Error Analysis Report

1.For Initial Script

Decisions and Justifications

1.Tokenization by Word Splitting:

Justification:

It allows the model to treat each word as a separate feature, capturing the basic semantic units.

2.Laplace Smoothing for Naive Bayes:

Justification:

Laplace smoothing (or add-one smoothing) is employed to handle the issue of zero probabilities, Laplace smoothing prevents this problem by assigning a non-zero probability to unseen words, maintaining the model's ability to make predictions for previously unseen terms. *Laplace smoothing is used to handle unknown words, and a pseudo count of 1/len(vocabulary) + 1 is added to prevent zero probabilities for unseen terms*

3.Inclusion of Stop Words:

Justification:

Including stop words in the vocabulary allows the model to consider them as features, capturing potential associations with the target class.

Model Accuracy

```
Training accuracy mean (3-fold cross-validation): 0.63
Testing accuracy mean (3-fold cross-validation): 0.44
```

Confusion Matrix

	performer	director	publisher	characters
performer	101	1	0	1
director	27	66	0	1
publisher	28	0	72	0
characters	63	17	2	21

Micro and Macro Averaged precision and recall

```
Micro-Averaged Precision, Recall: (0.65, 0.65)
Macro-Averaged Precision, Recall: (0.7832294878900001, 0.6516484197479859)
```

Interpretation

The model performs well in identifying instances of the "performer" class, as indicated by a high recall (98%). However, the precision is lower (46%), suggesting a higher rate of false positives. The model shows good performance for the "director" class with high precision (79%) and recall (70%).The model

excels in identifying instances of the "publisher" class, as indicated by high precision (97%) and recall (72%).The model struggles with the "characters" class, as indicated by a low recall (20%). However, when it predicts "characters," it is correct most of the time, as indicated by a high precision (91%) .The "Characters" class is frequently misclassified, primarily as "Performer" (55 times) and, to a lesser extent, as "Director" (20 times).

Do the misclassified texts share common attributes? If so, what are they?

Semantic Similarity:

The classes "Characters" and "Performer" appear to share semantic features, leading to frequent misclassifications between them.

Instances where the model misclassifies "Characters" as "Performer" (55 times) suggest that there might be overlapping language or context that makes it challenging for the model to distinguish between the two classes.

"Characters" being misclassified as "Director" (20 times) suggests a pattern of confusion between these classes, indicating potential similarities or ambiguities in the language used for both.

"Director" being misclassified as "Performer" (16 times) implies that there might be shared language characteristics or contextual nuances between these two classes leading to mispredictions.

What other reasons behind the errors did you identify?

Contextual Ambiguity: Ambiguous language or context within the text might contribute to misclassifications, making it challenging for the model to confidently assign a single class.

Sensitivity to Keywords: The model may be sensitive to specific keywords, phrases, or contextual nuances, contributing to misclassifications.

Interclass Confusion: The high misclassification rate between "Characters" and "Performer" suggests a need for improved feature distinction between these classes.

IMPROVING THE SCRIPT

Since we were using simple tokenization, we will now Implement more advanced tokenization and preprocessing using NLTK's stop words and the Porter Stemmer, this is to addresses the issue of semantic similarity between classes, especially evident in misclassifications involving "Characters" and "Performer." By excluding common words that may not contribute to class discrimination, the model becomes more focused on meaningful terms, potentially reducing misclassifications related to semantic overlap.

Remove common English stop words to focus on meaningful terms. Apply stemming to reduce words to their root form.

2.Improved Script

Decisions and Justifications

Other than the initial main.py code, I implemented more sophisticated tokenization and preprocessing steps. Utilized NLTK's stop words to filter out common English words, aiming to focus on more meaningful terms.

Applied stemming using the Porter Stemmer to reduce words to their root form, aiding in capturing core semantic meanings. Adjusted the training process to incorporate the enhanced preprocessing steps. By preprocessing the text with stop word removal and stemming, the model can focus on more relevant features, potentially improving its ability to discriminate between classes.

Model Accuracy

The updated model exhibits improved training accuracy (from 63% to 69%) and testing accuracy (from 44% to 54%) compared to the original model. The increase in accuracy suggests that the refined tokenization and preprocessing contribute positively to the model's performance.

```
Training accuracy mean (3-fold cross-validation): 0.69
Testing accuracy mean (3-fold cross-validation): 0.54
```

Confusion Matrix

	performer	director	publisher	characters
performer	99	2	0	2
director	16	76	0	2
publisher	13	1	86	0
characters	55	20	4	24

Micro and Macro Averaged precision and recall

```
Micro-Averaged Precision, Recall: (0.7125, 0.7125)
Macro-Averaged Precision, Recall: (0.7803396967331394, 0.7156713488948564)
```

Interpretation

Performer: The model is good at identifying instances of the "performer" class, as indicated by a high recall (96%). However, the precision is lower (54%), meaning that there are some false positives.

Director: The model performs well in terms of precision (76%) and recall (81%) for the "director" class.

Publisher: The model is excellent at identifying instances of the "publisher" class, as indicated by a high precision (96%) and recall (86%).

Misclassifications

The updated model still faces challenges in correctly classifying instances, particularly in the "Characters" class.

Misclassification Patterns: The "Characters" class is still frequently misclassified, but the refined model shows improvement in predicting other classes correctly.

Common Attributes: Misclassifications involving "Characters" may persist due to shared semantic features with other classes.

Do the misclassified texts share common attributes? If so, what are they?

Many misclassifications seem to involve confusion between "Characters" and "Performer" classes.

Instances of misclassification for "Director" often involve confusion with "Performer" as well

What other reasons behind the errors did you identify?

The model might be sensitive to specific keywords, phrases, or contextual nuances that contribute to misclassifications. The misclassifications may also be attributed to ambiguous language or context within the text. Certain phrases or expressions might be open to interpretation, making it challenging for the model to accurately assign a single class. Additionally, the classes "Characters" and "Performer" seem to share semantic features, leading to frequent misclassifications between them. The model may struggle to distinguish subtle differences in meaning between these classes.

Conclusion:

In summary, the error analysis provided crucial insights into the strengths and weaknesses of the initial classification script. The model demonstrated proficiency in certain classes, such as "performer" and "director," but faced challenges, particularly in accurately classifying instances within the "characters" category. The misclassifications revealed underlying issues related to semantic similarity, contextual ambiguity, and sensitivity to keywords.

To address these challenges, strategic changes were implemented in the script, including the incorporation of stop words and stemming during tokenization. These adjustments resulted in a noteworthy improvement in both training and testing accuracy. Specifically, the model exhibited enhanced performance in identifying instances of "performer," "director," and "publisher" classes.

However, despite these positive strides, the refined model still encounters challenges in correctly classifying instances, notably within the "characters" class.