Machine Intelligence Lab (UE21CS352A)

Naive Bayes Classifier

Dr. Pooja Agarwal, Dr. Alpha Vijayan

Email: poojaagarwal@pes.edu

alphavijayan@pes.edu

Teaching Assistants:

sanjanasrinidhi1810@gmail.com

karunakkc@gmail.com

Welcome to the Naive Bayes Text Classifier task! In this assignment, you'll implement a basic Naive Bayes Classifier to categorize text documents into two different classes - sports and elections. Before diving into the implementation, let's take a quick look at what a Naive Bayes classifier is.

Introduction to Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm commonly used for text classification, spam filtering, and sentiment analysis. It's based on Bayes' theorem, which calculates the probability of an event given the probabilities of related events. In text classification, Naive Bayes makes a simplifying "naive" assumption: it treats each feature (such as a word in a document) as conditionally independent, even though this might not hold true in reality.

Assignment Structure

- `naive_bayes_classifier_SRN.py` : This is the boilerplate code file where you will complete the missing parts of the NaiveBayesClassifier class implementation. **Replace SRN with your SRN. For example, if your SRN is PES2UG20CS549, rename this file as** **`naive_bayes_classifier_PES2UG20CS549.py`** .
- `test_naive_bayes.py` : This file conducts tests using the NaiveBayesClassifier class and compares its predictions with expected results.
  Change the import statement: '`from naive_bayes_classifier_SRN import`

`NaiveBayesClassifier`' with the **renamed boilerplate code file name**. Your task is to ensure the tests pass aftercompleting the implementation.

- `README.pdf` : This document provides an overview of the assignment, explains the purpose of each function, and gives instructions for completing the assignment.

Input data

The training data is given as two lists comprising strings:

- **sentences:** This list contains 30 sentences that describe various scenarios related to sports and elections.
- **categories:** This list corresponds to the categories of the sentences in the sentences list. Each element in the categories list indicates the overarching category to which the corresponding sentence belongs. The categories include "sports" and "elections".

The list of sentences has been converted into vectors using CountVectorizer(). This entire list represents the X vector in the provided code. The categories are provided as a separate list. Hence, there is no need for additional data encoding. Treat the vector X as a list. X is a numpy array.

Sample training input:

```
sentences = ["The fans cheered loudly as their team scored the winning goal.", "The ca
ndidates are neck and neck in the close election race.", "Despite a forgettable start,
the team managed to turn the game around."]
categories = ["sports", "elections", "sports"]
```

Instructions

1. Open `naive_bayes_classifier_SRN.py` and complete the missing sections in the NaiveBayesClassifier class:

- fit(X, y) : Train the Naive Bayes classifier with the provided training data.
- predict(X, class_probs, word_probs, classes) : Make predictions for the given test data.

  Replace SRN with your SRN in the file name. **Do not change any function definitions in this file**.

2. Run the `test_naive_bayes.py` script to test your classifier implementation against predefined test cases.

Ensure all tests pass successfully.

Change the file name of the boilerplate code in the import statement of this file. For example, if you renamed your boilerplate code file to `naive_bayes_classifier_PES2UG20CS549.py` , change the import statement to:

```
from naive_bayes_classifier_PES2UG20CS549 import NaiveBayesClassifier
```

Do not change anything else in the `test_naive_bayes.py` .

Command to run

```
python test_naive_bayes.py
```

Note: 1. If you are using MAC OS or Linux based system, replace python with python3.

2. Details regarding the function arguments and return values are provided in the boilerplate code.

Test Cases

`test_naive_bayes.py` contains a series of test cases, each consisting of a test sentence and the correct category it should be classified into. Your goal is to ensure your NaiveBayesClassifier implementation correctly predicts these categories for each test sentence. The test cases are auto-evaluated when you run test_naive_bayes.py, Your program will be evaluated against a set of hidden test cases as well.

Evaluation

Your assignment will be evaluated based on:

- Accurate implementation of NaiveBayesClassifier class methods.
- Successful passing of all test cases, including hidden test cases.