Naive Bayes and the Challenge of Spam Detection: A Statistical Analysis

17th December

Author: Srayoshi Bashed Mirza

Project Overview

Project Name	Naive Bayes and the Challenge of Spam Detection: A Statistical Analysis
Objective	Classify emails as **Spam** or **Not Spam** using: ● Text features (specific words or phrases). ● Probabilistic methods like Naive Bayes.
Concepts Covered	Joint Probability, Conditional Probability, Prior Probability, Posterior Probability, Naive Bayes Algorithm.
Tools	Microsoft Excel
Dataset	Spam email Dataset
Calculation Performed	**Prior Probabilities**: P(A), P(B) **Joint Probabilities:** ● P(A and B) = P(A)P(B) if two events are independent. ● P(A and B) = P(A)P(B/A) if two events are dependent **Conditional Probability**: P(A|B) = (P(A and B))/(P(B)) **Posterior Probability:** P(A|B) = (P(B|A)*P(A))/(P(B)) **Naive Bayes Algorithm:** P(C|X) = (P(X|C)*P(C))/(P(X))
Algorithm Used	Naive Bayes Algorithm
Expected Deliverables	- Probability calculations in Excel - Final email classification (Spam/Not Spam). -Drawbacks of Naïve bayes
Key Formulas	**Prior Probabilities**: P(A), P(B) **Joint Probabilities:** ● P(A and B) = P(A)P(B) if two events are independent. ● P(A and B) = P(A)P(B/A) if two events are dependent **Conditional Probability**: P(A|B) = (P(A and B))/(P(B)) **Posterior Probability:** P(A|B) = (P(B|A)*P(A))/(P(B)) **Naive Bayes Algorithm:** P(C|X) = (P(X|C)*P(C))/(P(X))
Learning Outcome	Hands-on understanding of basic probability concepts and their application in classification problems and reflecting the challenges of naïve bayes theorem.
Duration	1-2 hours depending on familiarity with Excel formulas

Abstract

Introduction

Probability is a fundamental mathematical concept that quantifies the likelihood of an event occurring, expressed as a number between 0 and 1. A probability of 0 indicates impossibility, while a probability of 1 indicates certainty. The higher the probability, the more likely the event is to happen. According to MIT, probability serves as a mathematical framework for analyzing phenomena with uncertain outcomes, providing tools for understanding and quantifying randomness. This makes it an essential tool in fields such as science, engineering, and finance.

In classification, probability is a cornerstone, as it allows for the quantification of uncertainty and the making of informed decisions. By assigning probabilities to different class labels, models can express their confidence in a prediction. This enables nuanced decision-making, such as setting thresholds for classification or combining multiple models. Additionally, probability-based metrics like AUC-ROC (Area Under the Receiver Operating Characteristic Curve) provide a comprehensive evaluation of model performance. In essence, probability empowers classifiers to handle the inherent uncertainty in real-world data, leading to more accurate and reliable predictions.

Naive Bayes is a classic example of a probabilistic classifier that uses Bayes' theorem to calculate the probability of a data point belonging to a particular class based on its features. The "naive" assumption in Naive Bayes is that features are conditionally independent, meaning the presence of one feature does not influence the presence of another, given the class. While this assumption is often unrealistic, it simplifies the calculations and surprisingly works well in many real-world scenarios. By calculating probabilities for each class, Naive Bayes can classify data, such as labeling emails as spam or not, or categorizing news articles.

However, despite its efficiency and popularity, Naive Bayes has some notable drawbacks. A significant limitation is the strong independence assumption between features, which often does not hold true in many real-world scenarios, potentially leading to inaccurate predictions when features are correlated. Another challenge is its sensitivity to zero-frequency features, which can drastically affect probability calculations. Additionally, Naive Bayes struggles with continuous features and often requires discretization or transformation techniques to handle them. Despite these drawbacks, Naive Bayes remains a widely used algorithm in tasks like text classification, where the simplicity of the model and the independence assumption can be reasonable approximations.

Objectives

The objectives of this project or experiments are as bellow:

- To demonstrate the application of Naive Bayes in a simple, practical scenario using Excel.
- To calculate the posterior probabilities of new emails being spam or not based on keyword occurrences and prior probabilities.
- To recognize the key assumption in Naive Bayes: that features are conditionally independent given the class.
- To observe how this assumption may not always hold in real-world data, leading to limitations in prediction accuracy.
- To classify a set of emails as "spam" or "not spam" using Naive Bayes and compare the predicted classifications with the actual labels (if available).
- To evaluate the accuracy of the Naive Bayes classifier in this context.
- To explore the impact of zero-frequency features (i.e., features that do not appear in certain classes) and their effect on the model's performance and predictions.
- To analyze whether Naive Bayes is an effective model for spam email classification based on the current dataset.
- To critically assess why Naive Bayes may or may not work well in this context, considering its assumptions and the characteristics of the dataset.

About the Dataset

This project utilizes two distinct datasets for training and testing the Naive Bayes spam classification model. Both datasets focus on spam email detection but serve different purposes in the classification workflow.

Training Dataset: Spam Email Dataset

Description:

The training dataset contains a collection of email text messages labeled as either spam or not spam. Each email is represented with two main components:

- **Text**: The content of the email, which includes the body of the message and, in some cases, subject lines or headers.
- **Label (spam_or_not)**: A binary indicator that classifies each email as either:
  - **1**: Spam (unwanted or junk email)
  - **0**: Not Spam (legitimate email)

Dataset Purpose:

This dataset is used to train the Naïve Bayes model. By analyzing the occurrence of keywords like *"*offer*"* and *"*win*"* in both spam and non-spam emails, the model calculates prior and conditional probabilities. These probabilities are critical for making predictions on new emails.

Key Characteristics:

- The dataset is clean and formatted, allowing straightforward analysis.
- The binary labels make it suitable for supervised classification tasks.
- The text content provides a basis for exploring the presence or absence of specific keywords to derive conditional probabilities.

Testing Dataset: Email Spam Dataset

Description:
The testing dataset consists of a new collection of email messages, also labeled as spam or not spam. Similar to the training data, it contains:

- **Text**: The content of the email message.
- **Label (spam_or_not)**: Binary labels indicating:
  - **1**: Spam
  - **0**: Not Spam.

Dataset Purpose:

The testing dataset is used to evaluate the performance of the Naive Bayes model trained on the first dataset. By applying the calculated probabilities from the training data, the model predicts whether each email in the test dataset is spam or not. The predictions are then compared with the actual labels to assess the model's accuracy and limitations. Specifically, the file name “lingSpam.csv” is used here to test.

Key Characteristics:

- This dataset provides unseen data to simulate real-world conditions for model evaluation.
- The presence of both text and binary labels enables performance comparison using accuracy metrics.
- It helps highlight the strengths and weaknesses of the Naive Bayes algorithm when applied to a new set of emails.

The **training dataset** is used to build the Naive Bayes model by calculating prior probabilities and conditional probabilities of keywords like "offer" and "win". The **testing dataset** is then used to validate the model by classifying new emails and evaluating its performance. Together, these datasets enable a comprehensive understanding of Naive Bayes' application in spam detection.

Methodology

The methodology section outlines the step-by-step approach used to implement the Naive Bayes classification algorithm to detect spam emails. This project leverages Microsoft Excel to calculate probabilities, classify emails, and evaluate the performance of the model.

Dataset Selection:

Two datasets were chosen for this project:

- Training Dataset: Used to calculate probabilities for the Naive Bayes model. Spam Email Dataset was selected for this.
- Testing Dataset: Used to evaluate the model's ability to classify new emails as spam or not spam. "lingSpam.csv" among the Email Spam Dataset from Kaggle.

Both datasets contain two columns:

- **Text**: The content of the email.
- **Label (spam_or_not)**: Binary classification where 1 = Spam and 0 = Not Spam.

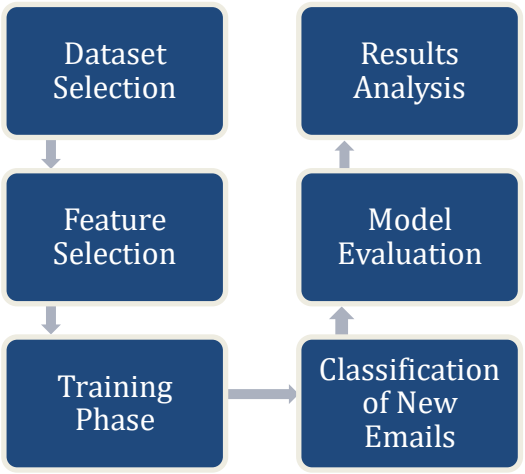

Figure 1: Methodology of the Spam Selection

Feature Selection

To simplify the project and focus on core probability concepts, two specific keywords were selected:

- "Offer"
- "Win"

These keywords commonly appear in spam emails and serve as features to evaluate their presence in spam and non-spam messages.

Training Phase:

The following probabilities were calculated from the **training dataset**:

Probability is essentially relative frequency—that is, it is the ratio of the number of times something happens to the number of possible outcomes. (By David Borman in Statistics 101).

a) Prior Probabilities

The prior probabilities were calculated to determine the likelihood of an email being **Spam** or **Not Spam**:

$$P(\text{Spam}) = \frac{Number\ of\ Spam\ emails}{\text{Total number of emails}}$$

$$P(\text{Not Spam}) = \frac{Number\ of\ Not\ Spam\ emails}{\text{Total number of emails}}$$

b) Conditional Probabilities

$$P(\text{Keyword} \mid \text{Spam}) = \frac{Number\ of\ Spam\ emails\ containing\ the\ Keyword}{\text{Total number of spam emails}}$$

$$P(\text{Keyword} \mid \text{Not Spam}) = \frac{Number\ of\ Not\ Spam\ emails\ containing\ the\ Keyword}{\text{Total number of not spam emails}}$$

Classification of New Emails

For each email in the **testing dataset**, the posterior probabilities were calculated using **Bayes' Theorem**:

Posterior Probability: A revised probability based on prior information

P (Spam | Keywords) = $P\ (Spam) * P(Offer \mid Spam) * P(Win \mid Spam)$

P (Not Spam | Keywords) = $P\ (Not\ Spam) * P(Offer \mid Not\ Spam) * P(Win \mid Not\ Spam)$

Model Evaluation

To assess the performance of the Naive Bayes model, predictions from the testing dataset were compared with the actual labels. A confusion matrix was created to evaluate key metrics, including:

- **True Positives (TP)**: Spam emails correctly labeled as spam.
- **True Negatives (TN)**: Non-spam emails correctly labeled as not spam.
- **False Positives (FP)**: Non-spam emails incorrectly labeled as spam.
- **False Negatives (FN)**: Spam emails incorrectly labeled as not spam

Results Analysis

The results from the evaluation reveal significant limitations of the Naive Bayes algorithm in this specific experiment. While Naive Bayes is often praised for its simplicity and efficiency, its underlying assumptions and constraints can lead to suboptimal performance in real-world scenarios

Performance Evaluation

The performance of the Naive Bayes classifier was assessed using the calculated metrics: **True Positives (TP)**, **True Negatives (TN)**, **False Positives (FP)**, and **False Negatives (FN)**.

The confusion matrix provided insights into the model's predictions:

Actual \ Predicted	Spam	Not Spam
Spam	True Positives	False Negatives
Not Spam	False Positives	True Negatives

From the results, it was evident that:

- A large number of **Spam emails were misclassified as Not Spam (high FN)**.
- The model had difficulty accurately identifying emails as Spam, reflecting a significant limitation in its predictive capability.
- **True Negatives (TN)** were relatively high, indicating that the classifier performed better in identifying Not Spam emails.

Evaluation Metrics

- **Accuracy**: 0.83
  - The classifier correctly identified 83% of the emails in the dataset. However, accuracy alone does not provide a complete picture of performance, especially in datasets where class distribution is imbalanced or where one class (Spam) is more critical to detect.
- **Precision**: 0
  - Precision measures the proportion of predicted Spam emails that were actually Spam. A precision of 0 indicates that the classifier did not correctly identify any Spam emails, resulting in significant false positives.
- **Recall**: 0
  - Recall measures the proportion of actual Spam emails that were correctly identified. A recall of 0 implies that the classifier failed to detect any of the Spam emails in the dataset, resulting in a high false-negative rate.

Analysis of Naive Bayes Performance

1. **Impact of the "Naive" Independence Assumption**
   The Naive Bayes algorithm assumes that features (keywords "Offer" and "Win") are conditionally

independent given the class (Spam or Not Spam). However, in real-world email data, this assumption is rarely valid. Features in emails often have complex relationships, such as correlated keywords or contextual dependencies.
    - For example, the keyword "Offer" might appear in both Spam and Not Spam emails but carry different contextual meanings depending on other associated words.
    - By ignoring these relationships, the algorithm misclassifies emails that contain similar keywords in different contexts.
2. **Limitations of Feature Selection**
    - The experiment relied on only two keywords, "Offer" and "Win," for classification. While these words are indicative of promotional content, they are insufficient to capture the full spectrum of features that differentiate Spam from Not Spam emails.
    - Emails without these keywords are classified solely based on prior probabilities, leading to inaccurate classifications.
3. **Failure to Detect Spam (Recall = 0)**
   The classifier's inability to detect any Spam emails (Recall = 0) highlights a major drawback of the Naive Bayes algorithm in this case:
    - The keywords chosen were not sufficiently representative of Spam content.
    - As a result, many actual Spam emails were labeled as Not Spam, increasing the false-negative count.
4. **Precision of 0 and Misclassification**
   A precision of 0 indicates that none of the emails predicted as Spam were actually Spam. This suggests:
    - The model was unable to differentiate between Spam and Not Spam emails effectively.
    - The use of limited features and the independence assumption contributed to high false positives, further degrading performance.

Key Observations

Strong Bias Toward the Majority Class:
The classifier predominantly labeled emails as Not Spam, reflecting a bias toward the majority class. This occurs because the conditional probabilities for Spam emails were not significant enough to outweigh those for Not Spam emails, given the limited features.

Overestimation of Accuracy:
Despite an accuracy of 83%, the model's inability to detect Spam emails (as indicated by Precision = 0 and Recall = 0) demonstrates that accuracy alone is not an adequate performance metric in this context.

Need for More Robust Features:
The reliance on two keywords was insufficient for robust classification. Including additional features such as email length, punctuation patterns, sender information, or more representative keywords could improve model performance.

Conclusion

This project aimed to explore the application of the Naive Bayes algorithm in a simple, practical scenario using Excel. By calculating the posterior probabilities of new emails being spam or not based on keyword occurrences and prior probabilities, we demonstrated how Naive Bayes works under the assumption that features are conditionally independent given the class. However, this assumption, although simplifying the calculations, was found to be a limitation when applied to real-world data, where features often exhibit interdependence.
The experiment also highlighted the challenges of applying Naive Bayes for spam email classification. While the model was able to classify emails based on the presence of certain keywords, the accuracy of these classifications was limited due to the algorithm’s reliance on conditional independence between features. This assumption does not always hold in complex, real-world scenarios, which can lead to poor performance, as observed in the experiment where Naive Bayes failed to correctly identify spam emails.

Another important aspect of this experiment was the evaluation of the Naive Bayes classifier's effectiveness in a practical scenario. The experiment involved comparing predicted classifications with actual labels, which revealed the model's limitations. Despite achieving an accuracy of 83%, the precision and recall metrics were both 0, highlighting the model's inability to correctly identify spam emails. This pointed to the impact of zero-frequency features that do not appear in certain classes which can significantly affect Naive Bayes predictions.

The analysis of Naive Bayes in the context of spam email classification underscores the importance of understanding the assumptions made by the model. While Naive Bayes is efficient and interpretable, its performance can suffer when the underlying assumptions do not align with the nature of the data. In this case, relying on only two keywords to classify emails proved insufficient to capture the complexity of spam email patterns.

In conclusion, while Naive Bayes remains a popular algorithm for text classification tasks, its application in spam email detection requires careful consideration of the dataset's characteristics. This project critically assessed the strengths and weaknesses of Naive Bayes and highlighted the need for more advanced algorithms and feature engineering techniques to improve spam email classification accuracy. Future work should explore alternative models and evaluate the impact of incorporating additional features to create more reliable spam detection systems.