K-nearest neighbors (KNN) - BoW - L1

May 7, 2024

1 Initialization

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:

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
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
```

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

2 Basic training

We define the model and train it first

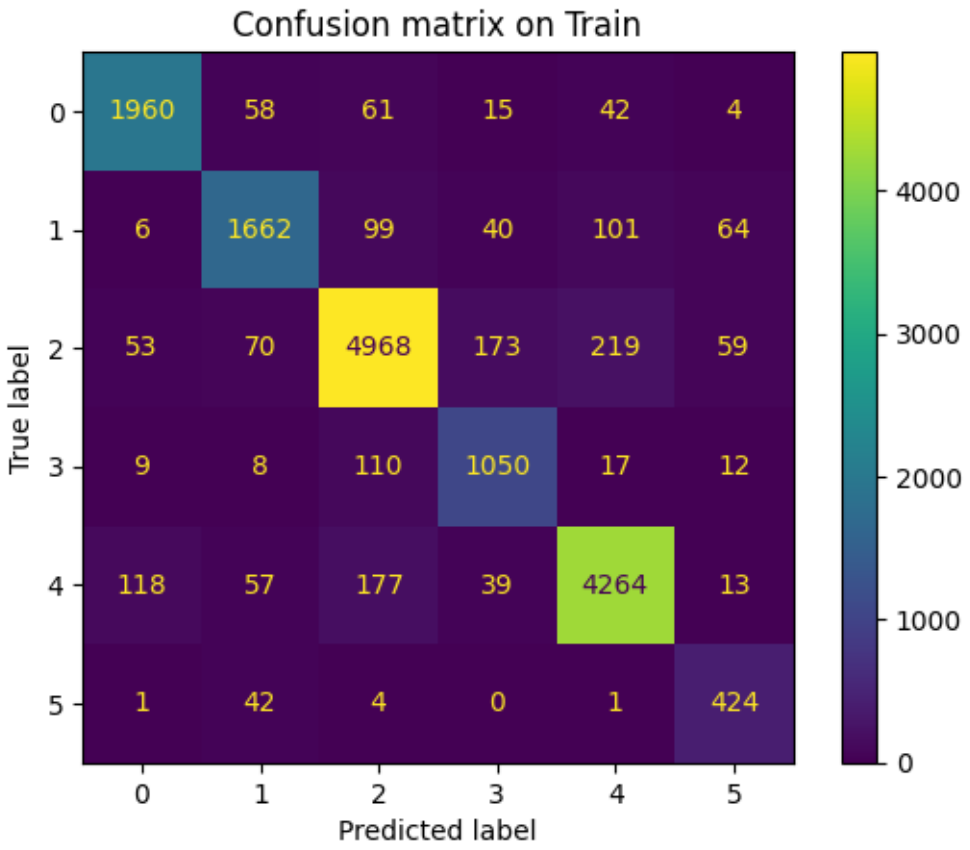
[ ]:
```
knn_model = KNeighborsClassifier(n_neighbors = 3)
knn_model.fit(X_train, y_train)
```

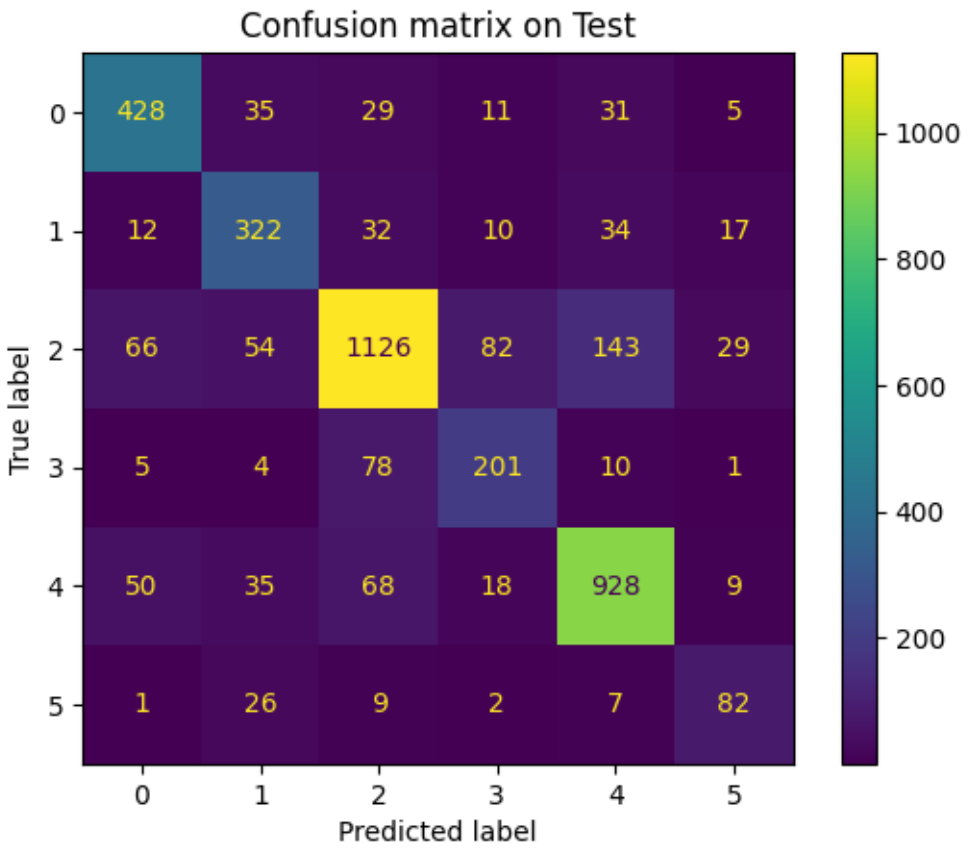
[ ]:
```
KNeighborsClassifier(n_neighbors=3)
```

Getting prediction on training set (without cross validation) then evaluate it!

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

```
Score of on train are:
        - Accuracy score: 0.8955
        - Micro F1 score: 0.8955
        - Macro F1 score: 0.8729
Score of on test are:
        - Accuracy score: 0.7718
        - Micro F1 score: 0.7717
        - Macro F1 score: 0.7261
```

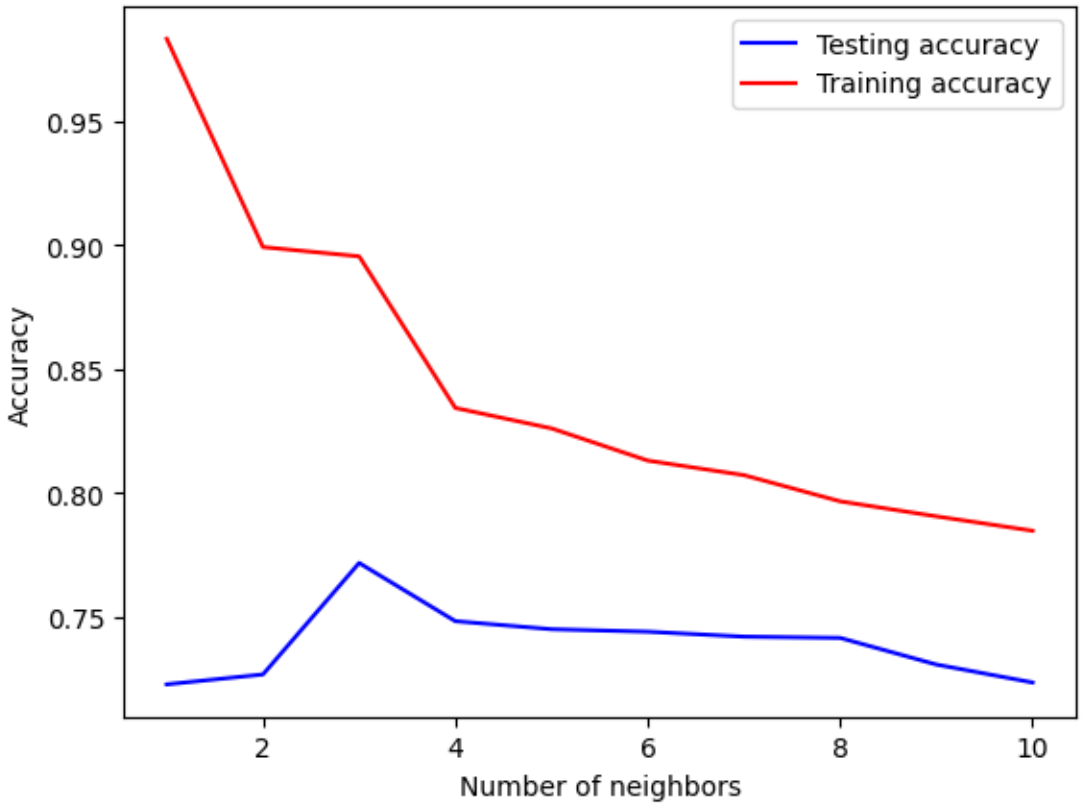




Now we draw the plot for a range of k-neighbors

[ ]:
```
# Setting the hyperparameter range
K = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
accuracy_list = list()
valid_accuracy_list = list()

for k in K:
  knn_model = KNeighborsClassifier(n_neighbors = k)
  knn_model.fit(X_train, y_train)

  data_pred_y = knn_model.predict(X_test)
  data_valid_y = knn_model.predict(X_train)

  accuracy_list.append([k, accuracy_score(y_test, data_pred_y)])
  valid_accuracy_list.append([k, accuracy_score(y_train, data_valid_y)])

accuracy_list = np.asarray(accuracy_list)
valid_accuracy_list = np.asarray(valid_accuracy_list)
```

```
plt.plot(accuracy_list[:, 0], accuracy_list[:, 1], label = "Testing accuracy",␣
 ↪color = 'b')
plt.plot(valid_accuracy_list[:, 0], valid_accuracy_list[:, 1], label =␣
 ↪"Training accuracy", color = 'r')
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



2.0.1 Model evaluation

[ ]:
```
dict_param = {'n_neighbors': np.arange(1, 51),
              'p': np.arange(1, 3),
              'weights': ['uniform', 'distance']}
best_knn_model = GridSearchCV(KNeighborsClassifier(), param_grid = dict_param,␣
 ↪n_jobs = 8, cv = 10, scoring = 'accuracy')

best_knn_model.fit(X_train, y_train)
```

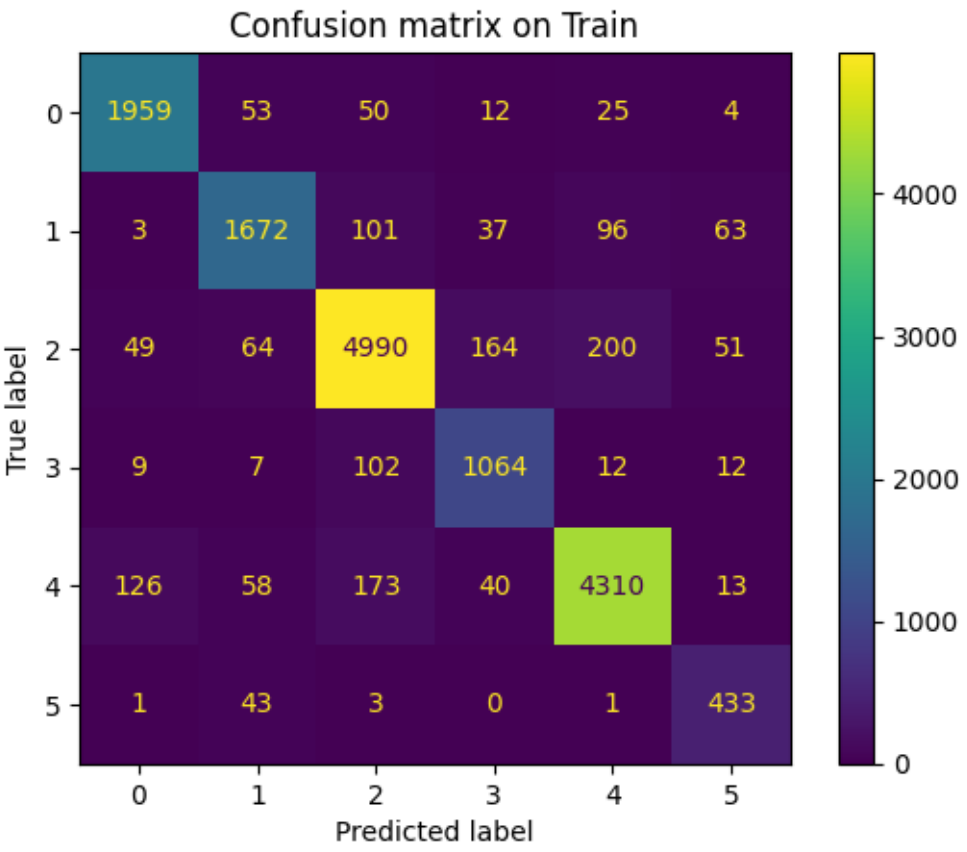
[ ]:
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(), n_jobs=8,
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,
 9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]),
                         'p': array([1, 2]),
                         'weights': ['uniform', 'distance']},
             scoring='accuracy')
```

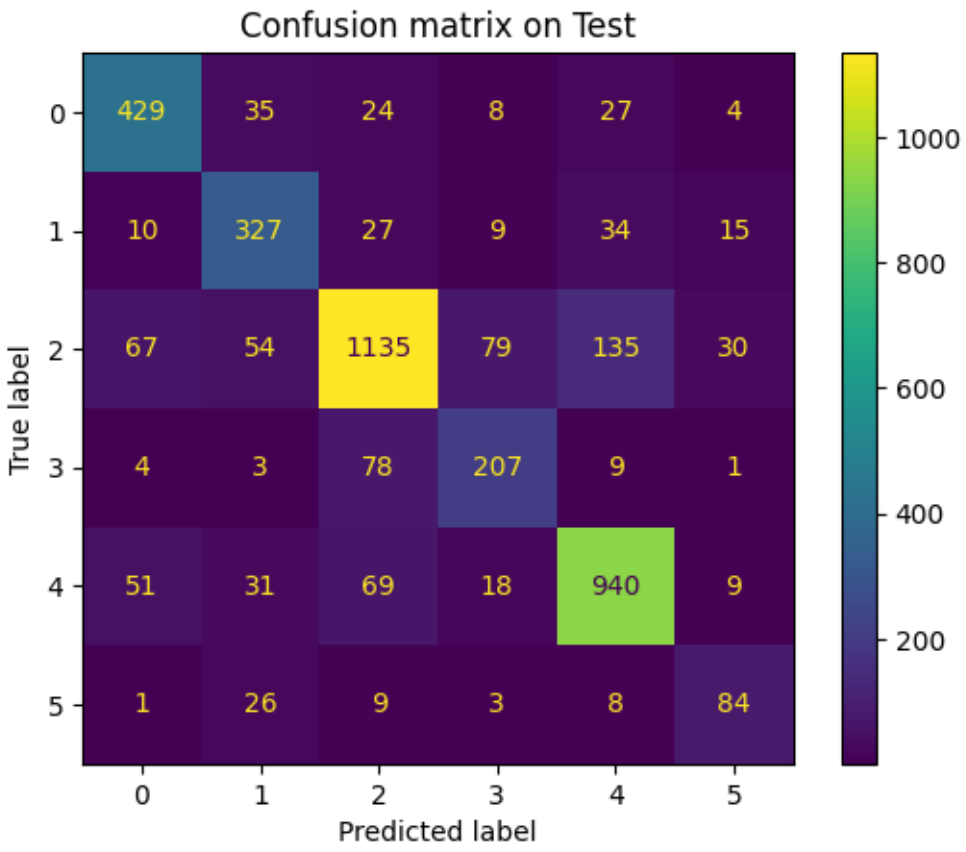
[ ]:
```
print("Best parameters for k_NN on BoW - L1 Dataset:", best_knn_model.
↪best_params_)
```

```
Best parameters for k_NN on BoW - L1 Dataset: {'n_neighbors': 3, 'p': 1,
'weights': 'uniform'}
```

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

```
Score of on train are:
        - Accuracy score: 0.9018
        - Micro F1 score: 0.9018
        - Macro F1 score: 0.8803
Score of on test are:
        - Accuracy score: 0.7805
        - Micro F1 score: 0.7805
        - Macro F1 score: 0.7362
```


Confusion matrix on Train
True label
Predicted label
0
1
2
3
4
5
1959 53 50 12 25 4
3 1672 101 37 96 63
49 64 4990 164 200 51
9 7 102 1064 12 12
126 58 173 40 4310 13
1 43 3 0 1 433
4000
3000
2000
1000
0



3 Export models

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

dump(best_knn_model, directory + "best_knn_model_bow_L1.joblib")
```

[ ]:
```
['data/models/best_knn_model_bow_L1.joblib']
```