Group 3 - Naive Bayes Classification Project

COMP 3009 - Applied Math for Data Science & AI

Jordon Abrams, Jonny Tahai, Sam Kippur

November 17th, 2025

1. Introduction

This project explores the Naive Bayes classification algorithm using a small weather dataset. The goal is to predict whether it will rain based on four features: Outlook, Temperature, Humidity, and Windy. By building the model manually with Python's built-in tools, we were able to see how each mathematical idea, particularly conditional probability and Bayes' theorem, works step by step in practice. The result is a simple but complete demonstration of how math and computation come together to form a working model.

2. Connecting the Project to the Course Objectives

This project connects mathematical concepts directly to algorithmic implementation. Bayes' theorem serves as the foundation, showing how theoretical probability translates into code that can make real predictions.

The project uses Bayes' theorem, written as: $P(Y|X) = [P(X|Y) \times P(Y)] / P(X)$ This relationship guided the entire implementation. By calculating priors and likelihoods, the program determines how probable it is that it will rain given observed weather conditions.

Probability is central to the model. Linear algebra principles appear in the way we represent categorical data as discrete, countable values. While Naive Bayes does not rely on gradient optimization like some machine learning models, the Laplace smoothing parameter (alpha = 1) functions similarly to a regularization term, preventing zero probabilities and ensuring numerical stability. The project also highlights how a single assumption of conditional independence can be powerful enough to produce meaningful predictions.

Finally, this work demonstrates how fundamental math ideas still drive widely used AI and machine learning systems. Naive Bayes remains an important model in areas such as spam detection, document classification, and medical diagnosis. By understanding its math, we better understand how larger, more complex algorithms are built on these same principles.

3. Mathematical Foundation

Bayes' theorem provides the rule for updating the probability of a hypothesis based on new evidence. In this project, the hypothesis is whether it will rain (Yes or No), and the evidence is a set of observed weather features.

$$P(Y|X) = [P(X|Y) \times P(Y)] / P(X)$$

Here, $P(Y)$ represents the prior probability of rain, $P(X|Y)$ represents the likelihood of observing certain conditions given rain, and $P(Y|X)$ is the posterior probability that it will rain given those conditions. Since $P(X)$ is constant across classes, the model focuses on maximizing $P(Y) \times \Pi P(xi \mid Y)$.

The Naive assumption states that all features are conditionally independent given the class: $P(x1, x2, ..., xn \mid Y) = \Pi P(xi \mid Y)$ Although this is rarely true in reality, since features like temperature and

humidity are often correlated, it allows the model to compute probabilities efficiently. This simplification is what makes Naive Bayes both mathematically straightforward and computationally fast.

4. Implementation Summary

The dataset contains 14 daily weather observations, each with four categorical features. Each record was stored as a small dictionary in Python, which made it easy to count feature occurrences for each class (Yes or No). We first calculated the prior probabilities: P(Yes) = 9/14 ≈ 0.6429 and P(No) = 5/14 ≈ 0.3571. After applying Laplace smoothing (alpha = 1), these became 0.625 and 0.375.

Conditional probabilities for each feature were then calculated using the formula: P(xi | Y) = (count(xi, Y) + alpha) / (count(Y) + alpha × k) where k is the number of possible values for that feature. For example, P(Outlook = Sunny | Yes) = 0.25 and P(Humidity = High | No) = 0.7143.

For a new day with the conditions: Sunny, Mild, High Humidity, and Windy = False, the model produced: P(Yes | X) = 0.4117 and P(No | X) = 0.5883, predicting No Rain.

To test model accuracy, we used Leave-One-Out Cross-Validation. The hand-coded version achieved 50% accuracy, while a version built using sklearn's MultinomialNB classifier (with the same smoothing) reached 64%. The difference is mainly due to how boolean features were encoded, but both approaches behave consistently with the underlying theory.

5. Results and Discussion

The model's computed priors and likelihoods reflected the small dataset's overall distribution. Rainy days occurred more frequently, leading to a higher prior probability for the "Yes" class.

Empirical Priors: P(Yes) = 0.6429, P(No) = 0.3571 Smoothed Priors: P(Yes) = 0.6250, P(No) = 0.3750 Example Likelihoods: P(Outlook = Sunny | Yes) = 0.25, P(Outlook = Sunny | No) = 0.50 P(Humidity = High | Yes) = 0.3636, P(Humidity = High | No) = 0.7143 Posterior Prediction: (Sunny, Mild, High, False) → Predicted: No Posterior Probabilities: {Yes: 0.4117, No: 0.5883}

While the overall accuracy is limited by the small sample size, the model behaves as expected mathematically. Each probability calculation contributes transparently to the final prediction, making the reasoning process easy to follow. The sklearn comparison helped verify the correctness of the logic and confirm that both implementations are built on the same foundation.

6. Independence Assumption

Naive Bayes assumes that features are conditionally independent once the class is known: P(x1, x2, ..., xn | Y) = Π P(xi | Y) In reality, this assumption is rarely perfect. For instance, Outlook and Humidity often vary together. However, even with this simplification, the algorithm still performs well on small datasets. It remains one of the clearest examples of how a mathematical assumption can simplify complex problems without losing interpretability.

7. Conclusion

Building a Naive Bayes model from scratch provided a clear understanding of how mathematics drives computation. Each step, from computing priors and conditional probabilities to evaluating predictions, showed how theoretical ideas become practical tools for data analysis. Although simple, the algorithm highlights the power of probability-based reasoning in machine learning. It shows that even basic mathematical structures can yield models that are efficient, interpretable, and still relevant in modern applications.