Natalia Torres 01/31/2025

Dylan Rodríguez Flores

IronHack - Puerto Rico

**Project NLP | Automated Customer Reviews Report**

**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 evaluate how a traditional ML solutions (NaiveBayes, SVM, RandomForest, etc) compares against a Deep Learning solution (e.g, a Transformer from HuggingFace) when trying to analyse a user review, in terms of its score (positive, negative or neutral).

**Problem Statement:**

The company receives thousands of text reviews every month, making it challenging to manually categorize and analyze, and visualize them. An automated system can save time, reduce costs, and provide real-time insights into customer sentiment.

Automatically classifying a review as positive, negative or neutral is important, as often:

- Users don't leave a score, along with their review

- Different users cannot be compared (for one user, a 4 might be great, for another user a 4 means "not a 5" and it is actually bad)

**Project goals:**

1. The ML/AI system should be able to run classification of customers' reviews (the textual content of the reviews) into positive, neutral, or negative.
2. You should be able to compare which solution yields better results:
3. One that reads the text with a Language Model and classifies into "Positive", "Negative" or "Neutral"
4. One that transforms reviews into tabular data and classifies them using traditional Machine Learning techniques

**Data Collection:**

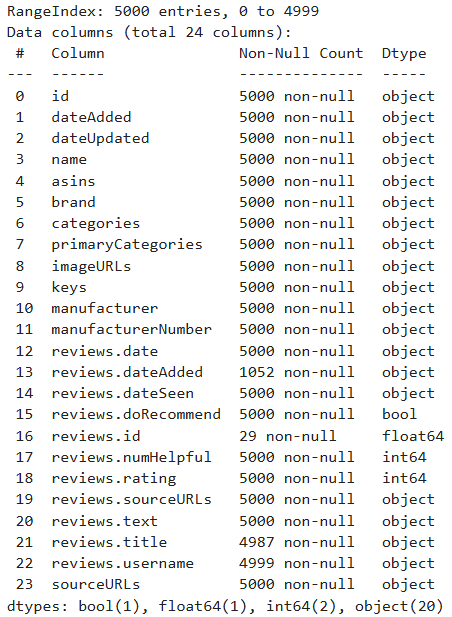
* You may use the publicly available and downsized dataset of Amazon customer reviews from their online marketplace, such as the dataset found [here](https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products/data).
* You also pick any product reviews datasets from [here](https://huggingface.co/datasets/amazon\_us\_reviews). Make sure your computing resources can handle both your dataset size and the machine learning processes you will follow.

In order to do this, you should transform all the scores with the following logic:

* Scores of 1,2 or 3: Negative
* Scores of 4: Neutral
* Scores of 5: Positive

**Exploratory Data Analysis (For both models)**

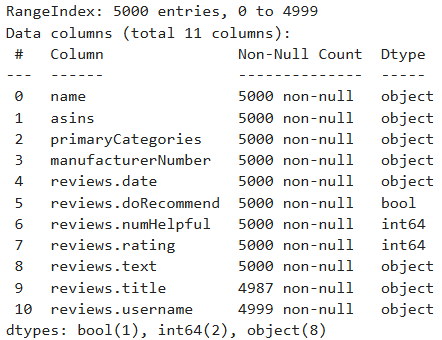
The dataset was downloaded from the provided webpage (Kaggle). After loading the dataset we reviewed the information using the .info() method and found that the dataset contained the following columns:



Upon further inspection, we decided to remove irrelevant columns from the data set to minimize visual clutter for more efficient processing. The columns selected for removal were:

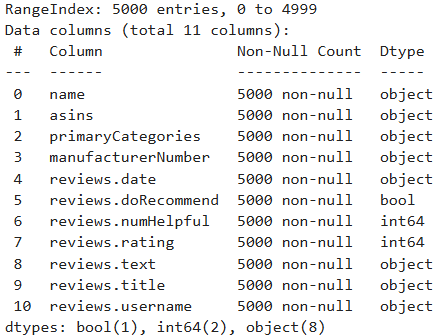
1. Id - non-essential
2. dateAdded - redundant
3. dateUpdated - redundant
4. Categories - redundant
5. imageURLs - non-essential
6. Keys - non-essential
7. Manufacturer - redundant
8. reviews.dateAdded - redundant
9. reviews.dateSeen - redundant
10. reviews.id - non-essential
11. sourceURLs - non-essential

After column removal:

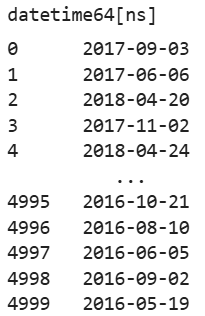


When the selected columns were removed, we proceeded to deal with the null values in the remaining cells. From the image above we can see that there’s exactly 14 values that are unaccounted for, 13 in the reviews.title column and a single username. To deal with this, we simply assigned the title ‘review’ to the empty titles and ‘anonymous’ to the single review without a username. Once these corrections are made, the dataset is ready for use.

Data set prepared:



Here we changed the data type in reviews.date to be a date format and we removed the hours.



**Traditional NLP & ML approach**

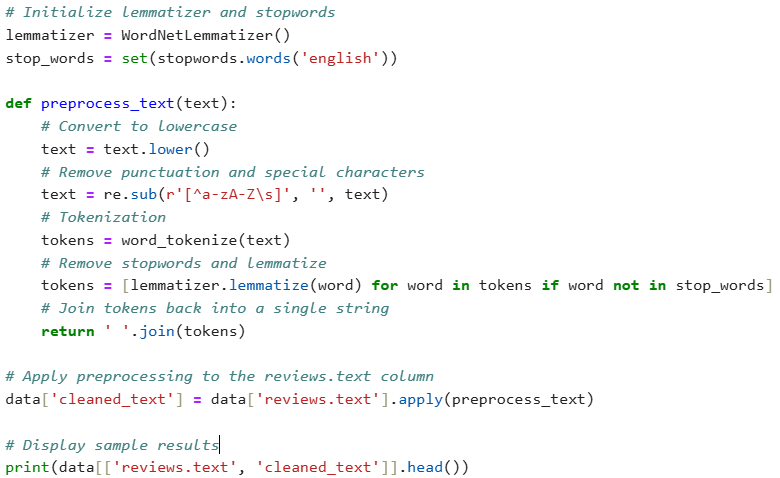
**1. Data Preprocessing**

**1.1 Data Cleaning**

* Remove special characters, punctuation, and unnecessary whitespace from the text data.
* Convert text to lowercase to ensure consistency in word representations.

**1.2 Tokenization and Lemmatization**

* Tokenize the text data to break it into individual words or tokens.
* Apply lemmatization to reduce words to their base or root form for better feature representation.



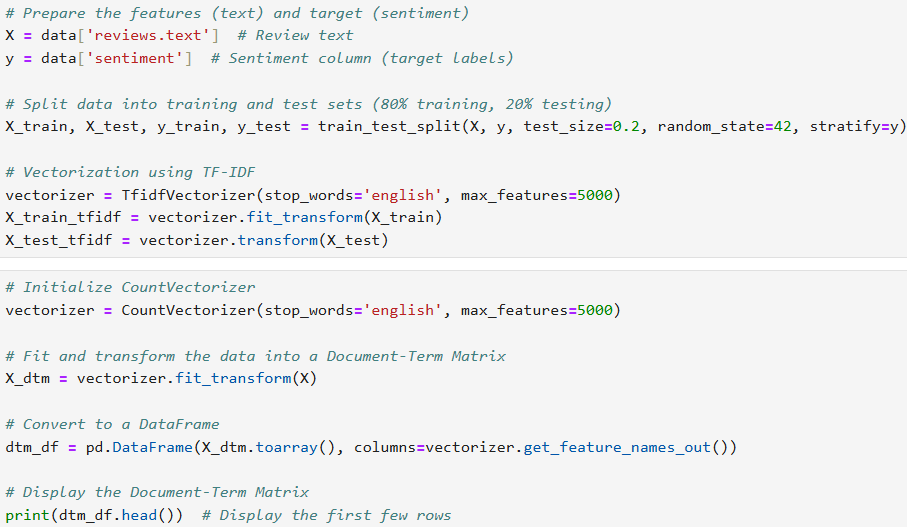
The function displayed above lowers the case of all the text, removes special characters, tokenizes the words, and finally lematizes them so they can then be added into a new column named ‘clean\_text’.

**1.3 Vectorization**

* Use techniques such as CountVectorizer or TF-IDF Vectorizer to convert text data into numerical vectors.
* Create a document-term matrix representing the frequency of words in the corpus.

Words must now be converted into a numerical representation before they can be used in a model. Vectorization transforms text into a structured format, allowing machine learning models to recognize patterns. For this model we used TF-IDF (Term Frequency-Inverse Document Frequency) for the following reasons:

* Adjusts BoW by giving more importance to rare words.
* Words that appear frequently across documents get lower importance.
* Example: "machine learning" is weighted more in a dataset about general technology than in a dataset entirely about AI.



**2. Model Building**

**2.1 Model Selection**

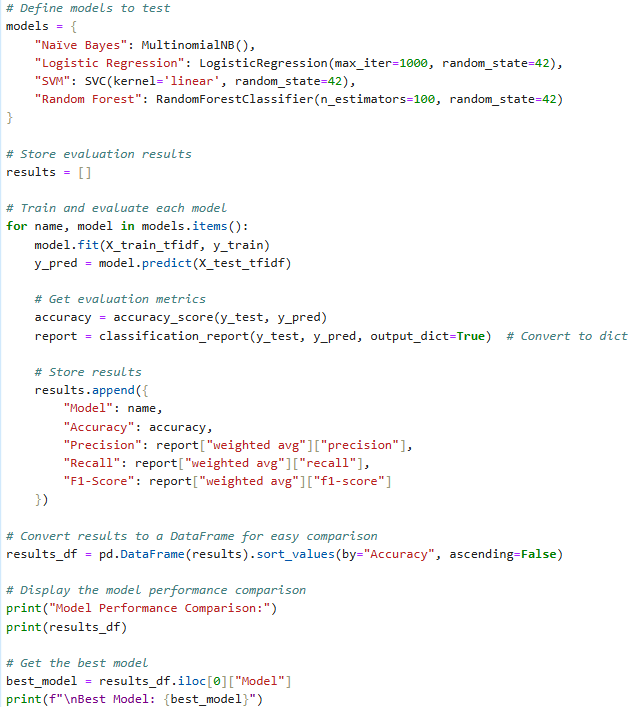
Explore different machine learning algorithms for text classification, including:

* Naive Bayes
* Logistic Regression
* Support Vector Machines
* Random Forest
* Evaluate each algorithm's performance using cross-validation and grid search for hyperparameter tuning.

**2.2 Model Training**

* Select the best-performing algorithm based on evaluation metrics such as accuracy, precision, recall, and F1-score.
* Train the selected model on the preprocessed text data.

We evaluated multiple models to determine which one performs best for sentiment classification. Each machine learning model has different strengths and weaknesses, so testing multiple approaches helps find the most effective one for our dataset. The following code runs a test on the models mentioned previously and returns the model with the highest results.



**3. Model Evaluation**

**3.1 Evaluation Metrics**

* Evaluate the model's performance on a separate test dataset using various evaluation metrics:
* Accuracy: Percentage of correctly classified instances.
* Precision: Proportion of true positive predictions among all positive predictions.
* Recall: Proportion of true positive predictions among all actual positive instances.
* F1-score: Harmonic mean of precision and recall.
* Calculate confusion matrix to analyze model's performance across different classes.

The model that displayed the best performance was the Random Forest which we utilized. After selecting the model, we did some hyperparameter tuning and found the following parameters returned the best results:

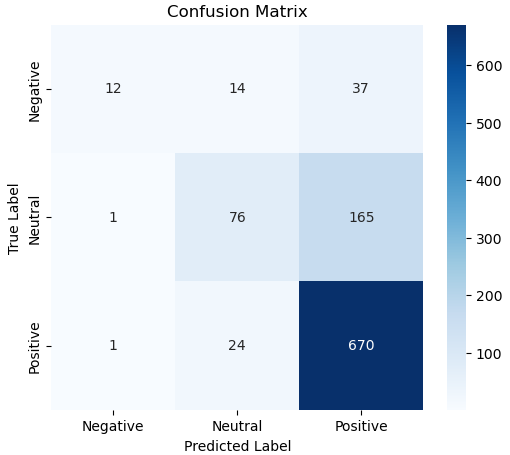
Best Hyperparameters: {'max\_depth': 30, 'max\_features': 'log2', 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'n\_estimators': 200}

Below is the code implementation of the model:



**3.2 Results**

* Model achieved an accuracy of 76% on the test dataset.
* Precision, recall, and F1-score for each class are as follows:
* Class Positive: Precision=77%, Recall=96%, F1-score=86%
* Class Negative: Precision=86%, Recall=19%, F1-score=31%
* Class Neutral: Precision=67%, Recall=31%, F1-score=43%
* Confusion matrix showing table and graphical representations



**Transformer approach (HuggingFace)**

A classification model, (bonus: summarization), and a dashboard are expected in this section.

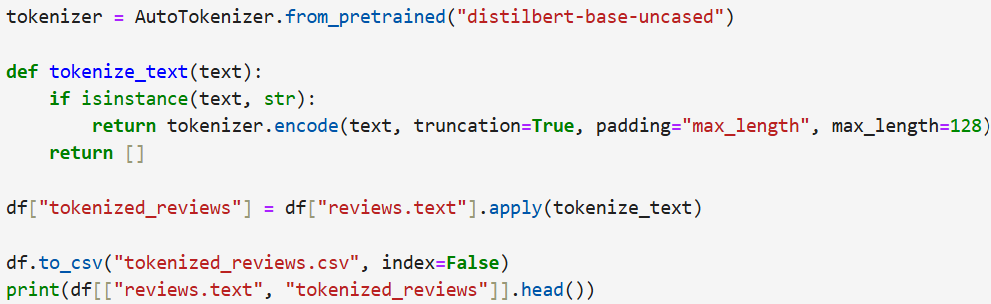
**1. Data Preprocessing**

**1.1 Data Cleaning and Tokenization**

* Clean and tokenize the customer review data to remove special characters, punctuation, and unnecessary whitespace.

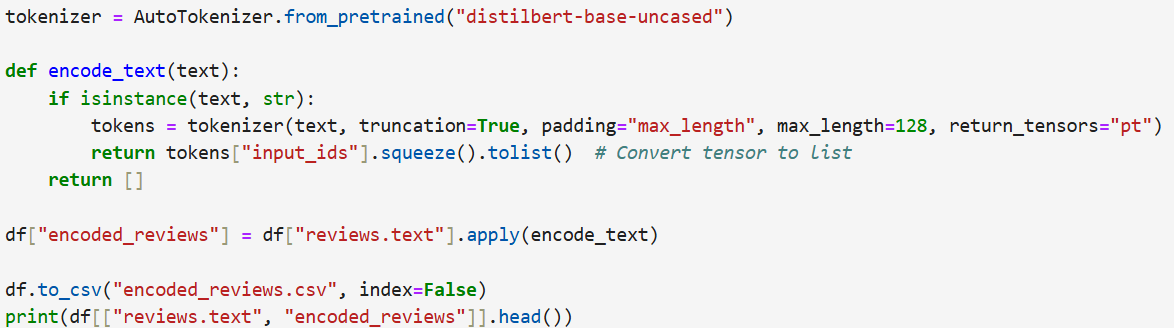


* Apply tokenization using the tokenizer provided by the HuggingFace Transformers to convert text data into input tokens suitable for model input.



**1.2 Data Encoding**

* Encode the tokenize input sequences into numerical IDs using the tokenizer's vocabulary.



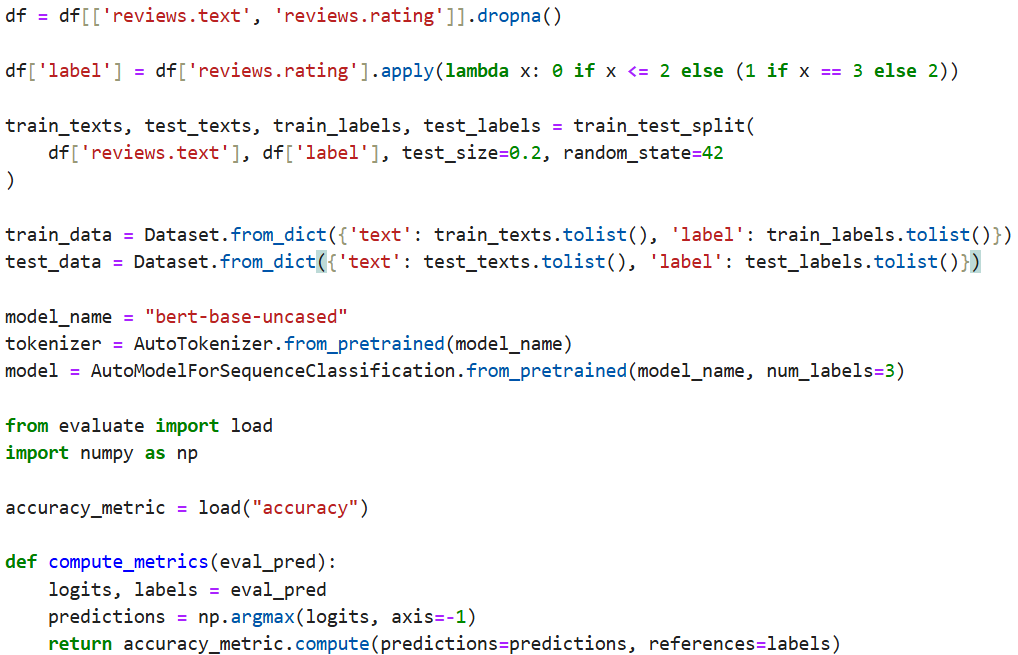
**2. Model Building**

**2.1 Model Selection**

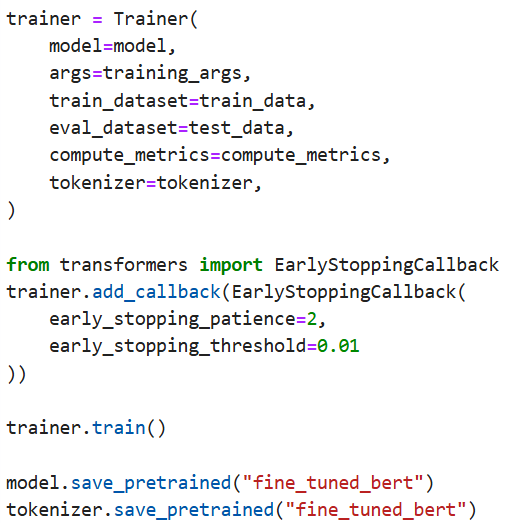
* Explore transformer-based models available in the HuggingFace Transformers, potentially:
* BERT (Bidirectional Encoder Representations from Transformers)
* RoBERTa (Robustly Optimized BERT Approach)
* DistilBERT (Lightweight version of BERT)
* Selected a pre-trained transformer model suitable for text classification tasks, and justify your choice.
* Share the accuracy using the pre-trained model on your data \*\*without\*\* fine-tuning. This is your base model.

**(BONUS) 2.2 Model Fine-Tuning**

* Fine-tuned the selected pre-trained model on the customer review dataset using transfer learning.
* Configured the fine-tuning process by specifying parameters such as batch size, learning rate, and number of training epochs.



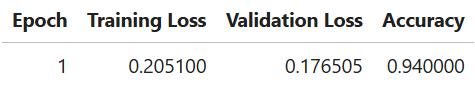




**3. Model Evaluation**

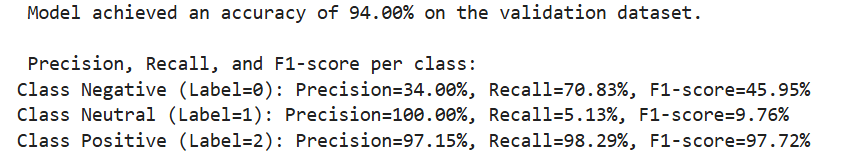
**3.1 Evaluation Metrics**

* Evaluate the base model and the fine-tuned model's performance on a separate validation dataset using standard evaluation metrics:
* Accuracy: Percentage of correctly classified instances.
* Precision: Proportion of true positive predictions among all positive predictions.
* Recall: Proportion of true positive predictions among all actual positive instances.
* F1-score: Harmonic mean of precision and recall.
* Calculate confusion matrix to analyze model's performance across different classes.



**3.2 Results**

* Model achieved an accuracy of X% on the validation dataset.
* Precision, recall, and F1-score for each class are as follows:
* Class Positive: Precision=X%, Recall=X%, F1-score=X%
* Class Negative: Precision=X%, Recall=X%, F1-score=X%
* Class Neutral: Precision=X%, Recall=X%, F1-score=X%
* Confusion matrix



**4. Conclusion**

Both models were fine tuned as much as possible, within the specified timeframe, to get the best results. The traditional ML model selected (Random Forest) achieved a 76% accuracy score overall. We found the model to be heavily biased towards positive reviews, being able to identify positives much more often than the other 2 classes. Perhaps a larger dataset would help train the model to make it more effective at detecting the negative and neutral words for better identification.