

The Naive Bayes algorithm

prepared by Hakkı Kokur / 150120033

1. Introduction

The Naive Bayes algorithm is a simple but powerful machine learning technique used for classification tasks. It's based on Bayes' theorem, which relates the conditional probabilities of two events.

The key idea behind Naive Bayes is that it assumes the features (or variables) in a dataset are independent of each other, given the class label. This "naive" assumption simplifies the calculations and makes the algorithm very efficient, even with large datasets.

With basic steps:

1. We start with some training data, which has examples labeled with their true class.
2. For each class, we calculate the probability of each feature occurring, based on the training data.
3. When we get a new, unlabeled example, we use Bayes' theorem to calculate the probability of that example belonging to each possible class.
4. We then assign the example to the class with the highest probability.

2. Methodology

The core of the Naive Bayes algorithm is calculating the probability of each class given the feature values. Since I'm working with a binary classification problem (play tennis or not), I need to calculate the probability of the "Yes" and "No" classes.

To do this, I first calculated the prior probabilities for each class by counting the number of instances in each class and dividing by the total number of instances. This gave me the probability of each class occurring without considering the feature values

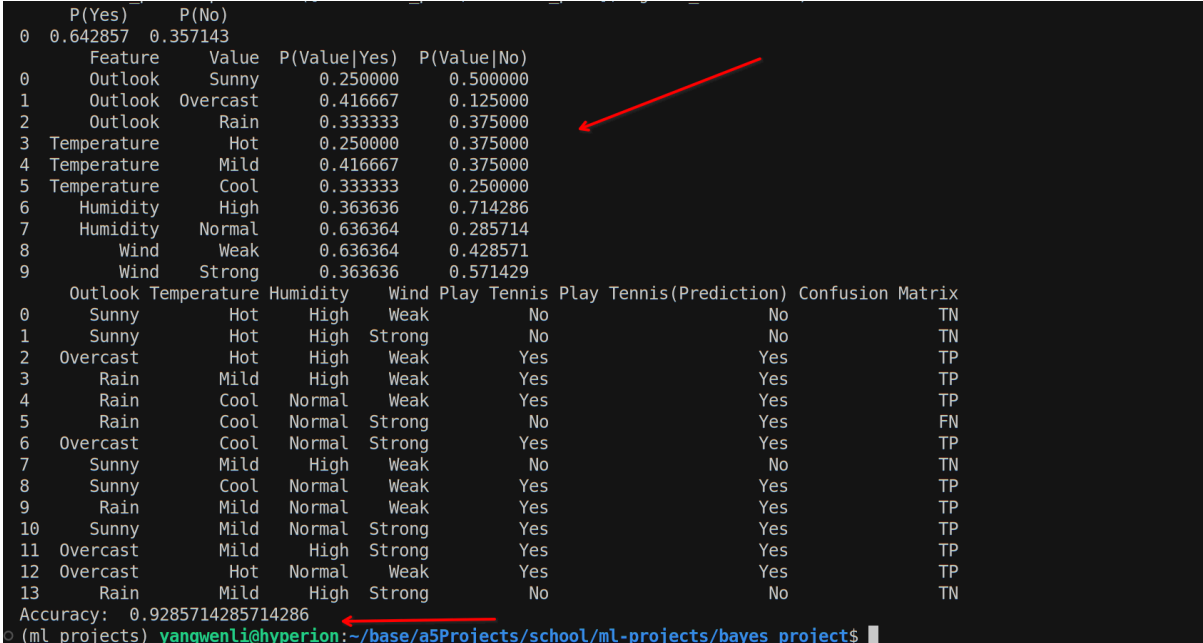
Next, I calculated the likelihood probabilities. For each feature and each class, I counted the number of times that feature value appeared when that class was the outcome. I then divided this count by the total number of instances of that class. This gave me the probability of each feature value occurring given the class.

However, I ran into a challenge here - some feature-class combinations had a count of 0, which would result in a likelihood probability of 0. This is problematic, as it would make the overall probability 0 no matter what the other feature probabilities were. To address this, I used Laplace smoothing, which adds a small constant (1) to all the counts before dividing. This ensures that no probability is ever exactly 0.

With the prior probabilities and likelihood probabilities calculated, I was able to use Bayes' theorem to calculate the posterior probability of each class given the feature values for a new instance. I simply multiplied the prior probability by the product of the likelihood probabilities for each feature, and the class with the highest posterior probability was the predicted output.

I implemented this process in a Python function that takes in a DataFrame of data and returns the predicted class labels for each instance.

3. Results



```

P(Yes)    P(No)
0 0.642857 0.357143

Feature  Value  P(Value|Yes)  P(Value|No)
0 Outlook  Sunny    0.250000      0.500000
1 Outlook  Overcast  0.416667      0.125000
2 Outlook  Rain      0.333333      0.375000
3 Temperature  Hot      0.250000      0.375000
4 Temperature  Mild      0.416667      0.375000
5 Temperature  Cool      0.333333      0.250000
6 Humidity    High      0.363636      0.714286
7 Humidity    Normal    0.636364      0.285714
8 Wind        Weak      0.636364      0.428571
9 Wind        Strong    0.363636      0.571429

Outlook Temperature Humidity Wind Play Tennis Play Tennis(Prediction) Confusion Matrix
0 Sunny          Hot      High  Weak      No           No           TN
1 Sunny          Hot      High  Strong    No           No           TN
2 Overcast       Hot      High  Weak      Yes          Yes          TP
3 Rain           Mild     High  Weak      Yes          Yes          TP
4 Rain           Cool     Normal Weak      Yes          Yes          TP
5 Rain           Cool     Normal Strong    No           Yes          FN
6 Overcast       Cool     Normal Strong    Yes          Yes          TP
7 Sunny          Mild     High  Weak      No           No           TN
8 Sunny          Cool     Normal Weak      Yes          Yes          TP
9 Rain           Mild     Normal Weak      Yes          Yes          TP
10 Sunny         Mild     Normal Strong    Yes          Yes          TP
11 Overcast      Mild     High  Strong    Yes          Yes          TP
12 Overcast      Hot      Normal Weak      Yes          Yes          TP
13 Rain          Mild     High  Strong    No           No           TN
Accuracy: 0.9285714285714286
  
```

Figure [1]: Result of Model According

Results show us the probability of the prior class and shows the probability of each likelihood. After the probability calculation, it uses the same dataset to make predictions. According to that we obtained the confusion matrix and accuracy of our model.

4. Discussion

Looking at the results, the Naive Bayes classifier performed quite well, achieving an accuracy of about 93%. This is a promising result, especially for a relatively simple algorithm like Naive Bayes.

One interesting aspect of the results is the prior probabilities. **The probability of "Play Tennis" is around 64%, while the probability of "Don't Play" is 36%.** This suggests that tennis is played more often than not in the given scenarios, which makes sense given the nature of the dataset.

When examining the likelihood probabilities, we can see some interesting patterns. For example, the likelihood of "Sunny" weather when tennis is played is much lower than when it

is not played. Conversely, "Overcast" and "Rain" conditions seem to be more favorable for playing tennis. This aligns with our intuition that people are more likely to play tennis on cloudy or rainy days rather than hot, sunny days.

The confusion matrix also provides some useful insights. Most of the instances were classified correctly, with only one false negative (where the model predicted "Don't Play" but the actual outcome was "Play"). This suggests that the model had some difficulty identifying the cases where tennis was played despite unfavorable conditions (like rain and strong wind).

One potential reason for this misclassification could be the assumption of feature independence in the Naive Bayes algorithm. In reality, the weather features are likely correlated to some degree - for example, rain is often accompanied by cooler temperatures and stronger winds. This violates the independence assumption and could lead to suboptimal probability estimates.

5. Conclusion

It has allowed me to understand the basic workings of the Naive Bayes algorithm, as well as some of its limitations and potential areas for improvement. I look forward to continuing to explore and develop my skills in machine learning and data analysis.

References

- *Introduction to Machine Learning* by Ethem Alpaydin
- *Pattern Recognition and Machine Learning* by Christopher Bishop
- <https://www.javatpoint.com/machine-learning-naive-bayes-classifier>
- <https://www.ibm.com/topics/naive-bayes>