UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: <Μιχαήλ Χρήστος Ναβάρο Αμαργιανός>
sdi: <*sdi2000151*>

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Fall Semester 2023*

Contents

1. Abstract

In this task we were ask to develop a sentiment classifier using logistic regression for a twitter dataset about the Greek general elections. Each row in the dataset contains an ID for each tweet, a sentiment label which can be POSITIVE, NEUTRAL or NEGATIVE, the text of the tweet, and the party the tweet is referring to. Our model should deal with 3 classes: POSITIVE, NEUTRAL, NEGATIVE. To tackle with this problem firstly I apply the pre-processing and analysis of the given data. After this I choose the metrics and I do some experiments with the parameters to find the best result

2. Data processing and analysis

2.1. Pre-processing

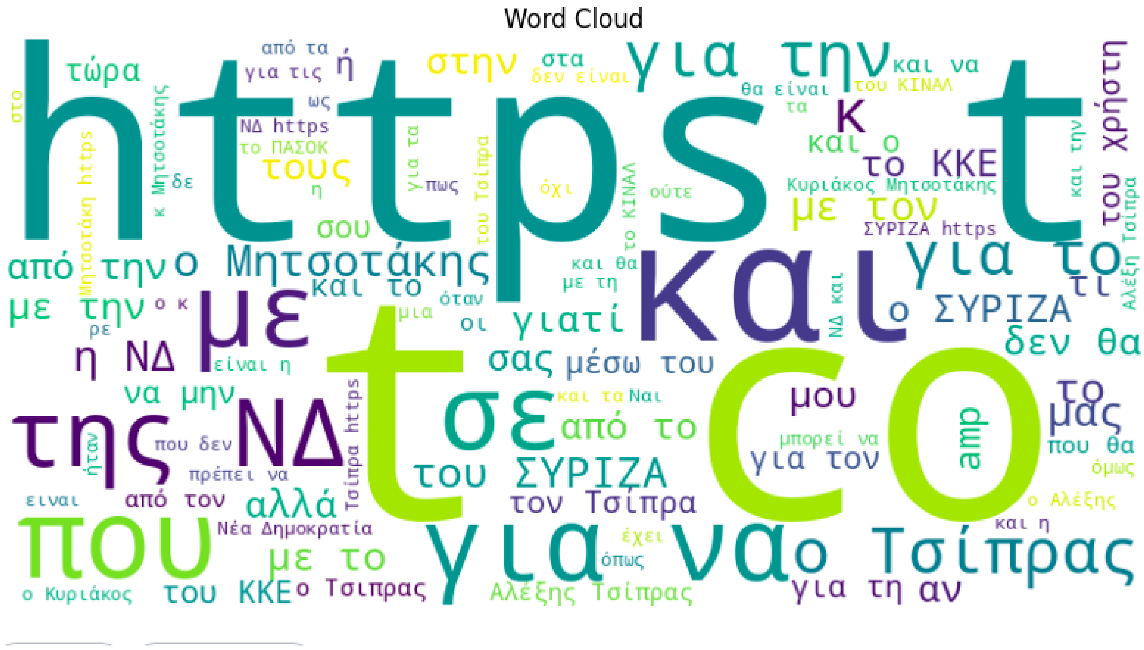
Data cleaning and regularisation are very important steps since they simplify and help the model distinguish the basic characteristics of each class. Firstly, I process the column with the "Text" according to some rules.

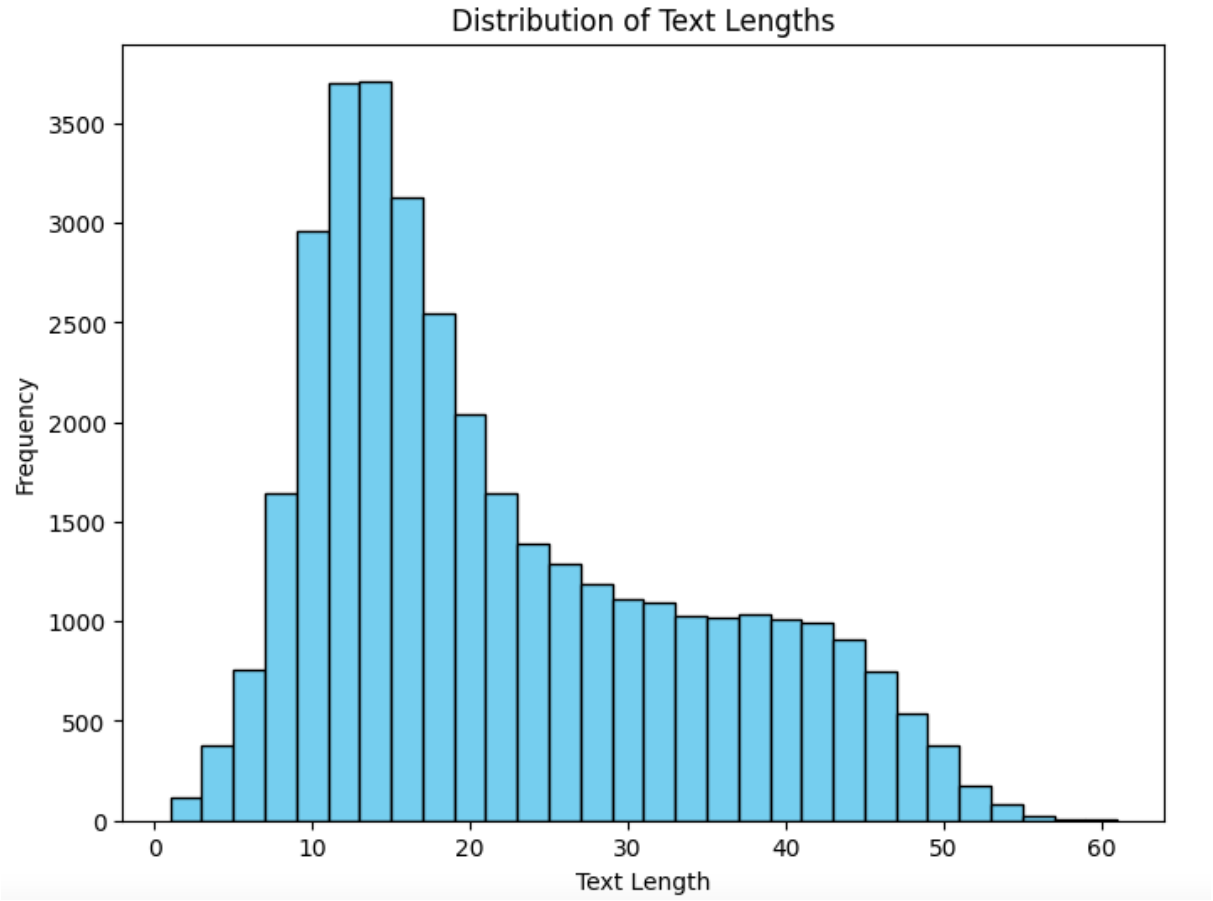
- Removing mentions, hashtags and links. These part of the text are not so important for the final conclusion.
- Converting capital to lowercase letters. Avoiding duplicates and limiting on the ascii codes of only lowercase characters
- Removing accends.
- Removing stop words. (https://www.translatum.gr/forum/index.php?topic=3550.0)
- Removing punctuation.

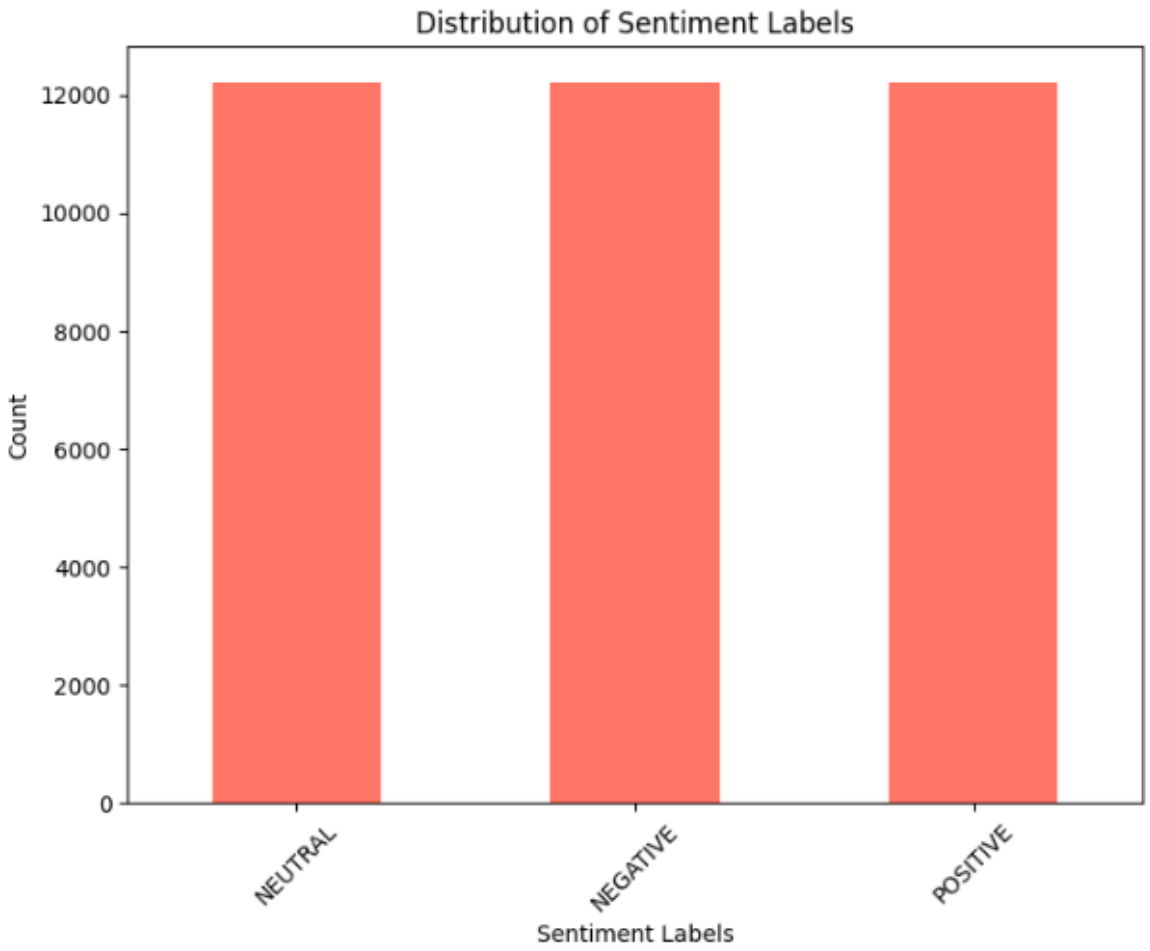
I had as a guide this site to get ideas https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/
For the pre-processed text we will use too the technique of Stemming as described in https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html, although in our problem the text is in greek language so we use "greek_stemmer" library

2.2. Analysis

In this step I create word clouds to visualize the most frequent words or terms in the text data and also I use visualizations like bar charts and histograms to display the distribution data and sentiment label.


Word Cloud
https
t
co
και
σε
με
της
ΝΔ
που
για
να
ο Τσίπρας
για την
για το
τώρα
στην
στα
δεν είναι
και να
του ΚΙΝΑΛ
θα είναι
τα
κ
και ο
το ΚΚΕ
του χρήστη
τους
σου
με τον
Κυριάκος Μητσοτάκης
ΣΥΡΙΖΑ https
από την
ο Μητσοτάκης
και το
ο ΣΥΡΙΖΑ
δεν θα
με την
η ΝΔ
γιατί
οι
σας
μέσω του
μου
από το
το
μας
του ΣΥΡΙΖΑ
για τον
amp
να μην
αλλά
από τον
τον Τσίπρα
που θα
όμως
ο Αλέξης
Νέα Δημοκρατία
με το
του ΚΚΕ
ο Τσιπρας
Αλέξης Τσίπρας
για τη
αν
ο Κυριάκος
ειναι
πρέπει να
που δεν
μπορεί να
και η
έχει
όπως
και τα
Ναι
μια
όταν
είναι η
ρε
ο κ
κ Μητσοτάκης
από τα
για τις
ή
ως
ΝΔ https
το ΠΑΣΟΚ
η
δε
πως
όχι
ούτε
για τα
του Τσίπρα
το ΚΙΝΑΛ
και θα
με τη
ΝΔ και
τι
Αλέξη Τσίπρα
και την
στο
Μητσοτάκη https
Τσίπρα https
ήταν


Distribution of Text Lengths
Frequency
Text Length
0
500
1000
1500
2000
2500
3000
3500
0
10
20
30
40
50
60


Distribution of Sentiment Labels
12000
10000
8000
6000
4000
2000
0
Count
NEUTRAL
NEGATIVE
POSITIVE
Sentiment Labels

2.3. Data partitioning for train, test and validation

To manipulate the data for train, test and validation, I create a Y label for the column "Sentiments". Although, this column has valuesof type "string" so I encoded them into type "int" (the encoding was done by using the library "sklearn.preprocessing" and the function "LabelEncoder")

2.4. Vectorization

For passing the "Text" data to the model we have to convert it firstly to a vector. So, for vectorization I choose the technique of Tfidf_Vectorizer of the libraty "sklearn.feature_extraction.text' This is a good method because it's a combiantion of count vectorization with the TFIDF transformer. (https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weightir)

3. Algorithms and Experiments

3.1. Experiments

Experiments have been done in the stage of "Labeling" and "Vectorization".

- I tried to merge the two columns of "Text" and "Party" to give as input to the vectorizer, but finally this experiment didn't return good results.
- Also I experimented by using two differents vectorizers the Count-Vectorizer and the Tfidf-Vectorizer. The conclusion is that in our problem the Tfidf-Vectorizer gives better results (f1_score: 0.3817)

All the experiment are commented in the main code.

3.2. Hyper-parameter tuning

To tune the hyper parameters of Logistic Regression I test various parameters by manual way. In order to avoid the variation between runs, I use everywhere the same random state.
The tests are done by comparing the score cross validation returns. https://scikit-learn.org/stable/modules/cross_validation.html

- Tolerance: I tried various values for tolerance but none made an important difference
- Inverse of regularisation strength: the conclusion is that the smaller value this parameter has the better result it gives. So i kept the value of 0.1
- Solver Parameter: I tried the solver 'newton-cg', 'lbfgs', 'sag', 'saga'. Solver 'saga' gives the best results.
- Iterations: I tried various values for the number of iterations but none made an important difference

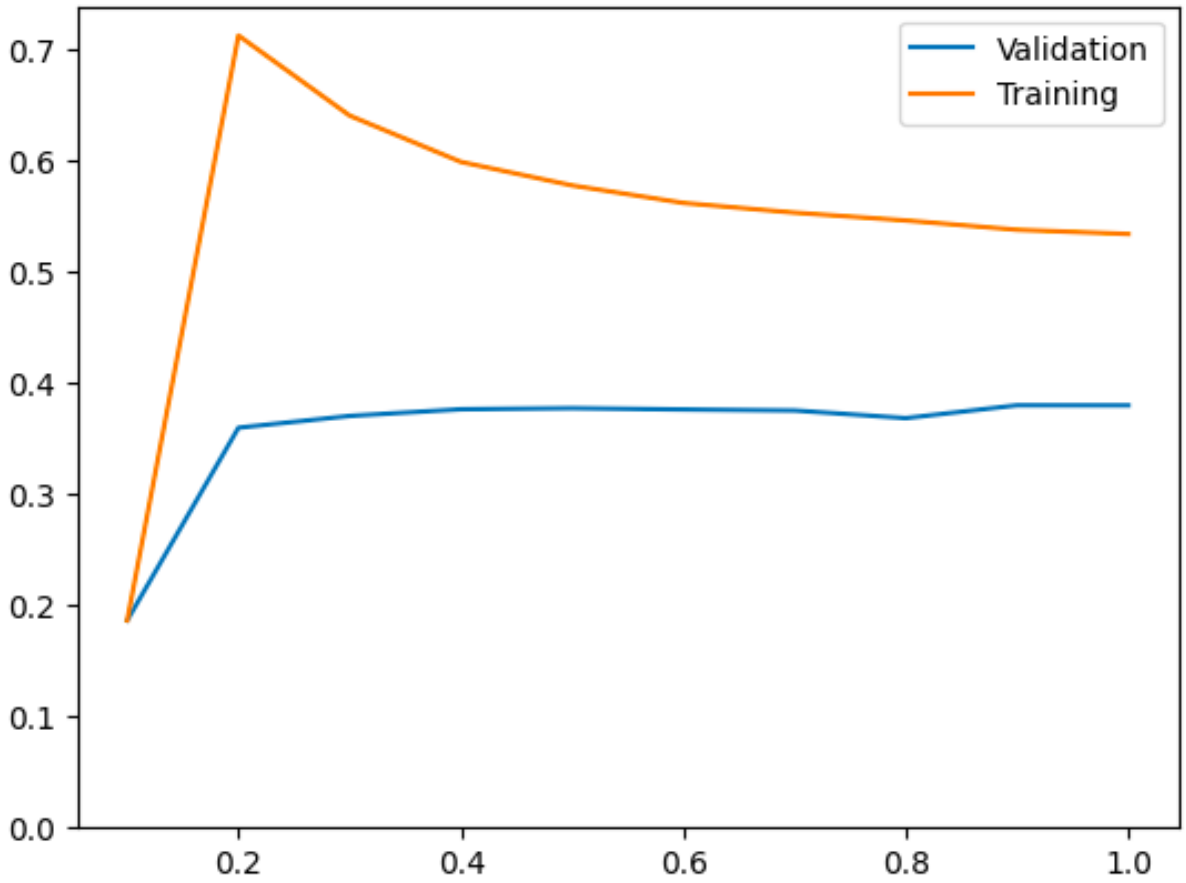
3.3. Optimization techniques

In this assignment I didn't use any optimization frameworks. I check manually the parameters of the Logistic Regression function and I kept the value that gave the optimal result.

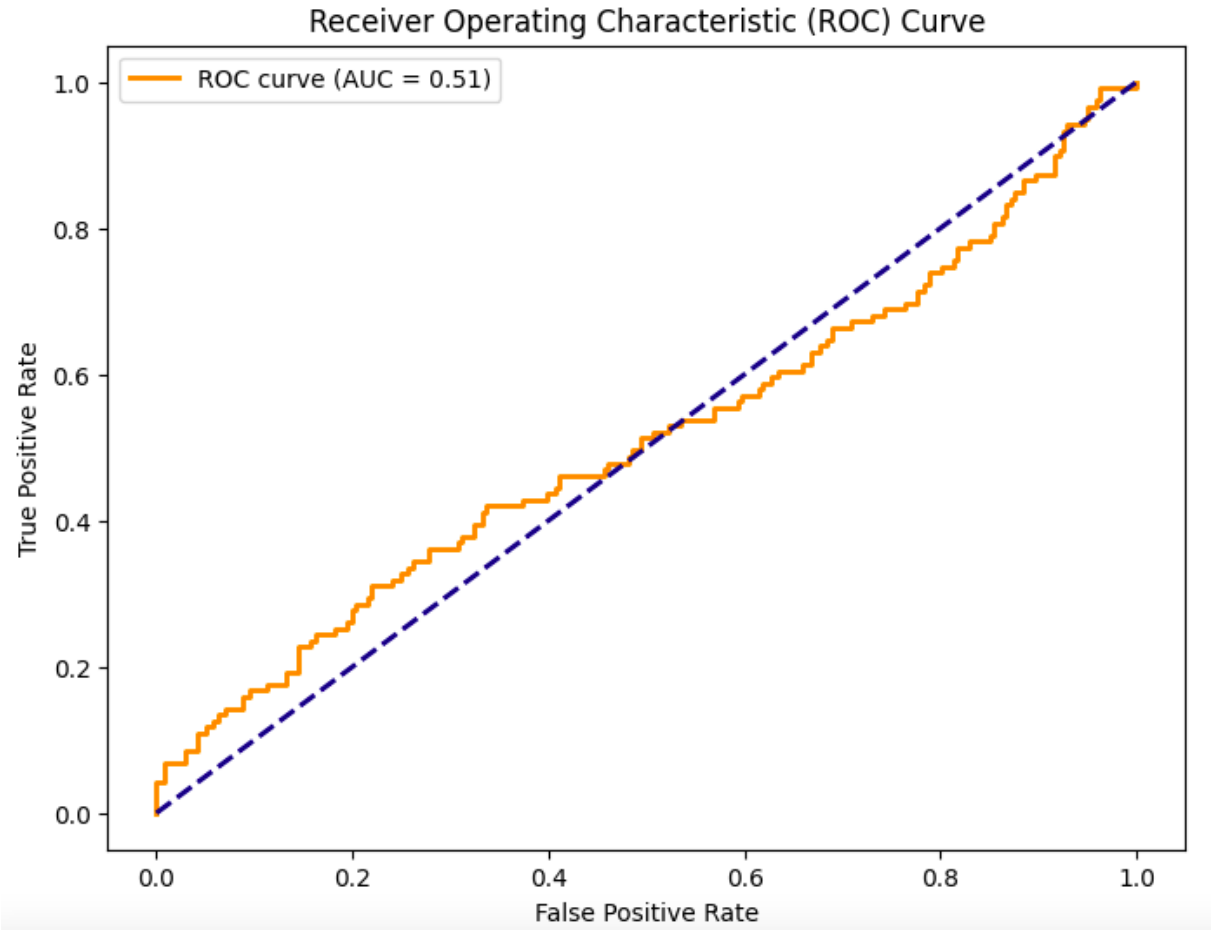
3.4. Evaluation

I evaluate the predictions using the F1 score function https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html .
This is a suitable metric, because robust when dealing with imbalanced class distributions and it also provides a reliable measure of model performance by considering false positives and false negatives equally. A notable point about F1 score is that it condenses the evaluation of a model's performance into a single metric

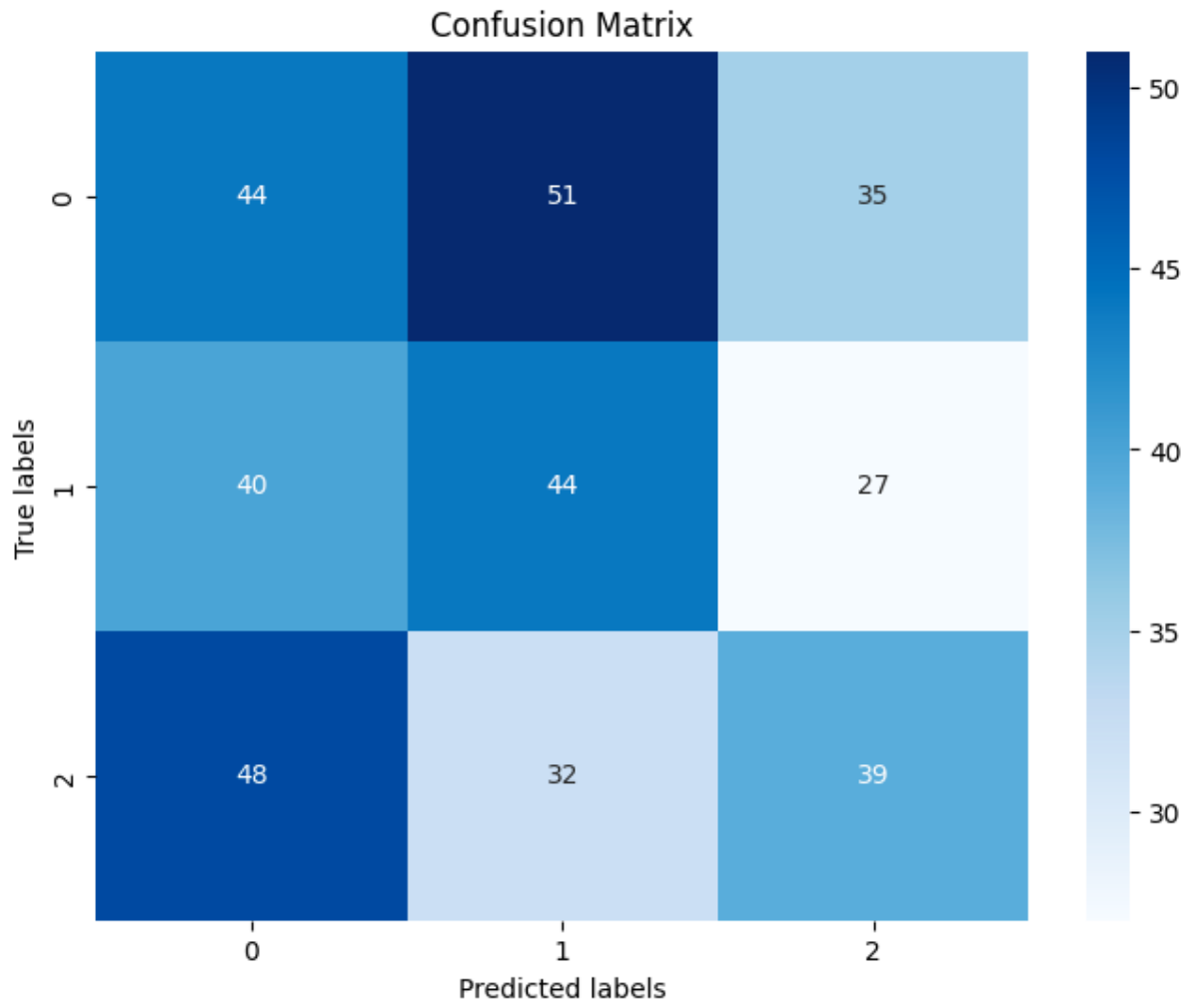
```
F1 Score (Training): 0.1867
F1 Score (Validation): 0.1867
F1 Score (Training): 0.7129
F1 Score (Validation): 0.3599
F1 Score (Training): 0.6408
F1 Score (Validation): 0.3706
F1 Score (Training): 0.5991
F1 Score (Validation): 0.3766
F1 Score (Training): 0.5780
F1 Score (Validation): 0.3775
F1 Score (Training): 0.5622
F1 Score (Validation): 0.3764
F1 Score (Training): 0.5535
F1 Score (Validation): 0.3755
F1 Score (Training): 0.5464
F1 Score (Validation): 0.3686
F1 Score (Training): 0.5381
F1 Score (Validation): 0.3801
F1 Score (Training): 0.5343
F1 Score (Validation): 0.3801
[61… <matplotlib.legend.Legend at 0x7ebcb51a2650>
```



3.4.1. Learning Curve.



3.4.2. ROC curve.



3.4.3. Confusion matrix.

4. Results and Overall Analysis

4.1. Results Analysis

According to the results of F1 score and our plot, the conclusion is that our model doesn't work very well. The score we get (0.38) is very low, that could be caused by a lot of factors:

- **Quality of Text**. Text noise, misspellings, abbreviations, or slang in the Greek language tweets can cause difficulties to do an accurate sentiment analysis.
  Label Quality is also an important factor. The labeling of sentiments could be inaccurate or subjective.
- **Linguistic Complexity**. Greek language is a very complex language with it's nuances, word morphology, or unique sentence structures might require specialized pre-processing or feature extraction techniques.
- The selected machine learning algorithm might not be the most appropriate for sentiment analysis in Greek text, so the model suitability it's not the best.

4.2. Learning Curve

The learning curve indicates the relationship between the model's performance and the training data. From the curve we perceive that the model it's not benefit significantly from training data, indicating underlying issues with the model's complexity or representation.

4.3. ROC Curve

The ROC curve illustrates the trade-off between true positive rate and false positive rate at a variety of thresholds. An almost linear diagonal ROC curve (resembling the x=y line) implies the model performs no better than random guessing for classifying sentiments. It indicates that the model's discrimination ability is poor and is not effective in distinguishing between different sentiment classes.

4.4. Confusion matrix

A confusion matrix illustrates a detailed breakdown of the model's predictions compared to the actual true sentiment labels. In the confusion matrix, we observe an uneven distribution of correctly and incorrectly instances of encoded label sentiments. This means that there is a significant number of false positives and false negatives, indicating the model's inability to correctly predict sentiments.

5. Bibliography

References

https://scikit-learn.org/stable/index.html https://www.translatum.gr/forum/index.php?topic=3550.0

https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/
https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html
https://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting
https://scikit-learn.org/stable/modules/cross_validation.html

https://www.w3schools.com/python/python_ml_auc_roc.asp
https://www.analyticsvidhya.com/blog/2021/08/understanding-bar-plots-in-python-beginners-g