Programming Assignment 1 - Simple Naive Bayes Implementation

Ardacan ÖZENER
150120026

Introduction

Overview of Naive Bayes Algorithm

The Naive Bayes algorithm is a probabilistic machine learning model based on Bayes' theorem. It assumes independence between features given the class label, simplifying the computation of posterior probabilities. Despite its simplicity, Naive Bayes is effective for classification tasks such as text categorization, spam detection, and sentiment analysis.

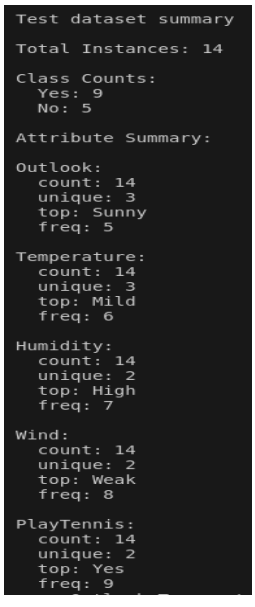
Objectives of the Assignment

- Implement a Naive Bayes classifier to classify the given dataset.
- Train the model using a JSON-formatted dataset.
- Evaluate its performance using appropriate metrics.
- Address challenges like zero probabilities using Laplace smoothing.

Methodology

Data Preparation

- The training and testing datasets are same and converted in JSON format.
- Each dataset was loaded and converted into a pandas DataFrame for easier manipulation.
- The target variable was `PlayTennis`, with categorical features such as `Outlook`, `Temperature`, `Humidity`, and `Wind`.
- `get_dataset` function prints dataset and its summary.



```
Test dataset summary

Total Instances: 14

Class Counts:
  Yes: 9
  No: 5

Attribute Summary:

Outlook:
  count: 14
  unique: 3
  top: Sunny
  freq: 5

Temperature:
  count: 14
  unique: 3
  top: Mild
  freq: 6

Humidity:
  count: 14
  unique: 2
  top: High
  freq: 7

Wind:
  count: 14
  unique: 2
  top: Weak
  freq: 8

PlayTennis:
  count: 14
  unique: 2
  top: Yes
  freq: 9
```

	Outlook	Temperature	Humidity	Wind	PlayTennis
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Implementation Details

1. Prior Probabilities
Calculated the prior probability for each class (`Yes` and `No`) based on the frequency in the training dataset.

2. Likelihoods
- Computed conditional probabilities for each feature value given the class label.
- Applied Laplace smoothing to address cases of zero probabilities.

3. Model Persistence
- Enabled saving of model's probability tables to a JSON file for reuse with the function `save_model`.
- Enabled loading of model's probability tables from a JSON file for reuse with the function `load_model`.

4. Testing and Evaluation
- Implemented a `predict` method to compute class probabilities for test instances.
- Constructed a confusion matrix and calculated metrics: Accuracy, Specificty, Precision, Recall, and F1-Score.

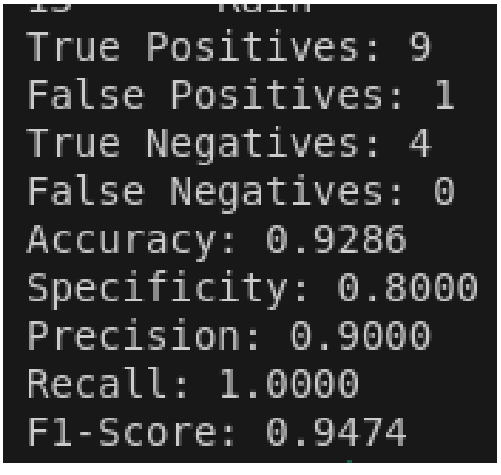
Challenges and Solutions

- **Zero Probabilities**: Addressed using Laplace smoothing to ensure numerical stability.
- **Handling Missing Features:** Skipped undefined features in test instances during probability calculation.

Results

Performance Metrics

The classifier was evaluated by using the training dataset.



Discussion

Analysis of Results

The results are not reliable because training and test datasets are same. Also, training dataset is small even training dataset and test dataset would be different, expected results would be low. All scores are close to 1, this indicates model works well.

- An accuracy of 0.9286 demonstrates the model's overall effectiveness in correctly classifying instances.
- The recall of 1.00 highlights that the model identified all positive cases without any false negatives.
- The precision of 0.90 reflects a small margin of false positives, indicating high reliability in its positive predictions.
- The specificity of 0.80 suggests that while the classifier performed well on negative cases, it was slightly less effective at avoiding false positives compared to its handling of positives.
- The F1-Score of 0.9474 balances precision and recall, confirming strong overall performance.

Reasons for Misclassifications

Model has false positive prediction on the sixth instance. Since training and test datasets are same, that instance is most probably an outlier.

Limitations

Target feature (`PlayTennis`) is hard coded on the script so this code will work only with datasets with same target feature. Other limitations like number of instances or number of the other features depend on the machine or Python environment.

Conclusion

Naive Bayes classifier implemented and evaluated successfully. Laplace smoothing applied. Test dataset was same with training dataset. However, accuracy was not 1. This indicates sometimes training dataset cannot be perfect and consistent.