Text Classification Assignment

DSA_202101_ 25:

Mohammed Adnan - 300345146

Kartik Banga - 300344046

Harish Nair - 300375297

Zeinab Hussein Keserwan - 300274519

University of Ottawa

DTI 5125: Data Science Applications

Professor: Arya Rahgozar

Winter 2024

1. Introduction

Authorship Attribution is a task within the domain of Natural Language Processing (NLP) that aims to identify and attribute written texts to their respective authors based on linguistic patterns and style. This task is very intricate due to diverse writing styles, language nuances, and vocabulary differences exhibited by various authors. Despite its intricacies, authorship attribution has played a crucial role in identifying anonymous or disputed texts, attributing authorship to historical documents, and even unmasking authors in the digital realm.

In this work, our objective is to build a machine learning model capable of predicting the authorship of a given text document based on its content. We started by extracting the text partitions needed for training and testing our machine learning model. We labeled them as per the author they belong to and converted them into numerical representations using methods such as Bag-of-Words, N-grams, TF-IDF (term frequency-inverse document frequency), word2vec, and doc2vec. We then used part of these numerically represented documents to train several machine learning models, namely K-Nearest Neighbors, Naïve Bayes, Multinomial Logistic Regression, Decision Trees, and Random Forest to predict authorship. Subsequently, we employed the trained models to categorize new and unseen documents based on their content. We chose the champion model to be the most accurate model in categorizing the unseen documents, and we studied the properties of the documents that threw the model off.

2. Data Extraction

We sourced our data from the Gutenberg Project library, a volunteer initiative which offers access to an extensive collection of over 60,000 eBooks. We selected six books written by six different authors and belonging to the same collection, “Harvard Classics,” which is a comprehensive resource for self-education and intellectual enrichment:

1. “The Alchemist” by Ben Jonson
2. “Apology” by Plato
3. “Don Quixote” by Miguel de Cervantes Saavedra
4. “David Copperfield” by Charles Dickens
5. “The Odyssey” by Homer
6. “Crime and Punishment” by Fyodor Dostoyevsky

All Gutenberg books start with the opening statement “START OF THE PROJECT GUTENBERG EBOOK,” followed by the title and author name, and end with the closing statement “END OF THE PROJECT GUTENBERG EBOOK.” To do any NLP effectively, we should extract partitions from the raw text that lies between the end of the author name and the beginning of the closing statement. To address this, for each book, we searched for the regular expressions “.*start.*ebook.*\n”, title, author name, and “.*end.*ebook.*\n” to accurately identify the start and end points of the true raw text content. We then extracted 200 random partitions of 100 words each from the raw text content of each book. This process resulted in 1200 partitions i.e. documents. We labeled each document as per the author it belongs to: ‘a’ for Ben Jonson, ‘b’ for Plato, ‘c’ for Miguel de Cervantes Saavedra, ‘d’ for Charles Dickens, ‘e’ for Homer, and ‘f’ for Fyodor Dostoyevsky. We then saved them in dataframe format and into “sample_partitions.csv:”

	Book Title	Book Author	Label	Partition
0	The Alchemist	Ben Jonson	a	He 's sent to , far and near , all over Englan...
1	The Alchemist	Ben Jonson	a	my Dol ; And thou mayst make his ransom what t...
2	The Alchemist	Ben Jonson	a	give your poor friend leave , though no philos...
3	The Alchemist	Ben Jonson	a	dignity . GRAY , badger . GRICE , cub . GRIEF ...
4	The Alchemist	Ben Jonson	a	mas acabada hermosura , que he visto en mi vid...
...	...	...	...	...
1195	Crime and Punishment	Fyodor Dostoyevsky	f	was still lying in the coffin , Svidrigaïlov w...
1196	Crime and Punishment	Fyodor Dostoyevsky	f	of merit ; I bet he has the Anna in his button...
1197	Crime and Punishment	Fyodor Dostoyevsky	f	on the staircase , listened long and intently ...
1198	Crime and Punishment	Fyodor Dostoyevsky	f	of feeling and education . Know then that my w...
1199	Crime and Punishment	Fyodor Dostoyevsky	f	I maintain that he is not cunning , not practi...

1200 rows × 4 columns

3. Data Cleansing

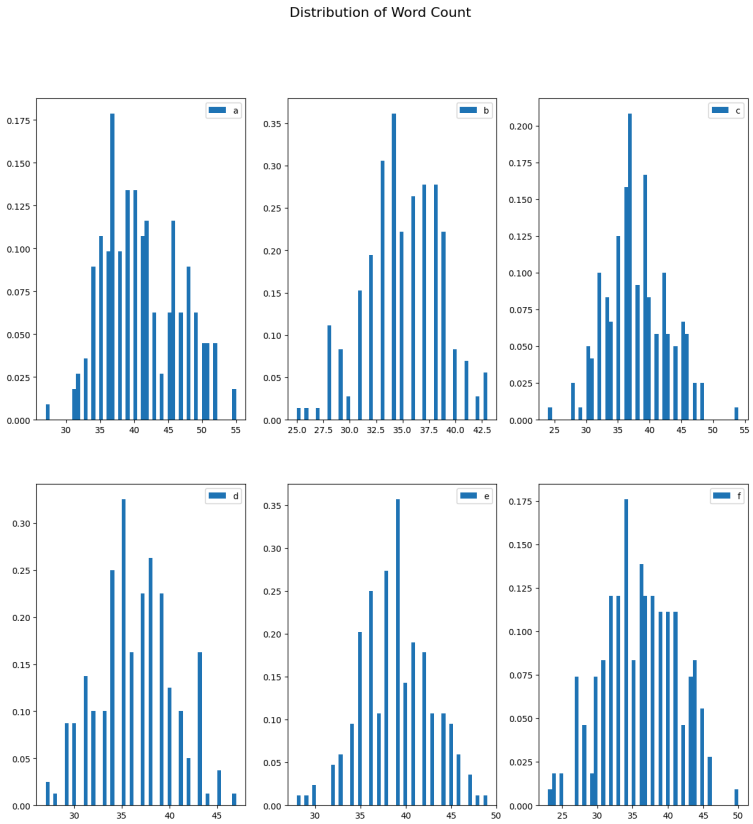
We started by converting each document to lowercase since our focus is on the meaning of words rather than their specific case. This will promote uniformity and consistency in the representation of text and will reduce the vocabulary size. We then removed the punctuations from each document by keeping the characters which are not in **!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~‘“”**. We then tokenized each document into a list of words or tokens using nltk.word_tokenize. We then removed the stopwords by keeping each word that is not in nltk.corpus.stopwords.words('english'). We then tried to reduce each word in each document to its root form by using the Porter Stemmer and the Word Net Lemmatizer from nltk. The results were as follows:

Label	Partition	Partition_stemmed	Partition_lemmatized
a	He 's sent to , far and near , all over Englan...	sent far near england counsel know fortun ka g...	sent far near england counsel know fortune ka ...
a	my Dol ; And thou mayst make his ransom what t...	dol thou mayst make ransom thou wilt dousabel ...	dol thou mayst make ransom thou wilt dousabel ...
a	give your poor friend leave , though no philos...	give poor friend leav though philosoph laugh t...	give poor friend leave though philosopher laug...
a	dignity . GRAY , badger . GRICE , cub . GRIEF ...	digniti gray badger grice cub grief grievanc g...	dignity gray badger grice cub grief grievance ...
a	mas acabada hermosura , que he visto en mi vid...	ma acabada hermosura que visto en mi vida face...	ma acabada hermosura que visto en mi vida face...
...	...	...	...
f	was still lying in the coffin , Svidrigaïlov w...	still lie coffin svidrigaïlov busi make arrang...	still lying coffin svidrigaïlov busy making ar...
f	of merit ; I bet he has the Anna in his button...	merit bet anna buttonhol put goe dine contract...	merit bet anna buttonhole put go dine contract...
f	on the staircase , listened long and intently ...	staircas listen long intent look last time pul...	staircase listened long intently looked last t...
f	of feeling and education . Know then that my w...	feel educ know wife educ highclass school daug...	feeling education know wife educated highclass...
f	I maintain that he is not cunning , not practi...	maintain cun practis probabl first crime suppo...	maintain cunning practised probably first crim...

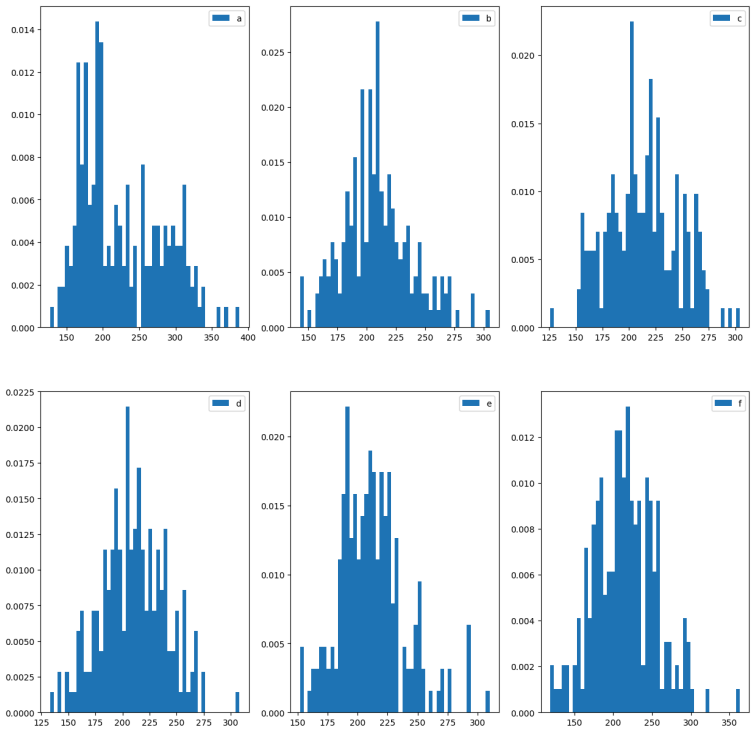
We decided to go with the lemmatized version of the documents as lemmatization is more accurate than stemming. In fact, lemmatizing reduces words to their base or dictionary form (lemma), ensuring that the resulting form is a valid word, while stemming reduces words to their root or base form (stem) by removing prefixes or suffixes, often without ensuring that the result is a valid word.

4. Data Analysis

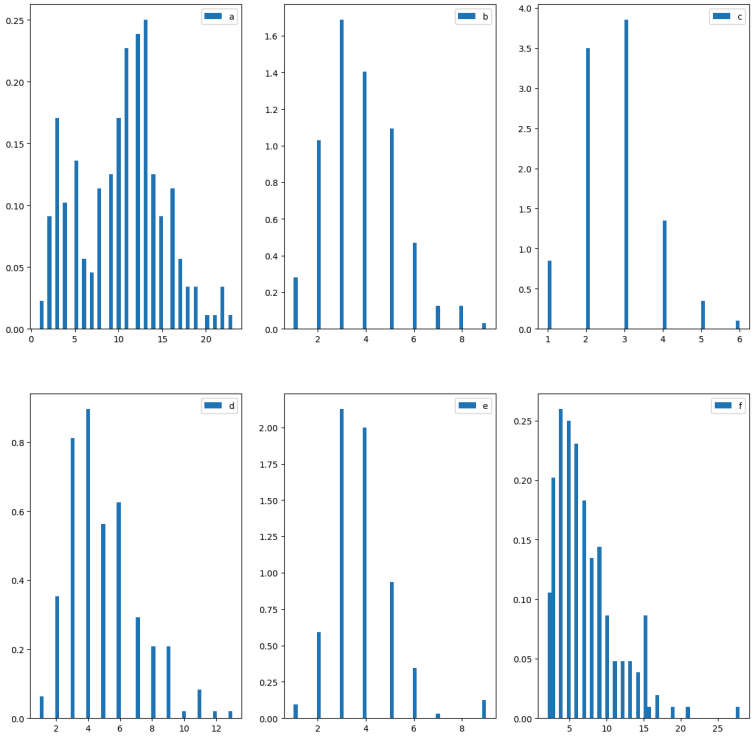
- **Length Analysis:** We computed the following features for each document: word count, character count, sentence count, average word length, and average sentence length. We plotted the distribution of each of these features among: 1) documents belonging to a certain author, 2) all documents. As we can see from the plots, the distributions among each author are normal distributions with means and standard deviations that slightly differ. Therefore, we can say that word count, character count, sentence count, average word length, and average sentence length are similar between the authors and thus wouldn't help in differentiating between their writing styles. Regarding the distributions of the features among all documents, word count is normal with mean approximately at 37 words per document (after removal of stop words), character count is normal with mean approximately at 200 characters per document, sentence count is skewed normal with mean approximately at 4 sentences per document, average word length is normal with mean approximately at 5.75 characters per word, and average sentence length is skewed normal with mean approximately at 10 words per sentence. Therefore, we can say that, in general, these authors use short sentences with medium-length words.

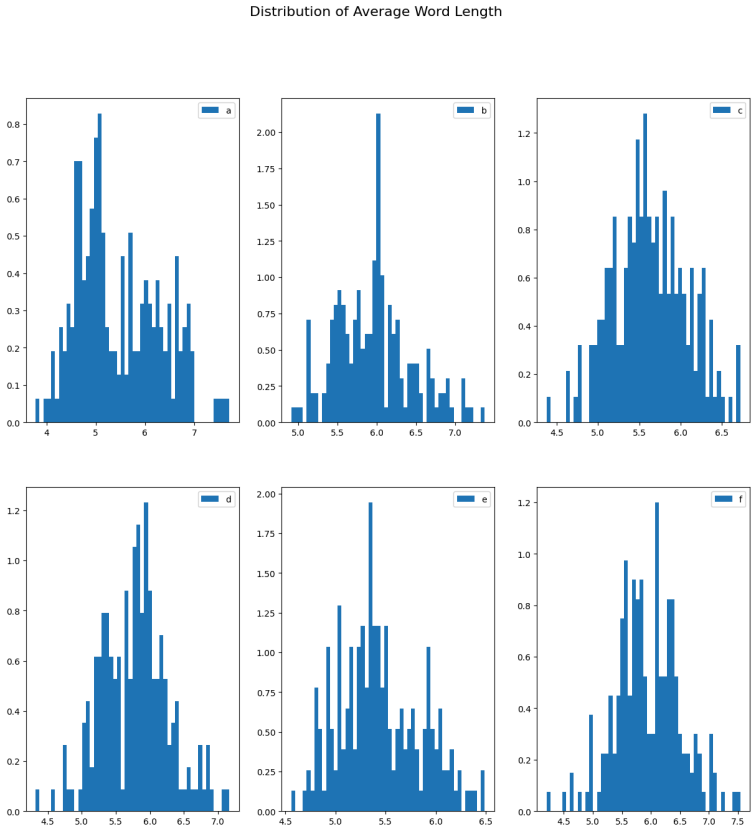


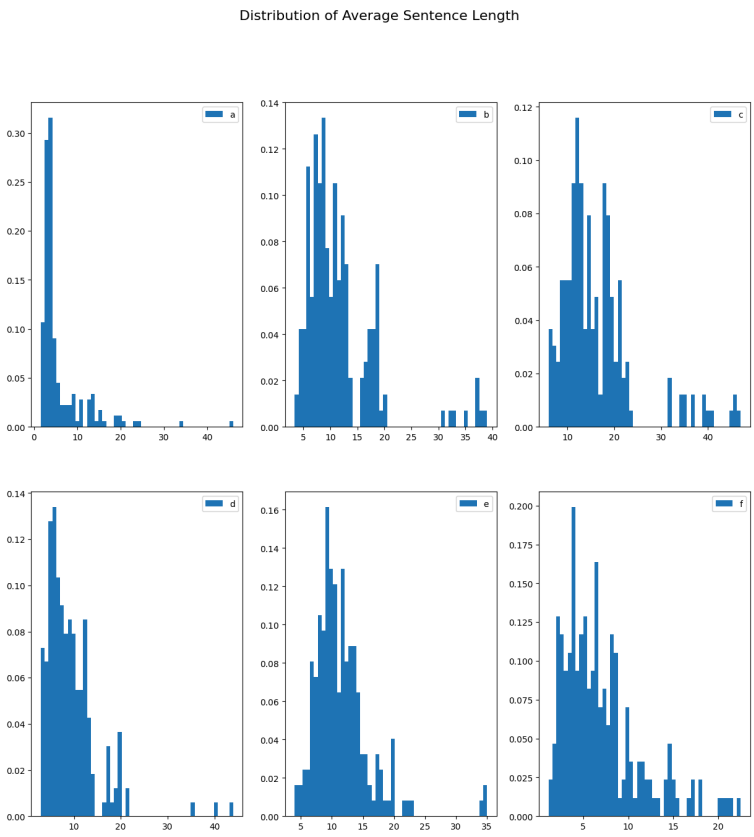
Distribution of Character Count

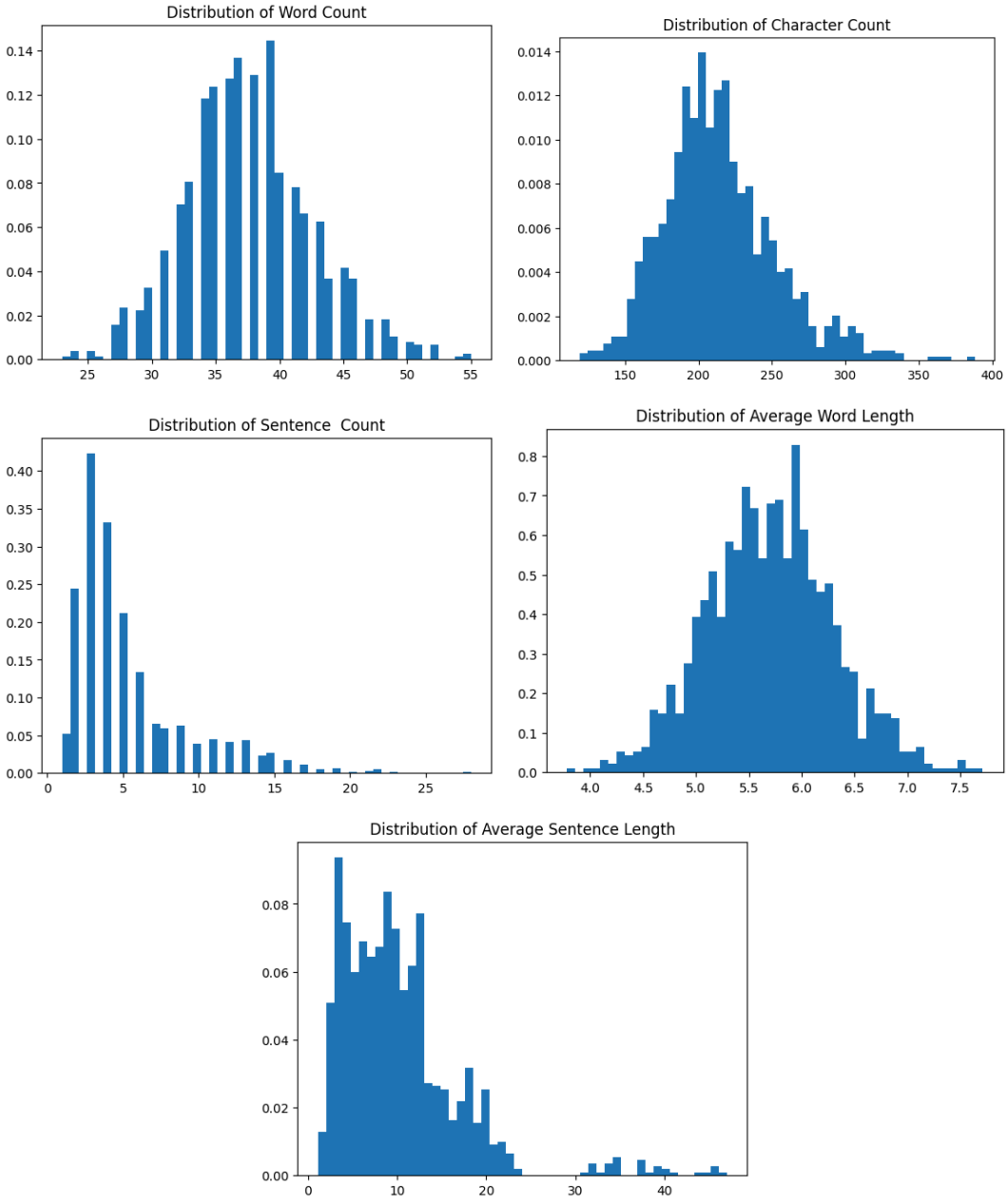


Distribution of Sentence Count



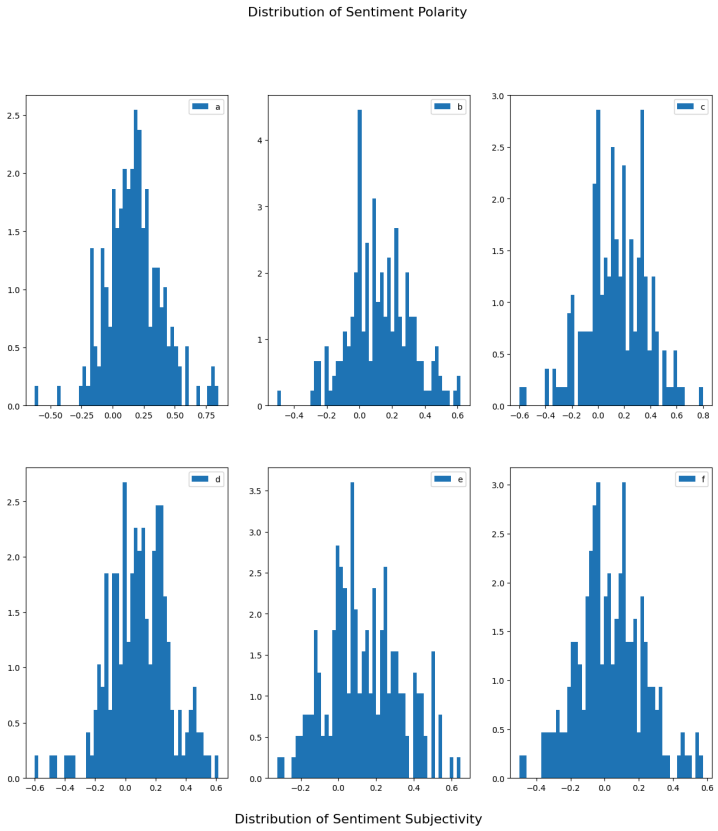

Distribution of Average Word Length
a
b
c
d
e
f

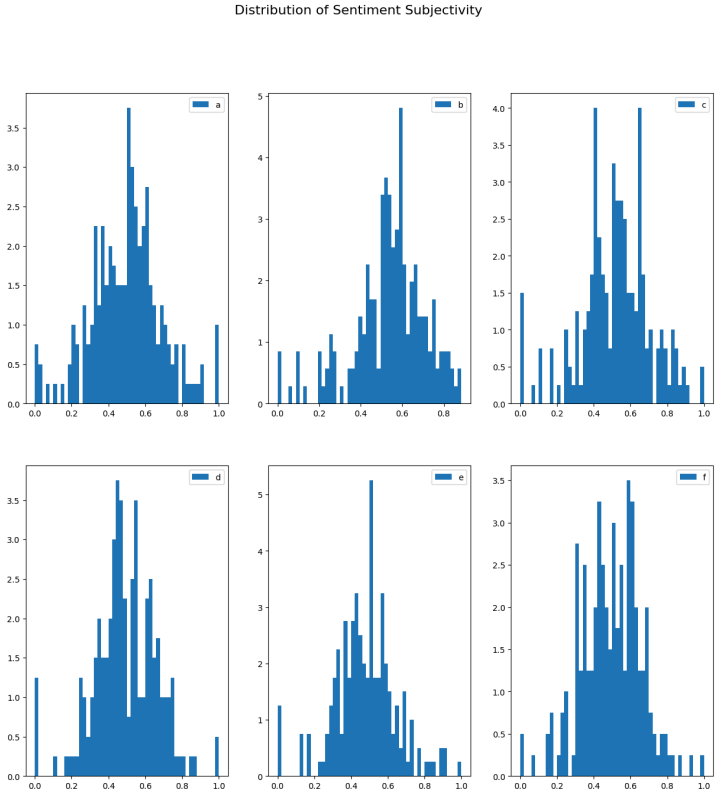

Distribution of Average Sentence Length
a
b
c
d
e
f

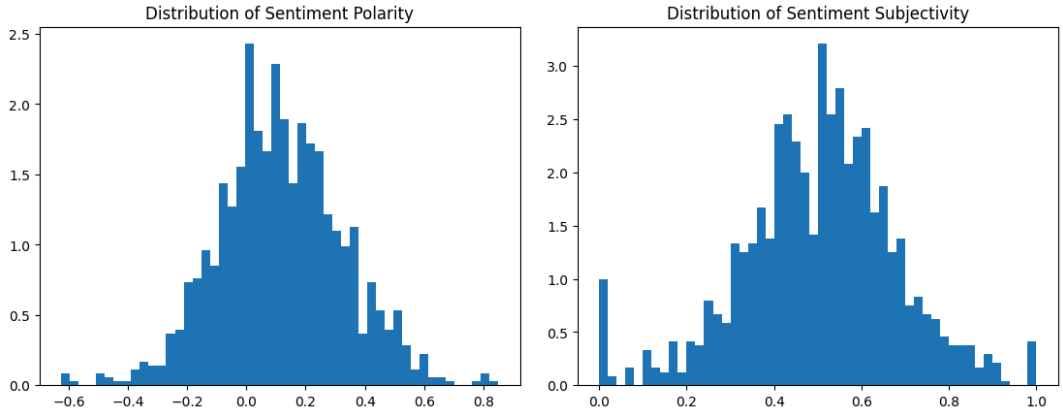


- **Sentiment Analysis:** It is the process of determining the tone or attitude of the author. It includes a polarity score which indicates the sentiment's positivity or negativity, ranging from -1 (negative) to 1 (positive). It also includes a subjectivity score which measures the text's objectivity or subjectivity, ranging from 0 (objective) to 1 (subjective). We computed both scores for each document using TextBlob. We plotted the distributions among: 1) documents belonging to a certain author, 2) all documents. As we can see from the plots, the distributions among each author are normal distributions with means and standard deviations that slightly differ. Therefore, we can say sentiment polarity and subjectivity are similar between the authors and thus wouldn't help in differentiating between their writing styles. Regarding the distributions among all documents, sentiment polarity is normal with mean approximately at 0.1, and sentence subjectivity is normal

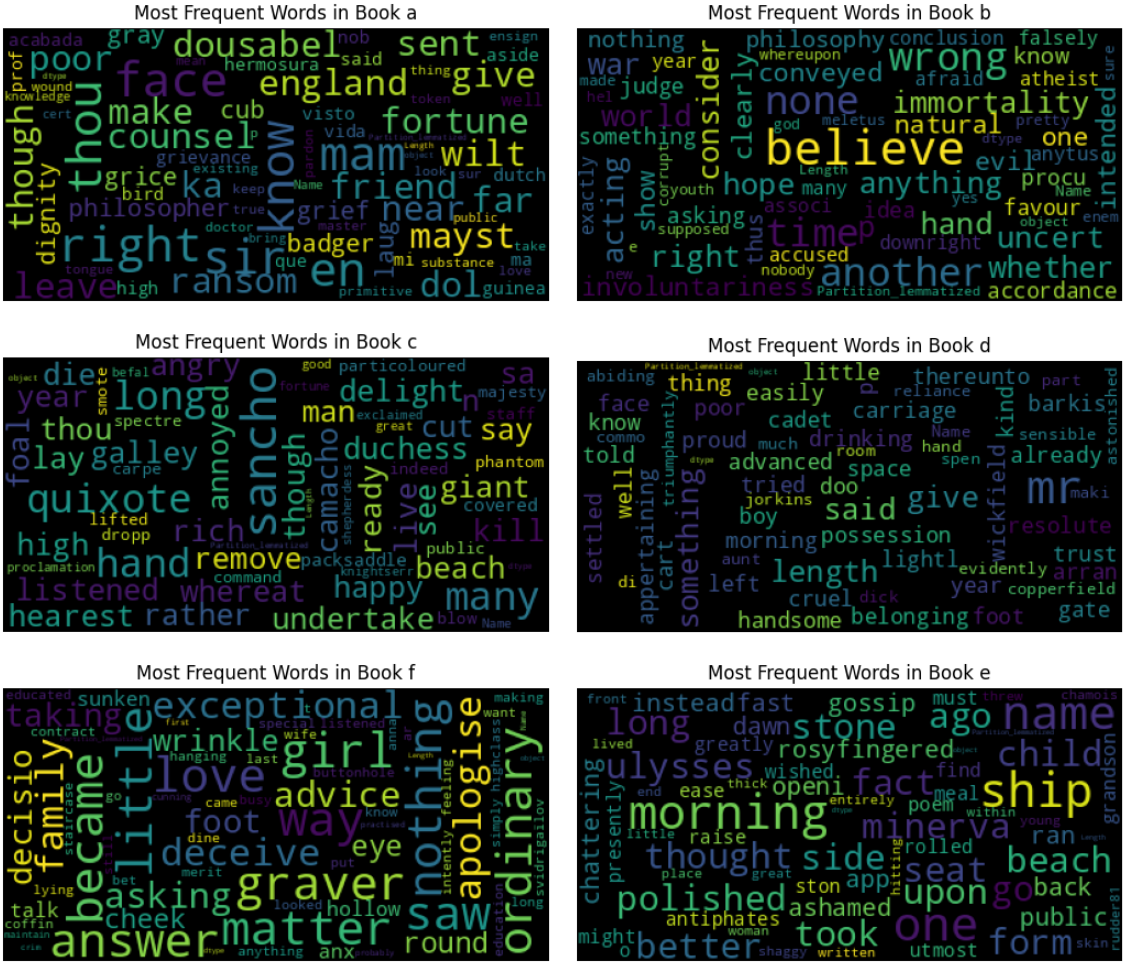
with mean approximately at 0.5. Therefore, we can say that all authors are neutral in average and have a mix of subjective elements (opinions) and objective elements (facts).

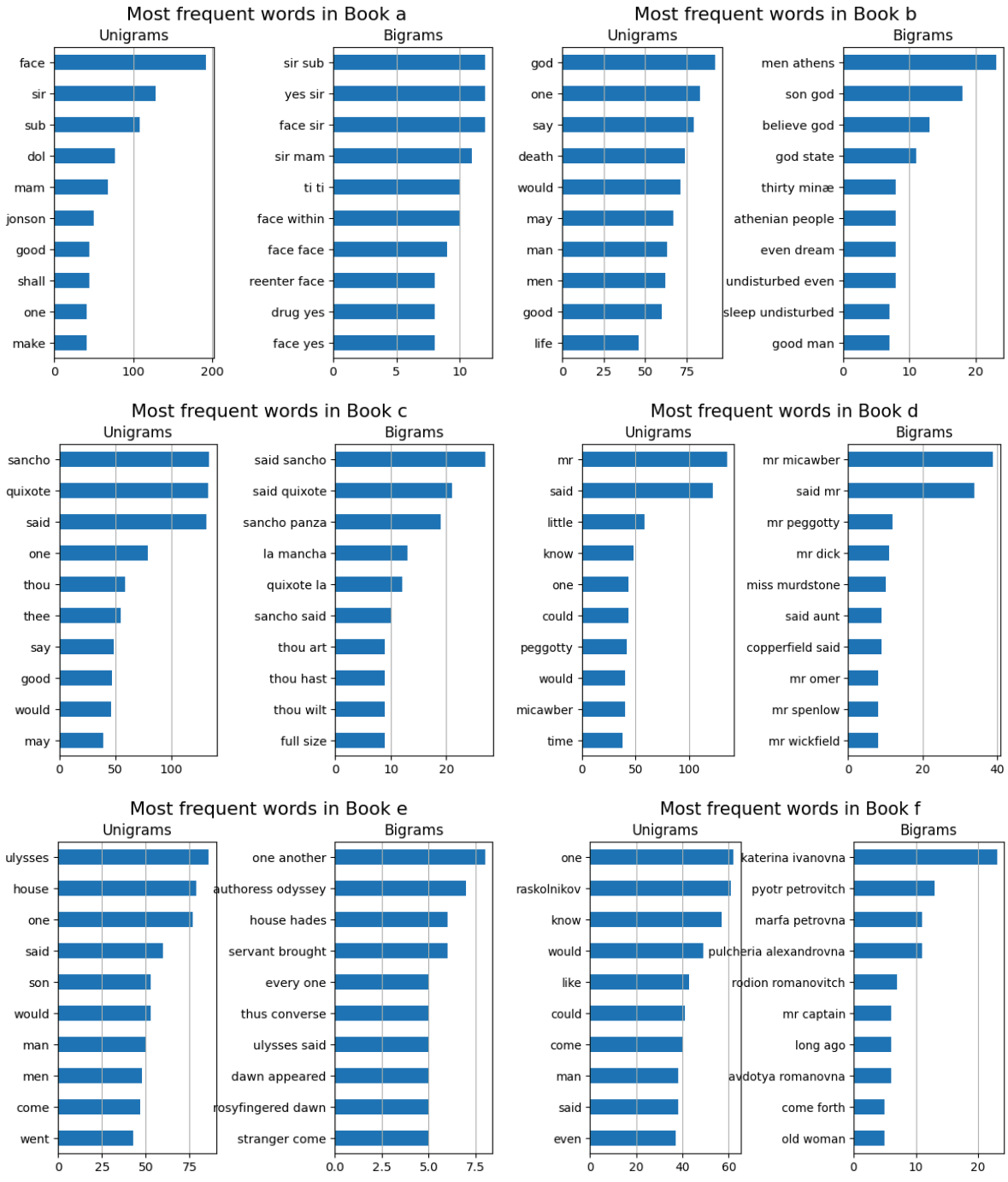






- **Frequency Analysis:** We visualized some of the frequent words in each book, i.e. for each author, using WordCloud. The larger the size of the word in the WordCloud, the more frequent it is in the book. We also plotted the 10 most frequent unigrams and bigrams in each book using nltk.FreqDist.





5. Splitting Data into Training and Testing

We shuffled the data and split it into training and testing using train_test_split of sklearn where 20% of the data was taken as test data, i.e. our test set contains 240 documents and our training set contains 960 documents.

6. Feature Engineering: Document Vectorization

Vectorization is the process of representing words, phrases, or entire documents as numerical vectors, where each dimension of the vector corresponds to a specific feature or aspect of the text. It is essential before doing any NLP as it enables machine learning algorithms to understand and work with textual information. In our work, we used Bag-of-Words, N-grams, TF-IDF, word2vec, and doc2vec to vectorize our documents:

- **Bag-of-Words:** This method treats each document as an unordered set of words and represents it as a vector of word frequencies. The process begins with the creation of a vocabulary, a unique set of words present in the entire corpus or collection of documents. Each document is then transformed into a vector, where the length of the vector corresponds to the size of the vocabulary. The elements of the vector represent the frequency of each word in the document. While BoW simplifies the representation of text, it does not consider the order of words or semantic relationships between them. Additionally, it treats all words equally, neglecting the importance of specific terms.
- **N-grams:** In this approach, a document is represented by sequences of contiguous words of length 'n' (referred to as n-grams). This allows the model to capture not only individual words but also the relationships between adjacent words.This can be particularly useful in tasks where word order and context are critical. The process involves creating a vocabulary of all unique n-grams present in the corpus, and each document is then transformed into a vector where each element corresponds to the count or presence of a specific n-gram. The choice of 'n' determines the size of the sliding window used to extract these sequences. In our work, we considered unigrams (individual words) and bigrams (pairs of adjacent words). So, the resulting feature matrix included both single words and two-word combinations.
- **TF-IDF:** This vectorization method aims to highlight terms that are distinctive to a document while downplaying common terms that appear frequently across many documents. It begins by creating a document-term matrix where each row corresponds to a document, and each column corresponds to a unique term in the entire corpus. The TF-IDF value for each term in a document is calculated as the product of its term frequency (TF), which is how often a term appears in a specific document, and its inverse document frequency (IDF), which penalizes terms that are too common across all documents. TF-IDF vectorization is particularly effective in capturing the discriminative power of terms within a document, making it valuable for tasks such as information retrieval, text classification, and document clustering.
- **Word2Vec:** This method transforms words into continuous vector embeddings. It captures the semantic relationships between words based on their context within a given corpus and consequently represents words with comparable meanings with vectors that are closer to each other. In our work, we trained the Word2Vec model on the corpus of our clean tokenized data to represent each word which occurs at least twice in the text corpus as a word vector of 100 dimensions. Then, for each document, we obtained the word vectors for each word and averaged them to obtain the vector representation of the document. It is important to note that word2vec is not really intended to create representations of documents. We just

crudely averaged word vectors to get a document level representation, but that loses information.

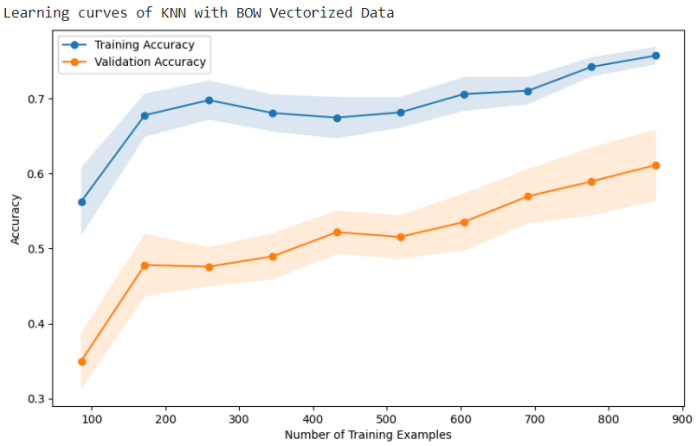
- **Doc2Vec:** This approach provides a means to represent entire textual documents as continuous vector embeddings. In our work, we trained the Doc2Vec model on the collection documents which we tagged using their index to represent each document with a vector of 100 dimensions.

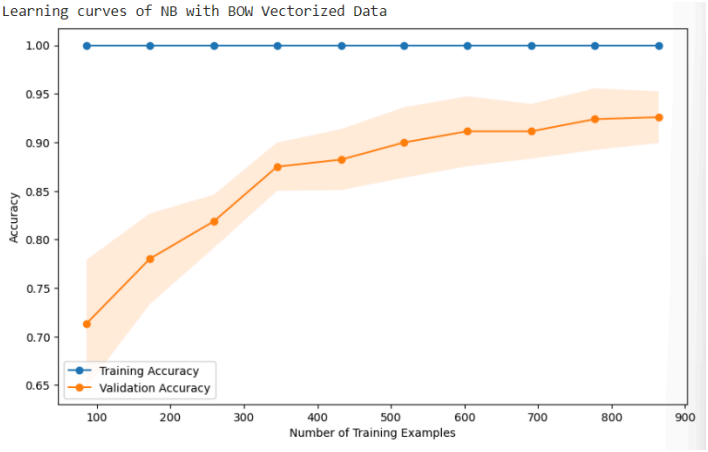
To identify the vectorizers which most accurately represented the documents, we took the Random Forest Classifier as a simple baseline model and trained it to classify each type of vectorized training data. We then tested the baseline classifier to predict the authorship of each type of vectorized testing data. We realized that the baseline classifier was able to accurately classify BOW, N-grams, and TF-IDF vectorized data (test accuracy around 86%), while it failed to classify word2vec and doc2vec vectorized data (test accuracy below 32%). Therefore, we can say that BOW, N-grams, and TF-IDF were more efficient in representing the data than word2vec and doc2vec. Hence, we decided to proceed with BOW, N-grams, and TF-IDF vectorized data.

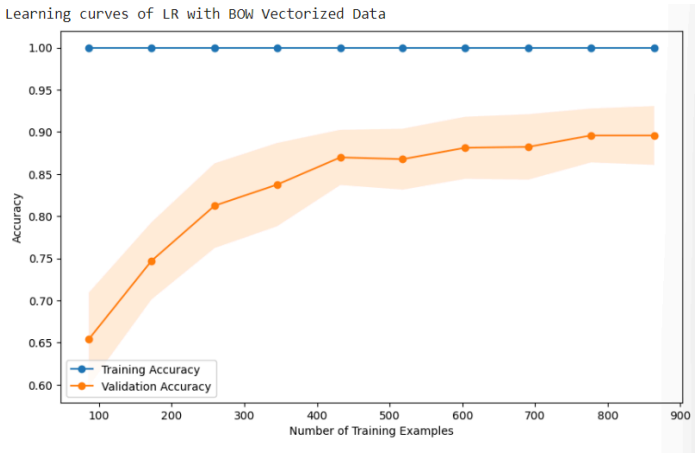
7. Training and Evaluation of Several Machine Learning Models

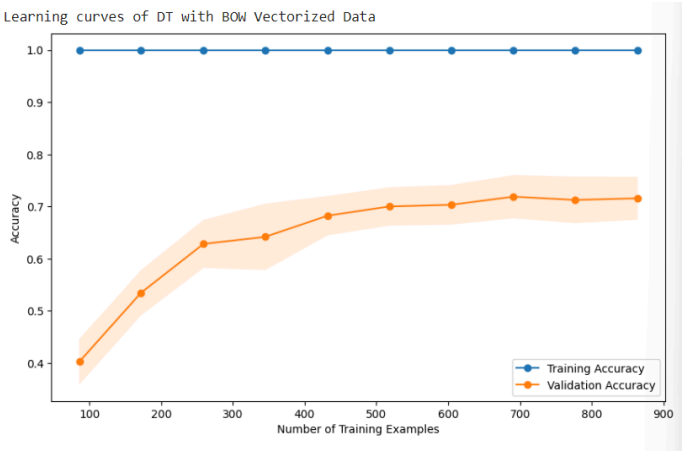
For authorship attribution, we trained each of the following machine learning classifiers KNN, NB, LR, DT, and RF on BOW vectorized training data, N-gram vectorized training data, and TF-IDF vectorized training data. We trained them and evaluated them repeatedly using 10-fold cross-validation. We plotted the learning curves for each classifier, i.e. the evolution of training and testing accuracies as the size of the training data set increases, as well as the evolution of the testing F1 scores and testing accuracies across the 10 folds. The results were as follows:

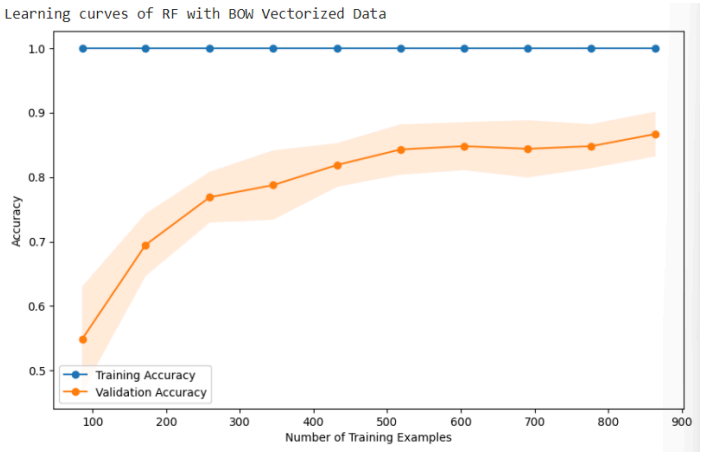
- **BOW-Vectorized Data:** We can see that for BOW-Vectorized data, the KNN classifier is too simple and cannot effectively capture the problem complexity and learn from the data, while NB, LR, DT, and RF were able to effectively capture the model complexity and learn from the data. However, there is a significant gap between the training and validation accuracies when using LR, DT, and RF indicating overfitting and poor generalization. This gap is much smaller for NB indicating good generalization. Based on the evolution of the testing F1 scores and testing accuracies across the 10 folds, we realize that KNN and DT are poor classifiers for BOW-vectorized data while NB, LR, and RF performed much better. All in all, the NB classifier is the best for BOW-vectorized data as it captures the problem complexity, learns well from the data without overfitting, and generalizes well.

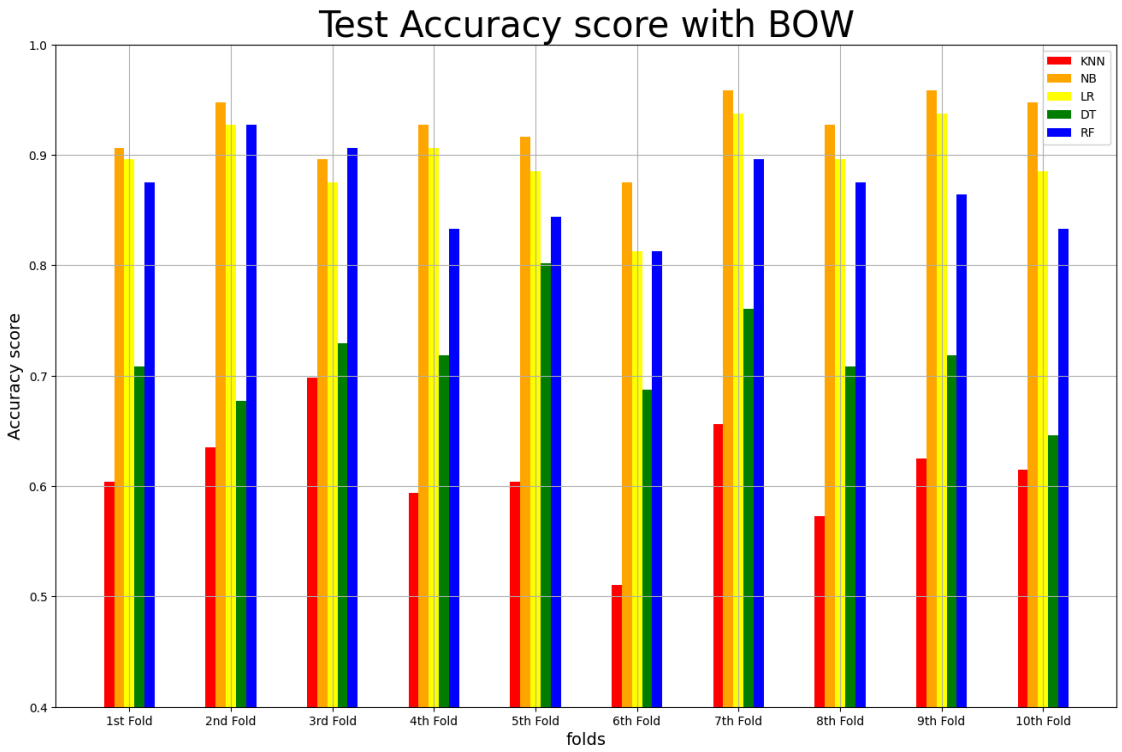


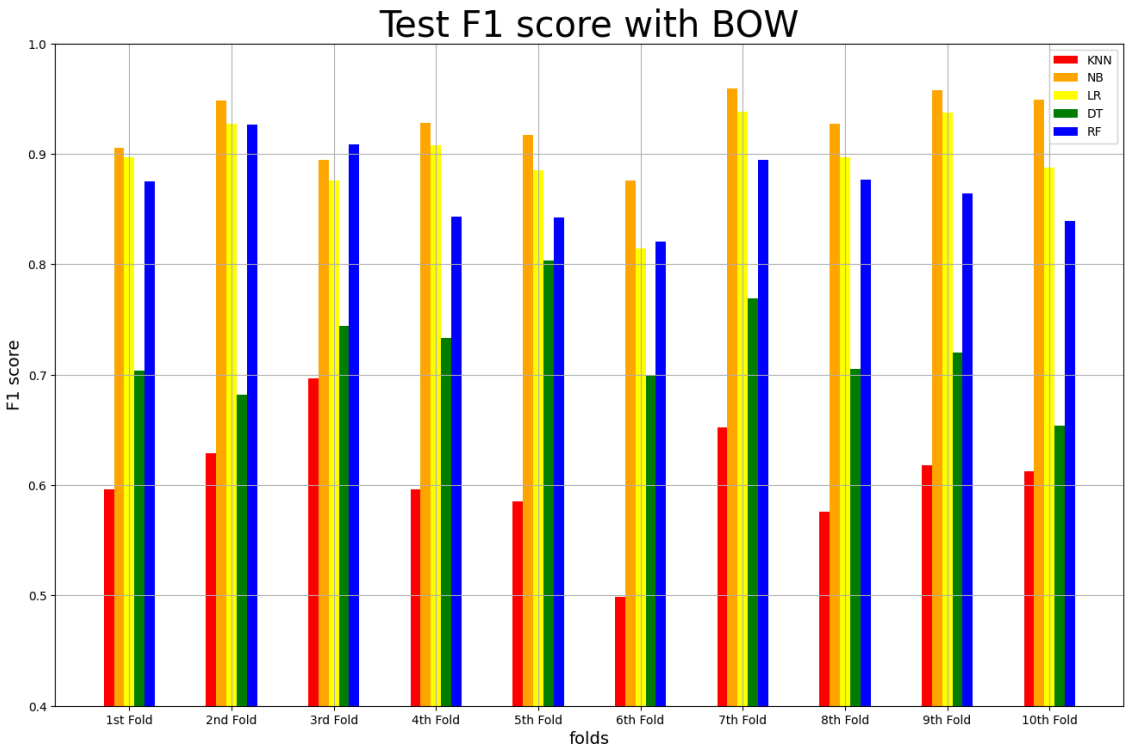

Learning curves of NB with BOW Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy


Learning curves of LR with BOW Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy

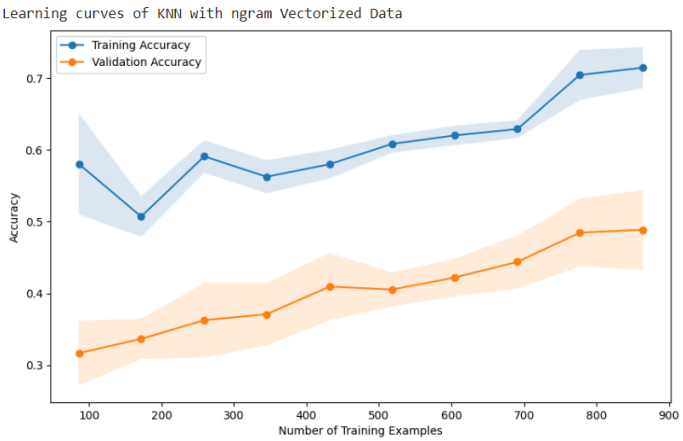

Learning curves of DT with BOW Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy

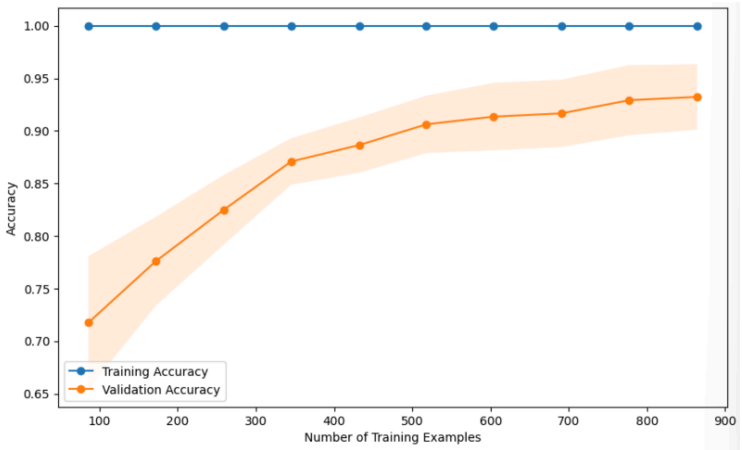

Learning curves of RF with BOW Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
100
200
300
400
500
600
700
800
900


Test Accuracy score with BOW
Accuracy score
folds
KNN
NB
LR
DT
RF
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1st Fold
2nd Fold
3rd Fold
4th Fold
5th Fold
6th Fold
7th Fold
8th Fold
9th Fold
10th Fold

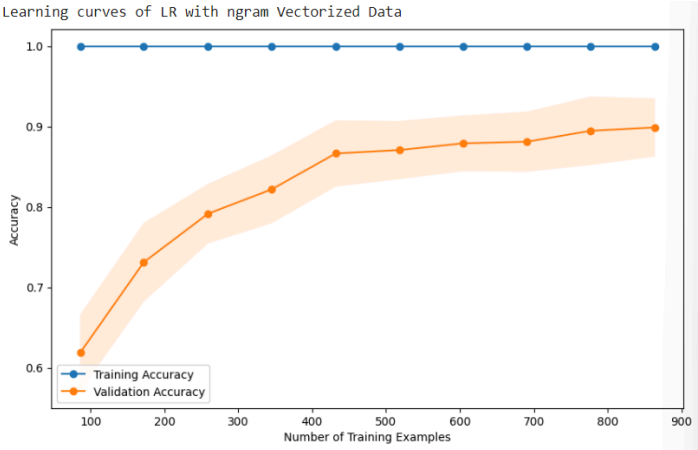


- **N-grams-Vectorized Data:** We can see that for N-grams-Vectorized data, the KNN classifier is too simple and cannot effectively capture the problem complexity and learn from the data, while NB, LR, DT, and RF were able to effectively capture the model complexity and learn from the data. However, there is a significant gap between the training and validation accuracies when using LR, DT, and RF indicating overfitting and poor generalization. This gap is much smaller for NB indicating good generalization. Based on the evolution of the testing F1 scores and testing accuracies across the 10 folds, we realize that KNN and DT are poor classifiers for N-gram-vectorized data while NB, LR, and RF performed much better. All in all, the NB classifier is the best for N-gram-vectorized data as it captures the problem complexity, learns well from the data without overfitting, and generalizes well.

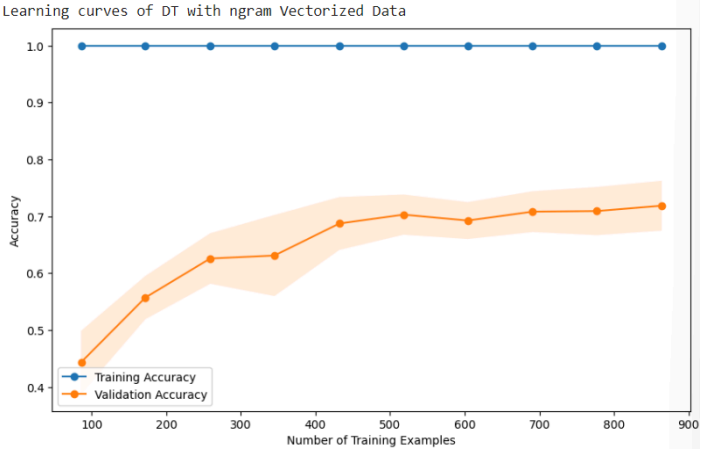


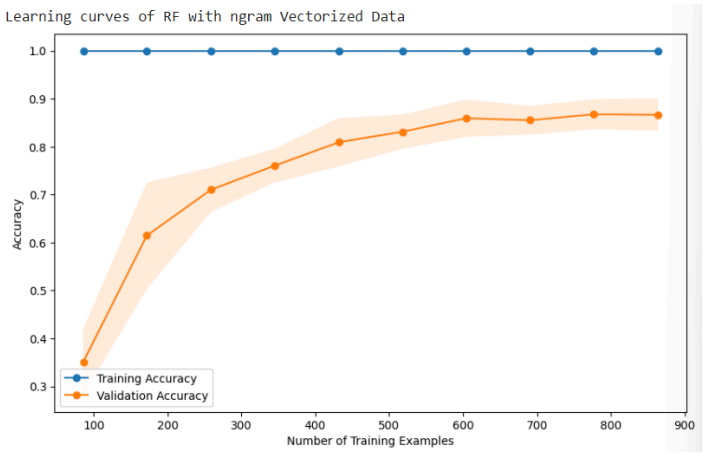


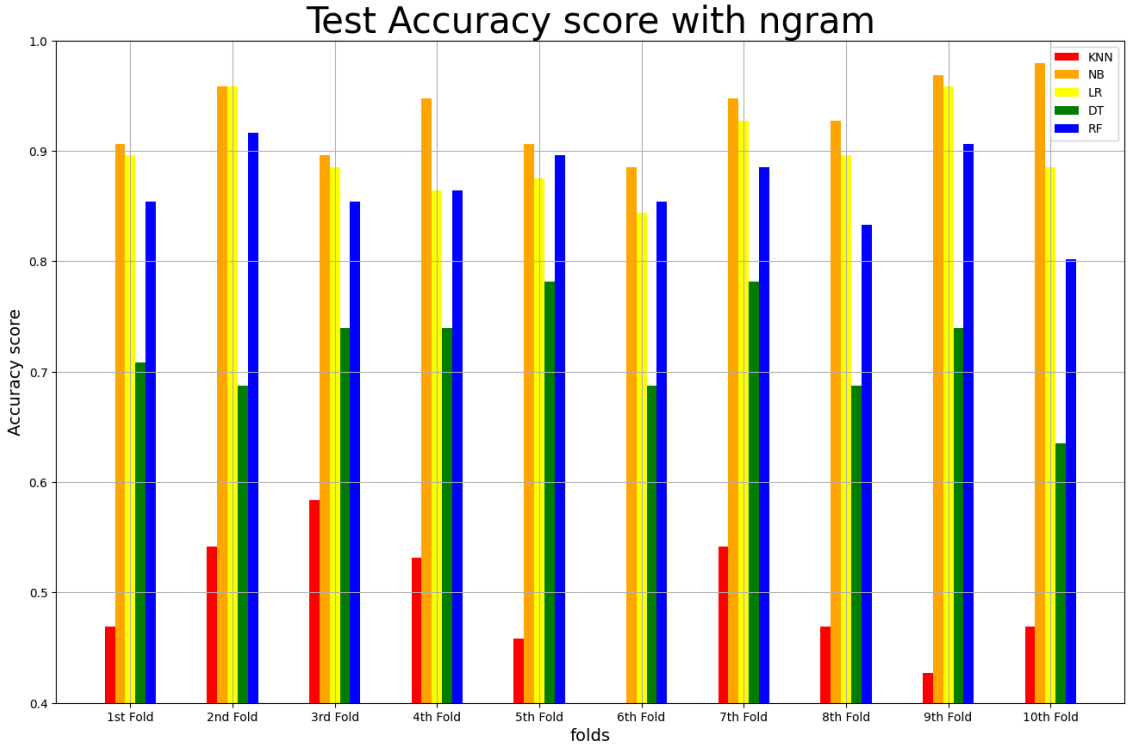
Learning curves of LR with ngram Vectorized Data

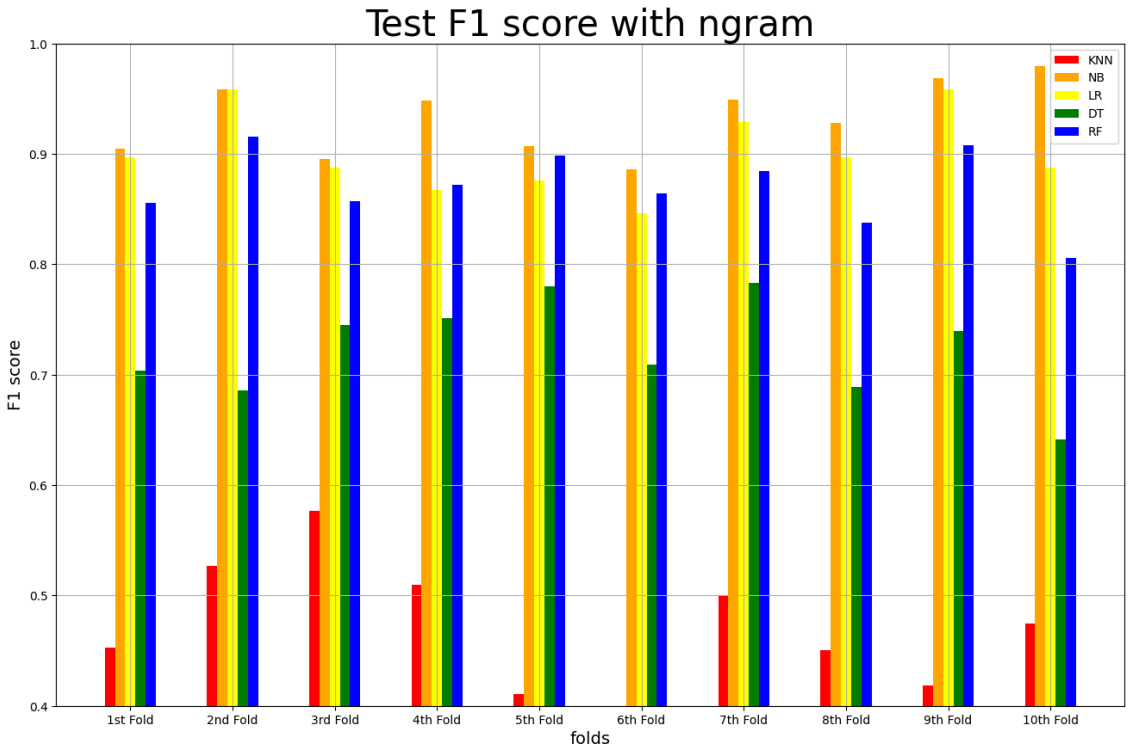


Learning curves of DT with ngram Vectorized Data

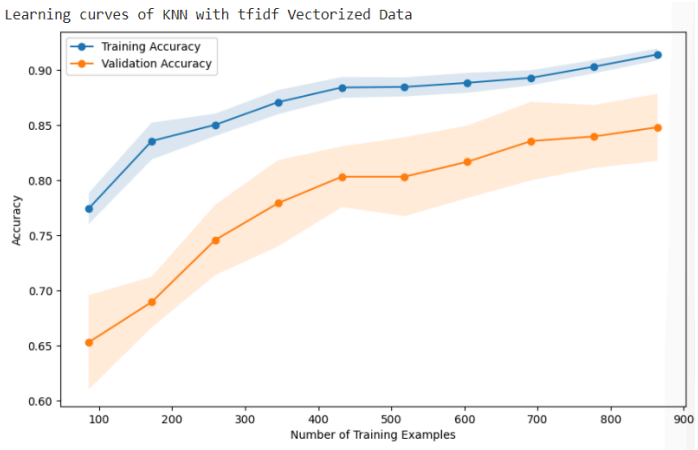


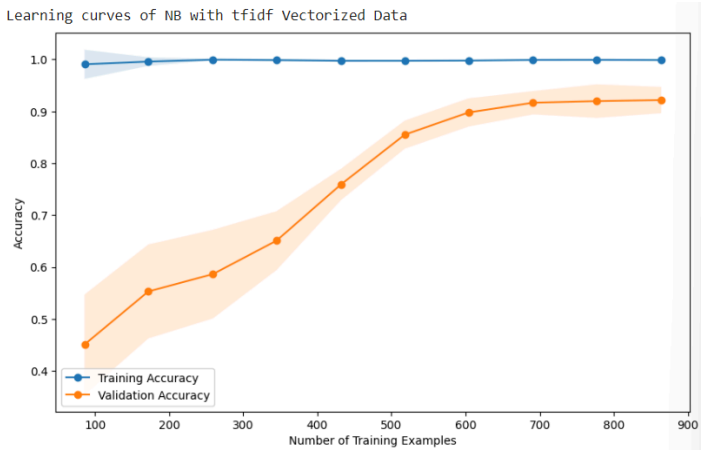

Learning curves of RF with ngram Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy

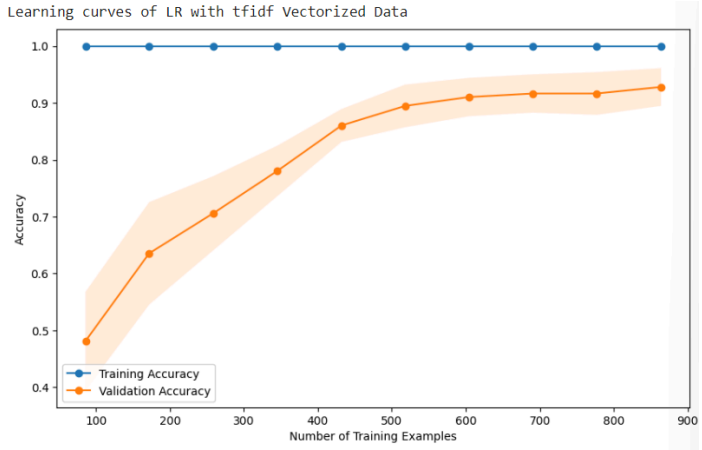

Test Accuracy score with ngram
Accuracy score
folds
KNN
NB
LR
DT
RF
1st Fold
2nd Fold
3rd Fold
4th Fold
5th Fold
6th Fold
7th Fold
8th Fold
9th Fold
10th Fold

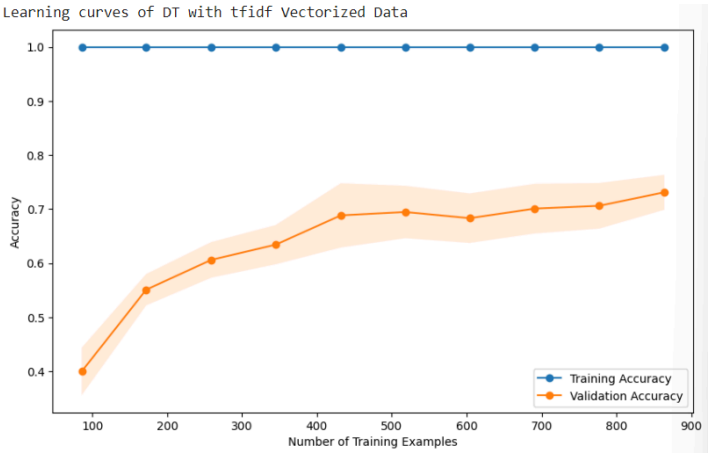


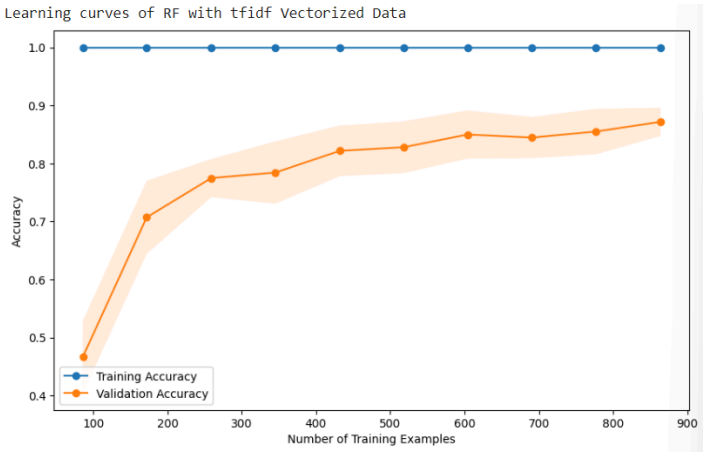
- **TF-IDF-Vectorized Data:** We can see that for TF-IDF-grams-Vectorized data, the KNN was able to learn from the data in contrast to when it was used with BOW-vectorized and N-gram-vectorized data, but the model needs more examples to converge. NB, LR, DT, and RF were able to effectively capture the model complexity and learn from the data. However, there is a significant gap between the training and validation accuracies when using LR, DT, and RF indicating overfitting and poor generalization. This gap is much smaller for NB indicating good generalization. Based on the evolution of the testing F1 scores and testing accuracies across the 10 folds, we realize that KNN and DT are poor classifiers for TF-IDF-vectorized data while NB, LR, and RF performed much better. All in all, the NB classifier is the best for TF-IDF-vectorized data as it captures the problem complexity, learns well from the data without overfitting, and generalizes well.

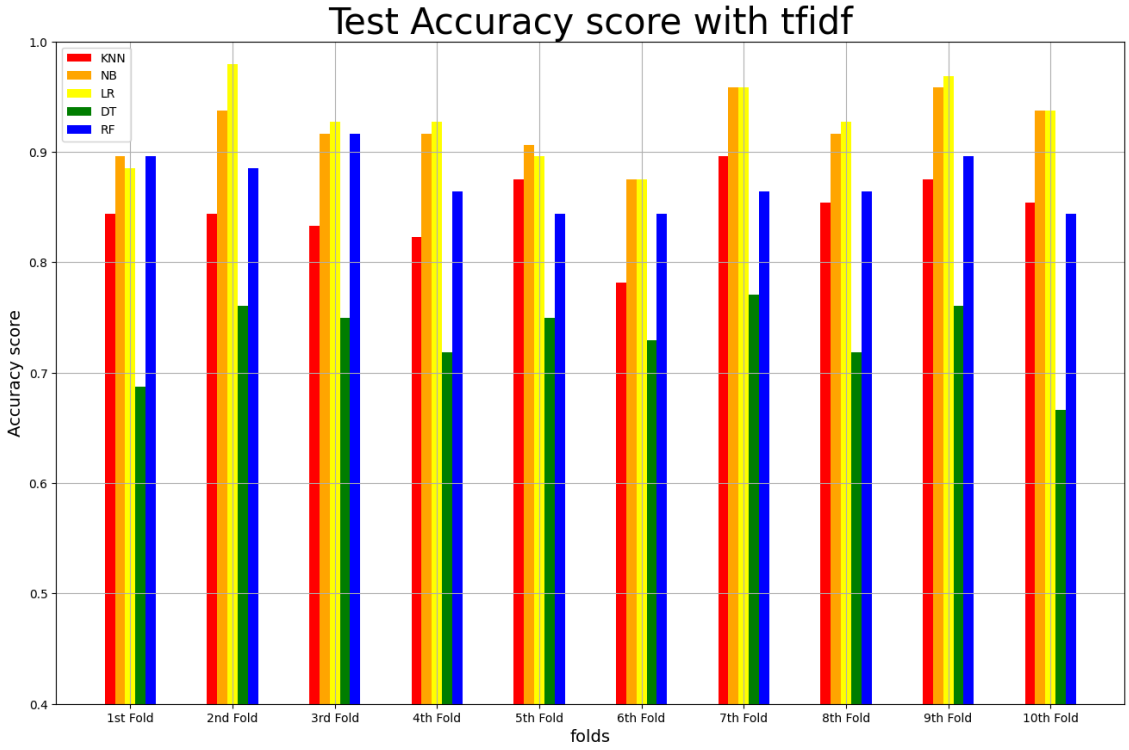


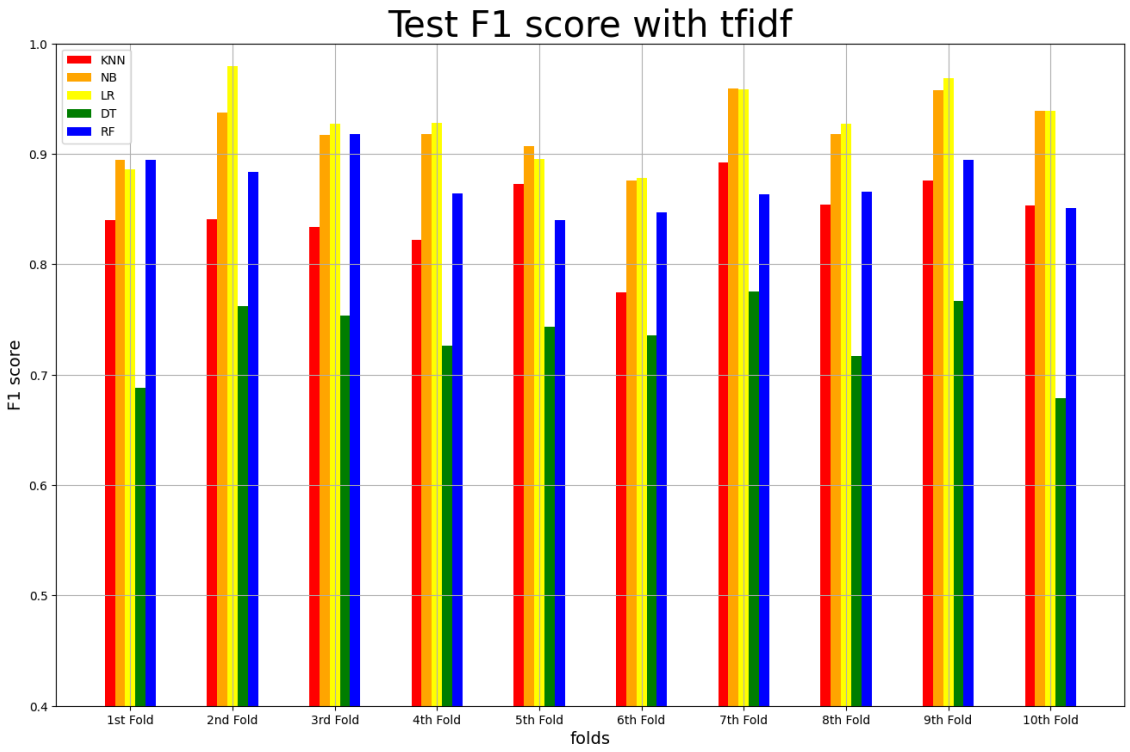

Learning curves of NB with tfidf Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy


Learning curves of LR with tfidf Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy


Learning curves of DT with tfidf Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy


Learning curves of RF with tfidf Vectorized Data
Accuracy
Number of Training Examples
Training Accuracy
Validation Accuracy


Test Accuracy score with tfidf
KNN
NB
LR
DT
RF
Accuracy score
folds
1st Fold
2nd Fold
3rd Fold
4th Fold
5th Fold
6th Fold
7th Fold
8th Fold
9th Fold
10th Fold



8. Champion Model

We summarized the mean testing accuracies and F1 scores across the 10-folds for each of the vectorization-classification combinations in the tables below:

- **Mean Testing Accuracies:**

	BOW	Ngram	TFIDF
KNN	61.145833	48.854167	84.791667
NB	92.604167	93.229167	92.187500
LR	89.583333	89.895833	92.812500
DT	71.562500	71.875000	73.125000
RF	86.666667	86.666667	87.187500

- **Mean Testing F1 Scores:**

	BOW	Ngram	TFIDF
KNN	60.616068	46.896071	84.599539
NB	92.640425	93.262244	92.242051
LR	89.677588	90.058627	92.880227
DT	72.133124	72.277736	73.463856
RF	86.914479	86.998301	87.230002

As we can see, the champion model is N-grams vectorization with Multinomial Naive Bayes Classifier as it achieved the highest accuracy at 93.23% and the highest F1 score at 93.26%. It is a well-performing model with a good balance between precision and recall, i.e. it is able to make accurate positive predictions while minimizing false positives and false negatives.

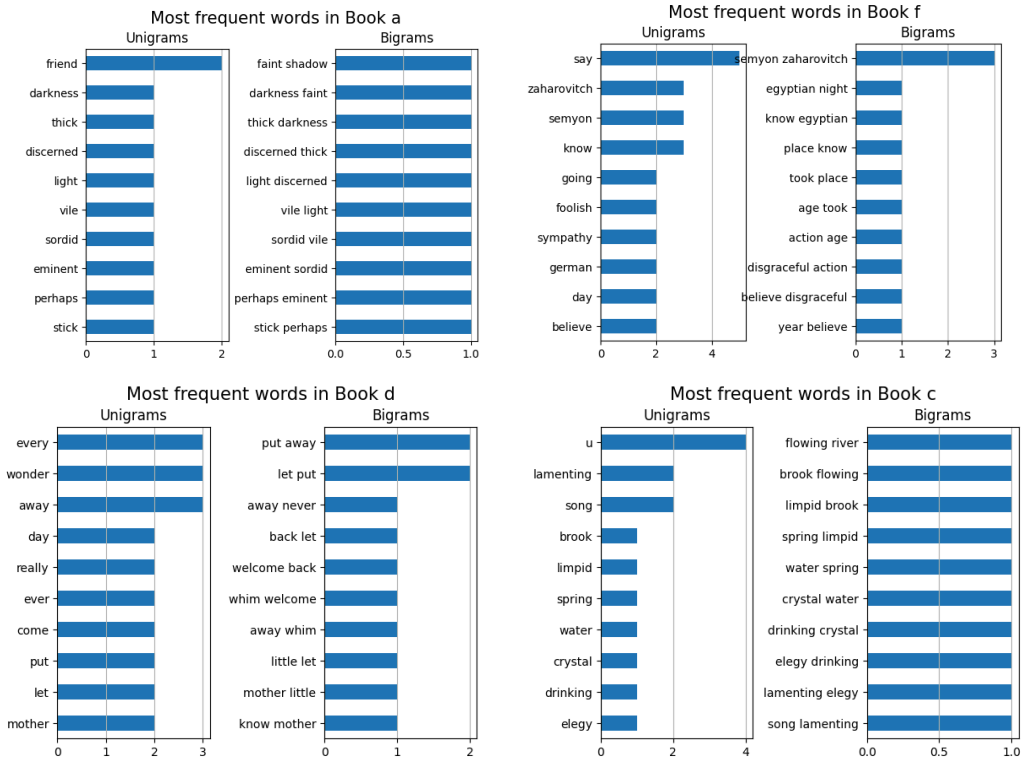
Also, the bias and variance loss for the champion model has been computed which was the lowest among every other model trained with other vectorization methods.

```
nb= MultinomialNB(alpha=1.0)
_, bias_ngram_nb, var_ngram_nb = bias_variance_decomp(nb, X_train_ngram.values, y_train_enc, X_test_ngram.values, y_test_enc, loss='0-1_loss', random_seed=42)
print('Bias of NB Classifier with N-gram vectorized data: {} \nVariance of NB Classifier with N-gram vectorized data: {}'.format(bias_ngram_nb,var_ngram_nb))
```

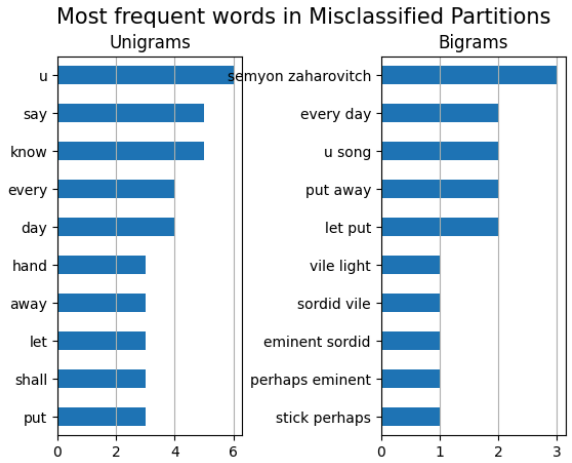
```
Bias of NB Classifier with N-gram vectorized data: 0.05416666666666667
Variance of NB Classifier with N-gram vectorized data: 0.0496458333333333
```

9. Error Analysis

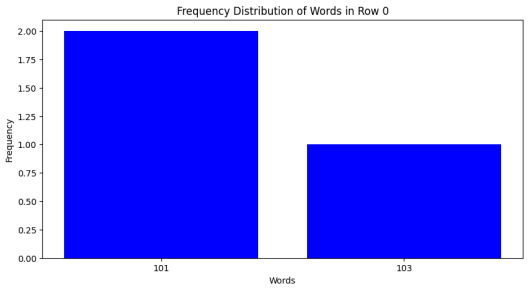
To do error analysis, the following techniques were used: grouping misclassified data and creating a frequency plot to identify patterns in the model's misclassifications. The graphic depicts a unigram and bigram of the most frequently occurring word in each book.

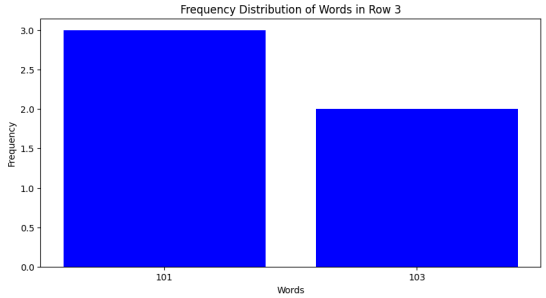


The overall final plots are shown below



From the plots, there appears to be no recurring pattern that appears to throw the model off. Diving deeper into the misclassified data by taking in number embedding data there appears to be a pattern

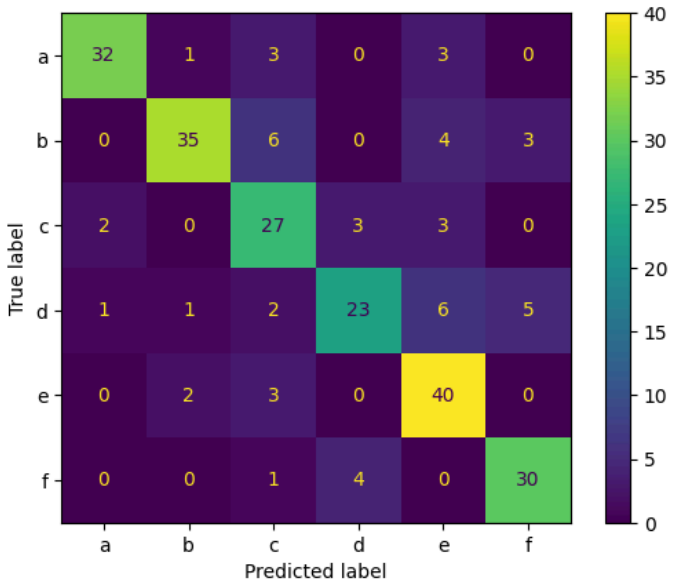




The majority of the misclassified data appears to contain a number as input, which threw the model off, as seen by the two plots above, shown in the samples of the first two rows that the model misclassifies.

10. Breaking down Champion model- decreasing accuracy

In this part, we attempt to tear down the champion model by diminishing its accuracy. The technique taken here is to introduce a new group of books by the same authors across various genres. The new books are scrapped and prepared in the same manner as the data collection process described above. The data is then lemmatized, and the partitions are converted using the ngram embedding vector. The champion model is then fed with this new data, resulting in prediction results that are contrasted to the actual outcomes, and a confusion matrix is presented below.



The confusion matrix shows that proper classification decreases along the diagonal while misclassified data increases. To further understand the lower accuracy, a classification report is created, as seen below.

	precision	recall	f1-score	support
0	0.91	0.82	0.86	39
1	0.90	0.73	0.80	48
2	0.64	0.77	0.70	35
3	0.77	0.61	0.68	38
4	0.71	0.89	0.79	45
5	0.79	0.86	0.82	35
accuracy			0.78	240
macro avg	0.79	0.78	0.78	240
weighted avg	0.79	0.78	0.78	240

The classification report shows that the f1 score, as well as the recall and precision scores, have all decreased. This decrease in f1 score accounts for around a 20% decline from the f1 score achieved by the champion model using the training data. We were successful in breaking down the accuracy of the champion model using this method.