Text Analytics

University of Pisa

Project by
Giuseppe Muschetta
Master Degree: Data Science and Business Informatics
Enrollment Number: 564026
Grade: Awarded 30 cum laude by Professor Laura Pollacci

Contents

Introduction

In the context of the ever-evolving landscape of e-commerce, the ability to understand customer feedback has become a crucial factor for businesses seeking to enhance their product offerings and service quality. This report details the development and deployment of a Natural Language Processing (NLP) project with the objective of performing sentiment analysis on Amazon product reviews.

The primary objective was to extract actionable insights from customer sentiments in order to inform strategic decision-making. This project employed advanced machine learning techniques and NLP tools to analyse hundred thousands of reviews in order to determine the underlying customer sentiments, which ranged from positive to negative.

The methodology section that follows outlines the specific technologies and algorithms employed, while subsequent sections discuss the results and their implications for business strategies. This study not only demonstrates the technical feasibility of sentiment analysis in a real-world application but also highlights its potential impact on business operations.

1 DATA UNDERSTANDING & EXPLORATION

1.0.1 Dataset Description

The dataset is derived from Amazon product reviews, providing a rich resource for understanding customer sentiments across various product categories. Below, we detail the dataset's structure, statistical properties, and initial observations.

1.0.2 Source of Data

The data for this analysis was obtained from the comprehensive Amazon Product Reviews dataset, available at

`https://www.kaggle.com/datasets/arhamrumi/amazon-product-reviews`

This source was selected due to its broad coverage of products and depth of review content.

1.0.3 Initial Data Overview

Upon loading, the dataset consists of approximately 568453 rows and 10 columns, where each row represents an individual review. The primary columns include:

- **Text**: Contains the text of the customer review.
- **Score**: Numerical rating given by the customer, ranging from 1 to 5.
- **Summary**: Contains a short summary of the review
- **Product ID**: The ID associated to the product.
- **ProfileName**: The string of the name or nick name associated to the user
- **Time**: Review time in timestamp

Other columns, such as "Id", "UserId", "HelpfulnessNumerator" and "HelpfulnessDenominator," are of minimal utility and will not be considered in our analysis.

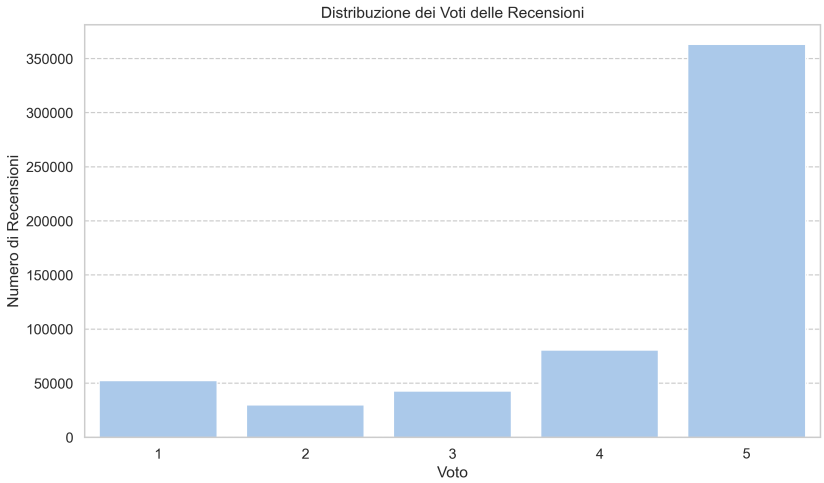
The column "Summary" was not included in the sentiment analysis because it would have provided the machine learning models with an excessive amount of information, which was therefore not considered. The column "ProductID" revealed the considerable number of products that were reviewed in the 568,000 reviews. The column "Time" gave us insights into how the sentiment of the reviews varied over time, how the number of reviews varied over time, and when we say "over time" we mean the entire time interval defined by the timestamp, i.e., from 1999 to 2012.

Furthermore, the "Time" column enabled the identification of the day of the week with the highest number of product reviews and the days with the lowest number of reviews. The "ProfileName" column facilitated the determination of whether the majority of reviews on Amazon are written by women or men.

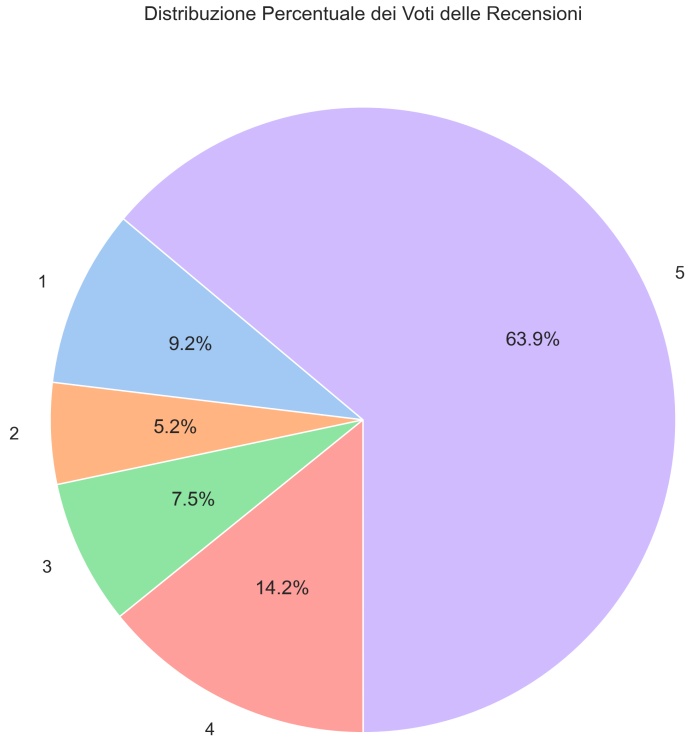
1.0.4 Statistical Summary

A summary of key numerical features in the dataset is presented below:

- **Ratings**: The 'Score' field averages 4.183 with a standard deviation of 1.31. The ratings are on a scale of 1 to 5, indicating a broad spectrum of customer satisfaction. This result is typical for the majority of Amazon Products. We can see the distribution of the review with an histogram and a pie-chart:



(a) Histogram of review scores



(b) Pie chart of review scores

Figure 1: Two images side by side

To be specific, 363122 reviews have a rating of 5/5, 80655 have a rating of 4/5, 42640 have a rating of 3/5, 29769 have only 2/5 and 52268 have only one star.

- **Review Length**: The mean length of the reviews, measured by the total number of characters, was determined to be 436, with a standard deviation of 445. Given that the average length of an English word is approximately 5.5 characters, we can estimate that the average review length in our Amazon dataset is around 80 words. This result was also confirmed by the calculations performed, which confirmed the average of exactly 80 words per review. The following image takes into account how the lengths of reviews are distributed, calculated by number of characters, also according to their sentiment, positive or negative.

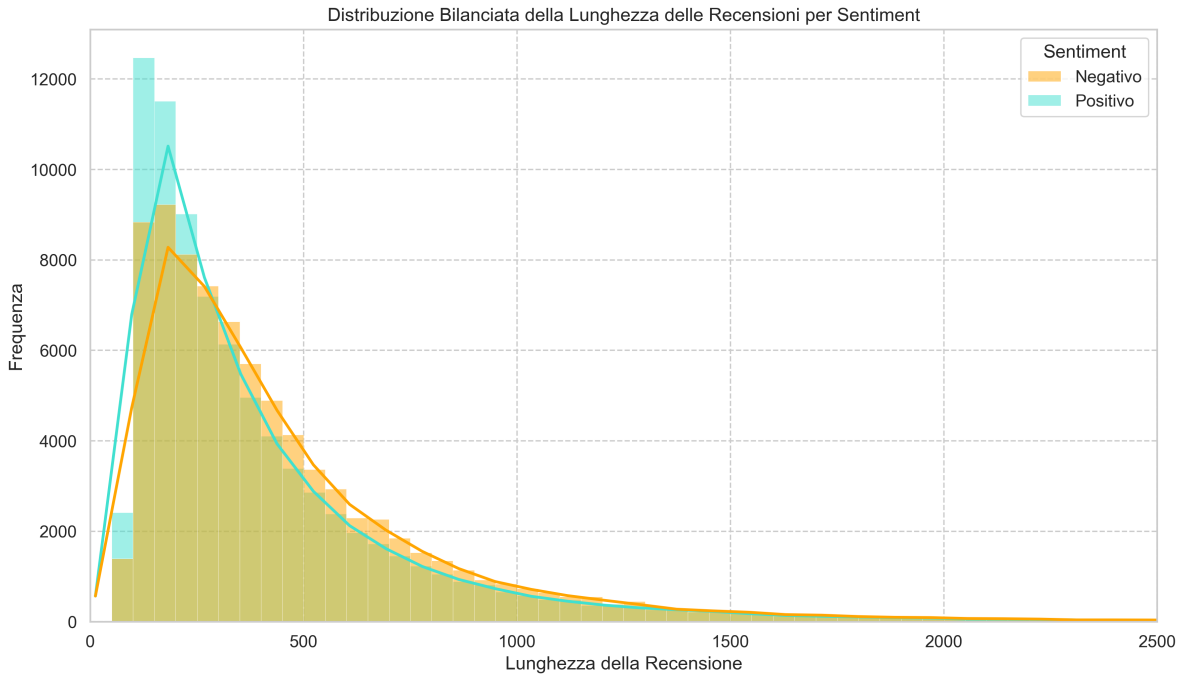


Figure 2: Distribution of review lengths based on sentiment type

- **Reviews by Gender**: Have men or women written more reviews? We studied this with a library called 'gender_guesser':

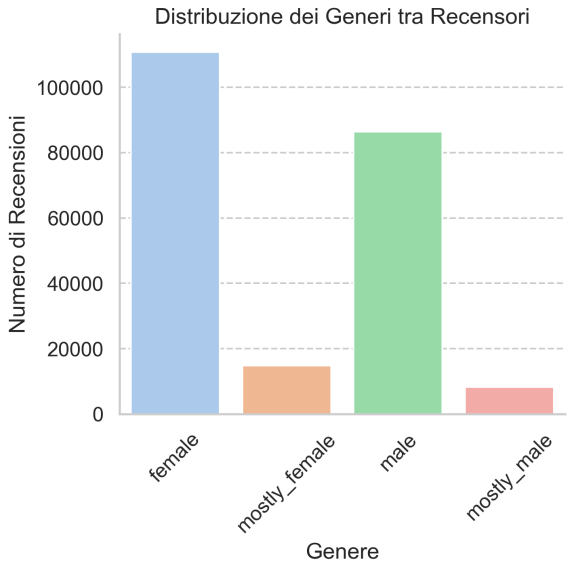


Figure 3: distribution of genders among reviewers

- **Reviews by day of the week**: Which are the days in which people tend to write more reviews? We did also this study and the following result might surprise you:

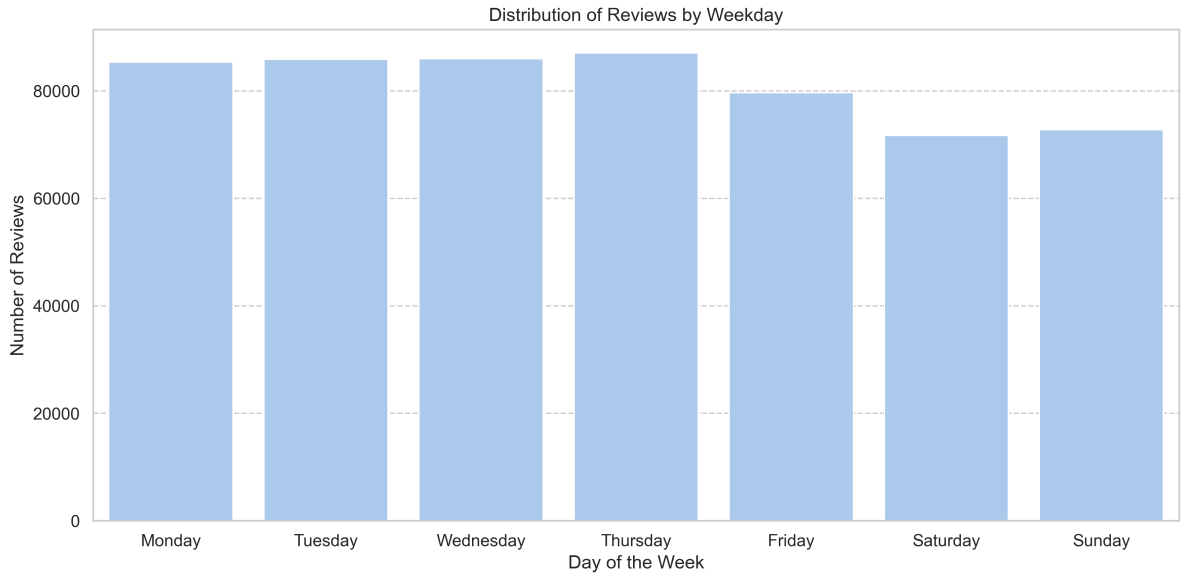


Figure 4: Distribution of reviews by weekday

- **Reviews by Time**: The dataset spans from 1999 to 2012. An analysis of this period reveals a significant trend in the volume of reviews. Initially, the number of reviews grew moderately; however, starting in 2007, there was a noticeable exponential increase. This surge can be attributed to several factors. Firstly, the widespread adoption of the internet during this period made online shopping more accessible to a broader audience. Secondly, the growing recognition and trust in the Amazon brand significantly contributed to increased consumer engagement. These factors combined have led to the rapid escalation in review submissions observed towards the latter part of the dataset's timeframe.

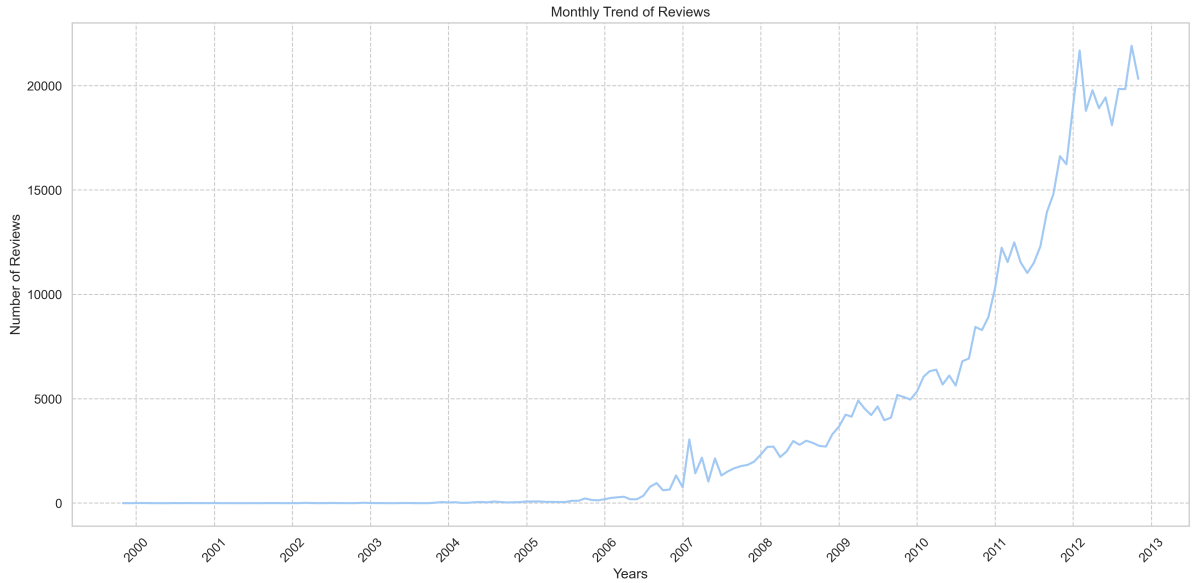


Figure 5: Monthly trend of reviews

- **Reviews by Sentiment**: The graph in Figure 6 illustrates the annual trend of positive and negative Amazon reviews from 1999 to 2012. It shows a moderate increase in both types of reviews until 2006, followed by an exponential growth in positive reviews from 2007 onwards. The surge in positive reviews significantly outpaces the growth of negative reviews, highlighting increased customer satisfaction and the rising popularity of Amazon during this period.

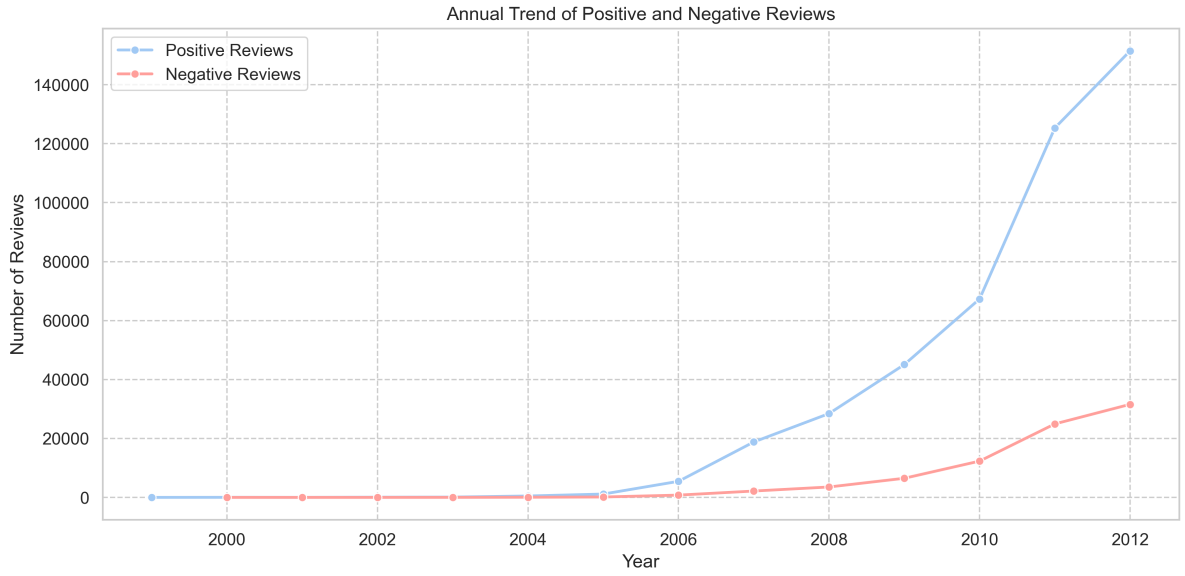


Figure 6: Annual trend of positive or negative reviews

1.0.5 Final Considerations

Our investigation of the Amazon review dataset, spanning from 1999 to 2012, has yielded valuable insights across various dimensions. The dataset, comprising approximately 568,453 reviews, provides a rich source of consumer feedback. Analysis of key features such as review scores, lengths, and temporal trends has revealed intriguing patterns. Notably, we observed a steady increase in the volume of reviews over time, with an exponential surge from 2007 onwards. This is indicative of heightened consumer engagement. Furthermore, investigations into review lengths, sentiment trends, and demographic distributions have offered insights into consumer behaviour.

2 DATA PREPARATION AND CLEANING

As evidenced by the data understanding phase, Amazon reviews exhibit a notable bias towards highly positive ratings. This is evidenced by the fact that the average rating of the reviews in the entire dataset is 4.183, and it is also noteworthy that 64% of the entire dataset comprises reviews with 5 stars. The objective is to divide the dataset into two subsets: one comprising reviews with positive sentiment and the other with negative sentiment. Reviews with neutral sentiment are excluded from the analysis as they provide no useful information. By focusing on reviews with positive and negative sentiment, it is possible to gain a deeper understanding of the nuances associated with these two emotional states. To achieve this, the data was divided as follows: all reviews with a rating of 4 or 5 were classified as positive, while reviews with a rating of 1 or 2 were classified as negative. It should be noted that this division is inherently imbalanced

due to the considerations made at the beginning of this process. In fact, there are 443,777 reviews with positive sentiment, while only 82,037 reviews have negative sentiment.

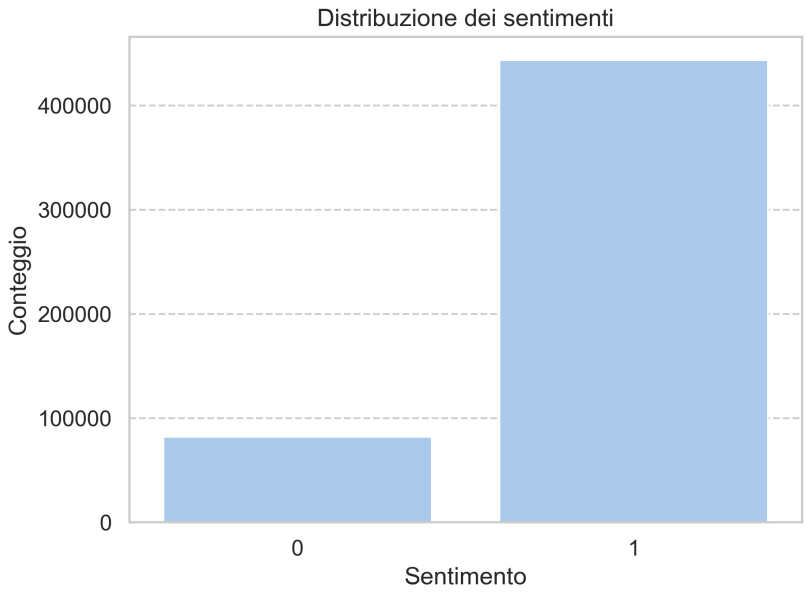


Figure 7: Total distribution of reviews by sentiment: Very unbalanced

A binary ranking in these conditions is not optimal. The optimal solution is to achieve perfect balance between the two sentiment classes and then downsample the majority class to match the minority class. This results in a balanced dataset with 82,037 positive reviews and 82,037 reviews with negative sentiment.

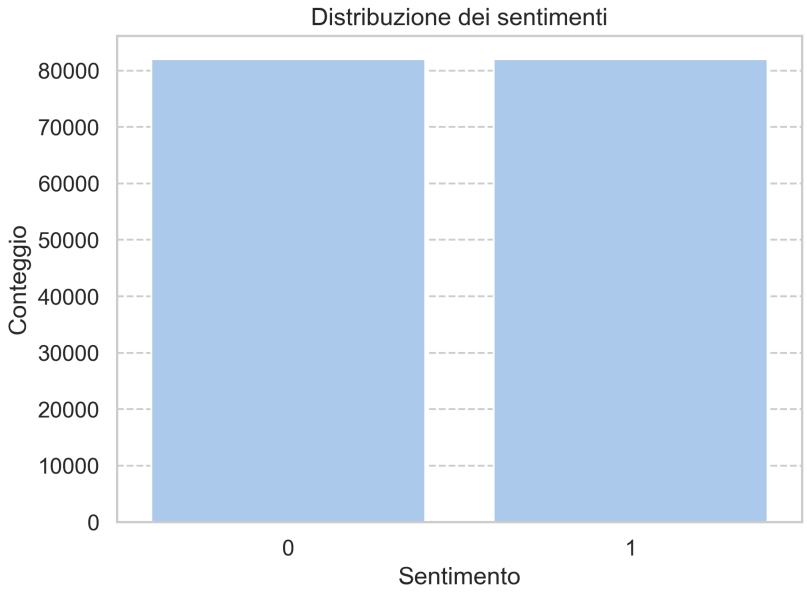


Figure 8: Total distribution of reviews by sentiment: Balanced

Once the dataset has been balanced, the heart of text analysis can be entered, commencing with the text processing phase. This involves tokenization, lemmatization, the management of contractions, the removal of numbers, punctuation, and useless spaces, as well as the filtration of web addresses, email addresses, HTML/XML tags, negations, which is a fundamental aspect of text analysis, collocations (bigrams), and so forth.

2.1 Libraries used for advanced text data processing:

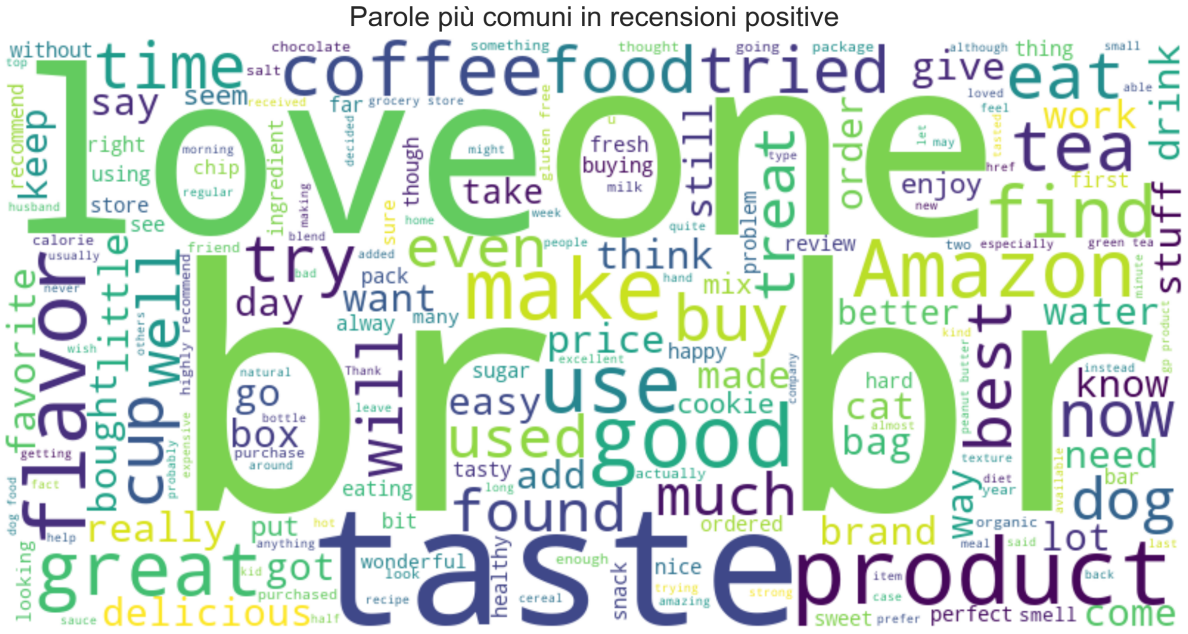
In the rapidly evolving field of data science, effective text processing and data cleaning are foundational to the extraction of meaningful insights from unstructured textual data. This section explores the essential techniques for refining raw data, including normalization, noise removal, and semantic enrichment. We will use state-of-the-art libraries to address these preliminary steps, in order to ensure the reliability of subsequent analyses, paving the way for sophisticated applications in natural language processing and beyond.

- **NLTK (Natural Language Toolkit)**: Utilized for loading a comprehensive suite of English stopwords, aiding in the removal of the most common, yet least informative words from texts.
- **BeautifulSoup**: This library is particularly efficient at stripping HTML tags from texts, which simplifies the cleaning of data extracted from web pages.
- **RE (Regular Expressions)**: Crucial for the removal of email addresses and URLs, ensuring that the text data remains focused on content rather than extraneous web elements.
- **spaCy**: Known for its robust and versatile text processing capabilities, spaCy integrates various components such as tokenization, tagging, and named entity recognition. Importantly, it performs lemmatization implicitly, enhancing the consistency of the processed text.
- **Emoji**: Emojis are preserved during text processing because they carry meaningful sentiment indicators that are valuable for sentiment analysis tasks.
- **Contractions**: This library expands contracted forms (*here's*, *you're*, *should've*), which is essential for standardizing text input and improving analysis accuracy.
- **Gensim**: Applied specifically for detecting collocations like bigrams, Gensim helps uncover frequently paired words, which can provide insights into common language patterns within the data set.
- **Custom Functions**: I have developed a series of functions tailored to accurately handle negations in text, a critical aspect when determining the overall sentiment of statements. Specifically, these functions detect negation tokens like "not," "never," and "no," and appropriately modify the subsequent token to reflect the negation, creating compounds such as "not_bad" or "no_talking." This method ensures that the sentiment analysis retains the full context of expressions, greatly enhancing the accuracy of the outcome.
- **Observation**: An important note is that lemmatization, a crucial aspect of text normalization, is efficiently managed by spaCy's sophisticated text processing pipeline.

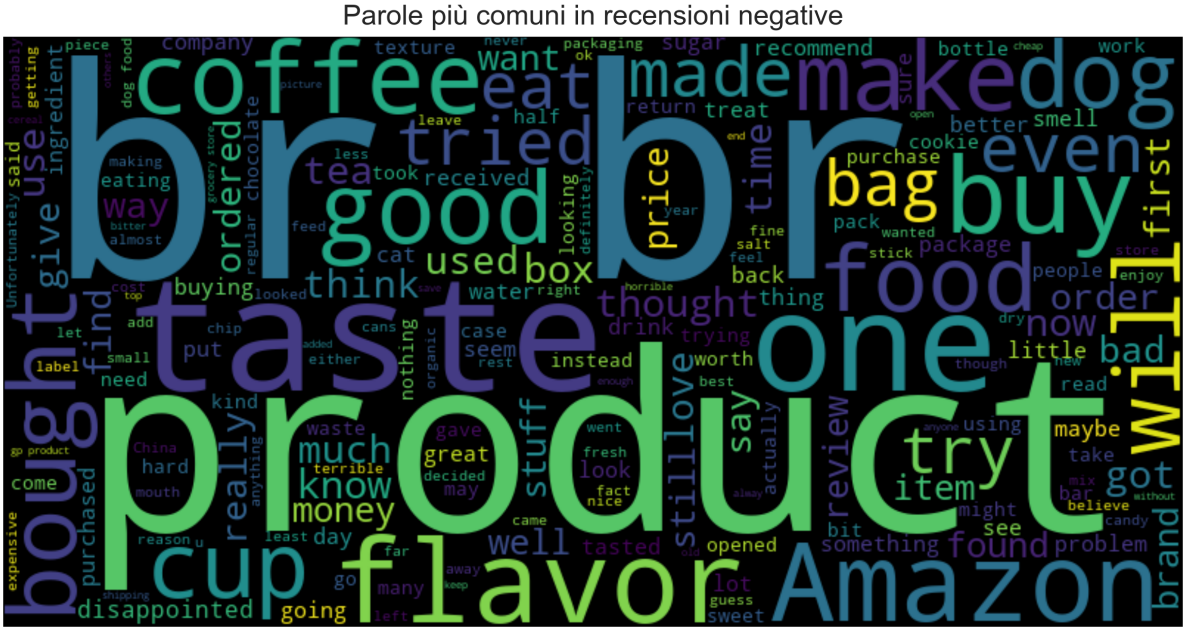
2.2 Wordclouds

To better highlight the effectiveness of the cleaning performed using advanced Text Analytics techniques and libraries, we can generate word clouds from both the raw and the cleaned, processed text. Each version of these word clouds will take into account the sentiment of the reviews. Thus, we will have two word clouds for the raw text: one representing all reviews with positive sentiment and another for those with negative sentiment. Similarly, we will produce two more word clouds for the cleaned text, each reflecting the positive and negative sentiments of the reviews.

We went from **these raw text Wordclouds...**:



(a) Raw text wordcloud for positive sentiment



(b) Raw text wordcloud for negative sentiment

Figure 9: Raw text wordclouds

The raw text typically contains a substantial abundance of tokens including various punctuation marks such as periods "." (331,627) and commas "," (237,252), alongside a myriad of letters and words like the pronouns "I" (223,429), "You", "me", etc. This inventory of tokens is indicative of the unfiltered nature of the data, encompassing a broad spectrum of lexical items used in daily communication.

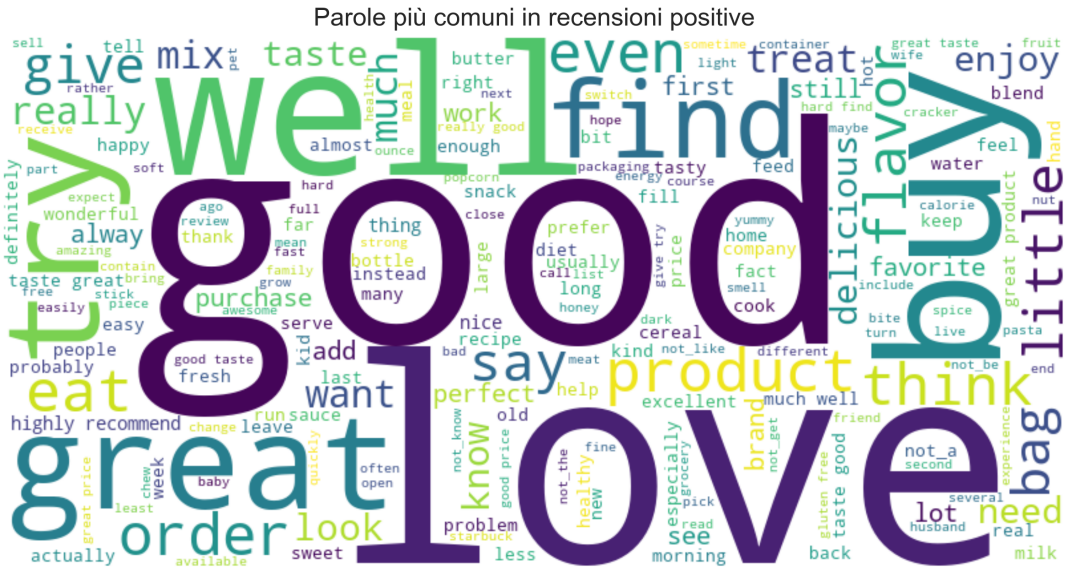
Moreover, the text exhibits a significant frequency of both definite articles like "the" (218,562) and indefinite articles such as "a" (179,712), alongside other common grammatical articles and widely used verbs like "to do." This prevalence highlights typical language usage patterns, essential for understanding linguistic constructs in natural language processing.

Emojis, too, are prevalent (>: 91,986; <: 91,664; /: 87,251) and will be intentionally preserved in the cleaned text. Their inclusion is strategic, as these symbols play a crucial role in enriching the text by amplifying the underlying sentiment, thereby providing deeper insights into emotional expressions within the text.

Another distinctive feature of the English language evident in the corpus is the frequent use of contractions such as 's, n't, 've, etc. Managing these contractions is vital for accurate text analysis, necessitating sophisticated handling to ensure that the subtleties of informal speech are preserved and accurately represented in analytical models. This management will be executed using the most advanced NLP libraries available, which will be meticulously integrated into our code and adeptly employed within the "clean-text" global text processing function.

In addition to addressing contractions, as previously noted, our text processing approach will also include the management of negations and the identification of bigrams pairs of words that, when combined, yield a meaning distinct from their separate interpretations. Recognizing such linguistic features is essential for constructing a more nuanced understanding of the text's semantic landscape. Examples of bigrams like "game over" illustrate the complexity of language, where combined lexical forms create specific connotations and implications that are crucial for comprehensive text analysis. This layered approach to text normalization ensures that the final processed text retains the richness of its original linguistic context while being structured for advanced analytical tasks.

...to cleaned and processed ones:



(a) Cleaned text wordcloud for positive sentiment

(b) Cleaned text wordcloud for negative sentiment

Figure 10: Cleaned text wordclouds

Upon immediate examination, it becomes evident that the cleaned text is devoid of punctuation, both definite and indefinite articles, numerical figures, email addresses, and HTML/XML tags.

The most commonly occurring words that neither impart negative nor positive sentiment have been eliminated through the application of a stopwords list. Furthermore, negative words deemed critical for preserving sentence meaning have been systematically organized into composite phrases represented as single tokens, such as not_good, not_tasty, no_other, never_go, and never_like, among others.

Additionally, contractions have been systematically expanded; for example, "she's" is converted to "she is." Through the process of lemmatization, all words are reduced to their base forms; for instance, in the sentence "she had the very best coffee of her life," the verb "had" is transformed to "have," and "best" is altered to "good."

3 ML MODELS FOR SENTIMENT ANALYSIS

Upon cleaning and processing the text, we are equipped to undertake sentiment analysis using a variety of machine learning algorithms. This analysis commences with the relatively simple yet robust Naive Bayes classifier, proceeds to Support Vector Machines (SVMs) which typically yield higher accuracy compared to Naive Bayes, and culminates in a sophisticated model employing a sequential neural network architecture. This advanced model comprises various layers, including one-dimensional convolutional layers (CONV1D), bidirectional Long Short-Term Memory (LSTM) layers, dense layers, MaxPooling layers, and multiple dense and dropout layers to mitigate overfitting.

The exploration of machine learning models for sentiment analysis in this project extends to examining a pre-trained model known as DistilBERT, a streamlined version of the BERT architecture. DistilBERT utilizes slightly more than half the parameters of the original BERT model. Remarkably, with our neural network model harnessing "only 2.5 million" parameters, we have achieved performance metrics basically on par with those of DistilBERT, which employs 66 million parameters. This achievement is arguably the most significant result of this Text Analytics project, demonstrating that highly efficient machine learning models can be developed without the extensive computational resources typically required for state-of-the-art outcomes.

3.1 Naïve Bayes: MultinomialNB

Before sentiment analysis can be conducted using the Naïve Bayes model (MultinomialNB), it is essential to ensure that the model receives an appropriate input. For this purpose, a pipeline has been constructed. The initial stage of this pipeline employs TF-IDF (Term Frequency-Inverse Document Frequency) for counting and vectorizing the text. Subsequently, as the second step in the pipeline, the SelectKBest method is used for feature extraction, leveraging the chi-squared scoring function. This method evaluates the significance of various features by calculating the chi-squared statistic between each feature and the class label. The top 'k' features demonstrating the strongest relationships with the class label are chosen for model training. This technique aids in enhancing the model's efficiency and accuracy by reducing the dimensionality of the feature space and concentrating on the most informative attributes.

The third phase of the pipeline involves the actual sentiment analysis model, the Multinomial Naïve Bayes classifier. This model is particularly well-suited for classification tasks involving discrete features (such as word counts in text classification). The Multinomial Naïve Bayes classifier applies Bayes' theorem, incorporating the assumption of independence between predictors. In practice, it calculates the probability of each class label given a set of input features, and assigns the class label with the highest probability to the input text.

The classifier's effectiveness in sentiment analysis stems from its ability to handle large volumes of data and its relative simplicity, which allows for quick training and prediction phases. Despite the simplicity and the independence assumption, which may not always hold true, this model often performs remarkably well on text data. The integration of this classifier into the pipeline allows for a seamless transition from raw text processing to sentiment prediction, streamlining the workflow and enabling efficient data handling and analysis.

3.1.1 Model Evaluation and Performance

In this section, we will explore the performance of the model by evaluating key metrics such as accuracy, precision, recall, and F1-score. These metrics will be assessed using tools such as the confusion matrix, the Receiver Operating Characteristic (ROC) curve with AUC properly displayed, and the precision-recall curve.

For binary classification tasks such as this, these metrics provide comprehensive insights into the model's effectiveness:

- **Accuracy** measures the overall correctness of the model across both classes
- **Precision** assesses the model's accuracy in predicting positive labels, which is crucial for applications where the cost of a false positive is high.
- **Recall or Sensitivity** indicates the model's ability to detect all relevant instances, essential in scenarios where missing a positive instance is costly.
- **F1-Score** is the harmonic mean of Precision and Recall, it provides a balance between precision and recall, offering a single metric that summarizes the model's accuracy in cases where an equilibrium between precision and recall is required.

Classification Report:

	precision	recall	f1-score	support
0	0.90	0.88	0.89	16408
1	0.88	0.90	0.89	16407
accuracy			0.89	32815
macro avg	0.89	0.89	0.89	32815
weighted avg	0.89	0.89	0.89	32815

Figure 11: Naive Bayes model classification report

To obtain the metrics discussed, we implemented a 5-fold cross-validation scheme. This method not only partitions the training set into five folds, using each one in turn for validation, but also ensures that the model is tuned across different segments of the dataset, enhancing the robustness of our findings.

The hyperparameter tuning was conducted using a grid search strategy with the following specifications:

- **TF-IDF Vectorizer Settings**:
  - `max_df` (maximum document frequency): Set to 0.5 to exclude terms that appear in more than 50% of the documents.
  - `min_df` (minimum document frequency): Set to 10 to exclude infrequently appearing terms.
  - `ngram_range`: Set to $(1, 2)$, allowing both unigrams and bigrams.
  - `use_idf`: Enabled (True), employing inverse document frequency weighting, crucial for sentiment analysis.

- `smooth_idf`: Enabled (True), to apply smoothing.

- **Feature Selection**:
  - `selkbest__k`: Number of top features set to 20000, focusing on the most relevant features.

- **Model Parameters**:
  - `model__alpha`: Smoothing parameter set to 0.5 to mitigate overfitting.
  - `model__fit_prior`: Set to True, indicating that the class priors should be learned from the data.

Utilizing these metrics, we can construct graphical representations that provide immediate insights into the efficacy of our model. Specifically, we refer to:

- ***Confusion Matrix*** further aids in visualizing the performance of the model by showing the true positives, true negatives, false positives, and false negatives, thus providing insight into the types of errors the model is making.

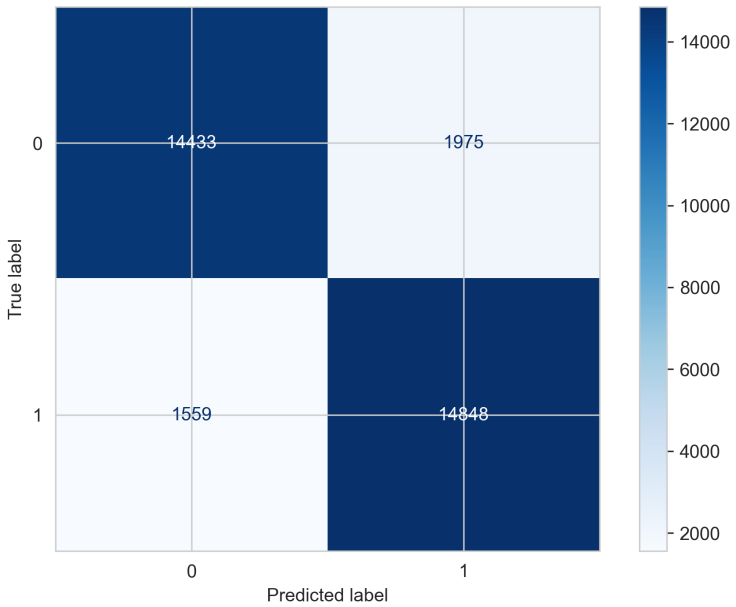


Figure 12: Confusion Matrix

- ***ROC Curve*** evaluates the trade-offs between true positive rate (sensitivity) and false positive rate (1-specificity) across various threshold settings.

- ***AUC: Area Under the Curve*** which quantifies the overall ability of the model to discriminate between the positive and negative classes irrespective of any specific threshold. The AUC value ranges from 0 to 1, where an AUC of 1 represents a perfect model that correctly classifies all positive and negative instances. An AUC value closer to 0.5 suggests that the model performs no better than random guessing. High AUC values indicate that the model has a high probability of ranking a randomly chosen positive instance higher than a randomly chosen negative one, across all possible classification thresholds.

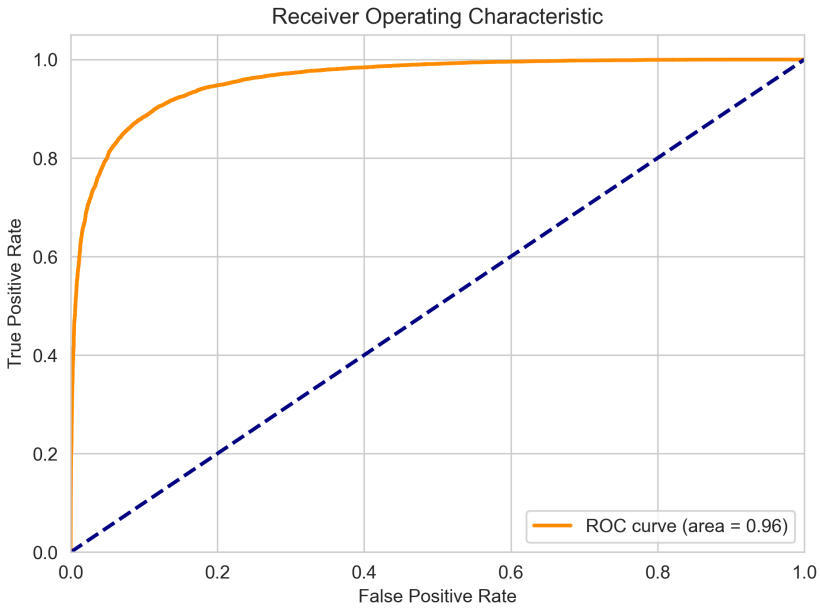


Figure 13: ROC curve with AUC displayed

- ***Precision-Recall Curve***: is a graphical representation that elucidates the trade-off between precision (the accuracy of positive predictions) and recall (the ability to identify all relevant instances) for a binary classifier at various threshold levels. It is particularly useful in scenarios where classes are imbalanced, providing insights into the classifier's performance across different thresholds without being influenced by the distribution of class labels. This curve is essential for understanding how changes in the threshold affect the balance between capturing relevant instances and maintaining accuracy in the positive predictions.

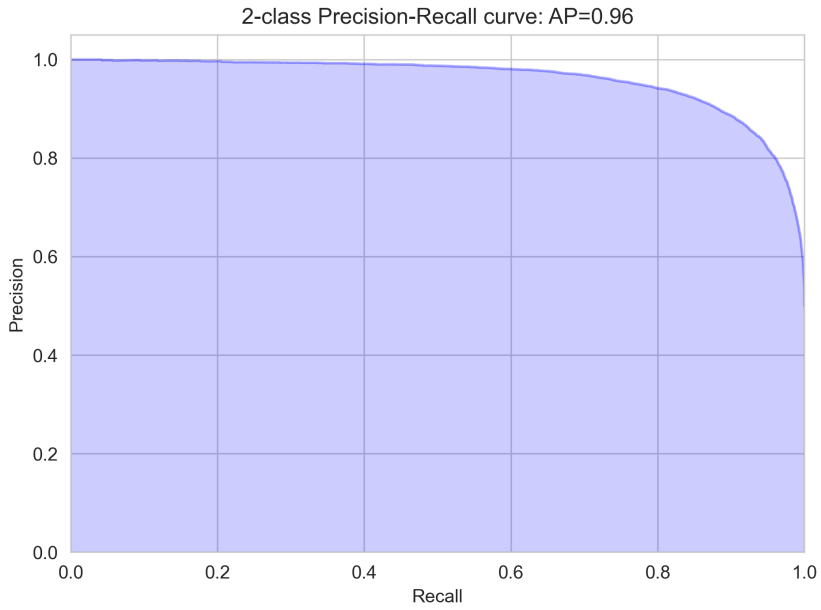


Figure 14: Precision-Recall curve

3.2 Support Vector Machine: LinearSVC

The Linear Support Vector Classification (LinearSVC) is a widely used algorithm in text analytics, renowned for its ability to handle high-dimensional data, such as those encountered in text classification tasks. This model operates on the principle of margin maximization, striving to achieve the widest possible margin between decision boundaries, which helps in enhancing classification accuracy and robustness.

Strengths of LinearSVC:

- Margin Maximization: Provides clear margins of separation between classes, reducing model complexity and enhancing generalization.
- Scalability: Efficiently handles large feature spaces typical in text data, making it suitable for large datasets.
- Flexibility: Supports different loss functions and penalties, allowing fine-tuning according to specific needs of the dataset.

Weaknesses of LinearSVC:

- Sensitivity to Noisy Data: Prone to overfitting especially in the presence of noise and overlapping class distributions.
- Parameter Sensitivity: Requires careful tuning of hyperparameters such as regularization and loss functions.

If we want to compare the LinearSVC model with the previous MultinomialNB one, we might assert that *LinearSVC generally outperforms Naive Bayes in text analytics* primarily due to its ability to handle feature interactions more effectively.

Unlike Naive Bayes, which assumes feature independence, LinearSVC takes into account the correlation between features. This leads to better performance, especially in complex text classification tasks where feature interdependencies are crucial.

Additionally, LinearSVC's loss minimization approach tends to result in better margin properties, providing a more definitive classification boundary that is beneficial in many practical applications.

The fact that LinearSVC outperforms Naïve Bayes can be seen from the classification report that follows:

Classification Report:

	precision	recall	f1-score	support
0	0.91	0.91	0.91	16408
1	0.91	0.91	0.91	16407
accuracy			0.91	32815
macro avg	0.91	0.91	0.91	32815
weighted avg	0.91	0.91	0.91	32815

Figure 15: LinearSVC model classification report

To achieve these stats we used 5-fold Cross-Validation with the following parameters grid:

- **TF-IDF Vectorizer Settings**:
  - `min_df` (minimum document frequency): Set at 10 to filter out terms that appear too infrequently.
  - `max_df` (maximum document frequency): Limited to 0.5 to exclude terms appearing in more than 50% of the documents.
  - `ngram_range`: Configured to $(1, 2)$ to capture both unigrams and bigrams.
  - `use_idf`: Enabled (True), to utilize inverse document frequency weighting, enhancing the model's focus on less frequent terms.
  - `smooth_idf`: Also set to True, applying smoothing to the IDF weights.
- **Feature Selection via SelectKBest**:
  - `k`: Number of top features set to 20000 to retain only the most influential features.
- **LinearSVC Model Parameters**:
  - `C` (Regularization parameter): Set at 1, balancing the trade-off between achieving low training error and a low testing error, which could lead to overfitting if too high.
  - `loss`: Specified as 'squared_hinge' for its effectiveness with LinearSVC.
  - `max_iter` (Maximum iterations): Fixed at 1000, suitable for small to medium-sized datasets.
  - `penalty`: L2 norm, used for penalization.
  - `dual`: True, indicating that the problem is to be solved in its dual form.
  - `tol` (Tolerance for stopping criteria): Set very fine at $1e-5$ to ensure a precise convergence.

The graphs of the confusion matrix, ROC curve (which also shows the area under the curve) and the precision-recall curve follows:

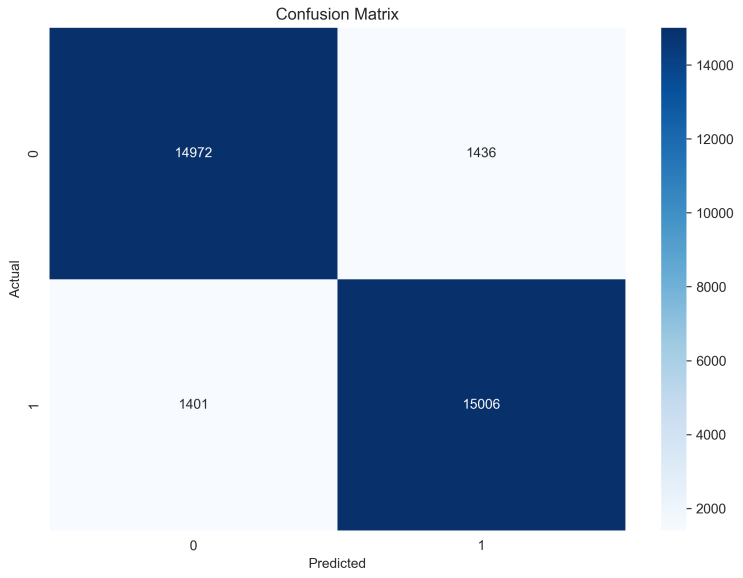


Figure 16: LinearSVC Confusion Matrix

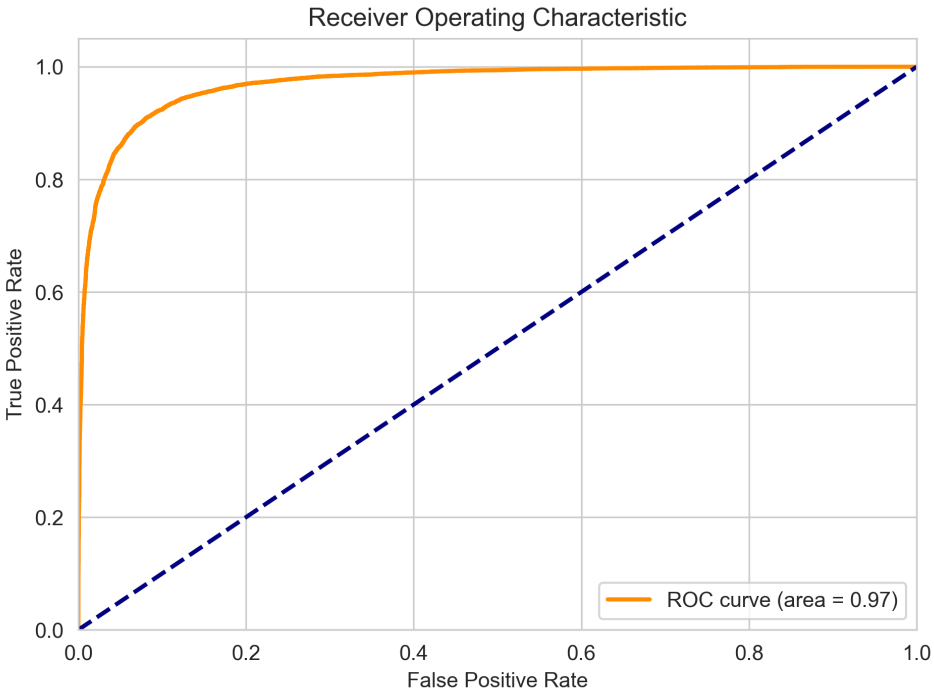


Figure 17: LinearSVC ROC Curve with AUC

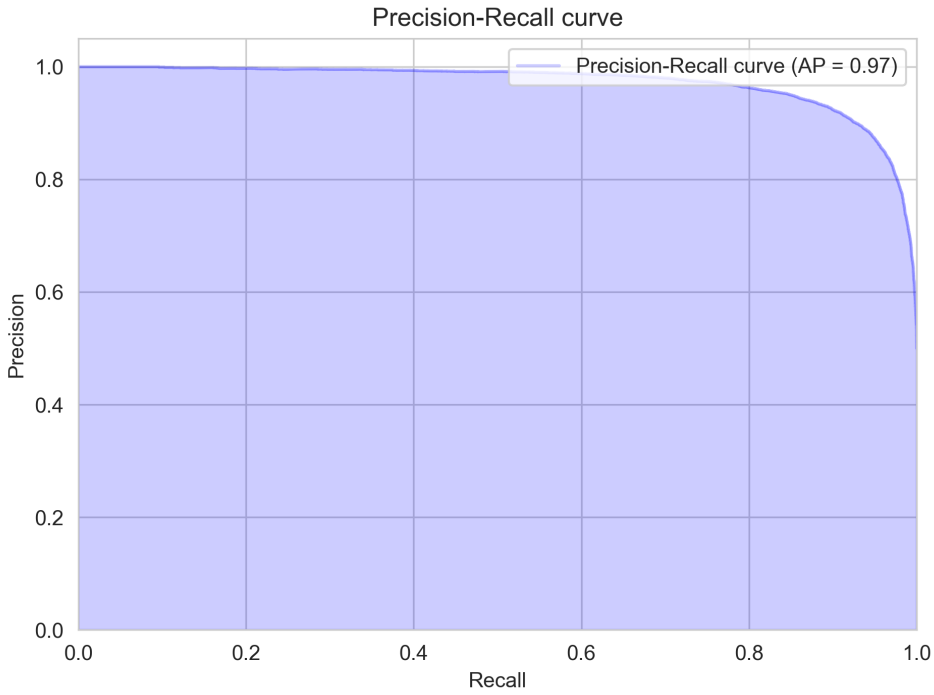


Figure 18: LinearSVC Precision-Recall curve

3.3 DistilBERT pre-trained Model

DistilBERT is a streamlined version of the more complex BERT (Bidirectional Encoder Representations from Transformers) model, designed for greater efficiency. This model retains approximately 97% of BERT's language understanding capabilities while being 4% smaller and 60% faster. It achieves this by reducing the number of Transformer layers from 12 to 6 and operating with about 66 million parameters, compared to BERT Base's 110 million. This reduction allows for quicker processing and less memory usage, making it suitable for environments with limited computational resources.

DistilBERT differs from its original version mainly in its size and processing speed, facilitated by a method known as ***knowledge distillation***. This process involves training DistilBERT to emulate BERT's behavior closely, which effectively preserves much of the original model's performance despite having fewer parameters. While DistilBERT offers significant advantages in terms of efficiency and cost, it generally delivers slightly lower performance compared to BERT, making it less ideal for tasks demanding the utmost precision.

In practical applications, particularly in sentiment analysis, DistilBERT serves as a highly effective tool due to its balance of performance and speed. It is especially beneficial for real-time analysis and applications on mobile devices or other platforms where rapid response times are crucial.

Describing and Training the DistilBERT Model

The model, `DistilBertForSequenceClassification`, is initialized from the pre-trained *distilbert-base-uncased* variant with two labels, signifying its utility for *binary* classification tasks. This initialization process leverages the pre-trained weights to better capture the nuances of language without the necessity of training from scratch.

Optimizer: The optimization of the model parameters is managed by AdamW, a variant of the Adam optimizer that specifically incorporates a decay of weight parameters. It is configured with a learning rate of 5×10^{-5} and an epsilon value of 10^{-8}, parameters that help in mitigating issues related to convergence and numerical stability.

Scheduler: A linear scheduler is employed to adjust the learning rate across training epochs. It is initialized with no warmup steps and spans the entirety of the training epochs calculated as the product of the number of batches in the training data loader and the total number of epochs. This scheduling aids in fine-tuning the learning rate, ensuring that it decreases linearly from the initial value to zero by the end of training, which optimizes convergence.

Training Procedure: Training is conducted on a GPU, utilizing Apple's Metal Performance Shaders (MPS) to expedite computations. Each training batch consists of input IDs, attention masks, and labels, which are processed to compute the model's loss. This loss is then backpropagated to update the model's weights. Gradient clipping is also applied with a maximum norm of 1.0 to prevent the exploding gradient problem, a common issue in training deep neural networks.

The average training loss is calculated at the end of each epoch, providing insight into the model's performance and convergence across iterations. This metric is crucial for monitoring training progress and making necessary adjustments to model parameters or training regimen.

This comprehensive setup not only ensures efficient training of the DistilBERT model but also maximizes its performance for downstream tasks such as sentiment analysis, where the ability to process and classify textual data swiftly and accurately is paramount.

3.3.1 Visual Performance of the DistilBERT model

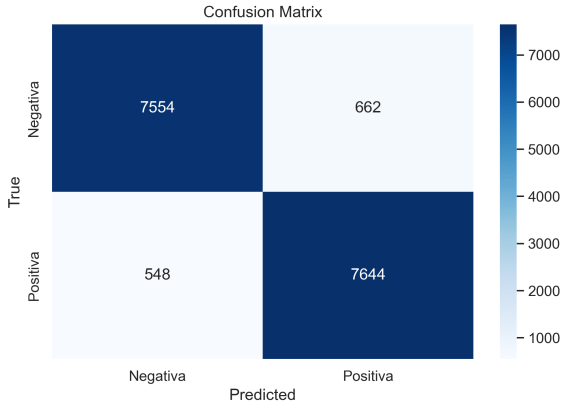


Figure 19: DistilBERT Confusion Matrix

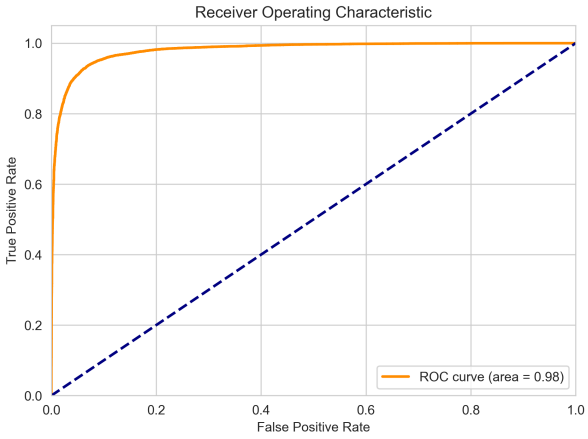


Figure 20: DistilBERT ROC Curve with AUC

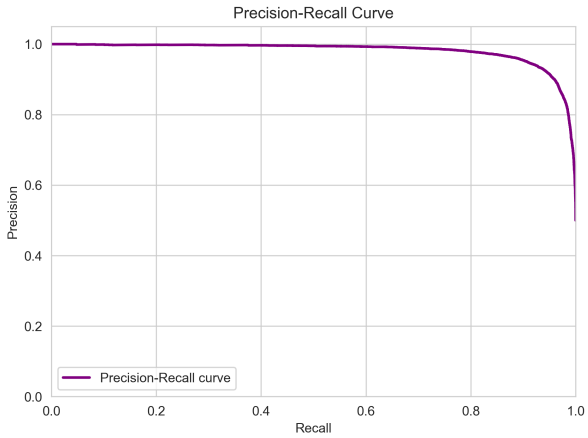


Figure 21: DistilBERT Precision-Recall curve

3.4 Deep Learning Model

It is our contention that this advanced deep learning model represents the pinnacle of this project. It is an advanced neural network that has been specifically tailored for the sentiment analysis of text data. This model incorporates several key components that are essential for processing the sequential and contextual nature of language.

The model begins with a *custom embedding layer*, which is pivotal for converting word indices into dense vectors. This layer uses a pre-trained and updatable weight matrix, enabling the model to generalize better across unseen data, thanks to the mask_zero=True parameter which ignores padding zeros in sequences.

Following the embedding, a *1D convolutional layer* with ReLU activation extracts local features while maintaining dimensionality due to the "same" padding. This layer is complemented by *batch normalization*, enhancing model stability and reducing internal covariate shift during training. The subsequent *MaxPooling1D* layer significantly reduces dimensionality, thus improving computational efficiency.

A critical feature of this network is the incorporation of ***two bidirectional LSTM layers***. The first of these layers captures long-term dependencies in both directions of the sequence, retaining connections across time steps by returning sequences. This setup is refined by a second bidirectional LSTM layer which further enhances the model's ability to learn complex patterns in the review data, crucial for accurate sentiment prediction.

To combat overfitting, the model includes *dropout layers* strategically placed after LSTM and dense layers. These layers randomly deactivate a fraction of neurons during training, forcing the network to learn more robust features. The *dense layer* preceding the output incorporates L2 regularization and Kaiming He initialization, optimizing activation outputs and contributing to the network's generalization abilities.

The final output layer employs a *sigmoid activation function*, tailored for binary classification tasks such as positive or negative sentiment detection. The model's compilation leverages the *RMSprop optimizer*, known for its efficacy with text data, and *binary crossentropy loss function* aligns with the binary nature of the sentiment classification. RMSprop optimizer is sometimes preferred over Adam in sentiment analysis for several specific reasons: firstly, RMSprop is designed to solve the diminishing learning rates problem by using a moving average of squared gradients to normalize the gradient itself. This mechanism helps in maintaining a more consistent learning rate during training, which can be crucial in dealing with the kinds of noisy or sparse gradient issues often found in text data. Additionally, RMSprop tends to be more robust to the choice of hyperparameters, particularly in scenarios where the dataset size is not very large, which is often the case in specific sentiment analysis tasks. This makes it a more straightforward and often more effective choice for maintaining stable and efficient convergence during the training of neural networks on textual data.

Overall, this neural network is optimized for high-dimensional text data, utilizing a sophisticated architecture that balances depth and complexity with computational efficiency and robustness against overfitting. Such characteristics make it highly suitable for the sentiment analysis of voluminous and nuanced Amazon review data, aiming to provide precise and reliable sentiment predictions.

The following section presents the visual design of the neural network created using the Graphviz library, a summary of the network compiled in textual format specifically by the libraries TensorFlow Keras, and the classification report of the model's performance in predicting the positive or negative sentiment of Amazon reviews.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 128)         1920128

 conv1d (Conv1D)             (None, None, 128)         49280

 batch_normalization (Batch  (None, None, 128)         512
 Normalization)

 max_pooling1d (MaxPooling1  (None, None, 128)         0
 D)

 bidirectional (Bidirection  (None, None, 256)         263168
 al)

 bidirectional_1 (Bidirecti  (None, 256)               394240
 onal)

 dense (Dense)               (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 2660353 (10.15 MB)
Trainable params: 2660097 (10.15 MB)
Non-trainable params: 256 (1.00 KB)
_________________________________________________________________
```

(a) Deep Neural Network Summary

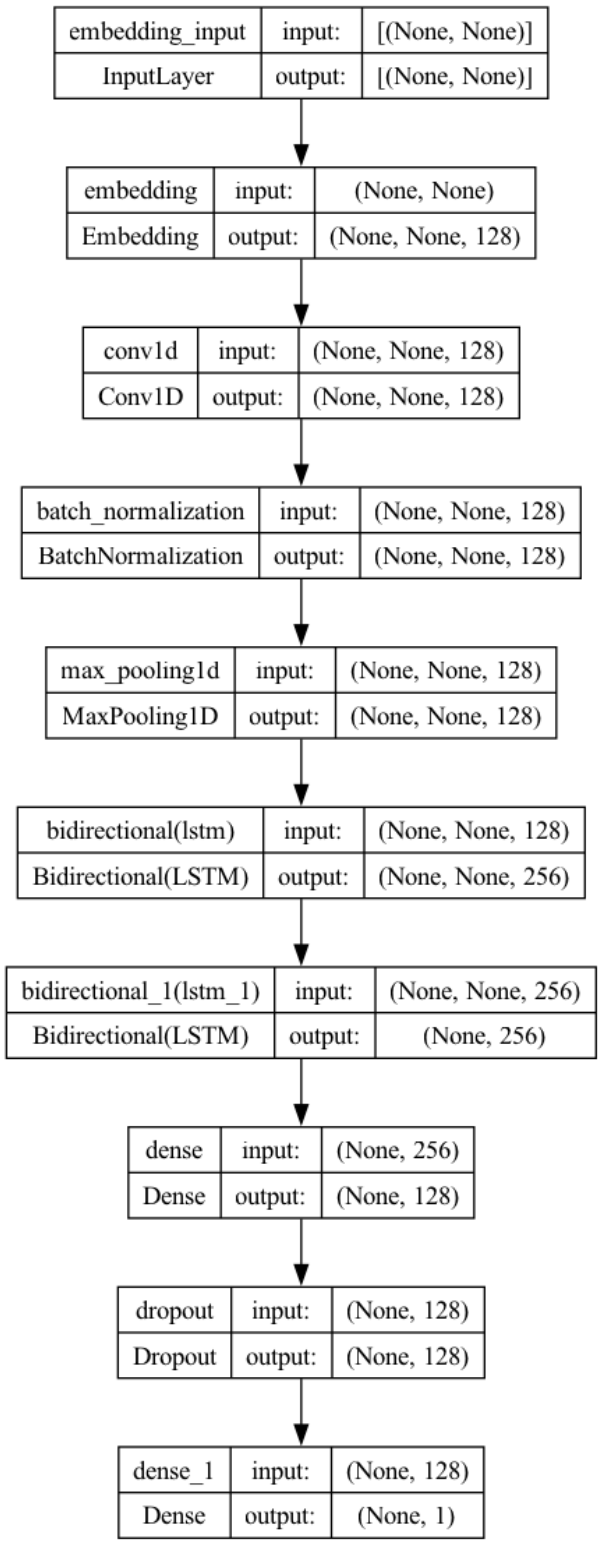
```
Classification Report:
              precision    recall  f1-score   support

     Negativa       0.93      0.92      0.93      8216
     Positiva       0.92      0.93      0.93      8192

     accuracy                           0.93     16408
    macro avg       0.93      0.93      0.93     16408
 weighted avg       0.93      0.93      0.93     16408

 Matrice di Confusione:
  [[7554  662]
  [ 548 7644]]
```

(b) Classification Report and Confusion Matrix



(c) Neural Network Architecture

Figure 22: Deep Neural Network Design

After a general description of the deep neural network given at the beginning of the section, we would like to describe, on the following page, the architecture of the network itself in more detail with the hyper-parameters used and their values.

3.4.1 Deep Neural Network Architecture:

- **Embedding Layer**
  - **Input Dimension**: Set to `20000`, covering most of the entire vocabulary.
  - **Output Dimension**: Defined by `128`, determining the size of the embedding vectors.
  - **Weights**: Initialized with a custom pre-trained `embedding matrix` to leverage prior knowledge and augment the sentiment classification
  - **Trainable**: Set to `True`, allowing the embeddings to update during training.
  - **Mask Zero**: Enables the layer to ignore padded zeros in sequences.
- **Convolutional Layer (Conv1D)**
  - **Filters**: Utilizes `128` convolutional filters to extract features from the embedded word representations.
  - **Kernel Size**: Each filter has a window of `three words` to capture the local context. A sufficient kernel size for Amazon reviews which are known to be not very lengthy, as we calculated in Data Understanding, the average length of amazon reviews are approximately 80 words.
  - **Activation**: Relies on the 'ReLU' function to introduce non-linearity.
  - **Kernel Initializer**: `GlorotUniform` initialization method, also known as Xavier optimization, to facilitate convergence. This initializer helps to prevent the vanishing or exploding gradients problem, which is crucial in deeper networks.
  - **Bias Initializer**: Starts with `zeros.`
  - **Padding**: Applies 'same' padding to maintain output length equal to input length.
  - **Strides**: Set to `1`, processing every element without skipping.
- **Batch Normalization Layer**
  - Adds a Batch Normalization layer between the Conv1D and MaxPooling1D layers to normalize the activations of the previous layer.
- **Pooling Layer (MaxPooling1D)**
  - **Pool Size**: Set to `2` to reduce the dimensionality by taking the maximum value over the window.
- **Recurrent Layer (Bidirectional LSTM)**
  - **Units**: Features `128 recurrent units`, capturing long-term dependencies from both directions of the sequences.
  - **Return Sequences**: The first LSTM returns full sequences for further temporal processing.
  - **Dropout**: Utilizes `30%` dropout to prevent overfitting by randomly setting that percentage of the input units to 0.
  - **Kernel Regularizer**: Employs L2 regularization with a lambda of `0.001`.

- **Dense Layer**
  - **Units**: Configured with `128` units, fully connected to the preceding layer.
  - **Activation**: Uses 'ReLU' for non-linear transformations.
  - **Kernel Initializer**: Applies `Kaiming He` normal optimization to initialize weights in a way that maintains the scale of the gradients.
- **Dropout Layer**
  - Further applies dropout with `30%` to reduce overfitting.
- **Output Layer**
  - **Activation**: Uses a `Sigmoid` function to output a probability, indicating the likelihood of the input belonging to the positive class.

3.4.2 Visual Representation of the Model Performance:

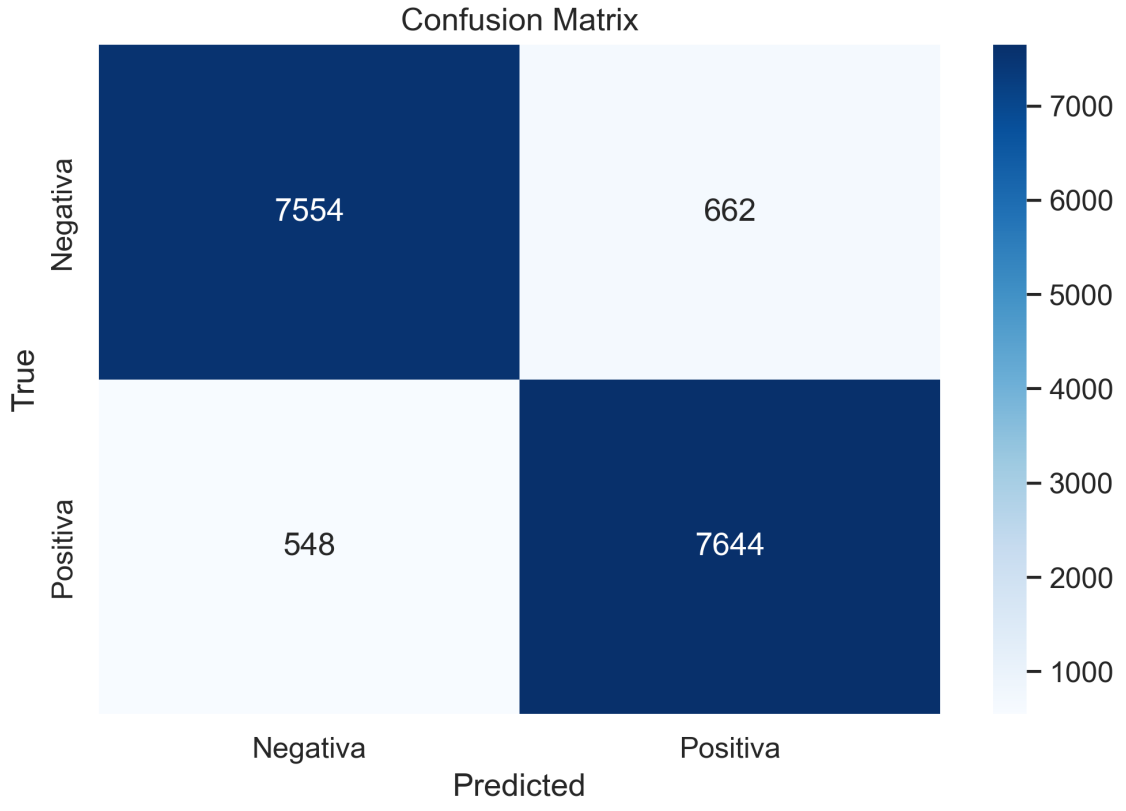


Figure 23: Confusion matrix

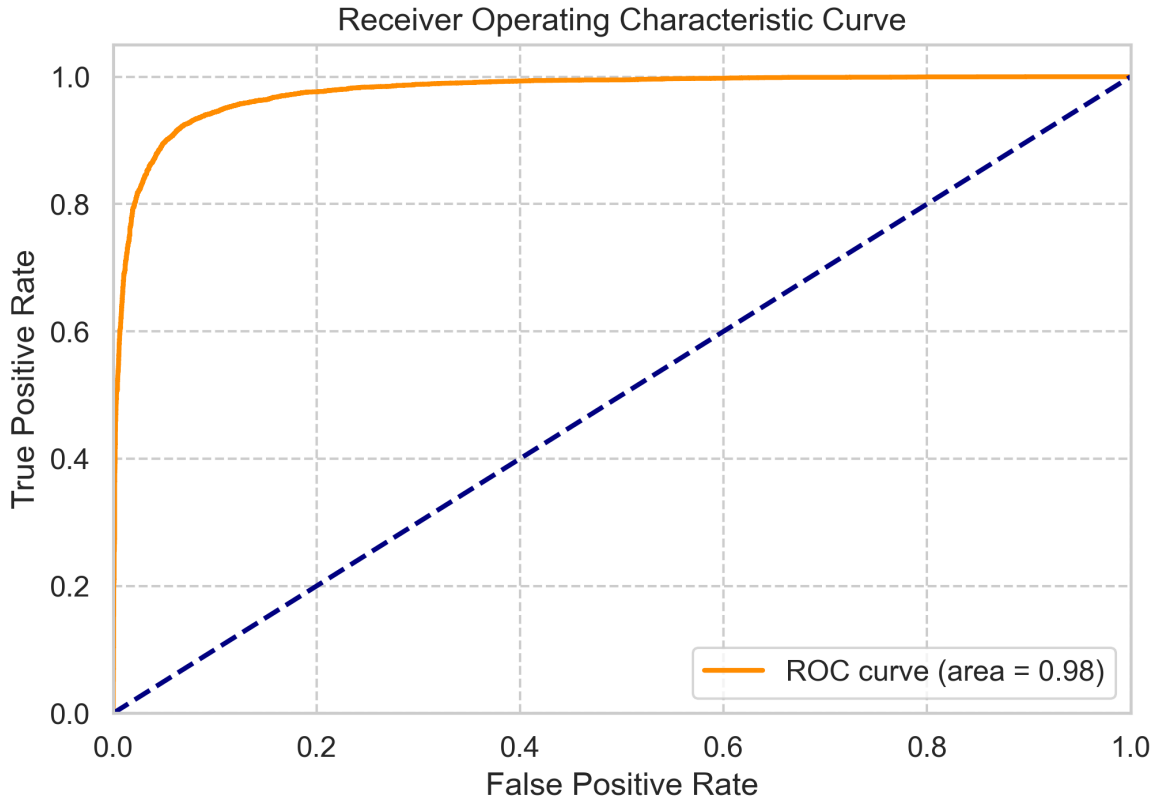


Figure 24: ROC Curve with AUC

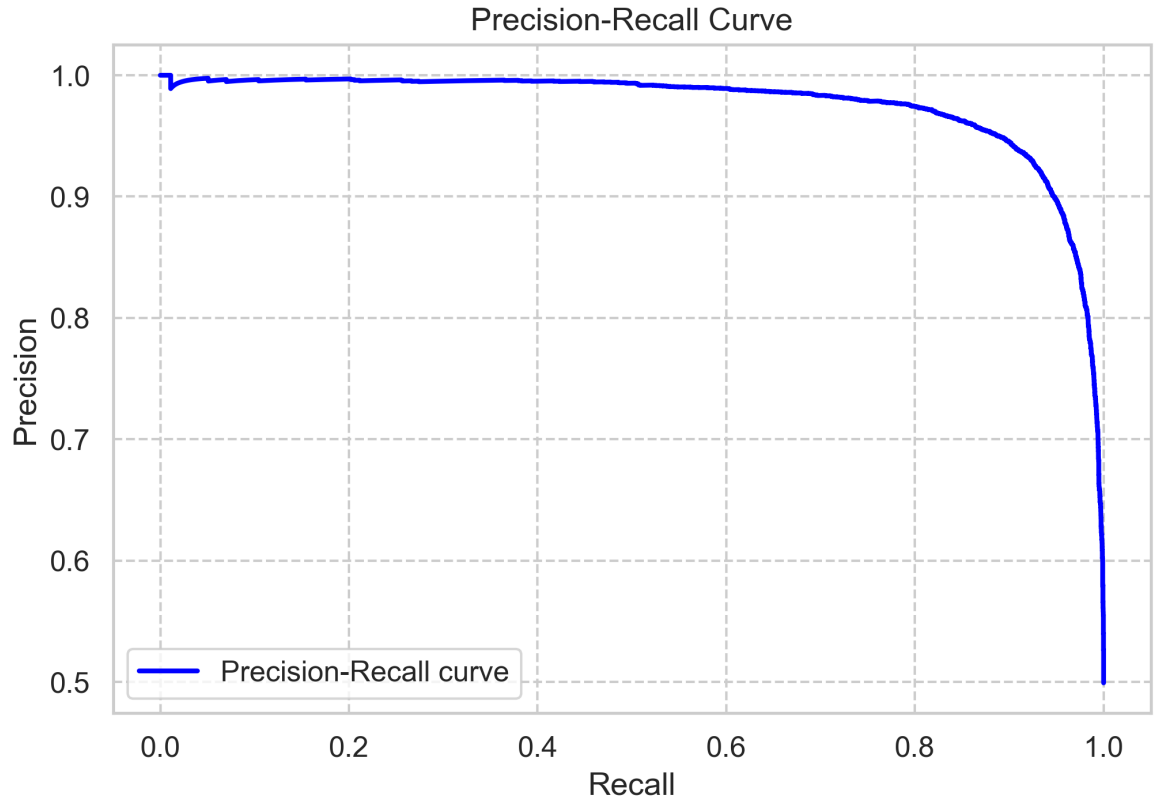


Figure 25: Precision-Recall Curve

Conclusion

This comprehensive report has systematically chronicled the journey from raw data exploration to sentiment analysis using a variety of advanced machine learning techniques, each contributing uniquely towards a nuanced understanding of consumer sentiments from Amazon product reviews.

The project began with an in-depth data understanding phase, where significant insights into the distribution, sentiment, and temporal trends of the reviews were garnered. This initial exploration was crucial in shaping the subsequent data preparation and cleaning phase, which was meticulously executed to ensure data quality and relevance through sophisticated text processing techniques. The implementation of various libraries and custom functions facilitated the removal of noise and normalization of text, setting a solid foundation for the analysis phase.

The sentiment analysis was tackled using three distinct approaches: Naïve Bayes, LinearSVC, and a cutting-edge Deep Learning model. Each model was carefully evaluated, revealing that while Naïve Bayes provided a robust baseline, LinearSVC improved upon it by handling feature interactions more effectively, thus enhancing classification performance. However, the real breakthrough was achieved with the deployment of the deep learning model, which integrated LSTM and convolutional layers to capture both long-term dependencies and local textual contexts with remarkable precision.

This deep learning model, despite being less complex in terms of parameter count compared to DistilBERT, demonstrated comparable performance, underscoring the efficiency and potential of optimized neural network architectures in processing natural language data. The model not only excelled in accuracy but also in its ability to discern subtle contextual nuances, making it a pivotal development in this project.