Create virtual environment: python -m venv venv

Activate virtual environment: venv\Scripts\activate

pip install scikit-learn

pip install pandas

pip install nltk

python source/main.py

Explanation of the output:

1. Accuracy: 0.9802690582959641

- **Explanation: Accuracy is the proportion of correct predictions (both spam and non-spam) out of the total number of predictions made.**
- **Interpretation: In this case, the model correctly classified 98.03% of all messages as either spam or not spam. This indicates that the model performs very well overall.**

2. Confusion Matrix:

[[961 4]

[18 132]]

Explanation: A confusion matrix is a table that helps us understand how well the model is performing by showing the counts of true positives, true negatives, false positives, and false negatives:

- **True Negatives (961): The number of non-spam messages correctly identified as non-spam.**
- **False Positives (4): The number of non-spam messages incorrectly identified as spam (Type I error).**

- **False Negatives (18): The number of spam messages incorrectly identified as non-spam (Type II error).**
- **True Positives (132): The number of spam messages correctly identified as spam.**

3. Classification Report:

	precision	recall	f1-score	support
0	**0.98**	**1.00**	**0.99**	**965**
1	**0.97**	**0.88**	**0.92**	**150**
accuracy			**0.98**	**1115**
macro avg	**0.98**	**0.94**	**0.96**	**1115**
weighted avg	**0.98**	**0.98**	**0.98**	**1115**

Explanation of Terms:

- **Precision: The proportion of true positive predictions (spam messages correctly identified as spam) out of all the positive predictions made by the model. For example, a precision of 0.97 for class 1 means that 97% of the messages identified as spam were actually spam.**
- **Recall: The proportion of true positives out of all actual positives in the data. For class 1 (spam), a recall of 0.88 means that 88% of all actual spam messages were correctly identified by the model.**
- **F1-Score: The harmonic mean of precision and recall. This metric balances precision and recall and is useful for understanding the overall performance of the model when there is an uneven class distribution. An F1-score of 0.92 for class 1 (spam) means that the model balances precision and recall well for spam messages.**
- **Support: The number of actual occurrences of each class in the dataset (965 non-spam and 150 spam messages).**

- **Interpretation:**
    - **For class 0 (non-spam), the model has a high precision (0.98), a perfect recall (1.00), and a very high F1-score (0.99).**
    - **For class 1 (spam), the model has a good precision (0.97), a decent recall (0.88), and an F1-score of 0.92. This means the model does a good job at identifying spam but misses some spam messages (as indicated by the recall being less than 1.00).**

4. Accuracy (Overall):

- **0.98: This means that out of 1115 total messages tested, the model correctly classified 98% of them.**

5. Macro and Weighted Averages:

- **Macro avg: The average metrics for each class, calculated without considering the class distribution. It shows how well the model performs across all classes equally.**
    - **Precision: 0.98**
    - **Recall: 0.94**
    - **F1-score: 0.96**
- **Weighted avg: The average metrics weighted by the number of true instances per class. This considers the class distribution in the dataset.**
    - **Precision: 0.98**
    - **Recall: 0.98**
    - **F1-score: 0.98**

Final Notes:

- **Interpretation for Testing: After the evaluation, the program prompts the user to enter an email to test or hit '0' for an example. This indicates that the program is interactive, allowing users to test new emails and see if they are classified as spam or not based on the trained model.**

In summary, this output tells us that the spam filter model performs very well, with high accuracy and good performance metrics across classes, although it has slightly lower recall for spam messages compared to non-spam messages.

Overall flow of the code:

1. **Preprocessing: Raw text data from the dataset is cleaned and prepared for analysis.**
2. **Feature Extraction: The cleaned text is transformed into numerical vectors using CountVectorizer.**
3. **Model Training: A Naive Bayes classifier is trained on the vectorized text data.**
4. **Evaluation: The model is tested on unseen data, and metrics like accuracy, precision, recall, and F1-score are calculated to assess its performance.**
5. **Deployment: The trained model and vectorizer are saved to disk. The test_spam function allows users to input new messages and get real-time predictions.**