Question 1： Input and Basic preprocessing

(1) **parse_data_line:** The content in the CSV is separated by tabs. In the previous "load_data" step, the content was split into lines. The "parse_data_line" method is used to extract the label and statement from each line. It checks if the label is either "positive" or "negative" and converts it to lowercase. The "label" represents the actual value, and "statement" is the text to be analyzed.
(2) **pre_process:** The "preprocess" method is used for word tokenization to clean the data and data normalization. We can also remove stop words and perform stemming on this text to extract more useful features. It helps improve the model's quality and performance.

Question 2： Basic feature extraction

This “to_feature_vector” function is applied to convert a list of tokens into feature vectors. The function receives a list of tokens as input and produces a dictionary in which the keys denote features, while the corresponding values indicate their weights.

I have considered two methods:

(1) **binary feature values:** This method focuses solely on whether words from the vocabulary are present in the text. For sentiment analysis, it may be more crucial to pay attention to the presence or absence of certain specific vocabulary, rather than just their frequency.
(2) **a bag-of-words representation:** The number of occurrences of a token in the preprocessed text. It can better capture the vocabulary information in the text. If the frequency of vocabulary usage is crucial for sentiment analysis, this method would be more suitable.

For some datasets, the binary feature representation might capture important patterns, while in others, the frequency of words might be more informative.

Question 3： Cross-validation

This function is used to evaluate the performance of a classifier using k-fold cross-validation, and we can use the “classification_report” function to output the performance of precision, recall, and f-score of every train data.

The goal of 10-fold cross-validation is to:
• **Maximize Data Utilization:** This technique divides the dataset into 10 approximately equal folds, so that we can utilize the data for testing and training without having to create a separate validation set. The following nine folds are utilized for training, and each fold acts as a test set. This optimizes the use of data, particularly when the dataset is small.
• **Sturdy Performance Assessment:** We can get a more accurate estimate of the model's ability to generalize to new, untested data by carrying out several iterations, where each fold takes a turn acting as the test set.

Question 4： Error analysis

In the provided image, it is evident that there are 1597 instances of True Positives, 712 instances of True Negatives, 157 instances of False Positives, and 218 instances of False Negatives.

Analysis of Positive Label:

False Positives (positive label classified as negative):

Possible reasons: (1). Misinterpretation of positive sentiment cues. (2). Presence of ambiguous or sarcastic language.

False Negatives (negative label classified as positive):

Possible reasons: (1). Negative sentiment expressed subtly. (2). Features inadequately capturing certain negative sentiment words.

Analysis of Negative Label:

False Positives (negative label classified as positive):

Possible reasons: (1). Presence of negative sentiment cues that are misinterpreted. (2). Complex sentence structures leading to misclassification.

False Negatives (positive label classified as negative):

Possible reasons: (1). Positive sentiment expressed subtly. (2). Features inadequately capturing certain positive sentiment words.

In conclusion:

The model has trouble expressing sentiment in subtle ways. It's possible that feature extraction could be improved to better capture nuanced sentiment. Investigating specific misclassified instances could provide insights into areas for improvement.

Refine the pre-processing steps to address the identified challenges, add more context, or modify feature extraction methods to further improve the classifier's accuracy.

Questions 5: Optimising pre-processing and feature extraction

FYI: The file corresponding to this question is located in a notebook named 'question5.ipynb,' adjacent to 'NLP_Assignment_1.ipynb.'

Improvement methods:

(1) Improve preprocessing: After using Lemmatizing to reduce words to their base or root form, the accuracy is 0.853357. I keep Lemmatizing and for further optimization, we can eliminate common stop words that may not contribute much to the meaning.

(2) features: Beyond unigram tokens, I tried bigram(accuracy is 0.809562) and trigram(accuracy is 0.741562). But when I tried to combine all features(unigram, bigram and trigram) together, the accuracy is 0.866940.

(3) after applying stylistic features like the number of words per sentence, we can capture additional information beyond the content itself. The accuracy is 0.852674.

(4) add cost parameter (C) to adjust the cost parameter to control the trade-off between achieving a smooth decision boundary and classifying training points correctly. And use class weights to experiment with assigning different weights to classes, especially in imbalanced datasets. They are crucial for achieving better performance.

(5) After combining the above methods together, the accuracy is 0.854755, camparing to NLP_Assignment_1.ipynb(accuracy is 0.852547), the performance has been improved to some extent.