NLP Classification Project

Assignment One

Mostafa Mohamed Mahmoud
Mohamed Fouad Abdelnaby
Sameh Mohamed Ismail
Mohamed Helmy Mostafa

University of Ottawa

Goal:

This assignment aims to implement classification strategy for sample (five) different book and sample (five) different authors and Separate and set aside unbiased random partitions for training, validation and testing to produce classification predictions and compare them; analyze the pros and cons of algorithms and generate and communicate the insights and to know the style for each other so when give the model any document from other book can detect the other.

Dataset:

We make search in Gutenberg digital books to collect this book to be and the same category and for different author to make a difficult to the model to predict the correct labeled data.

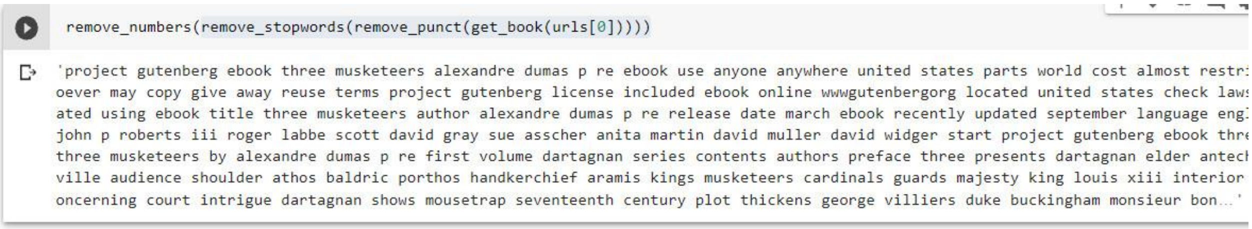
The Three Musketeers by AlexandreDumas

```
    - https://www.gutenberg.org/ebooks/1257
    - https://www.gutenberg.org/ebooks/1257.txt.utf-8
- Tarzan of the Apes by Edgar Rice Burroughs
    - https://www.gutenberg.org/ebooks/78
    - https://www.gutenberg.org/files/78/78-0.txt
- The Thirty-Nine Steps by John Buchan
    - https://www.gutenberg.org/ebooks/558
    - https://www.gutenberg.org/files/558/558-0.txt
- The Prisoner of Zenda by Anthony Hope
    - https://www.gutenberg.org/ebooks/95
    - https://www.gutenberg.org/files/95/95-0.txt
- Captain Blood by Rafael Sabatini
    - https://www.gutenberg.org/ebooks/1965
    - https://www.gutenberg.org/files/1965/1965-0.txt
### The 5 books (Adventure Genre) https://www.gutenberg.org/ebooks/bookshelf/82
```

1. Data preprocessing

Clean data:

We get the books by URL and make preparation and cleaning processes by download the stop word from NLTK (remove stop words, punctuation and numbers) and get word only to help us making classification and change all upper case word to lower case.

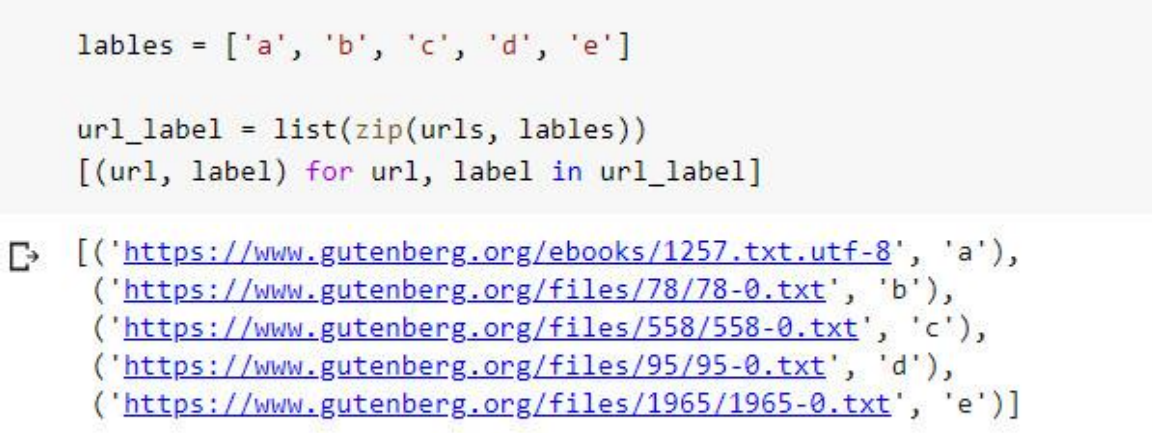


```
remove_numbers(remove_stopwords(remove_punct(get_book(urls[0]))))
```

```
'project gutenberg ebook three musketeers alexandre dumas p re ebook use anyone anywhere united states parts world cost almost restr
oever may copy give away reuse terms project gutenberg license included ebook online wwwgutenbergorg located united states check laws
ated using ebook title three musketeers author alexandre dumas p re release date march ebook recently updated september language engl
john p roberts iii roger labbe scott david gray sue asscher anita martin david muller david widger start project gutenberg ebook thre
three musketeers by alexandre dumas p re first volume dartagnan series contents authors preface three presents dartagnan elder antec
ville audience shoulder athos baldric porthos handkerchief aramis kings musketeers cardinals guards majesty king louis xiii interior
oncerning court intrigue dartagnan shows mousetrap seventeenth century plot thickens george villiers duke buckingham monsieur bon...'
```

Label data :

We make labeling for each partition in the book regarding author to could make prediction according to this lapel.

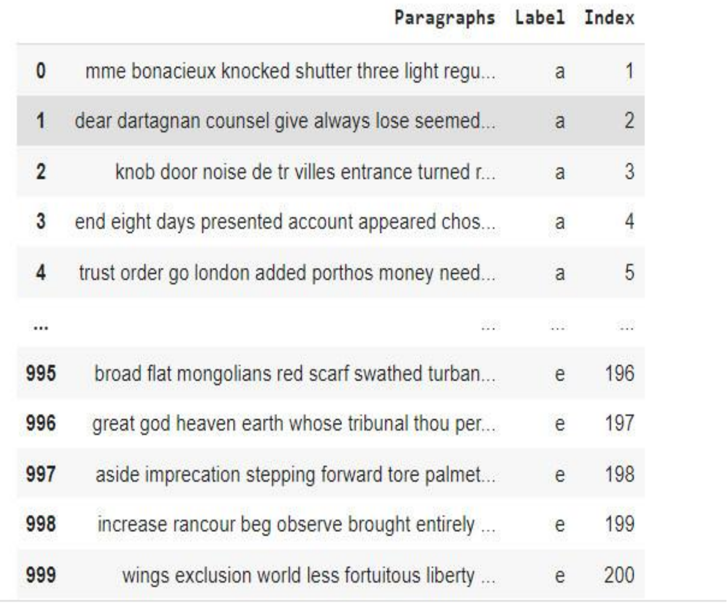


```
lables = ['a', 'b', 'c', 'd', 'e']

url_label = list(zip(urls, lables))
[(url, label) for url, label in url_label]
```

```
[('https://www.gutenberg.org/ebooks/1257.txt.utf-8', 'a'),
 ('https://www.gutenberg.org/files/78/78-0.txt', 'b'),
 ('https://www.gutenberg.org/files/558/558-0.txt', 'c'),
 ('https://www.gutenberg.org/files/95/95-0.txt', 'd'),
 ('https://www.gutenberg.org/files/1965/1965-0.txt', 'e')]
```

Partition data:

After cleaning processes, we start prepare data by splitting each book to 200 document and each document have at least 100 word.

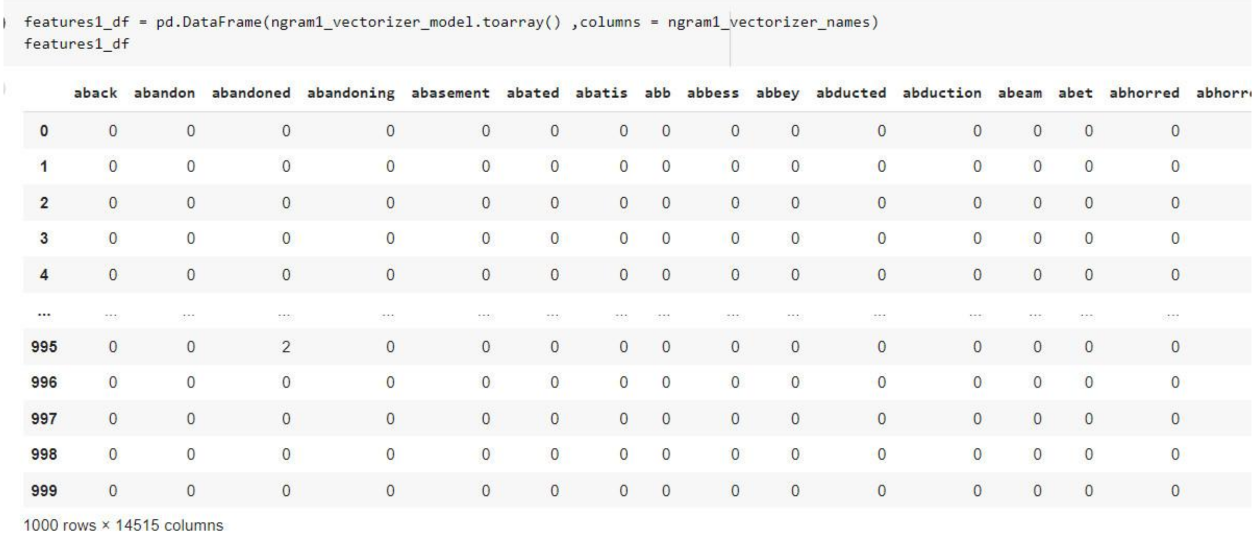


	Paragraphs	Label	Index
0	mme bonacieux knocked shutter three light regu...	a	1
1	dear dartagnan counsel give always lose seemed...	a	2
2	knob door noise de tr villes entrance turned r...	a	3
3	end eight days presented account appeared chos...	a	4
4	trust order go london added porthos money need...	a	5
...	...	...	...
995	broad flat mongolians red scarf swathed turban...	e	196
996	great god heaven earth whose tribunal thou per...	e	197
997	aside imprecation stepping forward tore palmet...	e	198
998	increase rancour beg observe brought entirely ...	e	199
999	wings exclusion world less fortuitous liberty ...	e	200

2. Text transformation

BOW (1 gram):

Make a tokenization for book could make bag of word (BOW) and can enter the document to the model to know the author of partition of written.



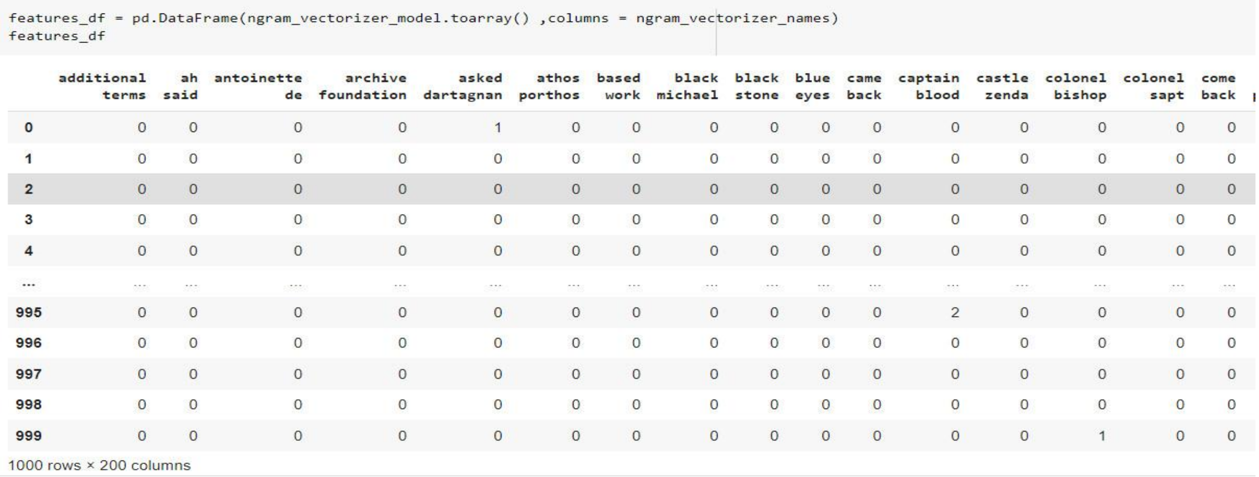
```
features1_df = pd.DataFrame(ngram1_vectorizer_model.toarray() ,columns = ngram1_vectorizer_names)
features1_df
```

	aback	abandon	abandoned	abandoning	abasement	abated	abatis	abb	abbess	abbey	abducted	abduction	abeam	abet	abhorred	abhorr
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	
996	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

1000 rows × 14515 columns

BoW + 2gram:

We apply the bag of word with 2 gram we get this matrix of data and we will enter it to model and will know the accuracy of prediction.



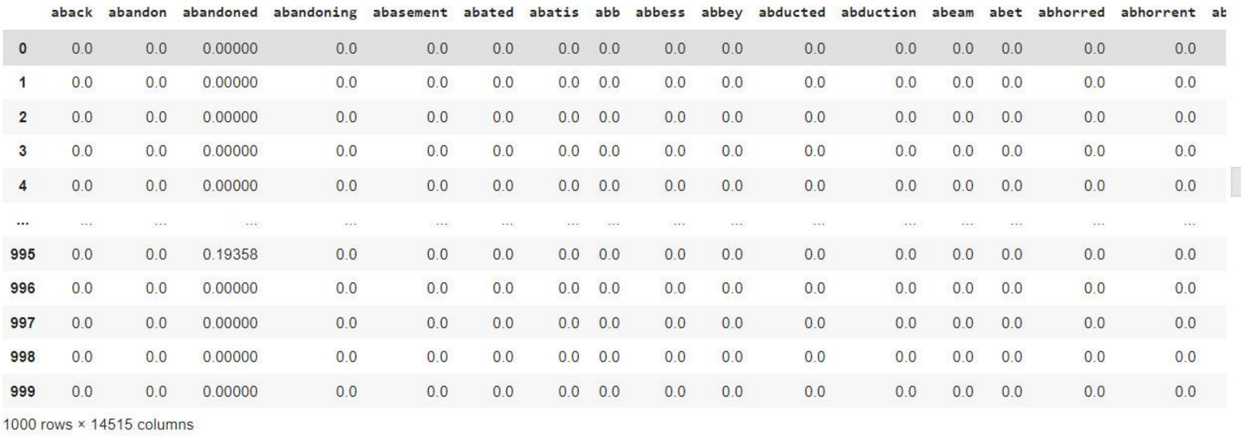
```
features_df = pd.DataFrame(ngram_vectorizer_model.toarray() ,columns = ngram_vectorizer_names)
features_df
```

	additional terms	ah said	antoinette de	archive foundation	asked dartagnan	athos porthos	based work	black michael	black stone	blue eyes	came back	captain blood	castle zenda	colonel bishop	colonel sapt	come back
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
996	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

1000 rows × 200 columns

TF-IDF:

Using tf-idf of the of document and give to the model to predict the document belong to any author



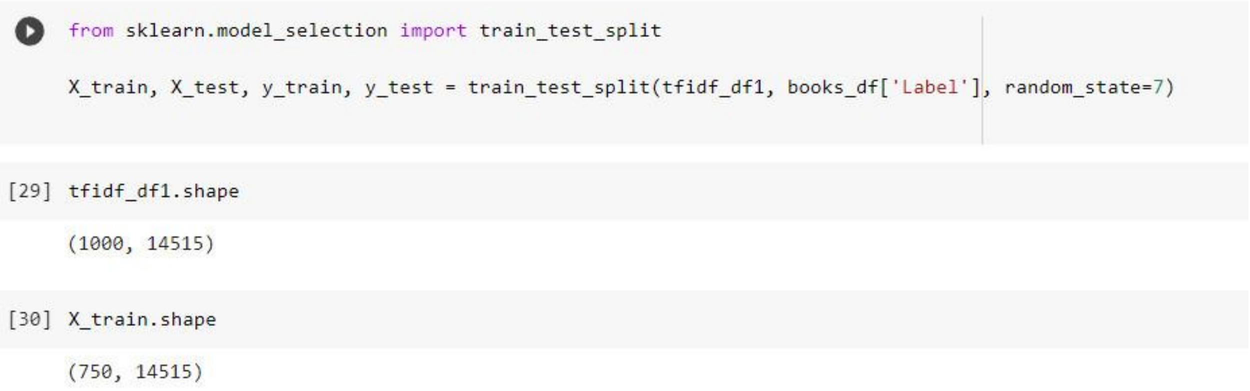
	aback	abandon	abandoned	abandoning	abasement	abated	abatis	abb	abbess	abbey	abducted	abduction	abeam	abet	abhorred	abhorrent	ab
0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
995	0.0	0.0	0.19358	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
996	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
997	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
998	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
999	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

1000 rows × 14515 columns

We make a stemming to all word in the 5 books to decrease the word we work on it and we will do the opration to the trainning data and test date that will enter the model to predict.

3. **Classification**:

We will divide the data random to training data and test data



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(tfidf_df1, books_df['Label'], random_state=7)
```

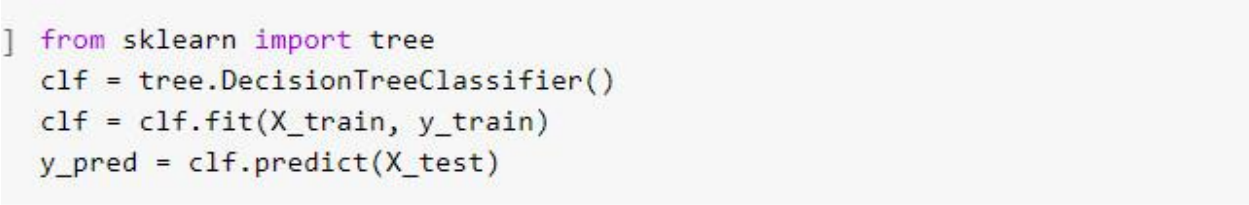
```
[29] tfidf_df1.shape

     (1000, 14515)
```

```
[30] X_train.shape

     (750, 14515)
```

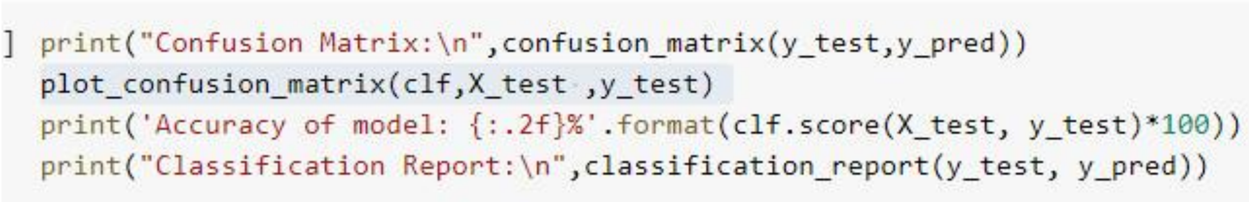
After doing all this operation we get 4 type of data about the books (TF-IDF 1 gram ,TF-IDF 2 gram ,BOW 1ngram ,BOW 2 gram). We will use the 3 model to get the prediction accuracy

TF-IDF (1 gram):

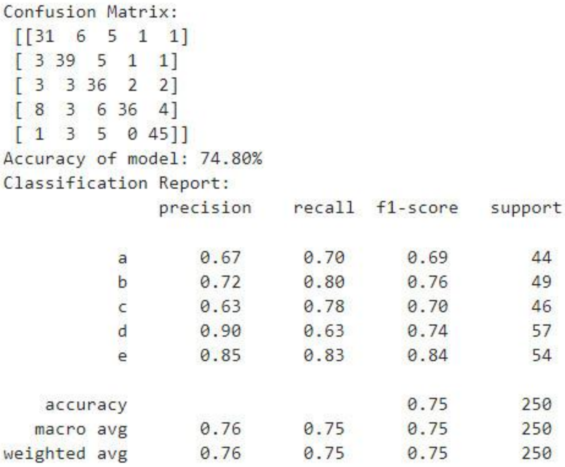
1. TF-IDF (1 gram) Decision Tree:



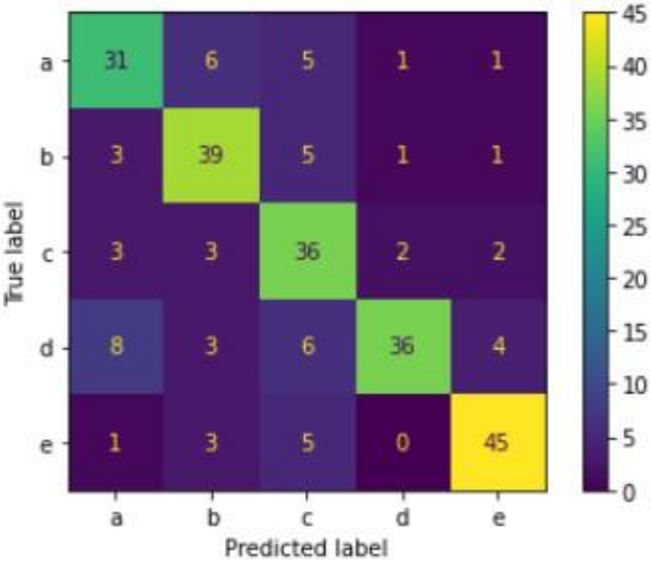
```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```



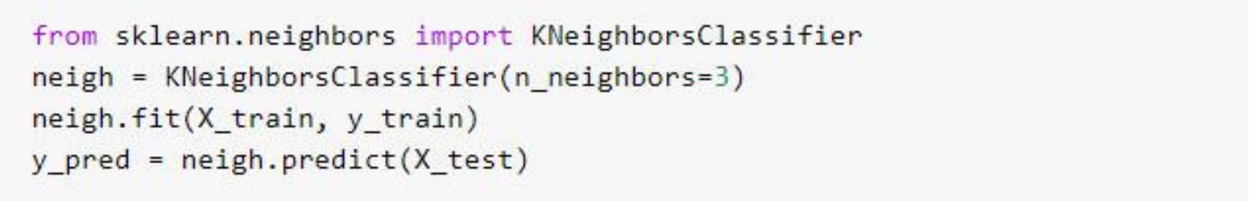
```
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
plot_confusion_matrix(clf,X_test ,y_test)
print('Accuracy of model: {:.2f}%'.format(clf.score(X_test, y_test)*100))
print("Classification Report:\n",classification_report(y_test, y_pred))
```



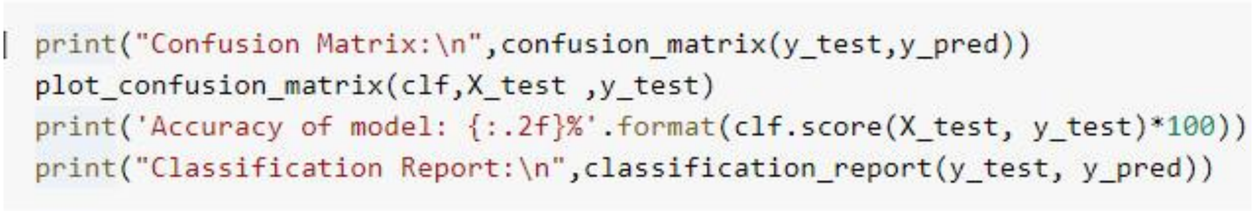
```
Confusion Matrix:
 [[31  6  5  1  1]
 [ 3 39  5  1  1]
 [ 3  3 36  2  2]
 [ 8  3  6 36  4]
 [ 1  3  5  0 45]]
Accuracy of model: 74.80%
Classification Report:
               precision    recall  f1-score   support

           a       0.67      0.70      0.69        44
           b       0.72      0.80      0.76        49
           c       0.63      0.78      0.70        46
           d       0.90      0.63      0.74        57
           e       0.85      0.83      0.84        54

    accuracy                           0.75       250
   macro avg       0.76      0.75      0.75       250
weighted avg       0.76      0.75      0.75       250
```



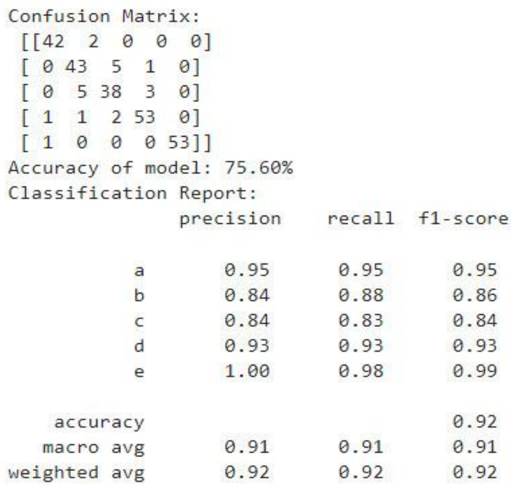
2- KNN:



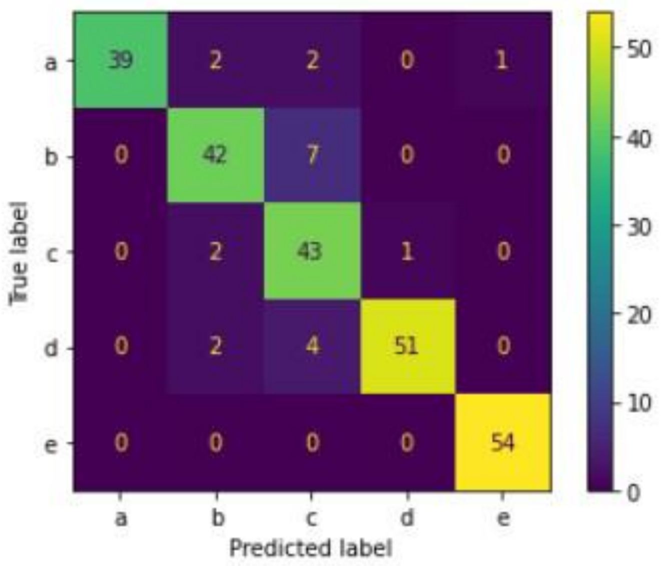
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train)
y_pred = neigh.predict(X_test)
```



```
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
plot_confusion_matrix(clf,X_test ,y_test)
print('Accuracy of model: {:.2f}%'.format(clf.score(X_test, y_test)*100))
print("Classification Report:\n",classification_report(y_test, y_pred))
```

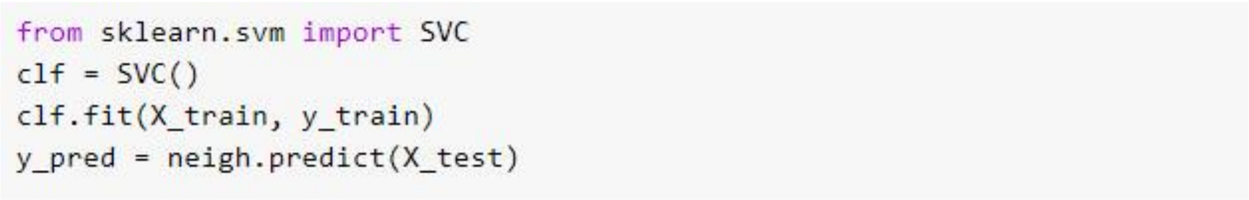


```
Confusion Matrix:
 [[42  2  0  0  0]
 [ 0 43  5  1  0]
 [ 0  5 38  3  0]
 [ 1  1  2 53  0]
 [ 1  0  0  0 53]]
Accuracy of model: 75.60%
Classification Report:
               precision    recall  f1-score

           a       0.95      0.95      0.95
           b       0.84      0.88      0.86
           c       0.84      0.83      0.84
           d       0.93      0.93      0.93
           e       1.00      0.98      0.99

    accuracy                           0.92
   macro avg       0.91      0.91      0.91
weighted avg       0.92      0.92      0.92
```

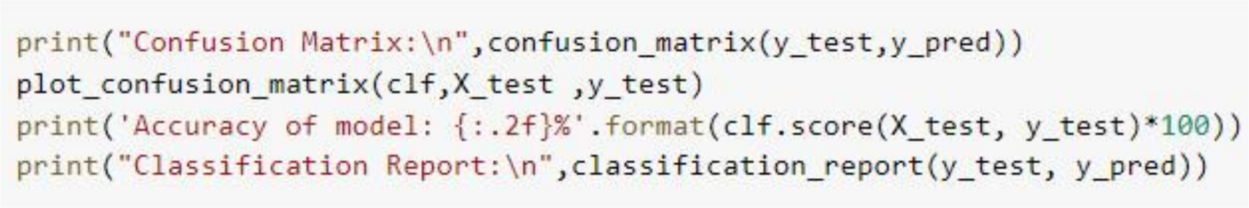


	a	b	c	d	e
a	39	2	2	0	1
b	0	42	7	0	0
c	0	2	43	1	0
d	0	2	4	51	0
e	0	0	0	0	54

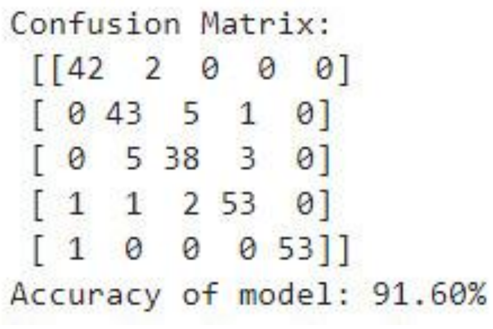
3- SVM:



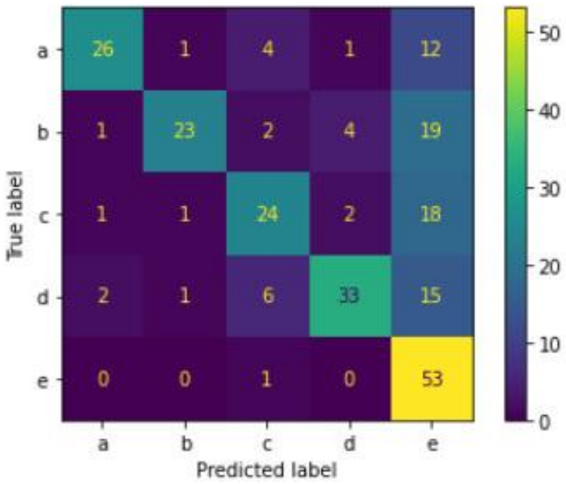
```
from sklearn.svm import SVC
clf = SVC()
clf.fit(X_train, y_train)
y_pred = neigh.predict(X_test)
```



```
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
plot_confusion_matrix(clf,X_test ,y_test)
print('Accuracy of model: {:.2f}%'.format(clf.score(X_test, y_test)*100))
print("Classification Report:\n",classification_report(y_test, y_pred))
```



```
Confusion Matrix:
 [[42  2  0  0  0]
 [ 0 43  5  1  0]
 [ 0  5 38  3  0]
 [ 1  1  2 53  0]
 [ 1  0  0  0 53]]
Accuracy of model: 91.60%
```

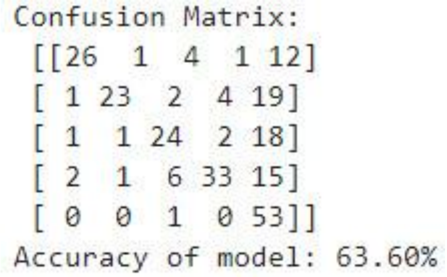


	a	b	c	d	e
a	26	1	4	1	12
b	1	23	2	4	19
c	1	1	24	2	18
d	2	1	6	33	15
e	0	0	1	0	53

TF-IDF (2 gram):

1. The decision tree tf-idf

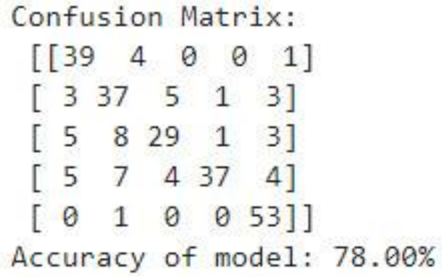
```
Accuracy of model: 63.60%
```



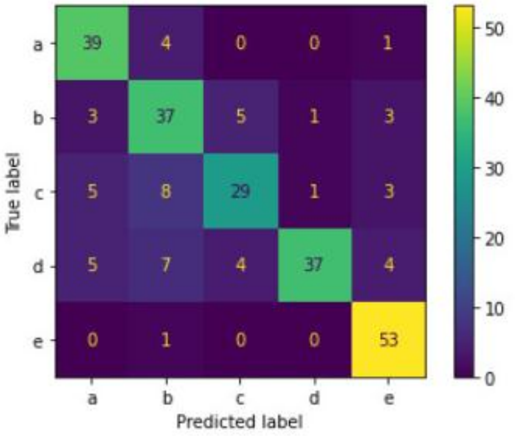
```
Confusion Matrix:
 [[26  1  4  1 12]
 [ 1 23  2  4 19]
 [ 1  1 24  2 18]
 [ 2  1  6 33 15]
 [ 0  0  1  0 53]]
Accuracy of model: 63.60%
```

2. KNN

```
Accuracy of model: 78.00%
```

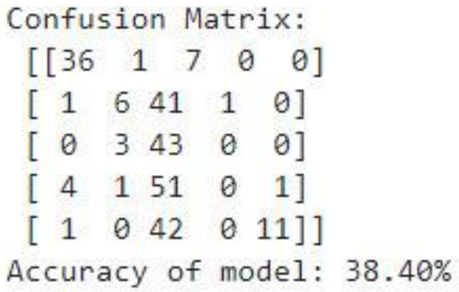


```
Confusion Matrix:
 [[39  4  0  0  1]
 [ 3 37  5  1  3]
 [ 5  8 29  1  3]
 [ 5  7  4 37  4]
 [ 0  1  0  0 53]]
Accuracy of model: 78.00%
```

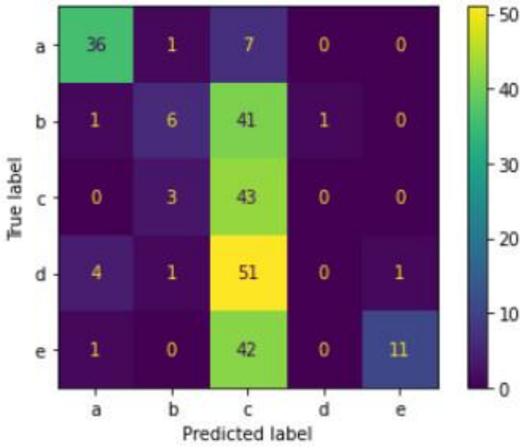


3. SVM model

Give accuracy = 38.4 % that not accepted in all so he can’t work with tf-idf 2 gram



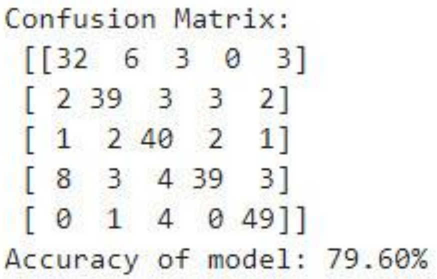
```
Confusion Matrix:
 [[36  1  7  0  0]
 [ 1  6 41  1  0]
 [ 0  3 43  0  0]
 [ 4  1 51  0  1]
 [ 1  0 42  0 11]]
Accuracy of model: 38.40%
```



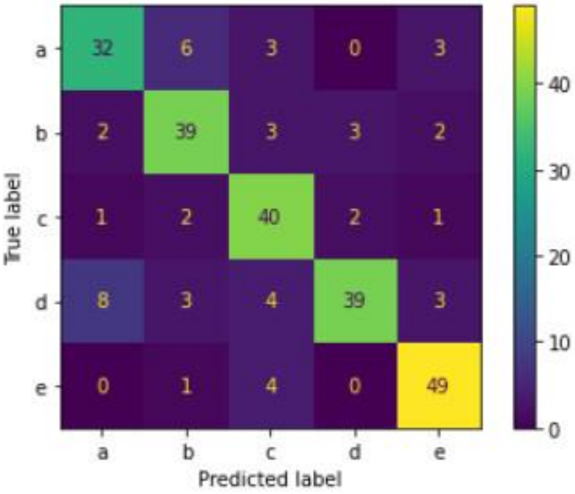
BOW Unigram:

1. The decision tree BOW 1ngram

```
Accuracy of model: 79.60%
```

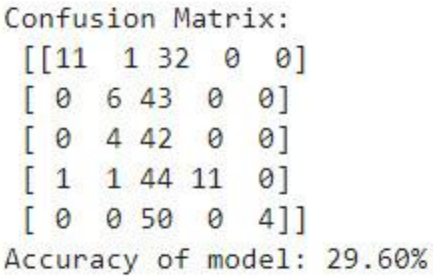


```
Confusion Matrix:
 [[32  6  3  0  3]
 [ 2 39  3  3  2]
 [ 1  2 40  2  1]
 [ 8  3  4 39  3]
 [ 0  1  4  0 49]]
Accuracy of model: 79.60%
```

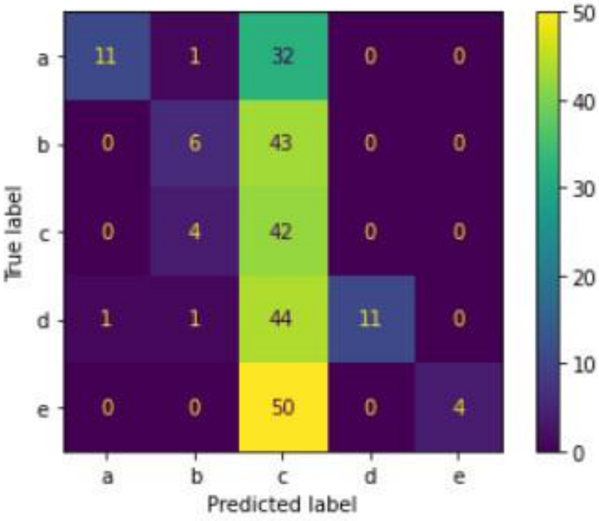


2. KNN   BOW 1ngram

```
Accuracy of model: 29.60%
```

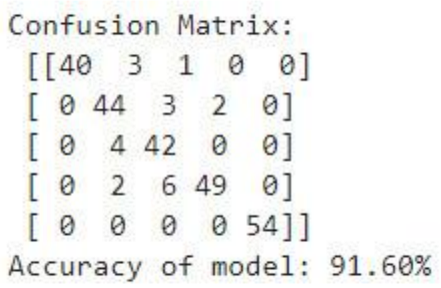


```
Confusion Matrix:
 [[11  1 32  0  0]
 [ 0  6 43  0  0]
 [ 0  4 42  0  0]
 [ 1  1 44 11  0]
 [ 0  0 50  0  4]]
Accuracy of model: 29.60%
```

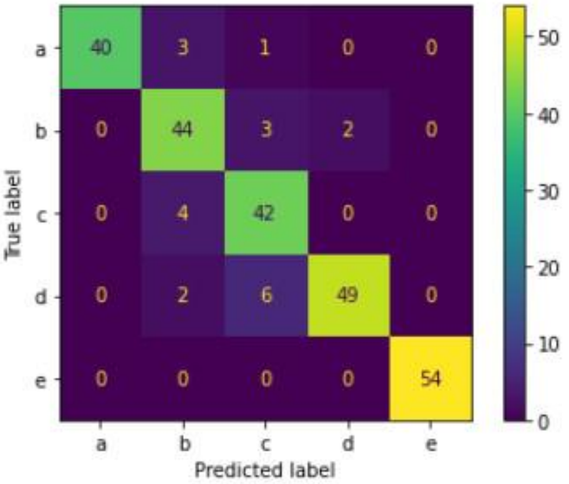


3. SVM model BOW 1ngram

```
Accuracy of model: 91.60%
```



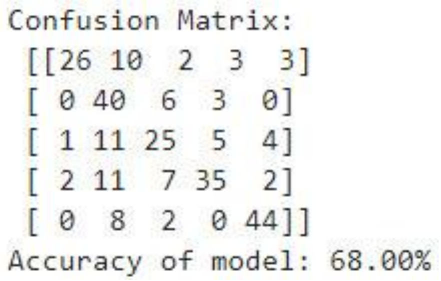
```
Confusion Matrix:
 [[40  3  1  0  0]
 [ 0 44  3  2  0]
 [ 0  4 42  0  0]
 [ 0  2  6 49  0]
 [ 0  0  0  0 54]]
Accuracy of model: 91.60%
```



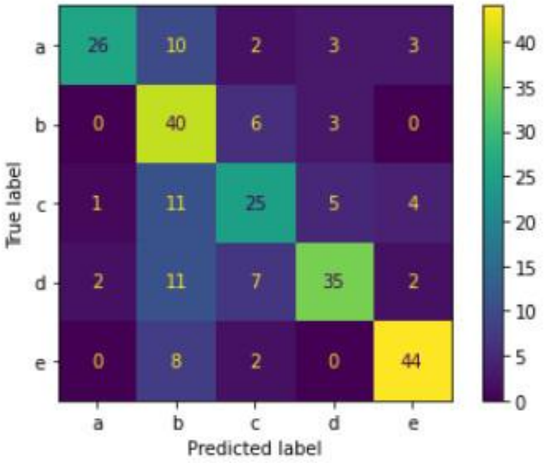
BOW 2 gram :

1. The decision tree BOW 2ngram

```
Accuracy of model: 68.00%
```

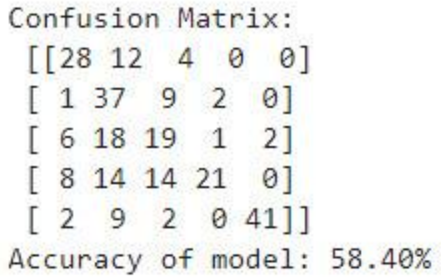


```
Confusion Matrix:
 [[26 10  2  3  3]
 [ 0 40  6  3  0]
 [ 1 11 25  5  4]
 [ 2 11  7 35  2]
 [ 0  8  2  0 44]]
Accuracy of model: 68.00%
```

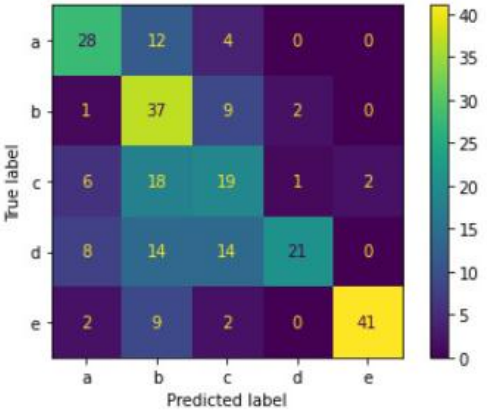


2. KNN   BOW 2ngram

```
Accuracy of model: 58.40%
```

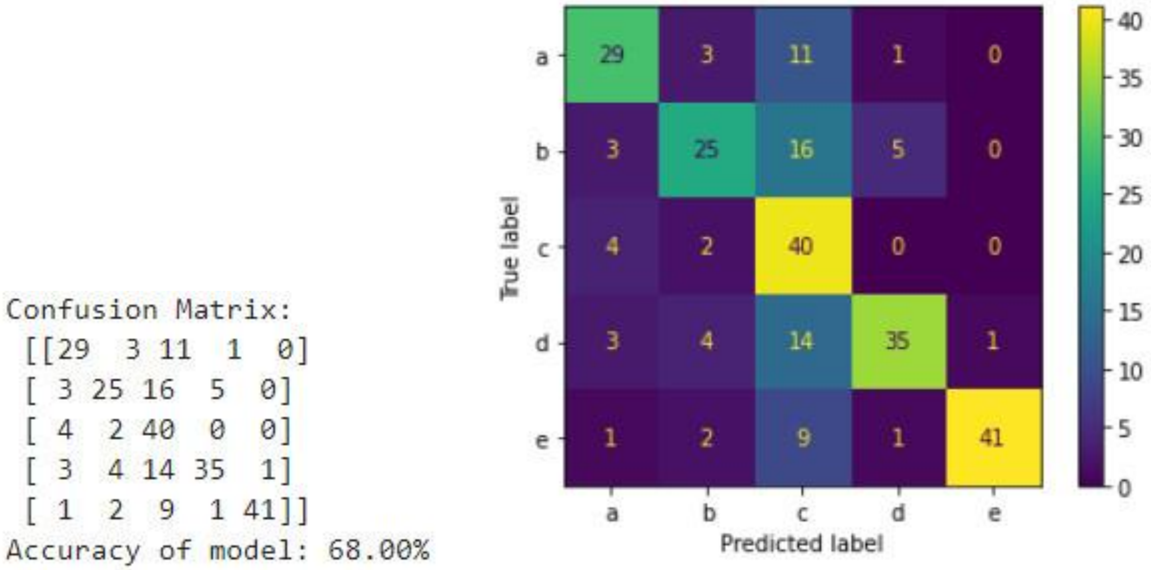


```
Confusion Matrix:
 [[28 12  4  0  0]
 [ 1 37  9  2  0]
 [ 6 18 19  1  2]
 [ 8 14 14 21  0]
 [ 2  9  2  0 41]]
Accuracy of model: 58.40%
```



3. SVM model BOW 2ngram:

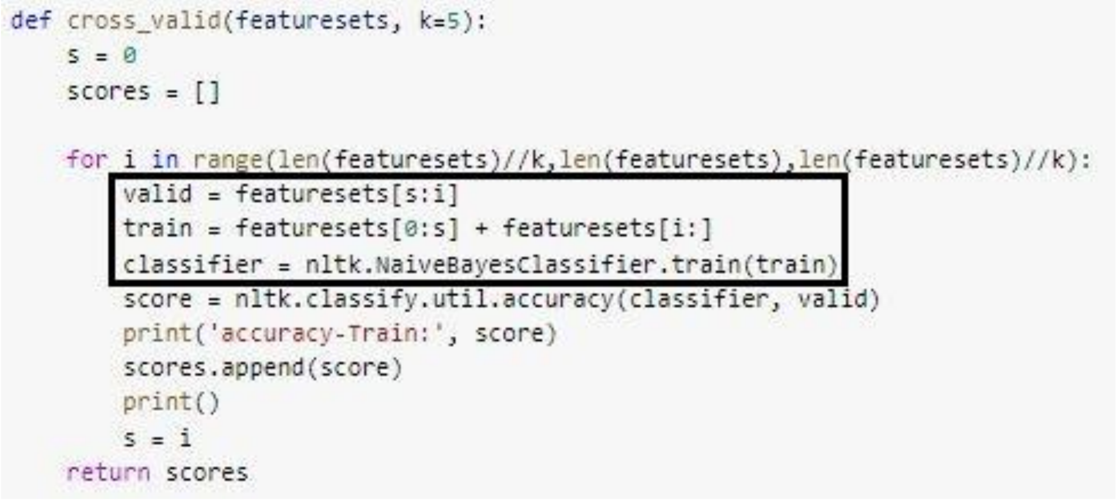
```
Accuracy of model: 58.40%
```



```
Confusion Matrix:
 [[29  3 11  1  0]
 [ 3 25 16  5  0]
 [ 4  2 40  0  0]
 [ 3  4 14 35  1]
 [ 1  2  9  1 41]]
Accuracy of model: 68.00%
```

4. Evaluation.

K-fold Cross validation is a strategy for determining how well the results of a statistical analysis will generalize to a different collection of data. It was utilized to compare and choose the winning model based on transformation and classification algorithm choices. The findings of the cross validation were used to determine our champion model.

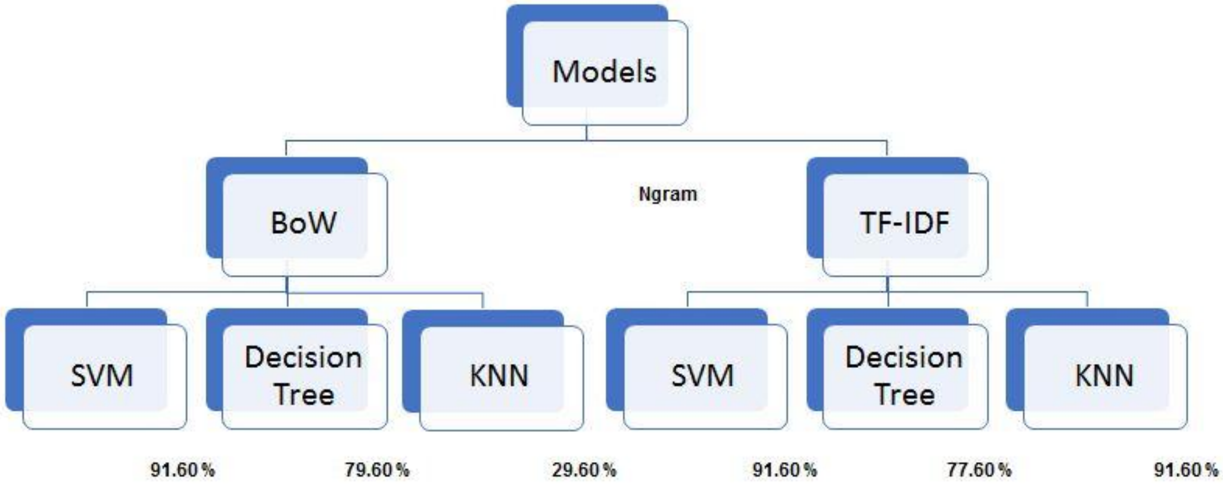


```
def cross_valid(featuresets, k=5):
    s = 0
    scores = []

    for i in range(len(featuresets)//k,len(featuresets),len(featuresets)//k):
        valid = featuresets[s:i]
        train = featuresets[0:s] + featuresets[i:]
        classifier = nltk.NaiveBayesClassifier.train(train)
        score = nltk.classify.util.accuracy(classifier, valid)
        print('accuracy-Train:', score)
        scores.append(score)
        print()
        s = i
    return scores
```

5. Error Analysis

The introduction paragraphs at the beginning and the ending of each book are one issue we encountered, and it was already mentioned in the data cleansing part. For all works, they are standard paragraphs given by Project Gutenberg. Those paragraphs had a negative impact on our model, and by deleting them, we were able to improve its accuracy.

6. Models Insights and The Champion One:



7. Bias and Variance

The Bias: The bias is a measure of how close the model can capture the mapping function between inputs and outputs

The Variance: The variance of the model is the amount the performance of the model changes when it is fit on different training data.

The bias and the variance of a model’s performance are connected and we calculate them in our project: