UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: Μιχαήλ Χρήστος Ναβάρο Αμαργιανός
sdi:2000151

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

Contents

1. Abstract

In this homework, I develop a sentiment classifier using deep neural networks for the Twitter dataset about the Greek general elections. For the input data I use Word2Vec word embeddings before pass it to the machine learning framework PyTorch I was ask to use.

2. Data processing and analysis

2.1. Pre-processing

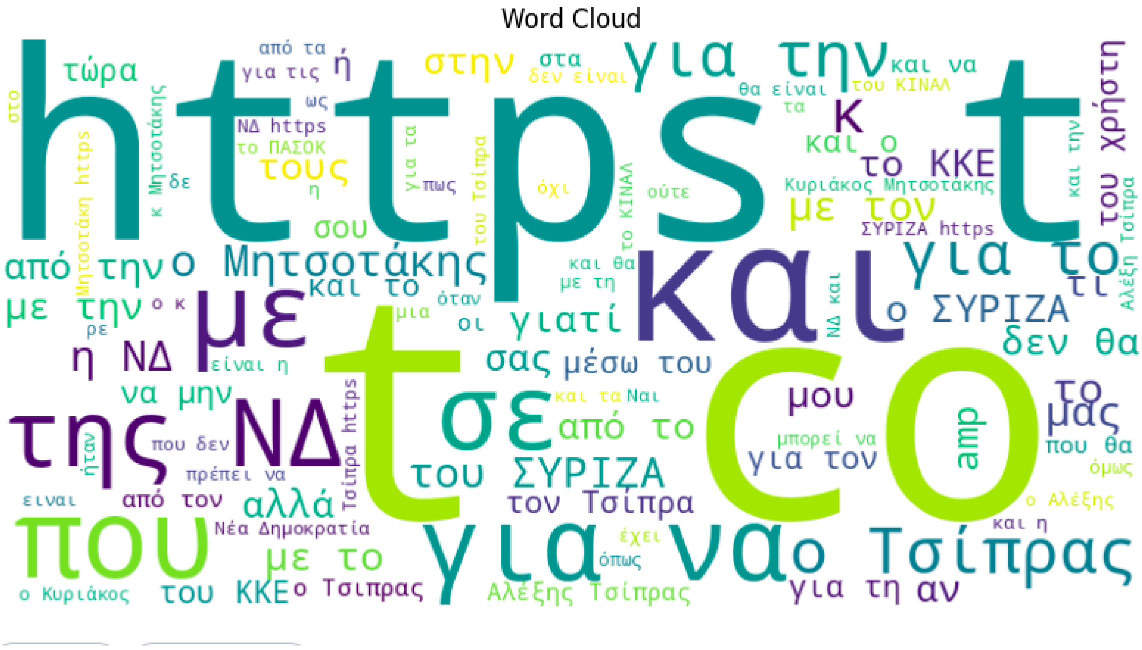
Data cleaning and regularisation are very important steps since they simplify and help the model distinguish the basic characteristics of each class. Firstly, I process the column with the "Text" according to some rules.

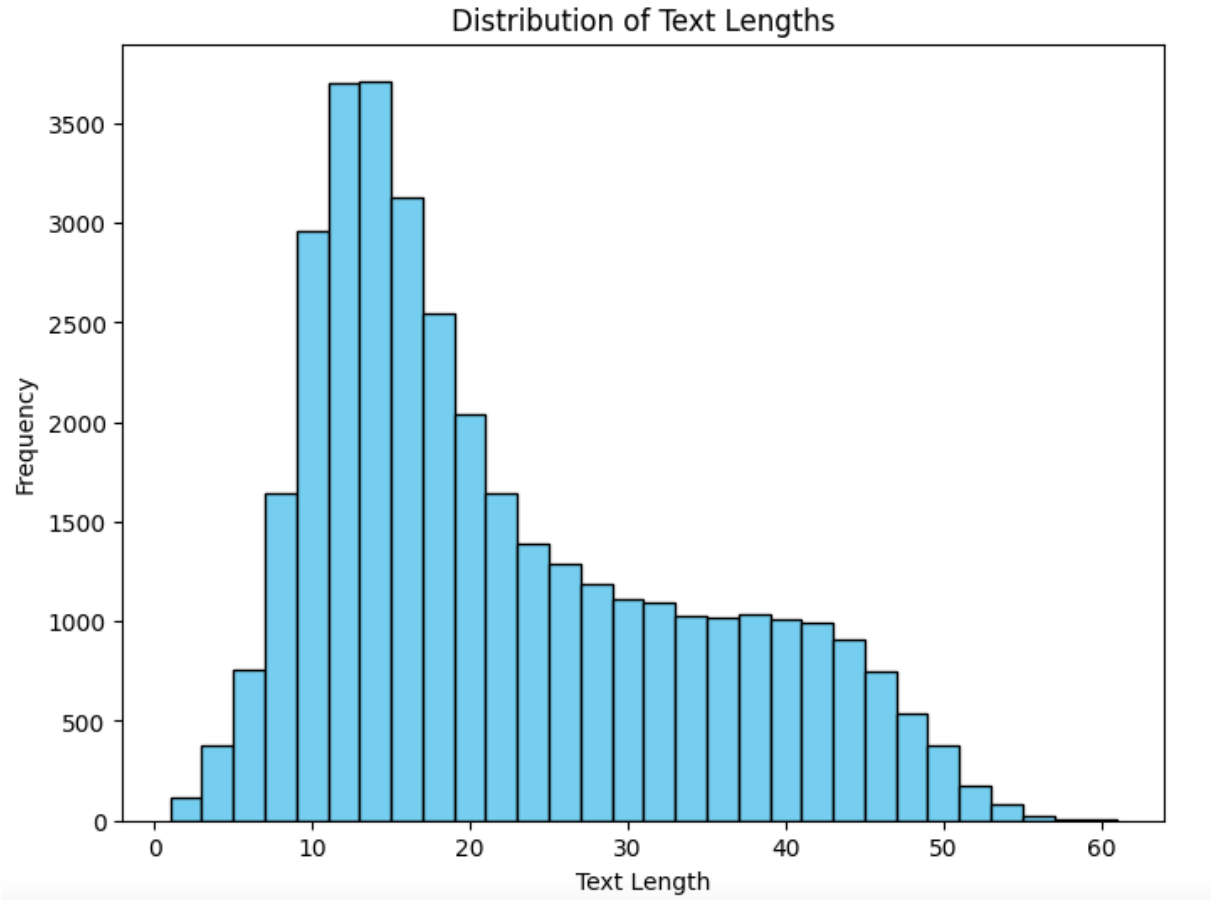
- Removing mentions, hashtags and links. These part of the text are not so important for the final conclusion.
- Converting capital to lowercase letters. Avoiding duplicates and limiting on the ascii codes of only lowercase characters
- Removing accends.
- Removing stop words. (https://www.translatum.gr/forum/index.php?topic=3550.0)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

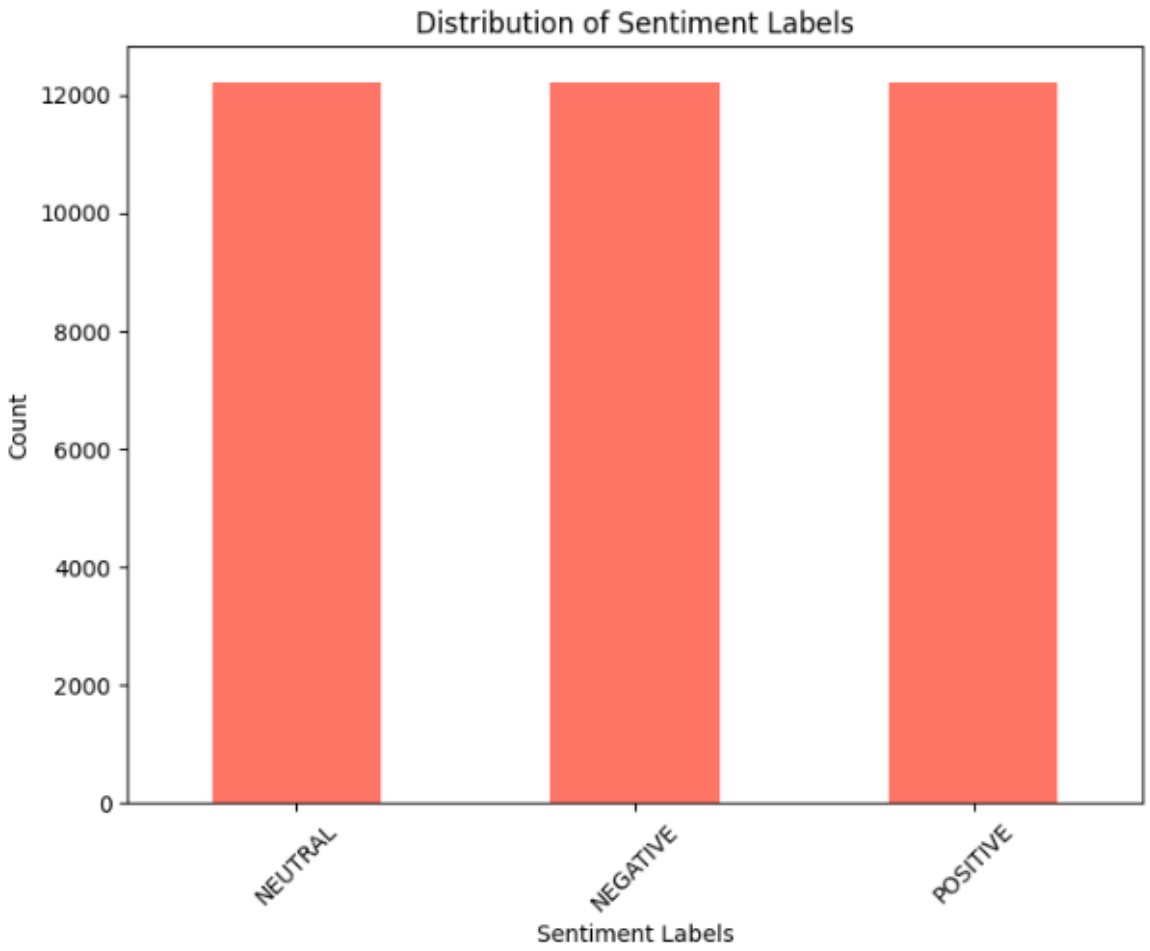
I had as a guide this site to get ideas https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/
For the pre-processed text we will use too the technique of Stemming as described in https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html, although in our 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
με
της
ΝΔ
σε
που
για
να
για την
ο Τσίπρας
για το
του ΣΥΡΙΖΑ
ο Μητσοτάκης
ο ΣΥΡΙΖΑ
το ΚΚΕ
με τον
από την
με την
η ΝΔ
τώρα
στην
στα
και να
του χρήστη
τους
σου
και το
γιατί
σας
μέσω του
δεν θα
μου
μας
από το
για τον
amp
αλλά
τον Τσίπρα
με το
του ΚΚΕ
ο Τσιπρας
Αλέξης Τσίπρας
για τη
αν
να μην
K
και ο
ΣΥΡΙΖΑ https
Κυριάκος Μητσοτάκης
ΝΔ https
το ΠΑΣΟΚ
του ΚΙΝΑΛ
Νέα Δημοκρατία
ο Κυριάκος
ο Αλέξης
πρέπει να
μπορεί να
που δεν
που θα
όμως
όπως
έχει
είναι η
δεν είναι
θα είναι
από τα
για τις
ή
ως
πως
όχι
ούτε
όταν
μια
οι
τι
το
ρε
ήταν
ειναι
και η
και θα
με τη
και τα
Ναι


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-weightin)

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.

Also, another experiment is to change the activation function. Instead of use only the ReLU function, I experiment too with the LeakyReLU, ELU, Sigmoid functions.
Finally, the ReLU gives better results

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 and I check all the value in a nested loop.
The tests are done by comparing the score cross validation returns. https://scikit-learn.org/stable/modules/cross_validation.html

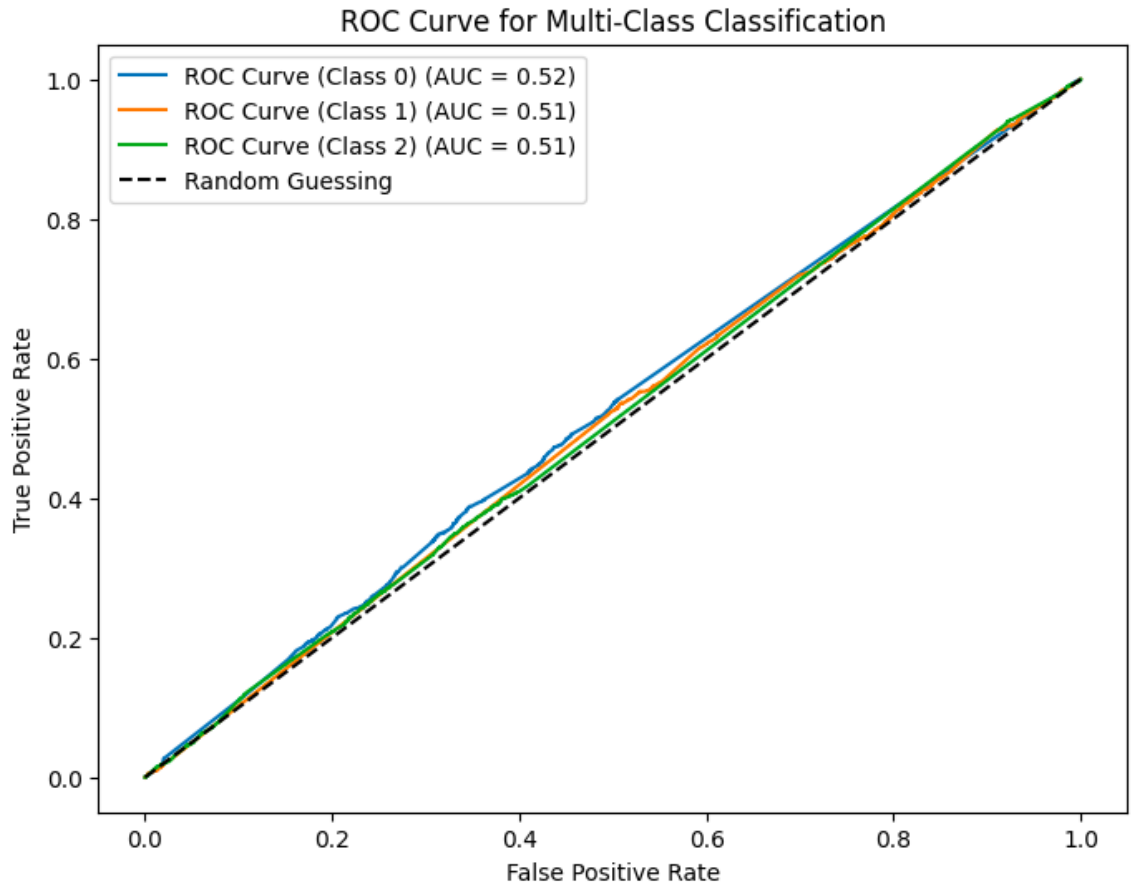
3.3. Optimization techniques

For optimization I use the methods of ReduceLROnPlateau scheduler and early stopping. ReduceLROnPlateau reduces learning rate when a metric has stopped improving and early stopping stops training when the model's performance on a validation

set starts to degrade, preventing overfitting. If there's no improvement in consecutive epochs, it stops the training loop.

3.4. Evaluation

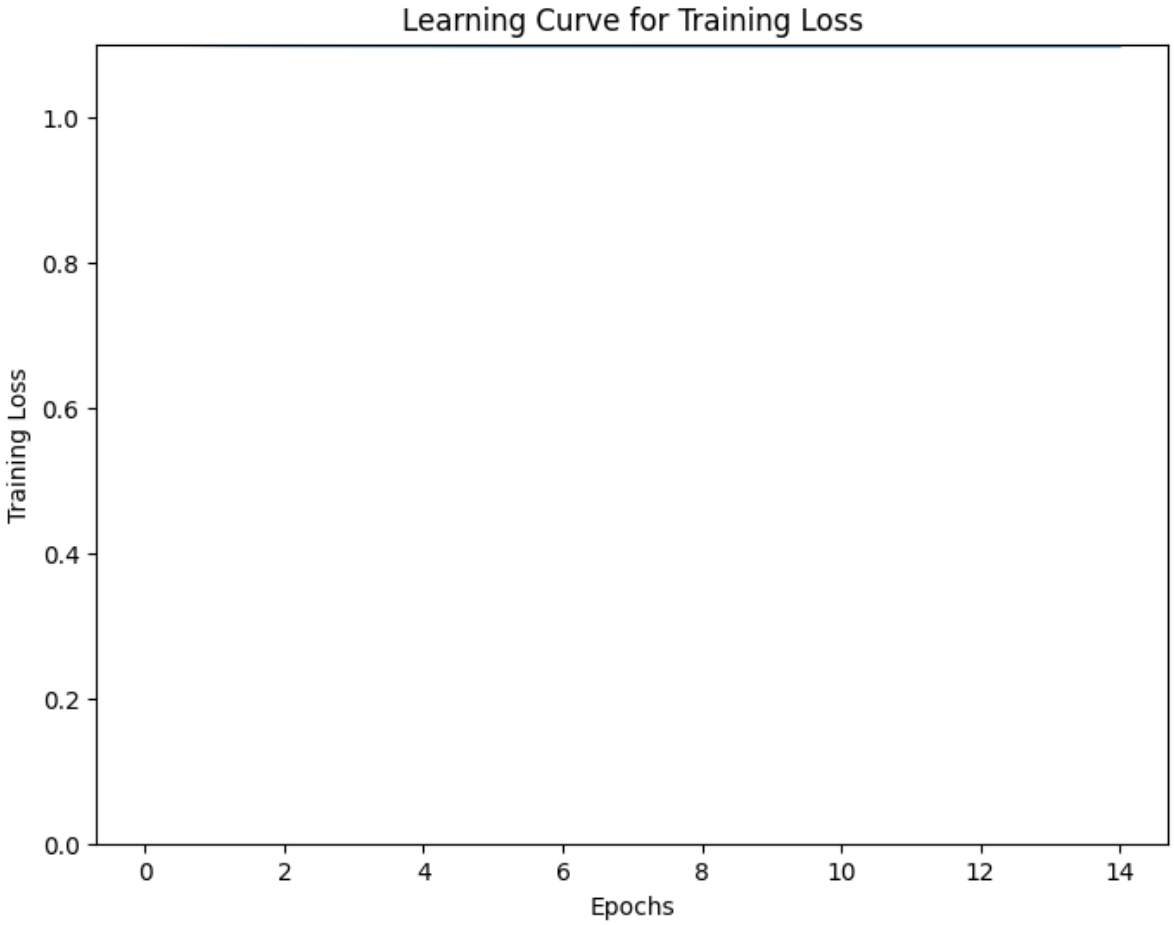
I evaluate the predictions using the F1 score, precision, recall function.
https://scikit-learn.org/stable/modules/model_evaluation.html#the-scoring-parameter-defining-
F1 score considers both precision and recall, making it robust for imbalanced datasets. It balances the trade-off between precision and recall, providing a harmonic mean that is suitable for imbalanced classes. Tweets often exhibit class imbalance, where the number of instances belonging to different sentiment classes (positive, negative, neutral) can be various.
Precision and recall capture different aspects of misclassifications. Precision focuses on the correctness of positive predictions, while recall emphasizes the ability to capture all positive instances. In sentiment analysis, misclassifying tweets can have different implications. For instance, misclassifying a strongly negative tweet as positive can significantly impact the sentiment analysis result.
So, in the problem of sentiment analysis with tweets, where the classes might be imbalanced and the correct identification of positive and negative sentiment tweets is essential, using F1 score, precision, and recall provides a more better understanding of the model's performance. https://medium.com/artificial-corner/going-beyond-accuracy-a-deep-dive-



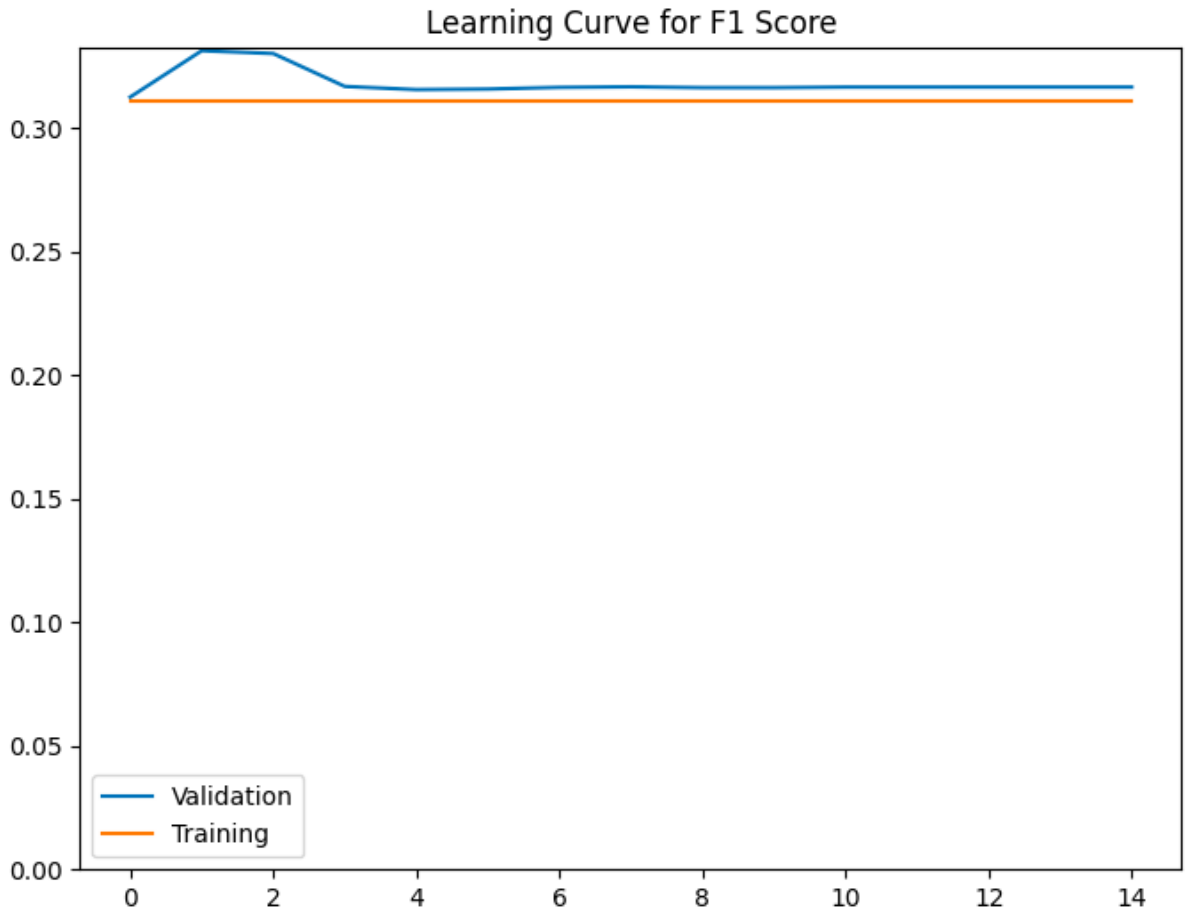
3.4.1. ROC curve.

3.4.2. Learning Curve.

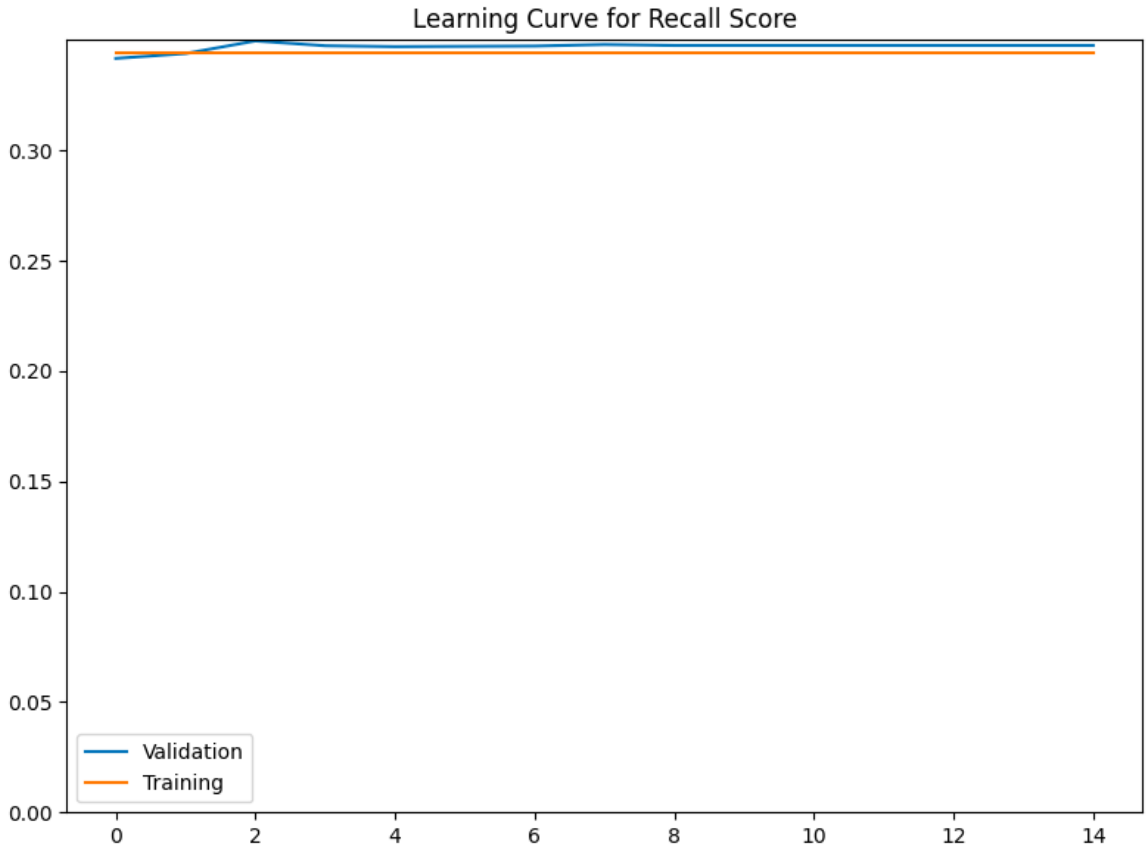
1. Learning Curve of Training Loss (***There is a blue line in the top border of the squere that indicates the Training Loss Learning Curve)



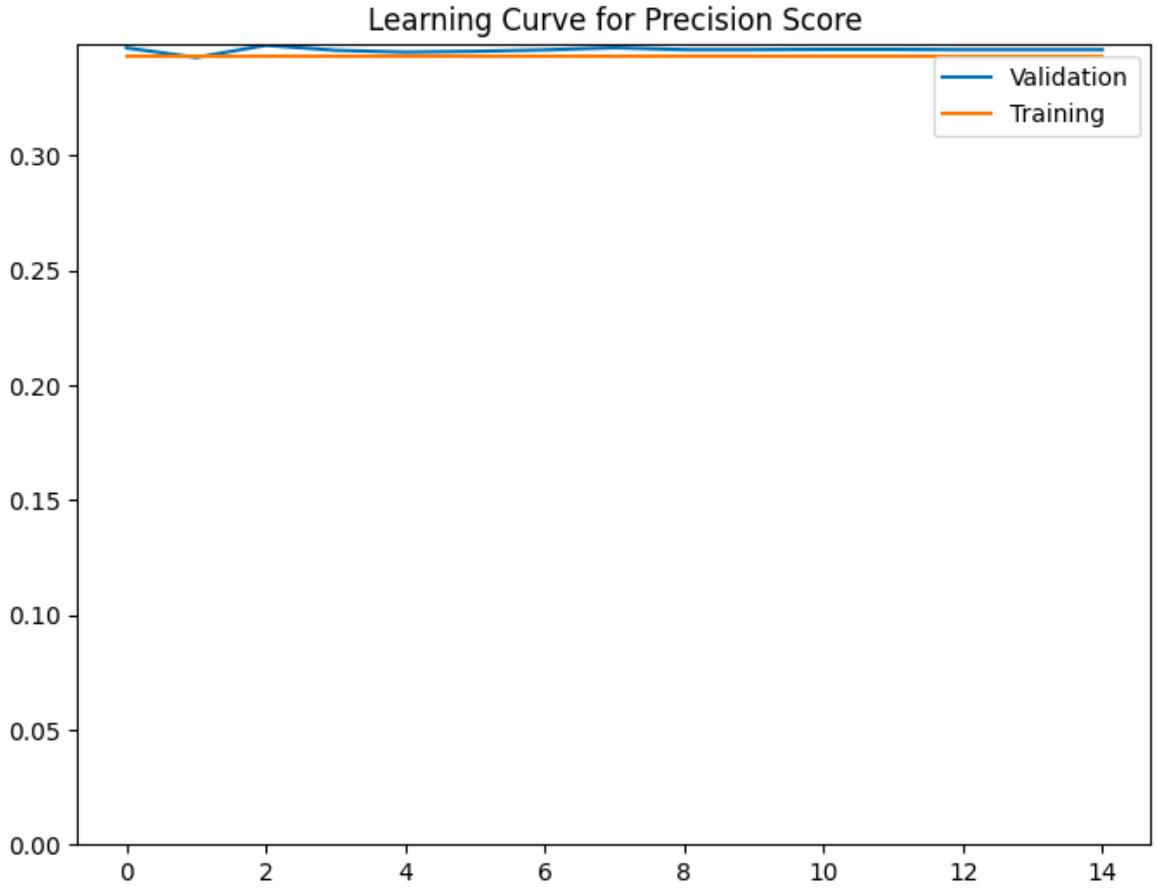
2. Learning Curve of F1 score



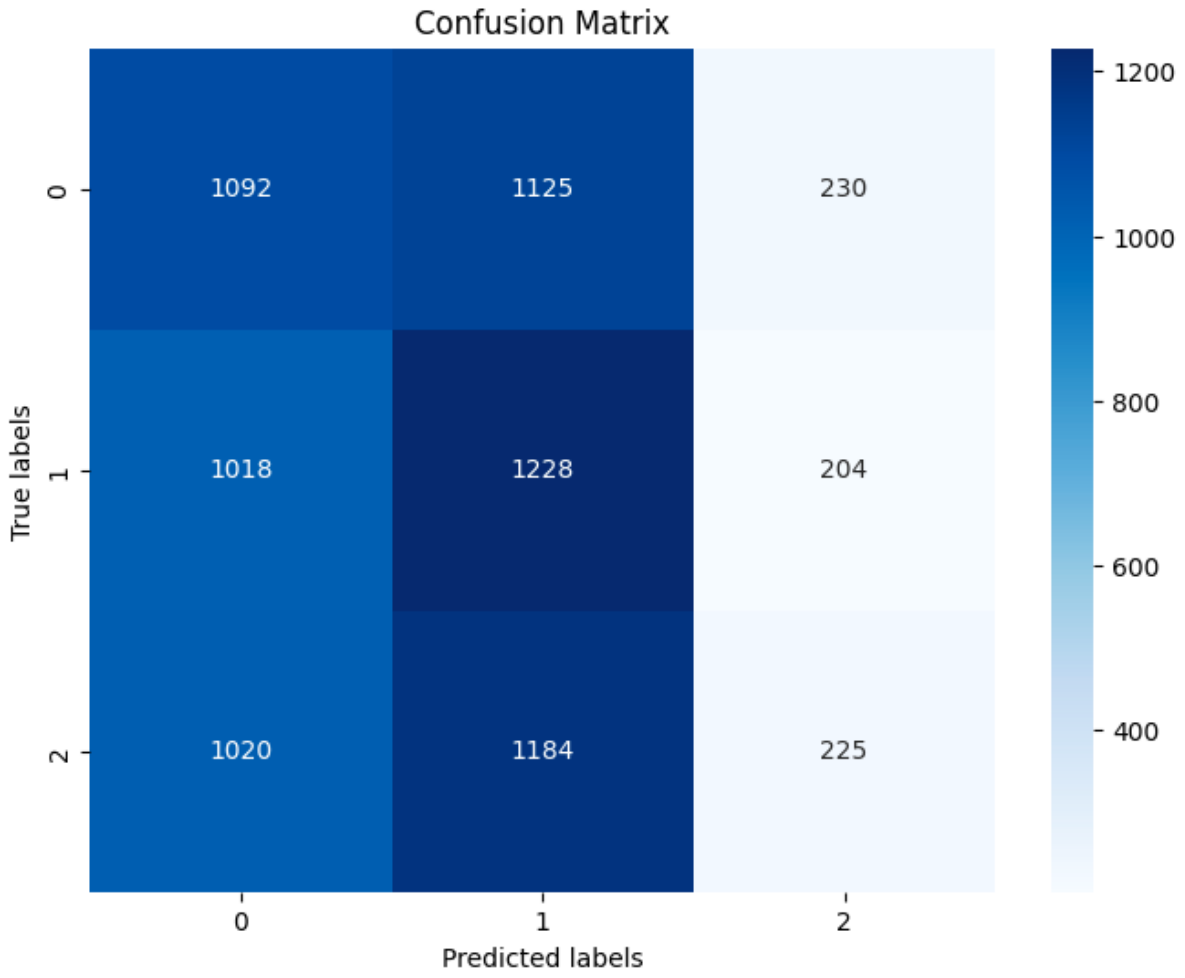
3. Learning Curve of Recall score



4. Learning Curve of Precision score



3.4.3. Confusion matrix.



4. Results and Overall Analysis

4.1. Results Analysis

About training and validation loss, I observe that the validation loss (1.0947) is slightly lower than the training loss (1.0986), suggesting acceptable generalization performance on unseen data. However, a significant disparity between the two could indicate potential overfitting or underfitting.
As mentioned above, the F1 score is a single metric that combines both precision and recall into a single value.
The result I get: Precision (0.3615 / 0.3516), Recall (0.3583 / 0.3482), F1-score (0.3452 / 0.3364), are not very good.
The precision of around 0.36 on validation and 0.35 on the test data suggests that when the model predicts a positive sentiment, it is correct around 35-36% of the time. The recall of approximately 0.36 on validation and 0.35 on the test data implies that the model can identify around 35-36% of the actual positive sentiments in the dataset. The F1-score, which combines precision and recall, sits around 0.35 on validation and 0.34 on the test data, indicating a balanced performance between precision and recall
So, according to the results of scores the conclusion is that our model doesn't work very well. The scores we get are 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. Comparison with the first project

Comparing with the first problem, I observe that for this problem Logistic Regression outperforms a deep neural network (DNN) Sentiment classifier. The difference in their performance it's close to null, only 2% better.
One reason may this happened it's that the features in the dataset don't contain complex relationships that require a deep hierarchy to be learned, simpler models like logistic regression might perform better.

5. Bibliography

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
https://medium.com/artificial-corner/going-beyond-accuracy-a-deep-dive-into-classification-me
https://medium.com/nerd-for-tech/fine-tuning-pretrained-bert-for-sentiment-classification-using
https://www.baeldung.com/cs/training-validation-loss-deep-learning
https://medium.com/analytics-vidhya/sentiment-analysis-for-text-with-deep-learning-2f0a0c