorization scheme significantly influences the performance of centroid-based spam filtering systems.

3.1.2 Concept of Centroid

The centroid is a fundamental construct in vector space models and forms the core of centroid-based classification. In the context of spam filtering, a centroid represents the "average document" of a class (e.g., spam or ham), providing a compact and interpretable summary of that class's lexical profile.

Mathematical Definition

For a class with document vectors, the class centroid is defined as the mean vector:

$$\mu_C = \frac{1}{n} \sum_{i=1}^{n} v_i$$

where each is typically a TF–IDF or term-frequency normalized vector. This centroid captures the "average position" as a single representative of the entire class in the high-dimensional feature space.

Historical Background

The centroid concept is closely linked to the Rocchio classifier (Salton, 1971), originally developed for relevance feedback in information retrieval. Rocchio's method uses centroids of relevant and non-relevant documents to refine query vectors, and this idea was later adapted for supervised text classification [17].

Role in Classification

Classification using centroids follows a straightforward principle:

1. Compute centroids for each class (spam and ham).
2. For a new message vector, compute similarity (e.g., cosine) with each centroid.
3. Assign to the class with the highest similarity.

This approach is computationally efficient since it requires only two similarity calculations per test document in a binary classification setting.

Variants

Weighted Centroids: Centroids can incorporate term weighting schemes beyond TF–IDF, such as the Class-Feature-Centroid (CFC) approach, which emphasizes features that best discriminate between classes [25].

Multiple Centroids per Class: Since spam messages are often heterogeneous (e.g., advertisements, phishing, lottery scams), a single centroid may not capture all variations. Partitioning training data into clusters and computing multiple centroids improves coverage of class subspaces [24].

Incremental Updating: Centroids can be updated incrementally as new labeled data arrive, making them suitable for streaming environments where spam content evolves over time [19].

Strengths and Limitations

Centroid classifiers are simple, interpretable, and computationally efficient. However, they struggle when class distributions are multimodal or heavily overlapping.

Advantages: simplicity, interpretability (the centroid's top-weighted terms reveal the characteristic vocabulary of spam), and low computational cost during both training and classification.

Limitations: effectiveness decreases when class distributions are not linearly separable or are highly multimodal, as often occurs with sophisticated and diverse spam campaigns.

In summary, the centroid provides a mathematically elegant and computationally efficient representation of a document class. Its utility in spam filtering lies in its ability to capture the overall lexical tendencies of spam versus legitimate emails, while still being simple enough to support fast, large-scale filtering.

3.1.3 Cosine Similarity

In centroid-based text classification, cosine similarity is the most widely used measure for comparing document vectors. Cosine similarity measures the closeness between two vectors by comparing their orientation rather than their magnitude. This property is particularly useful in spam filtering, where email lengths vary significantly.

Mathematical Definition

For two vectors x and y, the cosine similarity is defined as:

$$\textbf{cosine}(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|}$$

The resulting value ranges from to, but for non-negative vectors (e.g., TF–IDF representations), the range is. A value closer to 1 indicates a

higher degree of similarity.

Advantages

Length Independence: Cosine similarity measures the angle between vectors, making it robust to variations in document length. For example, a short spam email and a long spam email may both align closely with the spam centroid despite their different lengths.

Sparse Data Efficiency: Since most document vectors are sparse, cosine similarity computations can be implemented efficiently by focusing only on non-zero terms.

Discrimination Power: Empirical studies show that cosine similarity outperforms simple distance measures such as Euclidean distance in text categorization tasks [23].

Role in Classification

In spam filtering, the cosine similarity is used to determine the proximity of a test message vector to the spam centroid and ham centroid. The decision rule is: $\hat{y} = \arg\max_{c \in \{\text{spam,ham}\}} \text{cosine}(q, \mu_c)$ Alternatively, a difference score can be computed as: $\Delta = \text{cosine}(q, \mu_{\text{spam}}) - \text{cosine}(q, \mu_{\text{ham}})$ and classified as spam if, where is a tunable threshold.

Applications in Spam Filtering

Cosine similarity has been applied in spam detection to efficiently classify large volumes of email using centroid-based methods. Its simplicity allows real-time filtering while maintaining strong performance on benchmark datasets such as Ling-Spam and Enron [1, 24].

3.1.4 Application of Centroid Cosine Method in Spam Filtering

The Centroid Cosine Method integrates vectorization, centroid computation, and cosine similarity into a simple yet effective spam filtering framework. Its appeal lies in its low computational cost, interpretability, and ease of implementation, making it a popular baseline in text classification research.

Workflow

1. **Dataset Preparation Collect and preprocess labeled datasets of spam and ham emails. Popular corpora include Ling-Spam, SpamAssassin, and Enron email datasets [1, 5]. Apply preprocessing (tokenization, stopword removal, stemming/lemmatization).**

2. **Vectorization Represent each email as a vector using Bag-of-Words or TF–IDF features. Normalize vectors to unit length to make cosine similarity independent of document length.**

3. **Centroid Computation Compute centroids for spam () and ham () by averaging their training vectors. In cases where spam is highly diverse, multiple centroids can be created by clustering the spam training set [24].**

4. **Classification For a new email vector , compute cosine similarities:** $S_{\mathbf{spam}} = \mathbf{cosine}(q, \mu_{\mathbf{spam}}), \quad S_{\mathbf{ham}} = \mathbf{cosine}(q, \mu_{\mathbf{ham}})$

5. **Evaluation Performance is measured using metrics such as accuracy, precision, recall, F1-score, and false positive rate. False positives (legitimate emails marked as spam) are especially critical in real-world deployments [23].**

Real-World Applications

1. **The centroid cosine method is used as:**

2. **A standalone filter in lightweight systems where speed is prioritized.**

3. **A baseline model in research for benchmarking advanced classifiers.**

A component in hybrid systems, combined with probabilistic or ensemble methods to boost robustness [25].

In summary, the centroid cosine method provides a practical, transparent, and computationally efficient approach to spam filtering. Despite its limitations, it remains valuable both as a baseline for research and as a fast-filtering component in production systems.

3.2 Illustration

Here in this section we illustrate how the centroid cosine method works in the classification of Spam messages.

Training data:

Let's take three short messages Spam: "win money", Spam: "free money", Ham: "see you tomorrow"

Step 1: Vocabulary

{win, money, free, see, you, tomorrow}

Step 2: Vectorization

Each phrase becomes a binary vector (1 = word present, 0 = absent):
"win money" = [1, 1, 0, 0, 0, 0]
"free money" = [0, 1, 1, 0, 0, 0]
"see you tomorrow" = [0, 0, 0, 1, 1, 1]

Step 3: Centroids

• Spam Centroid = Average of spam-like vectors → from "win money" and "free money":
Spam Centroid = $\frac{[1,1,0]+[0,1,1]}{2}$ = [0.5, 1, 0.5]
• Ham Centroid = from "see you tomorrow":
Ham Centroid = [0, 0, 0, 1, 1, 1]

Classify a New Email:

Let's say a new email says: "free money now"
Vector: [0, 1, 1, 0, 0, 0]

Step 4: Cosine Similarity

Cosine Similarity with Spam Centroid:
similarity = $\frac{(0*0.5+1*1+1*0.5)}{\sqrt{0^2+1^2+1^2}\cdot\sqrt{0.5^2+1^2+0.5^2}} = \frac{1+0.5}{\sqrt{2}\cdot\sqrt{1.5}} \approx \frac{1.5}{1.414\cdot 1.225} \approx 0.87$
Cosine Similarity with Ham Centroid: similarity = 0 (no overlapping words)

Since the new email is much closer to the spam centroid, it's classified as spam.

3.3 Implementation

This section details the practical implementation of the Centroid cosine method for spam filtering, highlighting the key step and core code responsible for classification. This Method was implemented in Python to classify messages as spam or ham. The dataset preprocessing steps are as described in the coming Methodology section.

The implementation consisted of three main stages: Vectorization, Centroid construction and Classification using cosine similarity.

```python
def build_centroid_model(dataset):
    """Build class centroids (term-frequency normalized) from dataset"""
    class_word_counts = defaultdict(Counter)
    class_total_words = Counter()
    vocab = set()

    for label, text in dataset:
        words = tokenize(text)
        for w in words:
            class_word_counts[label][w] += 1
            class_total_words[label] += 1
            vocab.add(w)

    centroids = {}
    for label, counts in class_word_counts.items():
        total = class_total_words[label] if class_total_words[label] > 0 else 1
        centroid = {}
        for w in vocab:
            # term frequency normalized by total words in class
            centroid[w] = counts.get(w, 0) / total
        centroids[label] = centroid

    # precompute centroid norms for cosine
    centroid_norms = {}
    for label, centroid in centroids.items():
        norm = math.sqrt(sum(v * v for v in centroid.values()))
        centroid_norms[label] = norm if norm > 0 else 1.0

    return {
        'centroids': centroids,
        'centroid_norms': centroid_norms,
        'vocab': vocab
    }
```

```python
    # compute cosine similarity to each centroid
    sims = {}
    for label, centroid in model['centroids'].items():
        dot = 0.0
        for w, val in tf_vec.items():
            if w in centroid:
                dot += val * centroid[w]
        denom = model['centroid_norms'].get(label, 1.0) *
        math.sqrt(sum(v * v for v in tf_vec.values()))
        sims[label] = dot / (denom if denom > 0 else 1.0)

    # convert similarities to probability-like score for spam
    sim_spam = sims.get('spam', 0.0)
    sim_ham = sims.get('ham', 0.0)
    eps = 1e-9
    spam_prob = sim_spam / (sim_spam + sim_ham + eps)

    label = "Likely Spam" if spam_prob > 0.5 else "Likely Ham (Not Spam)"
```

The implementation produced a functional model capable of distinguishing between spam and ham emails. Its performance is analyzed in

the Evaluation chapter.

3.4 Advantages and Disadvantages of the Centroid Cosine Method in Spam Filtering

The Centroid Cosine Method has been widely studied as a baseline classifier in text categorization and spam filtering. Its effectiveness comes from its simplicity and efficiency, though it also presents notable limitations when compared to more advanced machine learning approaches.

Advantages

1. Simplicity and Interpretability The method is conceptually straightforward: each class is represented by its centroid, and classification is based on cosine similarity. It is easy to explain to non-technical users, and the centroid terms often reveal the dominant vocabulary of spam and ham classes [24].

2. Computational Efficiency Training involves only computing class centroids, making it far faster than iterative optimization methods such as SVMs or neural networks. Classifying new emails requires only a similarity calculation with centroids, which is efficient in high-dimensional sparse spaces [17]. By condensing each category into one representative centroid, it significantly cuts down processing time compared to techniques that must evaluate documents against every training example.

3. Robustness to Document Length Cosine similarity normalizes vector lengths, making the method effective regardless of whether an email is very short (e.g., "Win prize now") or long (e.g., a newsletter).

4. Scalability The approach is well-suited for large datasets, as centroids are compact representations and do not grow with the number of training samples.

5. Useful Baseline and Ensemble Component Often used as a baseline classifier in comparative studies [23]. Can be integrated into hybrid

or ensemble spam filters, where its simplicity complements more complex models [25].

Disadvantages

Apart from the advantages this technique shows vulnerability to corrupted and imbalanced datasets.

1. Oversimplified Representation Using a single centroid per class assumes the class distribution is unimodal. In reality, spam messages are highly diverse (phishing, promotions, malware), and one centroid cannot capture this variation effectively [26] as the central point can become distorted when certain categories overwhelm others or when meaningless features interfere with the process.

2. Concept Drift Sensitivity Spam vocabulary evolves rapidly. Static centroids become outdated unless updated frequently, making the method less effective in real-world dynamic environments [19].

3. Limited Discriminative Power Compared to discriminative models such as Support Vector Machines or deep learning, centroid-based classification generally yields lower accuracy, particularly on complex or imbalanced datasets [5].

4. Feature Dependence Performance is highly dependent on the chosen vectorization technique (e.g., BoW vs. TF–IDF vs. n-grams). Poor feature representation can severely reduce effectiveness.

5. High False Positive Risk Because cosine similarity is sensitive to shared tokens, legitimate messages containing spam-like vocabulary (e.g., "free seminar") may be incorrectly flagged as spam, a major usability issue [1].

Summary

The Centroid Cosine Method balances efficiency and interpretability with limited flexibility and adaptability. While it provides a strong baseline for spam filtering research, real-world deployments often re-

quire enhancements such as multiple centroids, incremental updates, or integration into ensemble systems.

Chapter 4

RESULT AND ANALYSIS

This chapter presents the analysis and experimental results of applying the Naïve Bayes classifier and the Centroid Cosine method to the SMS Spam Collection dataset. Both models were trained and tested on the same dataset under identical preprocessing conditions, ensuring fairness in evaluation. The primary objective of this study is to compare the effectiveness of these two methods in filtering spam messages from legitimate (ham) messages. Results are analyzed with respect to standard performance metrics including confusion matrix, accuracy, precision, recall, and F1 score. The chapter concludes with comparative insights and key observations derived from the study.

4.1 Methodology

The methodology adopted for this project involves applying two distinct classification techniques:

1. Naïve Bayes Classifier:

   - A probabilistic model that applies Bayes' theorem, assuming independence between features (tokens). It estimates the probability of a message being spam or ham based on word occurrences.
   - Uses the Bag-of-Words model for feature extraction.
   - Laplace Smoothing was applied to handle unseen tokens.

2. Centroid Cosine Classifier:

- **A vector space model where messages are represented as word frequency vectors. Class centroids are built for spam and ham messages, and classification is performed using cosine similarity to determine the closest centroid.**
- **Term frequency normalization was used for centroid construction.**

Both the methods followed the same preprocessing pipeline: tokenization, lowercasing, stopword removal and stemming.

Experimental Setup

A structured experimental setup was employed to evaluate the effectiveness of the Centroid Cosine Method in comparison with the Naïve Bayes classifier for spam filtering. This section describes the dataset, preprocessing steps, system configuration and parameter settings used in the experiment.

The experiments were conducted using the SMS Spam Collection dataset, which contains 5572 labelled SMS messages, of which 747 are spam and 4825 are ham.

Data Preprocessing

The dataset consisted of SMS messages labeled as 'spam' or 'ham (non-spam)'. Since raw textual data often contains noise, preprocessing was crucial.

- **Normalization: All text was converted to lowercase to avoid case-sensitivity issues.**
- **Noise Removal: Punctuation, numbers, and non-alphabetic symbols were stripped out.**
- **Stopword Removal: Frequently occurring but semantically weak words (e.g., "is", "the", "of") were removed.**
- **Stemming/Lemmatization: Words were reduced to their root forms (e.g., "running" → "run"), ensuring consistency.**

Feature Extraction

Two vectorization methods were used:

- **Bag of Words (BoW): Captures word frequency information.**
- **TF-IDF (Term Frequency–Inverse Document Frequency): Provides weighted importance of words based on their occurrence across documents.**

System Configuration

- **The programming language used here is Python and the libraries like collections, math were included.**
- **IDE used: VSCode**

Parameter settings

- **Naïve Bayes: Laplace smoothing ($\alpha = 1$)**
- **Centroid Cosine: similarity measured by cosine score**

Training and Testing

Both classifiers were trained on the same dataset, with class-specific centroids built for Centroid Cosine and conditional probabilities computed for Naïve Bayes.

Evaluation

Confusion matrices were generated, and evaluation metrics (accuracy, precision, recall, and F1-score) were computed.

4.2 Evaluation Matrix

Performance evaluation was conducted using standard classification metrics derived from the confusion matrix. These metrics capture not only the overall correctness of classification but also how well the models handle spam detection specifically.

Confusion Matrix

The confusion matrix summarizes predictions in terms of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

- **TP (True Positives): Spam messages correctly identified as spam.**
- **TN (True Negatives): Ham messages correctly identified as ham.**
- **FP (False Positives): Ham messages wrongly classified as spam.**
- **FN (False Negatives): Spam messages wrongly classified as ham.**

Classifier	TP	TN	FP	FN
Naïve Bayes	629	3792	1033	118
Centroid Cosine	649	3405	1420	98

Table 4.1: Confusion Matrix

Accuracy

Accuracy, also measured as efficiency, reflects the classifiers' ability to correctly identify both spam and ham messages.

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

Naïve Bayes: Efficiency $= \frac{629+3792}{5572} =$ **79.34%**
Centroid Cosine: Efficiency $= \frac{649+3405}{5572} =$ **72.76%**

Interpretation: Although both models achieve high accuracy, Naïve Bayes performs slightly better overall.

Precision

Of all emails predicted as spam, how many are actually spam.

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

Naïve Bayes: 37.85% ; Centroid Cosine: 31.37%

Interpretation: Naive Bayes has higher precision, meaning when it predicts spam, it is more often correct. This is important in applications where false alarms must be minimized.

Recall

Of all actual spam emails, how many are detected as spam.

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

Naïve Bayes: 84.20% ; Centroid Cosine: 86.88%

Interpretation: Centroid Cosine detects more spam (higher recall). This makes it more reliable in catching malicious or unwanted emails, even at the risk of slightly over-flagging.

F1 Score

Harmonic mean of Precision and Recall, ie, Balance between precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Naïve Bayes: 52.22% ; Centroid Cosine: 46.09%

Interpretation: Since the F1 score balances precision and recall, Naïve Bayes demonstrates stronger overall spam detection capability.

4.3 Comparative Analysis

To clearly visualize the performance of both classifiers, the results are presented below.

Metric	Naïve Bayes	Centroid Cosine
Accuracy	79.34%	72.76%
Precision	37.85%	31.37%
Recall	84.20%	86.88%
F1 Score	52.22%	46.09%

Table 4.2: RESULT

Graphical Representation:

The bar chart representation comparing the Accuracy, Precision, Recall and F1 score of both classifiers are shown below.

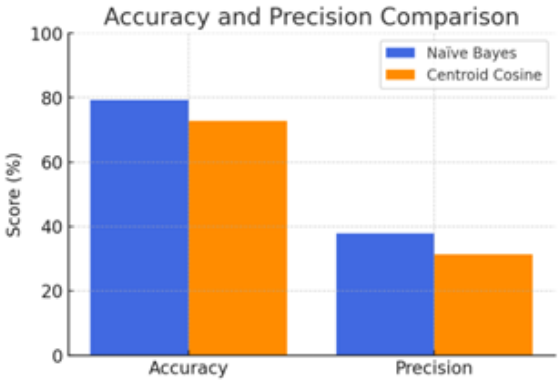


Figure 4.1: Accuracy and Precision Representation

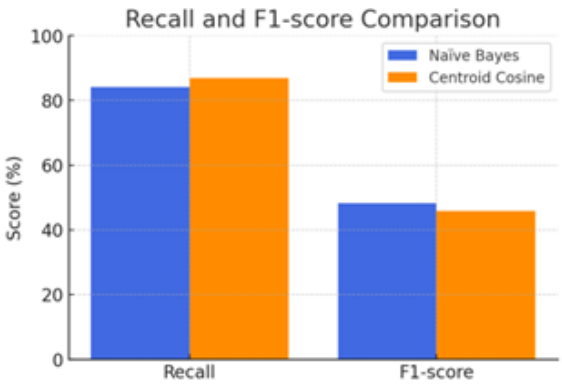


Figure 4.2: Recall and F1-score Representation

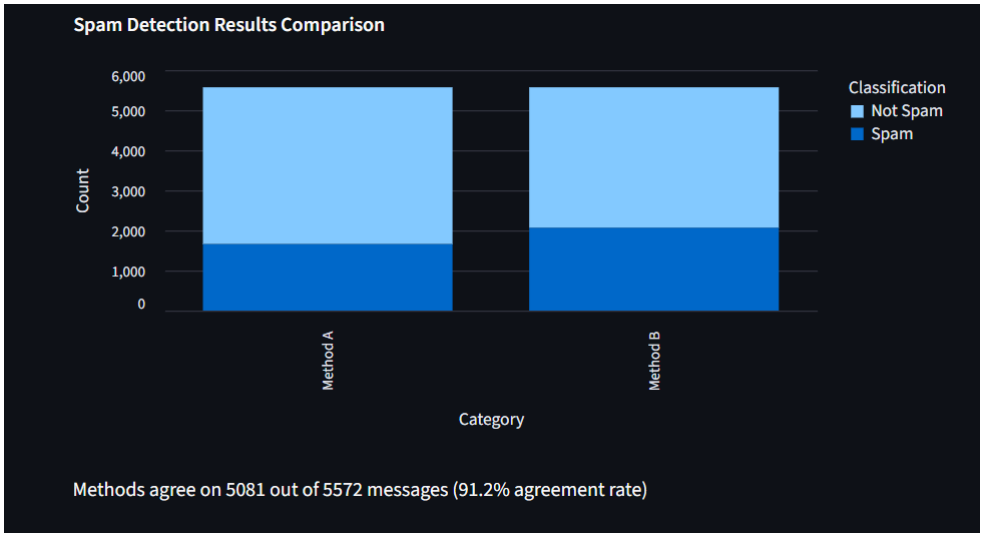
Comparative Insights:

- **Naïve Bayes outperforms Centroid Cosine in accuracy, precision and F1-score, making it more reliable in practice.**
- **Centroid Cosine achieved slightly higher recall, indicating it detects marginally more spam messages. However, this comes at the cost of higher false positives, which lowers both precision and overall efficiency.**
- **In real-world spam filtering, precision is crucial to avoid misclassifying legitimate emails as spam. By this measure, Naïve Bayes is better suited for deployment in practical filtering systems.**

Agreement Analysis

It is defined as the proportion of messages both classifiers agreed on. The purpose of this analysis was to determine not only their individual strengths but also how often they arrived at the same classification decision. Here the two classifiers achieved a 91.2% agreement rate, meaning that in the majority of cases, both methods classified the same messages consistently as either spam or ham. This high level of agreement suggests that despite their methodological differences—probabilistic modeling in Naïve Bayes versus similarity-based centroid calculation in Centroid Cosine—their decision boundaries overlap significantly for most

messages.



4.4 Observations

- Naïve Bayes achieved higher efficiency (79.34%) than the Centroid Cosine method (72.76%). This indicates that Naïve Bayes correctly classified more messages overall, demonstrating stronger general performance.
- The 91.2% agreement rate between the two methods further shows that they behave similarly for most inputs, but the efficiency difference lies in Naïve Bayes' ability to handle large-scale filtering tasks with fewer computational resources.
- While Centroid Cosine achieves slightly better recall (it catches more spam), it suffers from a higher false positive rate, lowering its overall efficiency. Naïve Bayes strikes a better balance, reducing false positives while maintaining strong recall.
- In spam filtering, efficiency is crucial because users demand both reliable spam detection and minimal disruption to legitimate communication. The higher efficiency of Naïve Bayes suggests it is more suitable for practical deployment in real-world spam filtering systems.
- Overall, Naïve Bayes is the more effective classifier for SMS spam filtering, offering a stronger balance between spam detection and

preservation of legitimate messages. Centroid Cosine, while simpler and effective at spam recall, produces more false alarms, which can reduce usability in practice.

Efficiency in Naïve Bayes classifier

- **Training Efficiency: Training involves counting word frequencies and estimating conditional probabilities. This makes training extremely fast and memory-efficient, even for large datasets.**
- **Prediction Efficiency: Classification requires multiplying a small number of probabilities, making predictions almost instantaneous.**
- **Resource Usage: Minimal storage is required since only probability tables for each word-class combination are needed.**

In this study, Naïve Bayes achieved an accuracy of 79.34%, demonstrating that its lightweight nature does not come at the expense of performance. This efficiency makes it particularly suited for real-time spam filtering on resource-constrained devices such as mobile phones.

Efficiency in Centroid Cosine classifier

- **Training Efficiency: Training consists of building centroid vectors for spam and ham classes, which involves averaging term frequencies. This is faster than iterative optimization methods but slightly slower than Naïve Bayes due to normalization steps.**
- **Prediction Efficiency: Classification requires computing the cosine similarity between an input vector and the centroids. This can be computationally more expensive than Naïve Bayes, especially with a large vocabulary, as vector dot products must be computed for every prediction.**
- **Resource Usage: Storage requirements are moderate since full centroid vectors must be retained. Memory usage increases with vocabulary size, but not as drastically as in more complex machine learning models.**

In the experiments, Centroid Cosine achieved an accuracy of 72.76% and slightly higher recall, showing that while it is efficient in capturing spam, its higher false positive rate reduces practical efficiency.

4.5 Conclusion

The main objective of this project was to study and implement a spam filtering using two mathematical methods, namely Naïve Bayes and Centroid Cosine, and to compare their accuracy, efficiency and overall performance. It also aimed to observe how these two approaches work and how well they perform in practice.

The experimental analysis provided useful insights into the strengths and weaknesses of both methods. Based on the findings obtained, it can be concluded that Naïve Bayes provides a more balanced and reliable performance across different evaluation metrics. This makes it more suitable for real-world applications where both accuracy and efficiency are equally important. However, the Centroid Cosine method also has its own advantage, as it can be useful in scenarios where recall is more important than precision, meaning cases where detecting the maximum amount of spam is the priority, even if some normal messages are wrongly flagged.

Overall, this project successfully achieved its objectives by implementing both the methods and providing a clear comparison of their capabilities. The findings highlight that there is no single "perfect" method, but the choice depends on the specific requirements of the application. In general, Naïve Bayes stands out as a more balanced and practical option for spam filtering tasks.

REFERENCES

[1] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., and Spyropoulos, C. D. (2000). "An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages.", In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval (pp. 160–167). ACM.

[2] Schneider, K.-M. (2003). "A comparison of event models for Naive Bayes anti-spam e-mail filtering." In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (pp. 307–314). ACL Anthology.

[3] Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). "A Bayesian approach to filtering junk e-mail." AAAI Workshop on Learning for Text Categorization.

[4] Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., Paliouras, G., and Spyropoulos, C. D. (2000). "An evaluation of Naive Bayesian anti-spam filtering." Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning, 9–17.

[5] Metsis, V., Androutsopoulos, I., and Paliouras, G. (2006). "Spam filtering with Naive Bayes— Which Naive Bayes?" CEAS 2006: Third Conference on Email and Anti-Spam.

[6] Forman, G. (2003). "An extensive empirical study of feature selection metrics for text classification." Journal of Machine Learning Research, 3(Mar), 1289–1305.

[7] Rennie, J. D., Shih, L., Teevan, J., and Karger, D. R. (2003). "Tackling the poor assumptions of Naive Bayes text classifiers." ICML, 3, 616–623.

[8] Katakis, I., Tsoumakas, G., and Vlahavas, I. (2006). "Dynamic feature space and incremental feature selection for the classification

of textual data streams." ECML PKDD Workshop on Knowledge Discovery from Data Streams.

[9] Delany, S. J., Buckley, M., and Greene, D. (2012). "SMS spam filtering: Methods and data." Expert Systems with Applications, 39(10), 9899–9908.

[10] Han, E. H., Karypis, G., and Kumar, V. (2001). "Text categorization using weight adjusted knearest neighbor classification. " Pacific-Asia Conference on Knowledge Discovery and Data Mining, 53–65.

[11] Salton, G., and Buckley, C. (1988). "Term-weighting approaches in automatic text retrieval." Information Processing and Management, 24(5), 513–523.

[12] Yang, Y., and Liu, X. (1999). "A re-examination of text categorization methods." Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 42–49.

[13] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). "Indexing by latent semantic analysis." Journal of the American Society for Information Science, 41(6), 391–407.

[14] Robertson, S. E., and Walker, S. (1994). "Some simple effective approximations to the 2Poisson model for probabilistic weighted retrieval." SIGIR '94: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 232–241.

[15] Zhong, S. (2005). "Efficient online spherical k-means clustering." Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, 5, 3180–3185.

[16] Salton, G. (1971). "The SMART Retrieval System: Experiments in Automatic Document Processing." Prentice Hall.

[17] Manning, C. D., Raghavan, P., and Schütze, H. (2008). "Introduction to Information Retrieval." Cambridge University Press.

[18] Han, E. H., and Karypis, G. (2000). "Centroid-based document classification: Analysis and experimental results." Principles of Data Mining and Knowledge Discovery (PKDD).

[19] Zhang, Y., Jin, R., and Zhou, Z. H. (2010). "Understanding bag-of-words model: a statistical framework." International Journal of Machine Learning and Cybernetics.

[20] Salton, G., Wong, A., and Yang, C. S. (1975). "A vector space model for automatic indexing." Communications of the ACM, 18(11), 613–620.

[21] Joachims, T. (1998). "Text categorization with Support Vector Machines: Learning with many relevant features." ECML.

[22] Yang, Y., and Pedersen, J. O. (1997). "A comparative study on feature selection in text categorization." ICML.

[23] Sebastiani, F. (2002). "Machine learning in automated text categorization." ACM Computing Surveys, 34(1), 1–47.

[24] Han, E. H., and Karypis, G. (2000). "Centroid-based document classification: Analysis and experimental results." PKDD.

[25] Tang, B., Luo, H., and Chen, J. (2005). "Dimensionality reduction using class-feature-centroid." ICMLC.

[26] Delany, S. J., Cunningham, P., and Smyth, B. (2005). "Spam filtering: Comparison of a memory-based approach to a centroid-based approach." Artificial Intelligence Review, 24(1), 73–92.

ST. TERESA'S COLLEGE (AUTONOMOUS)
DEPARTMENT OF MATHEMATICS AND STATISTICS