UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Pavlos Ntais*
Classifier: *Softmax Regression*

Course: *Artificial Intelligence II*
Semester: *Fall Semester 2023*

Contents

1. Abstract

The task given is to develop a sentiment classifier using logistic regression for a twitter dataset about the Greek general elections. The classifier deals with 3 classes: POSITIVE, NEUTRAL and NEGATIVE. I plan to analyze the data, clean it by pre-processing and finally start the training process.

2. Data processing and analysis

2.1. Pre-processing

The pre-processing proved to be a rather difficult task because of the vastness of the Greek language. After experimenting I came up with the following methods:

- Remove hashtags
- Remove mentions
- Remove URLS
- Remove excess tabs and white spaces
- Remove numbers and special characters
- Remove stop words
- Remove accents
- Basic emoji replacement
- Insert the party the tweet is being referred to at the beginning of the text
- Lemmatization using the spaCy library

2.2. Analysis

Upon visualizing the dataset, it became apparent that the data had a substantial degree of noise. For example spelling and grammar errors, mentions etc. Also, it naturally contained a lot of stop words, hash tags and emojies which also add noise. So, in order to process the data and remove the noise I removed stop words, hash tags and emojies and used lemmatization to group together inflected or variant forms of a word.

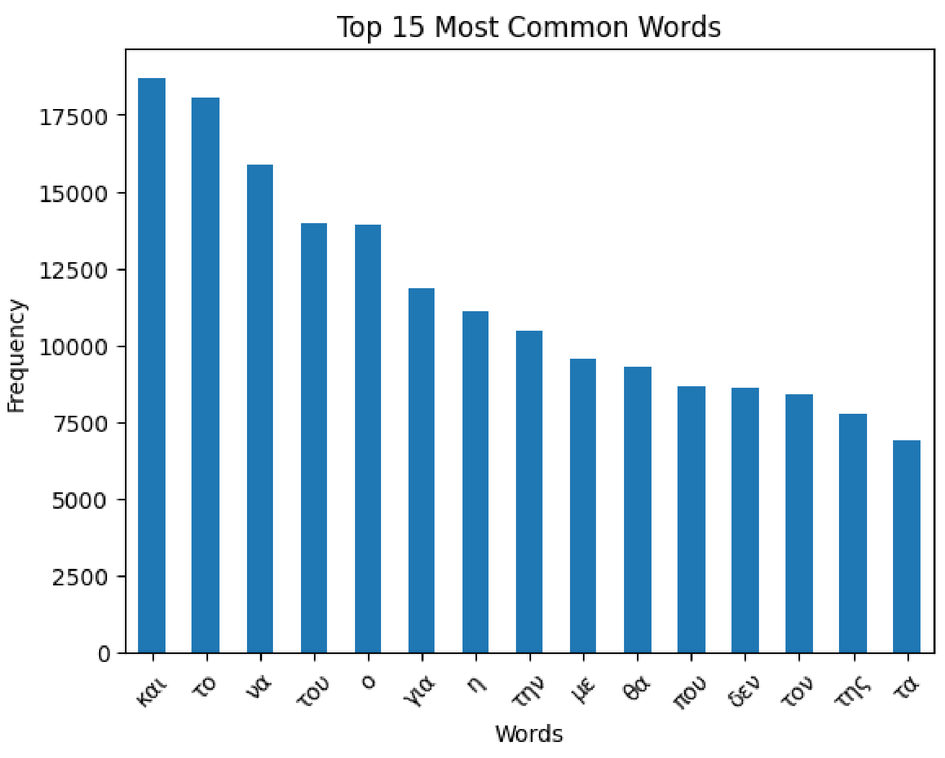

Figure 1: Before pre-processing

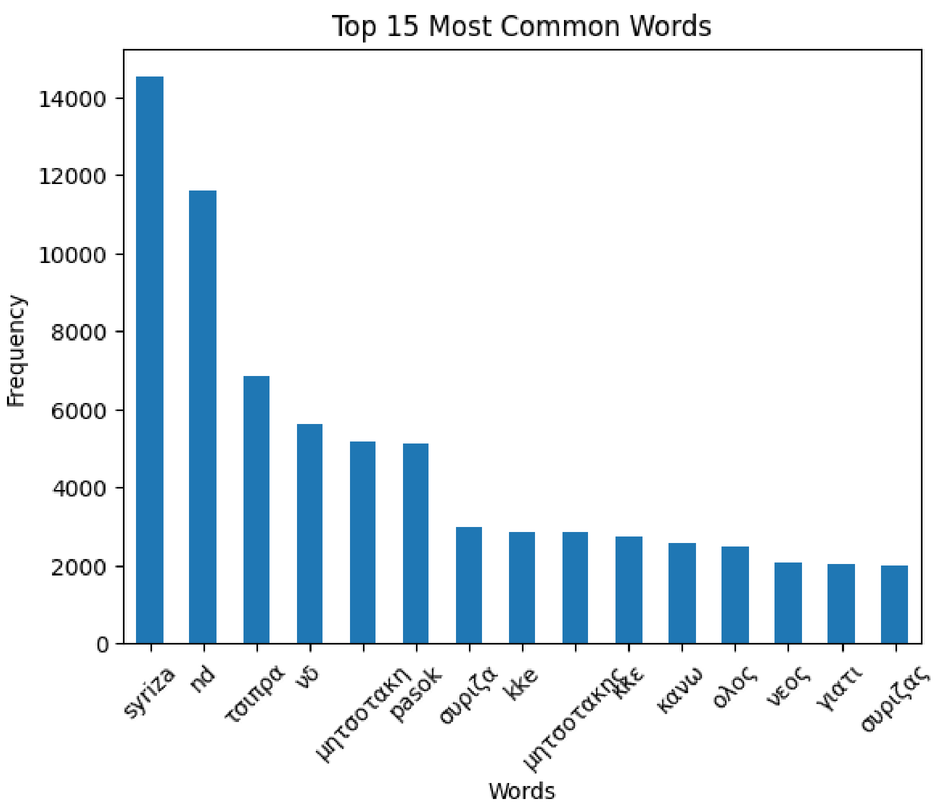

Figure 2: After pre-processing

2.3. Data partitioning for train, test and validation

For the training and testing I used the given datasets because I evaluated that their respective size is sufficient, so no partitioning occurred.

2.4. Vectorization

The following (3) vectorization techniques have been used:

- Count Vectorization
- Hashing
- TF-IDF (Term Frequency-Inverse Document Frequency)

Out of these 3, TF-IDF yielded the best results when it comes to overfitting and performance as will be explored below.

3. Algorithms and Experiments

3.1. Experiments

In order to construct the model along with different vectorization techniques, sklearn's LogisticRegression() was used, with:

- Parameters chosen by experimenting with hyperparameter tuning.
- Multi-class parameter equal to multinomial order in order for the model to support multiple classes.

Trial	Implementation	F1-Score
1	Count Vectorizer	0.3795950272656291
2	Hashing Vectorizer	0.3966363294778783
3	TF-IDF Vectorizer	0.3976494911086968
4	Lbfgs Solver	0.3978028494598377
5	Saga Solver	0.3976050894963219
6	Optuna #1	0.3997980626624540
7	Optuna #2	0.4001665135820380
8	Optuna #3	0.4012370804889413

Table 1: Trials

3.1.1. Table of trials.

3.2. Hyper-parameter tuning & Optimization techniques

For optimization I used the Optuna framework, an open-source tool designed for automating hyperparameter optimization. I primarily fine-tune my model running it (100 iterations), ultimately enhancing its performance. Specifically, ran optuna 3 times, each time reducing the search range in order to find the hyperparameters that give the best results.

3.3. Evaluation

I am evaluating the predictions using 3 metrics:

- ROC curve
  The ROC curve is a graphical representation of a binary classification model's ability to distinguish between positive and negative classes.
- Learning Curve
  A learning curve visualizes how a model's performance improves with more training data using the fscore. Fscore is a performance metric that combines precision and recall into a single value.
- Confusion matrix
  The confusion matrix is a table that quantifies the classification performance of a model by comparing predicted and actual class labels.

The most important thing I'll be taking in mind though, since neither precision nor recall seem to be our priority, is the fscore. It provides a balance between these 2 and makes the model more "generic".

3.3.1. Curves. Here are the curves of the trials showed above:
You have to zoom in order to see the results clearly.

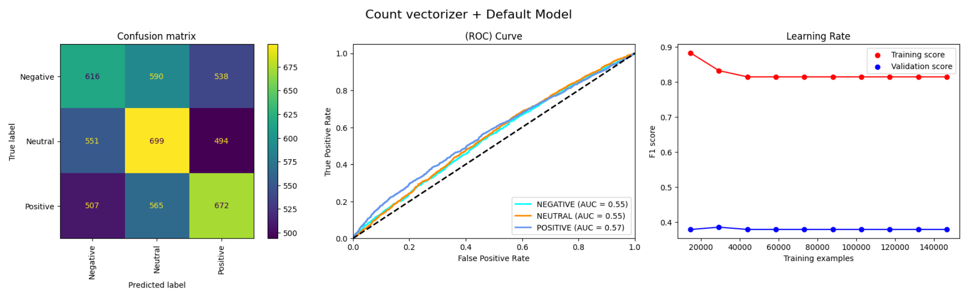

Figure 3: Count vectorizer

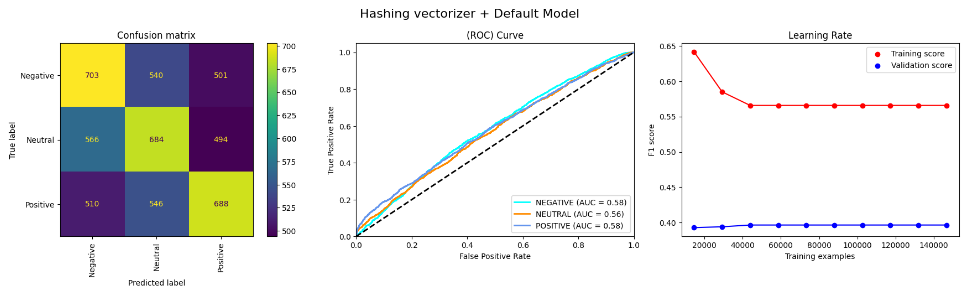

Figure 4: Hash vectorizer

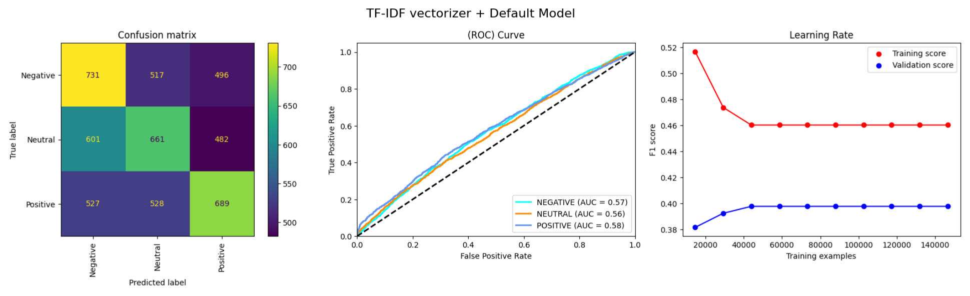

Figure 5: TF-IDF vectorizer

Out of these 3 vectorizers, and using the same default model, we can clearly see that the TF-IDF gave the best results when it comes to overfitting as well as score.

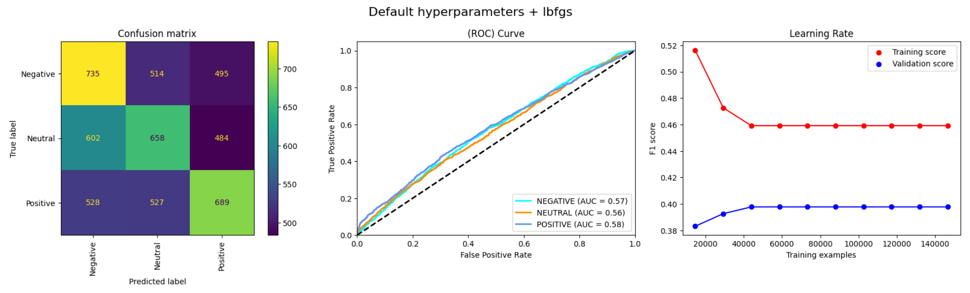

Figure 6: Lbfgs + default hyperparameters

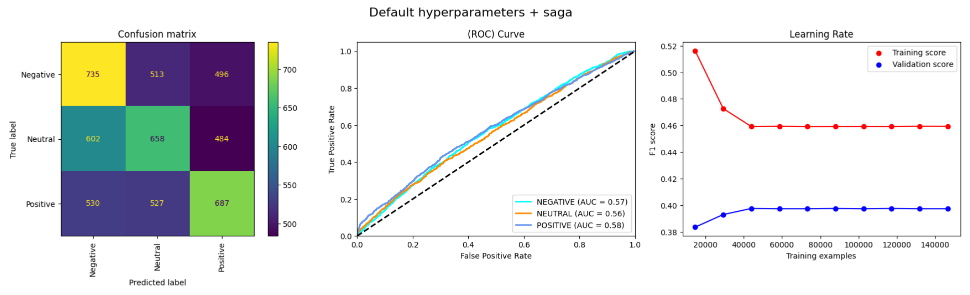

Figure 7: Saga + default hyperparameters

Out of these 2 solvers we can give the lbfgs solver a slight edge when it comes to the score and overfitting. So, I'll be using the lbfgs solver as well the TF-IDF vectorizer to run Optuna.

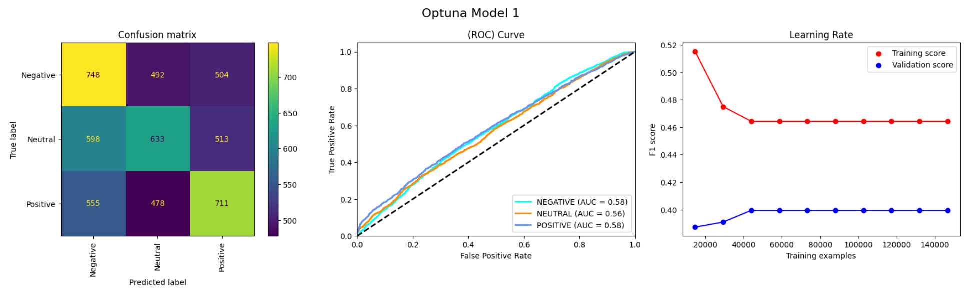

Figure 8: Optuna #1

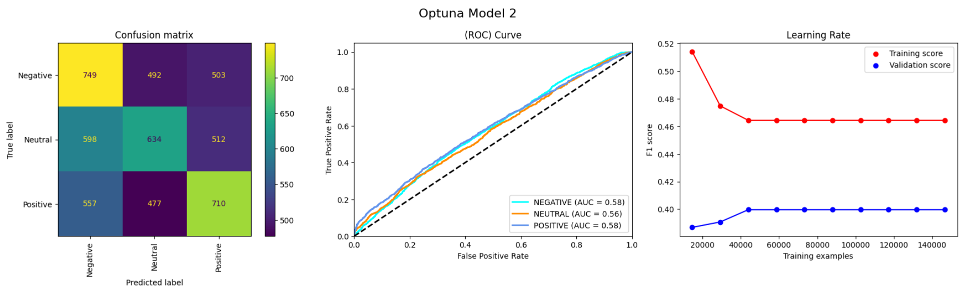

Figure 9: Optuna #2

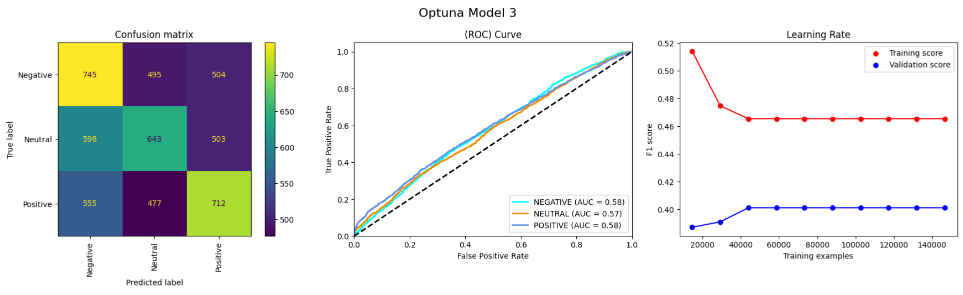

Figure 10: Optuna #3

4. Results and Overall Analysis

4.1. Results Analysis

Overall, although the performance of the model fell short of my expectations, there are several factors contributing to this outcome:

- The dataset presented inherent challenges due to its Greek language content. Greek, being linguistically complex and distinct from widely-used NLP languages, posed inherent difficulties for the model. Also, the limited availability of good performing Greek NLP libraries posed a significant challenge.
- The dataset itself may have contributed to the suboptimal performance. A good number of labels were not correct thus hindering the ability of the model to learn.
- While logistic regression has its merits, it might not be the most powerful option for intricate NLP tasks such as classifying political opinions correctly.

4.1.1. Best trial. Ultimately my best trial came after running hypertuning optimization using the TF-IDF vectorizer and Lbfgs solver.

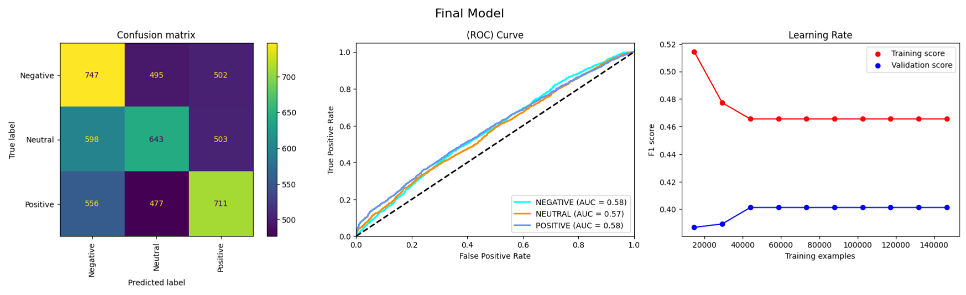

Figure 11: Final Model

5. Bibliography

References

[1] Stack Overflow. Stack overflow q&a. https://stackoverflow.com/.

[2] Scikit-learn. Scikit-learn documentation. https://scikit-learn.org/stable/.

[3] Stanford. Speech and language processing. https://web.stanford.edu/~jurafsky/slp3/.

[3] [1] [2]