UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Michail Chatzispyrou*
sdi: 1115202000212

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

Contents

1. Abstract

This work studied the development of a sentiment classifier, using Logistic Regression, for a twitter dataset regarding the Greek general elections. Each tweet consists of an ID, its actual text and lastly the class it belongs to: POSITIVE, NEUTRAL, NEGATIVE. The classifier handles all the classes. The experiments performed for this development utilized a variety of methods, including data processing, vectorizers and hyper-parameter tuning.

2. Data processing and analysis

2.1. Pre-processing

Pre-processing is an essential component of Artificial Intelligence (AI) and Machine Learning (ML), since it has the ability to convert raw input data into a more refined and useful form. Also, it can improve the quality and validity of the input data, which enhances both the efficiency and precision of the algorithms that come after.

An extensive data cleaning procedure was used to improve the dataset's quality and analytical appropriateness in order to create the current model. The act of removing links was crucial in getting rid of unnecessary site addresses and maintaining the dataset's primary emphasis. Similarly, by removing potential distractions from metadata, the systematic elimination of mentions and hashtags improved the dataset's readability. Text capitalization was standardized to lowercase in order to preserve uniformity throughout the dataset. Moreover, a further refinement phase required keeping only Latin or Greek-language words in order to guarantee linguistic consistency and remove non-alphabetic characters. Lemmatization was included as an additional option to the data cleaning process to further simplify the dataset in order to enable a more accurate and comprehensive analysis in the present study. Additionally, the removal of stopwords improved dataset conciseness by excluding common words with limited analytical value. Finally, accents were removed to ensure uniformity in text representation.

2.2. Analysis

We use a variety of tools to navigate the data's complexities as we begin our initial exploration phase and reveal its mysteries. Printing the dataframe (1) itself is the first step that is both straightforward and essential. Having a basic knowledge of the dataset's structure from this first look lets us examine its elements, column names, and the type of data it contains.

	New_ID	Text	Sentiment	Party
0	35027	#απολυμανση_κοριοι #απεντομωση_κοριος #απολυμα...	NEUTRAL	SYRIZA
1	9531	Έξι νέες επιστολές για τη Μακεδονία «καίνε» τη...	NEGATIVE	ND
2	14146	Ισχυρό ΚΚΕ, δύναμη του λαού στη Βουλή και στου...	POSITIVE	KKE
3	28716	@five2nds @anthi7vas Μνημονιακότατο το #ΜεΡΑ25...	NEUTRAL	KKE
4	32886	@ai_katerina Αυτό που είναι συγκλονιστικό είνα...	NEUTRAL	SYRIZA
...	...	...	...	...
36625	35374	@KourtakisJohn @kmitsotakis Ο Κούλης ο Μητσοτά...	NEUTRAL	ND
36626	7744	@enikos_gr @NChatzinikolaou @AdonisGeorgiadi Π...	NEGATIVE	ND
36627	35216	Η θέση του ΚΚΕ για την ασφάλεια των πολιτών κα...	NEUTRAL	KKE
36628	2855	@thanosplevris Μαρη κακομοίρα θυγατέρα του ναζ...	NEGATIVE	ND
36629	25500	@gijjstalking @SpirosR76 Εντάξει με έπεισες! Κ...	NEUTRAL	KKE

36630 rows × 4 columns

Figure 1: Train DataFrame

A snapshot of our dataset, consisting of a significant 36630 rows and carefully arranged into 4 columns, appears as we print the dataframe (1). Every tweet is given a unique identity by the first column. The tweets themselves are visible in the second column, which provides an insight into the textual world. Each tweet is a story contained inside the boundaries of the dataset. Furthermore, in the third column, we find the sentiment classes of neutral, positive, and negative. The last touch is found in the fourth column, where each tweet is matched with a particular political party based on its political resonance.

Next, we turn our attention to visualization, creating a bar plot (2) that provides a broad overview of the political fields included in the dataset. This visual aid offers a brief overview of how tweets are distributed among different political parties.

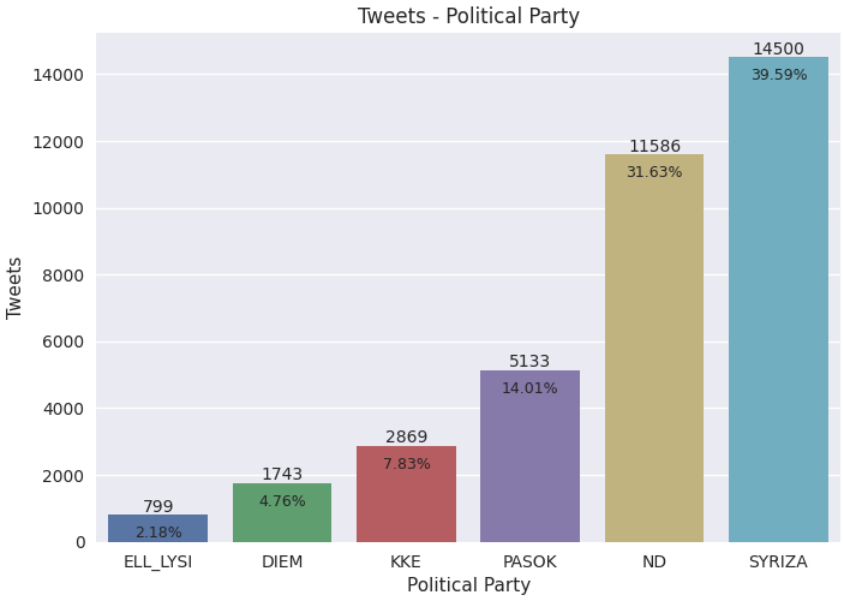

Figure 2: Tweets-Political Party Bar Plot

It can be observed that about 40% of the tweets examined deal with topics related to the political party called "SYRIZA". The New Democracy (ND) party comes in second, taking about 32% of the conversation on Twitter. The collection also contains references to "ELL_LYSI", which accounts for a little over 2% of the total and "PASOK," which accounts for a noteworthy 14%. Remarkably, the collection also includes references to "DIEM," but with a lesser frequency—less than 5%. At roughly 8%, the Communist Party of Greece (KKE) holds a significant portion.

To further our investigation, we create two separate bar plots, each of which highlights a different aspect of our dataset. The first bar plot (3) provides an in-depth comprehension of the distribution of neutral, positive, and negative attitudes inside each individual political party. At the same time, the second bar plot (4) turns our attention to a wider view, showing how tweets are distributed throughout the dataset in various sentiment categories.

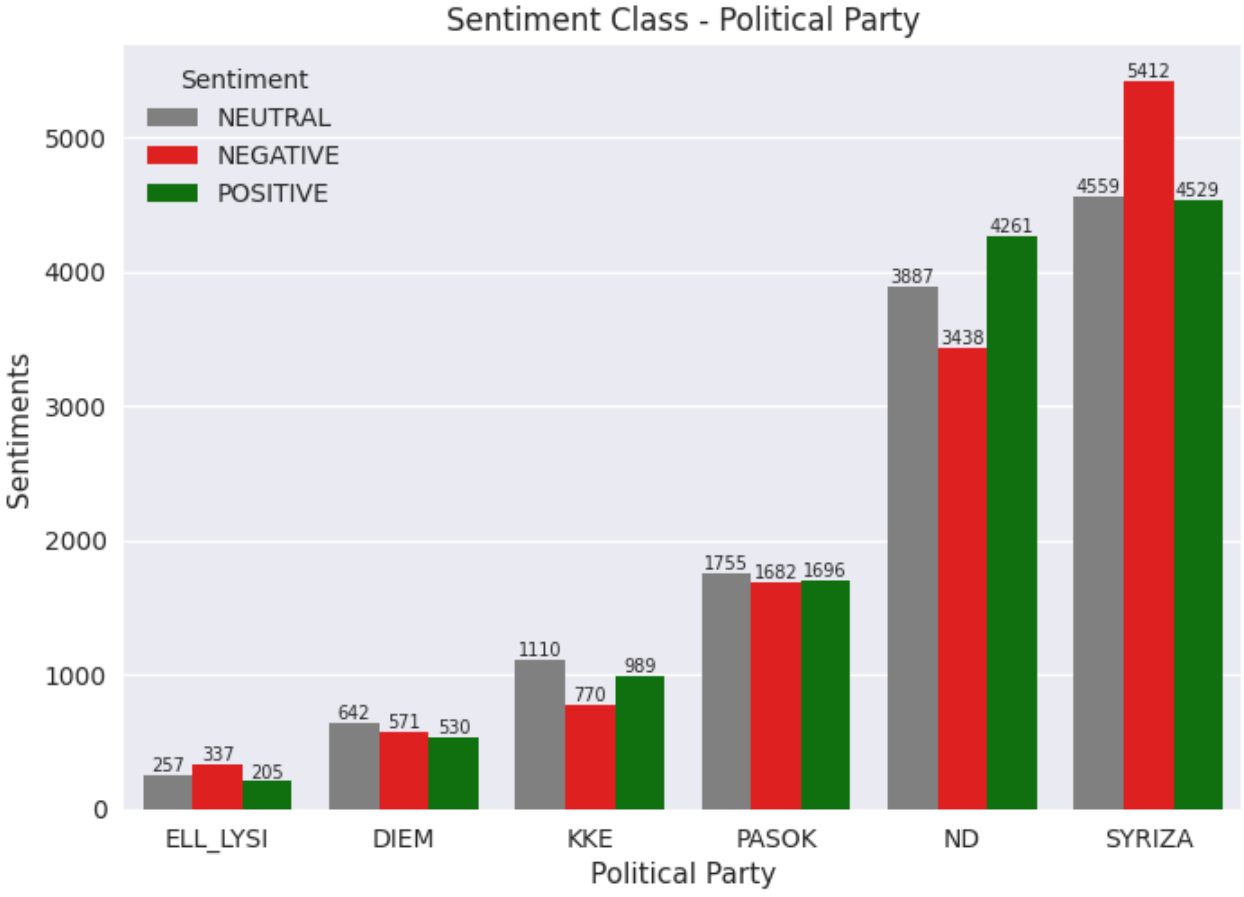

Figure 3: Sentiments-Political Party Bar Plot

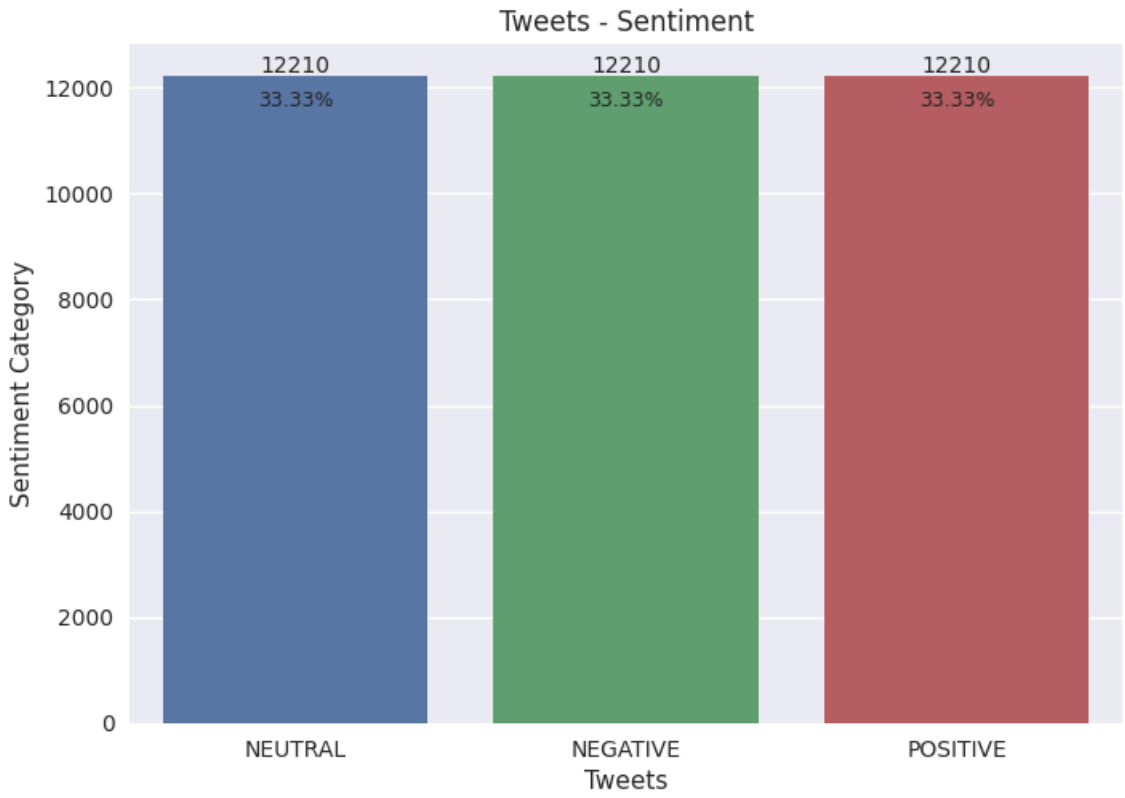

Figure 4: Tweets-Sentiment Bar Plot

Analysis reveals that while sentiment categories within each political party are not exactly balanced, they are admirably close to being balanced. This adds to an overall balanced dataset along with a fairly distributed distribution of tweets within every sentiment category. Fairness and ease of model training are two benefits of a balanced dataset. Unbiased exposure to every class ensures that no class has an undue influence on algorithms like Logistic Regression during training. Finally, evaluation criteria that facilitate easy interpretation, such as accuracy, become more trustworthy measures of model performance.

Finally, we will examine the impact of all three distinct datasets:

- No Pre-Processing (NP)
- Basic Pre-Processing (BP)
- Basic Pre-Processing & Lemmatization (PL)

We will use two different approaches to evaluate the frequency of common words in these datasets. In the first, words are arranged alongside their corresponding count distributions for every dataset using a dataframe (5). The second technique involves the use of word clouds (6,7,8), which is a type of graphic that visually summarizes the frequency of words in a particular dataset. A word cloud is a graphical representation of words that are sized according to their frequency in a dataset. It provides an easy-to-understand description of the keywords that are most frequently used.

	Common-Words-NP	Count-NP	Common-Words-BP	Count-BP	Common-Words-PL	Count-PL
0	:	36212	νδ	5620	τσιπρας	8662
1	#	34383	συριζα	4932	νδ	5619
2	https	25791	τσιπρα	4522	συριζα	4610
3	.	22585	μητσοτακης	4232	μητσοτακης	4222
4	,	21957	τσιπρας	4181	μητσοτακη	3796
5	@	17872	μητσοτακη	3786	κκε	2716
6	και	17413	κκε	2716	πασοκ	2055
7	το	15547	πασοκ	2058	κανω	1809
8	να	14973	κιναλ	1781	κιναλ	1781
9	του	14321	νεα	1334	αλεξη	1454
10	για	11080	συνεντευξη	1319	συνεντευξη	1388
11	ο	10328	εκλογες	1259	θελω	1266
12	την	9901	σκαι	1224	εκλογες	1258
13	!	9435	κανει	1127	χρονος	1220
14	με	8811	χρονια	1076	σκαι	1218
15	που	8523	γεννηματα	1066	κυβερνηση	1162
16	θα	8326	ελλαδα	1057	χωρα	1131
17	τον	8134	λεει	1025	γεννηματα	1066
18	η	7854	amp	965	ελλαδα	1053
19	της	7724	δημοκρατια	963	λεει	995

Figure 5: Common Words DataFrame

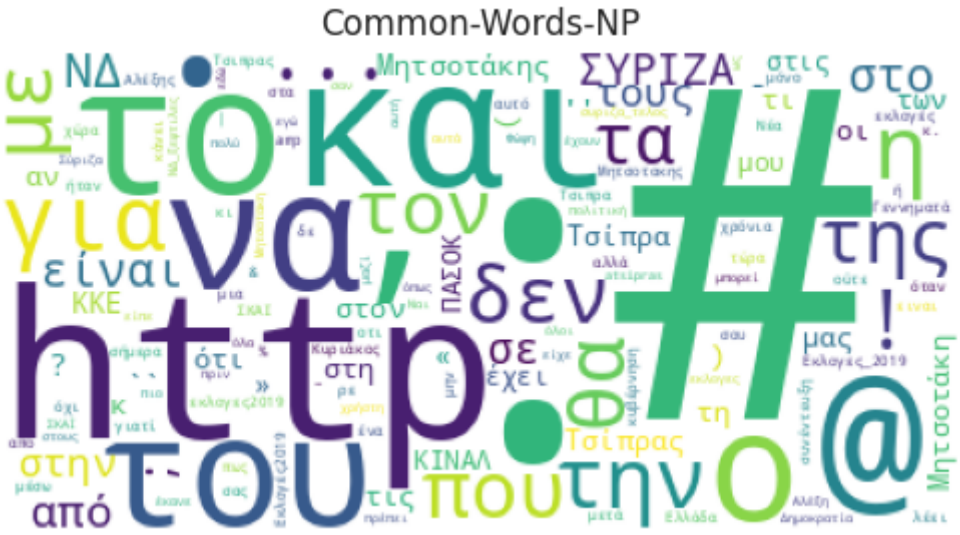

Figure 6: Common Words World Cloud NP

Figure 7: Common Words World Cloud BP

Figure 8: Common Words World Cloud PL

When looking at the raw dataset without any pre-processing, it is clear that a large percentage of its common terms are links, hashtags, mentions, and punctuation marks (commas, periods, etc.). Pronouns and articles also make up a considerable amount of the dataset. This raw composition adds noise and could render it more difficult for the model to identify significant trends. On the other hand, after undergoing basic pre-processing, the quality and significance of the words were substantially enhanced in the second dataset. Eliminating unnecessary components makes the text's main ideas more visible and improves the model's capacity to extract pertinent data. A deeper comprehension of language is visible in the third dataset, where lemmatization is combined with basic pre-processing. It is able to identify the similarities between words like "τσιπρα" and "τσιπρας," merging them in a suitable way. Nonetheless, there are also cases where the lemmatization process seems inconsistent, like "μητσοταχης"

and "μητσοταχη". That being said, a notable finding in the final two datasets is the lack of common terms that are similar to stop words or noise, as shown in the first dataset. As a result, the dataset's composition has improved qualitatively, highlighting the beneficial effects of pre-processing.

2.3. Data partitioning for train, test and validation

The data sets chosen were the ones that the instructors provided. This decision was made in order to prevent any further complications regarding the development of the code. A ratio of 0.156 is attained by utilizing data from the validation set (5232) and training set (33630).

2.4. Vectorization

In Machine Learning (ML) and Natural Language Processing (NLP), scikit-learn (sklearn) provides several versatile tools for text data processing, including CountVectorizer, TF-IDF Vectorizer, and HashingVectorizer. CountVectorizer is a straightforward method that converts a collection of text documents to a matrix of token counts, where each row represents a document, and each column represents a unique word in the entire corpus. TF-IDF Vectorizer, on the other hand, goes a step further by considering not only the frequency of a term in a document but also its importance in the entire corpus. It assigns a weight to each term based on its frequency-inverse document frequency (TF-IDF) to capture the significance of terms in distinguishing documents. HashingVectorizer is a memory-efficient alternative that hashes words into fixed-size integer indices, thereby reducing the need to store an explicit vocabulary and saving memory space. To test the impact of various pre-processing methods, in combination with the previously mentioned vectorizer methods, we ran experiments with the Logistic Regression model using three distinct datasets:

- No Pre-Processing (NP), the dataset was intact
- Basic Pre-Processing (BP), the dataset underwent the following modifications:
  - Removing links
  - Removing hashtags
  - Removing mentions
  - Removing stopwords
  - Removing accents
  - Lowercase
  - Removing non Latin or Greek-language words
- Basic Pre-Processing & Lemmatization (PL), all the Basic Pre-Processing with the addition Lemmatization

It's important to note that these experiments were conducted using cross-validation with 5 folds. Below are presented the results in table and bar plot format.

Vectorizers	Time Taken	Recall	F1-Score	Accuracy	Precision
Count-NP	138.56	0.378259	0.377749	0.378252	0.378164
TF-IDF-NP	48.67	0.391434	0.391172	0.391438	0.391640
Hashing-NP	387.79	0.387612	0.387042	0.387617	0.387832
Count-BP	37.28	0.378058	0.377312	0.378057	0.378633
TF-IDF-BP	26.87	0.388188	0.387867	0.388190	0.388275
Hashing-BP	323.95	0.384749	0.384264	0.384750	0.384691
Count-PL	44.52	0.378245	0.377555	0.378251	0.379079
TF-IDF-PL	21.18	0.378434	0.378263	0.378440	0.378547
Hashing-PL	342.01	0.375189	0.374858	0.375191	0.375124

Table 1: Vectorization Results

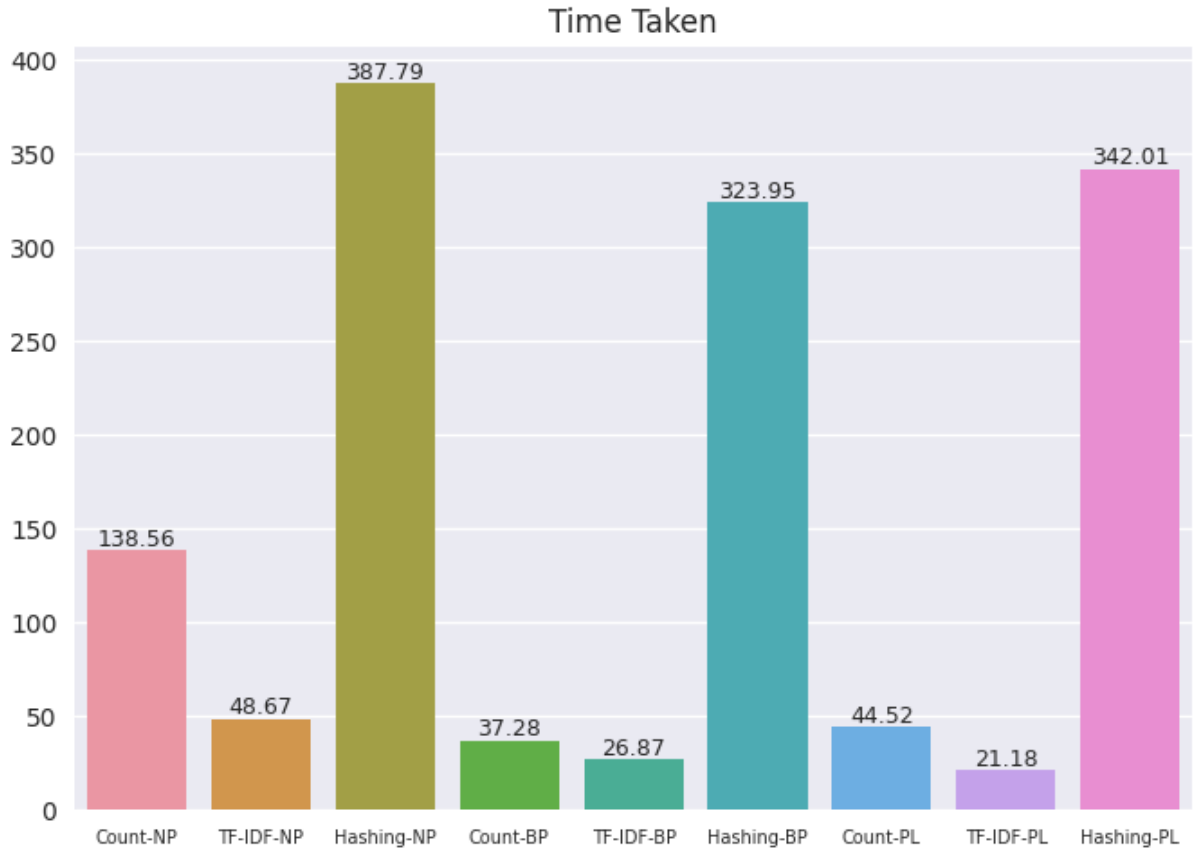

Figure 9: Time Taken Bar Plot

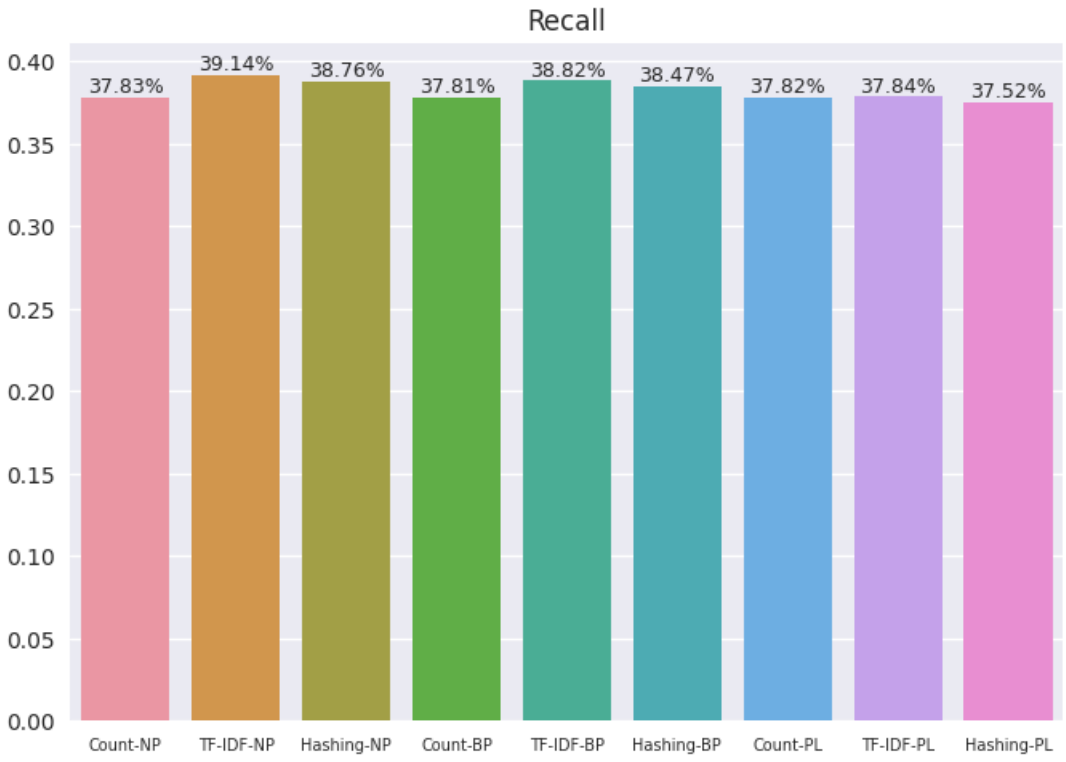

Figure 10: Recall Bar Plot

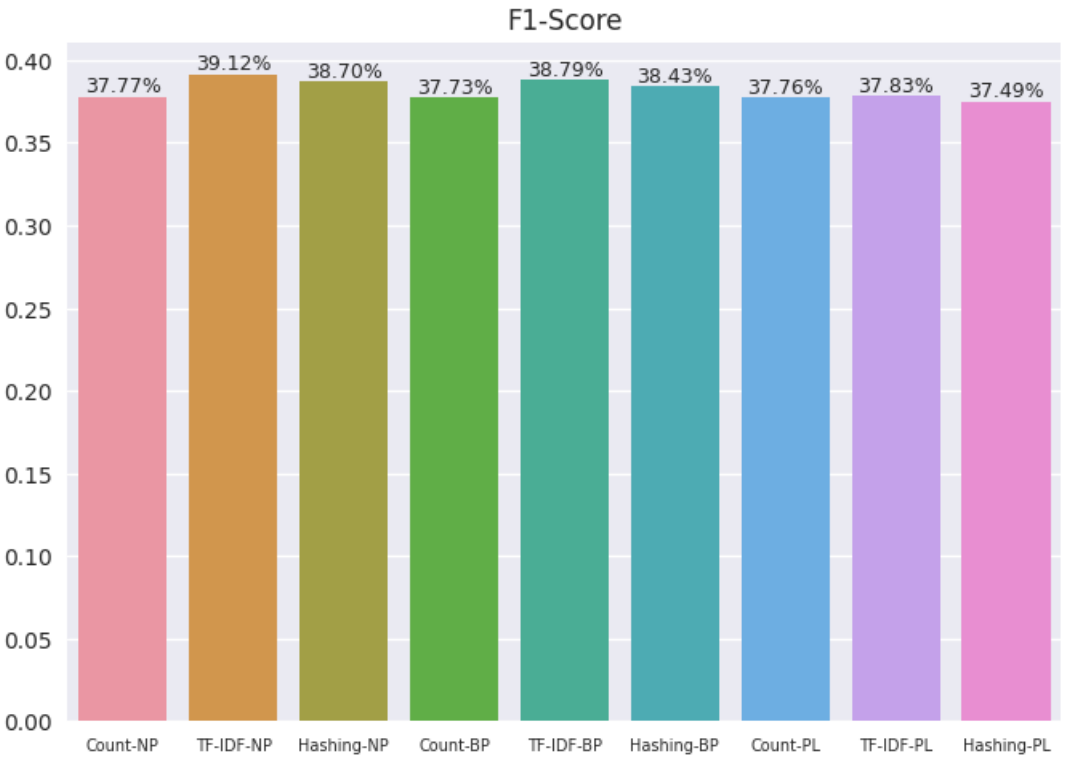

Figure 11: F1-Score Bar Plot

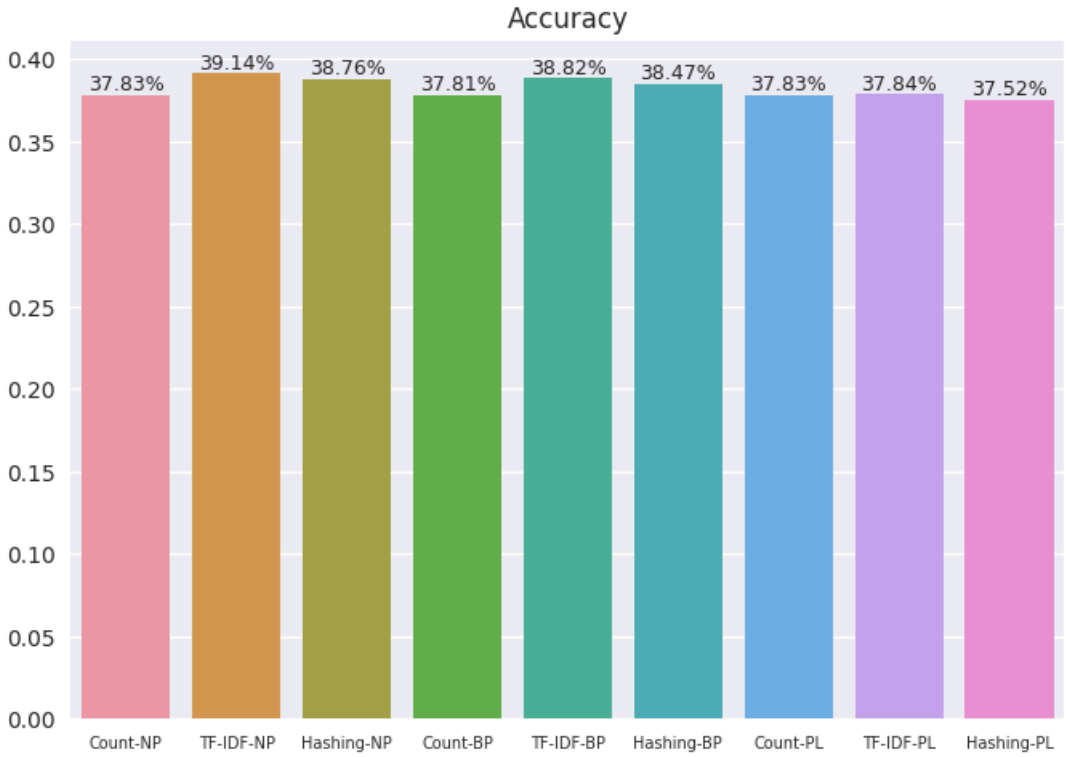

Figure 12: Accuracy Bar Plot

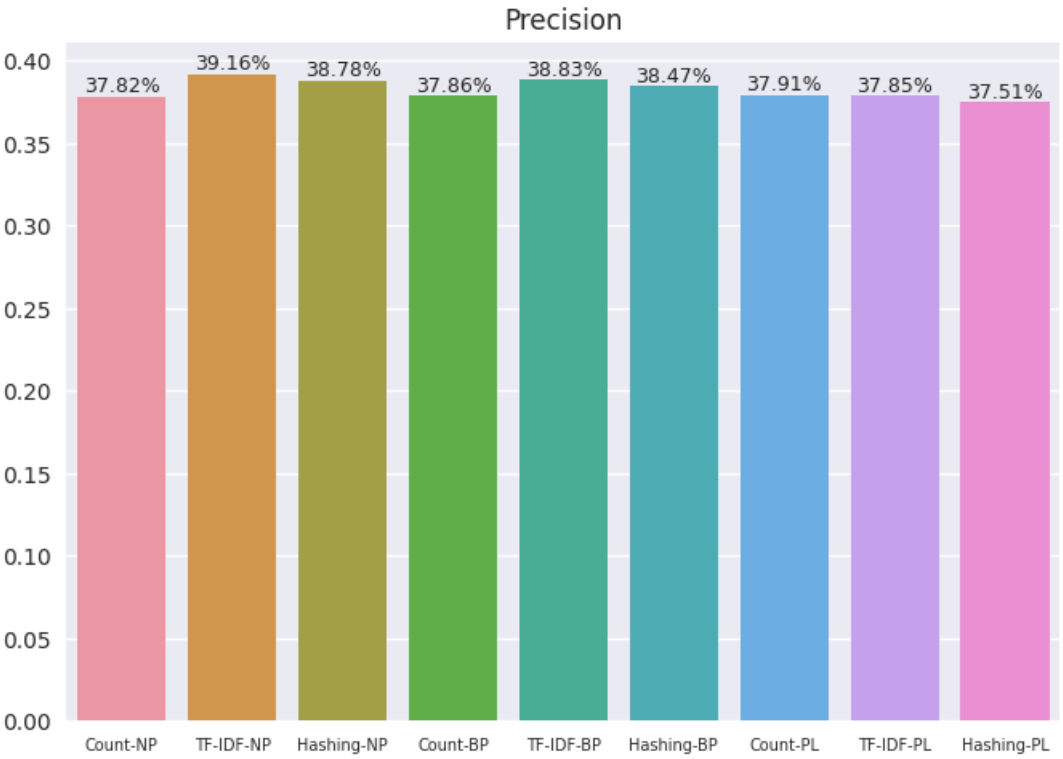

Figure 13: Precision Bar Plot

As evident from both the table (1) and the diagrams (9, 10, 11, 12, 13), the HashingVectorizer demonstrates excessive slowness across all three datasets, rendering it

inefficient for large sets, and therefore, it is discarded. Turning to the TF-IDF Vectorizer, it emerges as the fastest option across all three datasets. Notably, in the dataset following basic preprocessing, the CountVectorizer closely trailed the TF-IDF in terms of time. However, while the CountVectorizer is not as time-consuming as HashingVectorizer there is a vectorizer that achives shorter times, so it inevitable excluded. Thus, the choice of a vectorizer becomes clear. Paradoxically, the dataset without any processing yields the best accuracy across all metrics. However, it must be rejected due to its sensitivity to overfitting. When comparing the datasets with Basic Pre-Processing (without Lemmatization) and the one with Lemmatization, a slight drop of approximately 1% is observed. Considering all these factors, the optimal combination is determined to be the TF-IDF Vectorizer with Basic Pre-Processing.

There is one essential step to be completed before proceeding to the section of algorithms and experiments. We are ready to present the key terms that are necessary to classify tweets into the classes of neutral, negative, or positive sentiment. In order to do this, we will use the TF-IDF vectorizer on two datasets: one that has undergone No Pre-Processing (NP) and the other that has Basic Pre-Processing (BP). The final word clouds will function as visual representations, providing information on the most significant words that affect the classification procedure.

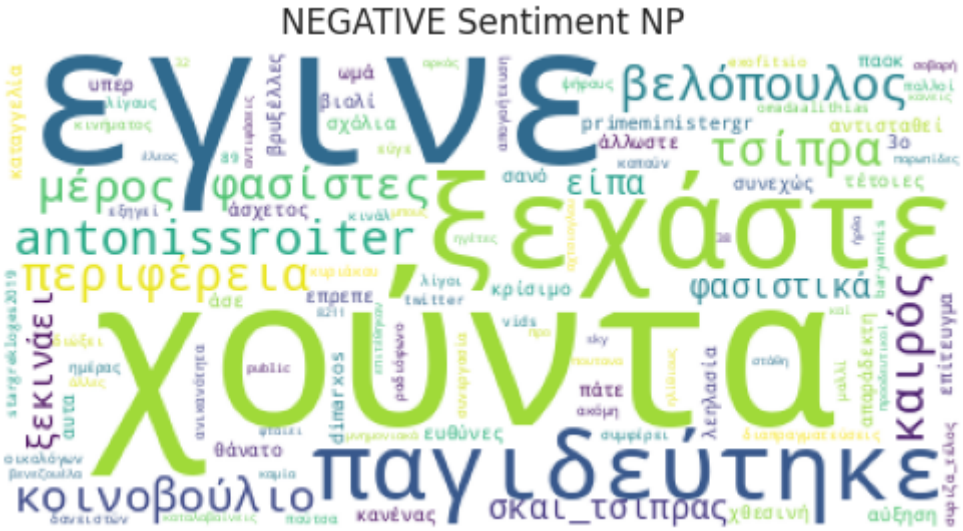

Figure 14: NEGATIVE World Cloud NP

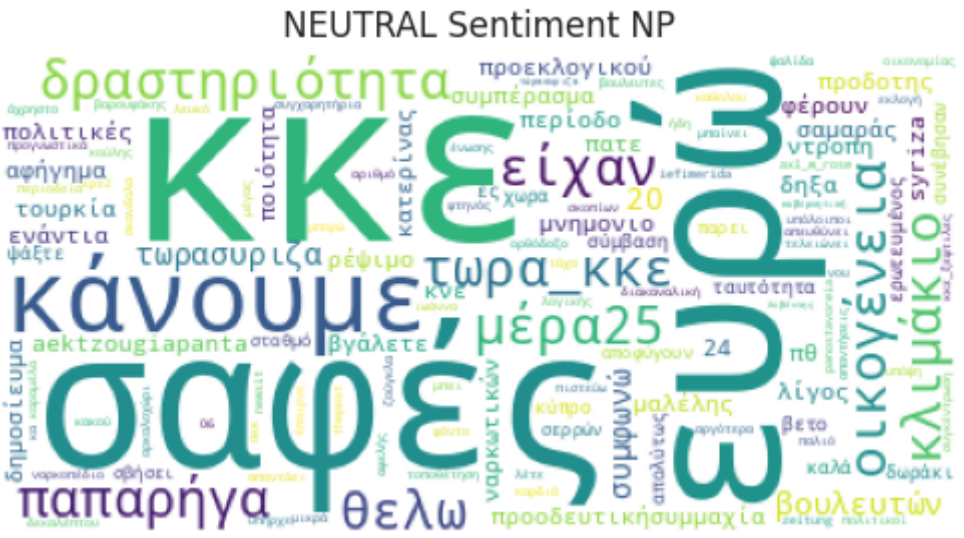

Figure 15: NEUTRAL World Cloud NP

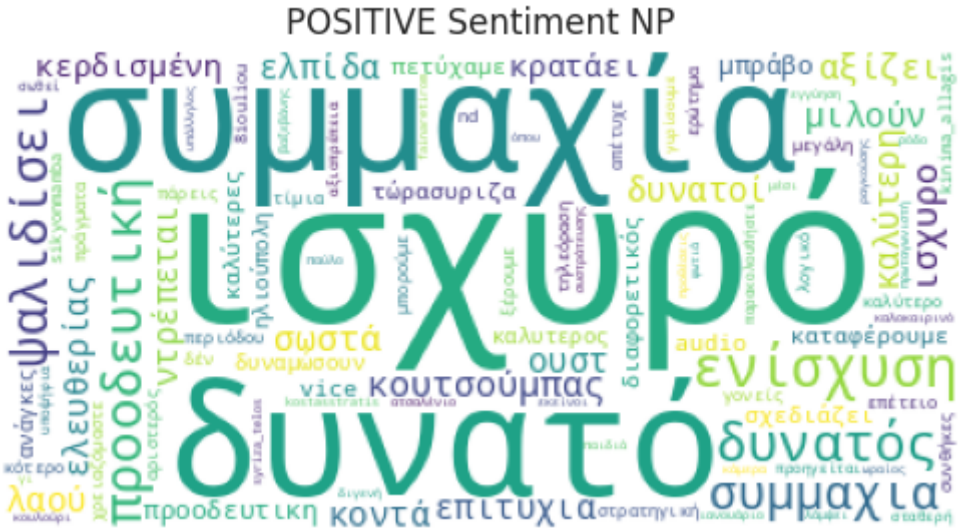

Figure 16: POSITIVE World Cloud NP

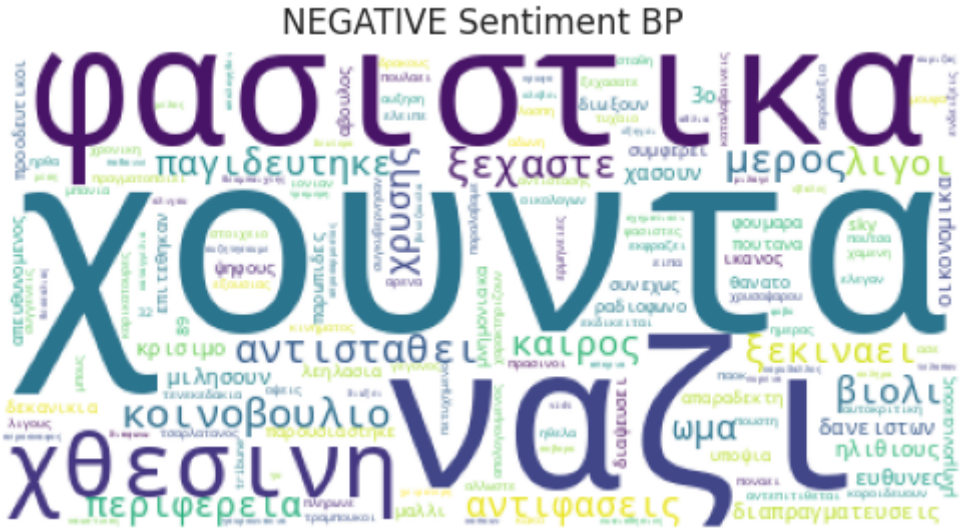

Figure 17: NEGATIVE World Cloud BP

Figure 18: NEUTRAL World Cloud BP

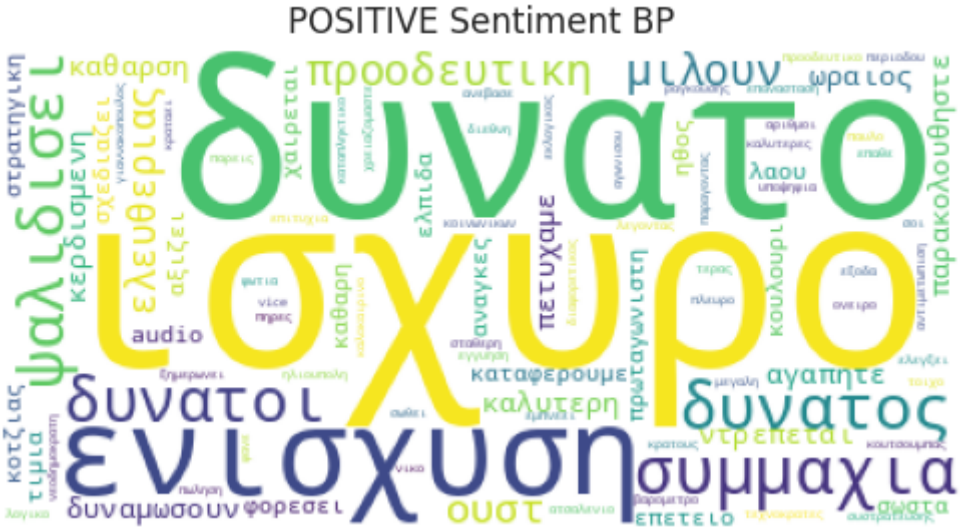

Figure 19: POSITIVE World Cloud BP

In reference to the raw dataset, examination of the negative class yields terms like "εγινε", "τσιπρα" and "βελοπουλος" with significant weights but no essential association with negativity. Terms like "χούντα" which refer to one of the worst eras in Greek history, naturally connote a great deal of negativity. On the other hand, given their inherent negative meanings, terms like "φασιστικά" "χουντα" and "ναζι" correctly carry a significant amount of weight in the Basic Pre-Processed (BP) dataset. However, inappropriate negative terms like "χθεσινη" and "κοινοβουλιο" throw off the precision of the model. Heavy terms like "KKE," a political party, "ευρώ" and "σαφές" are noted when we go to the neutral class of the No Pre-Processing (NP) dataset. On the other hand, inaccurate terms like "μνημονιο" and "προδοτης" introduce errors. The dataset in the neutral class with Basic Pre-Processing (BP) shows similarities in the most heavily weighted terms, but it keeps inappropriate placements such as "προδοτης" and "ναρκοπεδιο". Positive adjectives like "δυνατό" dominant in the unprocessed set's positive class, but undesirable terms like "ουστ" and "ντρεπεται" corrupt the outcomes. In conclusion, preprocessing improves keyword distribution, which supports the choice of Basic Pre-Processing (BP). **However, the classification errors occurred during the tweet extraction process affect the accuracy of the results.**

3. Algorithms and Experiments

Disclaimer: Keep in mind that the below analysis was made in the last execution of the notebook. If you choose execute the notebook again the results may be different. Thanks for your understanding.

In the conducted experiments, the classification was carried out into three classes using Logistic Regression with cross-validation and data preprocessing techniques. Logistic regression, a statistical technique that is frequently used for binary and multi-class classification applications. It simulates the likelihood that an instance will fall into a particular class. An evaluation technique called cross-validation was used to estimate the model's effectiveness. Data preprocessing involved preparing and polishing the dataset to optimize the Logistic Regression model's effectiveness.

3.1. Hyper-parameter tuning

For classification problems, Scikit-learn's implementation of Logistic Regression offers an adjustable and popular tool. Key parameters for the scikit-learn Logistic Regression model are C, penalty, and solver. The trade-off between properly fitting the training data and avoiding overfitting is influenced by the parameter C, which is the inverse of the regularization strength. Stronger regularization is indicated by a smaller C value, which can aid in preventing the model from fitting noise in the training set. Using options like 'l1' for L1 regularization, 'l2' for L2 regularization and 'elasticnet' for Elasticnet regularization, the penalty parameter controls the kind of regularization that is used. Regularization promotes enhanced generalization to previously unknown data by keeping the model from growing overly complex. The algorithm utilized for optimization during model training is specified by the solver parameter. In order to achieve optimal performance in a variety of classification scenarios, it is necessary that these parameters be fine-tuned, as each one is critical in determining the behavior of the Logistic Regression model.

3.2. Evaluation

A thorough set of four measures, each providing a unique perspective on the model's performance, will be included in the evaluation process. These measures are f1-score, accuracy, recall, and precision. the F1-score is given particular consideration since it may effectively balance recall and precision. The F1-score is calculated as follows:

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{1}$$

In addition, a confusion matrix and a Receiver Operating Characteristic (ROC) curve were used for more in-depth study of the top model found using these measures.

3.3. Experiments

During the first stage, I prioritized the penalty and solver parameters while ignoring parameter C. Following the instructions in the sklearn documentation, I methodically investigated every possible combination for these two parameters:

- lbfgs - [l2, None]
- liblinear - [l1, l2]
- newton-cg - [l2, None]
- newton-cholesky - [l2, None]
- sag - [l2, None]
- saga - [elasticnet, l1, l2, None]

To prevent convergence problems with the models, 'None' was left out of the analysis. It's also important to note that there was a practical difficulty with the 'newton-cholesky' solver during testing. Its resource requirements were greater than the capacity that was available, going above Kaggle's 30GB limit.

3.3.1. Table of trials.

Trial	C	Solver	Penalty	Time Taken	F1-Score
1	1.0	lbfgs	l2	27.97	0.387867
2	1.0	liblinear	l1	2.39	0.371025
3	1.0	liblinear	l2	3.71	0.384570
4	1.0	newton-cg	l2	6.48	0.388044
5	1.0	sag	l2	2.94	0.388044
6	1.0	saga	l1	27.02	0.368281
7	1.0	saga	l2	3.14	0.387675
8	1.0	saga	elasticnet	82.14	0.377604

3.3.2. Plot of trials.

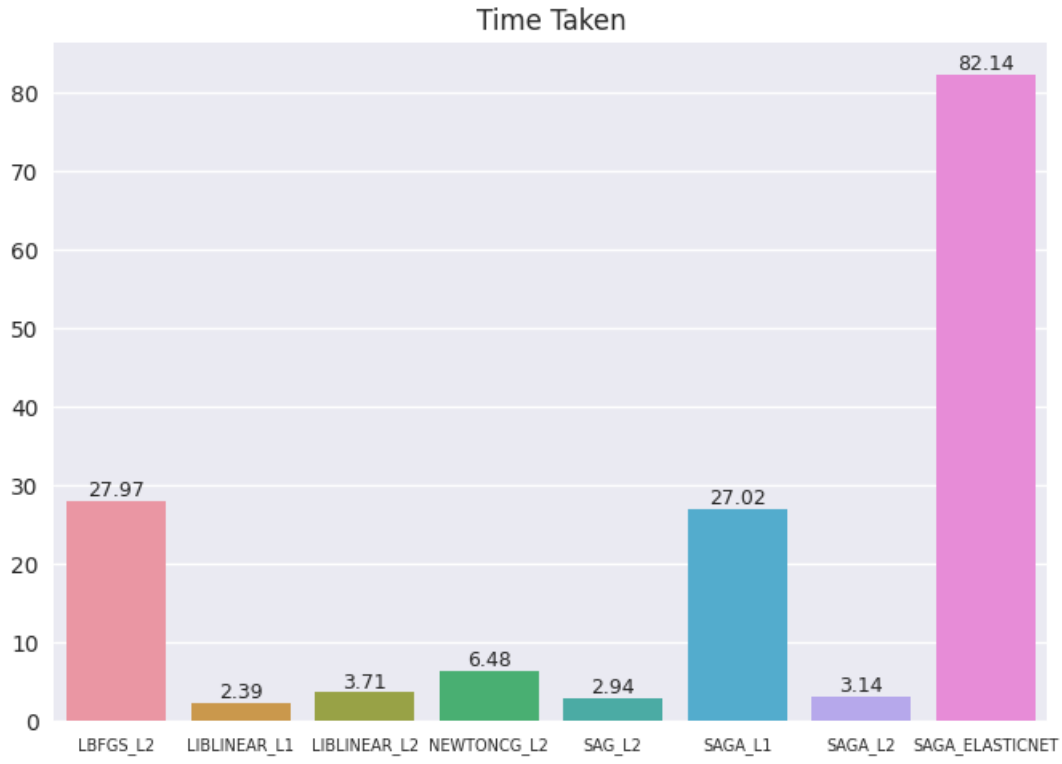

Figure 20: Time Taken Models

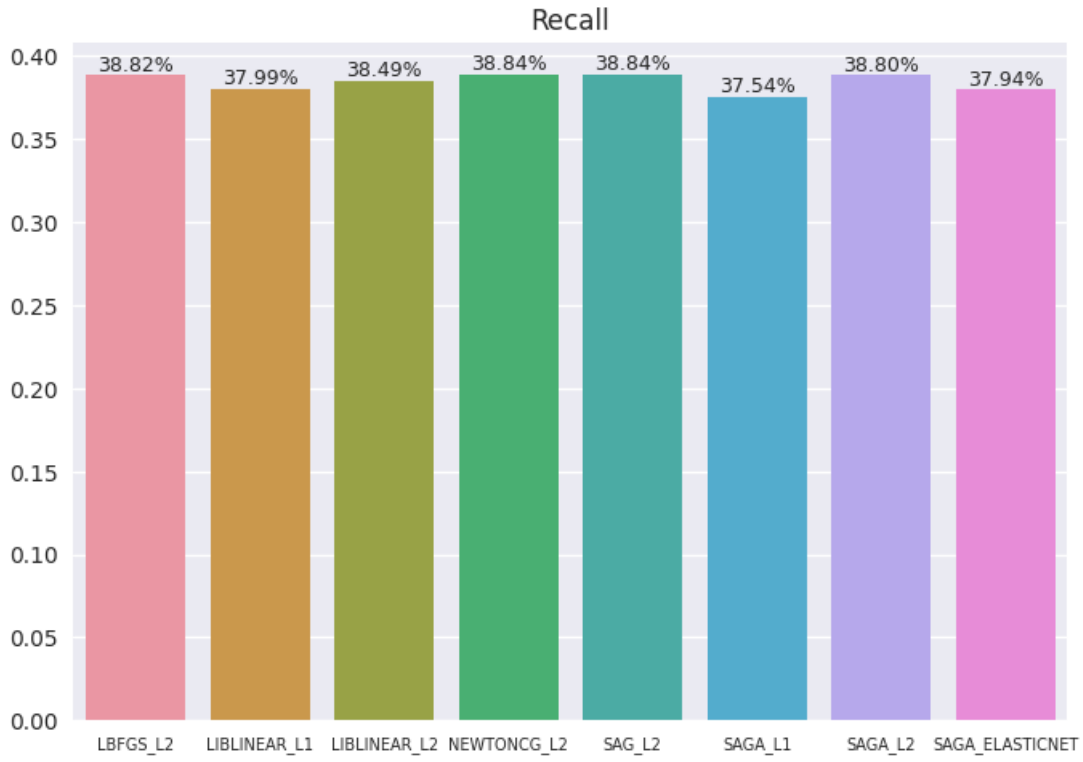

Figure 21: Recall Models

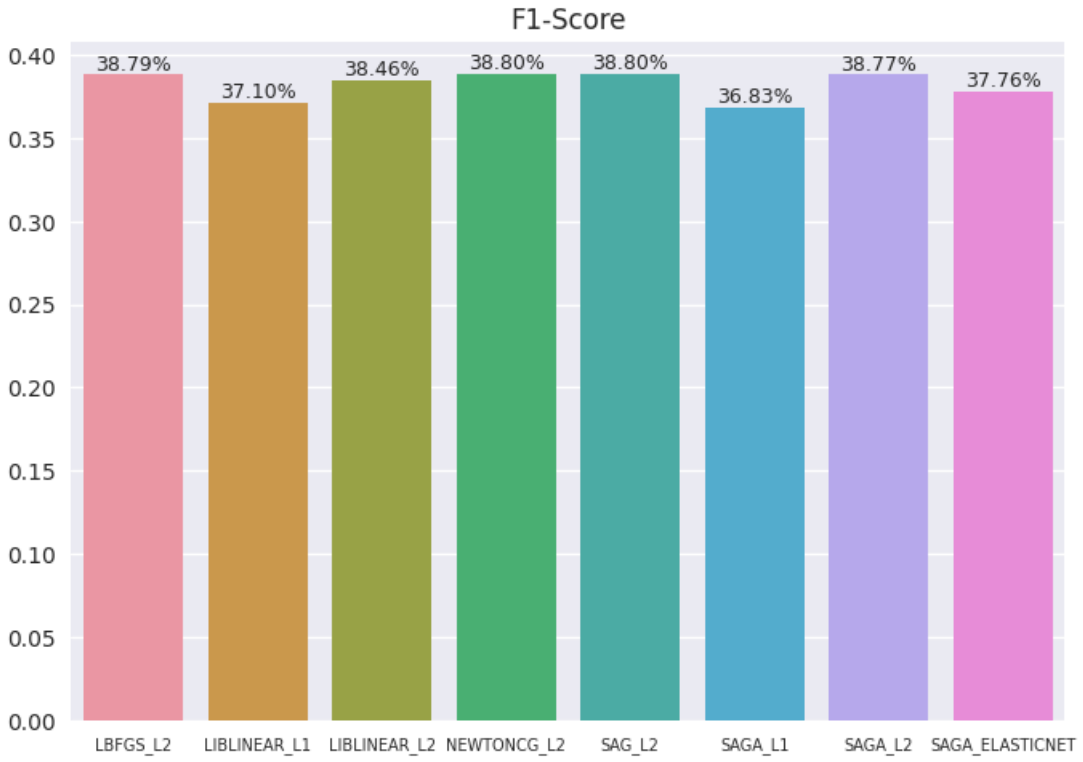

Figure 22: F1-Score Models

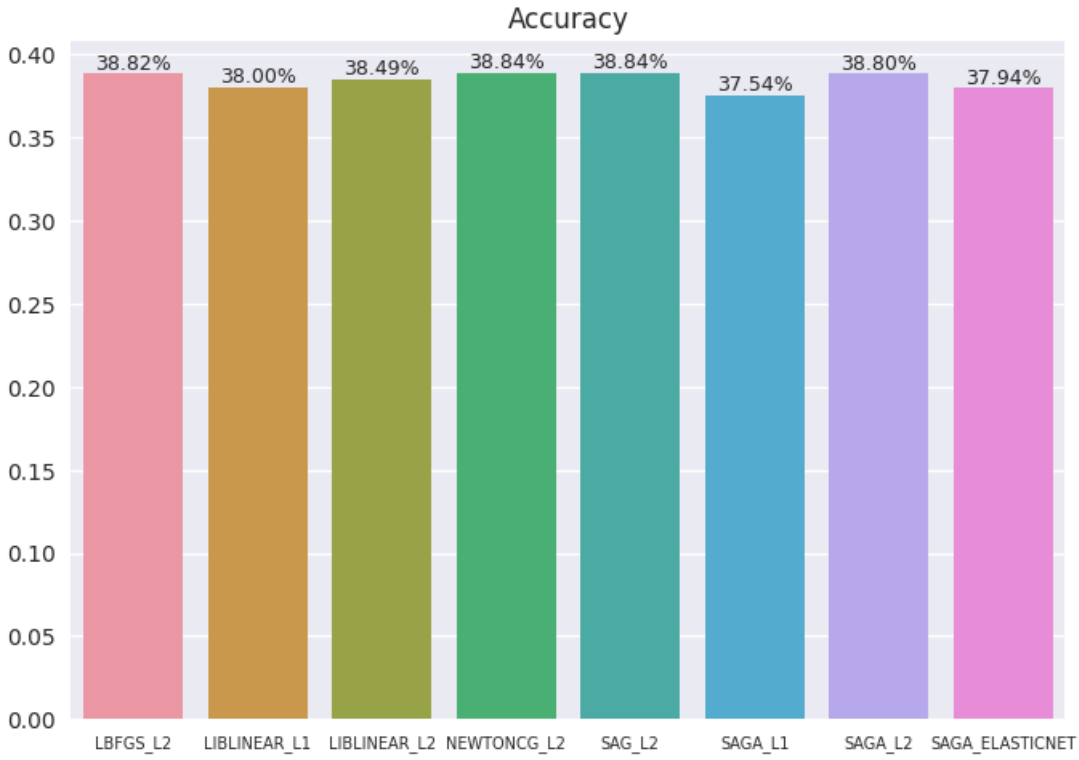

Figure 23: Accuracy Models

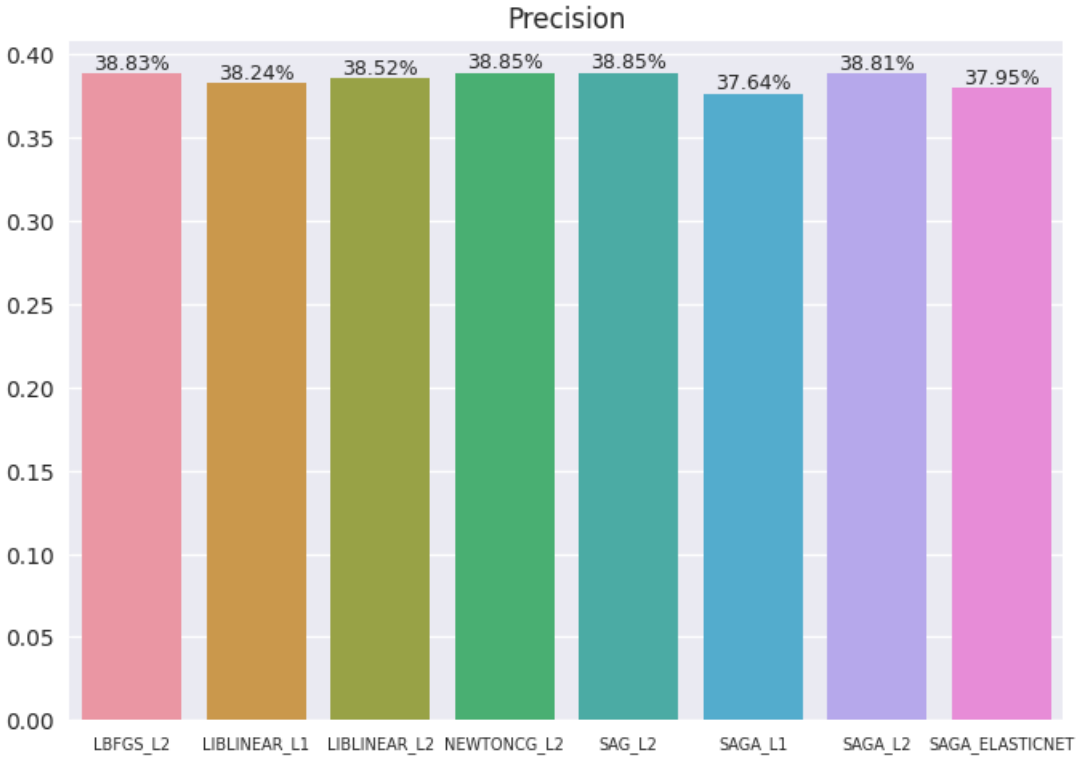

Figure 24: Precision Models

Given the variation in training times, which range from 2.39 seconds to 82.14 seconds, the dataset utilized for training and evaluation seems to be sizable. The newton-cg and sag solvers consistently produce the highest F1-scores of 0.3880 among the

evaluated solvers (lbfgs, liblinear, sag, and saga), demonstrating their efficacy in identifying the underlying patterns in the data. Remarkably, the liblinear solvers and lbfgs likewise perform rather well, but with marginally lower F1-scores. In terms of penalties—l1, l2, and elasticnet—all solvers consistently yield better results with the l2 penalty than the others. This implies that the regularization method known as l2, which penalizes the sum of squared coefficients, would be more appropriate for this specific dataset. It seems that the elasticnet penalty, which includes l1 and l2 penalties, produces a little lower F1-score, suggesting that a simpler regularization approach would work better in this situation. Additionally, the saga solver requires more training time but consistently shows competitive F1-scores, supporting all penalty kinds. This implies that there is a trade-off between model performance and computing efficiency.

Overall, the models that are shown have positive qualities, but one problem is consistent with all of the configurations: overfitting. The learning curves demonstrate this, as the trials can be widely divided into two categories. First, there are models whose training learning curves start high and then trend down, but they never go below the associated validation curves by a significant amount. Models in the second category have training curves that start off lower and then significantly increase until they significantly outpace the validation curves. Overfitting is indicated by both of these patterns.

3.3.3. Learning Curve of trials.

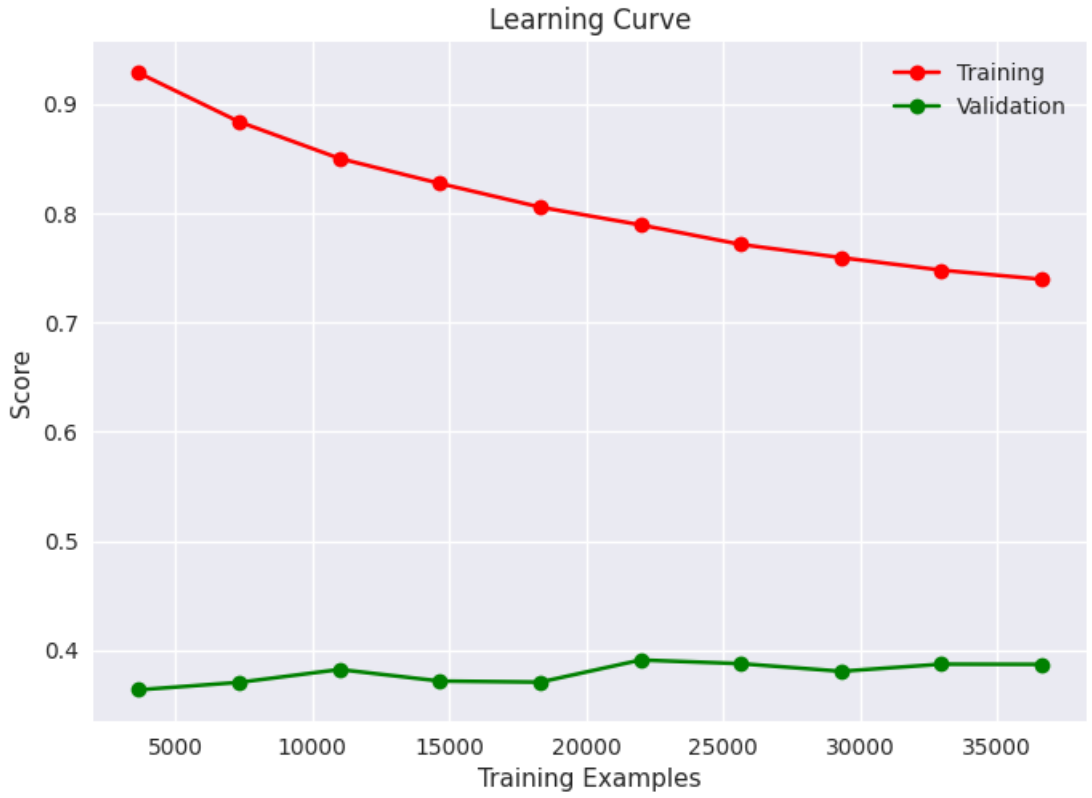

Figure 25: LBFGS & L2 Learning Curve

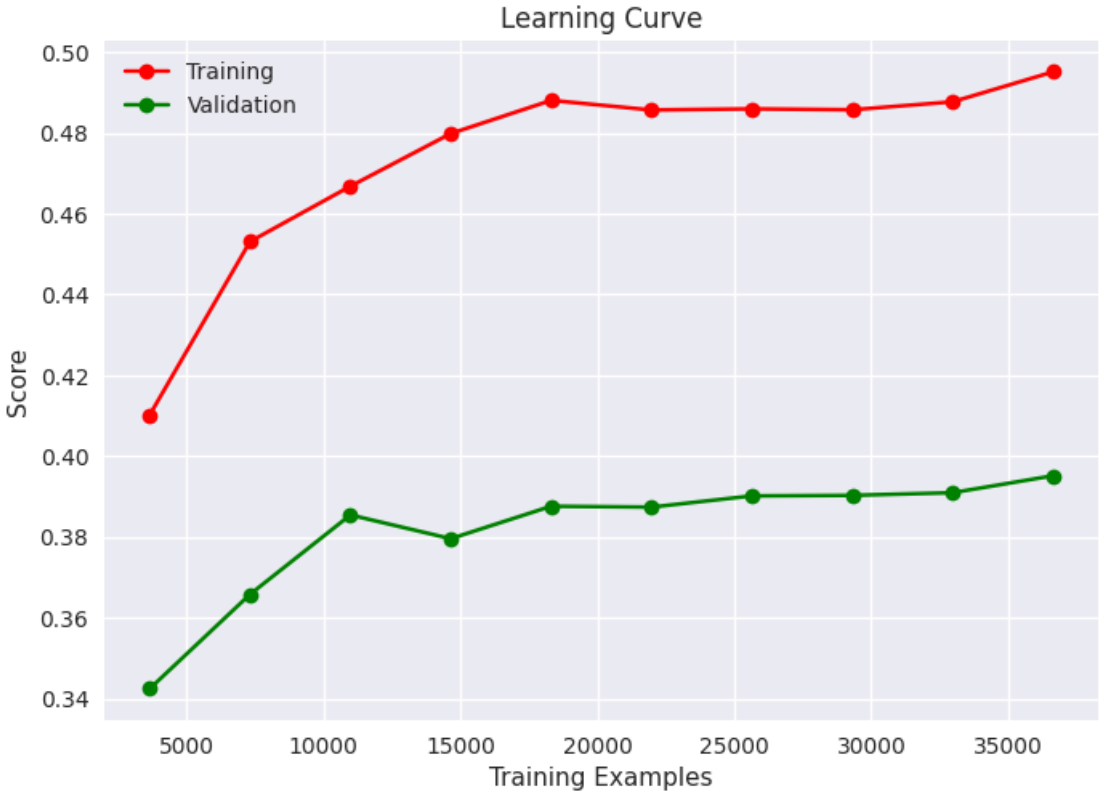

Figure 26: LIBLINEAR & L1 Learning Curve

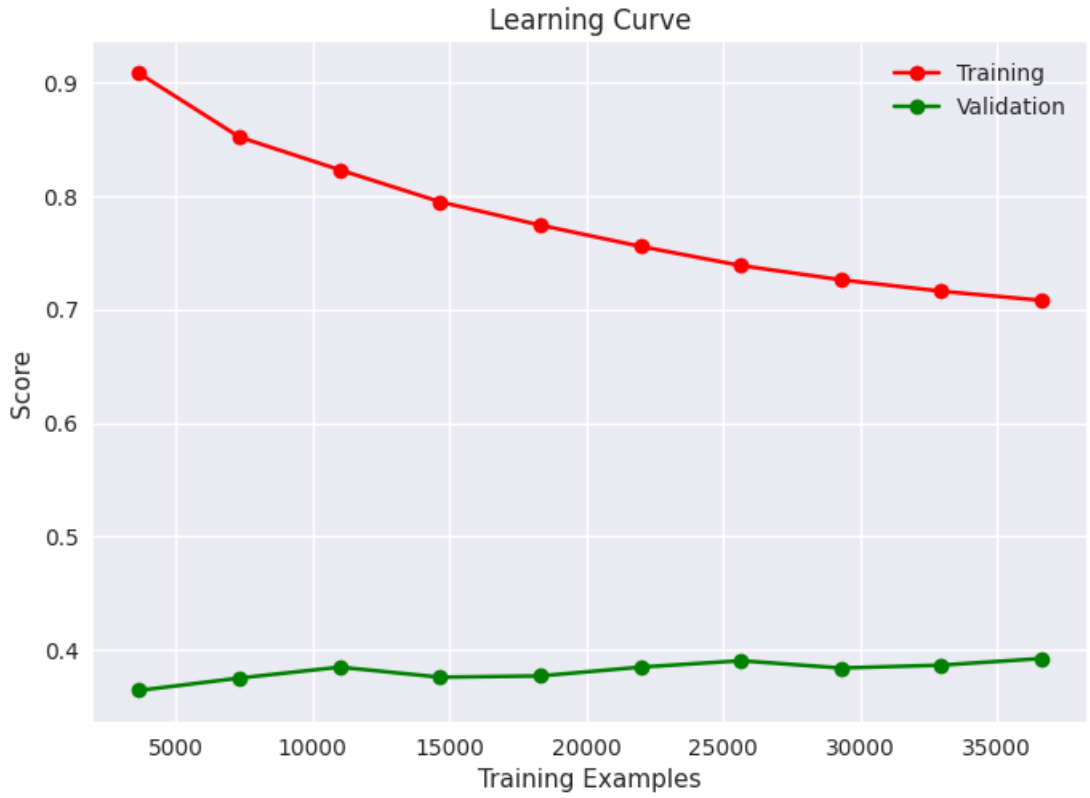

Figure 27: LIBLINEAR & L2 Learning Curve

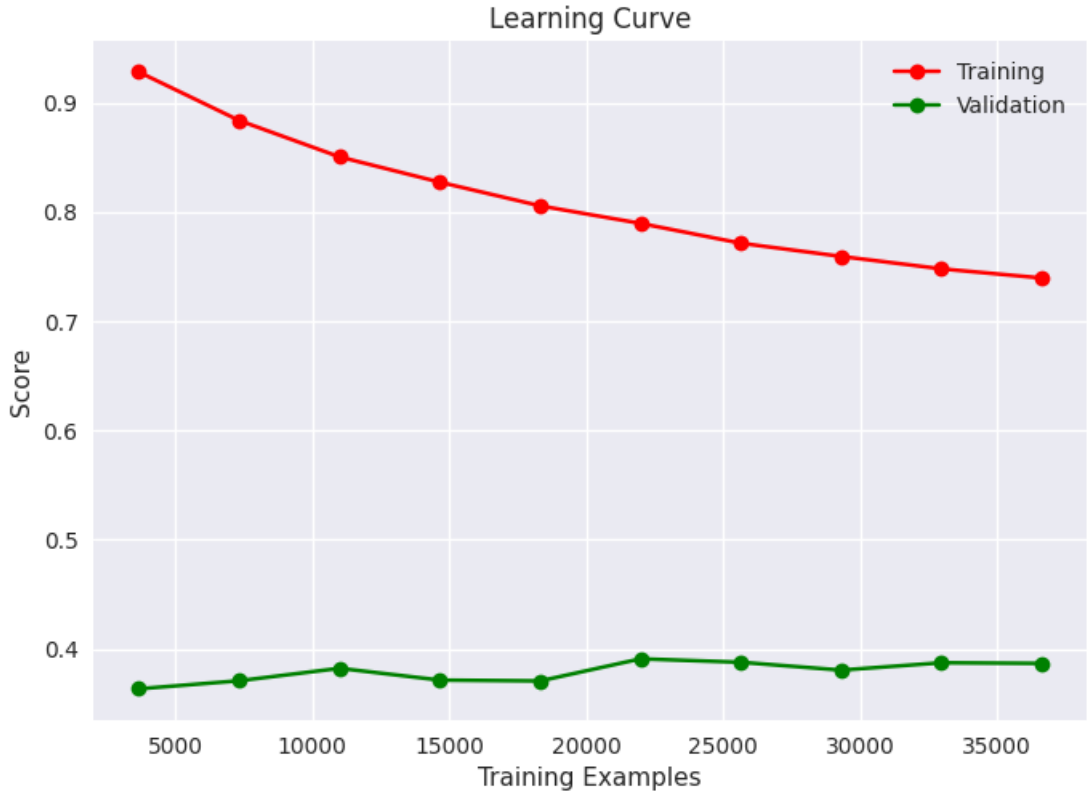

Figure 28: NEWTON-CG & L2 Learning Curve

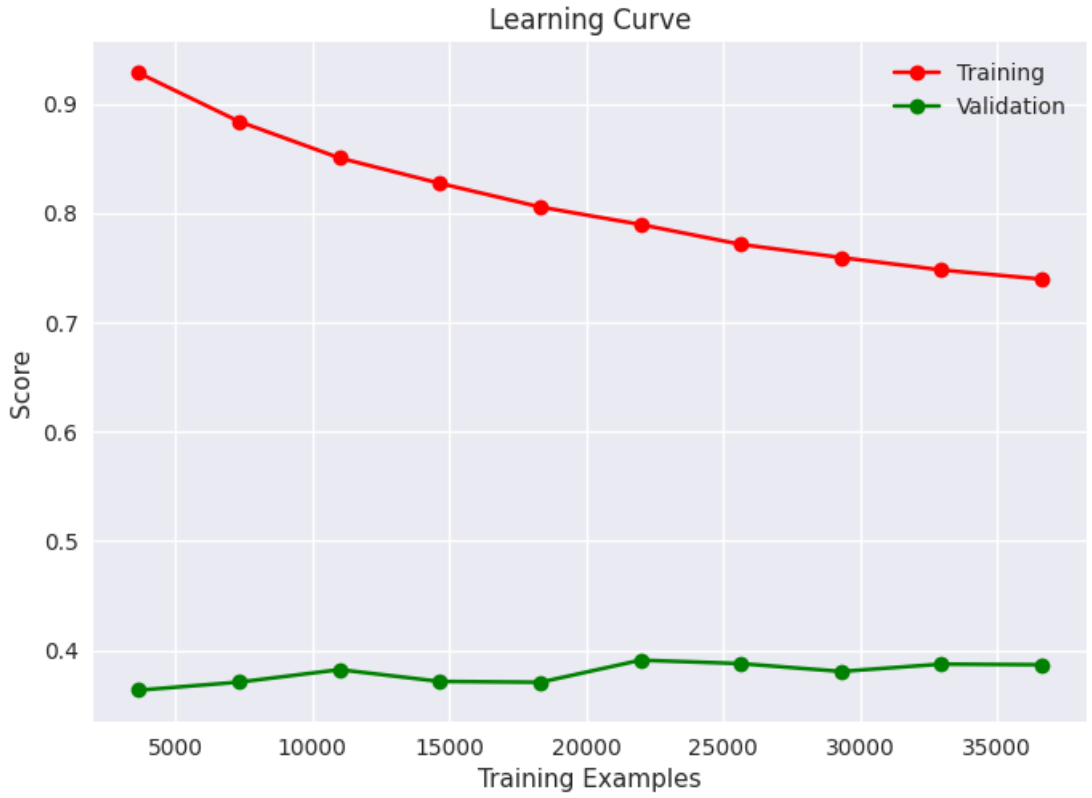

Figure 29: SAG & L2 Learning Curve

Figure 30: SAGA & L1 Learning Curve

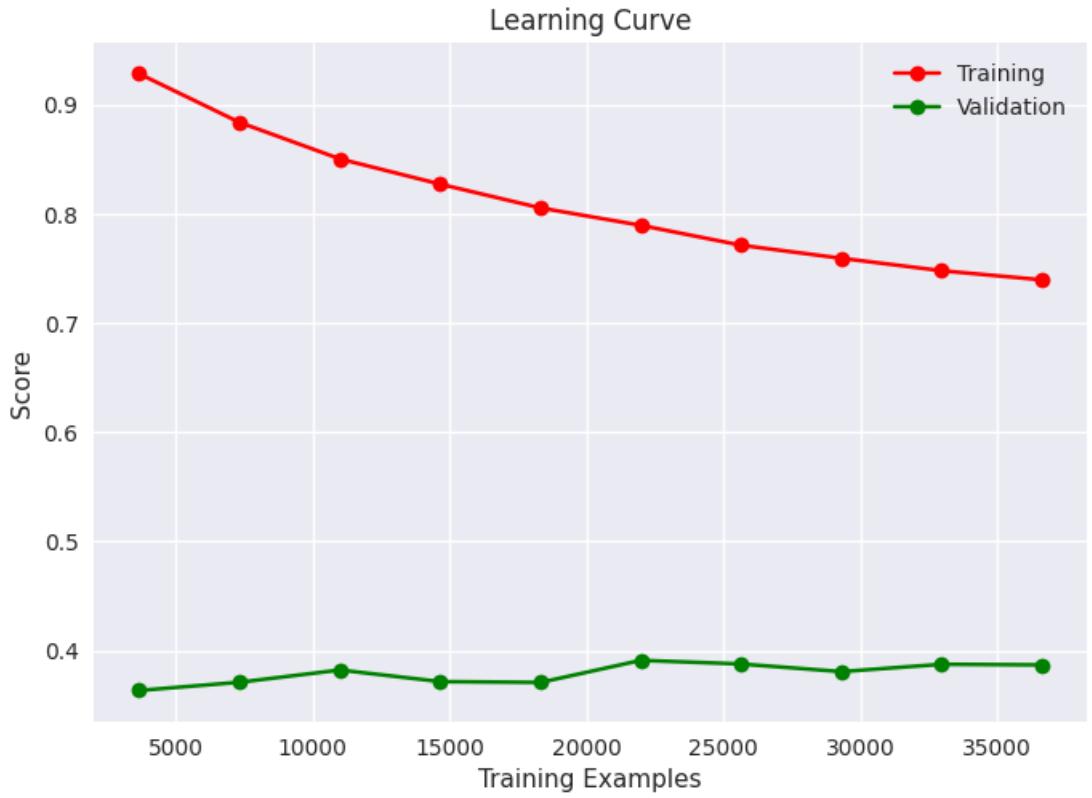

Figure 31: SAGA & L2 Learning Curve

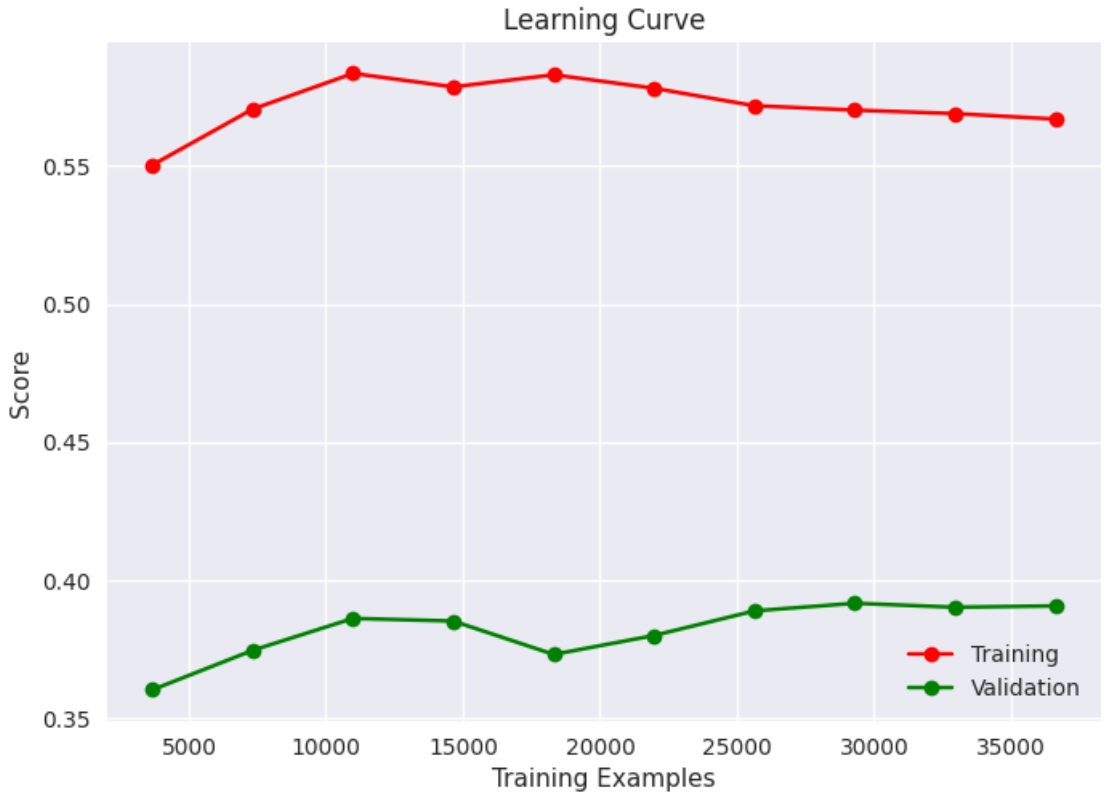

Figure 32: SAGA & ELASTICNET Learning Curve

When a training curve steadily declines but stays higher than the validation curve, like in the first category, it indicates that the model is too dependent on the details of the training set. Because the model becomes very specialized in capturing the unique characteristics of the training set—which may not be entirely reflective of the larger dataset—this situation might result in a lack of generalization on new data. Nevertheless, the second group represents a distinct overfitting phenomenon, as seen by a training curve that climbs sharply above the validation curve after beginning low. In one instance, the model fits noise or outliers unique to the training set in addition to capturing the innate patterns in the data. As a result, a complex model is produced that does well on training data but has issues generalizing to brand-new, unobserved cases.

3.4. Further Experiments

Our goal is to use the Logistic Regression parameter C to eliminate the overfitting issues that have been noted in the model trials before. As previously indicated, employing C to modify the regularization strength can aid in avoiding overfitting. We'll use the GridSearchCV (Cross-Validation with Grid Search) method since it's not feasible to manually test every combination. After GridSearchCV has finished running, we will look at the top parameter combinations for each C value other than the default one. This method enables us to evaluate whether modifying the regularization strength via C successfully reduces overfitting.

3.4.1. Learning Curve of trials.

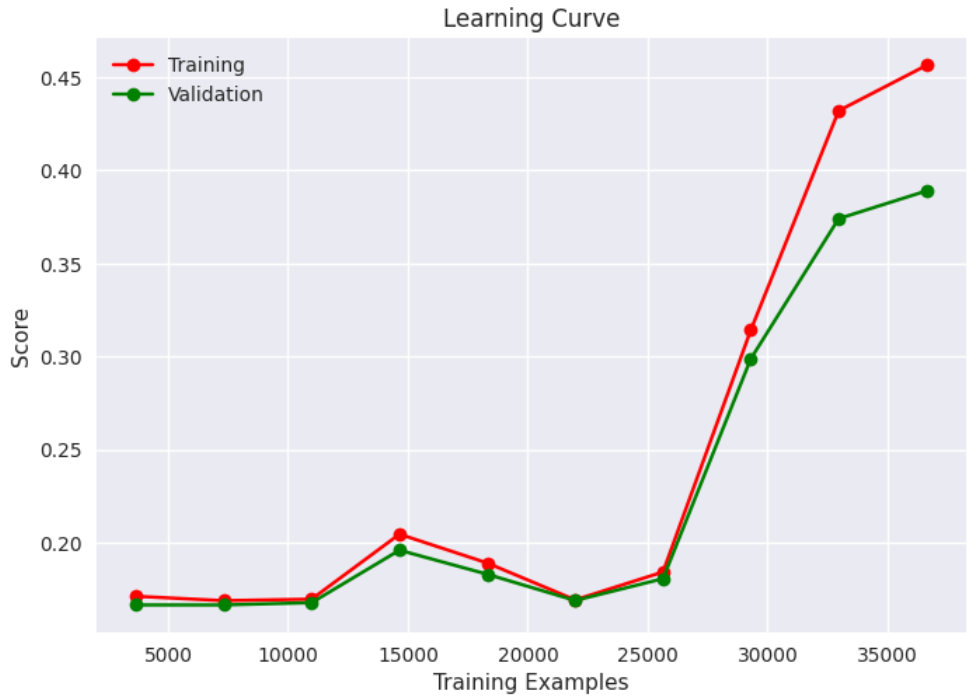

Figure 33: LIBLINEAR & L2 & C=0.001 Learning Curve

Figure 34: LBFGS & L2 & C=0.1 Learning Curve

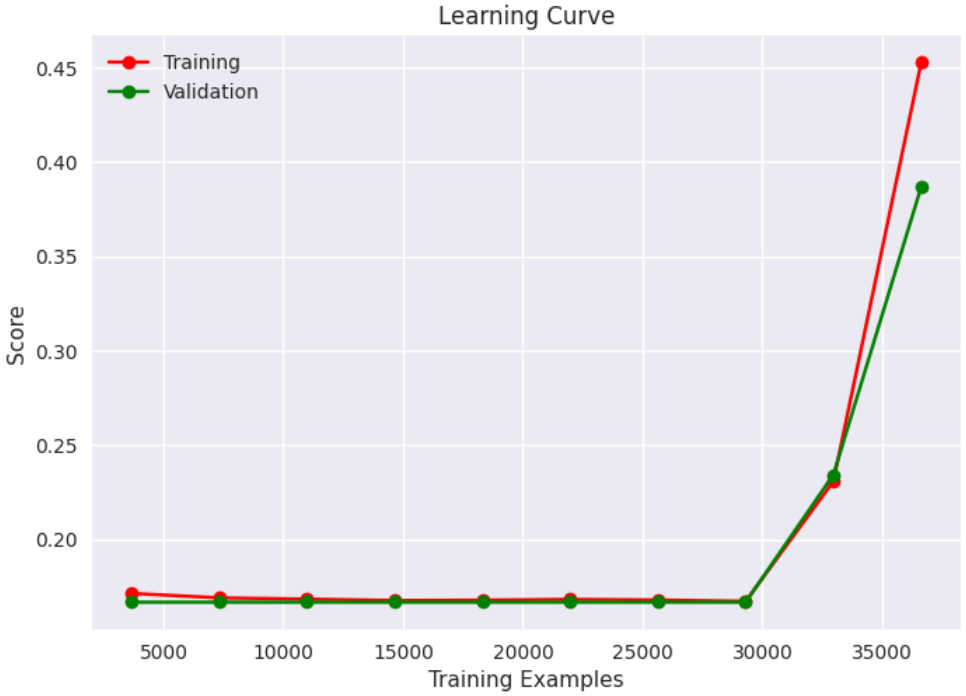

Figure 35: LBFGS & L2 & C=0.0001 Learning Curve

Figure 36: LIBLINEAR & L2 & C=0.01 Learning Curve

It is apparent that overfitting has been effectively reduced, although not in every model. Observations show that some models continue to show overfitting even when C values reach 0.01 and higher. Thus, it can be concluded that smaller regularization strength values are optimal for our model's deployment.

4. Results and Overall Analysis

4.1. Final Model

The final Logistic Regression model consists of the following:

- TF-IDF Vectorizer
- C = 0.001
- solver = liblinear
- penalty = l2

Below are the represenations of the ROC curve, learning curve and confusion matrix for the final mode:

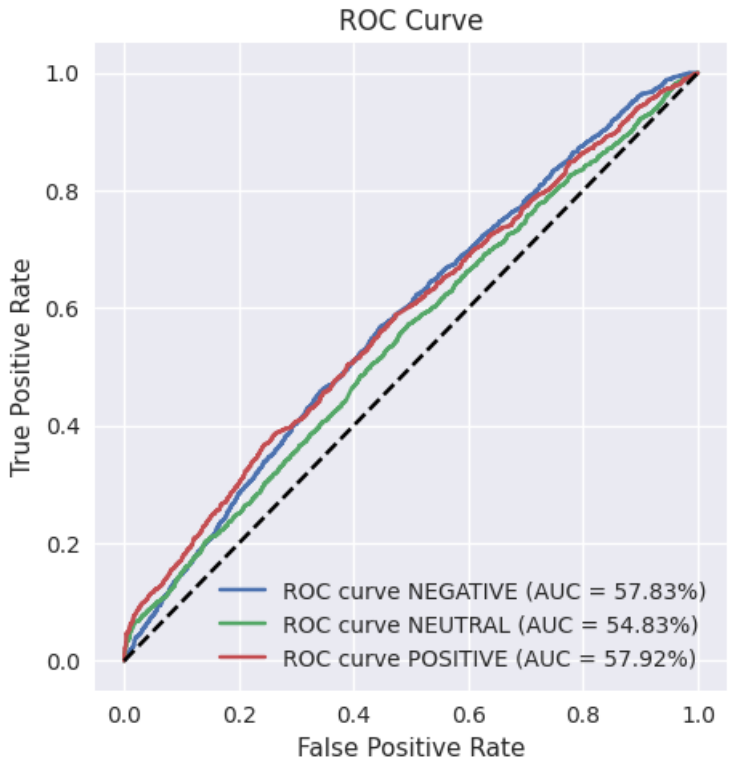

Figure 37: Final Model Roc Curve

In Class 1 (NEGATIVE), the AUC is more than 50%, which indicates that the ROC curve is comparatively superior than random chance. That isn't very high, though, suggesting that the model can only slightly outperform chance in distinguishing between instances of the NEGATIVE class. The AUC for Class 2 (NEUTRAL) is 54.83%, which is somewhat greater than 50%. This implies that although there is not much discrimination, the model performs marginally better for the NEUTRAL class than random chance. Class 3 (POSITIVE): The model can reasonably discriminate between cases for the POSITIVE class, according to the AUC of 57.92%. It's not a very strong performance, but it's better than chance.

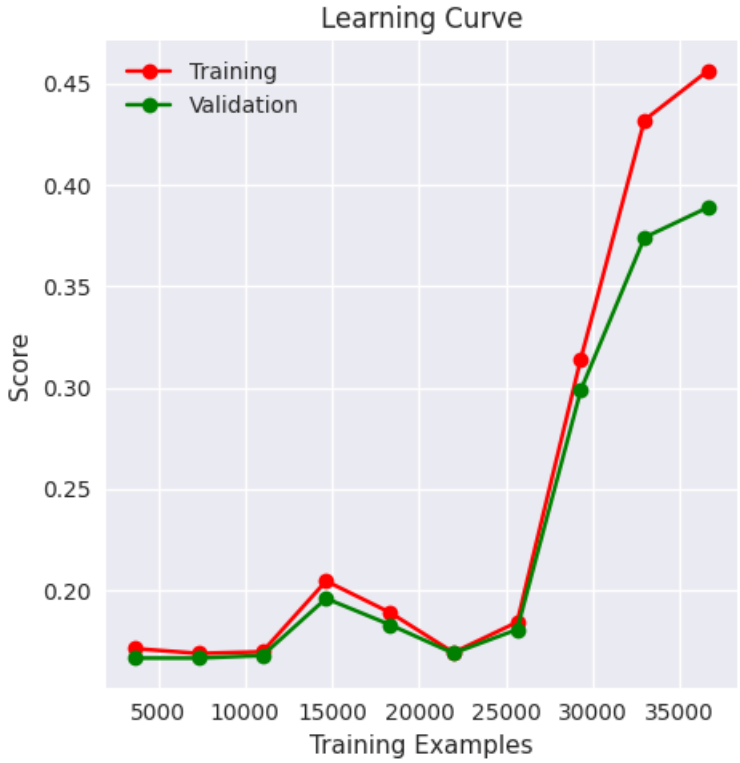

Figure 38: Final Model Learning Curve

The learning curve started off slightly, which was evidence of the early lack of skill in the training and validation datasets. But as the iterations went on, there was a noticeable increase. The training curve showed a consistent upward trend, indicating that the model was able to identify more intricate patterns in the data. Simultaneously, the validation curve had an upward trend, indicating the model's excellent applicability to previously unreported data.

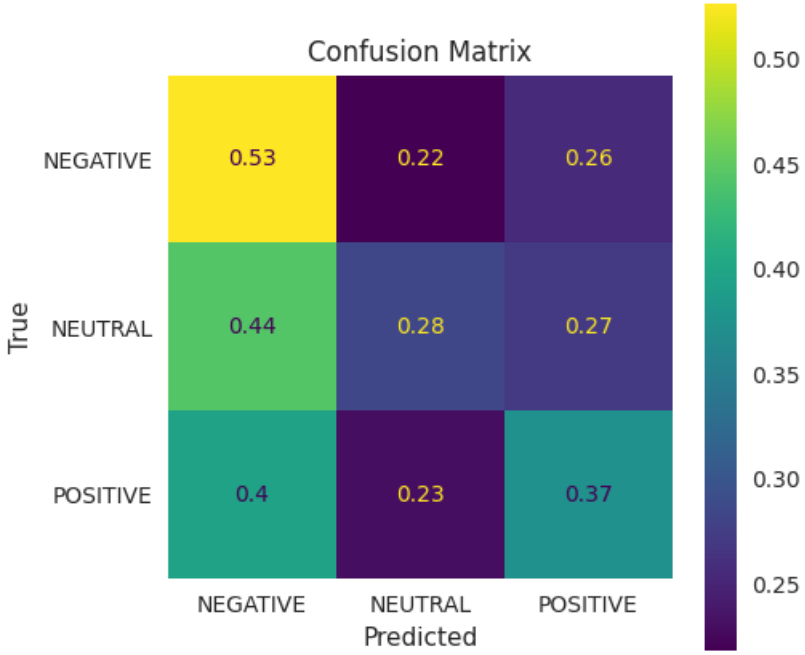

Figure 39: Final Model Confusion Matrix

A confusion matrix consists of the following:

- The true NEGATIVE is represented by the first row.
- The true NEUTRAL is represented by the second row.
- The true POSITIVE is represented by the third row.
- The predicted NEGATIVE is represented by the first column.
- The predicted NEUTRAL is represented by the second column.
- The predicted POSITIVE is represented by the third column.

The diagonal values are representing the correct prediction for each class:

- 53% for NEGATIVE
- 28% for NEUTRAL
- 37% for POSITIVE

5. Conclusion

Metrics show that the model is not as efficient as it could be; metrics are less than 40%. Regretfully, the restricted capabilities of the sole Greek text-supported tool (GreekStemmer) prevented any stemming test, since it lacked dynamism in comprehending different aspects of speech. Furthermore, Newton-Cholesky proved impracticable due of its high RAM requirements, which restricted research. Stochastic Gradient Descent (SGDClassifier) is the final setup I would have liked to try, especially with the 'log_loss' parameter, which is identical to logistic regression. This investigation was

limited, nonetheless, by instructional guidelines that oppose deviating from Logistic Regression. Despite these obstacles, the extensiveness of the procedure used lays a strong basis for further advancements. The constraints that have been found and the opportunities that remain untapped offer significant insights and may lead to improvements in future editions (next homeworks). This is very important as we go on to the next homework assignment, which is about enhancing our model by investigating different approaches like BERT, Feedforward Neural Networks (FNNs), and Recurrent Neural Networks (RNNs). These varied methods represent a critical step in enhancing the model's functionality and performance by offering a more creative and innovative environment.

6. Bibliography

References

[1] Ellogon.

[2] Greek-Stemmer/README.md at master · kpech21/Greek-Stemmer.

[3] Greek · spaCy Models Documentation.

[4] How to Evaluate a Logistic Regression Model?

[5] Multi-Class Sentiment Analysis on Twitter: Classification Performance and Challenges.

[6] Multiclass Receiver Operating Characteristic (ROC).

[7] Sentiment Analysis of tweets, Wordclouds, Textblob.

[8] sklearn.feature_extraction.text.CountVectorizer.

[9] sklearn.feature_extraction.text.HashingVectorizer.

[10] sklearn.feature_extraction.text.TfidfVectorizer.

[11] sklearn.linear_model.LogisticRegression.

[12] sklearn.metrics.confusion_matrix.

[13] sklearn.metrics.roc_curve.

[14] sklearn.model_selection.learning_curve.

[15] time — Time access and conversions.

[16] Token · spaCy API Documentation.

[17] Universal POS tags.

[18] wordcloud.WordCloud — wordcloud 1.8.1 documentation.

[19] Comparing various online solvers in Scikit Learn, May 2023. Section: Python.

[20] George B Aliman, Tanya Faye S Nivera, Jensine Charmille A Olazo, Daisy Jane P Ramos, Chris Danielle B Sanchez, Timothy M Amado, Nilo M Arago, Romeo L Jorda Jr, Glenn C Virrey, and Ira C Valenzuela. Sentiment Analysis using Logistic Regression. 2022.

[21] ArnavR. Scikit-learn solvers explained, March 2022.

[22] baeldung. What Is a Learning Curve in Machine Learning? | Baeldung on Computer Science, July 2020.

[23] Tim Bock. How to Show Sentiment in Word Clouds, February 2018.

[24] Jason Brownlee. How to Identify Overfitting Machine Learning Models in Scikit-Learn, November 2020.

[25] Amy @GrabNGoInfo. LASSO (L1) Vs Ridge (L2) Vs Elastic Net Regularization For Classification Model, March 2023.

[26] Dr Stylianos (Stelios) Kampakis. How to use learning curves in scikit-learn, April 2021.

[27] Vicky's Notes. Evaluating the Logistic Regression in Python, November 2022.

[28] RITHP. Logistic Regression and regularization: Avoiding overfitting and improving generalization, January 2023.

[29] Vinícius Trevisan. Multiclass classification evaluation with ROC Curves and ROC AUC, March 2022.