Report for Assignment 2: "Text Classification with Linear Classifiers"

Baltzi Sofia (P3351915), Kafritsas Nikos(P3351905), Mouselinos Spyridon (P3351914)
May 20, 2020

0. Introduction

The purpose of this document is to report the results for the 2nd Assignment of class Text Analytics, April – June 2020. Code for the assignment can be found by following the link:
https://colab.research.google.com/drive/1xSfDDj0WrLiWZT_8JzguKASKU02g2fcV?usp=sharing

1. Data

The dataset used in the assignment is the "Twitter Sentiment Analysis 2", found in Kaggle. It includes 99,989 tweets, which belong to either one of two classes: "Positive (1)" or "Negative (0)". Data set is not balanced, having 56,457 tweets belonging to class Positive and 43,532 to class Negative. The average length of tweet is 59 characters. A splitting schema of training (70% - 69,992 tweets), development (15% - 14,998 tweets) and test set (15% - 14,999 tweets) was applied. The ratio of Positive to Negative tweets was preserved while applying the splitting schema.

2. Data Preprocessing

In tweets one can find hashtags, links and contractions, thus, before applying any classification algorithm, dataset was processed by method *preprocess*. This method converts the tweets to lowercase, substitutes contractions with full phrases, removes links, special characters and lemmatizes, if needed, each word.

3. Classification Algorithms

In the next subsections, the classifiers used, as well as the results obtained by each one, will be presented. In order to find the best parameters for each model, grid search was performed on:

- the n-gram range: either unigrams and bigrams or unigrams and trigrams
- the use of idf or not
- min_df
- max_df
- the specifics of the classifier

Parameters were fitted on training set and tuned on development set. The set of parameters that gave the best results in the evaluation phase was used for predictions on test set.

Along with the classifiers used, there will be presented Confusion Matrices, calculated statistics and Learning curves.

3.1. Dummy classifier

As a baseline, a Dummy classifier was used, giving a comparison point for the rest of classifiers. This one replies with the most common of the two classes, giving train accuracy of 56.46% and test accuracy of 56.47%. Weighted F1 score was 41% in both train and test set. Resulting Confusion Matrix on test set is shown below:

Confusion Matrix		Predicted	
		Negative (0)	Positive (1)
Real	Negative (0)	0	6,530
	Positive (1)	0	8,469

3.2. Logistic Regression

The first classifier used was closed-form Logistic Regression. Along with grid search on feature transformation, the hyperparameter C (Inverse of regularization strength) was also tuned. The best values of parameters were found to be:

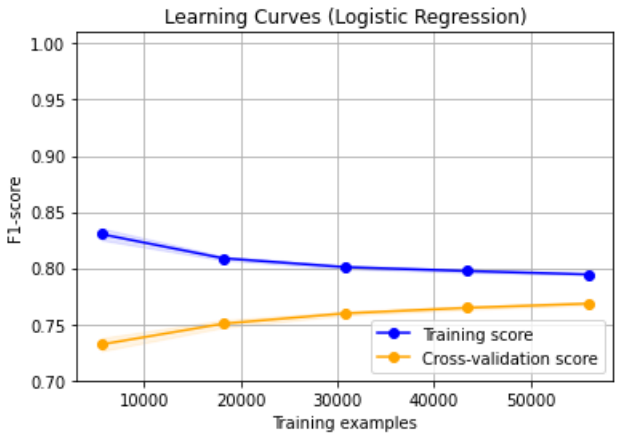
```
{'C': 1.0, 'use_idf': False, 'max_df': 0.7, 'min_df': 0.0, 'vect__ngram_range': (1, 2}
```

Resulting Confusion Matrix on test set is shown below:

Confusion Matrix		Predicted	
		Negative (0)	Positive (1)
Real	Negative (0)	4,664	1,866
	Positive (1)	2,071	6,398

Statistics on train, development and test set, along with Learning curves are presented below:

```
Train Set
train f1-score: 79.33%
              precision    recall  f1-score   support

           0       0.73      0.75      0.74     30472
           1       0.80      0.78      0.79     39520

    accuracy                           0.77     69992
   macro avg       0.77      0.77      0.77     69992
weighted avg       0.77      0.77      0.77     69992


Development Set
dev f1-score: 77.06%
              precision    recall  f1-score   support

           0       0.70      0.72      0.71      6530
           1       0.78      0.76      0.77      8468

    accuracy                           0.74     14998
   macro avg       0.74      0.74      0.74     14998
weighted avg       0.74      0.74      0.74     14998


Test Set
test f1-score: 76.47%
              precision    recall  f1-score   support

           0       0.69      0.71      0.70      6530
           1       0.77      0.76      0.76      8469

    accuracy                           0.74     14999
   macro avg       0.73      0.73      0.73     14999
weighted avg       0.74      0.74      0.74     14999
```



3.3. Logistic Regression SGD

The second classifier used was Logistic Regression with SGD. Along with grid search on feature transformation, the hyperparameters alpha (Constant that multiplies the regularization term) and penalty (Regularization term) were also tuned. The best values of parameters were found to be:

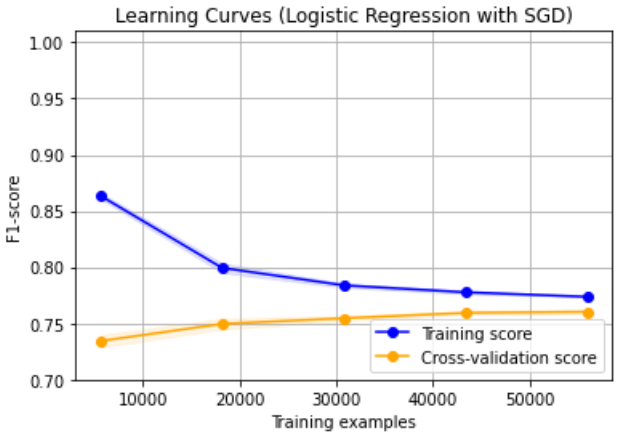
```
{'alpha': 0.0001, 'penalty': 'elasticnet', 'use_idf': True, 'max_df': 0.7, 'min_df': 0.0, 'ngram_range':(1, 2)}
```

Resulting Confusion Matrix on test set is shown below:

Confusion Matrix		Predicted	
		Negative (0)	Positive (1)
Real	Negative (0)	4,715	1,815
	Positive (1)	2,164	6,305

Statistics on train, development and test set, along with Learning curves are presented below:

```
Train Set
train f1-score: 77.21%
              precision    recall  f1-score   support

           0       0.70      0.73      0.72     30472
           1       0.79      0.76      0.77     39520

    accuracy                           0.75     69992
   macro avg       0.74      0.75      0.74     69992
weighted avg       0.75      0.75      0.75     69992


Development Set
dev f1-score: 76.24%
              precision    recall  f1-score   support

           0       0.69      0.72      0.70      6530
           1       0.77      0.75      0.76      8468

    accuracy                           0.74     14998
   macro avg       0.73      0.73      0.73     14998
weighted avg       0.74      0.74      0.74     14998


Test Set
test f1-score: 76.01%
              precision    recall  f1-score   support

           0       0.69      0.72      0.70      6530
           1       0.78      0.74      0.76      8469

    accuracy                           0.73     14999
   macro avg       0.73      0.73      0.73     14999
weighted avg       0.74      0.73      0.74     14999
```



3.4. Naive Bayes

The third classifier used was Naïve Bayes. Along with grid search on feature transformation, the hyperparameter alpha (Additive smoothing parameter) was also tuned. The best values of parameters were found to be:

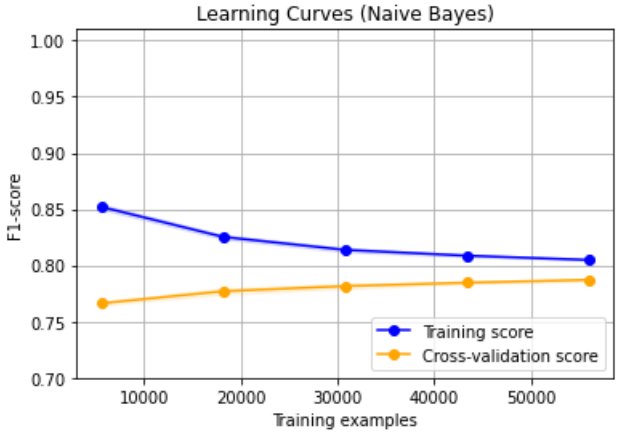
```
{'min_df': 0.0, 'max_df': 0.65, 'ngram_range': (1, 2), 'use_idf': False, 'alpha': 1.0}
```

Resulting Confusion Matrix on test set is shown below:

Confusion Matrix		Predicted	
		Negative (0)	Positive (1)
Real	Negative (0)	4,019	2,511
	Positive (1)	1,456	7,013

Statistics on train, development and test set are presented below:

```
Train Set
train f1-score: 80.31%
              precision    recall  f1-score   support

           0       0.77      0.65      0.70     30472
           1       0.76      0.85      0.80     39520

    accuracy                           0.76     69992
   macro avg       0.77      0.75      0.75     69992
weighted avg       0.76      0.76      0.76     69992


Development Set
dev f1-score: 78.63%
              precision    recall  f1-score   support

           0       0.75      0.62      0.68      6530
           1       0.74      0.84      0.79      8468

    accuracy                           0.74     14998
   macro avg       0.74      0.73      0.73     14998
weighted avg       0.74      0.74      0.74     14998


Test Set
test f1-score: 77.95%
              precision    recall  f1-score   support

           0       0.73      0.62      0.67      6530
           1       0.74      0.83      0.78      8469

    accuracy                           0.74     14999
   macro avg       0.74      0.72      0.72     14999
weighted avg       0.74      0.74      0.73     14999
```



3.5. MLPs

The fourth classifier used was MLP. Along with grid search on feature transformation, the hyperparameter hidden_layer_sizes was also tuned. The best values of parameters were found to be:

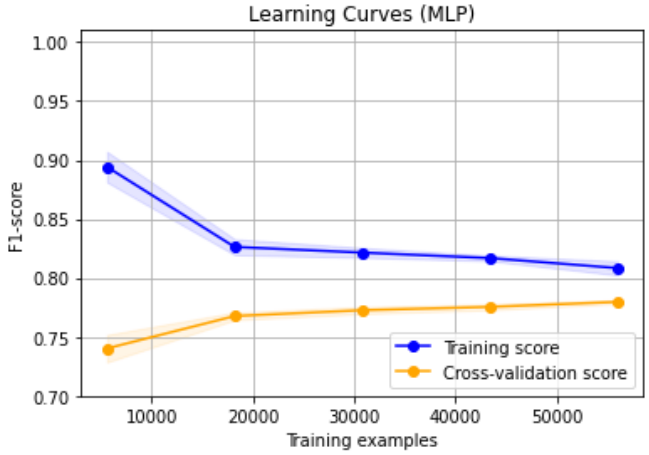
```
{'min_df': 0, 'ngram_range': (1, 2), 'hidden_layer_sizes': (20, 20)}
```

Resulting Confusion Matrix on test set is shown below:

Confusion Matrix		Predicted	
		Negative (0)	Positive (1)
Real	Negative (0)	4,342	2,188
	Positive (1)	1,743	6,726

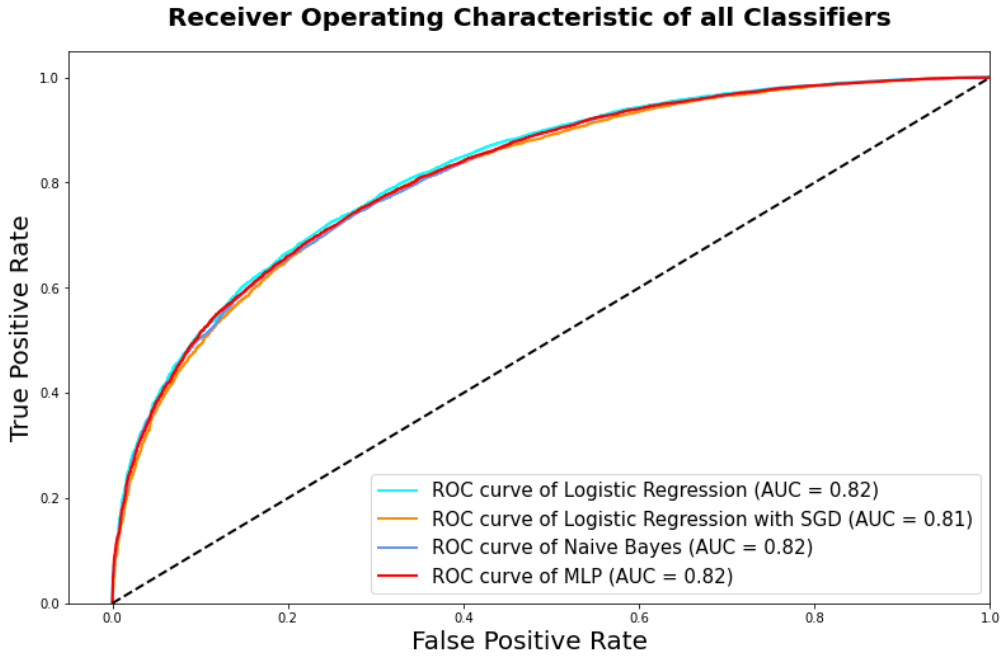
Statistics on train, development and test set are presented below:

```
Train Set
train f1-score: 80.14%
              precision    recall  f1-score   support

           0       0.75      0.70      0.72     30472
           1       0.78      0.82      0.80     39520

    accuracy                           0.77     69992
   macro avg       0.77      0.76      0.76     69992
weighted avg       0.77      0.77      0.77     69992


Development Set
dev f1-score: 78.00%
              precision    recall  f1-score   support

           0       0.72      0.67      0.70      6530
           1       0.76      0.80      0.78      8468

    accuracy                           0.74     14998
   macro avg       0.74      0.74      0.74     14998
weighted avg       0.74      0.74      0.74     14998


Test Set
test f1-score: 77.39%
              precision    recall  f1-score   support

           0       0.71      0.66      0.69      6530
           1       0.75      0.79      0.77      8469

    accuracy                           0.74     14999
   macro avg       0.73      0.73      0.73     14999
weighted avg       0.74      0.74      0.74     14999
```



4. ROC curves

In this section, ROC curves, that graphically depict the diagnostic ability of the classifiers, are presented. Curves show the trade-off between sensitivity (True Positive Rate) and False Positive Rate (1-specificity). Classifiers giving curves closer to the top-left corner indicate a better performance. Thus, the classifiers that were tuned for the purposes of this assignment are compared to one that gives points lying along the diagonal (FPR = TPR). It can be observed that all classifiers perform closely, with Logistic Regression being slightly better.



5. Bootstrapping

Taking all the above scores into consideration, the logistic regression classifier was found to have the highest macro-averaged F1 score on the test set. By employing the method of bootstrap statistical significance testing, this part attempts to check if the apparent differences in terms of macro-averaged F1 score between the logistic regression classifier and the rest of them are due to model superiority or simply by chance. More specifically, there are 2 hypotheses:

A) H_0: The classifier A (logistic regression) is not better than classifier B
B) H_A: The classifier A (logistic regression) is truly better than classifier B

Each comparison estimates a p-value (probability of obtaining an equal or better increase of the macro-averaged F1 score on the test set than the observed one, given that the 2 classifiers being compared have equal potential). The testing part involved creating 50 versions of the test set by sampling with replacement. The p-values are demonstrated below:

Classifier	P-value	Macro averaged f1 score (best= 0.7339)
Logistic Regression with SGD	0.08	0.7317
Naïve Bayes	0	0.7245
Multi Layered Perceptron	0.1	0.7311

The results show that only in the case of Naïve Bayes, the p-value are is less than 0.01, and thus the null hypothesis is rejected in favor of the alternative, which means logistic regression is a truly better classifier. For the other 2 classifiers, there is not enough evidence to believe that their alleged differences from the logistic regression is statistically significant (besides, the difference is very little).