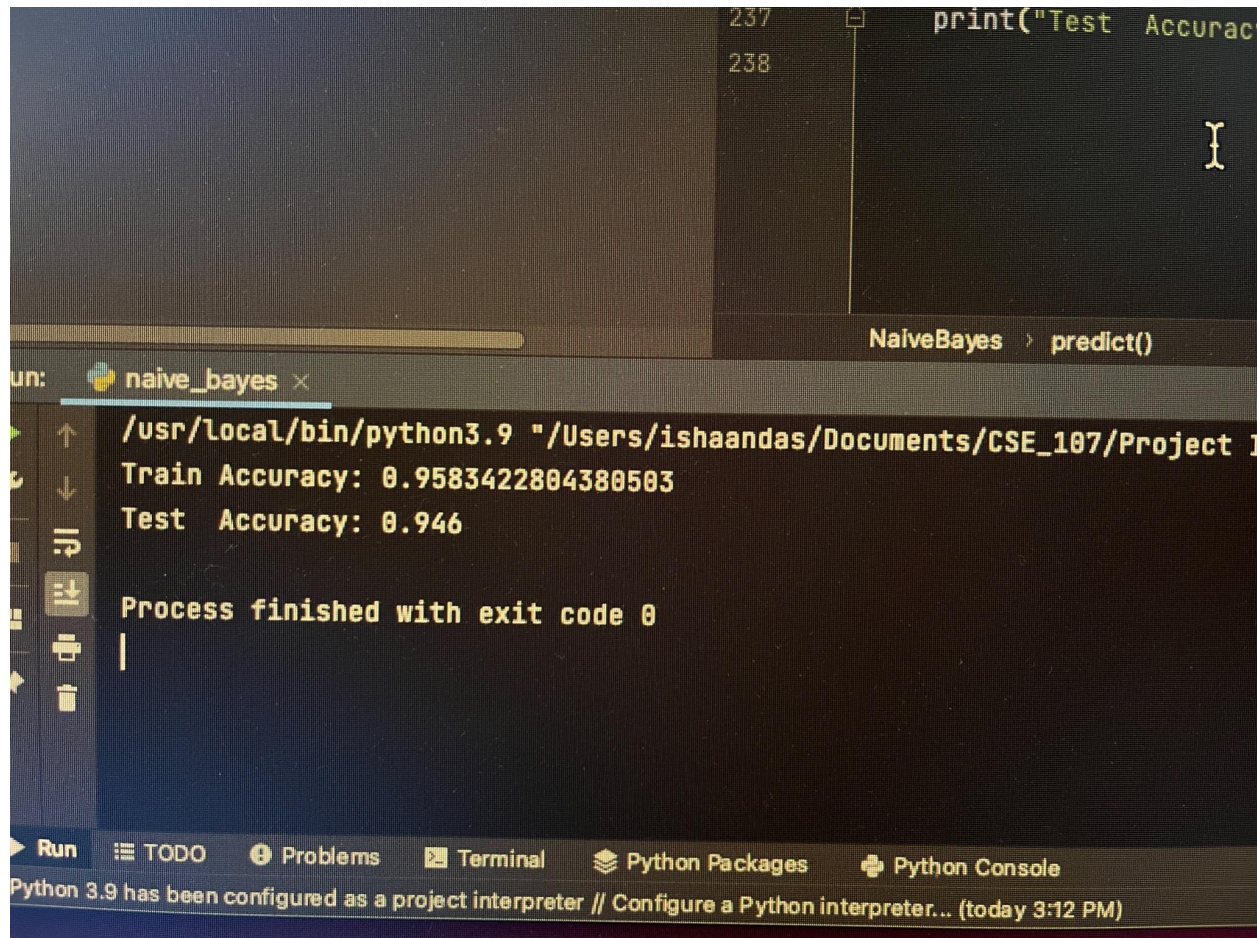


**Name:** Ishaan Das

**Email:** [isdas@ucsc.edu](mailto:isdas@ucsc.edu)

**Instructions:** To compile and run my program you can navigate, using terminal, to the folder where the naive\_bayes.py file is located, and then input the command “python3 naive\_bayes.py” into your terminal to run the program. The output will be the Train Accuracy and the Test Accuracy.

**Expected Output:**



```
237 print("Test Accuracy")
238
NaiveBayes > predict()

run: naive_bayes x
/usr/local/bin/python3.9 "/Users/ishaandas/Documents/CSE_107/Project 1
Train Accuracy: 0.9583422804380503
Test Accuracy: 0.946

Process finished with exit code 0
|

Run  TODO  Problems  Terminal  Python Packages  Python Console
Python 3.9 has been configured as a project interpreter // Configure a Python interpreter... (today 3:12 PM)
```

### Understanding:

- This program works because I first populate the dictionaries for the number of spam emails that contain a found word and the number of ham emails that contain a found word.
- With these dictionaries created from the training data, I am able to calculate the probability of words in an inputted email appearing in a spam email (ProbWordsGivenSpam) and the probability of words in an inputted email appearing in a ham email (ProbWordsGivenHam).
  - To do this I iterate through each word in the email and can then find the number of spam or ham emails that the word appears in using the dictionaries, and then I divide this number by the total number of spam training emails or ham training emails
  - For this calculation though, I need to add 1 to the numerator, which is the number of spam or ham emails that the specific word appears in, and add 2 to the denominator, which is the total number of training ham or spam emails. This is the Laplace smoothing which is used to avoid zeros in any of these quotients
  - I also need to calculate the log of this quotient, as a way to prevent underflow
  - I can then add this calculation to a running total of either ProbWordsGivenSpam or ProbWordsGivenHam
- I have also calculated the probability of a spam email and probability of a ham email by dividing the number of spam training emails by the total training emails (ProbSpam) and dividing the number of ham training emails by the total training emails (ProbHam)
  - I calculate the log of each of these quotients to avoid underflow
- Finally, I can return that the email is spam if the  $\text{ProbSpam} + \text{ProbWordsGivenSpam} > \text{ProbHam} + \text{ProbWordsGivenHam}$ , else return that the email is ham. This calculation is checking the probability of the email being a spam email given the words that the email contains vs. the probability of the email being a ham email given the words that the email contains. If the probability of it being spam is greater than the probability of it being ham, it'll say the email is spam, and default to the email being ham otherwise.

The accuracy for the training and test data are both around 95% indicating that this classifier, using this training data, is very accurate.