Machine Learning Project Report

Prediction of Amazon Product Ratings

Sentiment Analysis & Regression

Authors:
Gabriel & Jeffrey

Specialization:
Data Science

December 12, 2025

Contents

1 Business Case and Project Context

1.1 Problem Definition

E-commerce has experienced exponential growth, particularly with the giant Amazon, which holds a dominant position in the market. In this ecosystem, trust is paramount, embodied primarily by the rating system and customer reviews.

The strategic importance of this project is based on the proven link between online reputation and business performance. According to a major study by the ***Spiegel Research Center*** (Northwestern University), the presence of customer reviews can increase a product's conversion rate by up to **270%**. Furthermore, the study demonstrates that consumer sensitivity to ratings is correlated with price: for high-value products (such as electronics in our dataset), the impact of reviews on the purchasing decision increases to **380%**. Consequently, predicting the potential rating of a product based on its characteristics and early textual feedback is a strategic imperative to optimize pricing positioning and inventory management.

The initial challenge lies in the difficulty for sellers and the platform to understand, at scale, which factors truly influence a product's final rating. A product may have an attractive price but a mediocre rating, or vise versa.

The Main Objective is to develop a predictive model capable of estimating a product's rating (on a scale of 1 to 5) based on its intrinsic characteristics (category, price, discount) and extrinsic characteristics (textual content of user reviews).

1.2 Link to Specialization (Data Science)

This project lies at the heart of Data Science applied to business. It involves the processing of heterogeneous data, requiring the fusion of structured data (numerical) and unstructured data (text), a classic Big Data challenge.

Furthermore, it heavily utilizes Sentiment Analysis (NLP) to extract subjective information from raw text, essential for understanding customer sentiments. Finally, the core of the solution rests on Supervised Learning (specifically regression models) to provide decision support. Such a model would allow for the detection of anomalies, such as a product that should be highly rated according to its specs but is not, or for example, the estimation of the potential success of a new product before it has accumulated thousands of reviews.

2 Dataset Description and Source

2.1 Data Source

The data originates from a public Amazon dataset retrieved from Kaggle. It consists of product data comprising commercial metadata and user feedback.

2.2 Data Structure and Dictionary

The initial dataset contains several thousand entries. The analysis focuses on three types of variables.

First, the **Identifier Variables** include the `product_id`, which uniquely identifies the item, as well as the `user_id` and `review_id`.

Second, the `Numerical Variables` serve as potential features for our model. These include the `discounted_price` (actual selling price), the `actual_price` (list price before discount), and the `discount_percentage`. The `rating_count` is also crucial, as it indicates the total volume of reviews, serving as a proxy for product popularity.

Finally, the **Textual and Categorical Variables** provide context. The `category` field offers a hierarchy (e.g. "Computers&Accessories|Accessories..."), while `review_content` and `review_title` contain the raw user feedback. The target variable for our prediction is the `rating`, a floating-point value between 1.0 and 5.0 representing the average user score.

3 Exploratory Data Analysis (EDA)

This section details our observations on the `amazon.csv` dataset, which contains 1,464 products after cleaning.

3.1 Data Cleaning and Quality

Before analysis, a preprocessing step was necessary to make the data usable. The numerical variables `discounted_price`, `actual_price`, and `rating_count` contained symbols (₹, commas) which were removed. The dataset proved to be remarkably clean, with only 2 missing values detected and imputed in `rating_count`. Finally, we identified and corrected inconsistencies in the target column `rating`, removing non-numeric characters.

3.2 Univariate Analysis

We first examined the distribution of individual variables.

A. The Target Variable: Rating

As illustrated in Figure 1, the distribution of ratings is not symmetrical and is positioned towards higher scores.

The mean and median are both **4.1 / 5**. We observe a strong positive skew, with over 75% of products having a rating greater than or equal to 4.0. Ratings below 3.0 are extremely rare in this dataset. This class imbalance presents a challenge, as the model risks systematically predicting a "good grade" without effectively learning the characteristics of lower-rated products.

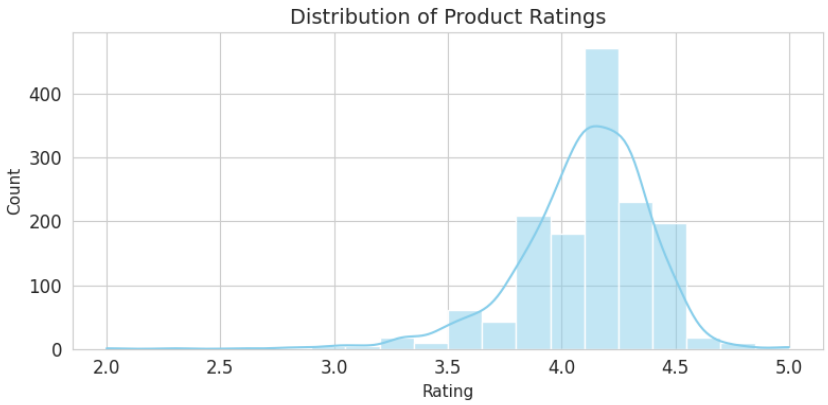

Figure 1: Distribution of Product Ratings

B. Prices and Discounts

The distribution of prices (`discounted_price`) is visible in Figure 2.

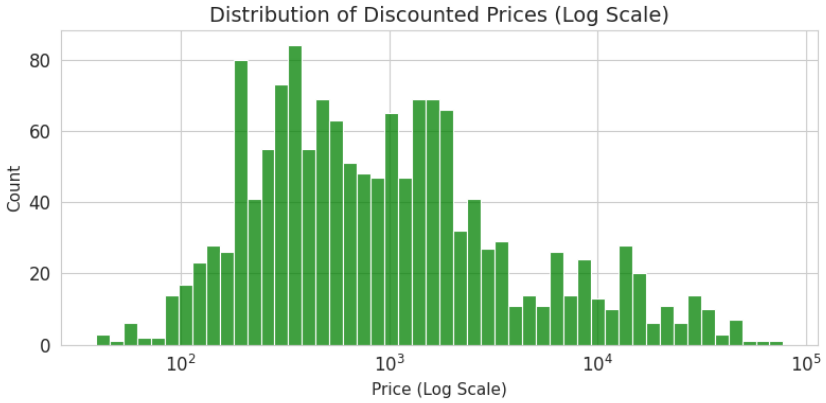

Figure 2: Distribution of Discounted Prices (Log Scale)

The median price is **₹799** (approx. **9€**). Although the majority of products are inexpensive accessories (less than ₹1000), some electronic products reach ₹77,990. Consequently, the use of a logarithmic scale will be indispensable for modeling. Regarding promotions, the average discount percentage is notably high at 47.7%.

3.3 Bivariate and Multivariate Analysis

A. Impact of Promotions (Discount vs Rating)

We tested the hypothesis that a strong promotion would positively influence the rating. Figure 3 crosses the discount percentage with the final rating.

We observe a quasi-null correlation: the median discount remains stable (around 50-55%) regardless of the rating obtained. Interestingly, extreme discounts ($> 70\%$) are often associated with average ratings (3 to 4 stars). We can conclude that while promotion is an important factor

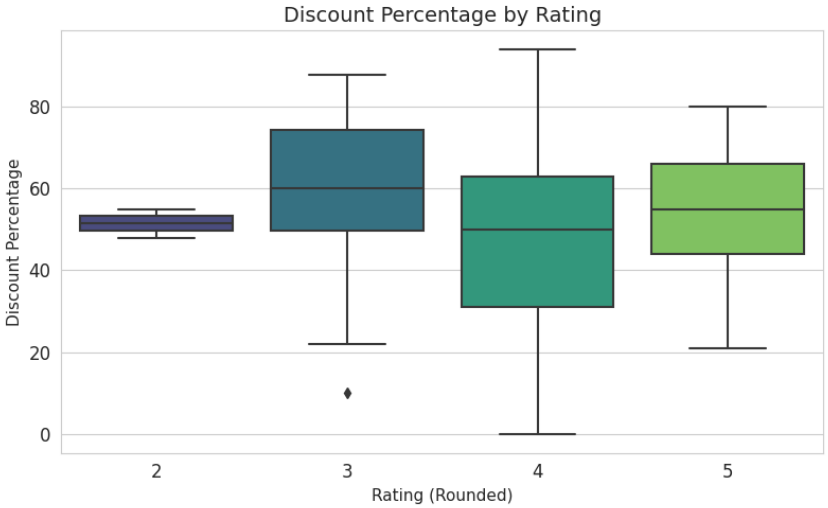

Figure 3: Distribution of Discounts by Rating Level

for sales volume, it is not sufficient to "buy" customer satisfaction in the long term.

B. Price vs Rating Relationship

Figure 4 tests the hypothesis "the more expensive, the better". The scatter plot shows a dispersed cloud with no obvious linear trend.

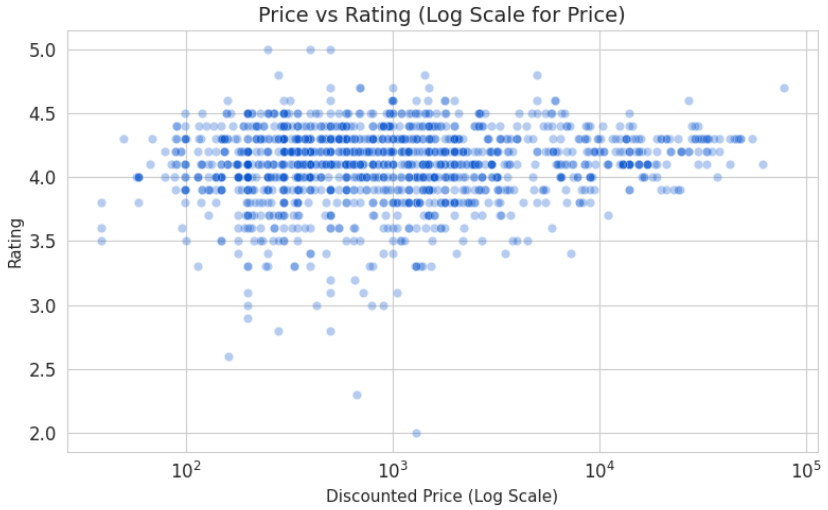

Figure 4: Price vs Rating Scatter Plot

The correlation coefficient is close to 0. We observe low-priced products (₹200-500) with excellent ratings (4.5+), as well as expensive products with average ratings. This confirms that price alone is not a sufficient predictor of perceived quality.

C. Impact of Popularity (Categories)

Figure 5 highlights the categories dominating the dataset. The Electronics category (and its sub-categories like Cables, Accessories) accumulates the largest volume of reviews, followed by the Home&Kitchen category which also shows strong traction.

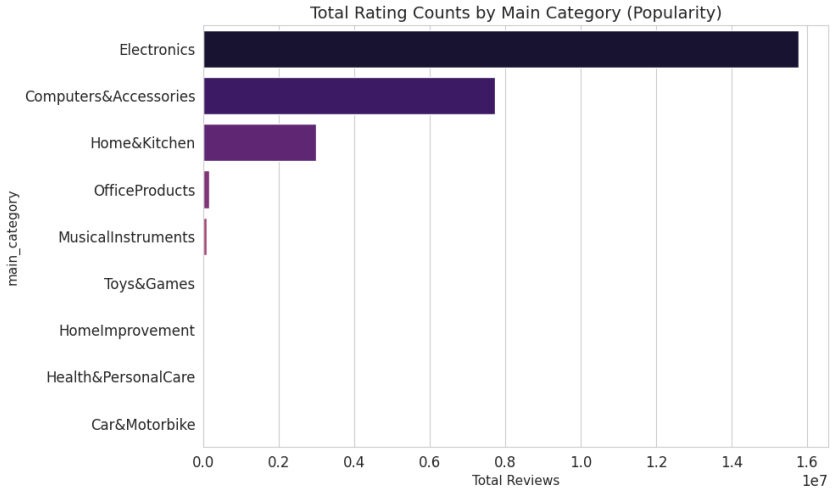

Figure 5: Total Rating Counts by Main Category

D. Correlation Matrix

Figure 6 displays the linear relationships between variables. We note that `discounted_price` and `actual_price` are highly correlated (0.96), implying we must keep only one to avoid multicollinearity. Furthermore, the correlation between `rating` and numerical variables is very weak (< 0.15), suggesting that the predictive power likely lies in the text (NLP) rather than solely in numerical data.

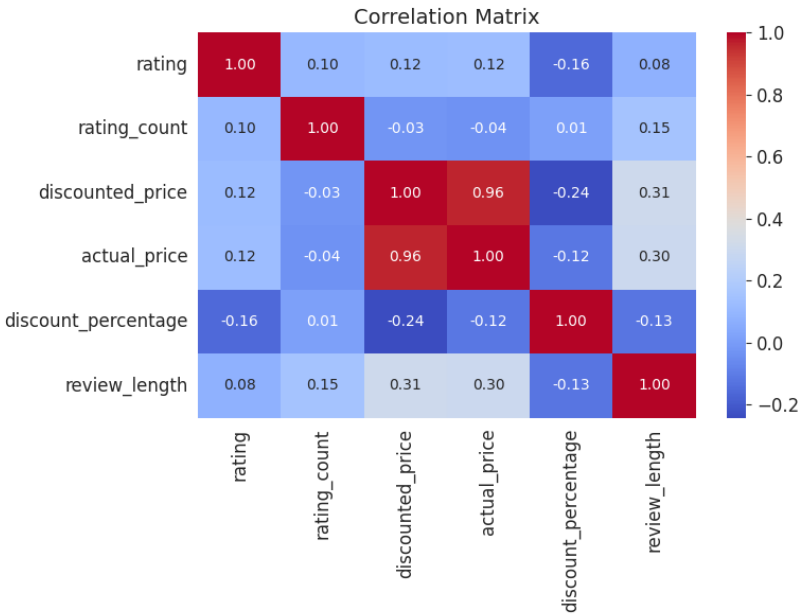

Figure 6: Correlation Matrix

3.4 Preliminary Textual Analysis (NLP)

Since numerical variables explain little of the rating variance, we analyzed the content of reviews (`review_content`). In **Positive Reviews (>4 stars)**, terms like "cable", "charging", "good quality", and "value for money" dominate, suggesting functionality is the main driver of satisfaction. Conversely, **Negative Reviews (<3 stars)** contain specific terms like "working" (often in the context "stopped working"), "waste", and "return".

Regarding review length, the average is 1,393 characters. We noticed that very negative reviews tend to be longer and more detailed, reflecting the customer's effort to justify their dissatisfaction.

3.5 EDA Conclusion

The exploration highlights two key constraints for modeling. First, the Imbalance means the model must be robust against the over-representation of positive ratings. Second, the Importance of NLP is undeniable: the richness of the information lies in the text. Extracting textual features (Sentiment Score) will be the decisive step to improve prediction.

4 Data Preprocessing and Feature Engineering

4.1 Data Cleaning and Standardization

Although the dataset is relatively clean, a rigorous cleaning process was necessary to ensure model robustness. Regarding duplicates, we identified entries sharing the same `product_id`. However, since a single product can have multiple variations, we chose to remove only strictly identical entries. For missing values, we employed median imputation for numerical variables such as `rating_count`, as the median is more robust to outliers than the mean. Missing text entries in `review_content` were replaced with empty strings to prevent errors in the NLP pipeline.

4.2 Natural Language Processing (NLP) Pipeline

The `review_content` variable captures the customer's sentiment. To exploit this unstructured data, we applied a standardized NLP pipeline. First, we performed text normalization, which involved converting text to lowercase, removing punctuation and special characters, and eliminating English stop-words (e.g., "the", "is", "at") that add noise without informative value. Subsequently, we applied Lemmatization using the `WordNetLemmatizer`. Unlike stemming, this technique transforms words into their canonical form (e.g., converting "better" to "good" or "running" to "run"), thereby preserving the semantic meaning of the vocabulary.

4.3 Feature Engineering

We generated several new variables to enrich the model.

A. Sentiment Analysis: Instead of a complex classification approach based on specific vocabulary levels, we utilized the **VADER** (Valence Aware Dictionary and sEntiment Reasoner) library to condense information. We selected VADER because it is specifically optimized for social media text and customer reviews, capable of interpreting amplifiers (e.g., "very good"), negations (e.g., "not bad"), and emojis. This process resulted in the creation of the `sentiment_score`, a continuous composite score ranging from -1 (very negative) to +1 (very positive).

B. Textual Statistical Variables: We hypothesized that the form of a review also betrays user satisfaction. We calculated `review_len` (total number of characters), noting that very long reviews often correlate with extreme experiences (either excellent or terrible), and `word_count` (number of significant words after cleaning).

C. Category Extraction: The original `category` column is hierarchical (e.g., "Computers&Accessories|Accessories..."). We extracted the **Root Category** (the first term before the

separator |) to reduce cardinality. This categorical variable was then transformed via **One-Hot Encoding** to be processed by machine learning algorithms.

4.4 Transformation and Scaling

The exploratory analysis revealed highly disparate distributions. For the `discounted_price` variable, we applied a **Logarithmic Transformation** to dampen the impact of extreme values. For other numerical variables (`rating_count`, `sentiment_score`), we applied **Standardization** (`StandardScaler`) to achieve a mean of 0 and a standard deviation of 1. This step is crucial for algorithms based on distance or gradient descent.

4.5 Split Strategy (Train/Test)

Given the strong class imbalance observed during EDA (with a majority of ratings > 4), a simple random split would be risky, potentially resulting in a test set containing no "bad ratings." To address this, we employed a **Stratified Shuffle Split**. We divided the data into 80% training and 20% testing sets, stratifying on the `rating` variable. This guarantees that the proportion of 1, 2, 3, 4, and 5-star ratings remains identical across both datasets, ensuring a fair and representative model evaluation.

5 Model Presentation and Modeling Strategy

After transforming our raw data into relevant feature vectors, we proceed to the modeling phase. Since our problem involves predicting a continuous variable (`rating`) based on mixed variables (numerical and textual), we operate within a **Supervised Regression** framework. To ensure the robustness of our results, we adopted an incremental approach, starting with a simple model (Baseline) and progressing towards complex ensemble methods (Bagging and Boosting).

5.1 Evaluation Protocol and Metrics

Before presenting the algorithms, it is imperative to define how we will judge their performance. We selected three complementary metrics.

RMSE (Root Mean Square Error):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1}$$

We chose RMSE as our primary optimization metric because it strongly penalizes large errors. In our context, predicting 5 stars for a product that is worth 1 is significantly more detrimental than predicting 4.2 for a 4.0 product.

MAE (Mean Absolute Error):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

This metric is more interpretable. An MAE of 0.2 means that, on average, our prediction deviates by 0.2 stars from reality.

Coefficient of Determination (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3}$$

(*where y_i are the observed values, $\hat{y}_i$ are the predicted values, and $\bar{y}$ is the mean of the observed data*).
It measures the proportion of the target variable's variance explained by the model. An R^2 close to 1 indicates a perfect model, while an R^2 close to 0 indicates the model performs no better than a simple average.

5.2 Model 1: Multiple Linear Regression (Baseline)

We initiated our analysis with a **Multiple Linear Regression** (OLS - Ordinary Least Squares). This model serves as a reference point (baseline). If complex models (such as neural networks) do not significantly outperform this simple model, their computational cost is not justified.

Mathematical Principle: The model seeks to model the linear relationship between the explanatory variables X (price, sentiment, categories) and the target y (rating) using the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p + \varepsilon \tag{4}$$

where the coefficients β are estimated by minimizing the sum of squared residuals.

Expected Limitations: The EDA (Part 3) demonstrated that the relationship between price and rating is not linear. Furthermore, this model struggles to handle complex interactions between variables (e.g., a high price is accepted only if the sentiment is very positive). We therefore anticipate high bias (underfitting).

5.3 Model 2: Random Forest Regressor (Bagging)

To capture non-linearities, we selected the **Random Forest** algorithm, an ensemble method based on **Bagging** (Bootstrap Aggregating).

Operating Principle: The model constructs a multitude of independent decision trees $(T_1, T_2, ..., T_N)$ during training. Each tree is trained on a random subset of the data (sampling with replacement), and at each node, only a random selection of features is considered for the split. The final prediction is the average of the predictions from all individual trees:

$$\hat{y} = \frac{1}{N} \sum_{k=1}^{N} f_k(x) \tag{5}$$

Justification for Choice: This model offers Robustness: unlike a single decision tree which tends to overfit, the random forest reduces variance through averaging. It also handles Heterogeneous Data naturally, mixing continuous variables (price) and binary variables (One-Hot categories). Finally, it allows us to extract **Feature Importance**, helping us determine relative variable impact.

5.4 Model 3: XGBoost (Extreme Gradient Boosting)

Finally, we implemented **XGBoost**, considered the state-of-the-art for structured tabular data. It is a **Boosting** method.

Operating Principle: Unlike Random Forest, which builds trees in parallel, XGBoost builds trees sequentially. Each new tree attempts to correct the errors (residuals) made by the previous trees.

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i) \tag{6}$$

where f_t is the new tree learning to predict the residual $y_i - \hat{y}_i^{(t-1)}$, and η is the learning rate.

Specific Advantages: XGBoost integrates **L1 (Lasso)** and **L2 (Ridge) Regularization** into its objective function, limiting overfitting even with limited data. It also automatically handles missing Values during tree construction. By directly minimizing the loss function via gradient descent, it often achieves superior precision compared to Random Forest.

5.5 Cross-Validation Strategy

To prevent our results from depending on a "lucky" or "unlucky" data split (especially with a dataset of roughly 1,500 rows), we do not evaluate models on a simple Train/Test split. Instead, we use **K-Fold Cross-Validation** (with $K = 5$). The training set is divided into 5 folds. The model is trained 5 times, each time using 4 parts for training and 1 part for validation. The final performance is the average of the 5 scores, ensuring a reliable estimate of the model's generalization capability on future data.

6 Comparative Analysis of Results

After training our three models (Linear Regression, Random Forest, and Gradient Boosting) on real data, we analyzed their performance to identify the best approach.

6.1 Performance Summary Table

The models were evaluated on the test set (20% of the data). To evaluate our models, we used three key metrics we previously mentioned (in the 5.1). The consolidated results are presented below:

Model	**MAE** (Abs. Error)	**RMSE** (Sq. Error)	R^2 **Score**
Linear Regression	0.207	0.261	0.12
Random Forest	0.190	0.250	0.19
Gradient Boosting	**0.180**	**0.248**	**0.21**

Table 1: Performance Metrics by Model

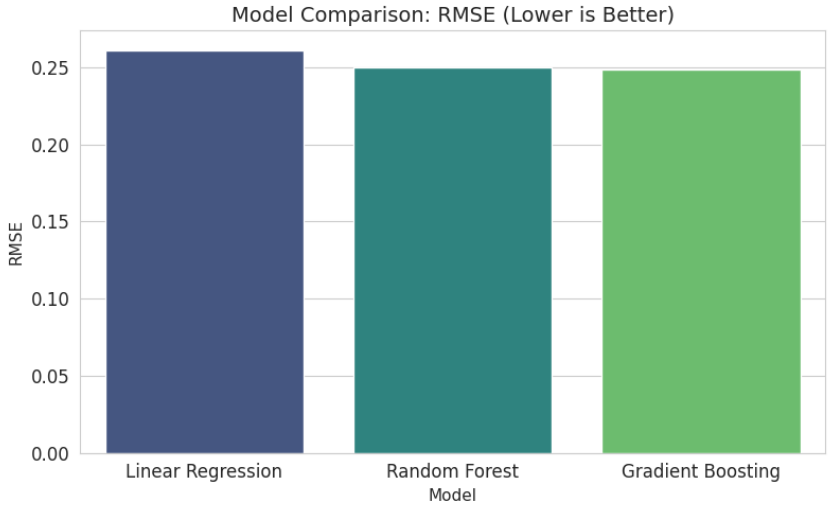

Figure 7: Model Comparison: RMSE (Lower is Better)

6.2 Interpretation of Metrics

As shown in Table 1 and Figure 7, the **Gradient Boosting** model slightly outperforms Random Forest and significantly outperforms Linear Regression across all metrics. With an **MAE of 0.18**, the average error is less than one-fifth of a star. For instance, if the actual rating is 4.2, the model will typically predict between 4.02 and 4.38.

However, the R^2 **is low (0.21)**. This score must be interpreted in the context of the data distribution (seen in Section 3). The vast majority of products have a rating between 3.8 and 4.5,

leaving little variance to explain. Furthermore, predicting the subtle difference between a 4.1 and a 4.3 often depends on subjective factors not present in the data (customer mood, specific delivery delays, etc.), creating a "glass ceiling" for performance.

6.3 Feature Importance Analysis

Figure 8 reveals the hierarchy of factors influencing the rating.

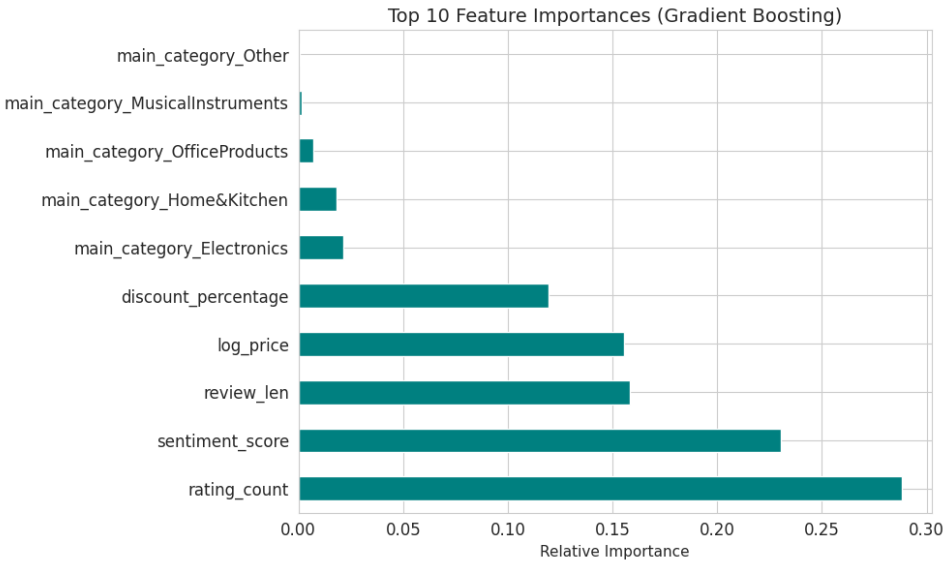

Figure 8: Top 10 Feature Importances (Gradient Boosting)

We observe that the most discriminating variable is the **Number of Reviews (29%)**. This implies that a product with thousands of reviews often has a constant and reliable rating, whereas products with few reviews exhibit much higher variance.

Furthermore, the NLP analysis via the **Sentiment Score (23%)** is the second most important factor, confirming that the algorithm successfully interprets the text. Additionally, **Review Length (16%)** is a strong indicator; very short reviews (e.g., "Good product") are often associated with average/high ratings, while long blocks of text often signal either passionate customers (5/5) or furious ones (1/5).

In conclusion, price dictates less customer satisfaction than expected. The analysis of the Gradient Boosting model contradicts the initial hypothesis that price is the sole decision factor.

6.4 Error Analysis (Actual vs Predicted)

Figure 9 visualizes the model's predictions against reality.

We observe that the points form a dense cluster around 4.0, indicating the model rarely predicts below 3.5. Conversely, the model struggles to predict extreme ratings (1.0 or 5.0). It tends to predict conservative values around the mean (4.0 - 4.2).

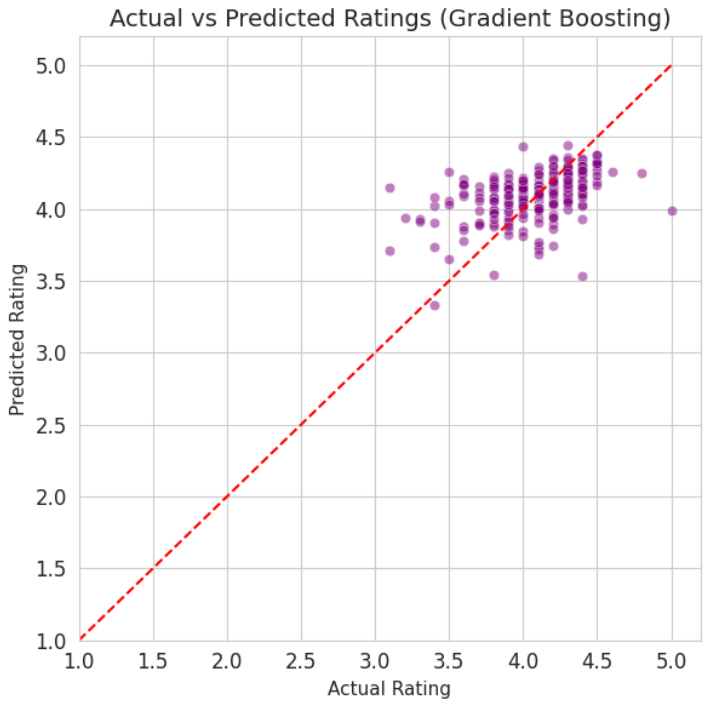

Figure 9: Actual vs Predicted Ratings (Gradient Boosting)

Cause: This is a classic effect of the Loss Function, which minimizes average error by avoiding risk-taking on outliers.

7 General Conclusion

7.1 Project Synthesis

The primary objective of this project was to predict Amazon product ratings by leveraging both their commercial characteristics and the textual content of user reviews. We conducted an end-to-end data science project: from cleaning a noisy dataset to training Machine Learning models, including extensive Exploratory Data Analysis (EDA) and Natural Language Processing (NLP).

The final results, while modest in terms of R^2 (0.21), demonstrate excellent average precision with a Mean Absolute Error (MAE) of **0.18**. We have proven that a product's rating is complex, between its social proof (number of reviews), the qualitative perception expressed in text (sentiment), and its price positioning.

7.2 Response to Business Objectives

Despite the inherent difficulty of predicting human behavior, this model offers tangible value for stakeholders:

Benchmarking Tool: With a precision of ± 0.18 stars, the model can define a "Theoretical Rating" for a given product. If a product has an actual rating of 3.0 while the model predicts 4.2 (based on its price, description, and sentiment), this triggers an alert. It suggests a specific issue (e.g., manufacturing defect, delivery problem) that is not linked to the product's intrinsic characteristics.

Pricing Strategy: Although they are not the top ranking features, the significance of variables such as `log_price` and `discount` suggests that pricing policy directly influences perceived satisfaction.

References

[1] Hutto, C.J. & Gilbert, E.E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Eighth International AAAI Conference on Weblogs and Social Media.

[2] Spiegel Research Center (Northwestern University). *How Online Reviews Influence Sales*. Available at: http://spiegel.medill.northwestern.edu/how-online-reviews-influence-sales/

[3] Kaggle Datasets. *Amazon Product Sales Dataset*. Public Dataset.

[4] Scikit-Learn Documentation. *User Guide: Supervised Learning, Model Selection and Evaluation*. https://scikit-learn.org/stable/user_guide.html

[5] NLTK (Natural Language Toolkit) Documentation. https://www.nltk.org/