Multinomial Naive Bayes - BoW_l1

May 3, 2024

1 Initializtion

Connect to Google Drive:

```
[ ]: # from google.colab import drive
     # drive.mount('/content/drive')

     # %cd '/content/drive/MyDrive/GitHub/emotion-dectection-from-text'
```

Preparing necessary packages (may need to add more):

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import pandas as pd

     from sklearn.naive_bayes import MultinomialNB
     from sklearn.model_selection import GridSearchCV, cross_val_score
     from sklearn.metrics import accuracy_score
     from joblib import dump, load

     from preset_function import evaluate_model, draw_learning_curve,␣
      ↪load_processed_data

     X_train_bow, X_test_bow, X_train_tfidf, X_test_tfidf, \
         X_train_bow_L1, X_test_bow_L1, X_train_tfidf_L1, X_test_tfidf_L1 =␣
      ↪load_processed_data('input')

     y_train, y_test = load_processed_data('output')

     %matplotlib inline
```

Select dataset:

```
[ ]: X_train = X_train_bow_L1
     X_test = X_test_bow_L1
```

2 Basic training

We define and train a model with default hyperparameter, which is alpha = 1:

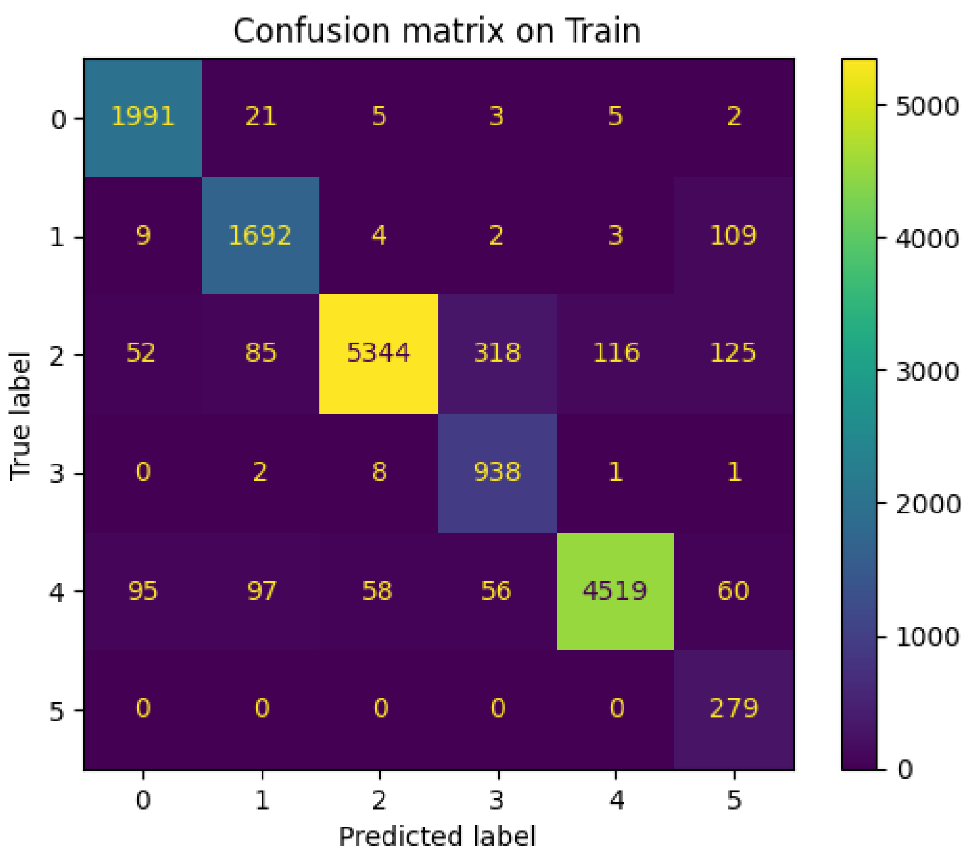
[ ]:
```
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
```

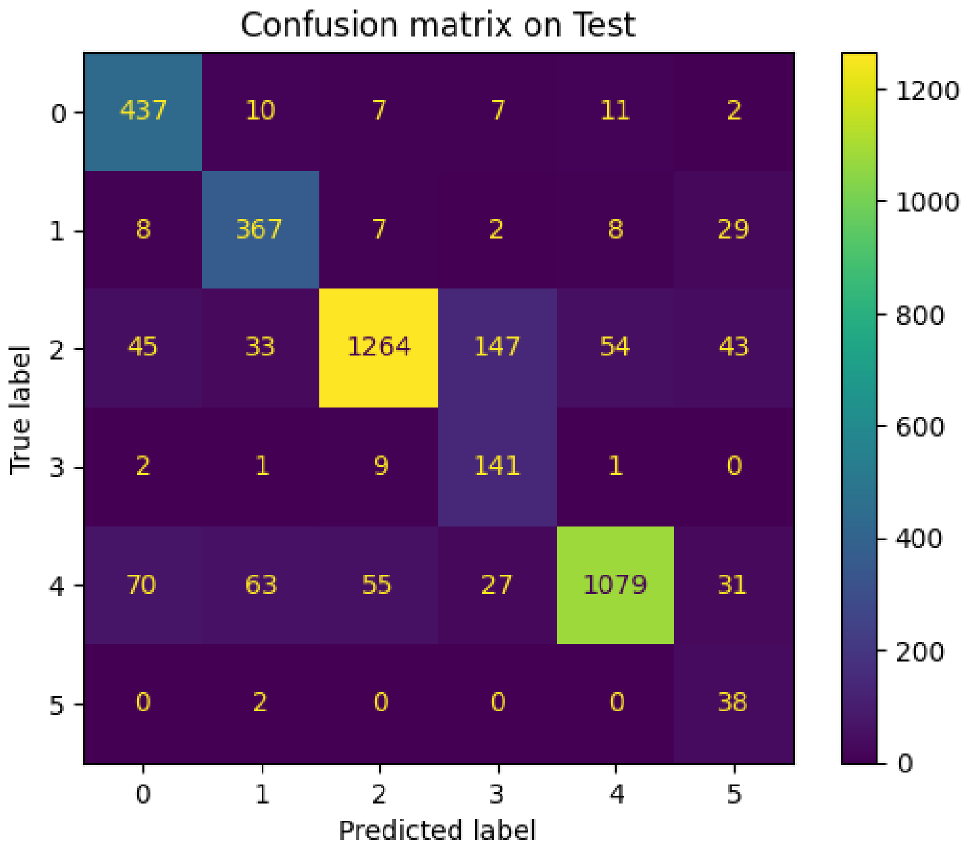
[ ]:
```
MultinomialNB()
```

Evaluate model using preset function:

[ ]:
```
evaluate_model(nb_model, X_train, X_test, y_train, y_test,␣
    ↪include_training=True)
```

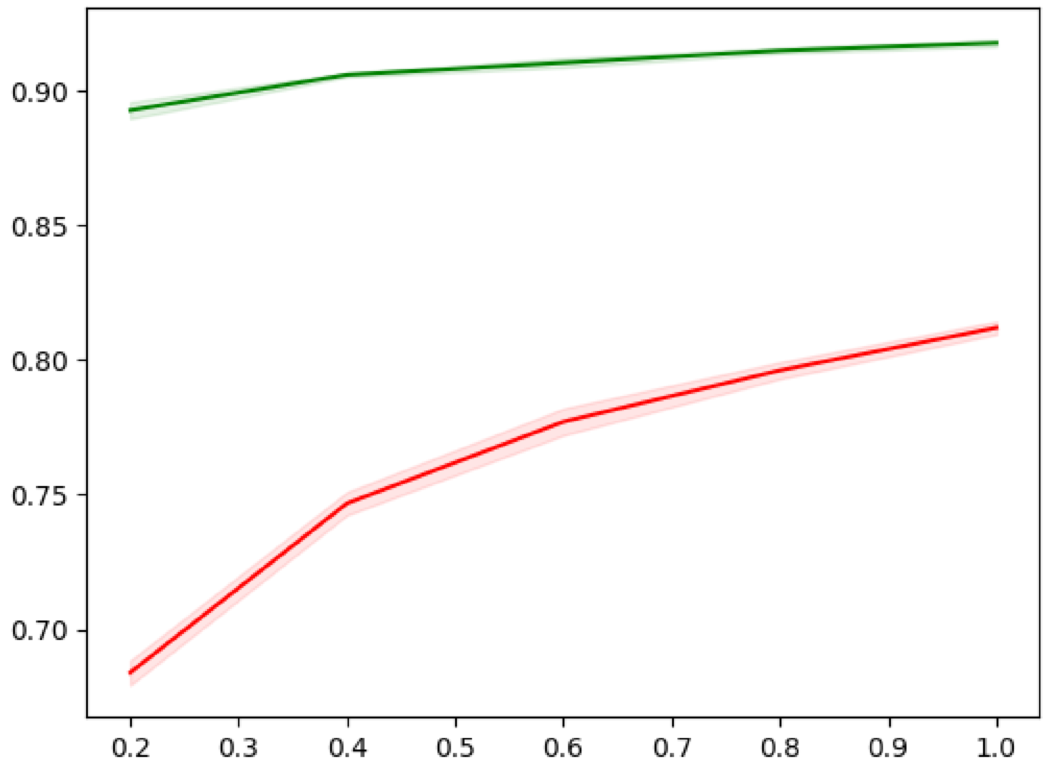
```
Score of on train are:
        - Accuracy score: 0.9227
        - Micro F1 score: 0.9227
        - Macro F1 score: 0.8710
Score of on test are:
        - Accuracy score: 0.8315
        - Micro F1 score: 0.8315
        - Macro F1 score: 0.7336
```





Draw the learning curve using preset function:

[ ]:
```
draw_learning_curve(nb_model, X_train, y_train)
```



3 Model selection

3.1 α parameter

First we try a hyperparameter range:

```
[ ]: # Setting the hyperparameter range
K = [0.0001, 0.001, 0.001, 0.01, 0.1, 1, 10]
```

```
[ ]: # Define a list in order to store accuracy points
cvs_list = list()
trs_list = list()

for k in K:
  # Define model for each K
  nb_model = MultinomialNB(alpha=k)
  nb_model.fit(X_train, y_train)

  # Calculate score of cross validation
  train_score = accuracy_score(y_train, nb_model.predict(X_train))
  cv_score = np.mean(cross_val_score(nb_model, X_train, y_train, cv=5,␣
 ↪n_jobs=8))
```

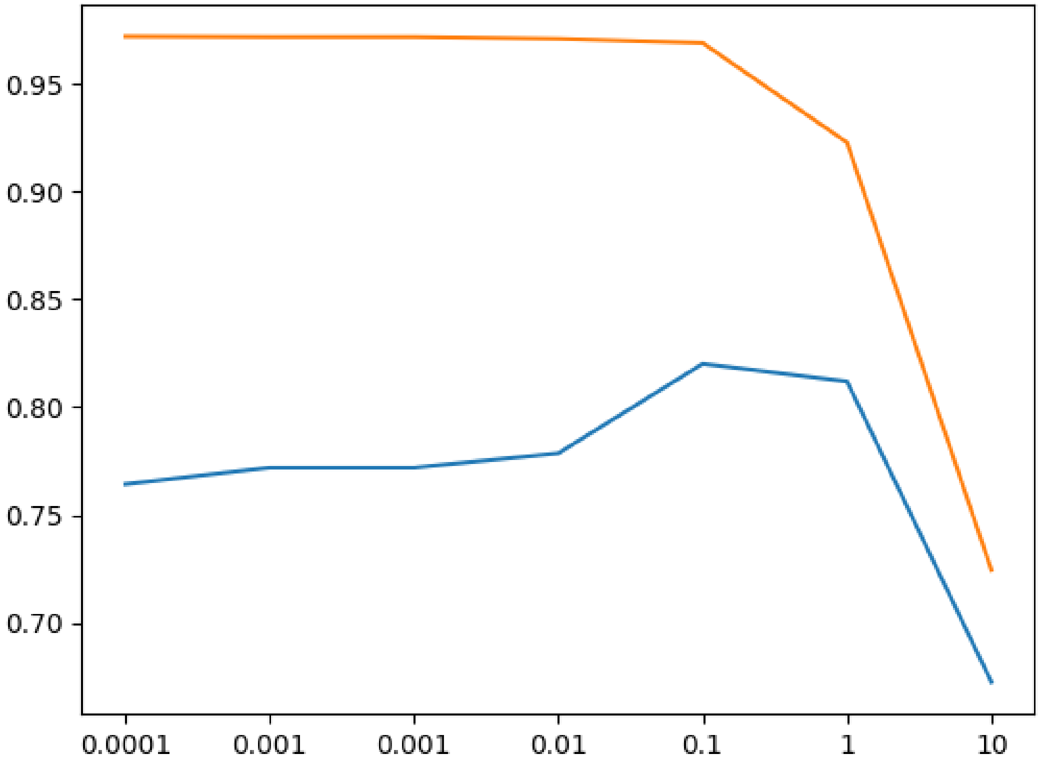
```
  trs_list.append(train_score)
  cvs_list.append(cv_score)
```

[ ]:
```
# Print the result
print(K)
print(trs_list)
print(cvs_list)

# Draw the plot
fig = sns.lineplot(x=list(range(len(K))), y=cvs_list)
fig = sns.lineplot(x=list(range(len(K))), y=trs_list)
fig.set_xticks(range(len(K)))
fig.set_xticklabels(K)
```

```
[0.0001, 0.001, 0.001, 0.01, 0.1, 1, 10]
[0.971875, 0.971625, 0.971625, 0.97075, 0.9688125, 0.9226875, 0.7245]
[0.7642499999999999, 0.771875, 0.771875, 0.7785624999999999, 0.8200625,
0.811875, 0.6725625]
```

[ ]:
```
[Text(0, 0, '0.0001'),
 Text(1, 0, '0.001'),
 Text(2, 0, '0.001'),
 Text(3, 0, '0.01'),
 Text(4, 0, '0.1'),
 Text(5, 0, '1'),
 Text(6, 0, '10')]
```



Iteration 1: From the result of above section, we can see the good value of α is near the value 0.1.

Scope to $\alpha = 0.1$.

Iteration 2: The good value of α is now near the value 0.25

```
[ ]: # Setting the hyperparameter range
K = [0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
```

```
[ ]: # Define a list in order to store accuracy points
cvs_list = list()
trs_list = list()

for k in K:
  # Define model for each K
  nb_model = MultinomialNB(alpha=k)
  nb_model.fit(X_train, y_train)

  # Calculate score of cross validation
  train_score = accuracy_score(y_train, nb_model.predict(X_train))
  cv_score = np.mean(cross_val_score(nb_model, X_train, y_train, cv=5,␣
 ↪n_jobs=8))
```

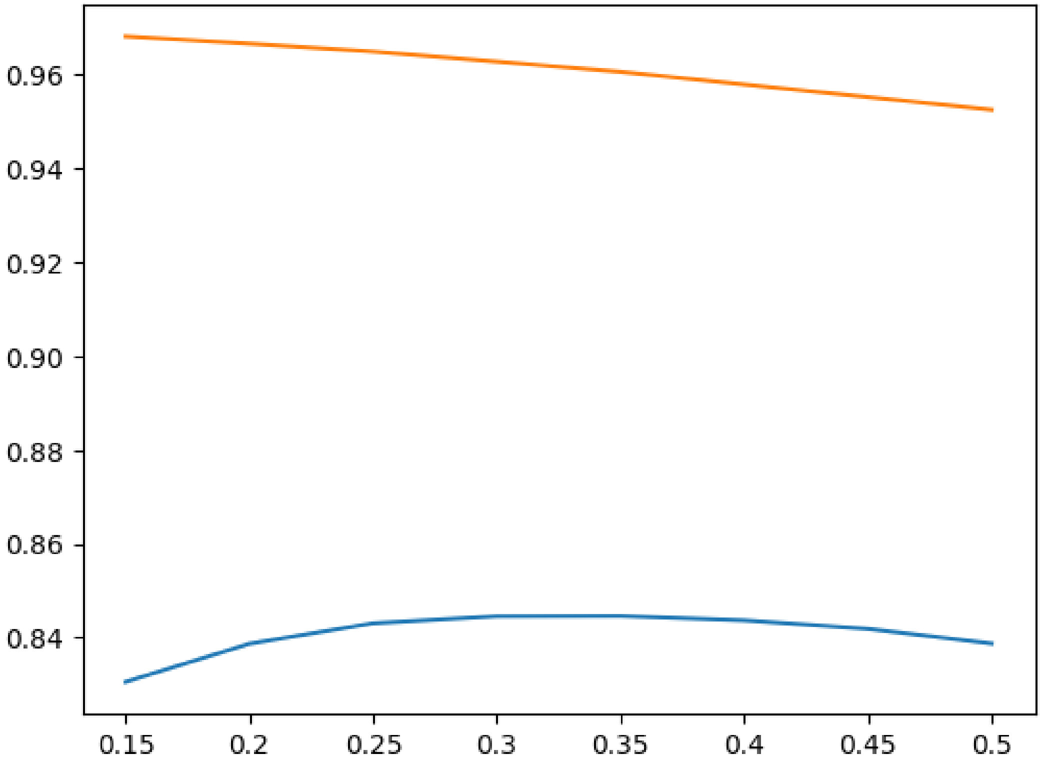
```
  trs_list.append(train_score)
  cvs_list.append(cv_score)
```

[ ]:
```
# Print the result
print(K)
print(trs_list)
print(cvs_list)

# Draw the plot
fig = sns.lineplot(x=list(range(len(K))), y=cvs_list)
fig = sns.lineplot(x=list(range(len(K))), y=trs_list)
fig.set_xticks(range(len(K)))
fig.set_xticklabels(K)
```

```
[0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
[0.9681875, 0.966625, 0.9649375, 0.9628125, 0.960625, 0.9579375, 0.95525,
0.9525625]
[0.8305624999999999, 0.8386875, 0.8430624999999999, 0.8445625,
0.8446250000000001, 0.8436875, 0.8418749999999999, 0.8387499999999999]
```

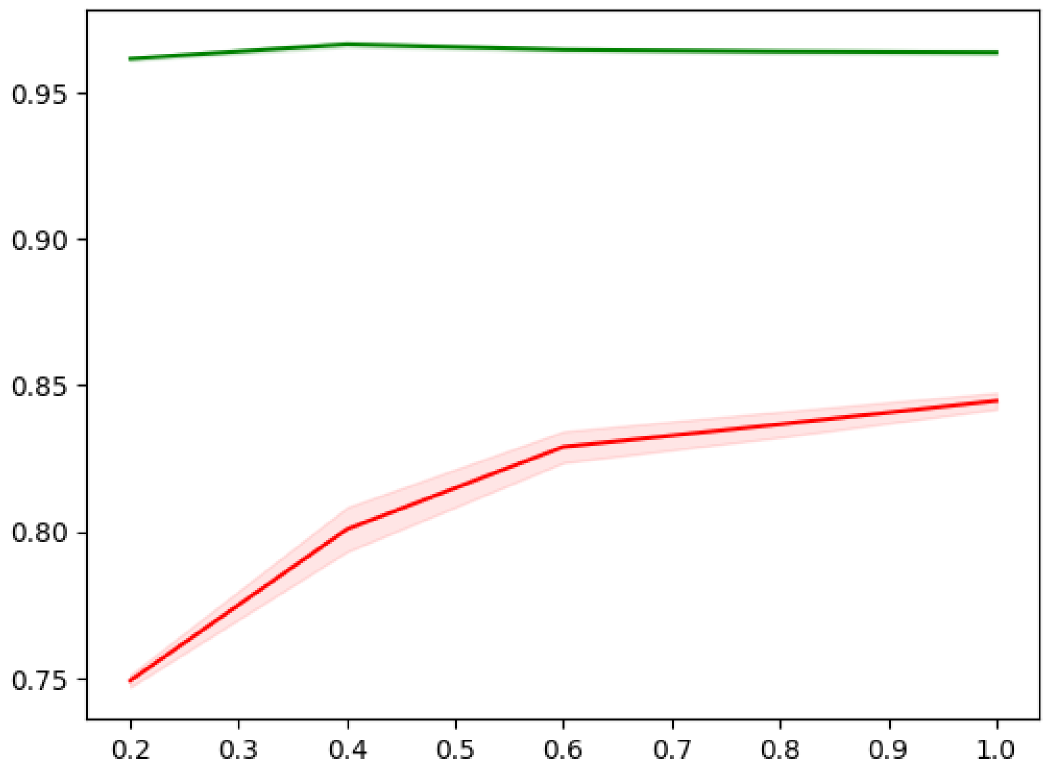
[ ]:
```
[Text(0, 0, '0.15'),
 Text(1, 0, '0.2'),
 Text(2, 0, '0.25'),
 Text(3, 0, '0.3'),
 Text(4, 0, '0.35'),
 Text(5, 0, '0.4'),
 Text(6, 0, '0.45'),
 Text(7, 0, '0.5')]
```



As the result, we can claim that $\alpha = 0.3$ give a model with good accuracy and avoid overfitting. We will test the model again in test set.

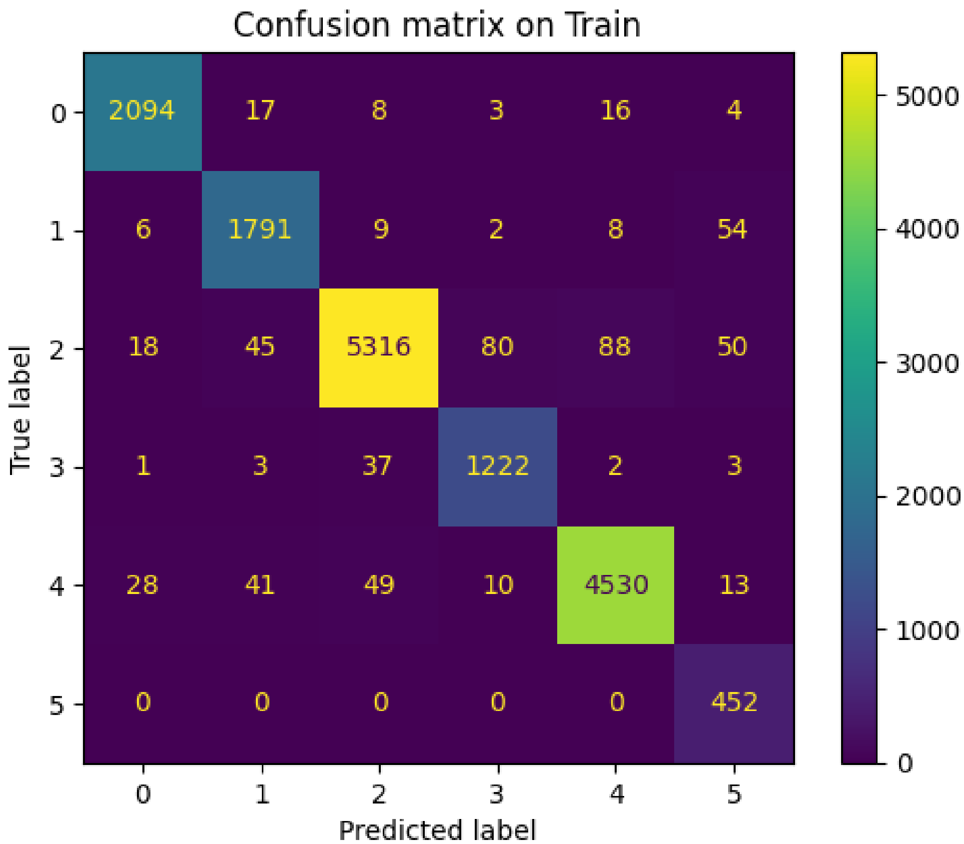
[ ]:
```
best_nb_model = MultinomialNB(alpha=0.3)
```

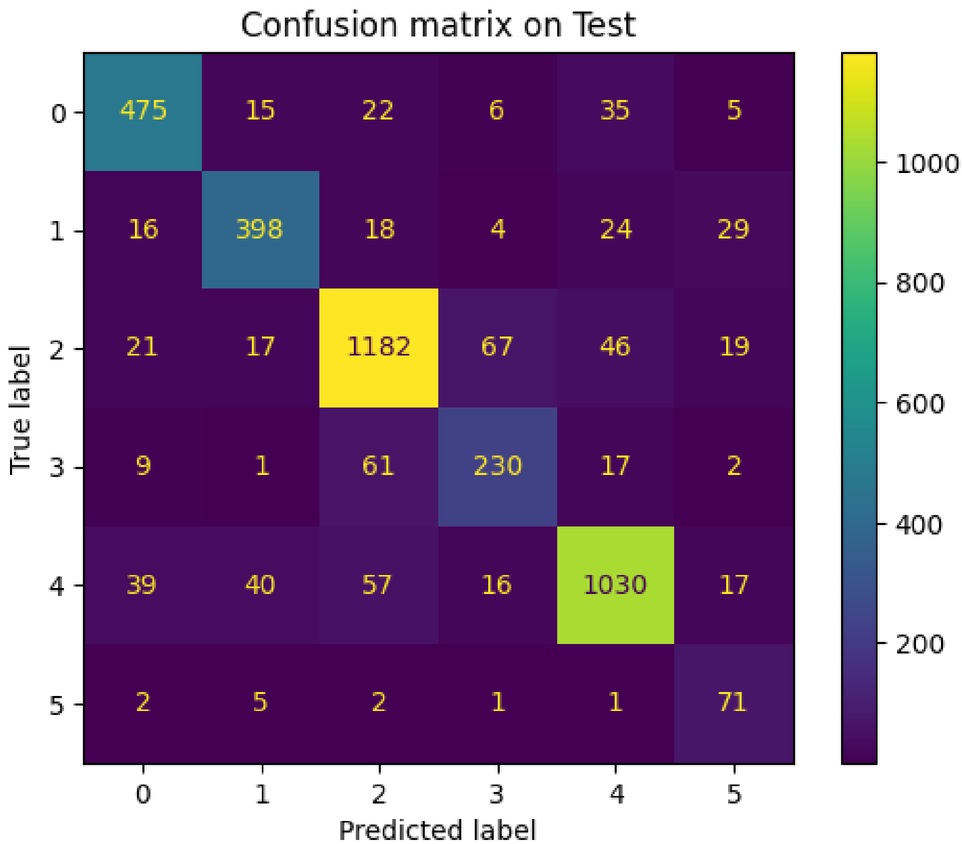
[ ]:
```
draw_learning_curve(best_nb_model, X_train, y_train)
```



[ ]:
```
best_nb_model.fit(X_train, y_train)
evaluate_model(best_nb_model, X_train, X_test, y_train, y_test,␣
 ↪include_training=True)
```

```
Score of on train are:
        - Accuracy score: 0.9628
        - Micro F1 score: 0.9628
        - Macro F1 score: 0.9483
Score of on test are:
        - Accuracy score: 0.8465
        - Micro F1 score: 0.8465
        - Macro F1 score: 0.7953
```


Confusion matrix on Train
True label
Predicted label
0 1 2 3 4 5
2094 17 8 3 16 4
6 1791 9 2 8 54
18 45 5316 80 88 50
1 3 37 1222 2 3
28 41 49 10 4530 13
0 0 0 0 0 452
0
1000
2000
3000
4000
5000



4 Export model

[ ]:
```
directory = "data/models/nb/"

dump(best_nb_model, directory + "best_nb_bow_l1_model.joblib")
```

[ ]: ['data/models/nb/best_nb_bow_l1_model.joblib']