TEXT CLASSIFICATION

Group 6

Testing Different Text Classifications Algorithms on multiple different books to find best text classifier for the selected books

Team:

- Esraa Badawi
- Esraa El-kot
- Salma Sultan
- Sondos Ali

Table of Contents

Books Selection 3

Data exploration and visualization 3

Visualizing Books text 3

Uni-grams, Bi-grams and word cloud Visualizations 4

Book 1 Partitions graphs 5

Book 2 partitions graphs 5

Book 3 partitions graphs 6

Book 4 partitions graphs 6

Book 5 partitions graphs 7

Preprocessing and Data Cleansing 7

Feature Engineering 8

Feature Extraction and Transformation: 8

BOW 8

TF/IDF 9

Stemming 9

Lemmatization: 10

Feature selection 12

Selecting top features from BOW 12

Using N-Grams 13

Text Classification Models 13

Models 13

SVM 13

Decision Tree 14

K-Nearest Neighbor 15

Naive Bias 15

Models comparison 16

Champion Model Analysis 16

Visualizing Incorrect predictions 17

Feature Importance 18

Decreasing Model Accuracy 18

Removing Selected Features 18
Testing Stemming or Lemmatizing books Partitions 19
Conclusion 20

Books Selection

A group of five books were selected from Gutenberg library, each book had a different author but all of them fall under detective and mystery stories category, the labels are used to identify the books partitions throughout the analysis and modeling.

Book name	Author	Book Label
Murder in the Gunroom	H. BEAM PIPER	a
Dead Men Tell No Tales	E. W. Hornung	b
Fire-Tongue	Sax Rohmer	c
The Murder on the Links	Agatha Christie	d
The Crystal Stopper	Maurice Leblanc	e

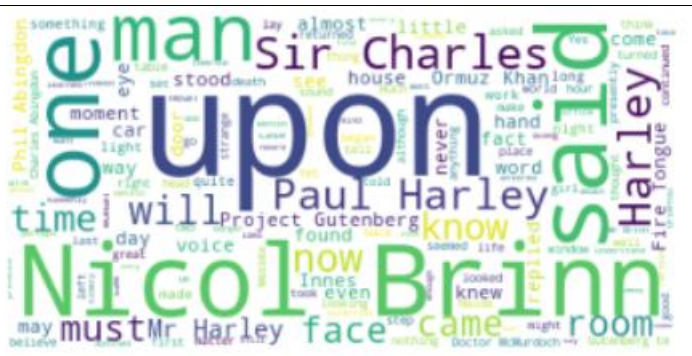
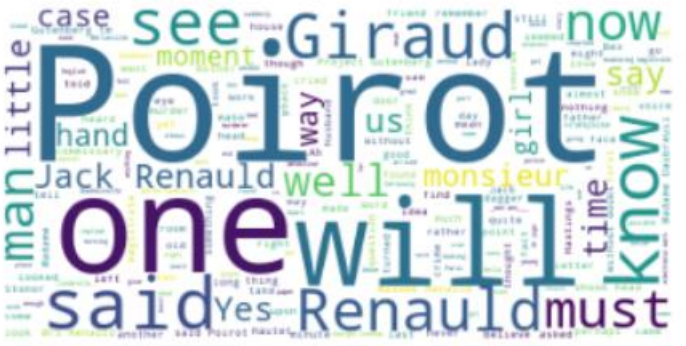
Data exploration and visualization

N-grams plots display most frequent words or words combinations in each text. And word cloud highlights significant textual data points. Both of these plots give an overall understanding of the partitions of each book

Visualizing Books text

The following word clouds are for every selected book.

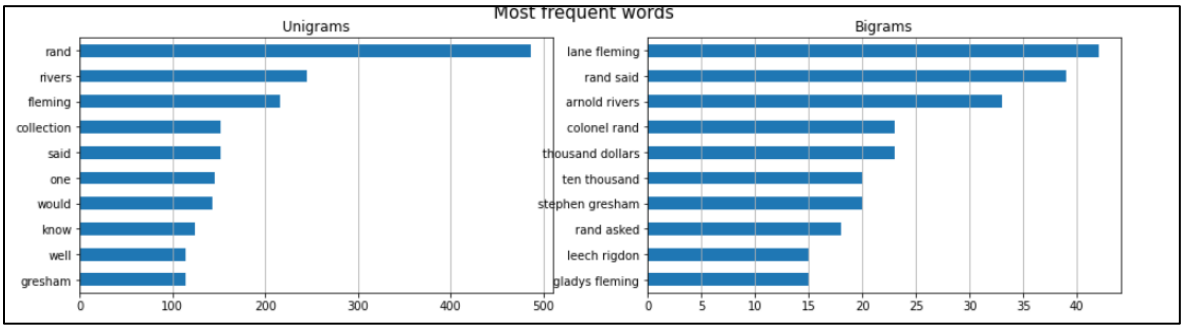
Book Label	Word cloud
a	
b	

c	
d	
e	

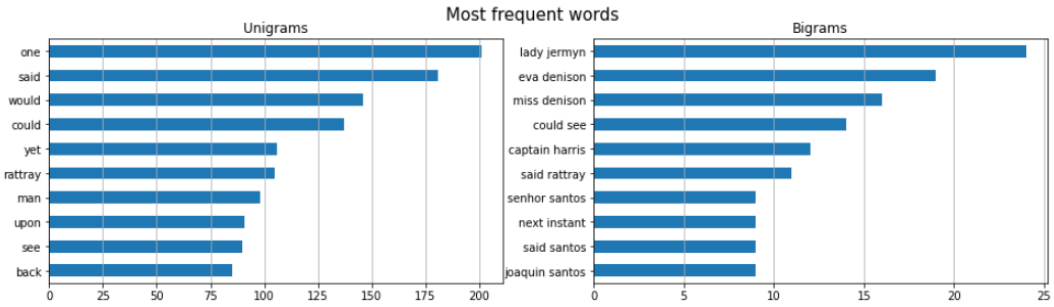
Uni-grams, Bi-grams and word cloud Visualizations

The following graphs are for every book's partitions.

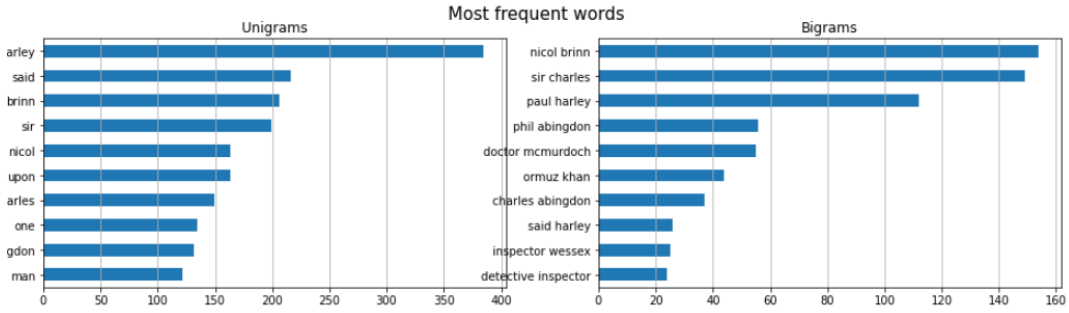
Book 1 Partitions graphs



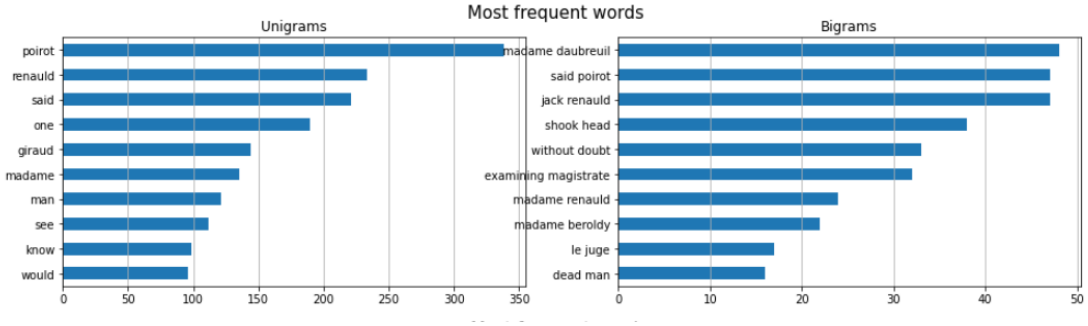
Book 2 partitions graphs

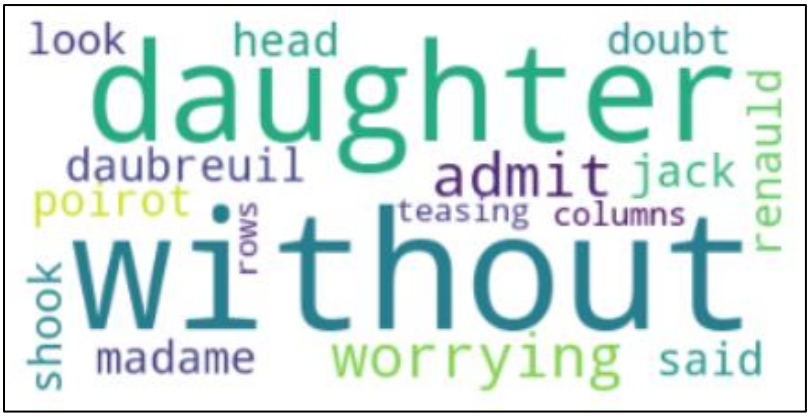


Book 3 partitions graphs

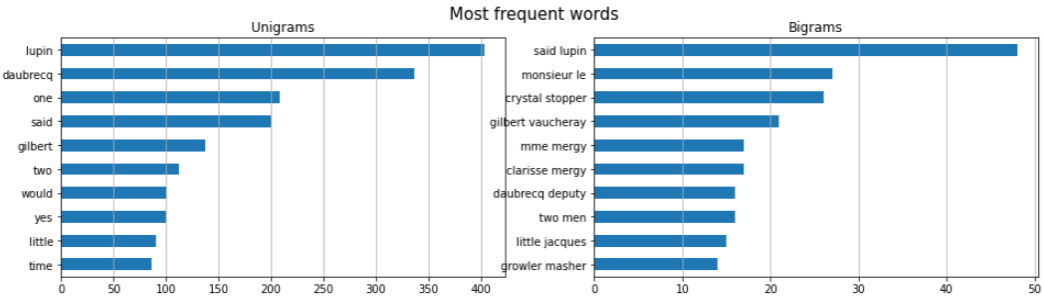


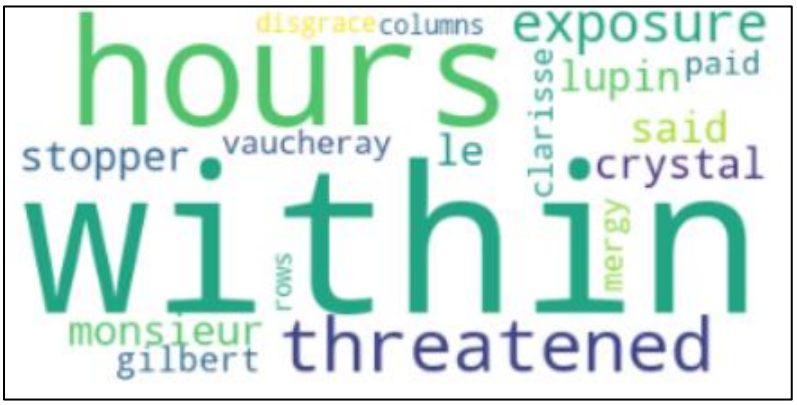
Book 4 partitions graphs





Book 5 partitions graphs





We can see that the books words are similar to each other and that every books main characters names is repeated frequently.

Preprocessing and Data Cleansing

The following steps were followed to transform the raw data into useful and efficient format.

1. The books were downloaded from Gutenberg online library, the extra padding added by Gutenburg library which included copyrights information was removed and only the book text was retrieved.
2. The book's sentences were tokenized using nltk word tokenizer.
3. Stop words and punctuation marks were removed, so that models can focus on unique information that can be used for classification.

```
stop_words = set(stopwords.words('english'))
book_words = word_tokenize(book_txt)
filtered_book_words = book_words
filtered_book_words = [w for w in filtered_book_words if not w.lower() in stop_words]
```

```
filtered_book_words=[word.lower() for word in filtered_book_words if word.isalpha()]
```

4. The first 100 words were skipped to avoid getting cover page, table of contents, and introduction chapter text in the selected books partitions which would not have helped

in classifying the book. It is an important step at data cleaning because they are unnecessary or useful input for the model and can affect the performance.

5. 200 book partitions were acquired from all the books, each books partitions had a unique label and all of the partitions were added to a dataframe to facilitate further processing, the books partitions were shuffled to avoid model classifying using books partitions sequential index.

	Partition	Label
0	[corrected, staying, shall, meet, flashed, smi...	d
1	[ago, rand, said, stephen, gotten, cased, duel...	a
2	[nobody, else, wants, rand, intended, collecti...	a
3	[coldly, one, accused, yet, well, answer, ques...	d
4	[noticed, little, touch, yeah, clean, gat, gen...	a
...	...	...
995	[low, tone, hear, said, recognized, language, ...	d
996	[start, police, searches, numerous, time, ten,...	e
997	[promised, find, cousin, benefactor, elder, si...	e
998	[placed, policemen, gate, prevent, one, passin...	e
999	[may, give, pain, renauld, understand, present...	d

Figure 1- Final books dataframe sample

Feature Engineering

Feature Extraction and Transformation:

BOW

Bag of words is used to transform raw text partitions to a numerical format using words occurrences counts. CountVectorizer was used to model bag of words for books text partitions and transform partitions into numerical vectors.

	aarvo	aback	abaft	abandon	abandoned	abandoning	abashed	abbalac	abducting	aberystwyth	...	youngish	youngster	youth	youths	zambesi	zeal	zealous	zigzag	zone	zorger
Part0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Part995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

1000 rows × 12549 columns

Figure 2-A sample of the generated BOW

TF/IDF

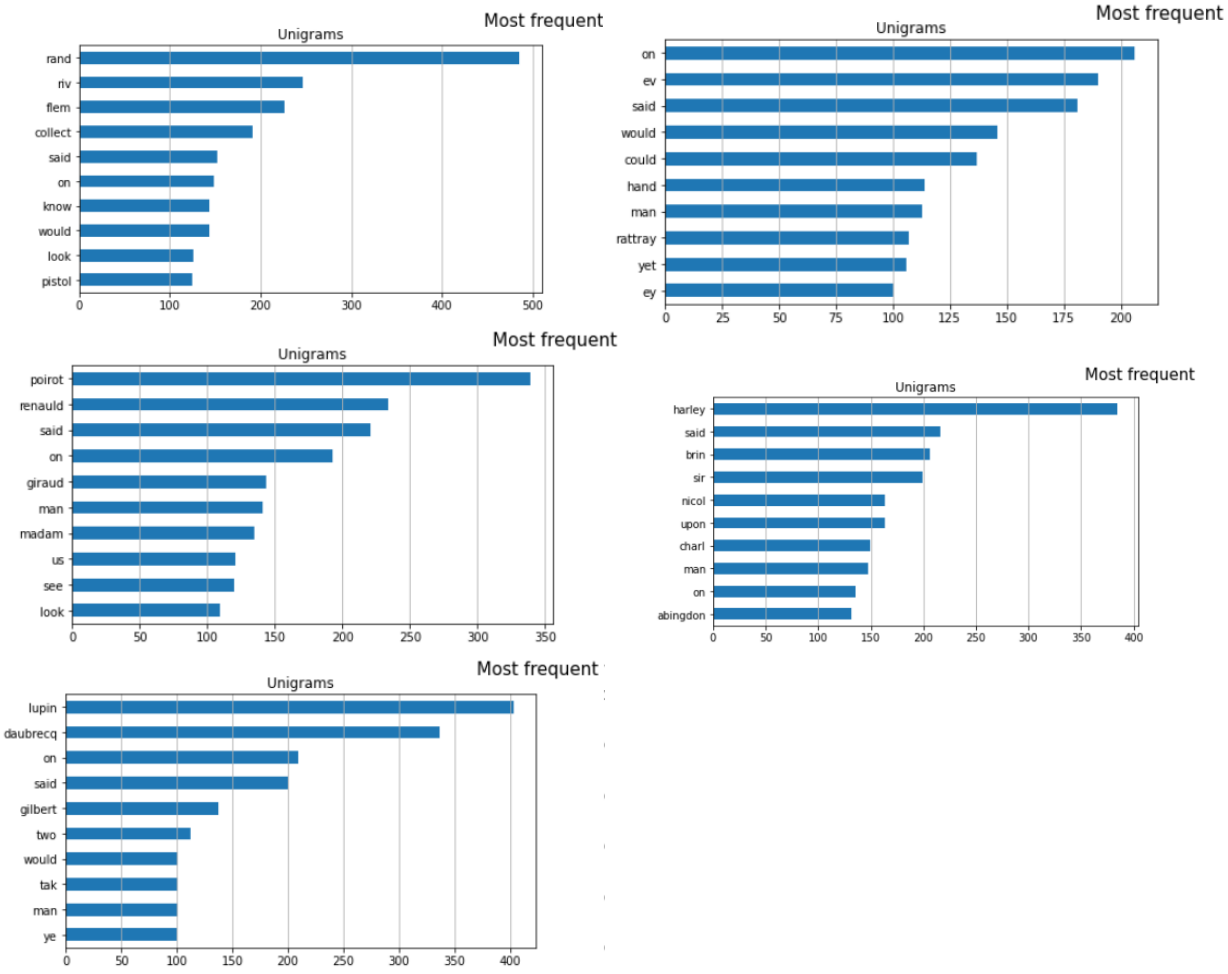
TF/IDF gives higher score for more important words and lower score for less important words. The TF-IDF score was calculated for each word in the text using TfidfTransformer.

```
abashed         7.215608
abducting       7.215608
aberystwyth     7.215608
abhorred        7.215608
abject          7.215608
                   ...
consult         7.215608
consumed        7.215608
consuming       7.215608
consummated     7.215608
consumption     7.215608
Name: idf_weights, Length: 1000, dtype: float64
```

Figure 3-a sample of the TF/IDF scores

Stemming

The words were stemmed to their roots by using LancasterStemmer stemmer from nltk, but it is not accurate because it can make a mistake in returning the word to its origin. After stemming, the most frequent words in stemmed books partitioned were plotted by using uni-gram.



It is apparent that some words became gibberish and not correct English words, which is why this method was not used in in processing the data used to train and test models.

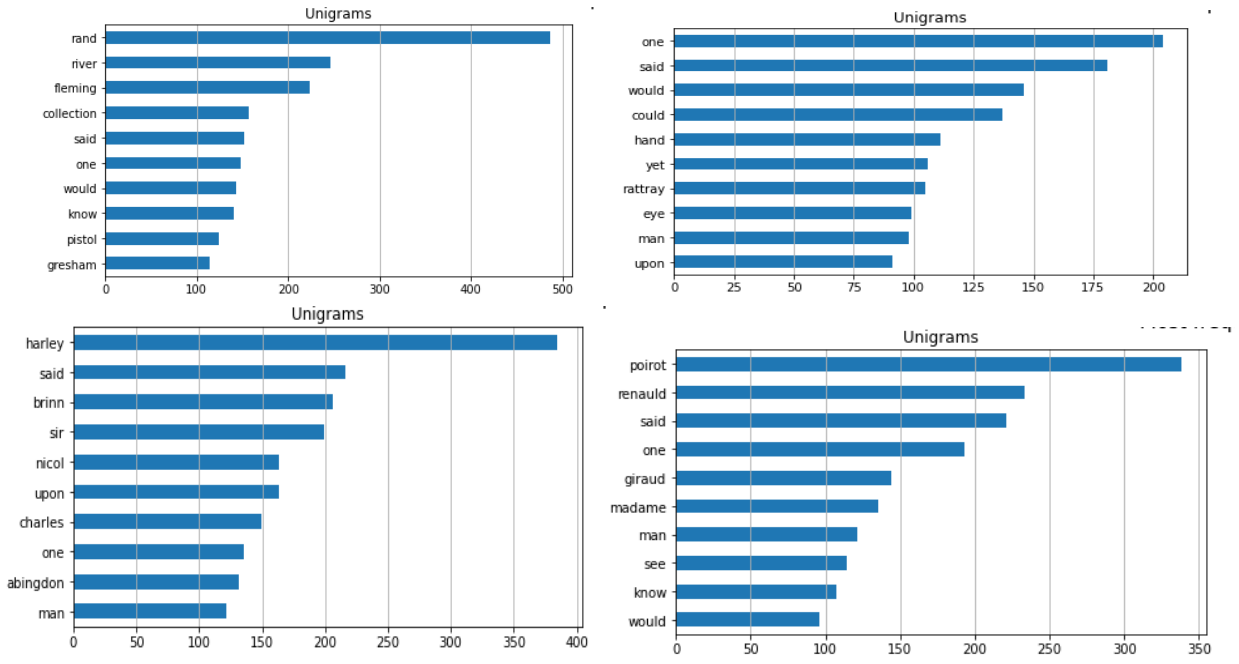
Lemmatization:

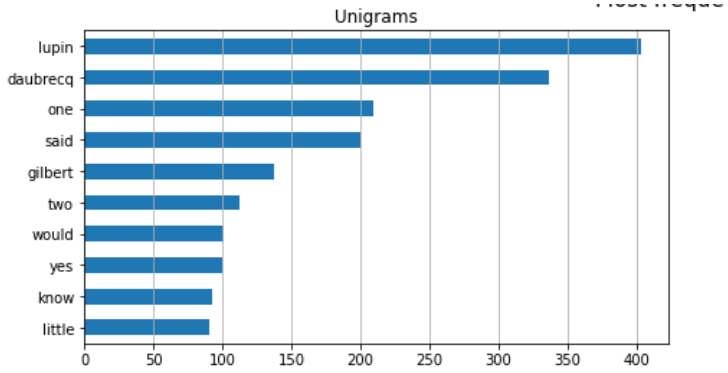
Lemmatization is like stemming but it produces much better results, WordNetLemmatizer from nltk was used to get words roots. But from the following graphs we can see that it did not manage to get many words roots successfully because part of speech was not sent to the lemmatizer.

Lemm Partition
corrected staying shall meet flashed smile au ...
ago rand said stephen gotten cased dueling set...
nobody else want rand intended collection well...
coldly one accused yet well answer question wi...
noticed little touch yeah clean gat generally ...
...
low tone hear said recognized language bastard...
start police search numerous time ten month ag...
promised find cousin benefactor elder sister e...
placed policeman gate prevent one passing with...
may give pain renauld understand present mothe...

Figure 4-A sample of the lemmatized books partitions

The most frequent words in lemmatized books partitioned were plotted by using uni-gram.





Since the lemmatizer did not change many words, it was not used in in processing the data used to train and test models.

Feature selection

Selecting top features from BOW

To reduce the number of features, top features were selected from BOW by specifying that min_df equals 50, meaning that we ignore terms that appear in less than 50 of the partitions.

	abingdon	able	across	ago	ah	almost	along	already	also	always	...	woman	word	words	work	world	would	years	yes	yet	young
Part0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part1	0	0	0	1	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	2	0	2	0	0
Part3	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	1	0
Part4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	5	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Part995	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
Part996	0	0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
Part997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Part998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
Part999	0	0	0	0	0	0	0	1	0	1	...	0	0	0	0	0	4	0	0	0	0

1000 rows × 300 columns

Figure 5-A sample of the selected BOW features

Using N-Grams

N-Grams were used with CountVectorizer to include bi-grams as well as uni-grams to the BOW's vocabulary by specifying ngram_range to (1,2), To limit the number of features of the BOW model we used min_df equal to 30 which means that single words and bi grams were ignored if they appeared in less than 30 of the documents.

	arnold rivers	charles abingdon	could see	doctor mcmurdoch	lane fleming	last night	madame daubreuil	nicol brinn	ormuz khan	paul harley	phil abingdon	rand said	said lupin	said poirot	shook head	sir charles	without doubt
Part0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Part1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Part2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Part3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Part4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Part995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Part996	0	0	0	0	0	0	0	1	2	0	1	0	0	0	0	0	0
Part997	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Part998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Part999	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

1000 rows × 17 columns

Figure 6-A sample of the bi grams found in the books text paritions

Text Classification Models

The following steps were followed in the modelling stage:

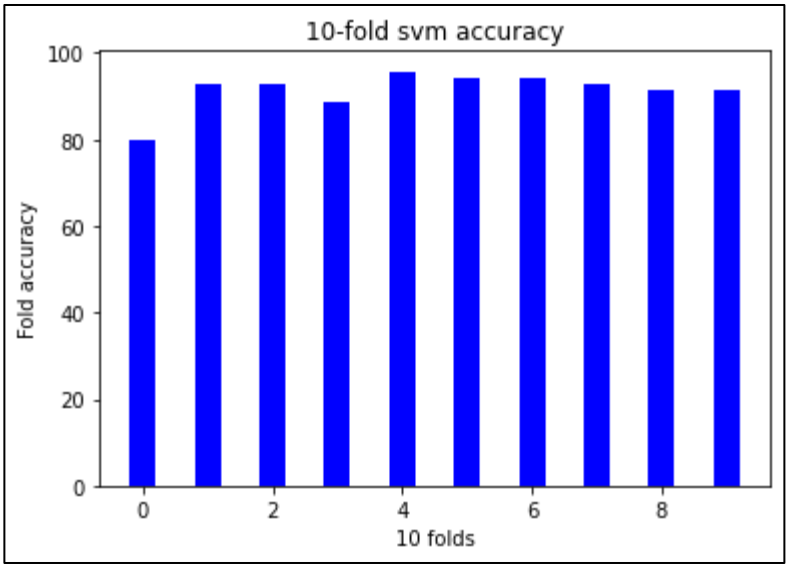
1. The data was divided into train and test with ratios of 70% and 30% respectively.
2. Four different models were selected to test books classifications, each classifier was tested using a different pipeline- pipelines facilitate the fitting and prediction of the models by applying all the operations in the pipeline automatically on the input data-. The selected classifiers were:
   - SVM
   - Decision tree
   - K nearest neighbor
   - Naïve Bias
3. 10-Folds cross validation was applied for each model and folds errors were plotted.

Models

SVM

The pipeline for svm consists of count vectorizer with min_df equal to 50, TF-IDF transformer and the model itself. Svm model 10-folds cross validation was with accuracy of **(91%)** with **(0.08)** standard deviation.

We calculated the average expected loss, bias and variance for each model. The Average expected loss for SVM is **(0.381),** bias **(0.307)** and variance **(0.075).**

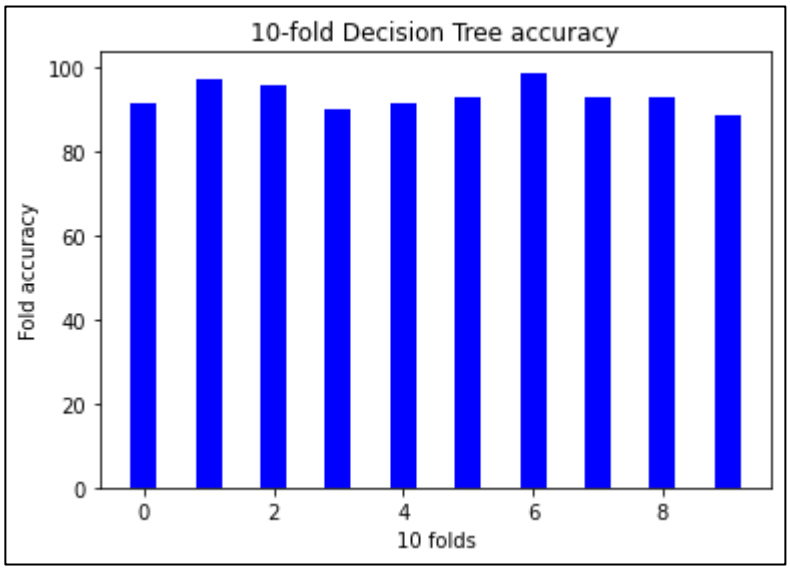


Decision Tree

The pipeline for decision tree consists of count vectorizer -with min_df equal to 30 and ngram_range equal to (1,2)-, TF-IDF transformer and the model itself.

Decision Tree model 10-folds cross validation was with accuracy of **(92%)** with **(0.07)** standard deviation.

The Average expected loss for decision tree is **(0.308),** bias **(0.167)** and variance **(0.142).**

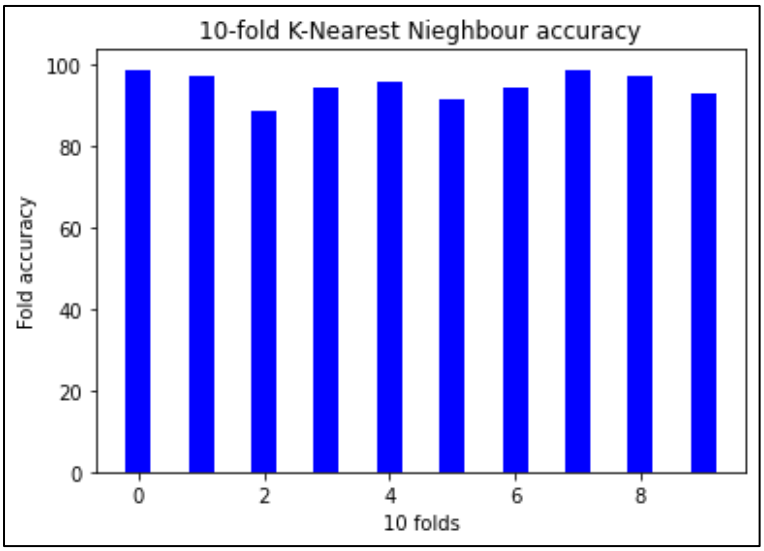


K-Nearest Neighbor

The pipeline for K-Nearest Neighbour consists of count vectorizer -with min_df equal to 30 and ngram_range equal to (1,2)-, TF-IDF transformer and the model itself.

K nearest neighbor model 10-folds cross validation was with accuracy of **(95%)** with **(0.04)** standard deviation.

The Average expected loss for K nearest neighbor is **(0.376),** bias **(0.187)** and variance **(0.189).**

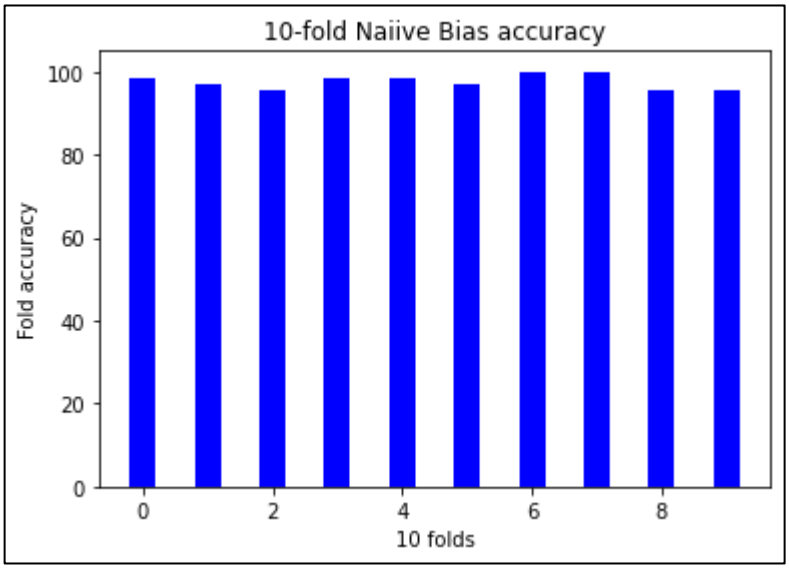


Naive Bias

The pipeline for Naïve Bias consists of count vectorizer -with min_df equal to 30 and ngram_range equal to (1,2)-, TF-IDF transformer and the model itself.

Naïve bias model 10-folds cross validation was with accuracy of **(97%)** with **(0.04)** standard deviation.

The Average expected loss for Naïve bias is **(0.132),** bias **(0.073)** and variance **(0.058).**



Models comparison

Table 1-Errors of different models

Model	10-Folds error	Variance	Bias
SVM	91%	0.184	0.049
Decision Tree	94%	0.130	0.089
KNN	95%	0.184	0.116
Naïve Bias	98%	0.066	0.086

Regarding to the average expected loss, bias and variance: the champion model is **Naïve bias**. As it has the lowest error and variance which means there is no over fitting and the model is well trained.

Champion Model Analysis

naïve Bias model was evaluated using test data and the confusion matrix was calculated to determine incorrect predictions.

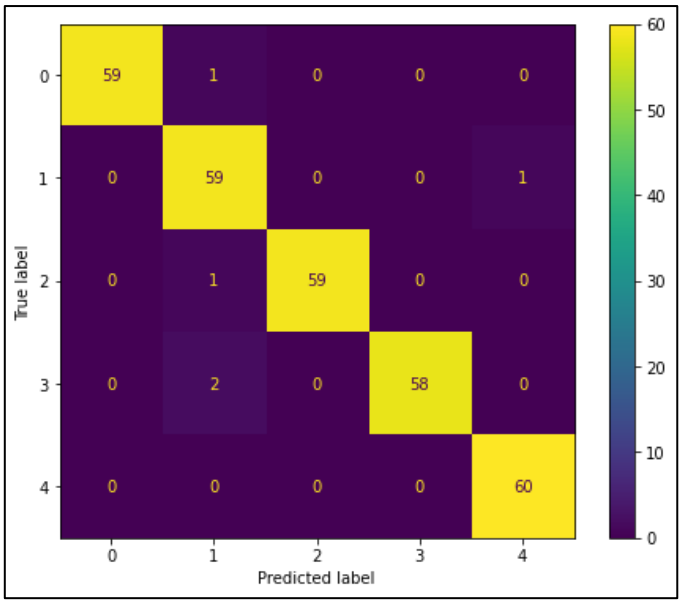


Figure 7-Confusion matrix for Naive Bias classifier

Visualizing Incorrect predictions

The incorrect predications were visualized using following word cloud.

Figure 8-A word cloud to visualize incorrect predictions text

Feature Importance

The most important features in each book given by the classifier were the following:

```
Book 'a' Important Features:['rand' 'rivers' 'fleming' 'pistols' 'collection' 'gresham' 'would' 'one'
 'well' 'like']

Book 'b' Important Features:['would' 'one' 'yet' 'said' 'could' 'never' 'still' 'upon' 'back' 'less']

Book 'c' Important Features:['harley' 'brinn' 'sir' 'nicol' 'nicol brinn' 'abingdon' 'sir charles'
 'charles' 'upon' 'paul']

Book 'd' Important Features:['poirot' 'renauld' 'madame' 'giraud' 'said' 'magistrate' 'villa' 'one'
 'monsieur' 'crime']

Book 'e' Important Features:['lupin' 'daubrecq' 'gilbert' 'one' 'clarisse' 'said' 'two' 'yes' 'shall'
 'thing']
```

Figure 9-List of important features per book according to Naive Bias classifier

Decreasing Model Accuracy

Removing Selected Features

From the features importance given by the model, It was noticed that multiple main characters names were used in correctly classifying books partitions. So, all the names that were found important by the classifier were removed from the BOW model transformer which is used to vectorize the data before passing it to the model for prediction. So, the model got confused and was not able to determine accurately the labels as it used to do before removing main characters names from BOW.

The names were removed from the BOW and not from the books partitions to avoid changing the data used throughout the assignment.

The Accuracy dropped to **0.87 (+/- 0.10)**

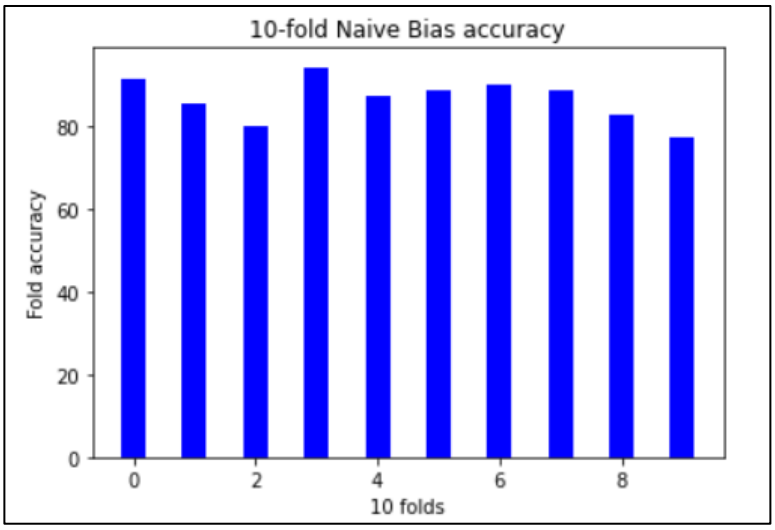


Figure 10- 10-Folds Naive Bias classifier accuracy after removing main characters names from BOW

Testing Stemming or Lemmatizing books Partitions

we applied some more feature engineering like stemming or lemmatization of books partitions. Feed them to the model to see how will they affect the model accuracy.

Accuracy(stemming): **0.97 (+/- 0.02)** – did not affect the accuracy.

Accuracy(lemmatization): **0.98 (+/- 0.03)** – raised the accuracy by **1%.**

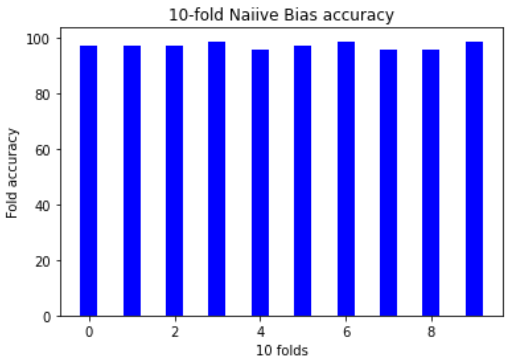


Figure 11 – 10-Folds Naïve Bais classifier accuracy using stemmed books partitions

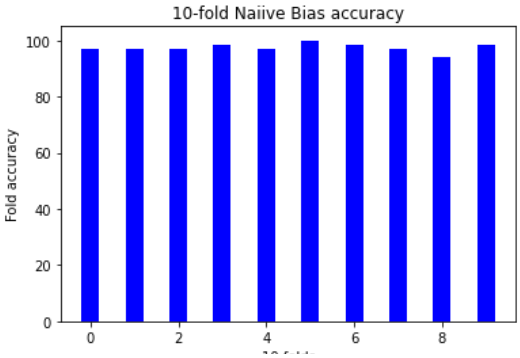


Figure 12 10-Folds Naïve Bais classifier accuracy using lemmatized books partitions

Conclusion

The objective of the assignment was to explore different text classification models using different feature extraction techniques. Inspect the results and choose the champion model then do further analysis to understand why the champion model was performing well on the collected books partitions corpus.

Five books in the detective and mystery stories category were selected to create the text partitions that will be used throughout the assignment. Different combinations of feature extraction techniques and classification models were used. The selected champion model was Naïve Bais classifier using BOW – with max_df equal 30 to and ngrams_range equal to (1,2)- and TF-IDF for feature extraction. The error analysis of the champion model showed that the model was performing well because it was using the character names found in the books. Also, the stemmed partitions and lemmatized partitions were used to measure their effect on the model's accuracy but they had little to no effect on the accuracy.