Implementation of a Simple Naive Bayes Classifier in Python

15.11.2024

Necati Koçak - 150120053

Introduction

Naive Bayes is a foundational probabilistic machine learning algorithm that applies Bayes' Theorem to classification tasks. It is termed "naive" because it assumes that all features are conditionally independent, which simplifies computations. This project implements the Naive Bayes Classifier from scratch in Python and applies it to the "Play Tennis" dataset, a small toy dataset often used for demonstration purposes in machine learning.

The primary objectives of this project are to:

1. Understand the foundational concepts of the Naive Bayes algorithm.
2. Build a classifier from scratch without relying on pre-built machine learning libraries.
3. Evaluate the classifier's performance on the "Play Tennis" dataset.

Dataset

The **"Play Tennis"** dataset describes weather conditions along with the decision to play tennis. It consists of 14 examples with the following features:

1. **Outlook**: Sunny, Overcast, Rain
2. **Temperature**: Hot, Mild, Cool
3. **Humidity**: High, Normal
4. **Wind**: Weak, Strong
5. **PlayTennis (Target)**: Yes, No

Data Preparation:

- The dataset was manually loaded into a pandas DataFrame and saved as a JSON file for future usage.
- As all features are categorical, no additional transformations were needed.

Methodology

1. Model Training:

a. The Naive Bayes classifier computes:

i. **Prior Probabilities**: The proportion of each class in the dataset.

ii. **Conditional Probabilities**: The probability of each feature value given a class.

b. **Laplace Smoothing** was applied to handle zero-probability cases, ensuring robust predictions even for unseen combinations of features.

c. The trained model, including prior and conditional probabilities, was stored as a JSON file for reuse.

2. Prediction:

a. Using Bayes' Theorem, the classifier computes the likelihood of each class given a new instance's feature values. The predicted class is the one with the highest posterior probability.

$$P(C|X) = \frac{P(C) \cdot P(X|C)}{P(X)}$$

b. During predictions, the model parameters were loaded from the JSON file.

3. Logging and Evaluation:

a. Every classification operation was logged into a **classification_log.txt** file. This included the instance's feature values, predicted class, and actual class.

b. Model accuracy was calculated as the proportion of correctly classified instances:

$$Accuracy = \frac{\text{Correct Predictions}}{\text{Total Instances}}$$

Results

1. Prediction Example:

For a test instance:

Input: `{'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Weak'}`

- **Prediction**: `Yes`

2. Overall Accuracy:

- On the dataset, the model achieved an accuracy of **92.86%**.

3. Misclassifications:

- Misclassifications were observed in a few instances due to the small dataset size and the independence assumption.

Conclusion

This project successfully demonstrated the implementation of a Naive Bayes classifier using the "Play Tennis" dataset. The model achieved a reasonable accuracy of 92.86%, showcasing the effectiveness of Naive Bayes in handling small

datasets. This exercise also highlighted the simplicity and practicality of probabilistic classifiers.

References

1. Alpaydın, E. *Introduction to Machine Learning*.
2. Python Documentation: Pandas and JSON libraries.
3. Assignment Requirements Document.