Naïve Bayes: Capstone Project Report

Student Name: Zafir Shaikh

Course: Machine Learning AI

Institution: Humber IGS

Course Code: BINF\_5507

Date: July 21, 2025

Github:- https://github.com/Money2107/Capstone-Final-Project/blob/main/Naive\_Bayes\_Capstone\_Report.ipynb

## 1. Introduction

Naïve Bayes is a family of simple yet powerful probabilistic classifiers based on applying Bayes’ Theorem with the “naïve” assumption of feature independence. It is widely used in text classification tasks such as spam filtering, sentiment analysis, and document categorization due to its speed, simplicity, and surprisingly strong performance.

Despite its simplicity, Naïve Bayes often performs competitively with more complex algorithms, especially when the assumption of independence holds approximately true. It is especially effective when dealing with large feature spaces and small datasets.

---

## 2. How It Works

Naïve Bayes applies Bayes’ Theorem as follows:

P(C|X) = [P(X|C) \* P(C)] / P(X)

Where:

- \( P(C|X) \): Posterior probability of class \( C \) given features \( X \)

- \( P(X|C) \): Likelihood of features given class \( C \)

- \( P(C) \): Prior probability of class \( C \)

- \( P(X) \): Probability of the data (normalization factor)

\*\*Key Assumption:\*\* All features in \( X = \{x\_1, x\_2, ..., x\_n\} \) are conditionally independent given the class \( C \), i.e.,

\[P(X|C) = \prod\_{i=1}^n P(x\_i|C)\]

This greatly simplifies computation and allows the model to be trained quickly even on high-dimensional data.

## 3. Use Cases & Performance

Applications:

- Spam detection: Email classifiers can flag messages based on word probabilities.

- Sentiment analysis: Used in NLP to classify text as positive, negative, or neutral.

- Document classification: Topic modeling, news grouping, etc.

Strengths:

- Fast training and prediction

- Works well with high-dimensional data

- Performs surprisingly well in many real-world tasks

Limitations:

- Assumes features are independent (which may not be realistic)

- Can perform poorly when feature dependence is strong

- Struggles with continuous features unless adapted (e.g., Gaussian Naïve Bayes)

## 4. Code Implementation Overview

Though no code is shown here, Naïve Bayes is easy to implement using libraries like `scikit-learn`. Variants include:

- Multinomial Naïve Bayes: Used for count-based features (e.g., word counts)

- Bernoulli Naïve Bayes: Used for binary/boolean features

- Gaussian Naïve Bayes: For continuous features assumed to follow a Gaussian distribution

Common Steps:

1. Convert text to numeric format (e.g., using `CountVectorizer` or `TF-IDF`)

2. Fit Naïve Bayes classifier to training data

3. Predict on new data and evaluate accuracy

## 5. Comparison / Visualization

| Feature | Naïve Bayes | Logistic Regression |

|------------------------------|----------------------------|-----------------------------|

| Type | Probabilistic | Discriminative |

| Assumes Feature Independence | Yes | No |

| Training Speed | Very Fast | Moderate |

| Handles Large Feature Space | Yes | Yes |

| Handles Correlated Features | Poorly | Well |

> Naïve Bayes may underperform when features are highly correlated but excels in sparse or text-based domains.

Image Placeholder: Word frequency heatmap or class probability visualization

## 6. References

1. Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach. Pearson.

2. scikit-learn Documentation – [https://scikit-learn.org](https://scikit-learn.org)

3. Wikipedia: Naïve Bayes Classifier – [https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier](https://en.wikipedia.org/wiki/Naive\_Bayes\_classifier)