Report of Naive Bayes Assignment

Step 1

Console output:

```
      >>> print(df['label'].value_counts().iloc[0], df['label'].value_counts
      4818 745
    """

    doctest.testmod()
```

```
TestResults(failed=0, attempted=2)
```

Step 2

Console output:

```
      1113
      >>> np.isclose(y_train.value_counts(normalize=True)['spam'], 0.13393258426966292, atol=1e-3)
      np.True_
      >>> np.isclose(y_test.value_counts(normalize=True)['ham'], 0.8660674157303371, atol=1e-3)
      np.True_
    """

    doctest.testmod()
```

```
Training set size: 4450 messages
Testing set size: 1113 messages

Class distribution in training set:
label
ham     3854
spam     596
Name: count, dtype: int64
Spam ratio: 13.39%

Class distribution in testing set:
label
ham     964
spam    149
Name: count, dtype: int64
Spam ratio: 13.39%

Examples from the training set:
                                                text label
0  Me too baby! I promise to treat you well! I be...   ham
1  YOU HAVE WON! As a valued Vodafone customer ou...  spam
2                          When did dad get back.     ham
3  Daddy, shu shu is looking 4 u... U wan me 2 te...   ham
4                          Remember on that day..     ham
TestResults(failed=0, attempted=4)
```

Step 3

Console output:

```
    """
      >>> print(pipeline.steps)
      [('vectorizer', CountVectorizer(min_df=2, stop_words='english')), ('classifier', M
    """

    doctest.testmod()
```

```
TestResults(failed=0, attempted=1)
```

Step 4

Console output:

```
    """
      >>> np.isclose(accuracy, 0.98472597, atol=10e-3)
      np.True_
      >>> np.isclose(spam_precision, 0.95833333, atol=10e-3)
      np.True_
      >>> np.isclose(spam_recall, 0.92617450, atol=10e-3)
      np.True_
      >>> np.isclose(false_positive_rate, 0.00622407, atol=10e-3)
      np.True_
    """

    doctest.testmod()
```

```
Accuracy: 0.9847

Classification Report:
              precision    recall  f1-score   support

         ham       0.99      0.99      0.99       964
        spam       0.96      0.93      0.94       149

    accuracy                           0.98      1113
   macro avg       0.97      0.96      0.97      1113
weighted avg       0.98      0.98      0.98      1113


Confusion Matrix:
[[958   6]
 [ 11 138]]

Spam Detection Metrics:
Spam Precision: 0.9583 (higher is better)
Spam Recall: 0.9262 (higher is better)
False Positive Rate: 0.0062 (lower is better)
TestResults(failed=0, attempted=4)
```

```python
# Compare with baseline (always predict majority class)
majority_class = y_train.mode()[0]
baseline_accuracy = (y_test == majority_class).mean()
print(f"\nBaseline (majority class) accuracy: {baseline_accuracy:.4f}")
print(f"Our model improvement over baseline: {accuracy - baseline_accuracy:.4f}")
```

```
Top Spam-indicating words:
claim: 5.3464
prize: 4.9786
150p: 4.8245
tone: 4.5491
18: 4.4991
cs: 4.4731
500: 4.4731
guaranteed: 4.4190
100: 4.3320
uk: 4.3013
1000: 4.2695
landline: 4.2028
awarded: 4.2028
www: 4.1314
ringtone: 4.1314
150ppm: 4.1314
collection: 4.0136
5000: 3.9266
16: 3.9266
000: 3.9266


Top Ham-indicating words:
gt: -4.7366
lt: -4.7290
lor: -4.0698
da: -4.0473
later: -3.8466
wat: -3.6531
amp: -3.5212
ask: -3.4820
said: -3.4130
home: -3.3840
cos: -3.3541
doing: -3.3541
come: -3.2916
morning: -3.2588
really: -3.2588
lol: -3.2075
sure: -3.1898
ll: -3.1673
gud: -3.1535
nice: -3.1348

Baseline (majority class) accuracy: 0.8661
Our model improvement over baseline: 0.1186
```

Reflection:

Using your notebook results, answer the following questions:

1. The Naive Bayes algorithm makes a fundamental "naive" assumption that is known to be incorrect in real-world text data, yet the algorithm still performs well for our spam filter application.
    - What is this "naive" assumption?

- Provide an example from email text that contradicts this assumption.
- Why does Naive Bayes still perform well for spam detection despite this incorrect assumption?
- How might this assumption affect performance differently in other NLP tasks compared to spam detection?

My Ans: The naive assumption is the conditional independence between each features given the class label (y). A contradiction example might be: P(hotel|Spam) might not be independent from P(trip|Spam). Naive Bayes still performs well probably because the correlated spam words wouldn't hinder the classification of Spam/non-spam email, as they would increase the probability of the Spam email being classified as Spam. When the sequence of the sentence matters, but not simple classification, the performance would be affected by wrong models.

2. Run the "Examine most informative features" code section below to identify the most informative words for spam and ham classification.

- List the top 5 words most strongly associated with spam and explain why these words make intuitive sense for spam detection.
- List 2-3 words that are strongly associated with ham (legitimate emails) and explain their significance.
- Identify at least one word in the top results that surprised you, and explain why.
- If you were to improve the spam filter, would you manually add or remove any specific features (words) based on this analysis? Why or why not?

My Ans: According to the result, the top 5 words associated with spam are "claim", "prize", "150p", "tone", and "18". These words connect with the contents, such as "claim your prize", "18+", which are quite spammy. The top 3 words with legitimate emails can be: "later", "said", "ask". These

words are often used in our daily life or work. The words "gt", "lt" actually surprised me because these are not human-used words, but HTML-entity artifacts. These HTML artifacts can both appear in spam or legitimate emails. Therefore, to improve the spam filter, we can manually remove these non-human language words so that we can reduce noise.

3. When implementing a Naive Bayes spam filter in a real-world email system, various trade-offs must be considered.

- Explain the trade-off between precision and recall in the context of spam filtering. Which metric would you prioritize and why?
- How would you adjust the model to minimize the risk of important legitimate emails being classified as spam? Be specific about which parameters you would modify.
- Our model used only unigrams (single words) as features. Discuss one advantage and one disadvantage of extending the model to include bigrams (word pairs) for spam detection.
- Multinomial Naive Bayes uses Laplace (add-one) smoothing to handle unseen words. Explain why this smoothing is necessary and how it would affect classification of an email containing words not seen during training.

My Ans:

Precision measures: among all emails that we mark as spam, how many of them are true spam emails; while recall measures: among all true spam emails, how many do we mark as spam. They are related to the two types of errors. I think in real-world scenario, the precision would be more useful, because people care more about whether their legitimate emails are mistakenly marked as spam. To minimize the risk of important legitimate emails being classified as spam, we can enhance the decision threshold or adjust the prior P(spam).

One advantage of including the bigram is that, it takes phrases into account; however, including both unigram and bigram would increase the data and decrease the efficiency, and that would be a disadvantage of it.

Smoothing is important to avoid including "zero possibility" when encountering unseen words.