UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: ΙΩΑΝΝΑ ΠΟΥΛΟΥ
sdi: <sdi2100161>

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

Contents

1. Abstract

<Briefly describe what's the task and how you will tackle it.>

The goal of the task is to create a sentiment classifier using logistic regression for a twitter dataset about the Greek general elections.However, the main aim of the exercise is not to develop the best model,but rather to familiarize ourselves with the libraries and experiment over various techniques,which I will describe in detail later.

2. Data processing and analysis

2.1. Pre-processing

<In this step, you should describe and comment on the methods that you used for data cleaning and pre-processing. In ML and AI applications, this is the initial and really important step.

For example some data cleaning techniques are: Dropping small sentences; Remove links; Remove list symbols and other uni-codes.>

The first step i did for this task was to clean the data in order to increase the quality of them.Specifically,I removed from my data stop words,urls,emojis,spaces,mentions, panctuation and others special characters.Moreover, I dropped small and very large words and i removed the verbs and the accents of the words in order not to let them affect my data.

2.2. Analysis

<In this step, you should also try to visualize the data and their statistics (e.g., word clouds, tokens frequency, etc). So the processing should be done in parallel with an analysis. >

It is worth noting that before and after Pre-processing,I visualized the data using word clouds and tokens frequency diagrams.In this way ,i could understand how efficient the data clean was and it helped me to make Pre-processing better.For example,thanks to word clouds,i get rid of the word 'χ',which for some reason was the most common word.The word clouds before and after the data cleaning are listed below, while you can create tokens frequency diagrams by running the related code in my notebook.

Figure 1: Word Cloud before Preprocessing

Figure 2: Word Cloud after Preprocessing

2.3. Data partitioning for train, test and validation

<Describe how you partitioned the dataset and why you selected these ratios>

I let train and validation datasets as they was given to us.The ratio 80:20 is the optimal since we want variety of training data,but we also don't want our model overfits them.It is worth noting that,among others, I drop from Xtrain the column "Party", because the cost of using the Parties was greater than their contribution to model's performance.

2.4. Vectorization

<Explain the technique used for vectorization>

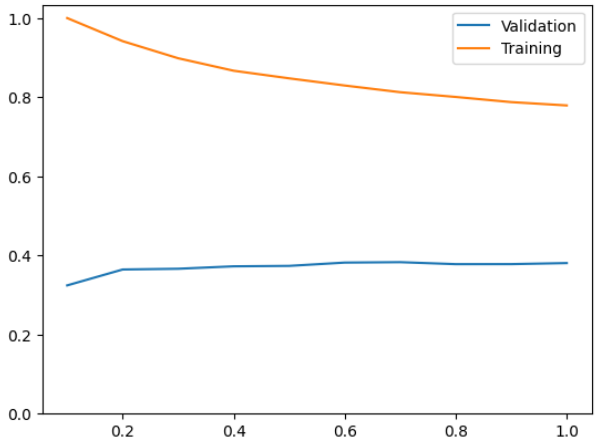

Figure 3: Learning Curve after Pre-process using CountVectorizer

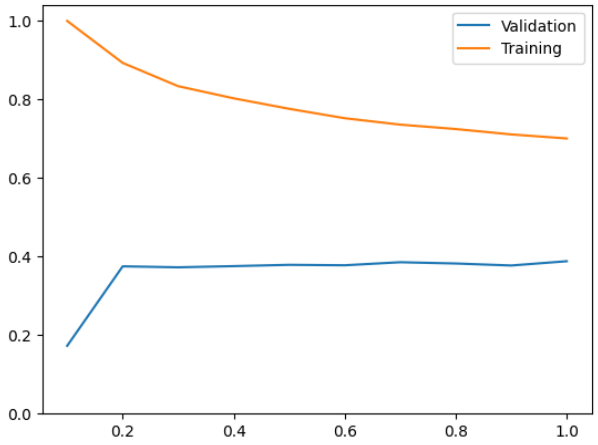

Figure 4: Learning Curve after Pre-process using TF-IDF

3. Algorithms and Experiments

3.1. Experiments

<Describe how you faced this problem. For example, you can start by describing a first brute-force run and afterwords showcase techniques that you experimented with. **Caution:** we want/need to see your experiments here either they increased or decreased scores. At the same time you should comment and try to explain why an experiment failed or succeeded. You can also provide plots (e.g., ROC curves, Learning-curves, Confusion matrices, etc) showing the results of your experiment. Some techniques you can try for experiments are cross-validation, data regularization, dimension reduction, batch/partition size configuration, data pre-processing from 2.1, gradient descent>

In this section,i will represent you the results of some techniques i used,in order to check the scores that they create.It is worth noting, that you can also run the related code in my Notebook to see by yourself the results.

3.1.1. FIRST BRUTE-FORCE RUN. In this case,we start training our model without using any methods,including Pre-processing.As a result, there are words that don't contribute much to the meaning of the text and add noise to the data.Moreover,text data is inherently high-dimensional, with almost each word representing a unique feature and because of this large number of features, the F1 training score is exceptionally high,while the F1 validation score reaches a ceiling (0.38-0.39), leading to overfit.

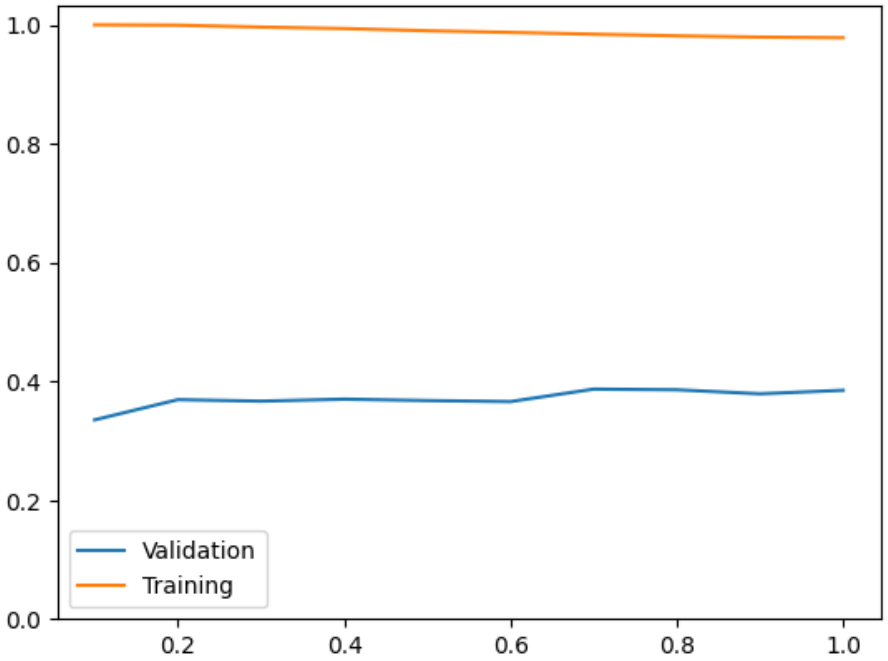

Figure 5: Learning Curve Before Pre-processing

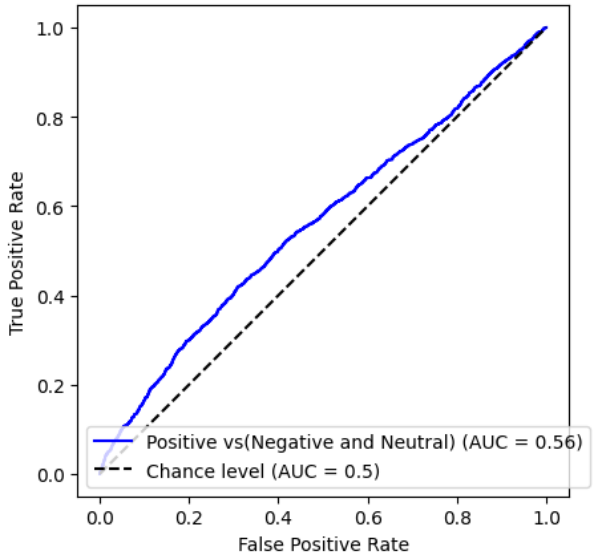

Figure 6: ROC Curve(Positive vs the rest)

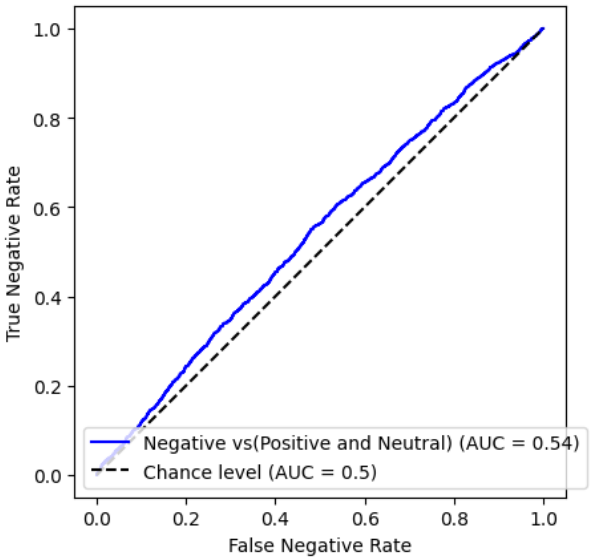

Figure 7: ROC Curve(Negative vs the rest)

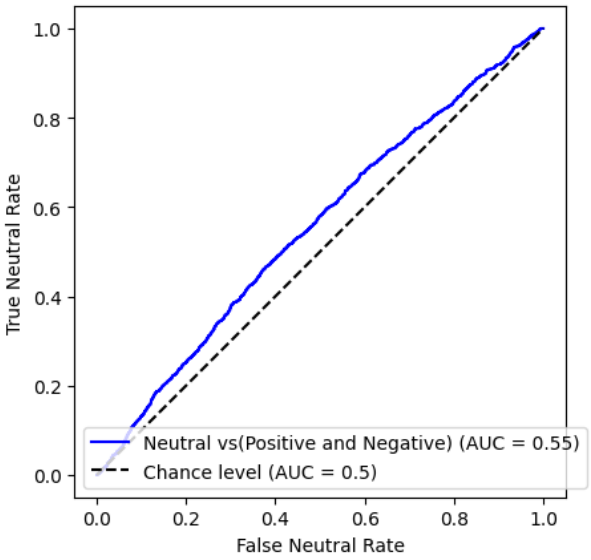

Figure 8: ROC Curve(Neutral vs the rest)

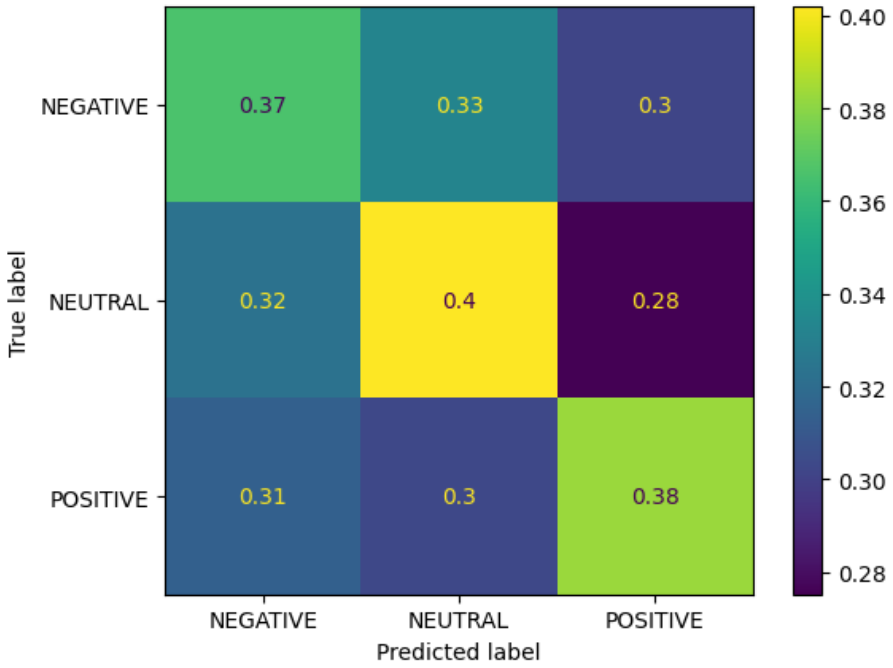

Figure 9: Confusion Matrix Before Preprocess

3.1.2. PRE-PROCESSING. As we explained before, Pre-processing is the initial and one of the most important steps before training our model.More specifically, Pre-processing refers to the techniques applied to initial data (before they are used as an input into a model) in order to clean and organize them.In our model,as we expected, the results after pre-processing are better than before, but still overfitting occurs, which may be due to insufficient data or model's complexity.Moreover, ROC curves still remaining almost the same.

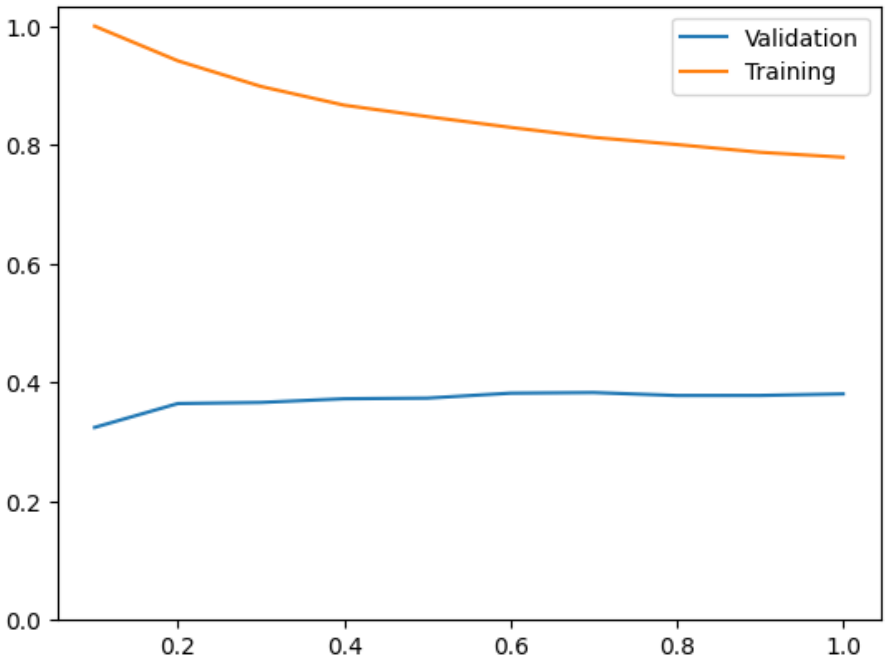

Figure 10: Learning Curve After Pre-processing

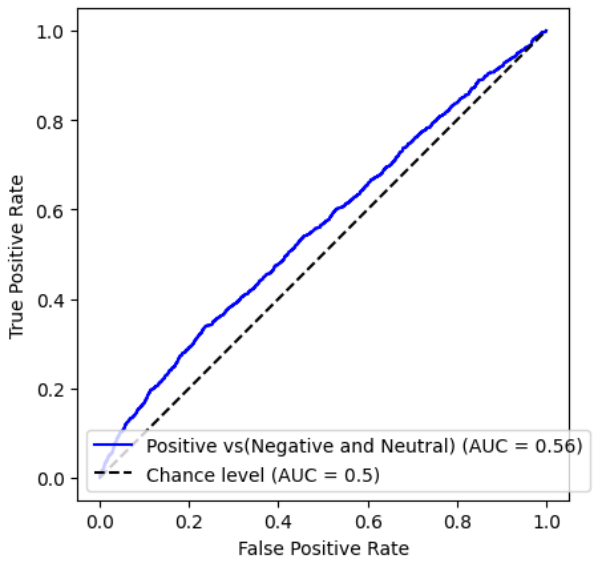

Figure 11: ROC Curve(Positive vs the rest)

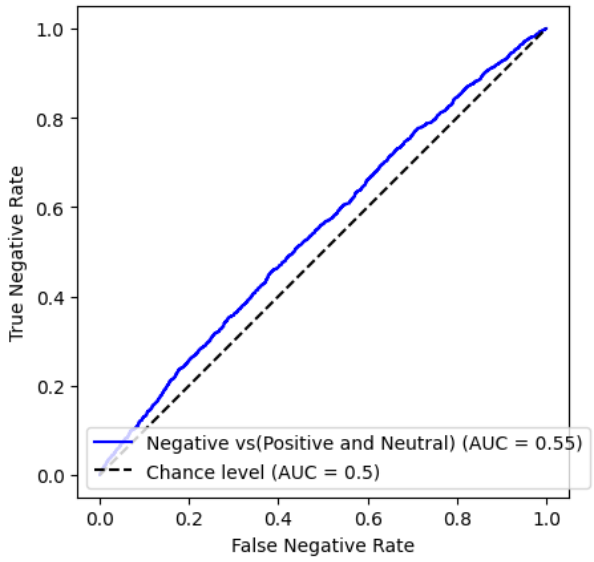

Figure 12: ROC Curve(Negative vs the rest)

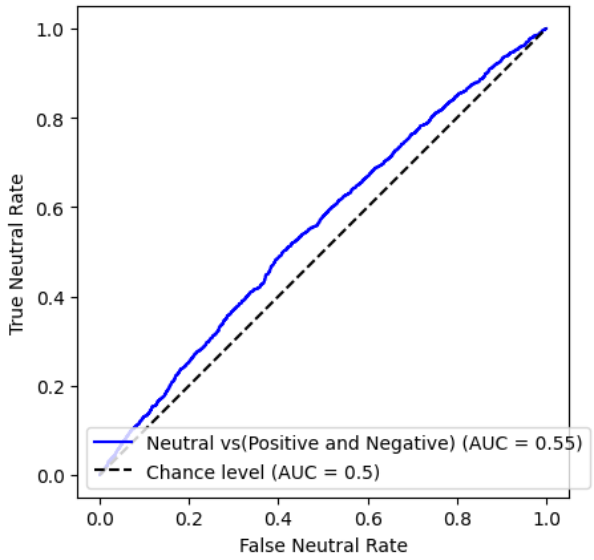

Figure 13: ROC Curve(Neutral vs the rest)

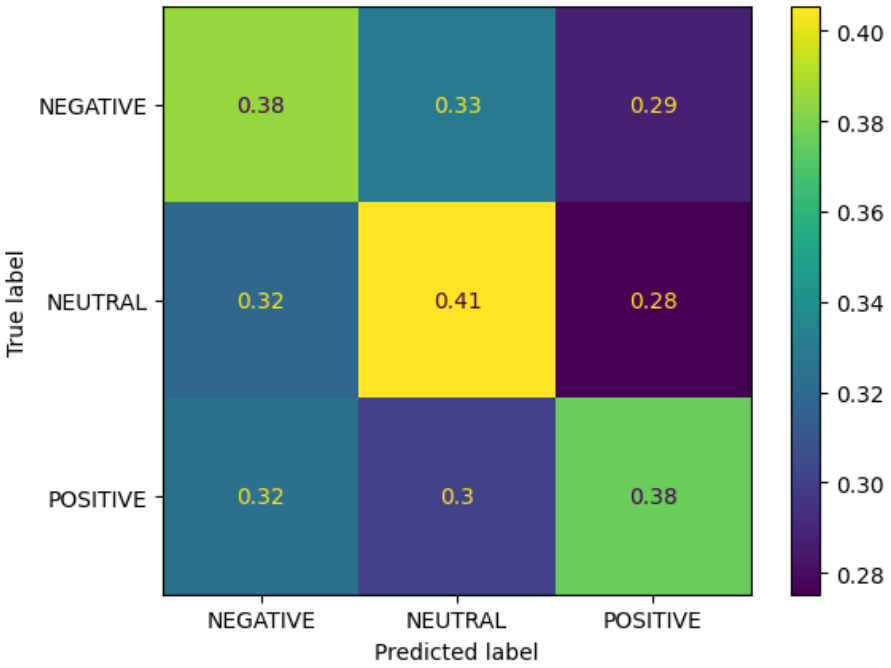

Figure 14: Confusion Matrix After Preprocess

3.1.3. DIMENSION REDUCTION. Dimension reduction is a technique used in machine learning to decrease the number of features in a data set. The goal is to simplify the data set, making it more manageable and often improving the performance of machine learning algorithms.As for us, the results from the usage of Dimension reduction are much better than before, because overfit was decreased significantly and our model corresponds similar to known and unknown data.Additionally, it is worth mentioning that i ran optuna optimization to find the number of components that best fits the data set.

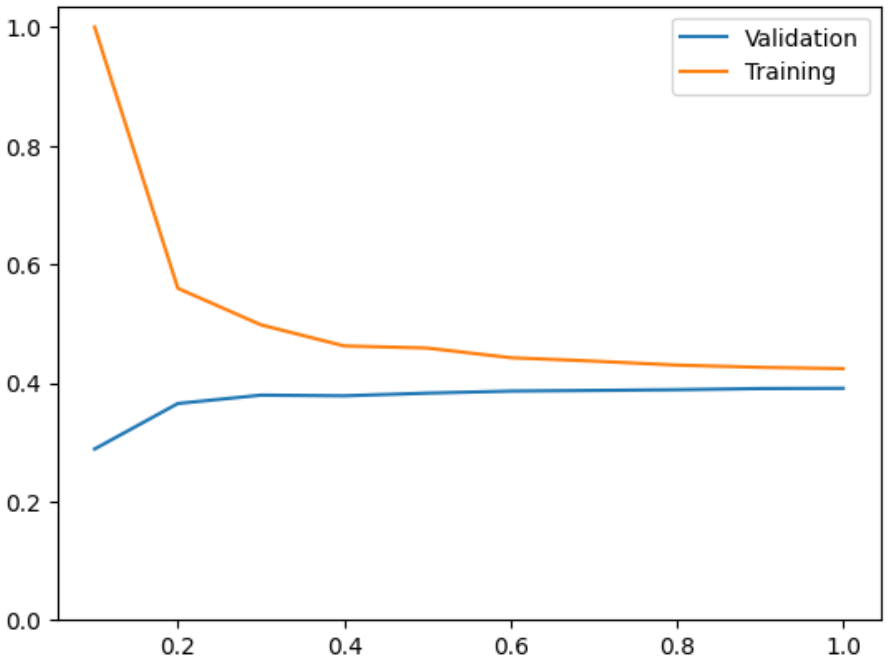

Figure 15: Learning Curve Before Pre-processing

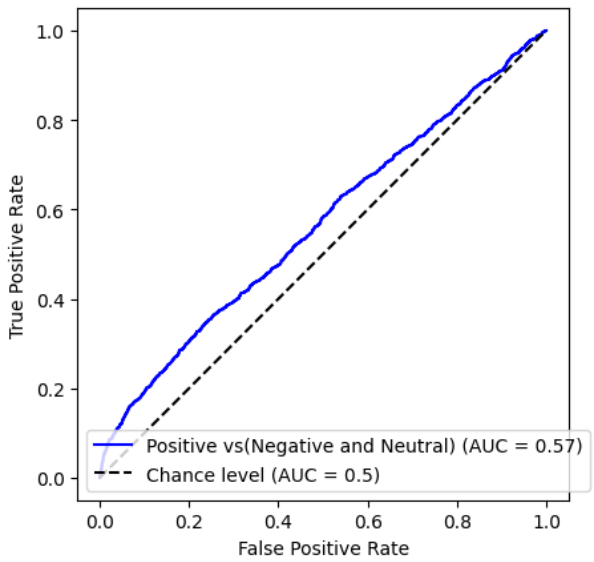

Figure 16: ROC Curve(Positive vs the rest)

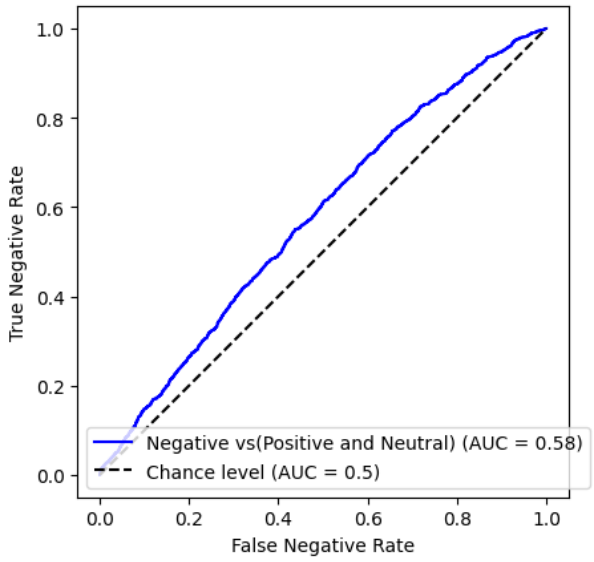

Figure 17: ROC Curve(Negative vs the rest)

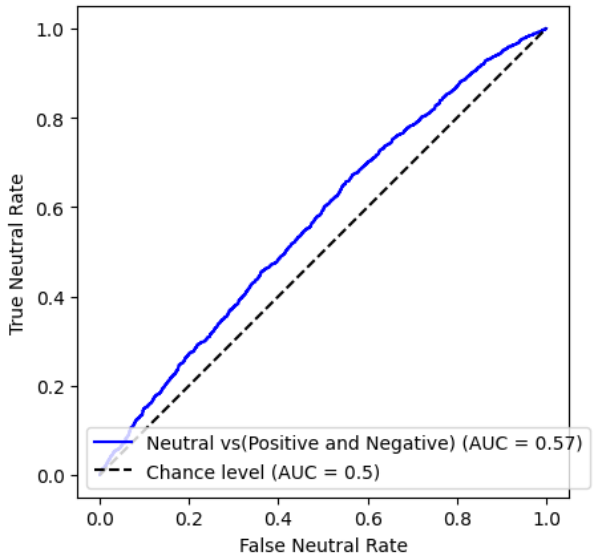

Figure 18: ROC Curve(Neutral vs the rest)

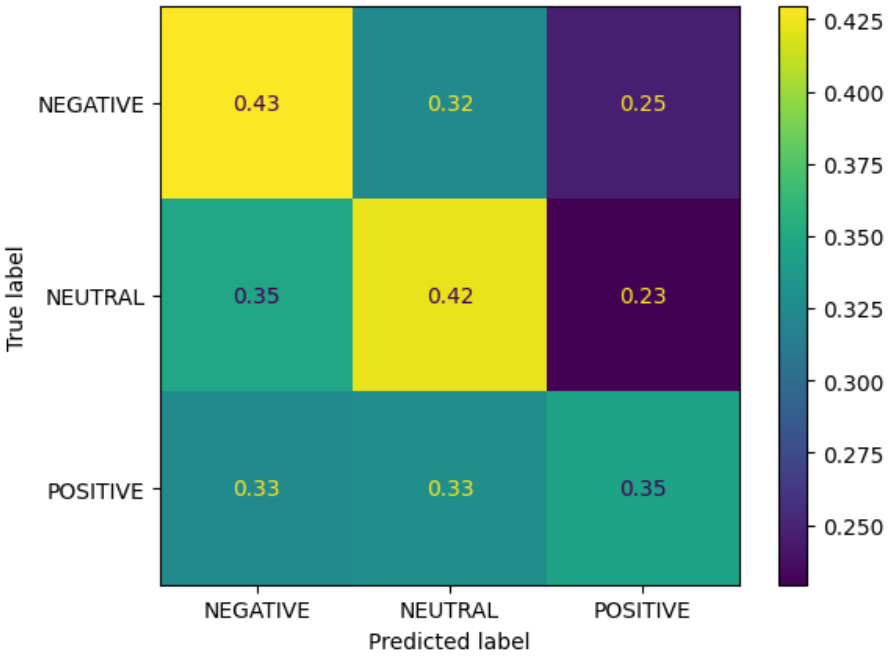

Figure 19: Confusion Matrix Using Dimension Reduction

3.1.4. K FOLD CROSS VALIDATION. The idea of k fold validation is simple. The data-set is divided into K subsets and the model is trained and evaluated K times, each time using a different fold as the validation set and the remaining folds as the training set and usually is helpful when we have a small set of training and validation data.Examining the results of its use in our case,we observe that the f1 training score is each time about 0.82,the f1 validation score is about 0.37 and the size of over fitting remains almost the same.This is because, by splitting the data set into multiple folds, k-fold cross-validation helps mitigate the impact of variability in the data. Each fold serves as both a training set and a validation set in different iterations, ensuring that the model is exposed to a variety of data patterns.

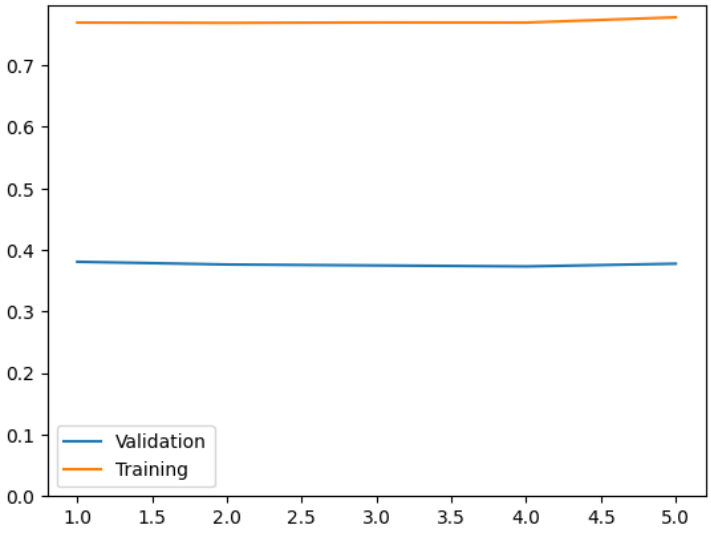

Figure 20: F1 scores for each fold

3.1.5. DATA REGULARIZATION. Regularization is a technique used in machine learning in order to reduce overfitting.In more detail,regularization adds a penalty term to the logistic regression cost function.There are two types of penalties:L1 and L2.L1 adds the absolute values of the coefficients as a penalty to the cost function, while L2 adds the squared values of the coefficients.For experimentation,I tried both methods.For L2 penalty, i used optuna optimization with the intention of finding the best value for the inverse of regularization strength(C) with the default solver algorithm 'lbfgs' and as for the results, the performances were quite similar with the experiment using the default C, but the training time was halfed.

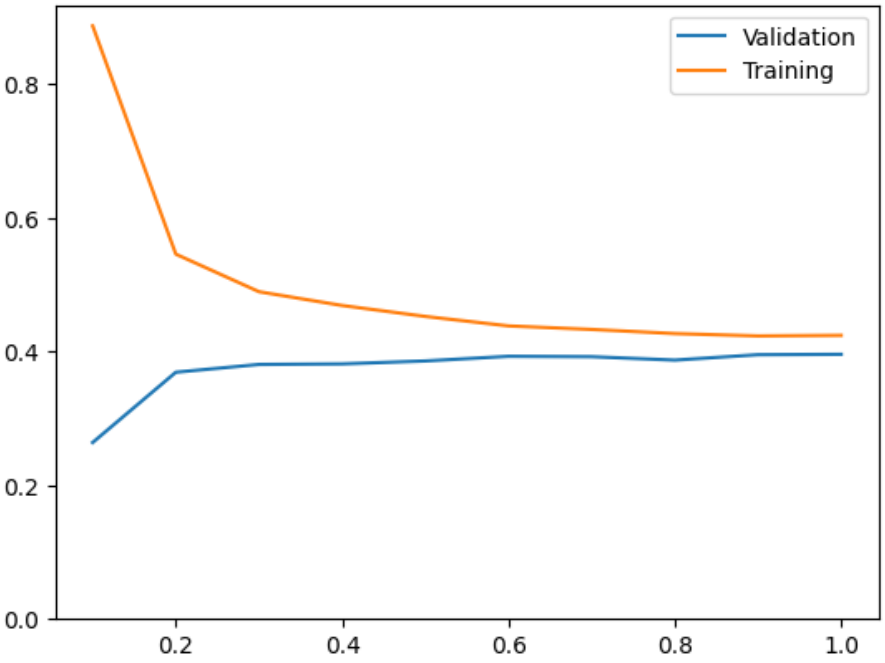

Figure 21: Learning Curve Using Dimension Reduction with Optimal C

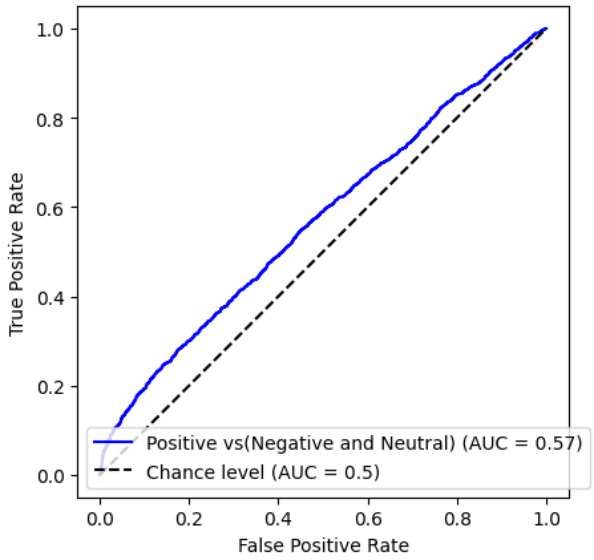

Figure 22: ROC Curve Using Dimension Reduction with Optimal C

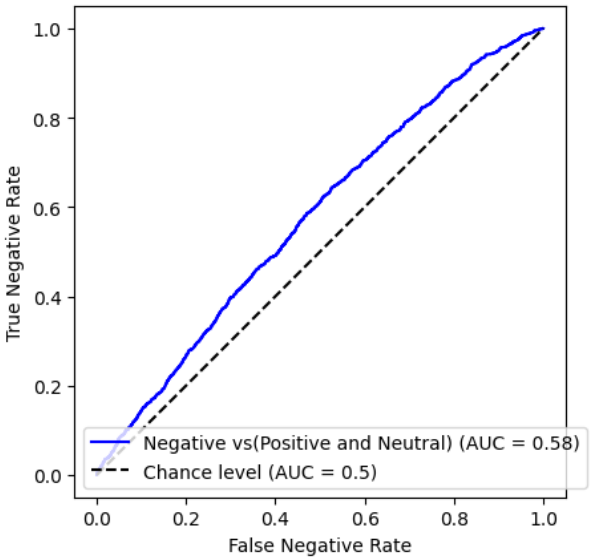

Figure 23: ROC Curve Using Dimension Reduction with Optimal C

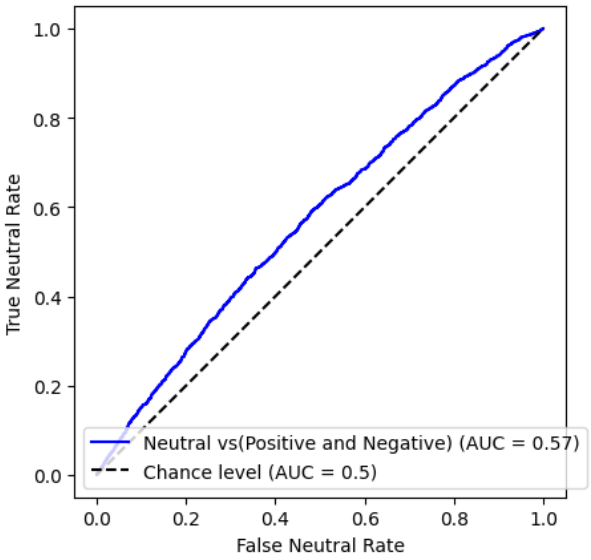

Figure 24: ROC Curve Using Dimension Reduction with Optimal C

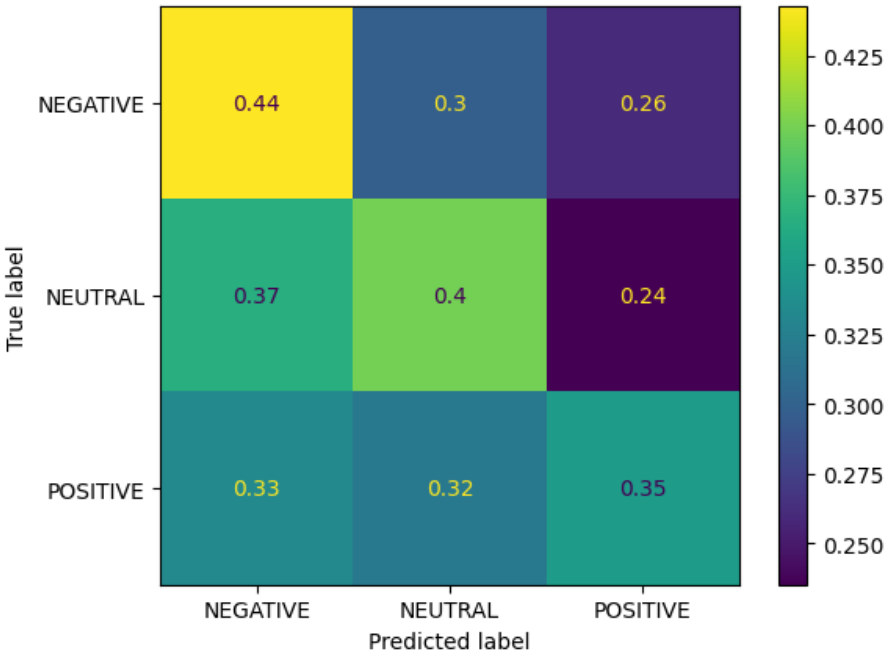

Figure 25: Confusion Matrix Using Dimension Reduction with Optimal C

Regarding to L1 penalty, I tried the combination (L1 penalty,saga solver,default C), which was quite slow.The difference between two experiments is that, with lbfgs solver the f1 training score starts from about 0.9 and is decreasing till around 0.4, while with saga solver the score starts from around 0.6.

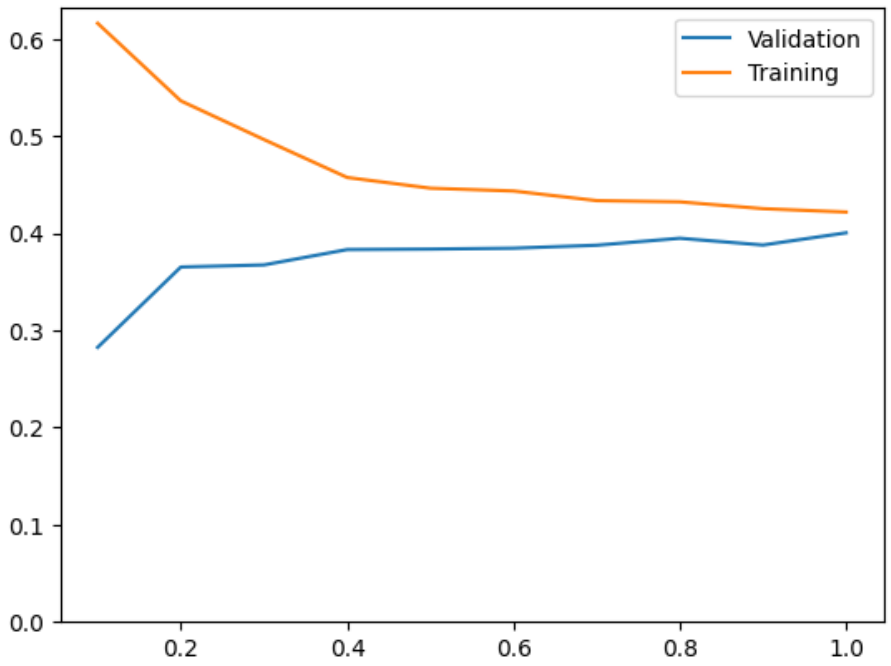

Figure 26: Learning Curve Using Dimension Reduction with L1 penalty and Saga Solver

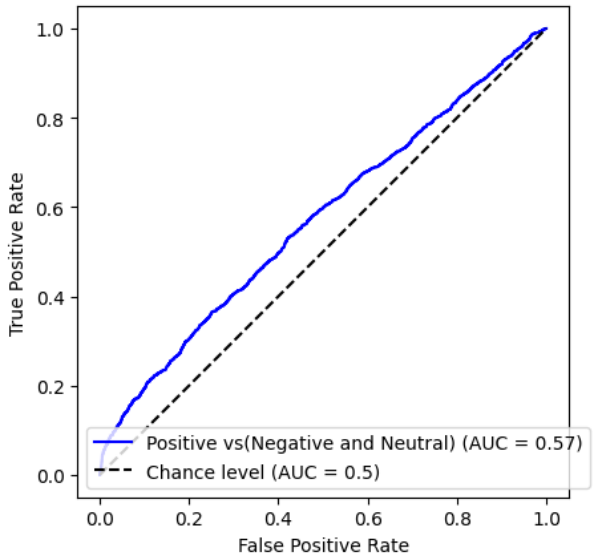

Figure 27: ROC Curve Using Dimension Reduction with L1 penalty and Saga Solver

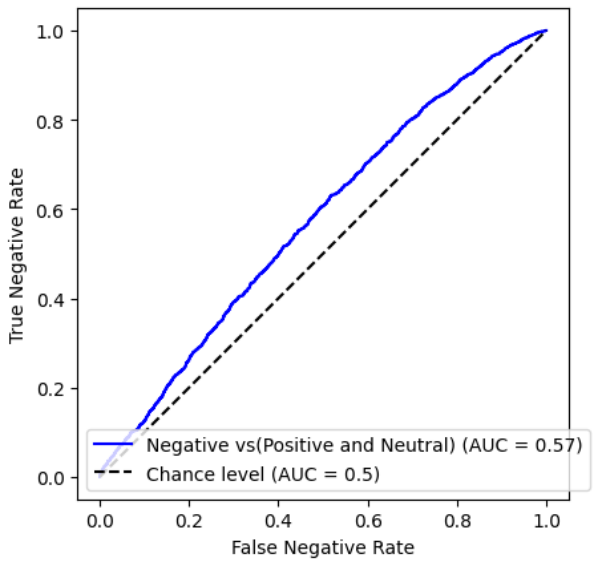

Figure 28: ROC Curve Using Dimension Reduction with L1 penalty and Saga Solver

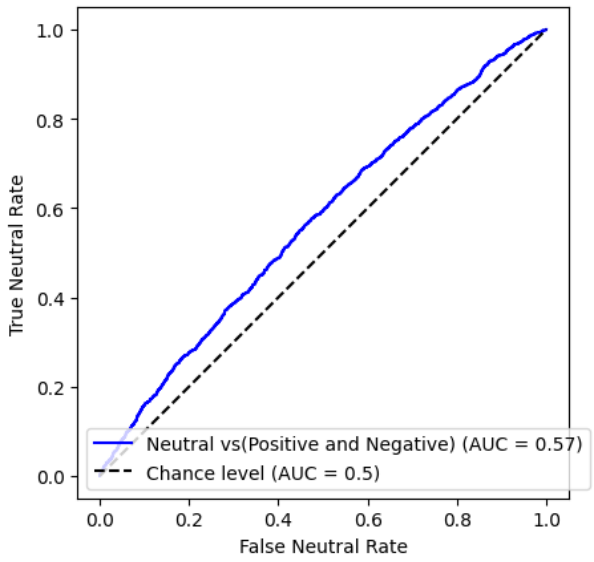

Figure 29: ROC Curve Using Dimension Reduction with L1 penalty and Saga Solver

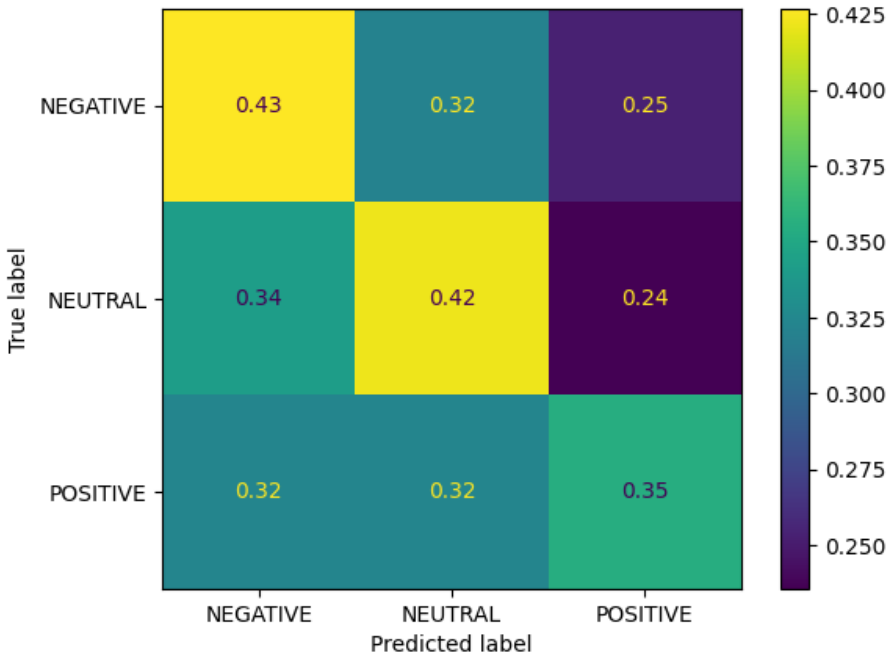

Figure 30: Confusion Matrix Using Dimension Reduction with L1 penalty and Saga Solver

3.2. Hyper-parameter tuning

<Describe the results and how you configured the model. What happens with under-over-fitting??> Hyperparameters are configurations that influence the behavior of the algorithm, the performance of the model and the size of over and underfitting.In other words, different hyperparameter values can lead to completely different classifiers.In our model, the principal hyperparameters I tried to tune was the number of components used in dimensional reduction and the hyperparameters used in Logistic Regression like the inverse of regularization strength,the solver and the penalty.For the hyperparameters C and number of components,I used optuna framework to find the optimal values.As for the penalties and solvers,I tried both L1 and L2 penalties with various solvers.The difference between L1 and L2 is that L1 can force less important coefficients to be exactly zero and it is useful when dealing with a large number of features,while L2 is desirable when we want to maintain all the features.As for solvers, I used the lbfgs solver,which is good for smooth optimazation problems and the saga,which is designed for optimization problems involving large datasets.

3.3. Optimization techniques

<Describe the optimization techniques you tried. Like optimization frameworks you used.>
For the exercise implementation i used Optuna framework.Optuna is a optimization framework used in machine learning, which tries to find optimal hyperparameters in order to improve the effictiveness of a model.As for me, I used this framework to find the best value for various hyperparameters (number of components in dimension reduction,Inverse of regularization strength in Logistic Regression etc.) and in this way i avoid all the experiments i should have try in order to find them.(Noting:If you want to run the Optuna code in my Notebook change the variable optimization to True.)

3.4. Evaluation

<How will you evaluate the predictions? Detail and explain the scores used (what's fscore?). Provide the results in a matrix/plots> <Provide and comment diagrams and curves>

For evaluating my model,I used F-score, ROC curves and Confusion matrices.

3.4.1. ROC curve. ROC curve is another evaluating measure,which is created by plotting the true positive rate opposed to the false positive rate at various thresholds,while AUC is the area under the ROC curve.In our case,AUC is around 0.54-0.58,which means that the classifier's accuracy is close to random guessing.

3.4.2. Learning Curve. F-score is a classification algorithm's performance metric, that combines precision and recall into a single measure and it ranges from 0 to 1, where a higher value indicates better performance.After the above experimentation, I end up that f1 validation score ranges from 0.38 to about 0.4,while the f1 training score is differentiated depending on the experiment,resulting smaller or bigger overfit.

Trial	Training Score	Val Score
Brute-Force run	0.98	0.38
Pre-processing	0.78	0.39
Dimension Reduction	0.42	0.40
Dimension Reduction and Data Regularization	0.42	0.39
with L2 penalty and lbfgs solver and optimal C		
Dimension Reduction and Data Regularization	0.42	0.40
with L1 penalty and saga solver		

Table 1: Trials

3.4.3. Confusion matrix. A confusion matrix represents the prediction's results in matrix form and it shows how many prediction are correct and incorrect on a validation or a test. set.Taking into consideration the confusion matrices above, i conclude that around 38 to 40 percent of the predictions are correct.

Noting: The plots for Learning Curves,ROC Curves and Confusion Matrices are listed above.

4. Results and Overall Analysis

4.1. Results Analysis

<Comment your results so far. Is this a good/bad performance? What was expected? Could you do more experiments? And if yes what would you try?> <Provide and comment diagrams and curves>

In conclusion, the best accuracy the logistic regression model can reach is about 0.4.This is because, a significant number of Sentiments in the dataset is wrong.However, as i said above, the goal of the Project was to experiment with various techniques.Obviously, there are many techniques and experiments, I don't present in the Notebook.For example, I could run more test,using the Elasticnet penalty and other solvers (liblinear,Newton-cg) or I could use more techniques like batch size configuration.

4.1.1. Best trial. Given all the experiments, I end up that the best trial was the one that combines the Dimension Reduction method with lbfgs solver,L2 penalty and op-

timal C(occured by Optuna), because it creates the smallest overfit at the best time. <Showcase best trial>

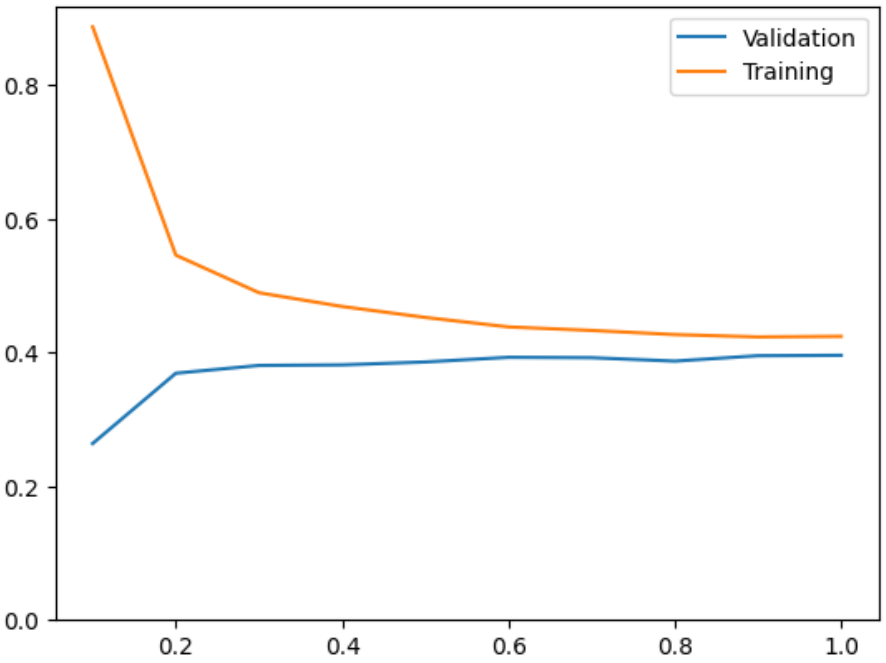

Figure 31: Learning Curve Using Dimension Reduction with Optimal C

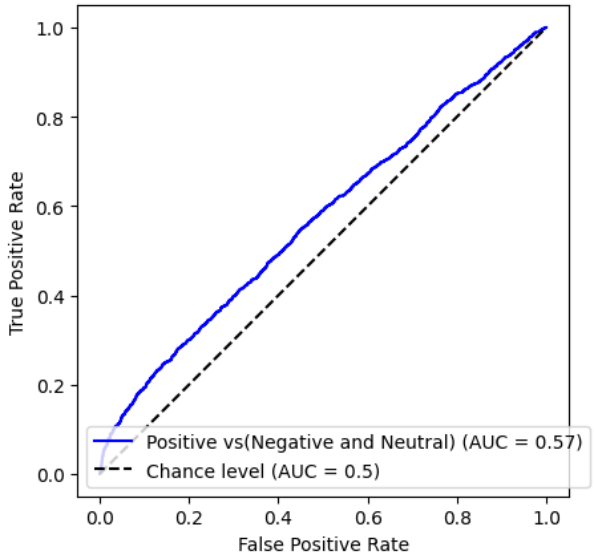

Figure 32: ROC Curve Using Dimension Reduction with Optimal C

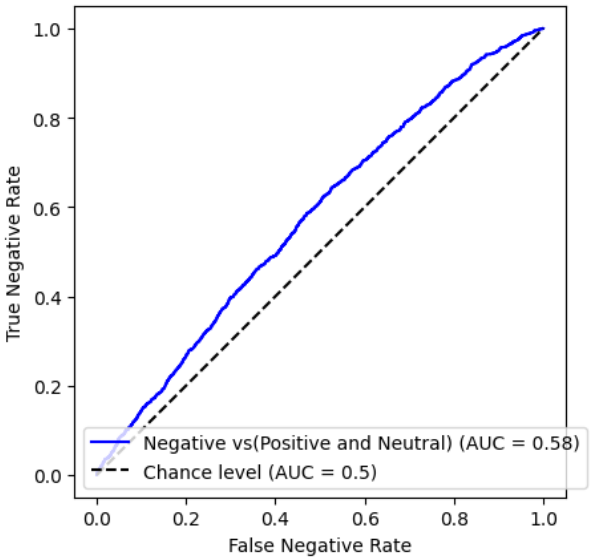

Figure 33: ROC Curve Using Dimension Reduction with Optimal C

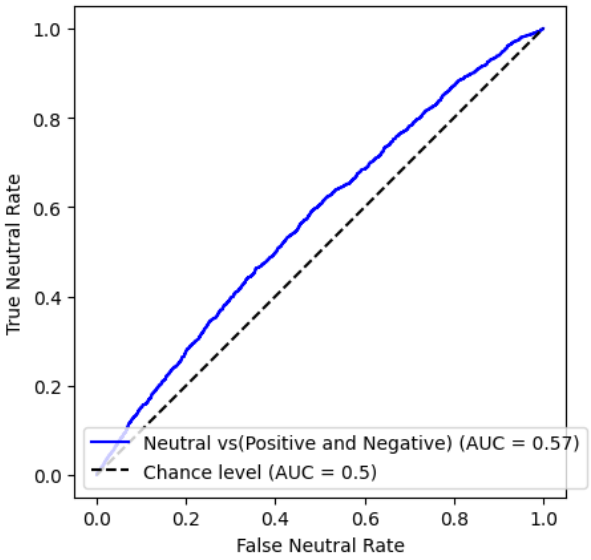

Figure 34: ROC Curve Using Dimension Reduction with Optimal C

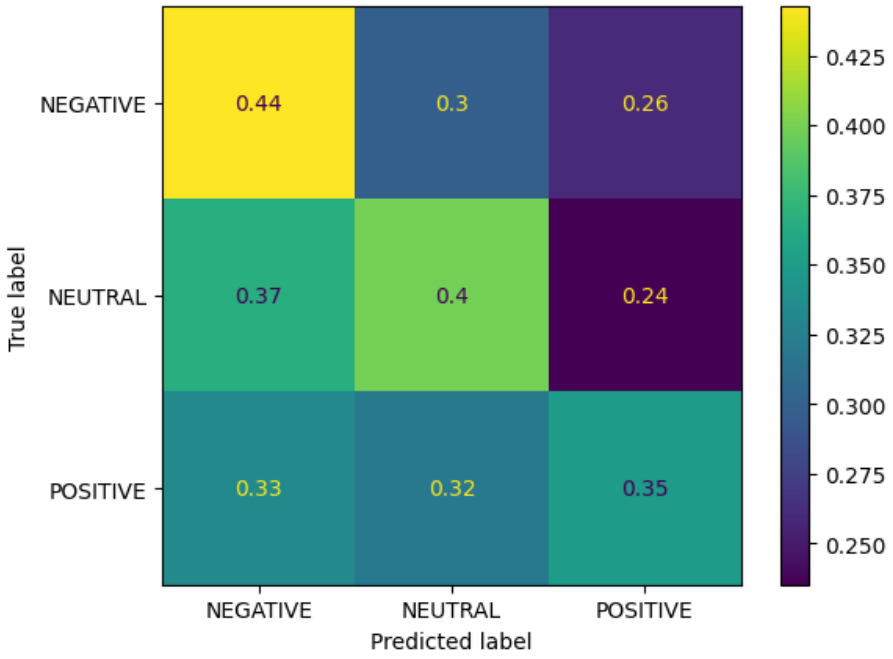

Figure 35: Confusion Matrix Using Dimension Reduction with Optimal C

4.2. Comparison with the first project

<Use only for projects 2,3,4>
<Comment the results. Why the results are better/worse/the same?>

4.3. Comparison with the second project

<Use only for projects 3,4>
<Comment the results. Why the results are better/worse/the same?>

4.4. Comparison with the third project

<Use only for project 4>
<Comment the results. Why the results are better/worse/the same?>

5. Bibliography

References

[1] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984.

<You should link and cite whatever you used or "got inspired" from the web. Use the / cite command and add the paper/website/tutorial in refs.bib>
<Example of citing a source is like this:> [1] <More about bibtex>

REFERENCES

- Sklearn Documentation: <https://scikit-learn.org/stable/>

- Emoji Removal: <https://gist.github.com/slowkow/7a7f61f495e3dbb7e3d767f97bd7304b>
- Beginners Guide To Truncated SVD For Dimensionality Reduction: <https://analyticsindiama
  guide-to-truncated-svd-for-dimensionality-reduction/>