# **Project NLP | Business Case: Automated Customer**

Group3:

-Edward Cruz

-Alex Castillo

-Joshua Lopez

-Gerardo Gonzalez

## **Executive Summary**

This business case outlines the development of an NLP model to automate the processing of customer feedback for a retail company. The goal is to classify customer reviews into positive, negative, or neutral categories to help the company improve its products and services. The second part is to GenerativeAI to summarize reviews broken down into review score (0-5), and broken down into product categories - if the categories are too many to handle, select a top-K categories. Create a clickable and dynamic visualization dashboard using a tool like Tableau, Plotly, or any of your choice.

## **Project goals**

* The ML/AI system should be able to run classification of customers' reviews (the textual content of the reviews) into positive, neutral, or negative.
* For a product category, create a summary of all reviews broken down by each star or rating (we should have 5 of these).
  + If your system can't handle all products categories, pick a number that you can work with (eg top 10, top 50, Etc)

## 

## **1)Traditional NLP & ML approaches**

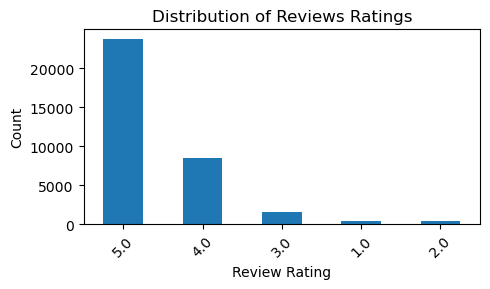
## Dateset used: 1429\_1.csv

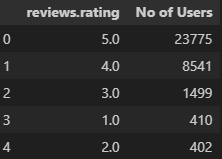
## From: https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products

## Data Pre-processing:

* We analyzed the dataset to determine which columns were really useful for the case.
* Check features that had null values and clean them.
* After removing columns tat had most null values. By checking the distributions of the other features we determined that for the sentiment classifier, the most important feature are “reviews.text” (model input features) and “reviews.rating” (model target) .

**Noted:** From the analysis we found that the dataset is very unbalance.

Figure 1: Original Data Ratings Distribution



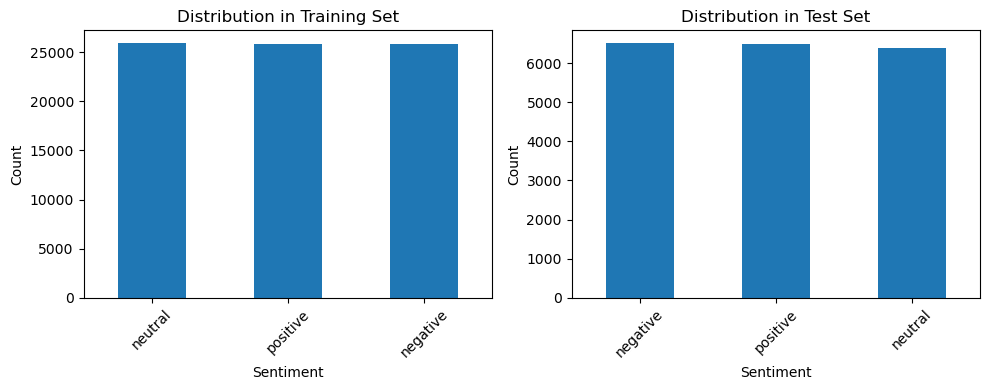
(b) Texts Tokenize and Remove stopwords

* 1. First split the dataset into features and target
  2. To the input feature column (review.tex):
* Clean stop words (NLTK)
* Convert to lowercase
* Remove punctuations
  1. To the target column (reviews.rating):
* creates a function to map rating to sentiments.
* Positive (ratings 4-5)
* Negative (ratings 1-2)
* Neutral (ratings 3)

(c) Vectorize text data using TF-IDF and Balancing

-Vectorize the clean review.text.

- I apply SMOTE on the vectorize data to balance the dataset.

Figure 2: Distribution after grouping and Balancing

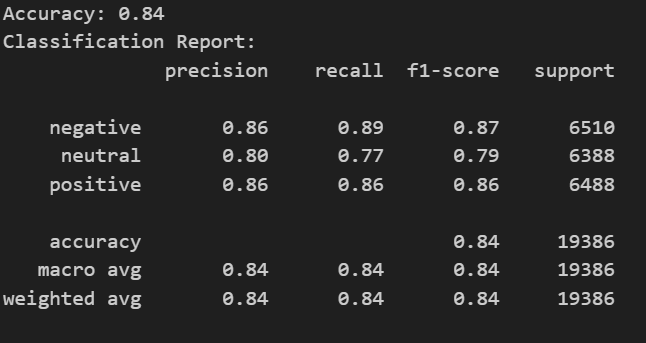
(d) Split data into Training and Test

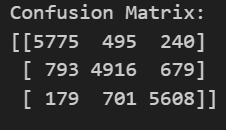
-Train 80%

-Test 20%

(e)Train using Different Traditional NLP and ML models

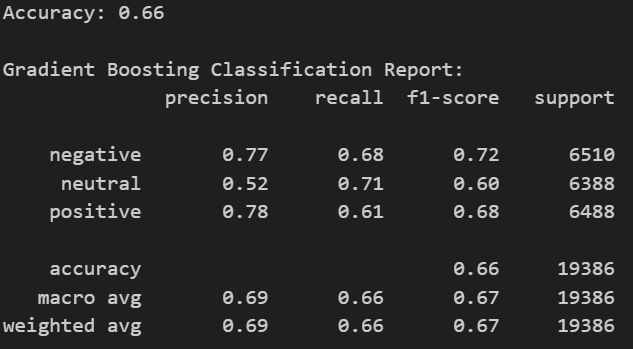
1. Naive Bayes:

Figure 3: Naive Bayes Classification Report

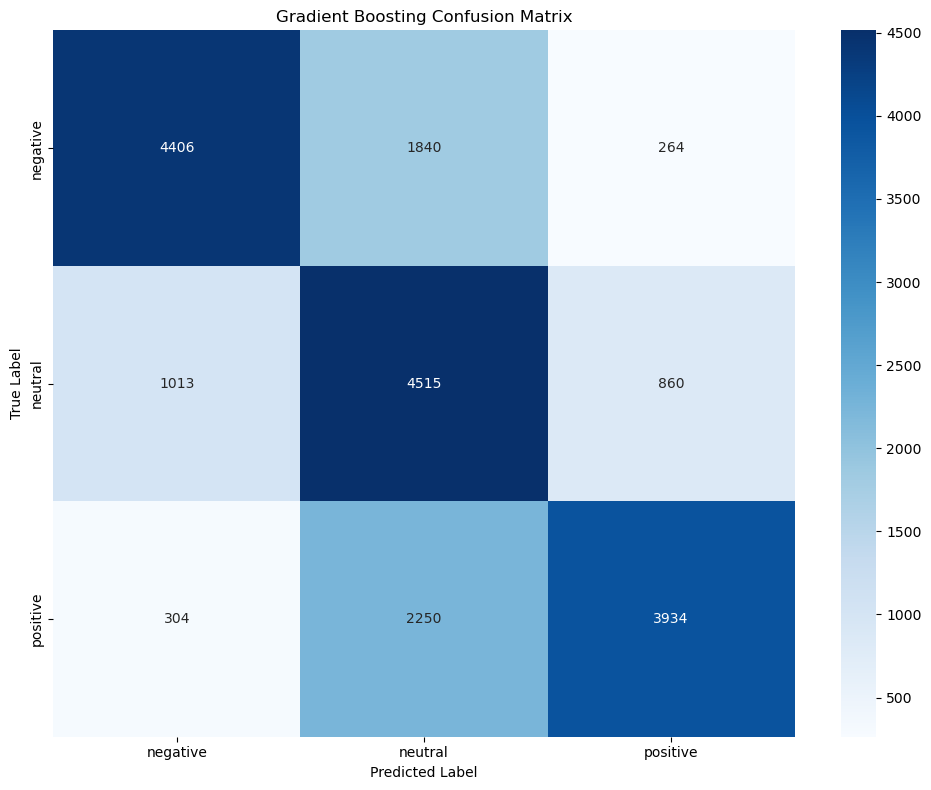
Figure 4: Confusion Matrix

2)Gradient Boosting

Report:

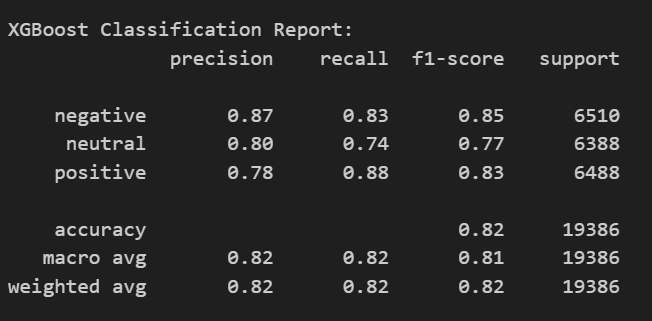
Figure 5: Evaluations Classification Report Scores

Confusion Matrix:

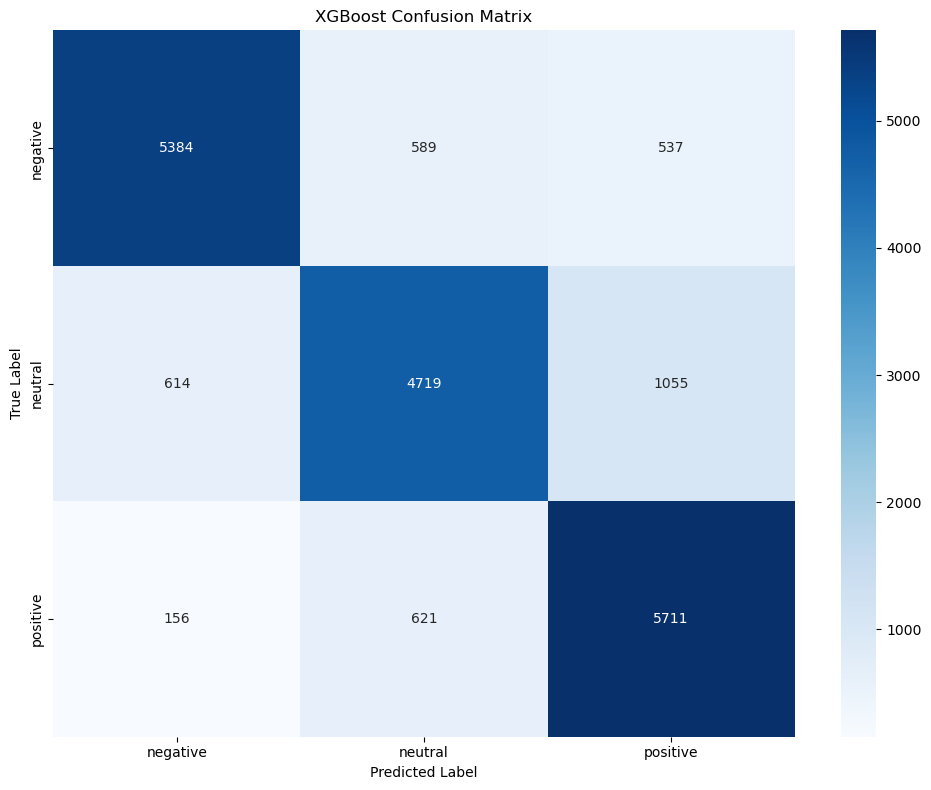
Figure 6: Confusion Matrix

3)XGBoost

Report:

Figure 7: Classification Report Scores

Confusion Matrix:

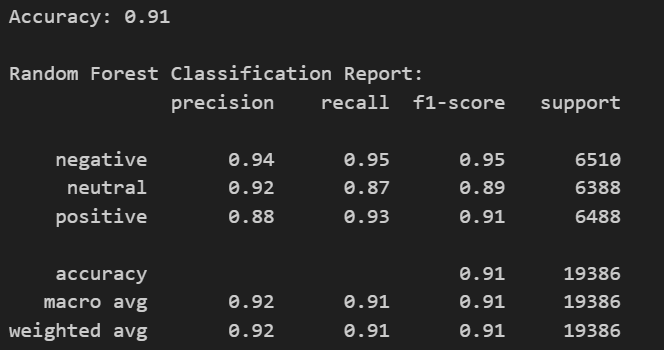
Figure 8: Confision Matrix

4)Random Forest L2 Regularize

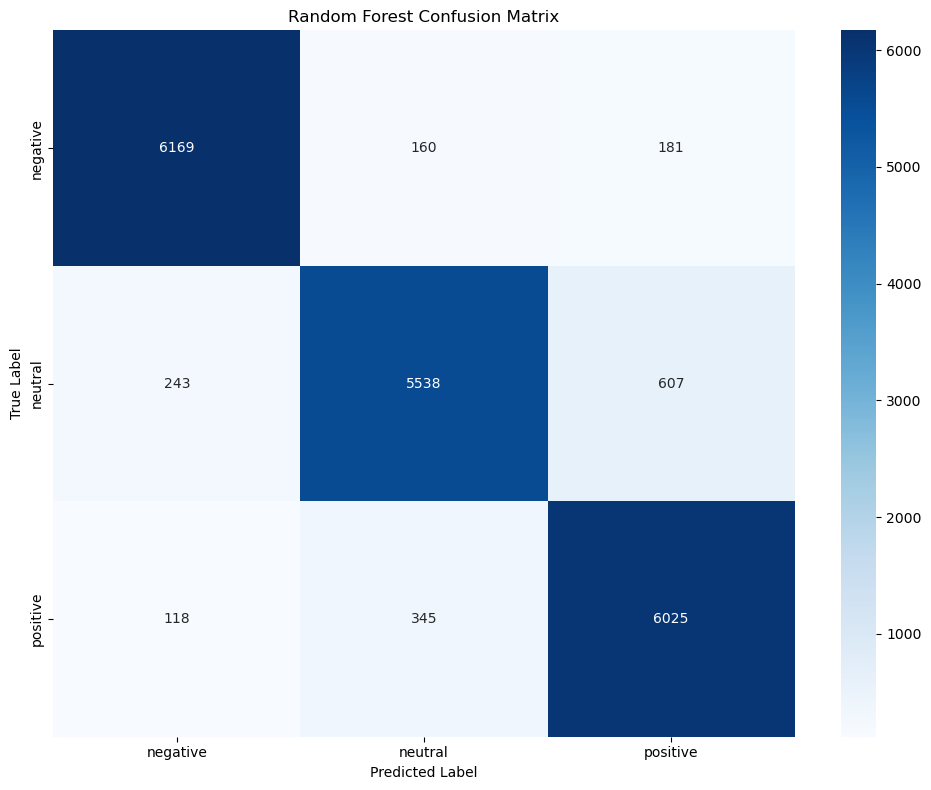
Parameter:



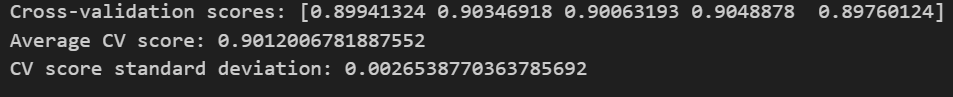
Report:

Figure 9: Evaluation Classification Report Scores

Confusion Matrix:

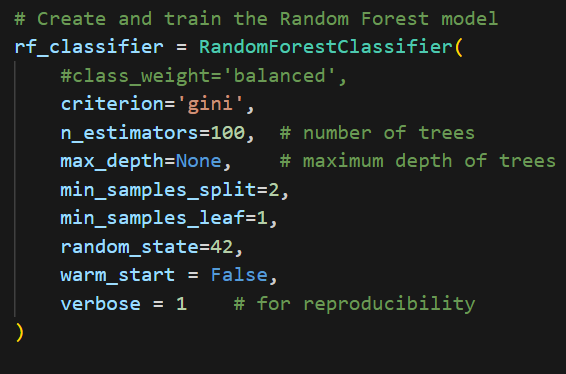
Figure 10: Confusion Matrix

Cross-Validation:

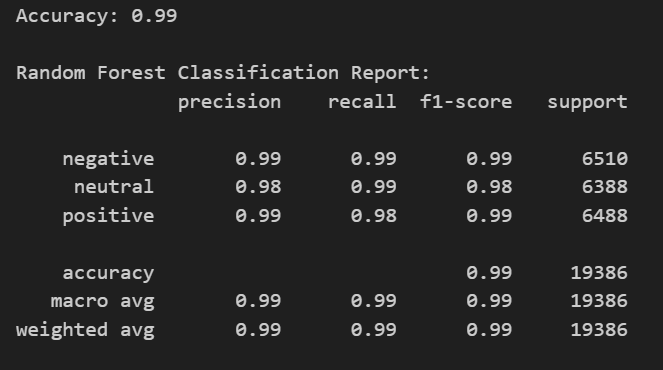
Figure 11: Cross - Validation

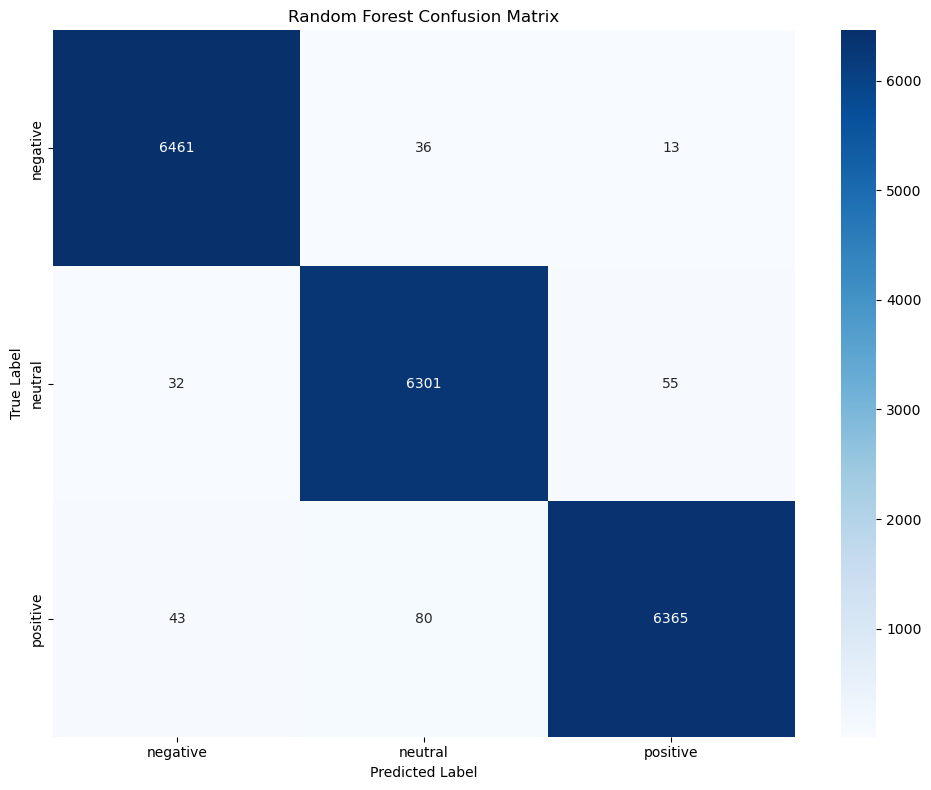
5) Random Forest (BEST)

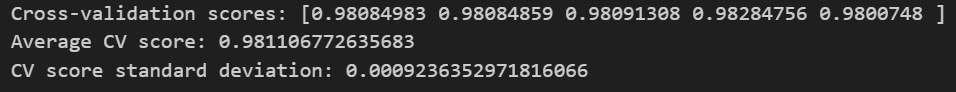
Parameter:



Report:

Figure 12: Random Forest Classification Report Evaluation

Figure 13: Confusion Matrix

Figure 14: Cross Validation Analysis

**Analysis:**

#These cross-validation results:

1. Consistency across folds:

- The scores across all 5 folds are very consistent (ranging from 0.980 to 0.982)

- The standard deviation is very small (0.00092), which is excellent

- This consistency suggests that your model is stable and performs similarly across different subsets of the data

2. Average CV Score:

- Average score of 0.981 (98.1%) is slightly lower than your test set accuracy of 0.99

- This small difference (about 0.9%) between CV and test performance suggests that your model isn't severely overfitting

- It's performing consistently well across different data splits

3. Analysis:

- The high scores with low variance suggest your model is genuinely learning the patterns in your data

- The small gap between CV scores and test accuracy indicates minimal overfitting

- The extremely low standard deviation (0.00092) shows remarkable stability in model performance

# **Conclusion:**

Based on these cross-validation results, I would say that the model is NOT overfitting. The reasons are:

1. Very stable performance across different data splits

2. Small difference between CV and test performance

3. Extremely low standard deviation in CV scores

This Random Forest model appears to be genuinely good at this classification task, likely because:

- The relationship between review text and sentiment might be relatively straightforward

- You probably have a good feature representation

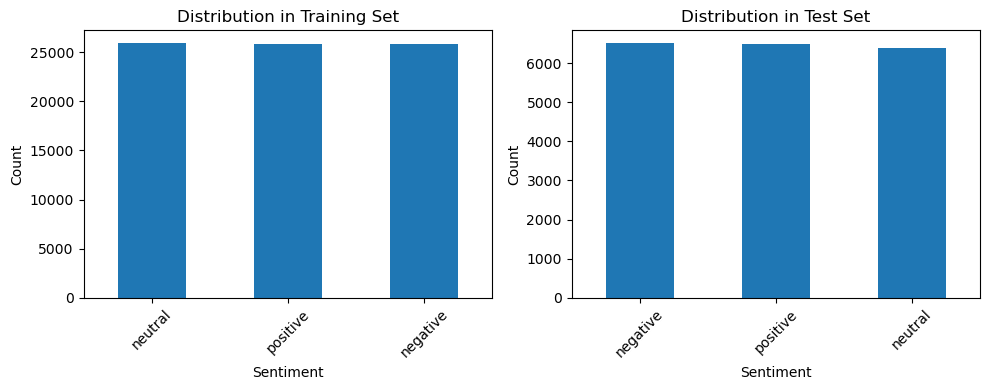
- The classes are well-balanced after your preprocessing

2) **Sequence-to-Sequence modeling with LSTM**

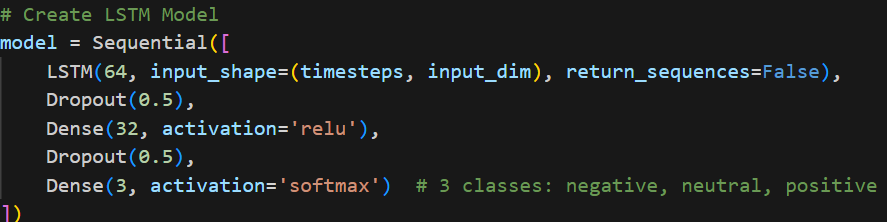
* Goal: Build a Biderectional LSTM model to predict the review class i.e., negative, positive, or neutral.

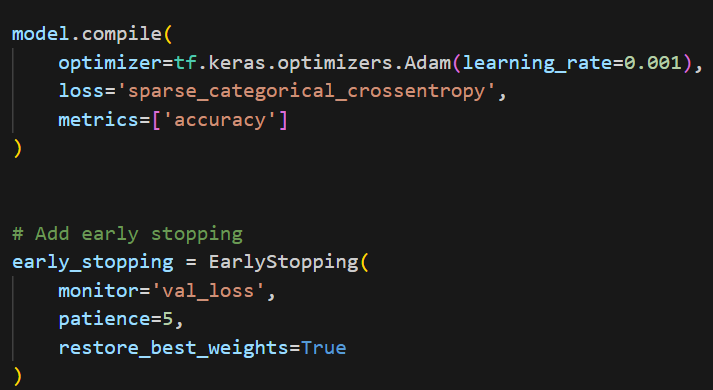
1. I did the same data preprocessing as for the traditional NPL models:

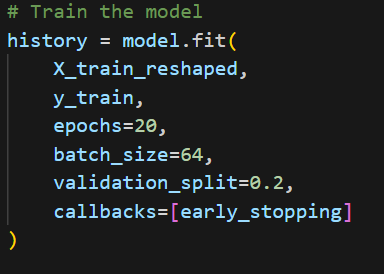
* Deleted unnecessary columns
* Remove nulls
* Remove stop-word, punctuation.
* Convert to lowercase
* Word tokenization
* Vectorize the data
* Balance the data with SMOTE
* Split data into training and validation datasets

Figure 15: Distribution after Balancing Data

1. Train LSTM MODEL

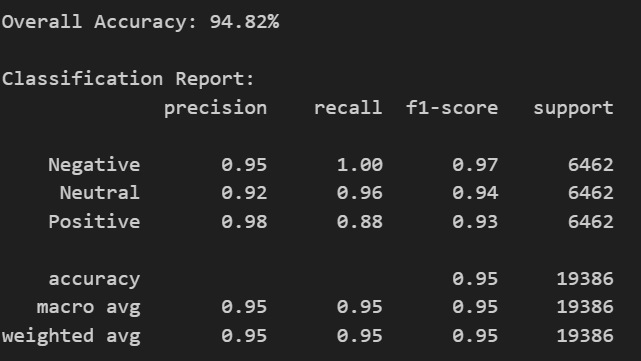
* Reshape input data for LSTM (samples, timesteps, features)
* Architecture: 



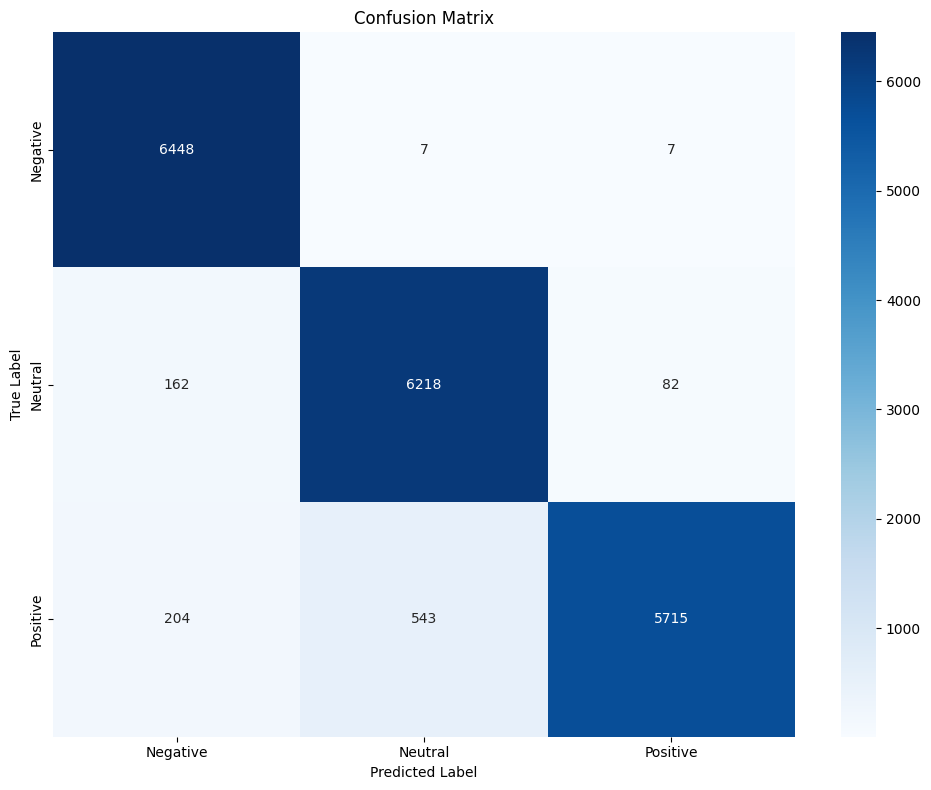


1. Evaluation

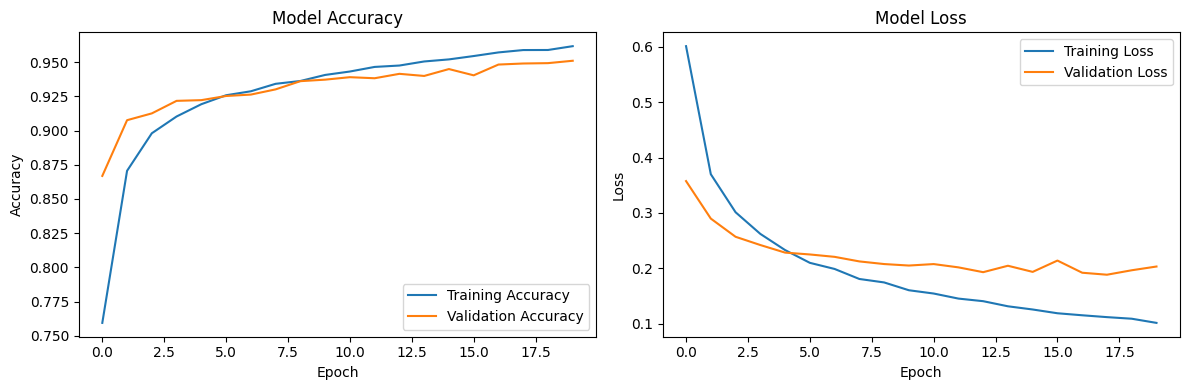
Report:

Figure 16: Evaluation Report

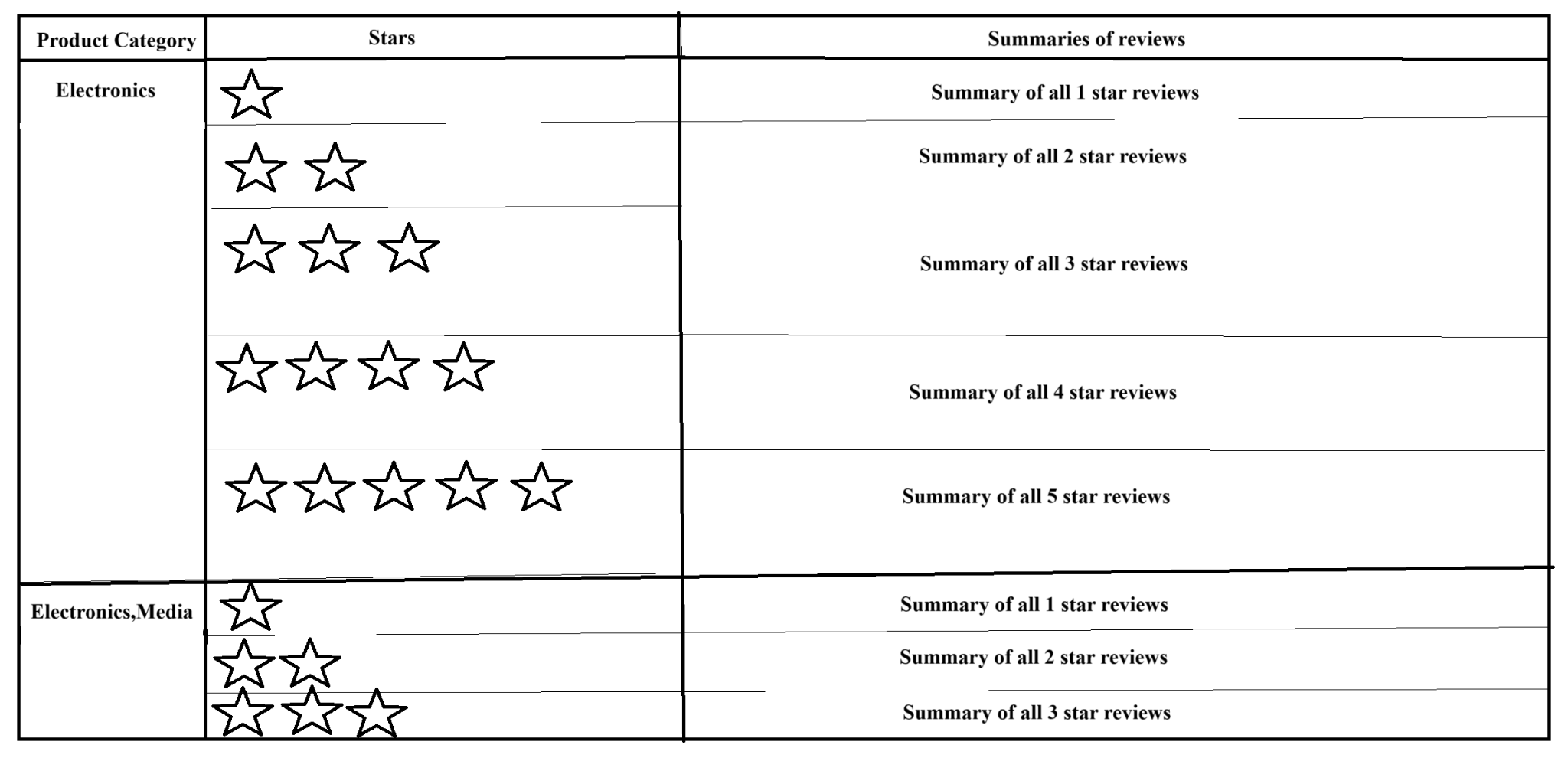
Confusion Matrix:

Figure 17: Confusion Matrix

Accuracy and Loss plots:

Figure 18: Loss and Accuracy Plots

**3)** **Transformer approach (HuggingFace API) for Summarys**

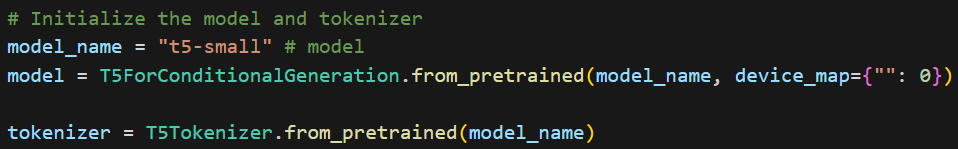
The goal was to have **a summary of all reviews broken down by each star or rating, for all product categories**. It would look like this, where every product had, for each rating, a single summarization of all reviews combined of that rating. 

The dataset used: **Datafiniti\_Amazon\_Consumer\_Reviews\_of\_Amazon\_Products.csv**

from [https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products](https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products/data?select=Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv)



The model used: **t5-small** from <https://huggingface.co/google-t5/t5-small>



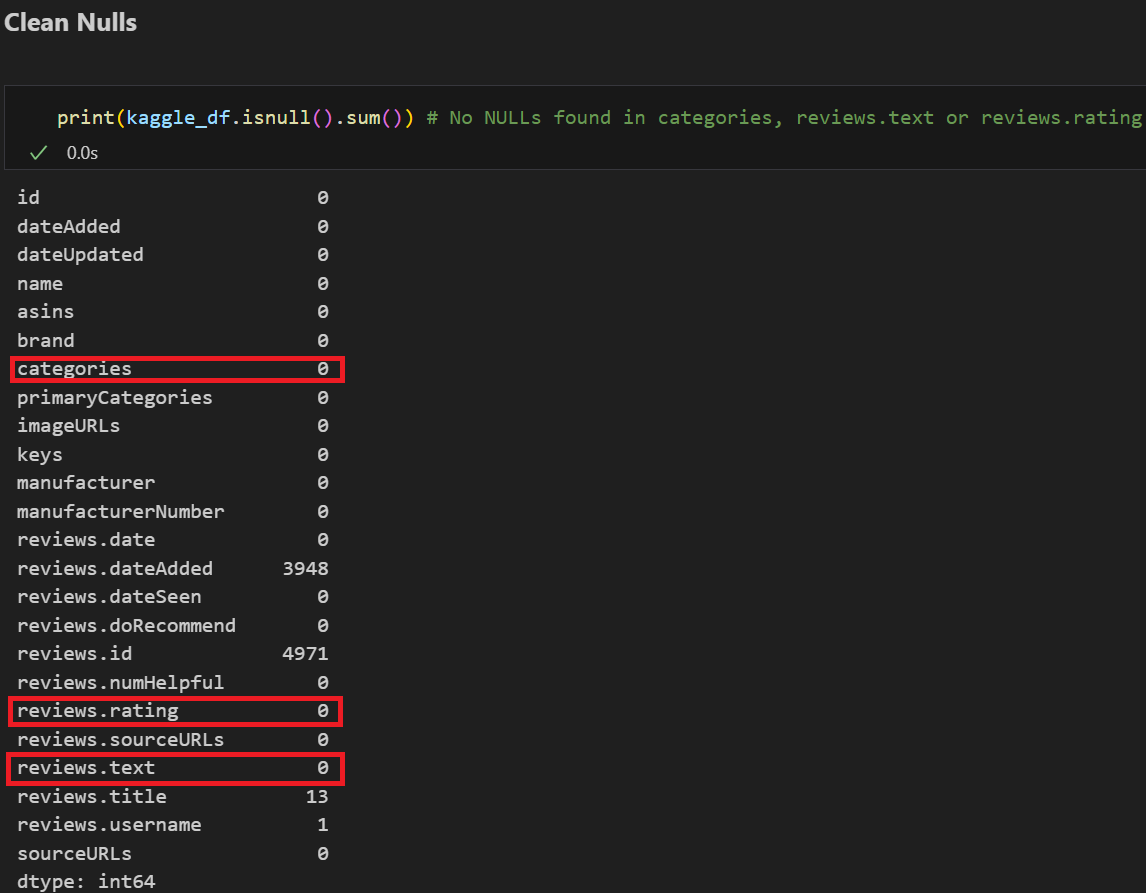
**1) Foundation Model**

**a)Data Preprocessing**

**1) Data Cleaning and Tokenization**

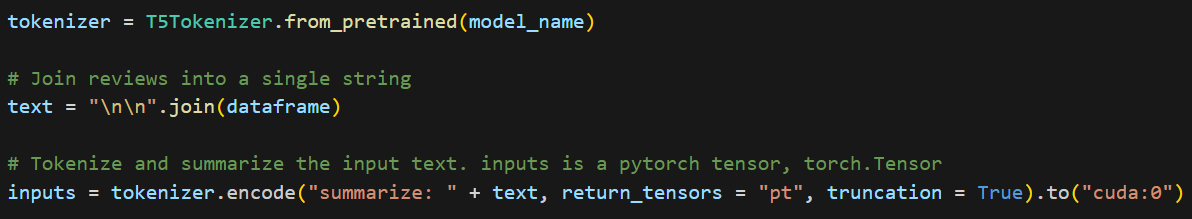
**-Cleaning NULLS**

Since the columns that were going to be used were **“categories”, “review.rating”** and **“review.text”**, and they didn’t have **NULL** values, eliminating the **NULL** fields was not needed.

Figure 19: Dataset information

**b)Tokenization**

For the tokenization the T5 library provided an instruction to encode the data to be summarized.

Figure 20: Tokenizer

**c) Metrics**

Summarization models use **ROUGE metrics** instead of accuracy scores to verify how good the model is. The ROUGE scores used were **rouge1, rouge2, rougeL** and **rougeLsum.**  It is preferred to use **rougeL** since it uses the **least common sequence** to produce its score. Since its **rougeL** score is 0.087, it is not a good model to use.

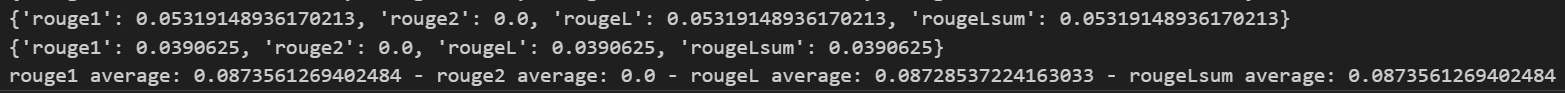
The **scores** for the **pre-trained** model are:

**rouge1 average:** 0.0873561269402484

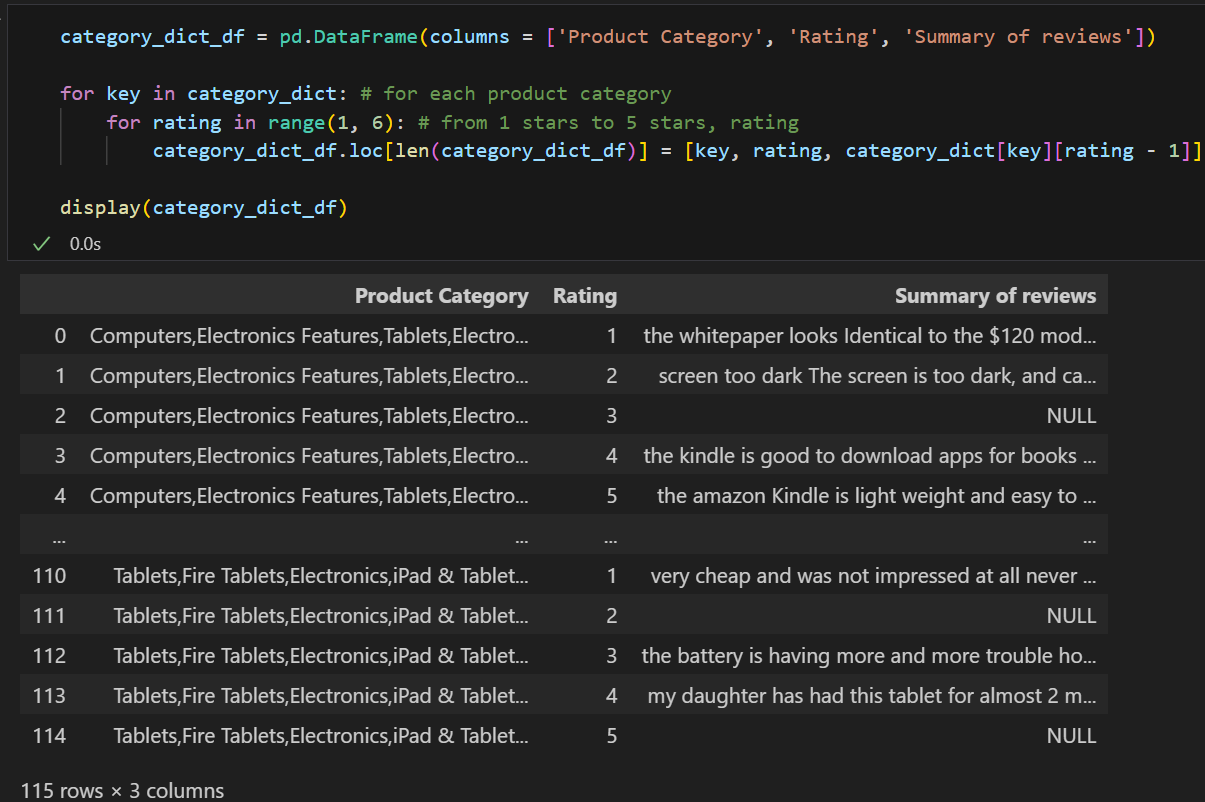
**rouge2 average:** 0.0

**rougeL average:** 0.08728537224163033

**rougeLsum average:** 0.0873561269402484

Figure 21: Rouge Metric Results

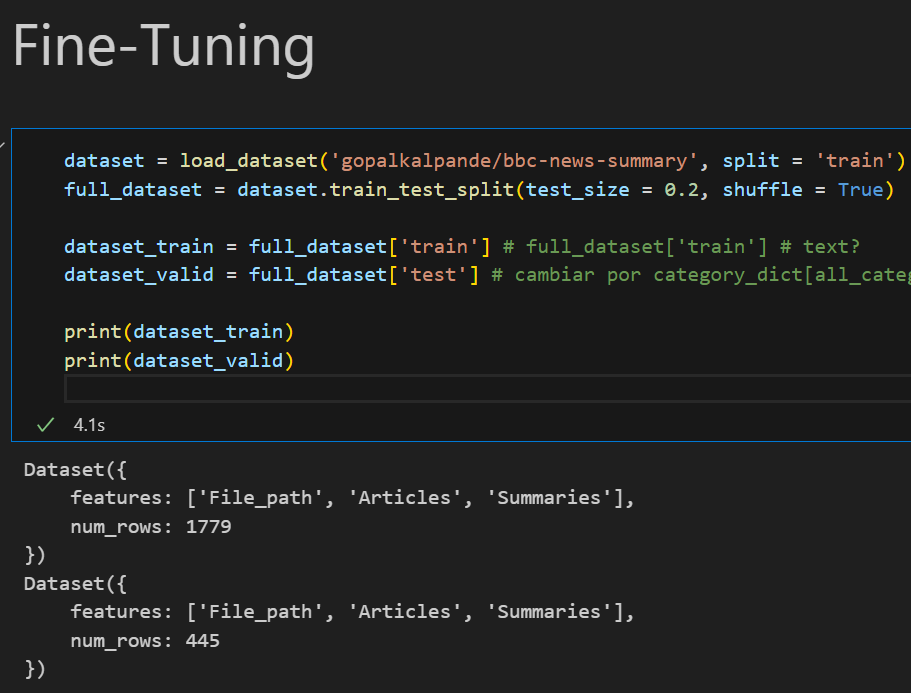
**d) End Result:**

Figure 22: Results Example

**2) Fine Tuning Model**

The dataset used: **'gopalkalpande/bbc-news-summary'**

from **Dataset library**

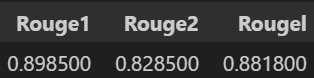


The model used: **t5-small** from <https://huggingface.co/google-t5/t5-small>

**a) Tokenization**

For the tokenization the T5 library provided an instruction to encode the data to be summarized.

**b) Metrics**

Figure 23: Rouge Results (Fine tune Model)

The **scores** for the **fine-tuned** model were:

**rouge1:** 0.0898500

**rouge2:** 0.828500

**rougeL:** 0.881800

It had **10x** a better **rougeL score** than the previous model. So this model would be a whole lot more usable for text summarizaton.

**4) Transformer approach (HuggingFace API) for Sentiment**

**1) Foundation model (No Fine tunning)**

-For this case, we use the BERT model. This model has already be pre-trained to classify products reviews into ratings.

**-Dataset:** 1429\_1.csv

from [https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products](https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products/data?select=Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv)

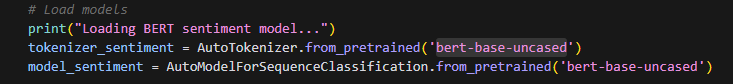
**-Foundation Model**: **bert-base-uncased**

**a) Data Preporcessing:**

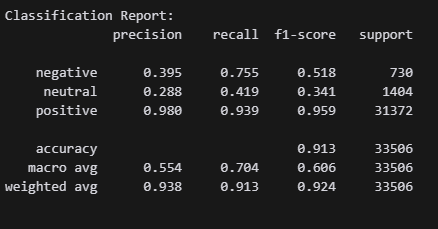
* Clean nulls
* Drop unnecessary columns
* Remove short and super long reviews.
* Group together rating and Map them to Sentiment: Rating 1-2 = Negative,

Rating 3 = neutral and rating 4-5 = positive.

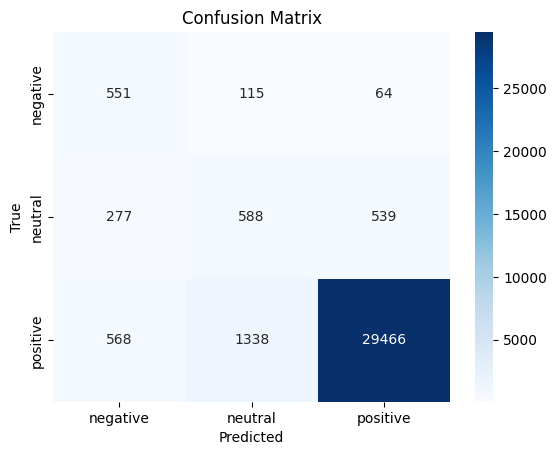
**b) Load Model and Tokenization**

Figure 24: Model and Tokenizer

**c) Classification Report:**

Figure 25: Evaluation Results

Confusion Matrix:

Figure 26: Confusion Matrix

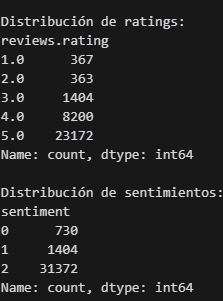
**2) Fine tuned Model (BERT)**

-We decided to do full fine-tune after not improving the result with the LoRA Configuration. Since our dataset has 33,506 reviews and we have the computational resources.

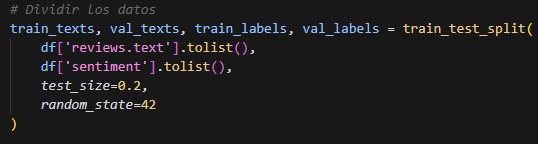
**a) Data Preporcessing:**

-Similar data preprocessing as in the Foundation model.

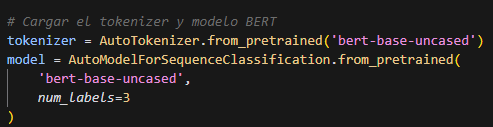
How data distribution looks before grouping them into sentiment:

Figure 27: Data Distribution

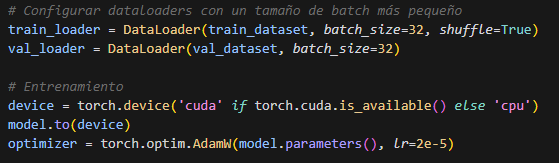
**b) Split data into Training and Validation**



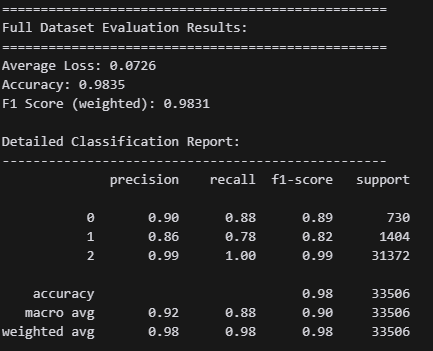
**c) Load Model and Tokenizer:**

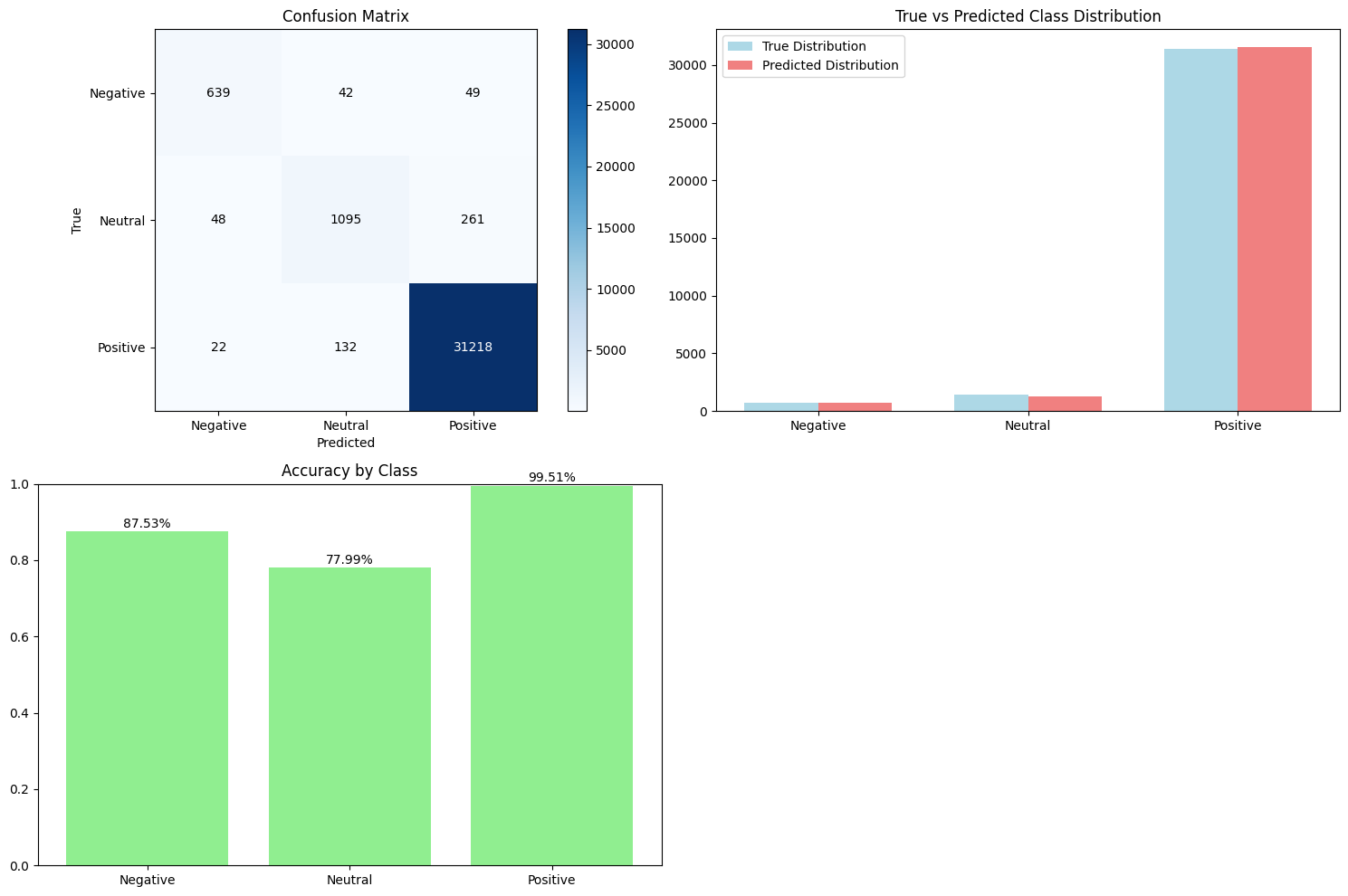
Figure 28: Tokenizer and Model

**d) Configure dataloader and Train model:**



**e) Evaluation using the full dataset:**

Figure 29: Evaluation Results

Figure 30: Evaluation Plots

BERT Sentiment Analysis Model Performance Analysis

## Overall Performance

- Accuracy: 97.95%

- F1 Score (weighted): 97.87%

- Loss: 0.0755

## Class-wise Performance

1. Positive Sentiment (Class 2)

- Precision: 99%

- Recall: 100%

- F1-score: 99%

- Best performing class due to majority representation

2. Neutral Sentiment (Class 1)

- Precision: 82%

- Recall: 70%

- F1-score: 76%

- Moderate performance, struggles with recall

3. Negative Sentiment (Class 0)

- Precision: 87%

- Recall: 84%

- F1-score: 86%

- Good performance despite small representation

## Key Observations

1.Class Imbalance Impact

- Model performs exceptionally well on majority class (Positive)

- Maintains good performance on minority classes despite imbalance

2. Confusion Matrix Analysis

```

[[ 615 76 39]

[ 71 985 348]

[ 17 136 31219]]

```

- Most misclassifications occur between neutral and positive classes

- Very few negative reviews misclassified as positive (39 cases)

3. Training Behavior

- Model converged well with decreasing loss

- No significant overfitting observed

- Consistent performance across training epochs

## Recommendations

1. Consider techniques to address class imbalance:

- Oversampling minority classes

- Class weights in loss function

- Collecting more negative and neutral reviews

2. Potential improvements:

- Longer training with early stopping

- Data augmentation for minority classes

- Fine-tuning hyperparameters

Overall, the model shows strong performance despite the significant class imbalance, making it suitable for production use in sentiment analysis tasks.