uOttawa

Text Classification Assignment 1

team 10

Ahmed Abdo Amin Abdo

Ahmed Abd Ellatife Mohamed Salah Eldine

Alaa Tohamy Mohamed AbdElwahab

Sherif Mohamed Abdelaziz Ahmed

1) Preparing the data

we have chosen 5 books from Gutenberg nltk library (Melvillemoby_dick.txt", "chesterton-ball.txt", "austen-emma.txt", "bryant-stories.txt", "edgeworth-parents.txt") all of them with same genre ("fiction")

then we have chosen 200 random partitions of each book every partition is a list containing 100 words

2) Preprocessing data

we have removed stop words and all characters that are not important in the data and lowered all the words to be efficient in training and testing

arranged books partitions, book name, and author name in a data frame

	The sentences	Book Name	Author Name
0	moby dick herman melville etymology supplied l...	melville-moby_dick	Herman Melv
1	greek cetus latin whoel anglo saxon hvalt dani...	melville-moby_dick	Herman Melv
2	fancied sung leviathan many nations generation...	melville-moby_dick	Herman Melv
3	ye strike splintered hearts together ye shall ...	melville-moby_dick	Herman Melv
4	whirlpooles called balaene take much length fo...	melville-moby_dick	Herman Melv
...	...	...	...
995	pains learn could write neat legible hand foun...	edgeworth-parents	Maria Edgew
996	bits paper writing bills father came bill hand...	edgeworth-parents	Maria Edgew
997	ever scold susan without wrong last as soon se...	edgeworth-parents	Maria Edgew
998	good boys put visit lamb went immediately brot...	edgeworth-parents	Maria Edgew
999	give well earned praise pleasure little subjec...	edgeworth-parents	Maria Edgew

1000 rows × 3 columns

After labeling the TARGET (author Name) column

	The sentences	Book Name	Author Name
0	moby dick herman melville etymology supplied l...	melville-moby_dick	1
1	greek cetus latin whoel anglo saxon hvalt dani...	melville-moby_dick	1
2	fancied sung leviathan many nations generation...	melville-moby_dick	1
3	ye strike splintered hearts together ye shall ...	melville-moby_dick	1
4	whirlpooles called balaene take much length fo...	melville-moby_dick	1
...	...	...	...
995	pains learn could write neat legible hand foun...	edgeworth-parents	3
996	bits paper writing bills father came bill hand...	edgeworth-parents	3
997	ever scold susan without wrong last as soon se...	edgeworth-parents	3
998	good boys put visit lamb went immediately brot...	edgeworth-parents	3
999	give well earned praise pleasure little subjec...	edgeworth-parents	3

1000 rows × 3 columns

3) Feature engineering

In this step, we have used BOW, N-gram, and TFIDF to transform the words into numeric values so the machine can understand easily and train it

BOW vectorizer

	moby	dick	herman	melville	etymology	supplied	late	consumptive	usher	grammar
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	1	0	0	0	0
996	0	0	0	0	0	1	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0

1000 rows × 12202 columns

N-gram vectorizer

	moby dick	dick herman	herman melville	melville etymology	etymology supplied	supplied late	late consumptive	consumptive usher	usher grammar	grammar school
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	0	0	0	0	1
996	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0

1000 rows × 80676 columns

TFiDF vectorizer

	moby	dick	herman	melville	etymology	supplied	late	consumptive	usher	grammar
0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
995	0.0	0.0	0.0	0.0	0.0	0.087502	0.0	0.0	0.0	0.0
996	0.0	0.0	0.0	0.0	0.0	0.089299	0.0	0.0	0.0	0.0
997	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
998	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0
999	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0

1000 rows × 12202 columns

4) Training the model

We used 3 algorithms to train our model: SVM, KNN, and decision tree every one of them used with the previous 3 feature engineering methods

So, we obtain 9 models and calculate classification report for each model.

The result of these algorithms is:

SVM with BOW:

```
BOW with SVM
f1-score: 0.9727272727272728
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.94      0.97      0.95        65
           0       0.97      1.00      0.98        62
           2       1.00      1.00      1.00        74
           4       0.95      0.94      0.95        67
           3       1.00      0.95      0.98        62

    accuracy                           0.97       330
   macro avg       0.97      0.97      0.97       330
weighted avg       0.97      0.97      0.97       330
```

SVM with N-gram:

```
N-Gram with SVM
f1-score: 0.7818181818181819
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.57      0.85      0.68        65
           0       0.84      0.66      0.74        62
           2       1.00      0.85      0.92        74
           4       0.82      0.75      0.78        67
           3       0.82      0.79      0.80        62

    accuracy                           0.78       330
   macro avg       0.81      0.78      0.78       330
weighted avg       0.81      0.78      0.79       330
```

SVM With TFiDF:

```
TFiDF with SVM
f1-score: 0.9939393939393939
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       1.00      1.00      1.00        69
           0       0.99      1.00      0.99        69
           2       1.00      1.00      1.00        66
           4       0.98      0.98      0.98        62
           3       1.00      0.98      0.99        64

    accuracy                           0.99       330
   macro avg       0.99      0.99      0.99       330
weighted avg       0.99      0.99      0.99       330
```

Decision Tree With BOW:

```
Decision Tree
f1-score: 0.8090909090909091
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.72      0.78      0.75        65
           0       0.75      0.81      0.78        62
           2       0.98      0.88      0.93        74
           4       0.81      0.76      0.78        67
           3       0.79      0.81      0.80        62

    accuracy                           0.81       330
   macro avg       0.81      0.81      0.81       330
weighted avg       0.82      0.81      0.81       330
```

Decision Tree With N-Gram:

```
Decision Tree
f1-score: 0.693939393939394
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.43      0.89      0.58        65
           0       0.92      0.53      0.67        62
           2       0.92      0.89      0.90        74
           4       0.76      0.57      0.65        67
           3       0.94      0.55      0.69        62

    accuracy                           0.69       330
   macro avg       0.79      0.69      0.70       330
weighted avg       0.79      0.69      0.71       330
```

Decision Tree with TFiDF:

```
Decision Tree
f1-score: 0.7909090909090909
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.72      0.74      0.73        69
           0       0.82      0.87      0.85        69
           2       0.90      0.82      0.86        66
           4       0.73      0.82      0.77        62
           3       0.80      0.70      0.75        64

    accuracy                           0.79       330
   macro avg       0.79      0.79      0.79       330
weighted avg       0.79      0.79      0.79       330
```

K nearest neighbors with BOW:

```
KNN Model
f1-score: 0.7090909090909091
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.48      1.00      0.65        65
           0       0.64      0.69      0.67        62
           2       0.98      0.77      0.86        74
           4       1.00      0.51      0.67        67
           3       1.00      0.56      0.72        62

    accuracy                           0.71       330
   macro avg       0.82      0.71      0.71       330
weighted avg       0.83      0.71      0.72       330
```

K nearest neighbors with N-Gram:

```
KNN Model
f1-score: 0.24848484848484848
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.21      1.00      0.34        65
           0       0.00      0.00      0.00        62
           2       1.00      0.11      0.20        74
           4       1.00      0.10      0.19        67
           3       1.00      0.03      0.06        62

    accuracy                           0.25       330
   macro avg       0.64      0.25      0.16       330
weighted avg       0.66      0.25      0.16       330
```

K nearest neighbors With TFiDF:

```
KNN Model
f1-score: 0.7333333333333333
-------------------------
classification report:
-------------------------
```

	precision	recall	f1-score	support
1	0.55	0.83	0.66	69
0	0.62	0.96	0.75	69
2	1.00	0.27	0.43	66
4	1.00	0.76	0.86	62
3	0.98	0.84	0.91	64
accuracy			0.73	330
macro avg	0.83	0.73	0.72	330
weighted avg	0.82	0.73	0.72	330

5) Analysis of Bias and Variability and Cross-Validation

We made a function to obtain “Mean Square Error and Variance and Bias and k Fold cross Validation” to each model to have a chance to choose with our champion model

```python
# calculate mes, bais, varience and cross validation
def calPerformance(model, X_train, y_train, X_test, y_test):
  scores = cross_val_score(model, X_train, y_train, cv=10)
  mse, bias, var=bias_variance_decomp(model ,X_train,y_train,X_test,y_test,loss='mse', num_rounds=200, random_seed=1)
  print('MSE: %.3f' % mse)
  print("Bias:%.3f"%bias)
  print("Variance:%.3f"%var)
  print(f"cross validation :{scores}")
  print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mean(), scores.std()))
  return mse, bias, var,scores
```

Results of Mean Square Error and bias and variance for each model

```
SVM With BOW
MSE: 0.33246969696969697, Bias: 0.24805522727272725 ,variance: 0.0844144696969697
SVM With NGram
MSE: 1.2066363636363635, Bias: 1.0028580303030306 ,variance: 0.2037783333333333
SVM With TFiDF
MSE: 0.28030303030303033, Bias: 0.22888803030303032 ,variance: 0.051414999999999995
-----------------------------------------------------------------
Decision Tree With BOW
MSE: 1.1732424242424242, Bias: 0.45627606060606063 ,variance: 0.7169663636363637
Decision Tree  With NGram
MSE: 1.5018636363636362, Bias: 1.0322202272727272 ,variance: 0.4696434090909091
Decision Tree  With TFiDF
MSE: 1.430742424242424, Bias: 0.6000234090909091 ,variance: 0.830719015151515
-----------------------------------------------------------------
knearest neighbors With BOW
MSE: 1.7336212121212122, Bias: 1.1957531060606061 ,variance: 0.5378681060606061
knearest neighbors  With NGram
MSE: 2.9855151515151515, Bias: 2.0422583333333333 ,variance: 0.9432568181818182
knearest neighbors  With TFiDF
MSE: 1.3290757575757575, Bias: 0.8186573484848485 ,variance: 0.5104184090909092
```

Results of 10 cross-validations, Std, and mean accuracy for each model:

SVM with BOW

```
cross validation :[0.98507463 0.97014925 0.98507463 0.94029851 0.97014925 0.95522388
 0.98507463 0.97014925 0.92537313 1.        ]
0.97 accuracy with a standard deviation of 0.02
```

SVM with N-gram

```
cross validation :[0.68656716 0.74626866 0.68656716 0.64179104 0.70149254 0.74626866
 0.79104478 0.8358209  0.58208955 0.73134328]
0.71 accuracy with a standard deviation of 0.07
```

SVM with TFiDF

```
cross validation :[0.95522388 0.98507463 0.94029851 0.95522388 0.98507463 0.98507463
 1.         0.92537313 0.97014925 1.        ]
0.97 accuracy with a standard deviation of 0.02
```

Decision Tree With BOW

```
cross validation :[0.68656716 0.85074627 0.7761194  0.7761194  0.80597015 0.7761194
 0.74626866 0.89552239 0.74626866 0.74626866]
0.78 accuracy with a standard deviation of 0.06
```

Decision Tree With N-Gram

```
cross validation :[0.64179104 0.71641791 0.62686567 0.67164179 0.6119403  0.7761194
 0.71641791 0.71641791 0.58208955 0.68656716]
0.67 accuracy with a standard deviation of 0.06
```

Decision Tree with TFiDF

```
cross validation :[0.82089552 0.76119403 0.73134328 0.71641791 0.76119403 0.74626866
 0.80597015 0.79104478 0.8358209  0.70149254]
0.77 accuracy with a standard deviation of 0.04
```

K nearest neighbors with BOW

```
cross validation :[0.62686567 0.73134328 0.65671642 0.70149254 0.62686567 0.68656716
 0.67164179 0.82089552 0.6119403  0.56716418]
0.67 accuracy with a standard deviation of 0.07
```

K nearest neighbors with N-Gram

```
cross validation :[0.23880597 0.2238806  0.23880597 0.26865672 0.23880597 0.23880597
 0.20895522 0.28358209 0.2238806  0.2238806 ]
0.24 accuracy with a standard deviation of 0.02
```

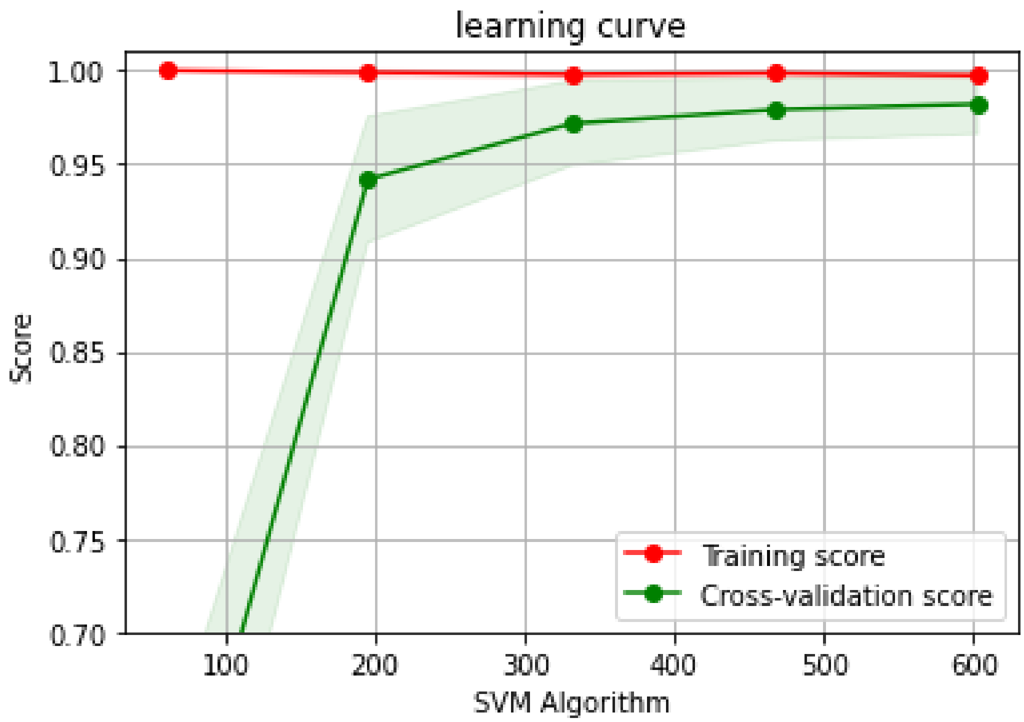
K nearest neighbors With TFiDF

```
cross validation :[0.59701493 0.62686567 0.50746269 0.55223881 0.74626866 0.71641791
 0.6119403  0.56716418 0.65671642 0.6119403 ]
0.62 accuracy with a standard deviation of 0.07
```

6) Champion model

Our champion model is SVM WITH TFiDF because it has the least bias and variance: 0.228 & 0.051 in order

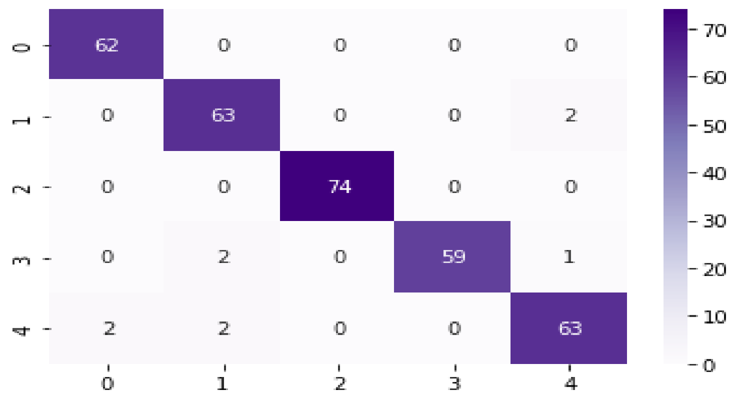
7) Validation curve to champion Model (SVM with TFiDF)



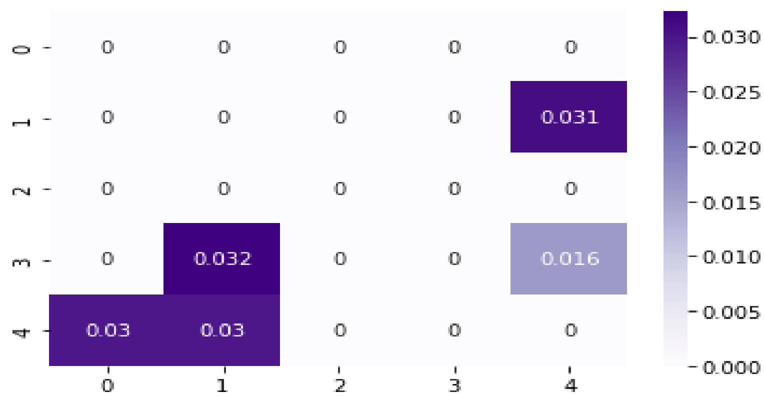
8) Error Analysis and Confusion Matrix of each model

SVM with BOW:

```
-------------------------
Confusion Matrix:
-------------------------
```

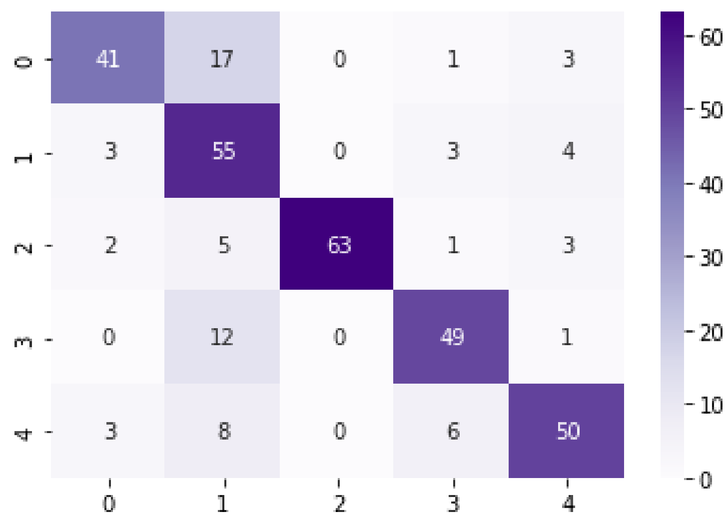



```
Error Analysis
```

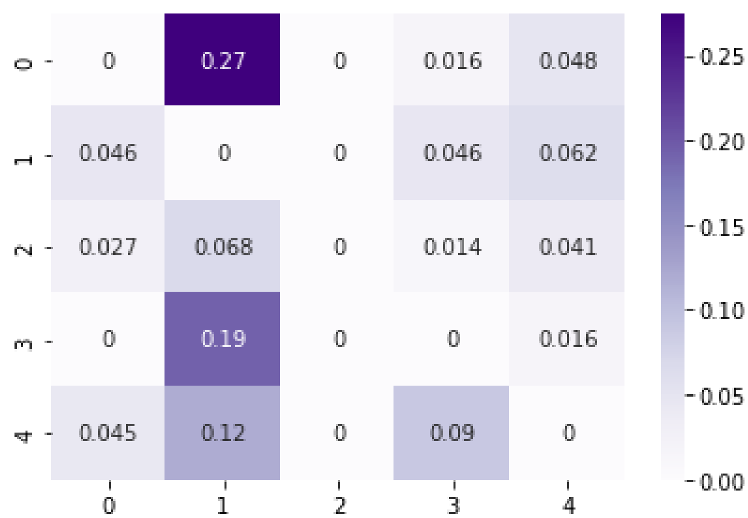


SVM with N-gram:

```
-------------------------
Confusion Matrix:
-------------------------
```



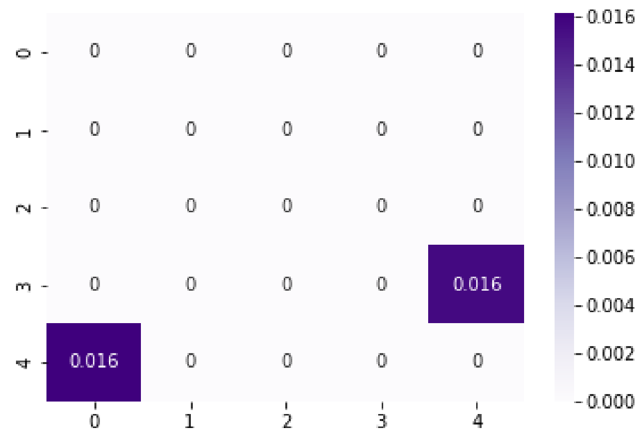

```
Error Analysis
```



SVM With TFiDF:

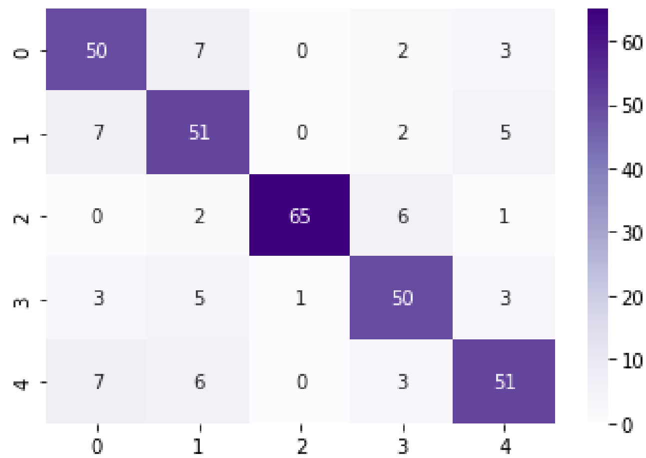
```
-------------------------
Confusion Matrix:
-------------------------
```


```
Error Analysis
```

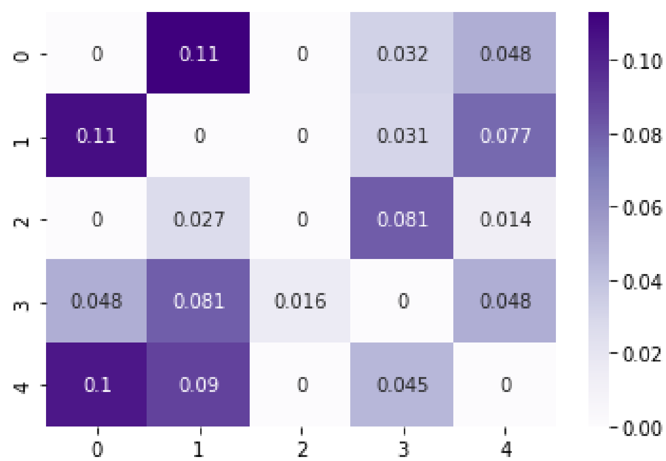


Decision Tree With BOW:

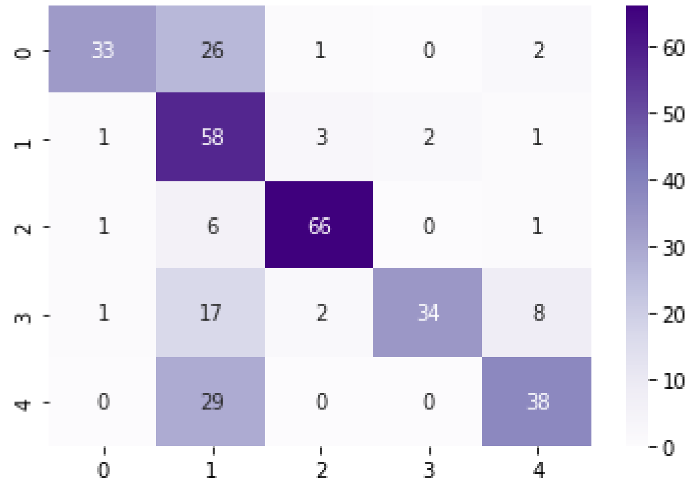
Confusion Matrix:



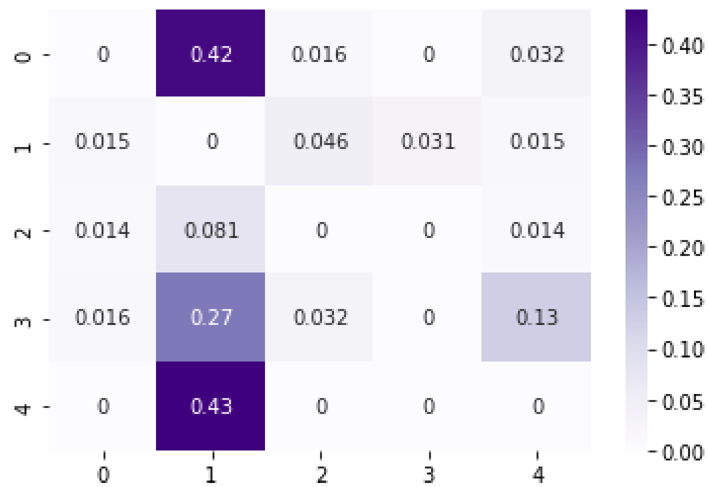
Error Analysis



Decision Tree With N-Gram:

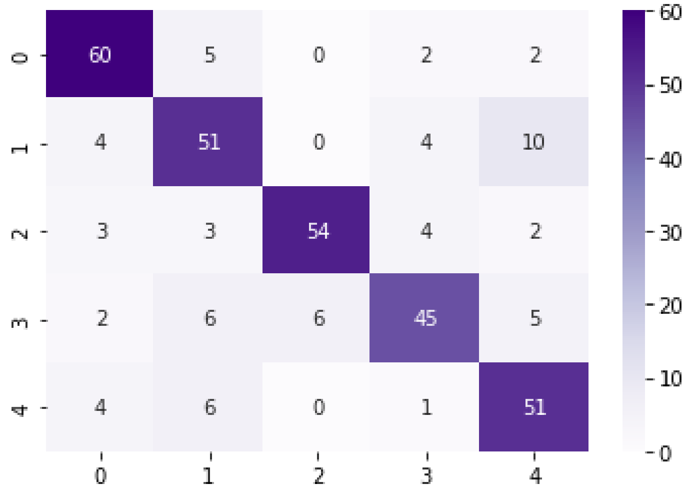


Error Analysis

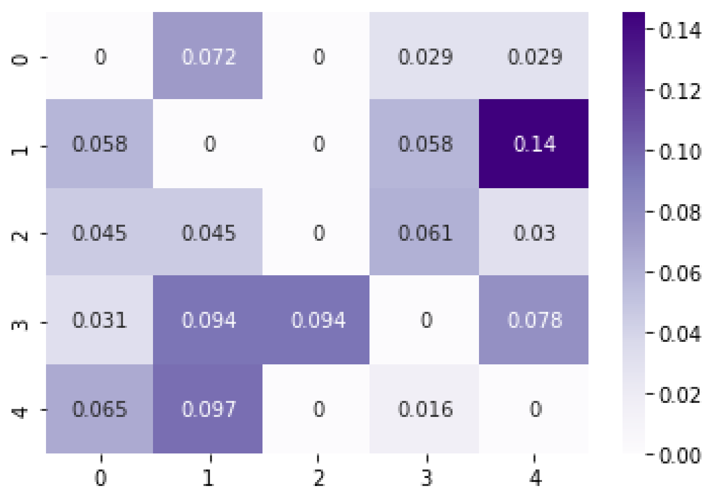


Decision Tree with TFiDF:

```
-------------------------
Confusion Matrix:
-------------------------
```

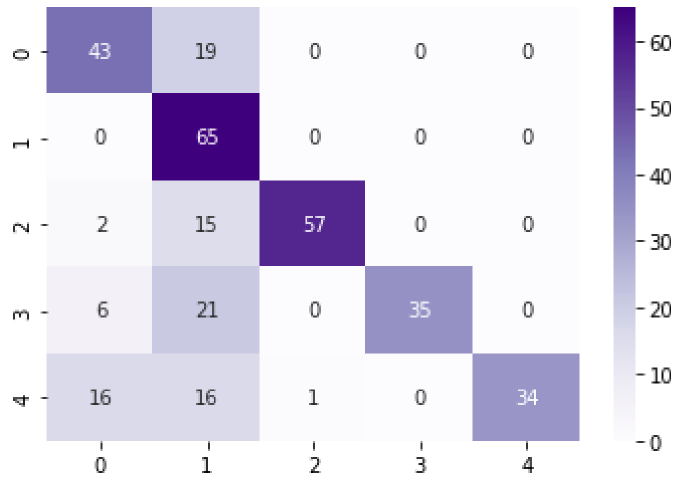



```
Error Analysis
```

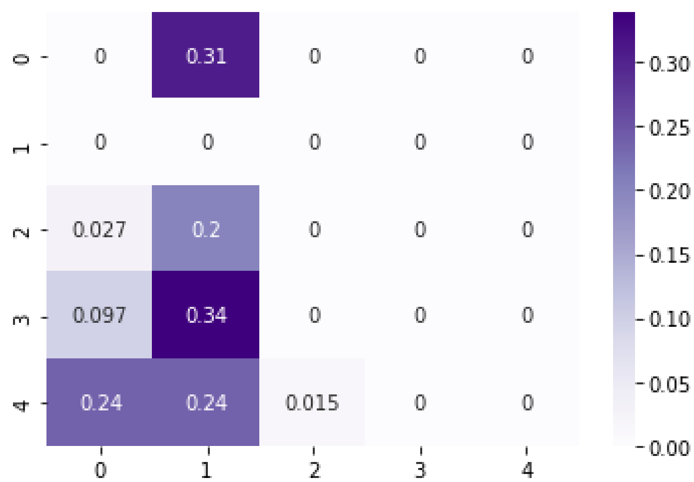


K nearest neighbors with BOW:

```
-------------------------
Confusion Matrix:
-------------------------
```

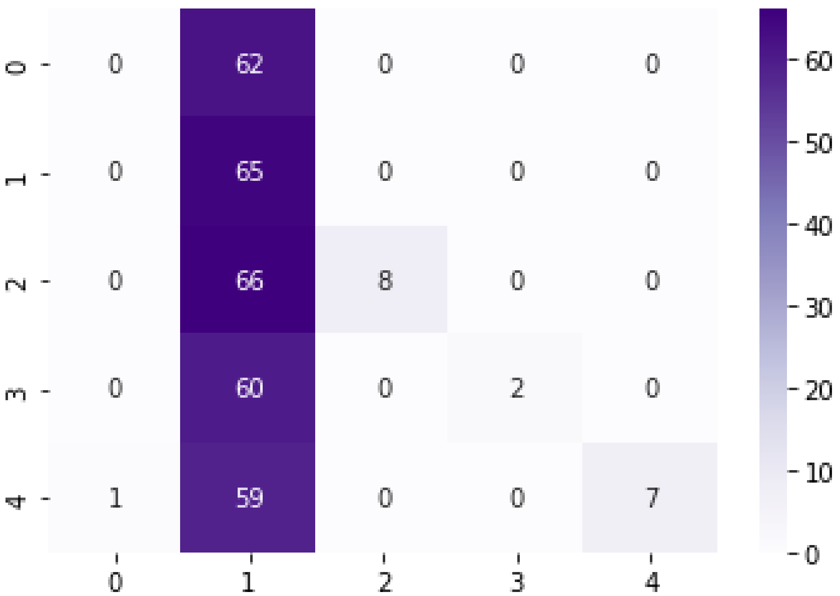



```
Error Analysis
```

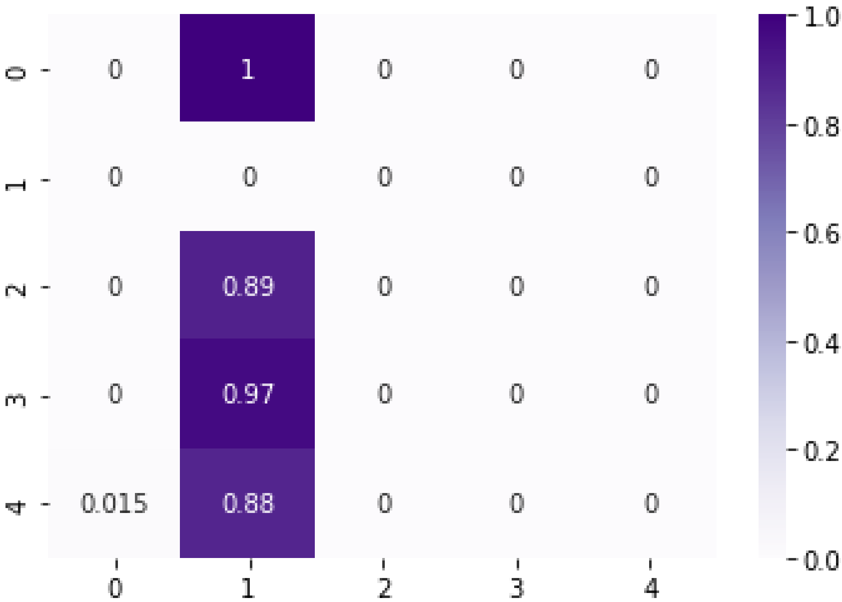


K nearest neighbors with N-Gram:

```
-------------------------
Confusion Matrix:
-------------------------
```

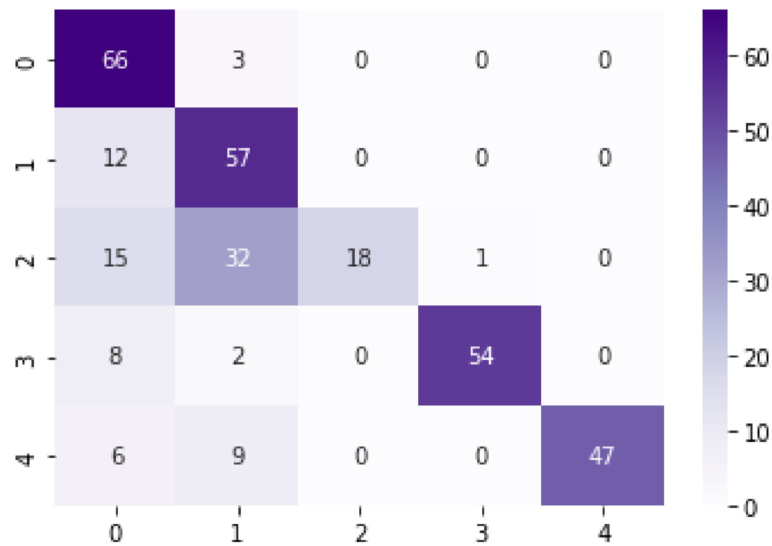



```
Error Analysis
```

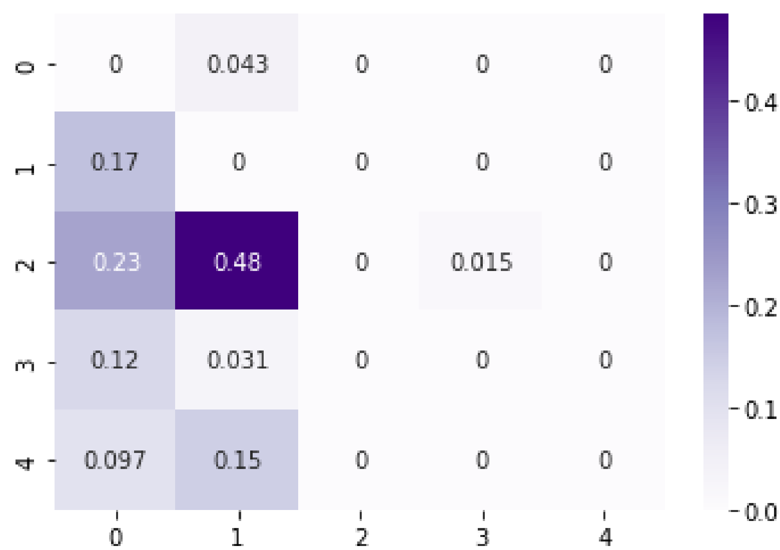


K nearest neighbors With TFiDF:

```
-------------------------
Confusion Matrix:
-------------------------
```




```
Error Analysis
```



9) Visualizations of the results

BOW

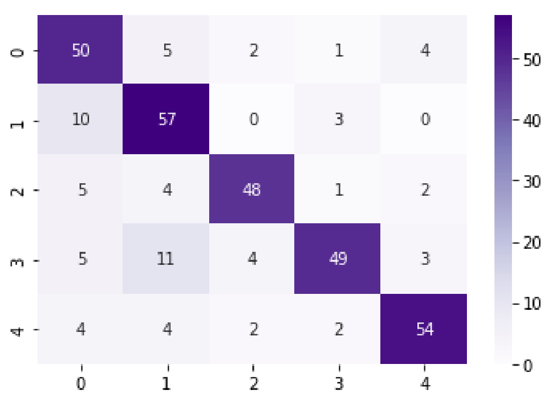
N-Gram

TFiDF

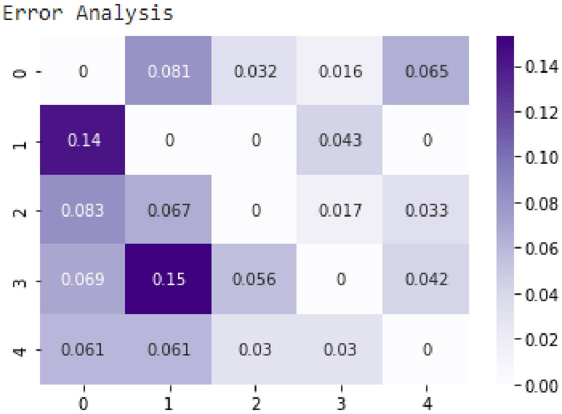
10) reduction of the accuracy of data by about 20% to the champion model (SVM with TFiDF)

```
TFiDF with SVM
f1-score: 0.7818181818181819
-------------------------
classification report:
-------------------------

              precision    recall  f1-score   support

           1       0.70      0.81      0.75        70
           0       0.68      0.81      0.74        62
           2       0.86      0.80      0.83        60
           4       0.86      0.82      0.84        66
           3       0.88      0.68      0.77        72

    accuracy                           0.78       330
   macro avg       0.79      0.78      0.78       330
weighted avg       0.79      0.78      0.78       330
```

```
-------------------------
Confusion Matrix:
-------------------------
```




```
Error Analysis
```




```
MSE: 1.313
Bias:0.726
Variance:0.587
cross validation :[0.79104478 0.7761194  0.82089552 0.80597015 0.80597015 0.76119403
 0.70149254 0.82089552 0.85074627 0.82089552]
0.80 accuracy with a standard deviation of 0.04
```

11) Reinforcement Learning

RL with LSTM

```
#model
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers


embedding_vector_features=len(X_TFiDF_features)

model=keras.Sequential()

model.add(keras.layers.Embedding(2441,embedding_vector_features,input_length=2441))

model.add(keras.layers.LSTM(64,input_shape=(X_TFiDF_Vec.shape),activation='relu',return

model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Dense(4,activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accura

print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 2441, 12202)       29785082

 lstm (LSTM)                 (None, 2441, 64)          3140352

 dropout (Dropout)           (None, 2441, 64)          0

 dense (Dense)               (None, 2441, 4)           260

=================================================================
Total params: 32,925,694
Trainable params: 32,925,694
Non-trainable params: 0
_________________________________________________________________
```

```
#X_test_TFiDF, y_test_TFiDF , X_train_TFiDF , y_train_TFiDF
model.fit(X_train_TFiDF,y_train_TFiDF,validation_data=(X_test_TFiDF,y_test_TFiDF),epochs=120,batch_size=64)
```

```
results = model.evaluate(X_test_TFiDF,y_test_TFiDF)
```

```
y_pred = model.predict(X_TFiDF_features)
```

```
y_pred
```