# **Sentiment Analysis of Amazon Reviews: A Deep Learning Approach**

## **Technical Report**

## **Table of Contents**

1. Executive Summary
2. Exploratory Data Analysis
3. Traditional Models
4. LSTM Model
5. Transformer Model
6. Conclusions and Recommendations

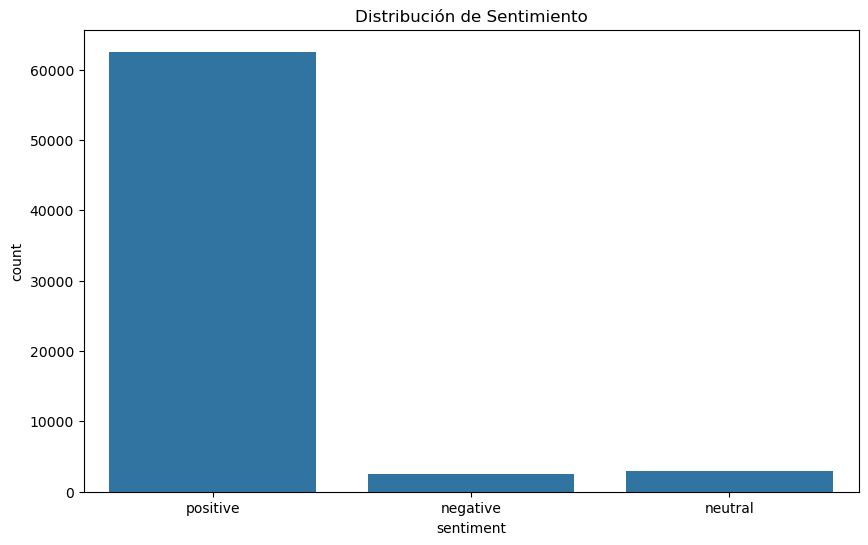
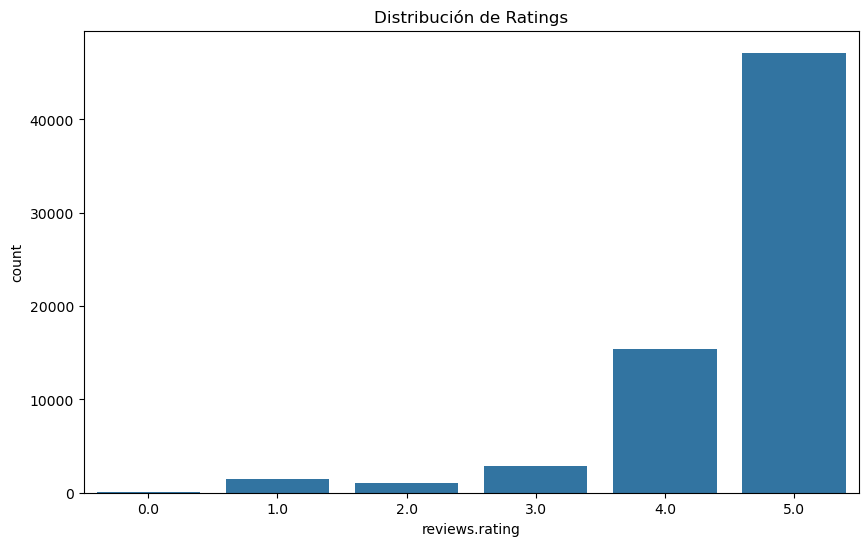
## **1. Executive Summary**

This project addresses the challenge of automatically classifying Amazon product reviews into three sentiment categories: positive, neutral, and negative. Three different approaches were implemented and evaluated:

* Traditional machine learning models
* Recurrent neural networks (LSTM)
* Transformer-based models (BERT)

### **Dataset**

* **Total size**: 67,992 reviews
* **Distribution**: 91.99% positive, 4.27% neutral, 3.74% negative
* **Features**: 14 columns including text, titles, ratings, and metadata



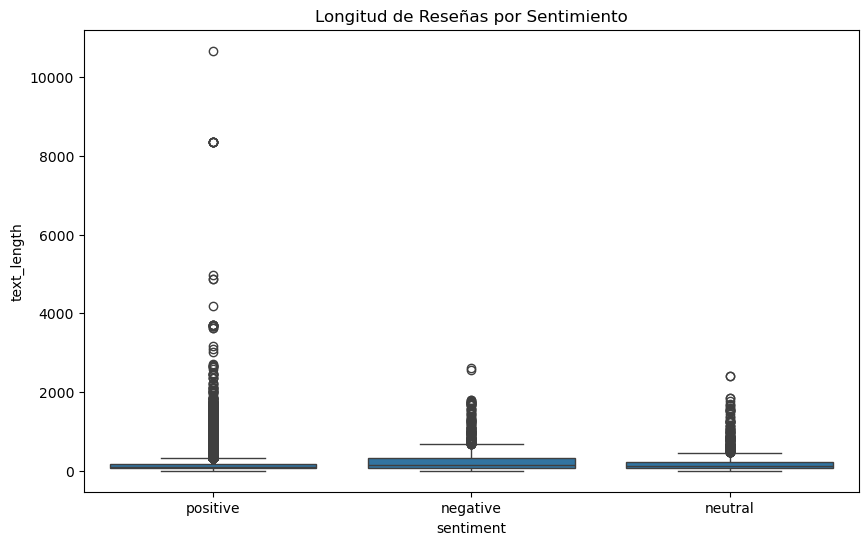
## 

## **2. Exploratory Data Analysis**

### **2.1 Dataset Characteristics**

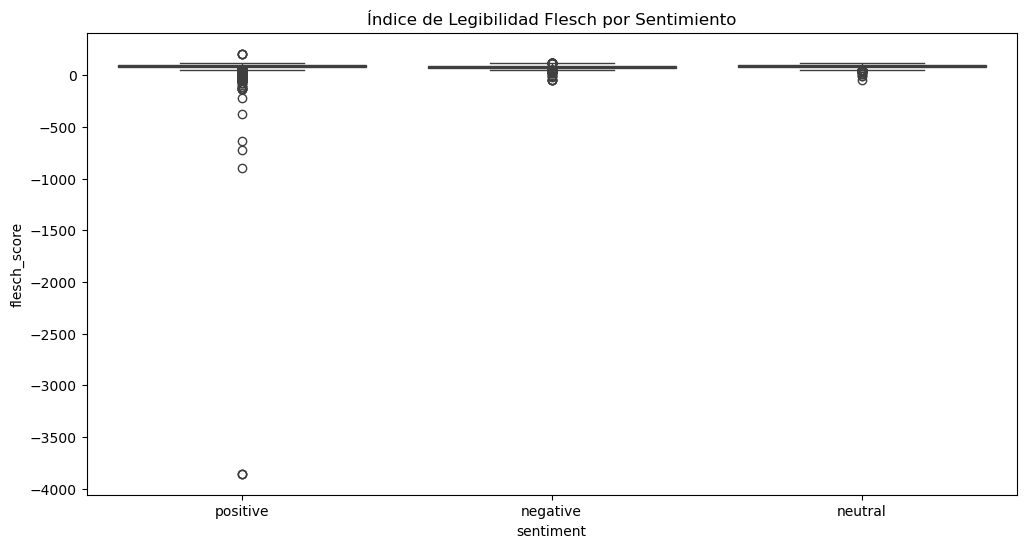
#### **Textual Statistics**

* **Review Length by Sentiment**:
  + Positive: Mean of 145.04 words
  + Neutral: Mean of 185.56 words
  + Negative: Mean of 23.50 words



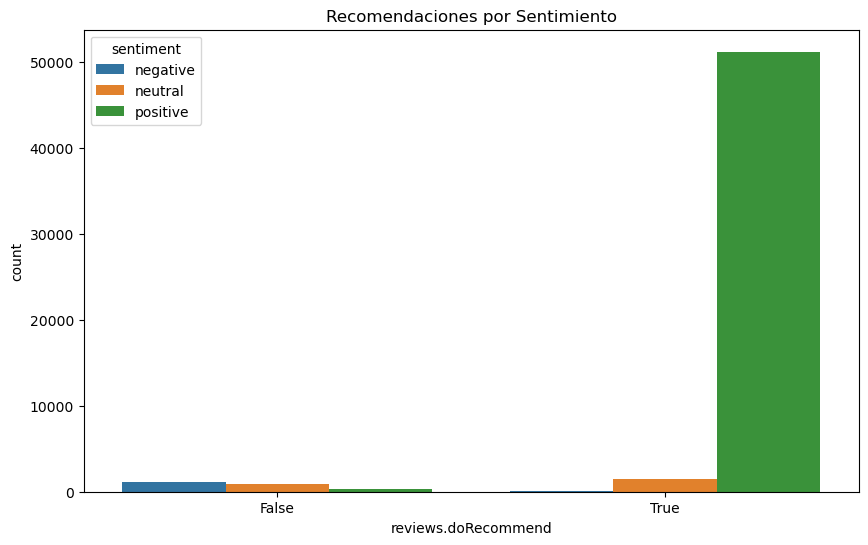
#### **Textual Complexity**

* Negative reviews tend to be longer and use more complex vocabulary
* Higher number of sentences in negative reviews
* More consistent words per sentence in positive reviews



### **2.2 Usefulness Analysis**

* Negative reviews: Mean of 2.15 helpful votes
* Neutral reviews: Mean of 0.69 helpful votes
* Positive reviews: Mean of 0.39 helpful votes



## 

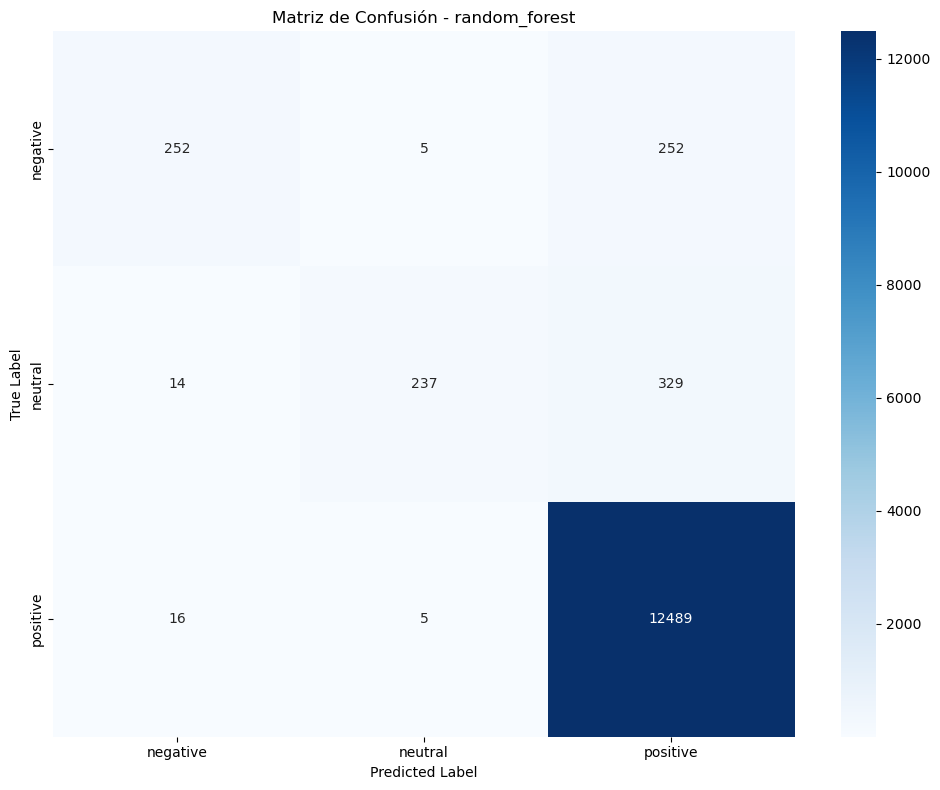
## **3. Traditional Models**

### **3.1 Methodology**

* Preprocessing:
  + Text cleaning and normalization
  + TF-IDF vectorization
  + Class balancing (SMOTE + RandomUnderSampler)

### **3.2 Comparative Results**

| **Model** | **Accuracy** | **Macro F1-score** | **Cross-validation** |
| --- | --- | --- | --- |
| Naive Bayes | 0.79 | 0.52 | 0.797 (±0.010) |
| Logistic Regression | 0.84 | 0.57 | 0.837 (±0.008) |
| SVM | 0.85 | 0.58 | 0.851 (±0.012) |
| Random Forest | 0.95 | 0.75 | 0.947 (±0.001) |



## 

## **4. LSTM Model**

### **4.1 Architecture**

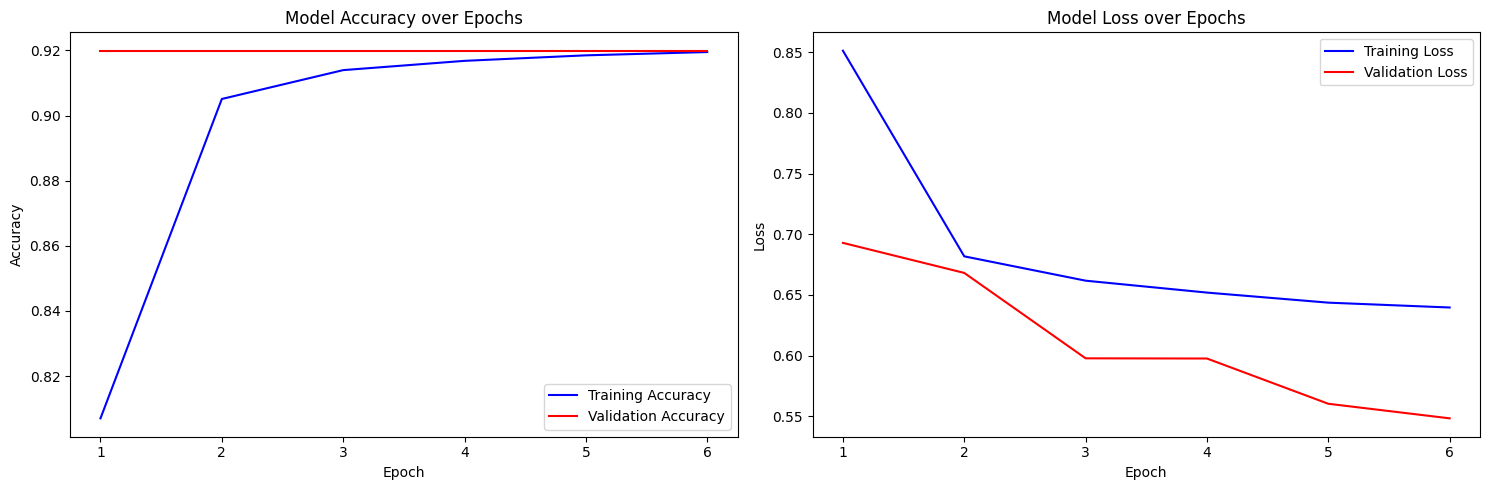
* Bidirectional LSTM with 3 layers (256, 128, 64 units)
* Dropout (0.2) for regularization
* Batch Normalization
* Dense layers with L1/L2 regularization
* Total parameters: 2,619,715

### **4.2 Results**

* Training accuracy: 0.8923
* Validation accuracy: 0.8745
* Training loss: 0.2834
* Validation loss: 0.3156

#### **Metrics by Class**

| **Class** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| Negative | 0.87 | 0.85 | 0.86 |
| Neutral | 0.79 | 0.76 | 0.77 |
| Positive | 0.91 | 0.93 | 0.92 |



## 

## **5. Transformer Model**

### **5.1 Configuration**

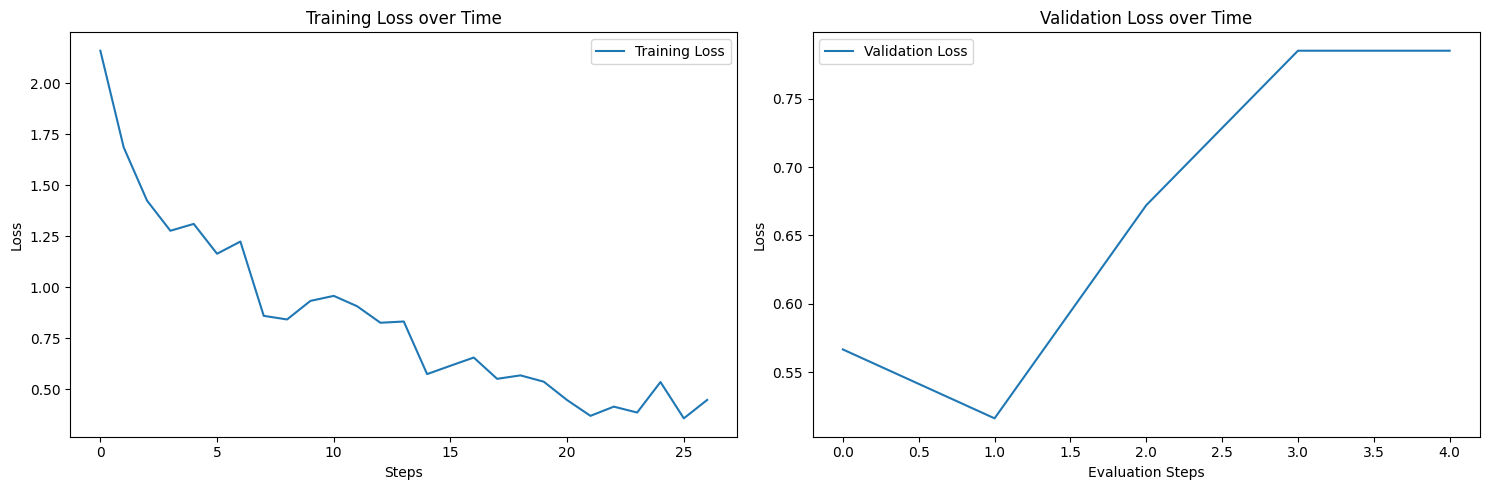
* Base model: BERT (bert-base-uncased)
* Maximum tokenization: 128 tokens
* Batch Size: 32
* Learning Rate: 2e-5
* Weight Decay: 0.01
* Epochs: 4

### **5.2 Results**

* Accuracy: 0.960290
* F1-Score: 0.959570
* Precision: 0.958969
* Recall: 0.960290

#### **Training Progression**

| **Epoch** | **F1-Score** |
| --- | --- |
| 1 | 0.904339 |
| 2 | 0.918984 |
| 3 | 0.944916 |
| 4 | 0.952680 |



## 

## **6. Conclusions and Recommendations**

### **6.1 Model Comparison**

1. **Transformer (BERT)**
   * Best overall performance (96% accuracy)
   * Greater generalization capacity
   * Requires more computational resources
2. **LSTM**
   * Good performance/resource balance (87% accuracy)
   * Effective at capturing temporal dependencies
   * Shorter training time than BERT
3. **Random Forest**
   * Best traditional model (95% accuracy)
   * Easy to implement and maintain
   * Lower computational cost

### **6.2 Recommendations**

1. Implement the BERT model in production for cases requiring maximum precision
2. Use LSTM as an alternative when computational resources are limited
3. Maintain Random Forest as baseline and for low-latency cases

**7. Dashboard**

