1. **Performance Report**

a) Accuracy

For calculating the training accuracy, we implemented the 3-fold cross-validation strategy where the training dataset was split into 3 folds. Out of these 3 folds, 2 folds were used for training and 1 fold was used for validation. We got predictions on the 1 fold that was used for validation and calculated the accuracy for it. We repeated these steps until each of the 3 folds was used as the validation set.

Then we took the average of the 3 accuracy values (0.8740, 0.8695, 0.8648) and got the **final average training accuracy as 0.8695.**

With respect to the Test Dataset, after training our model on the complete training data, we got the predictions of the model on the test data and **found the Test Accuracy to be 0.8700.**

Training Accuracy	**Testing Accuracy**
0.8695	0.8700

b) Confusion Matrix

We got the following confusion matrix for the test dataset.

Prediction \ Actual	**director**	**publisher**	**performer**	**characters**
director	78	1	4	6
publisher	3	92	3	4
performer	2	2	90	5
characters	11	5	6	88

c) Precision and Recall

We got the following values for Precision and Recall.

	Precision	**Recall**
director	0.8764	0.8298
publisher	0.9020	0.9200
performer	0.9091	0.8738
characters	0.8000	0.8544
Macro avg	**0.8719**	**0.8695**
Micro avg	**0.8700**	**0.8700**

2. Justification of Design

First of all we tried to split the input sentences based on whitespace to get individual words from each sentence. For the cleaning of these sentences, we removed special characters, symbols, punctuations from these words using python function string.isalnum(). We also converted each of the words to lowercase. We effectively converted the full input file into a 2-dimensional list of sentences with each sentence list composed of individual words (tokens). We also created our Vocabulary (set of unique words) for the corpus from these tokens.

For the training of the Naive Bayes, we are implementing the Bag of Words technique as the featurization technique. In this technique, we represent each sentence as the list of counts of all words present in that sentence. For training the sentences using Naive Bayes, we have to calculate the probabilities as per the formula. There can be a situation where we observe a word in a particular class but not for other class sentences. So when we try to find the likelihood of this word with respect to other classes, the probability value will be 0 which would make the whole probability for the other classes as 0 irrespective of other words in the sentence. To solve this issue, we followed the technique of Laplace Smoothing (add-1), where we add 1 to the count of each word and adjust the probability of all words.

Another situation can be that we encounter a word in the test sentences which is not part of our vocabulary created using the training data. To handle this scenario, we are following the standard approach of simply ignoring such words while calculating probabilities.

Since our model is not overfitting (train and test accuracies are comparable) and overall accuracy is pretty high, we decided not to utilize the head and tail entities as that may result in overfitting of our model and would also increase the dimensionality further.

3. Error Analysis

Looking at the confusion matrix above in 1.b), we observe that for 11 data records, our model is incorrectly classifying director label sentences as character labels. For example, for the sentence *“At the age of six he played the main role in Satyajit Ray ’s film "" Sonar Kella”* our model made this wrong prediction. We think this happened because in general, the sentences of the two labels are very similar. The sentences of both these classes describe some person who is either assuming the role of a character in the film or is the main creator of the film. So features (words) like movie name, character name, etc would be common for all sentences of these 2 labels. Eg. we observed that for the class “director” and for the class “character” some common features included “actress”, “film”, “released”, “performed”, “starred”, “role”.

Hence this can confuse our model between the 2 labels. We expect less common features between these 2 labels and the other 2 labels of publisher, performer, and this is what we observe in our model predictions as well. (only 2 records were misclassified as a performer and only 3 records were misclassified as publisher when the actual label was director)

To further improve this result, 1 solution can be to increase the training size for our model which would give it more instances to learn how to distinguish between the labels. Right now we have 520 instances of the “publisher” class, 502 instances of the “director” class, 493 instances of the “performer” class, and 485 instances of the “characters” class. We can observe that data records for “characters” class are slightly less than others. So increasing the number of instances may further increase the performance of our model.

This observation is further validated because as we can see the precision for the class “characters” is the lowest at 0.8 (see table in 1.c) . So increasing the no of instances for the “characters” class may help to improve precision (reduce false positives) as well.

But overall, we are getting good results as our model is not overfitting (train and test accuracy are comparable) so this model definitely solves the task of relation-extraction for the given corpus.