Marmara University - Faculty of Engineering

Department of Computer Engineering

Intro to Machine Learning (Fall 2024)

Submit Date: 03/11/2024.

Naive Bayes Classifier

Student Number (ID)	Name	Surname
150120998	Abdelrahman	Zahran

Sections Of the Report: -

1. Introduction

- **Naive Bayes Algorithm**:

The Naive Bayes algorithm is a fundamental probabilistic classification technique based on Bayes' Theorem. It makes a strong assumption that the features in the dataset are conditionally independent given the class label. This simplification allows for efficient computation, making it a widely used algorithm in various domains, especially for tasks like text classification (e.g., spam detection) and sentiment analysis. Despite its "naive" assumption of feature independence, the Naive Bayes algorithm often performs surprisingly well, even in cases where this assumption is violated. It calculates the probability of each class by combining the prior probability of the class with the likelihood of the observed features, and then selects the class with the highest posterior probability.

- **Objectives**:

The main objective of this assignment is to implement a Naive Bayes classifier from scratch, without using pre-built machine learning libraries like scikit-learn. We aim to test our implementation using the "Play Tennis" dataset, which consists of weather-related features and a target variable indicating whether a tennis game was played. Additionally, we will handle common challenges such as zero probabilities and evaluate the model’s performance using a rigorous validation technique.

2. Methodology

- **Data Preparation**:

    - **Dataset Overview**: We used the "Play Tennis" dataset, which includes weather-related features (Outlook, Temperature, Humidity, Wind) and a target variable (PlayTennis) indicating if a tennis game was played. The dataset is small, with 14 records, making it suitable for demonstrating the Naive Bayes algorithm.
    - **Data Loading**: The dataset was loaded from a CSV file using pandas, a powerful data manipulation library in Python. We performed basic checks to ensure data integrity, including confirming the presence of required columns and handling any missing values.
    - **Handling Categorical Data**: The Naive Bayes algorithm inherently works well with categorical features, so we did not need to perform additional encoding. The feature values were used as they appeared in the dataset, making the implementation straightforward.
- **Implementation Details**:
    - **Prior Probabilities**: The prior probabilities represent the likelihood of each class (Yes or No) in the absence of any feature information. We calculated these probabilities based on the frequency of each class in the dataset.
    - **Likelihood Calculation**: The likelihood of observing each feature value given a class was computed. To address the issue of zero probabilities (i.e., when a feature value has not been observed for a particular class), we applied Laplace smoothing. This ensures that all probabilities are non-zero, making the model more robust.
    - **Model Training**: We combined the calculated prior probabilities and likelihoods into a model, which we saved in a structured JSON format. This format facilitates easy retrieval and use during the prediction phase.
    - **Prediction Logic**: The classifier calculates the posterior probability for each class using the Bayes' Theorem formula. We used log probabilities to avoid underflow issues, a common problem when dealing with very small probability values.
- **Handling Challenges**:
    - **Zero Probabilities**: One of the primary challenges of the Naive Bayes algorithm is handling cases where a feature value has not been observed in the training data for a specific class, leading to a zero probability. We implemented Laplace smoothing to overcome this, which ensures that all probabilities are non-zero and the model remains reliable.
    - **Error Handling**: To make the implementation more robust, we added error handling mechanisms. For instance, we checked if the file paths were valid and if the dataset contained the expected columns. These checks help ensure that the program runs smoothly and can handle common issues gracefully.

---

3. Results

- **Performance Metrics**:
    - We evaluated the performance of the Naive Bayes classifier using Leave-One-Out Cross-Validation (LOOCV). This approach is rigorous and ensures that each instance in the dataset is used as a test case while the model is trained on the remaining data. This maximizes the use of the available data and provides a reliable estimate of the model's performance.
    - The overall accuracy of the model was **93%**, indicating that the Naive Bayes classifier was effective in identifying patterns in the "Play Tennis" dataset.

- **Tables/Figures**:
  - **Table 1: Performance Summary**:

Metric	*Value*
Accuracy	93%
Total Instances	14

  - **Figure 1: Confusion Matrix** (Optional): A confusion matrix could be used to visualize the misclassifications, providing insights into where the model struggled.
- **Calculation Logs**:
  - Detailed calculation steps were recorded in a log file (naive_bayes_calculations.txt). This file demonstrates how the model computed the probabilities for each instance, providing transparency and a better understanding of the decision-making process.

---

4. Discussion

- **Analysis of Results**:
  - The Naive Bayes classifier performed well on the "Play Tennis" dataset, achieving an accuracy of **93%**. This high accuracy suggests that the algorithm was able to capture the relationships between the weather features and the decision to play tennis.
  - However, a few misclassifications were observed. These misclassifications may be due to the small size of the dataset, which limits the model's ability to generalize, and the inherent assumption of feature independence made by the Naive Bayes algorithm.
- **Possible Reasons for Misclassifications**:
  - **Feature Dependence**: One of the key assumptions of Naive Bayes is that all features are independent given the class label. In reality, this assumption often does not hold. For example, the features "Humidity" and "Outlook" might be correlated in the context of weather conditions, affecting the model's predictions.
  - **Data Sparsity**: The "Play Tennis" dataset is small, with only 14 instances. This means that some feature values are underrepresented, making the model sensitive to noise and outliers. A larger dataset would likely provide a better estimate of the true probabilities and improve the model's performance.
- **Limitations**:
  - **Independence Assumption**: The Naive Bayes algorithm's assumption of feature independence can limit its effectiveness in cases where features are highly correlated. This limitation may affect the model's performance on more complex datasets.
  - **Handling Continuous Data**: Our implementation is designed for categorical data. To extend the classifier to handle continuous features, we would need to use a Gaussian Naive Bayes variant, which assumes a normal distribution for continuous features.
  - **Scalability**: The Leave-One-Out Cross-Validation approach, while thorough, is computationally expensive, especially for large datasets. In real-world scenarios with thousands or millions of records, more efficient validation techniques, such as k-fold cross-validation, would be preferable.

---

5. Conclusion

- **Summary**:
  - We successfully implemented a Naive Bayes classifier from scratch and tested it using the "Play Tennis" dataset. The model demonstrated reasonable performance, achieving an accuracy of **93%**, given the naive assumption of feature independence.
  - We addressed key challenges, such as zero probabilities, by implementing Laplace smoothing. We also ensured the implementation was robust through proper error handling and data validation checks.
- **Lessons Learned**:
  - Implementing the Naive Bayes algorithm from scratch provided a deeper understanding of its underlying principles. We learned the importance of handling zero probabilities using smoothing techniques and the benefits of using log probabilities to avoid numerical issues.
  - We also recognized the limitations of the algorithm, particularly its reliance on the independence assumption, and the need for more sophisticated models when dealing with complex or highly correlated data.

---

6. References

- **Textbooks**:
  - Alpaydin, E. (2010). *Introduction to Machine Learning*. MIT Press.
  - Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- **Documentation**:
  - Python documentation for libraries like NumPy and pandas:
    - NumPy Documentation
    - Pandas Documentation
- **Additional Resources**:
  - Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
  - Wikipedia contributors. "Naive Bayes classifier." *Wikipedia, The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/wiki/Naive_Bayes_classifier

---