Classification (NLP)

Objective:

The purpose is to apply different classification models to detect one of the wanted targets we need when dealing with text so we went through natural language processing to apply some steps of preparing data to work on.

Choose data

These are five different samples from Gutenberg digital books about English Language for the authors:

- *William Malone Baskervill and James Witt Sewell*
- *Goold Brown*
- *Edward Sapir*
- *Samuel Johnson*
- *Sherwin Cody*

Preprocessing and Data Cleansing

*Read each book and apply encoding and tokenization, tokenization is the process of breaking down a text paragraph into smaller chunks such as words or sentence. 'So, we here apply regular expression to obtain only words and tokenize into words at the same time

```
for book,label_file in zip(url,string.ascii_letters):
  response = request.urlopen(book)
  my_book = response.read().decode('utf8').lower()
  tokenizer = RegexpTokenizer('[a-z]{2,}')
  my_book = tokenizer.tokenize(my_book)
```

*remove stop words and apply lemmatization to reduce words to their basic words

```
my_book = [lemmatizer.lemmatize(word) for word in my_book if word not in stop_words]
```

*Create random samples of 200 documents of each book, Prepare the records of 100 words records for each document, label them as a, b and c etc. as per the book they belong to

```
for i in range(len(my_book)):
      part1 = " ".join(my_book[i:i + per_line])
      all_list.append(part1)

tmp_list=random.sample(all_list, k=200)
list_of_random_items.extend(tmp_list)
list_label.extend([label_file]*len(tmp_list))
#clear lists
my_book=[]
all_list = []
```

	partition	label
0	termination word ending ize ise yse english ve...	a
1	make long though italy short tal stronger ic t...	a
2	usually italian sound unless accented ja bah p...	a
3	rather agree usually appointed honorable offic...	a
4	penult antepenult second syllable end third na...	a
...	...	...
995	two syllable seldom compared otherwise deplora...	e
996	harmless liability cost expense including lega...	e
997	termination european language preposition unle...	e
998	observed dr wallis irregular formation preteri...	e
999	preterit even solemn language crept felt dwelt...	e

1000 rows × 2 columns

Figure 1 the shape of data after preprocessing

Feature Engineering:

We cannot directly use text for our model. You need to convert these texts into some numbers or vectors of numbers. We have here some different approaches like (BOW, TF-IDF, n_gram …)

- BOW : converts text into the matrix of occurrence of words within a document
- TF-IDF: TF --- just count the number of words occurred in each document, so Term frequency is basically the output of the BOW model. , IDF ---- IDF is the logarithmically scaled inverse ratio of the number of documents that contain the word and the total number of documents.
- N_gram : contiguous sequence of n items from a given sample of text or speech

```
countvec = CountVectorizer()
cdf = countvec.fit_transform(df['partition'])
bow = pd.DataFrame(cdf.toarray(), columns = countvec.get_feature_names())
```

Figure 2 Applying BOW

	aad	ab	abandon	abbess	abbey	abbot	abbott	abbreviate	abbreviation	abdegilns	abel	aberration	abhorrence	abhorrent	a
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	
2	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	
4	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	0	0	0	
996	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	
998	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	

1000 rows × 12212 columns

Figure 3 // Subset from data after BOW

```
Final Result of TF-IDF
      aad   ab  abandon  abbess  abbey  ...  zion  zip  zon  zophorus  zure
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      0.0     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       0.0   0.0
3     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      0.0     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  ...   0.0  0.0  0.0       0.0   0.0
996   0.0  0.0      0.0     0.0    0.0  ...   0.0  0.0  0.0       0.0   0.0
997   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      0.0     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  0.0       0.0   0.0

[1000 rows x 12212 columns]
```

Figure 4 // Subset from data after TF-IDF

Building Models

Different types of models will be applied to each an every single kind of transformation to make analysis and comparison and choose the highest performance of them

```python
def build_model(model , X_train , y_train , X_test , y_test , cv):
    #model before cross_val
    genral_model = model.fit(X_train,y_train)

    y_pred = genral_model.predict(X_test)

    accuracy =  cross_val_score (estimator=model ,X= X_train,y= y_train ,cv=  cv )
    avg_accuracy = np.mean(accuracy)
    predictions = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_pred)
    #plot model_confusion_matrix classification_report
    print(classification_report(y_test, predictions))
    model_confusion_matrix = confusion_matrix(y_test, predictions)
    print(model_confusion_matrix)
    plot_confusion_matrix(model, X_test, y_test)
    print('avg_accuracy: ' , avg_accuracy)
    print('test_accuracy : ' , test_accuracy)

    return y_pred, genral_model
```

Figure 5// here is the main function in which we pass the type of the Model we want with every transformation

Results for each model

((BOW))

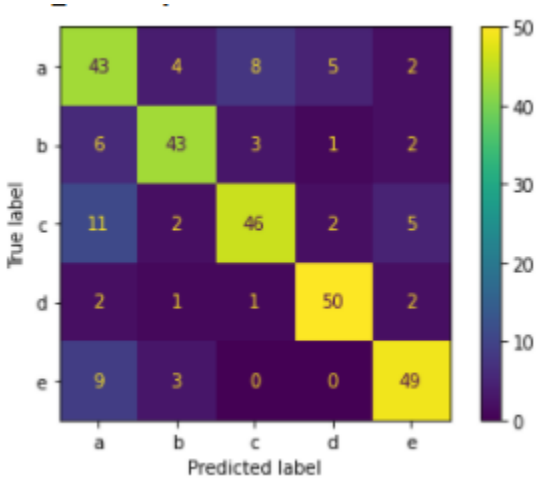


Figure 6 // confusion matrix for (decesion tree) after BOW transformation

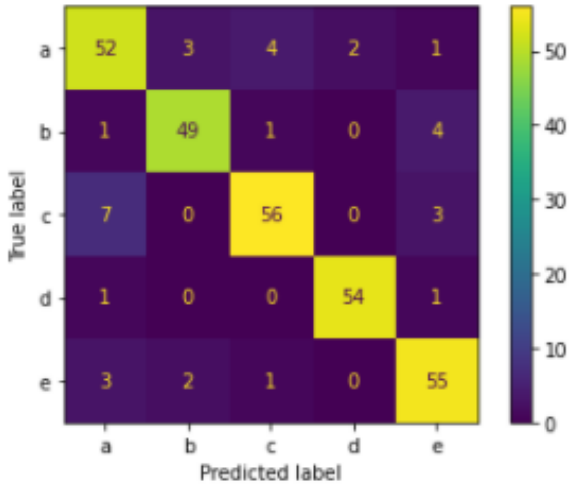


*Figure 7// confusion matrix for (**support vector**) after BOW transformation*

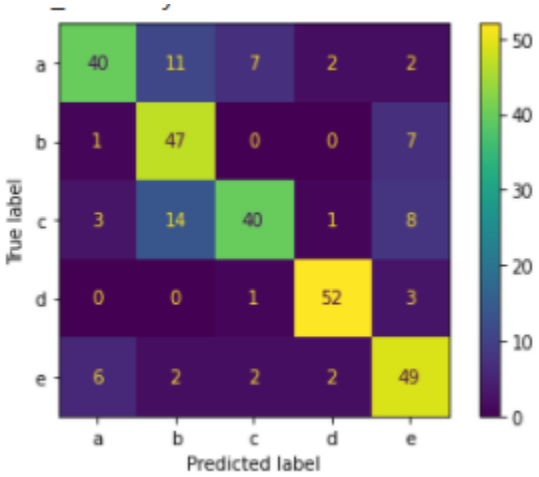


*Figure 8//confusion matrix for (**K_neighbor**) after BOW transformation*

((N_gram))

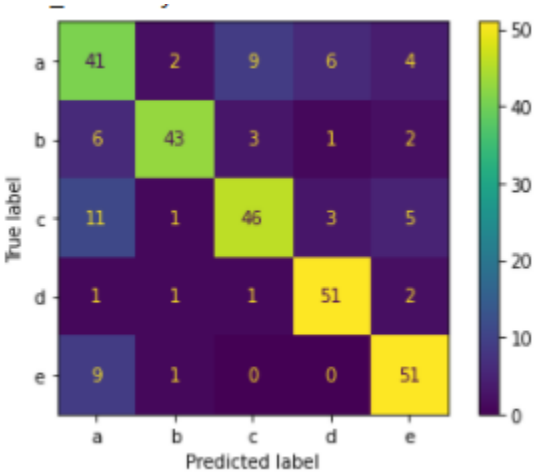


*Figure 9//F/ confusion matrix for (**decesion tree**) after N_gram transformation*

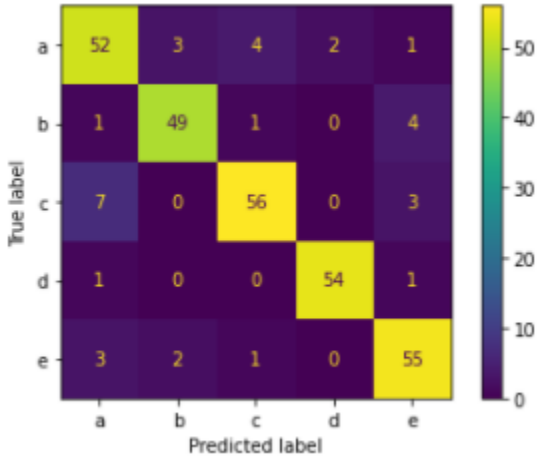
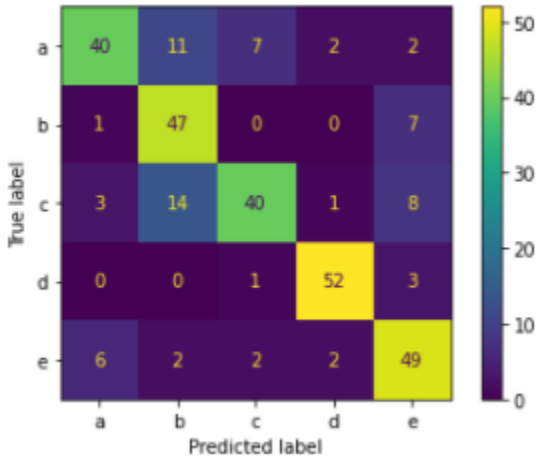


*Figure 10//confusion matrix for (**support vector**) after N_gram transformation*

*Figure 11 //confusion matrix for (**K_neighbor**) after N_gram transformation*



((TF-IDF))

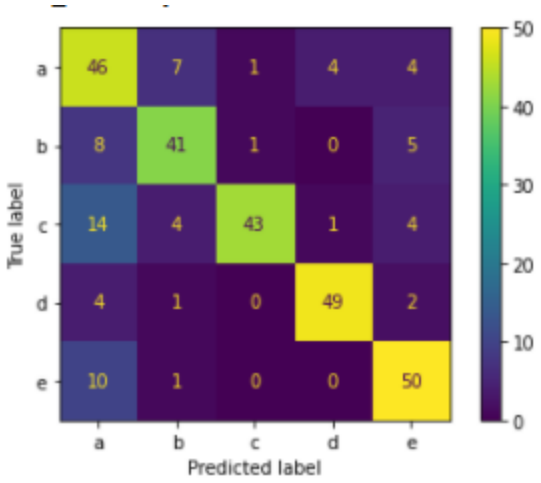


*Figure 12//confusion matrix for (**decision tree**) after TF-IDF transformation*

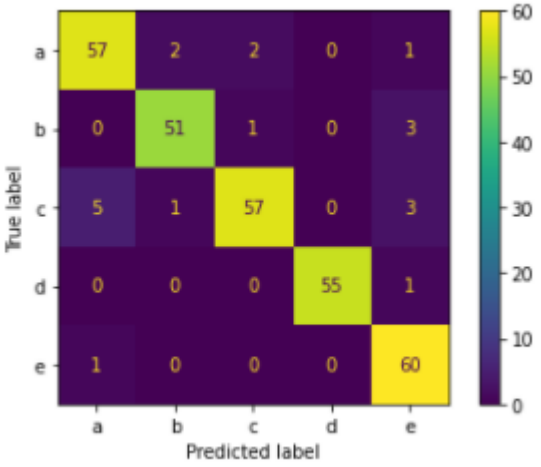


*Figure 13//confusion matrix for (**support vector**) after TF-IDF transformation*

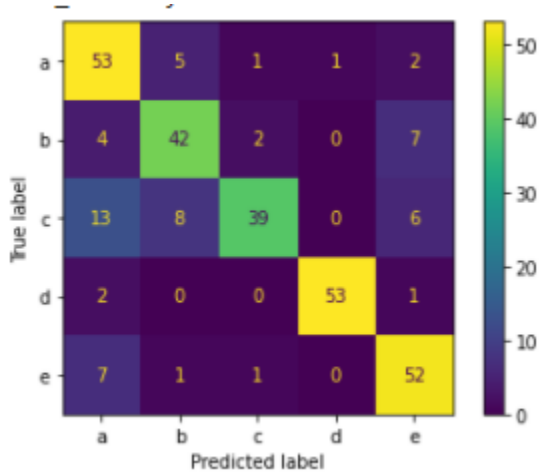


*Figure 14//confusion matrix for (**K_neighbor**) after TF-IDF transformation*

Evaluation

According to our implementation the confidence of the model increases when we applied 10 fold cross validation

Error Analysis

It is very important steps to go back and analysis the behavior of each model to detect the causes of error to eliminate it as much as possible

First, we here detect the portions that is equivalent to misclassification labels so that we can later apply different kinds of transformation mentioned above to reach to the cause of error

Champion model

By getting observations from the above figure, which contains values of bias and variance for each model, we reached a conclusion that our champion model is SVM after TF-IDF transformation as it has the minimum values of bias and variance.

after choosing the champion model we tried to change the parameters to reduce the accuracy by 20 % as shown:

```
# Train support vector  Classifer
tf_svm_model_champion = svm.SVC(kernel = "poly", max_iter=-1 ,decision_function_shape='ovr')
#build_model(TF_svm_model , X_tf_train , y_tf_train , X_tf_test , y_tf_test , 10)
y_pred_svm_tf_play, SVM_model_tf_play = build_model(tf_svm_model_play, X_tf_train , y_tf_train , X_tf_test , y_tf_test , 10)
```

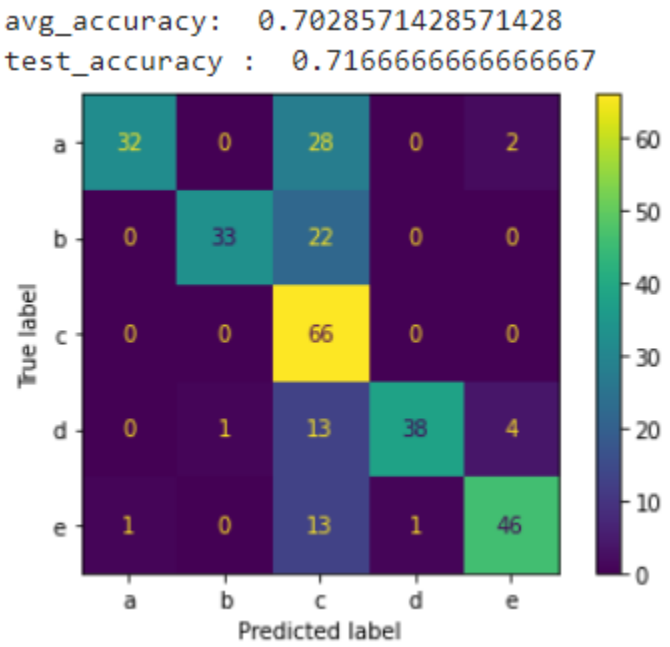


Figure 15 the confusion matrix and accuracy that reduced by about 20%

```
#Estimate bias, variance and MSE to SVM model Classifer after the tf-idf transform
SVM_TF_mse , SVM_TF_bias, SVM_TF_var = bias_variance_est(tf_svm_model_champion , X_tf_train , y_tf_train , X_tf_test , y_tf_test)

MSE: 0.373
Bias: 0.377
Variance: 0.135
```

Figure 16: bias and variance calculation

	error_partition	error_label	error_index	True labels
0	usually italian sound unless accented ja bah p...	a	2	c
1	concerning tax treatment donation received out...	a	9	a
2	lung stomach seeing nearer mainland expected b...	a	11	a
3	number torrent descended constant cataract one...	a	12	e
4	without accent since accent men attache closel...	a	23	c
...	...	...	...	...
66	animal kingdom one book honor human race emers...	b	269	a
67	word ing six kind according use well meaning f...	b	278	d
68	got ring box beak intent let fall rock sagacit...	b	282	a
69	laughlin relic recollect right pronounced tong...	b	293	c
70	type damage disclaimer limitation set forth ag...	b	299	b

71 rows × 4 columns

Figure 17// misclassification caused by decision tree classifier with BOW

In order to choose the champion model, some parameters need to be calculated like mean square error, bias and variance.

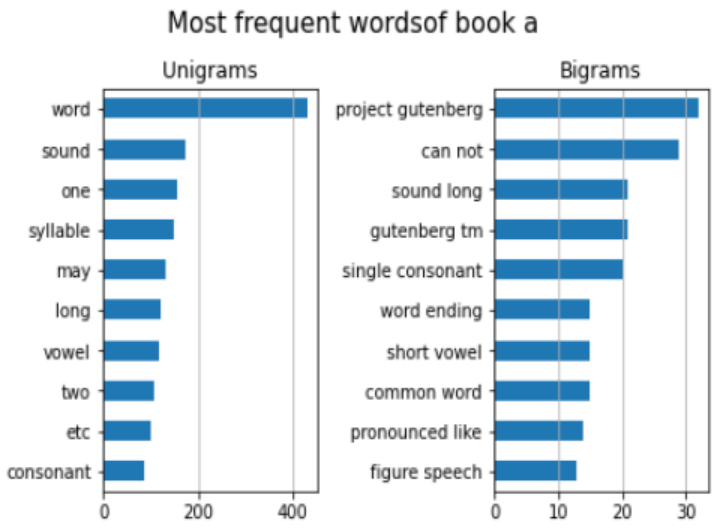
Here is the results of each model

	model name	model bias	model var
0	bow DecisionTree model bow	0.336667	0.178333
1	SVM model bow	0.170000	0.060000
2	KNeighbors model bow	0.310000	0.153333
3	DecisionTree model ngram	0.336667	0.193333
4	SVM model ngram	0.170000	0.060000
5	KNeighbors model ngram	0.310000	0.153333
6	DecisionTree model TF-IDF	0.343333	0.231667
7	SVM model TF-IDF	0.150000	0.046667
8	KNeighbors model TF-IDF	0.293333	0.148333

Figure 18 : value of bias and variance for each model

Word Frequency

We have calculated word frequency for each book and visualized them to get insights about our books which would help us in error analysis and features engineering.



WordCloud of book a