UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Christina Diamanti*
sdi: sdi1800046

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Spring Semester 2025*

Contents

1. Abstract

This assignment is about developing a sentiment classifier using Logistic Regression model and only TF-IDF vectorizer, for English-language Twitter data. The dataset consists of tweets. Especialy, we have three csv files, one for training, one for validation and one for testing. Train and Validation set consists of three columns ID, Text and Label (to label every text as positive or negative - with 1 and 0), the Test set is used only for predictions and consists of two columns ID and Text.
The task involves Exploratory Data Analysis (EDA), text preprocessing, model training and evaluation using metrics such as accuracy, precision, recall, and F1-score, and model testing.
To achieve this, we preprocess the text by removing unnecessary elements and convert it into numerical features using TF-IDF vectorizer. Then we train Logistic Regression model, optimize its hyperparameters, and evaluate its performance.

2. Data processing and analysis

2.1. Pre-processing

Pre-processing is the most important step to ensure that the data is clean and properly formatted for model training. First, duplicate tweets were removed. Next, unwanted symbols such as HTML, @mentions, numbers, and special characters were removed, and all text was converted to lowercase. After that, missing values (NaN) were dropped. You can see below the text (of train dataset) before cleaning and then the cleaned text column.

	ID	Text	Label
0	189385	@whoisralphie dude I'm so bummed ur leaving!	0
1	58036	oh my god, a severed foot was foun in a wheely...	0
2	190139	I end up "dog dialing" sumtimes. Wha...	1
3	99313	@_rachelx meeeee toooooo!	0
4	157825	I was hoping I could stay home and work today,...	0
...	...	...	...
148383	99894	just love the jonas brothers its tooo bad i w...	0
148384	61015	another day gone by....time is moving so fast...	0
148385	36598	fuck college, i'm just gonna marry rich. : fuc...	1
148386	83799	ZOMGZ NEW SONG FTW. remember that night. <3	1
148387	185558	http://twitpic.com/7mwrd - Arby's took down th...	0

	ID	Text	Label
0	189385	dude im so bummed ur leaving	0
1	58036	oh my god a severed foot was foun in a wheely ...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	meeeee toooooo	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	just love the jonas brothers its tooo bad i w...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college im just gonna marry rich fuck co...	1
148386	83799	zomgz new song ftw remember that night lt	1
148387	185558	arbys took down their roastburger coupon ...	0

Seeing that many Tweets have repetitive characters in a row, (such as "meeeee tooooo" and "tooo bad" in the example above) we used a function to reduce these repetitive characters in two. Below you can see the same text column with reduced characters.

	ID	Text	Label
0	189385	dude im so bummed ur leaving	0
1	58036	oh my god a severed foot was foun in a wheely ...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	mee too	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	just love the jonas brothers its too bad i wi...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college im just gonna marry rich fuck co...	1
148386	83799	zomgz new song ftw remember that night lt	1
148387	185558	arbys took down their roastburger coupon bu...	0

The words "im" and "just" appear to be too frequent and meaningless for model training so we remove them. Also, we replace some slang words (such as "ur", "ily", "fb") with their original form. Below you can see the final cleaned text column of train dataset. We apply the same text processing in all three datasets.

	ID	Text	Label
0	189385	dude so bummed you are leaving	0
1	58036	oh my god a severed foot was foun in a wheely ...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	me too	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	love the jonas brothers its too bad i wil neve...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college gonna marry rich fuck college gon...	1
148386	83799	zomgz new song ftw remember that night lt	1
148387	185558	arbys took down their roastburger coupon but i...	0

The removal of Stopwords (such as a, an, the, of, in, for) and lemmatization techniques did not ease the model training process, on the contrary, they made model training more difficult and led to a reduction in accuracy and overfitting. For this reason, these techniques were removed from the final project. Below, in the Trial Table you will see this difference in results.

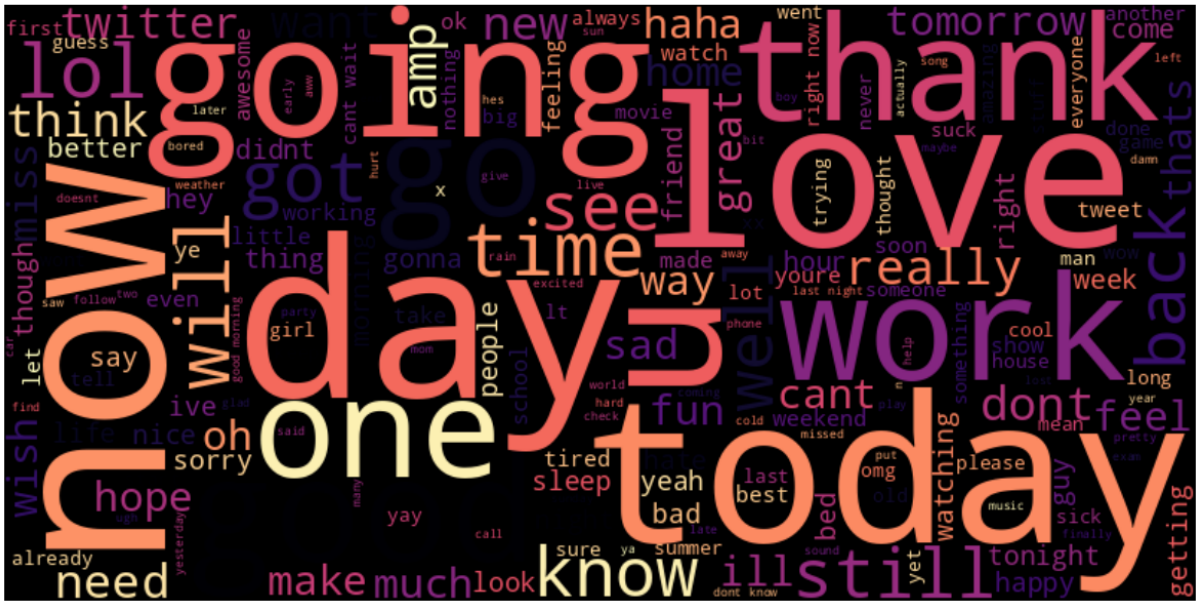
2.2. Analysis

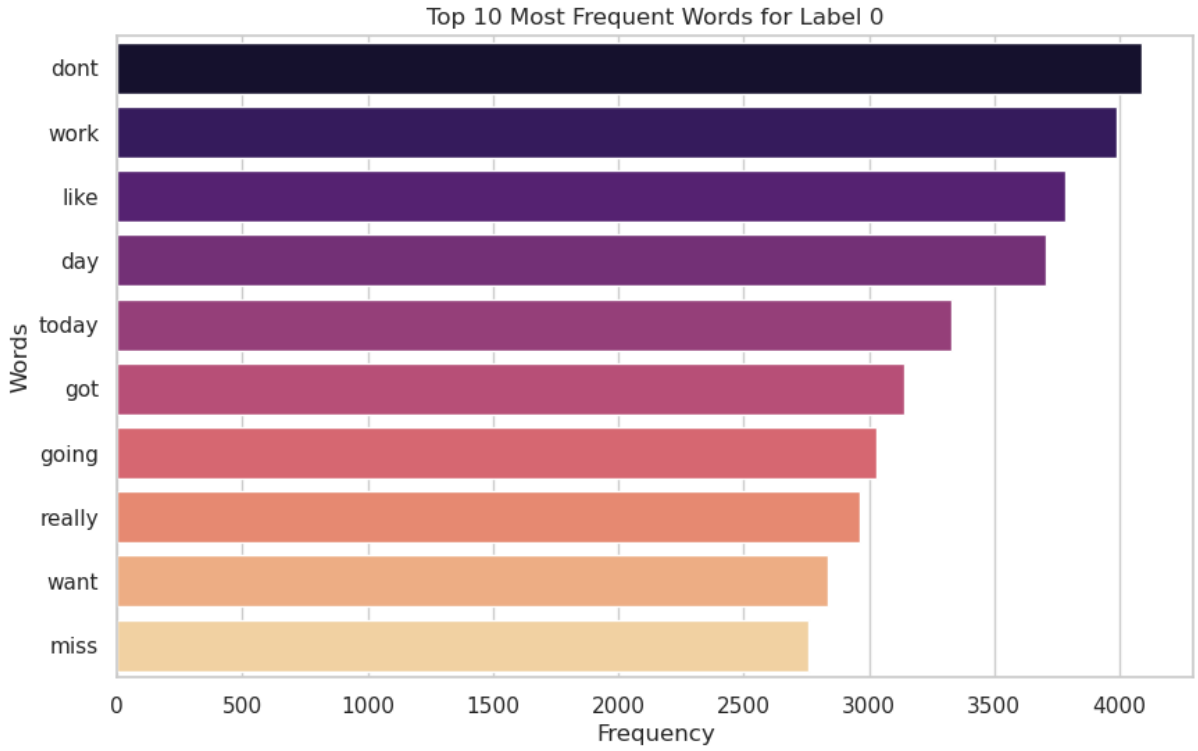
To get a clearer picture of our data we tried to visualize it by using wordclouds and also printing the number of every label per dataset. We also made a plot of the Top 10 most frequent words per label. Below you can see the wordcloud of every dataset and the plots for the Top 10 words.

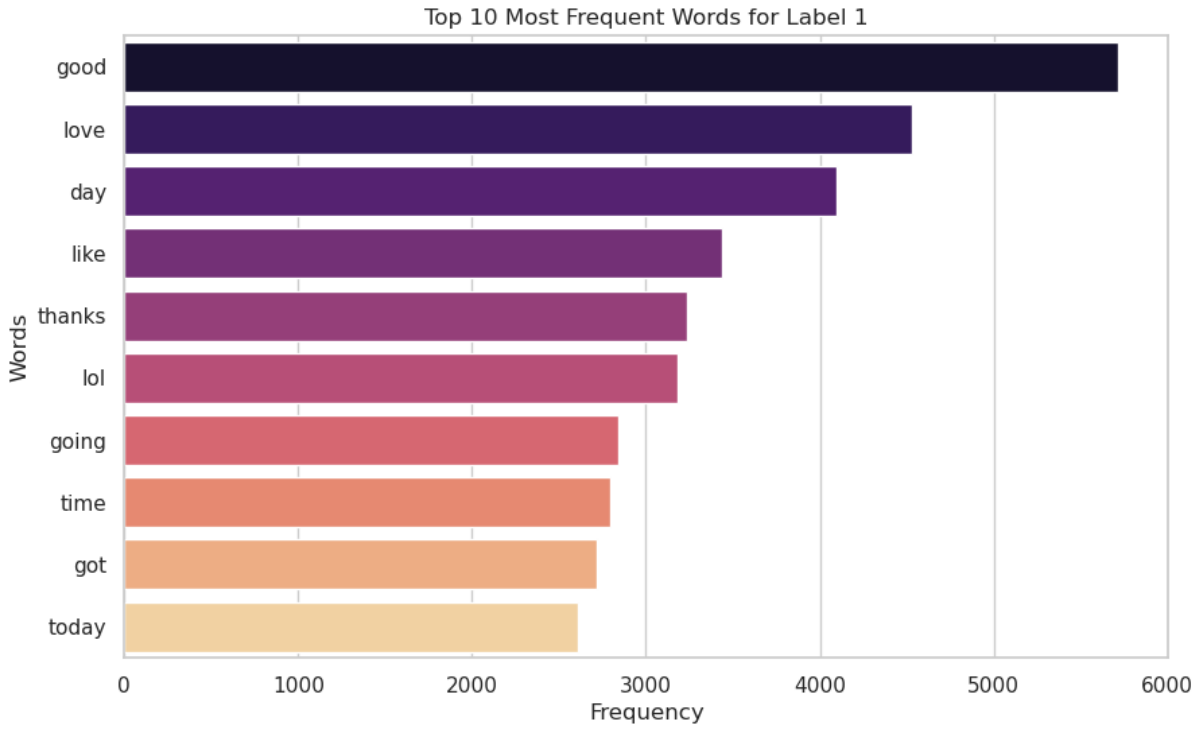
Word Cloud for Training data

Word Cloud for Validation data

Word Cloud for Tetsing data







2.3. Data partitioning for train, test and validation

The datasets were already split into 3 parts, Train set was used to train Logistic Regression model, Validation set was used to evaluate the predictions of the model based on metrics such as accuracy, precision, recall, and F1-score and Test set was used to make predictions with our trained model and save them in a csv file.

2.4. Vectorization

In this implementation, the Term Frequency-Inverse Document Frequency (TF-IDF) Vectorizer was used to convert text into numerical features (`TfidfVectorizer` from `sklearn.feature_extraction.text`). TF-IDF assigns weights to words based on their importance in a document relative to the entire dataset. The TF-IDF score is calculated as:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

where:

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

and

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing the term t}}\right)$$

3. Algorithms and Experiments

3.1. Experiments

In the first trial (trial 1), we used word lemmatization, then we trained the model using the default parameters (penalty='l2', *, dual=False, tol=0. 0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='deprecated', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None).
Then, to increase the accuracy, experiments and changes were made using hyperparameters. In particular, increasing max_iter to 200, 300, or 1.000 did not change anything, probably because the model has already learned the best possible from the data and more iterations do not offer any further improvement.
The next experiments involve changes in parameter C. First testing C with values less than 1.0 (default value), such as 0.1, 0.001 but this had the effect of reducing accuracy. Testing higher C also had the effect of reducing accuracy ($C = 2$, 5 or 10), as shown in the table below (trial 2). It was observed that for C=0.9 we had a minimal increase in accuracy (trial 3) so we kept $C = 0.9$.
Then experiments were done with the solver parameter, where we concluded that 'saga' and 'liblinear' give slightly better accuracy (both are the same), so 'liblinear' was chosen (trial 4) as it supports penalty = "l1" and "l2", and it runs faster than 'saga'.
The other parameter changes did not improve the accuracy, so we ended up with the definition of Logistic Regression with hyperparameters $C = 0.9$ and solver='liblinear' with penalty = 'l2' (the default).
Since the model could not produce even better accuracy, changes were made to the hyperparameters of the TF-IDF vectorizer, where the parameter ngram_range=(1, 2) was added so that the model could use both unigrams and bigrams to capture not only single words, but also important relationships between words that often appear together. This technique increased accuracy (trial 5).
Finally, we run the notebook online in Kaggle, where we remove lemmatization technique, since Kaggle did not support lemmatization with the nltk library, while the

spaCy library took too long to lemmatize the words, and an even greater increase in accuracy was observed (trial 6).

Since the hyperparameters did not seriously increase the accuracy, we came to the conclusion that data preprocessing plays a very important role for model training and so we returned and focused on it.
In the last trial below (Trial 7) you can see a significant increase in metrics. To achieve this we did not include stopwords removal, since it seems that these words help the model to understand and distinguish with better accuracy the label for each text. Then we saw that some words that appear in the "most common words per label" are not particularly meaningful, so we removed them. Also, we reduced repeated characters in text and we replace some slang words with their original form, as mentioned above.

3.1.1. Table of trials. Below there is a table that contains the trials described.

Trial	Precision	Recall	F1-score	Accuracy
1	0.7646	0.7803	0.7724	77.01%
2	0.7543	0.7632	0.7587	75.74%
3	0.7649	0.7819	0.7733	77.09%
4	0.7652	0.7823	0.7736	77.12%
5	0.7781	0.7738	0.7760	77.66%
6	0.7784	0.7762	0.7773	77.77%
7	0.8064	0.7911	0.7987	80.06%

Table 1: Trials

3.2. Hyper-parameter tuning

The optimization of hyperparameters (we used manual tuning) had a limited effect on the performance of Logistic Regression, with accuracy ranging between $75\% - 77\%$. This shows that the model did not suffer heavily from over-fitting or under-fitting, and that its performance is more influenced by the nature of the data than by tuning. The parameter C and the penalty parameter (L1/L2), did not produce significant differences, which may suggest that Logistic Regression model has reached its limit in this problem.

3.3. Optimization techniques

Various optimization techniques were used, mainly through different solvers. First the solver 'lbgfs' (which is the default) was tested with penalty 'l2', then 'saga' with penalty 'elasticnet' (a combination of 'l1' and 'l2') and then 'liblinear' with penalty 'l1' and with penalty 'l2'. As mentioned above, 'saga' and 'liblinear' had the best accuracy performance, while 'liblinear' was faster than 'saga', so the solver 'liblinear' with penalty 'l2' was chosen.

3.4. Evaluation

To evaluate the predictions of the model, we used val_dataset.csv and the metrics Accuracy, Precision, Recall and F1-score. The accuracy of the model reached 80.06%, while Precision (0.8064) and Recall (0.7911) show that the model maintains a good balance between false positives and false negatives.
F1-score (0.7987),

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

as a harmonic mean of Precision and Recall, confirms the stable performance of the model.

Below there are plots and diagrams of ROC curve, Learning Curve and Confusion matrix. We will show the same diagrams for the model without optimizations (Trial 1) and for optimized model (Trial 7).

3.4.1. ROC curve. The Receiver Operating Characteristic (ROC) curve depicts the relationship between True Positive Rate (TPR) and False Positive Rate (FPR). Area Under the Curve (AUC) measures the overall performance of the model, in the plots below, the two ROC curves with AUC 0.85 and 0.88, shows that the model, in both cases, has good classification ability (since AUC > 0.8), with the second (optimized) model showing a slightly better performance.

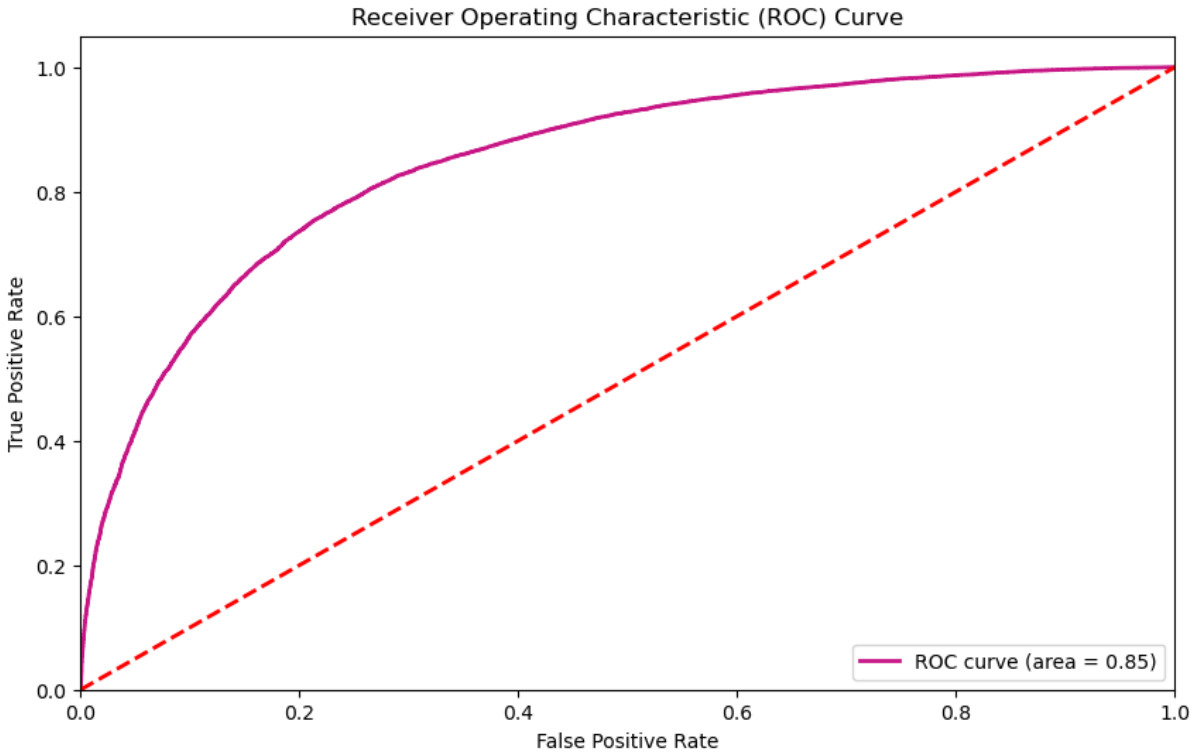


Figure 1: Trial 1 ROC curve

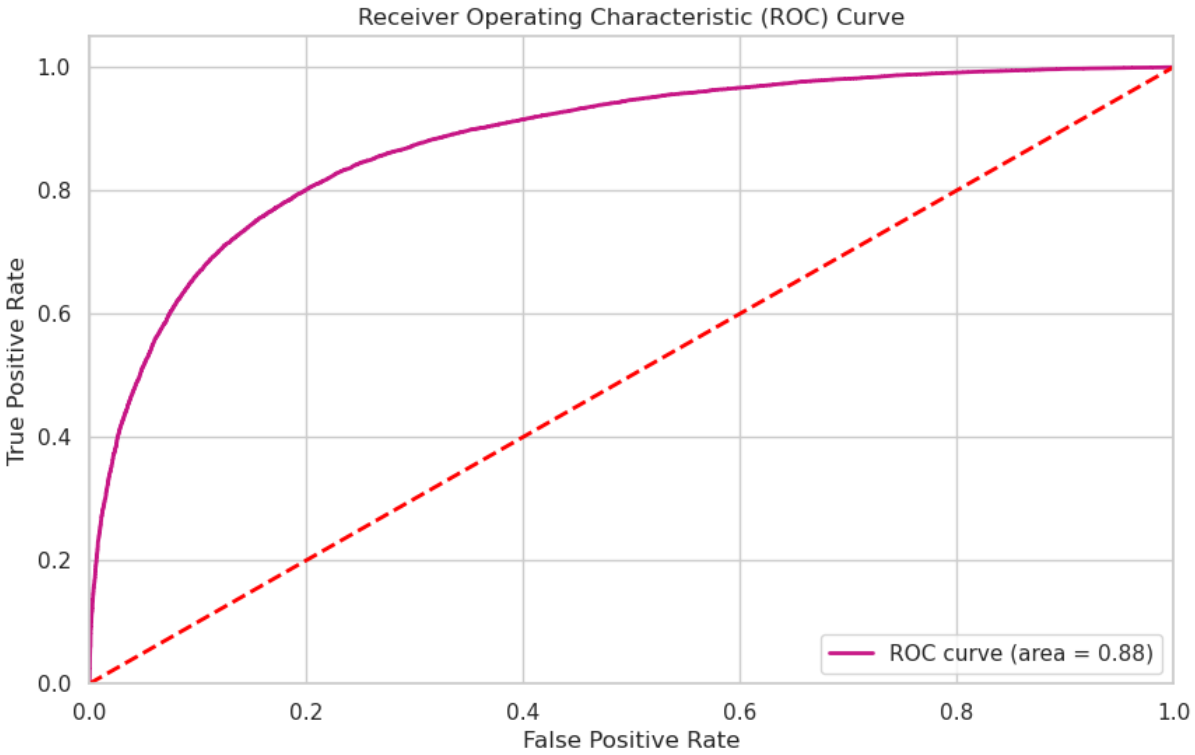


Figure 2: Trial 7 ROC curve

3.4.2. Learning Curve. Comparison of the learning curves below shows that the Logistic Regression model without hyperparameters (Trial 1) shows over-fitting, as the training score is quite high (0.86) and decreases as the training set size increases. The validation score is significantly lower (0.74 initially, reaching 0.76), indicating a large gap between training and validation performance.
On the other hand, the training score of the optimized model (Trial 7) starts higher (0.93) but decreases less than in simple Logistic Regression. The validation score is higher (0.75 initially, reaching almost 0.80), and the distance between training and validation accuracy is smaller. This indicates that the optimized model has reduced over-fitting, as the gap between training and validation accuracy has narrowed and validation accuracy has improved.

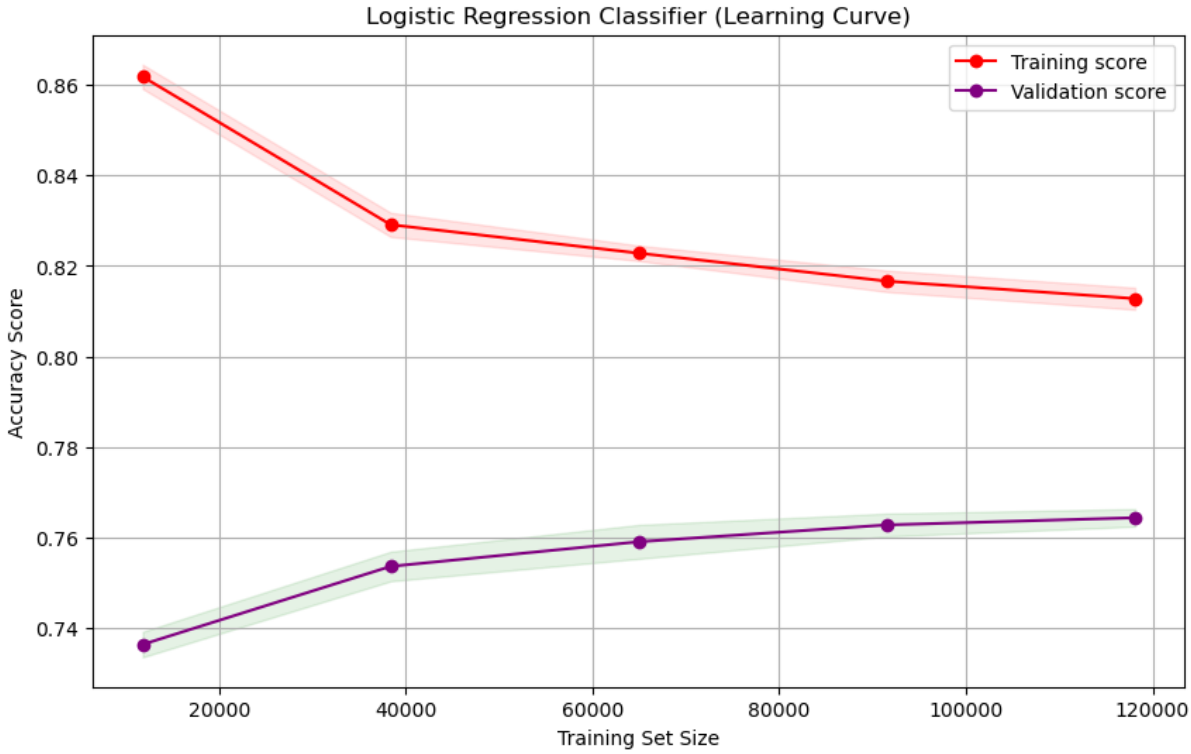


Figure 3: Trial 1 Learning curve

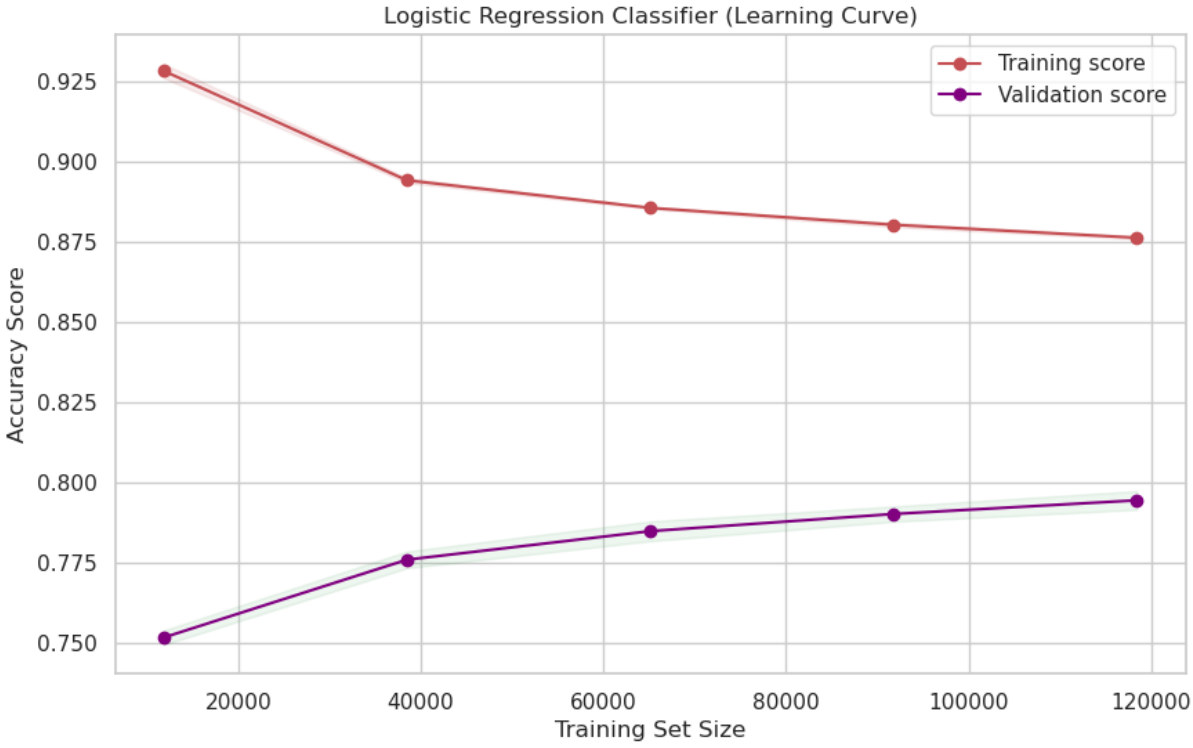


Figure 4: Trial 7 Learning curve

3.4.3. Confusion matrix. After data preprocessing and hyperparameter tuning of the Logistic Regression model, we observe clear improvements in its performance. False Positives decreased $(5.068 \rightarrow 4.017)$, meaning the model makes fewer incorrect positive predictions. At the same time, True Negatives increased $(16.044 \rightarrow 17.148)$, indicating a better ability to correctly classify negative cases.
Additionally, False Negatives slightly decreased $(4.635 \rightarrow 4.419)$, reducing the number of missed positive cases. True Positives also increased $(16.463 \rightarrow 16.733)$, showing improved accuracy in identifying positive instances.

These changes suggest that the model now generalizes better. The results are visualized in the plots below.

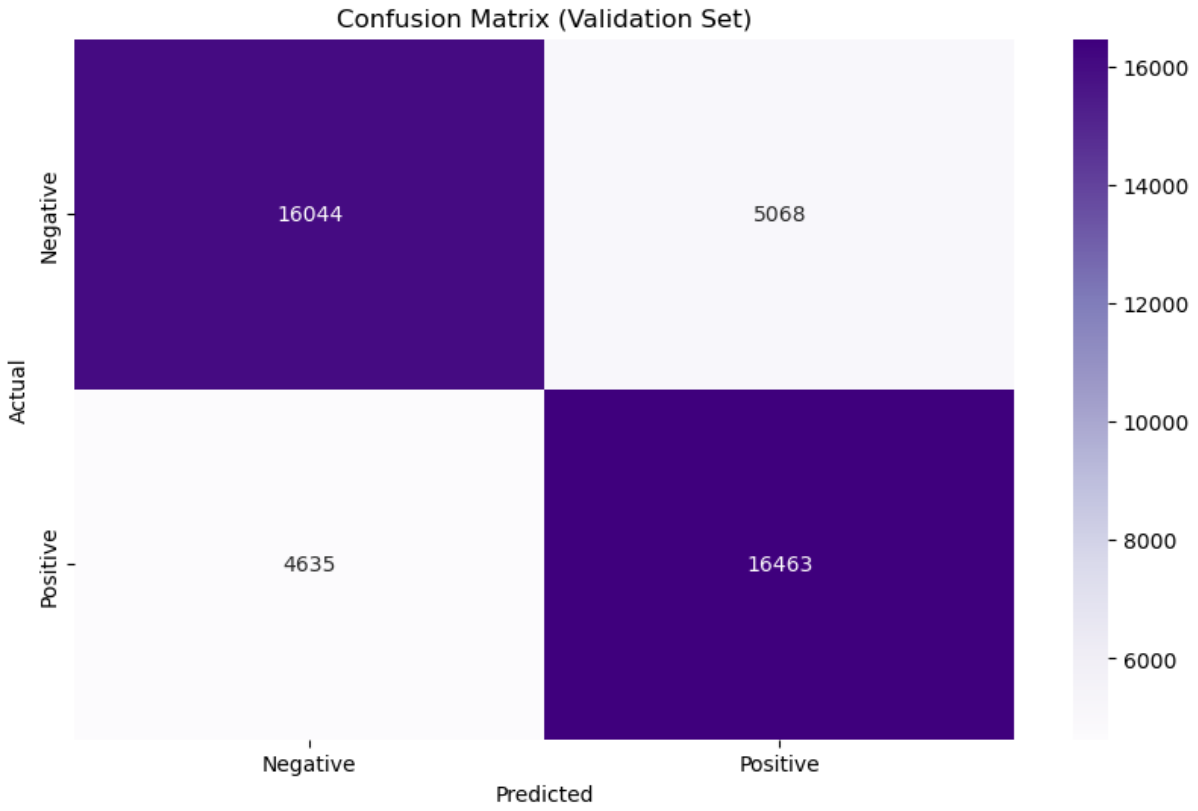


Figure 5: Trial 1 Confusion matrix

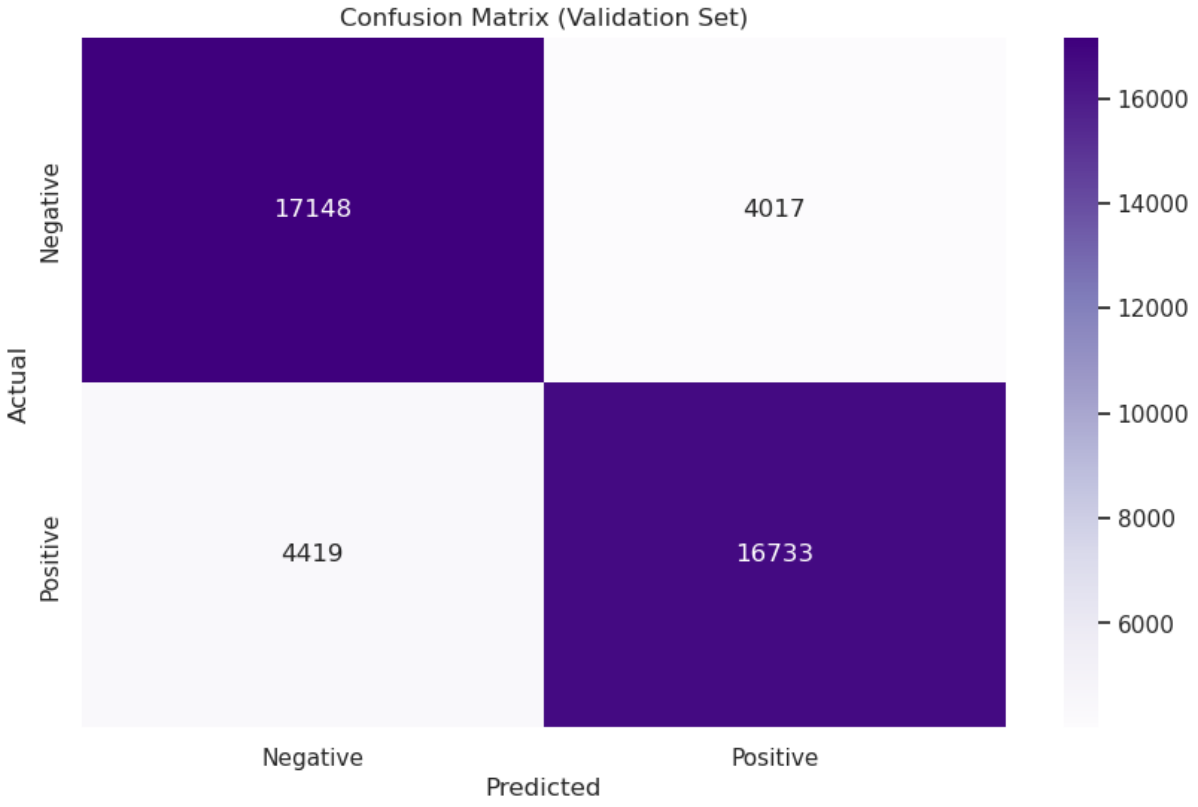


Figure 6: Trial 7 Confusion matrix

4. Results and Overall Analysis

4.1. Results Analysis

The results show that the best performance of the Logistic Regression model was achieved with C=0.9, solver='liblinear' and penalty='l2', as well as by optimizing the TF-IDF vectorizer by adding ngram_range=(1,2).

The final accuracy of the model was 80.06%, which is satisfactory but not excellent. The F1-score (0.7987) indicates that the model has a good balance between recall and precision, which means that it does not overly favor one category over the other.

The ROC Curve showed that the AUC improved from 0.85 to 0.88 after applying optimizations, indicating that the model has good discriminatory power. However, the Learning Curve shows that there is still some over-fitting, as the performance on the validation set remains slightly lower than the training set.

The Confusion Matrix revealed that the model significantly reduced False Positives $(5,068 \rightarrow 4.017)$ and increased True Negatives $(16,044 \rightarrow 17.148)$, which improves its ability to correctly identify negative examples. As mentioned above, there was a slight decrease in False Negatives $(4,635 \rightarrow 4.419)$ and an increase in True Positives $(16.463 \rightarrow 16.733)$ indicating that the model showing improved accuracy in identifying both positive and negative instances.

Overall, the model performs well, but there is certainly room for improvement. To reduce over-fitting we could try adding more data. Also, because Logistic Regression is a simple linear model, we could use a more complex model such as SVM or Random Forest, which are more resistant to over-fitting.

4.1.1. Best trial. The best experiment (Trial 7) achieved a maximum accuracy of 80.06%, which was the highest of all the other trials.

The final metric values for the Best Trial (Trial 7) are:
Accuracy: 80.06%
Precision: 0.8064
Recall: 0.7911
F1-score: 0.7987

In this trial, we removed lemmatization and stopwords technique, since without it gave better results in training and validation. Also added ngram_range=(1,2) to the TF-IDF vectorizer, allowing the model to recognize not only single words (unigrams) but also pairs of words (bigrams), thus improving contextual understanding.

5. Bibliography

References

[1] GeeksforGeeks. Auc-roc curve, 2024.

[2] GeeksforGeeks. Python lemmatization with nltk. Online, 2024.

[3] GeeksforGeeks. Understanding tf-idf (term frequency-inverse document frequency), 2024.

[4] Daniel Jurafsky and James H. Martin. Speech and language processing, chapter 4, 2024.

[5] Daniel Jurafsky and James H. Martin. Speech and language processing, chapter 5, 2024.

[6] Scikit-learn Developers. Model evaluation in scikit-learn, 2024.

[7] Scikit-learn Developers. sklearn.feature_extraction.text.tfidfvectorizer, 2024.

[8] Scikit-learn Developers. sklearn.linear_model.logisticregression, 2024.

[9] Stack Overflow Community. Using sklearn's logisticregression with saga solver and elasticnet penalty, 2024.

[10] V7 Labs. F1 score guide, 2024.