UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Amazon Best Seller Rank Prediction Using Machine Learning

NAME:Chen Junhao
Matrix No.:MCS241030

Innovating Solutions

Chapter 1
Introduction

Research Question

- What **product** attributes most influence Amazon best-seller rankings?
- How accurately can a **model** predit ranking based on review count, rating, and price?
- Which **machine learning approaches** are most suitable for predicting rank in such datasets?

Research Objective

- To **explore** and **analyse** factors contributing to Amazon best-seller rankings
- To **develop** and **evaluate** predictive models using machine learning techiques
- To **present** actionable insights through data visualisation
- To **establish** a methodological foundation for future research on e-commerce analytics

Research Scope

Dataset

Name: **Amazon Best Seller dataset**

Source: **Kaggle**

Key features

product_price, product_star_rating,

product_num_ratings, rank, country

Tools

- **Programming Language: Python**
- **IDE: Google Colab**
- **Libraries: pandas, NumPy, seaborn, matplotlib ...**

Chapter 2
Research Gap

Research Question

PAPER TITLE	Models used	Focus	Weaknesses
Best Seller Rank (BSR) to Sales: An empirical look at Amazon	Regression	Mapping BSR to actual sales using scraped data	Small dataset, limited to one category, simple model
Understanding and predicting online product return behavior	Clustering	ML for segmenting e-commerce customers	Not focused on BSR or sales; lacks model evaluation
Application of Machine Learning Algorithms for Customer Segmentation	MLP, Clustering	Predicting customer behavior and preferences	Indirectly related to BSR; vague evaluation details
Machine Learning-Based Customer Behavior Analysis for E-commerce Platforms	NLP (Natural language processing)	Predicting return behavior from review text	Focused on returns, not sales or BSR; depends on review quality

Reaserch gap

Gaps Identified

⚠️Few models directly predict BSR using ML.

⚠️Rare integration of **text + numeric** features.

⚠️Most studies use small or single-category

While previous studies provide useful insights into customer behavior and e-commerce dynamics, most suffer from significant dataset limitations. For instance, Sharma et al. (2020) only examined 10 individual products in a single category, with limited features and a relatively small dataset. Others focus primarily on user reviews without structured sales data or category diversity.

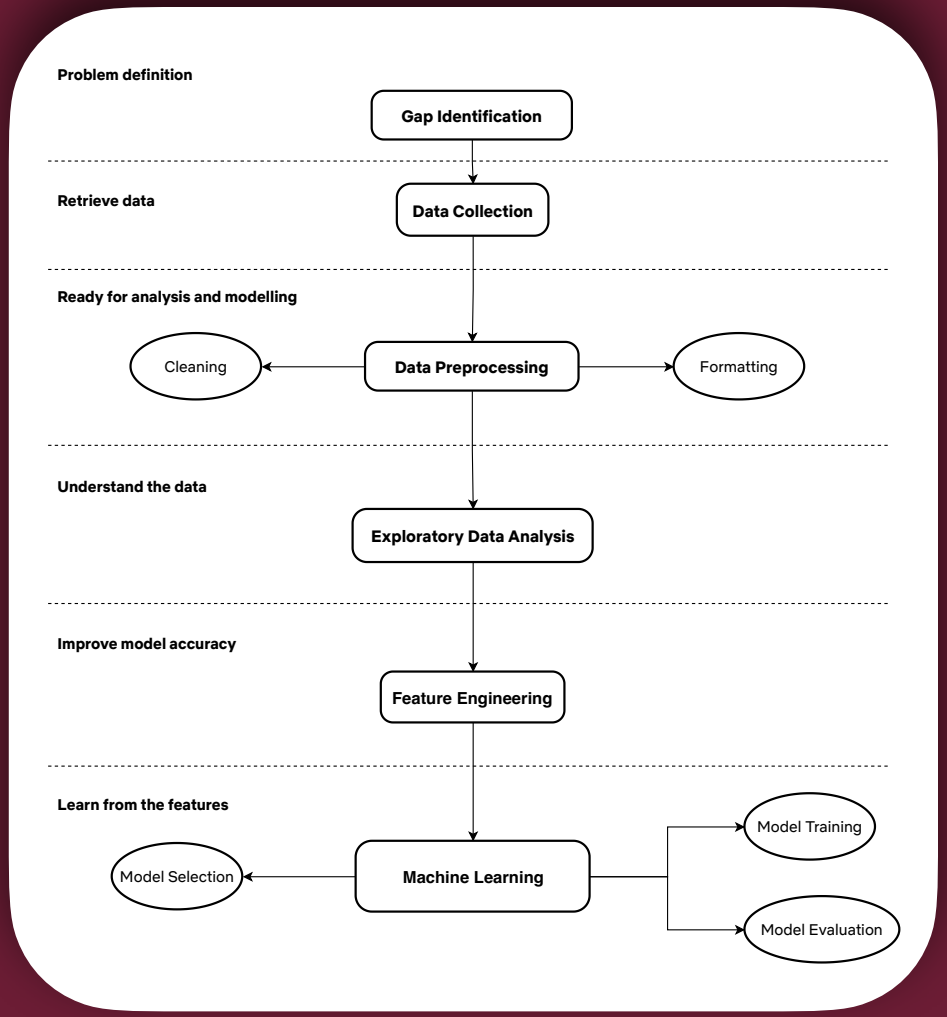
My focus

My novelty

- Build a multi-feature ML model to predict BSR.
- Use sales data, price, reviews, product metadata.
- Apply interpretable models for better insights

Chapter 3
Methodology

Framework



This research framework includes the following steps:

- Gap identification: Align with the research objectives and gaps identified in Chapter 2
- Data collection: Gather product data from Amazon across different categories
- Data preprocessing: Clean and format the data for analysis and modelling
- EDA: Understand the data by checking patterns, trends, and relationships between features
- Feature engineering: Pick or create the best features (e.g. price, reviews) to improve model accuracy
- Machine learning: Train the models using the dataset to learn from the features.

Data Collection

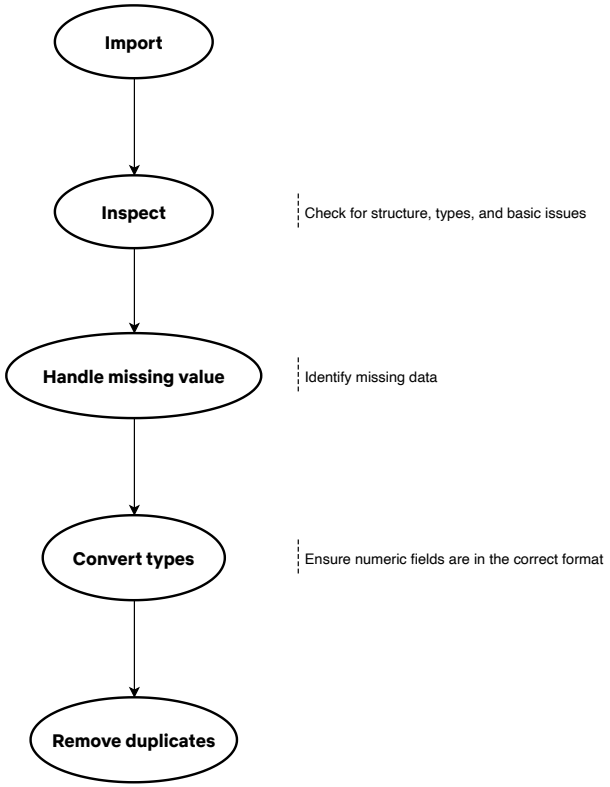
Method

Python programming language

Data

- Product Title
- Category
- Price
- Number of Reviews
- Star Rating
- Review Texts
- Best Seller Rank (BSR)

Data Preprocessing



Data preprocessing is a crucial step before building your machine learning models. It ensures that your data is **clean, consistent, and structured** for accurate analysis and modelling.

Feature engineering

- Selecting Relevant Structured Features
- Removing Irrelevant or Redundant Features
- Encoding Categorical Variables

The goal here was to optimise the quality of information fed into the models.

Machine Learning

UTM UNIVERSITI TEKNOLOGI MALAYSIA

Model Selection

- Compare baseline and advanced models (Linear Regression, Decision Trees, Random Forest)

Model Training

- Apply train-test split and cross-validation
- To train models on processed features

Model Evaluation

- Assess performance using R^2
- To compare results across models

Linear Regression

Formula

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

- $\hat{y}$: Predicted BSR
- β_0: Intercept
- β_i: Coefficients for each feature
- x_i: Independent variables (e.g. price, rating, number of reviews)

Evaluation Metrics for Regression

R-squared (R^2)

- y_i: Actual BSR
- $\hat{y}_i$: Predicted BSR
- $\bar{y}$: Mean of actual BSR values
- A value closer to 1 indicates better fit.

Formula

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

Evaluation Metrics for Regression

Formula

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

Mean Absolute Error (MAE)

- Measures the average absolute difference between actual and predicted BSR.
- Interpreted in the same units as the target variable.

Chapter 4&5
Initial findings & Summary

Introduction

Aims of Chapter 4:

- This chapter presents the **results and interpretation** of the machine learning models developed in Chapter 3.
- I compare **model performance** and analyse which product features contribute most to accurate predictions.
- The aim is to **predict** Amazon Best Seller Rank (BSR) using structured product data.

Introduction

Gaps Identified

⚠Few models directly **predict BSR** using ML.

⚠Rare integration of **text + numeric** features.

⚠Most studies use **small or single-category** datasets.

My novelty

- Build a multi-feature ML model to predict BSR.
- Use sales data, price, reviews, product metadata.
- Apply interpretable models for better insights

Data Cleaning

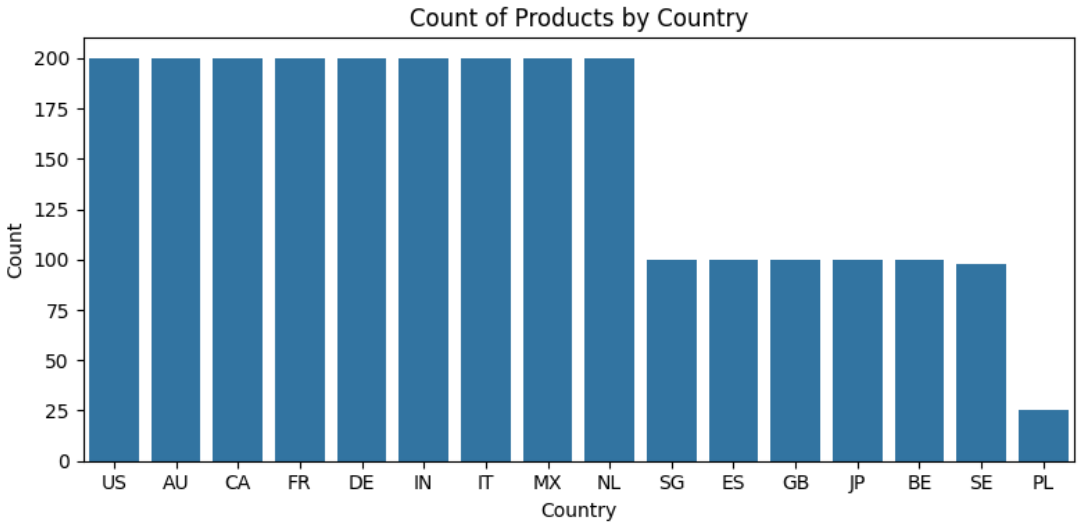
```python
# data cleaning
df['product_price'] = df['product_price'].astype(str)
df['product_price'] = df['product_price'].str.replace(r'[\$,]', '', regex=True)
df['product_price'] = df['product_price'].str.extract(r'(\d+\.\d+|\d+)')
# Convert to float
df['product_price'] = pd.to_numeric(df['product_price'], errors='coerce')
# Convert other columns to numeric
df['product_star_rating'] = pd.to_numeric(df['product_star_rating'], errors='coerce')
# Drop rows with missing values
df_cleaned = df.dropna(subset=['product_price', 'product_star_rating'])
# Reset index
df_cleaned.reset_index(drop=True, inplace=True)
# Preview cleaned data
print(df_cleaned.head())
df_cleaned.shape
```

```
                                        product_title  product_price  \
0  TurboTax Deluxe 2024 Tax Software, Federal & S...          55.99
1  TurboTax Premier 2024 Tax Software, Federal & ...          82.99
2  TurboTax Home & Business 2024 Tax Software, Fe...          95.99
3  TurboTax Business 2024 Tax Software, Federal T...         143.99
4  H&R Block Tax Software Deluxe + State 2024 wit...          49.97

   product_star_rating  product_num_ratings  rank country
0                  4.2               6511.0     1      US
1                  4.1               2738.0     2      US
2                  4.2               1672.0     3      US
3                  4.0                389.0     4      US
4                  3.9               1683.0     5      US
(2027, 6)
```

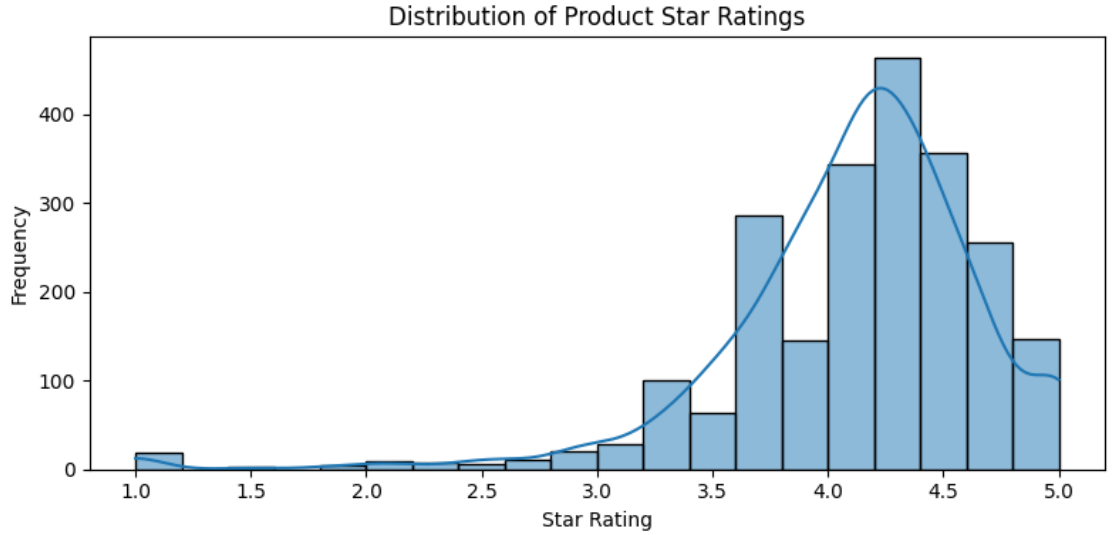
- **Handling missing value**
- **Data type conversion**

Dropping 396 incomplete rows of data

EDA



US, AU, CA, FR, DE, IN, IT, MX, NL markets likely represent Amazon's core global marketplaces with robust data availability.



- Ratings are right-skewed (positively skewed)
- Very few low-rated products
- Peak at 4.3–4.4

Feature engineering

```python
# feature engineering
# Create a new feature: review_density (number of ratings per dollar)
df_cleaned['review_density'] = df_cleaned['product_num_ratings'] / df_cleaned['product_price']

# Create a new feature: weighted_rating (star rating × number of ratings)
df_cleaned['weighted_rating'] = df_cleaned['product_star_rating'] * df_cleaned['product_num_ratings']

# Apply log transformation to number of ratings to reduce skewness
df_cleaned['log_num_ratings'] = np.log1p(df_cleaned['product_num_ratings'])  # log(1 + x)

# One-hot encode the 'country' column
df_cleaned = pd.get_dummies(df_cleaned, columns=['country'], drop_first=True)

# Preview new features
print(df_cleaned[['product_price', 'product_star_rating', 'product_num_ratings',
                  'review_density', 'weighted_rating', 'log_num_ratings']].head())

# Define feature columns
features = ['product_price', 'product_star_rating', 'product_num_ratings',
            'review_density', 'weighted_rating', 'log_num_ratings']

# Add any one-hot encoded columns (e.g., country_US)
features += [col for col in df_cleaned.columns if 'country_' in col]

# Drop rows with missing values in features and target
model_data = df_cleaned[features + ['rank']].dropna()
```

Created new features to capture deeper insights:

- **review_density** = number of reviews per dollar
- **weighted_rating** = rating × number of reviews
- **log_num_ratings** = log-transformed review count (to reduce skewness)

Encoded categorical data:

Applied one-hot encoding to **country** column

Model Evaluation results

Training Set (80%)

Test Set (20%)

Models

- Linear Regression
- Decision Tree
- Random Forest

Evaluation metrics

- R^2 Score
- RMSE
- MAE

Model Evaluation results

```python
# Train the Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Make Predictions
y_pred = model.predict(X_test)

# Evaluate the Model
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)

print(f"R² Score: {r2:.3f}")
print(f"RMSE: {rmse:.2f}")
print(f"MAE: {mae:.2f}")
```

```
R² Score: 0.084
RMSE: 27.66
MAE: 23.80
```

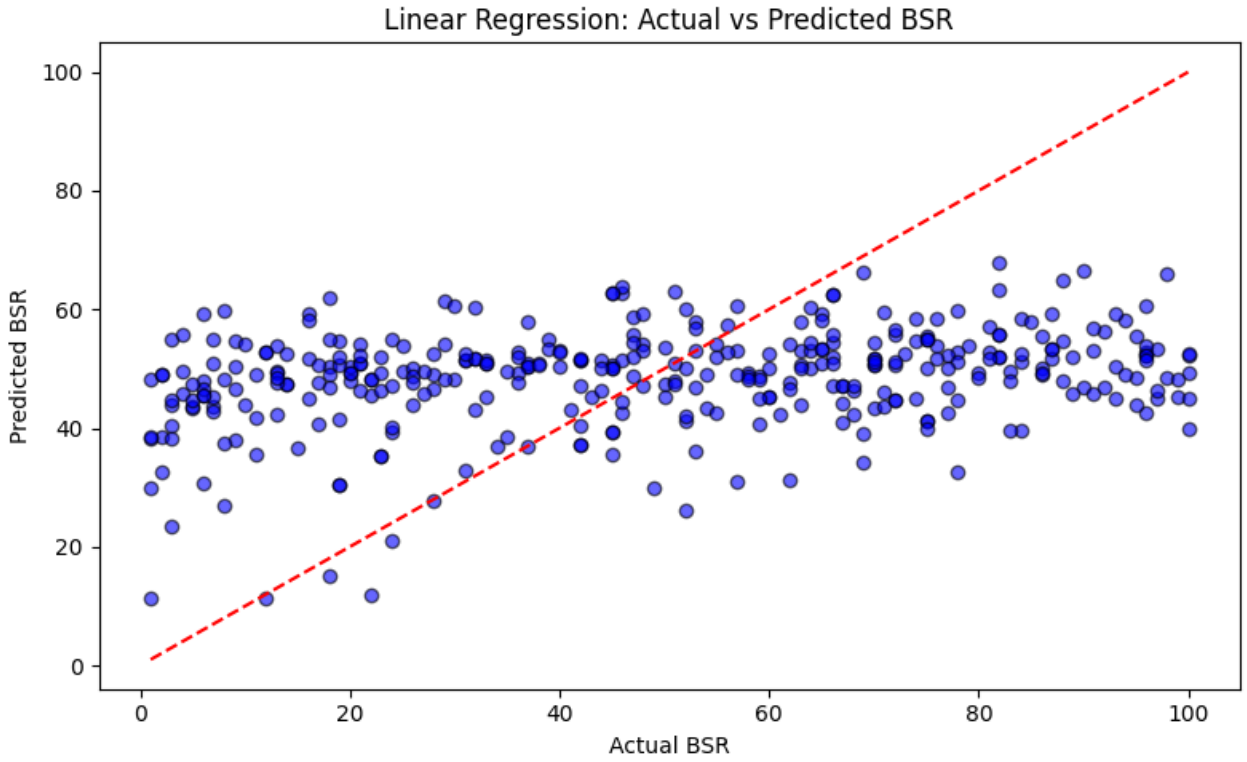
```
Decision Tree Performance:
R² Score: 0.309
RMSE: 24.02
MAE: 11.84

Random Forest Performance:
R² Score: 0.575
RMSE: 18.85
MAE: 13.89
```

Model Evaluation results

MODEL	R^2 Score	RMSE	MAE
Linear Regression	0.084	27.66	23.80
Decision Tree	0.309	24.02	11.84
Random Forest	0.575	18.85	13.89

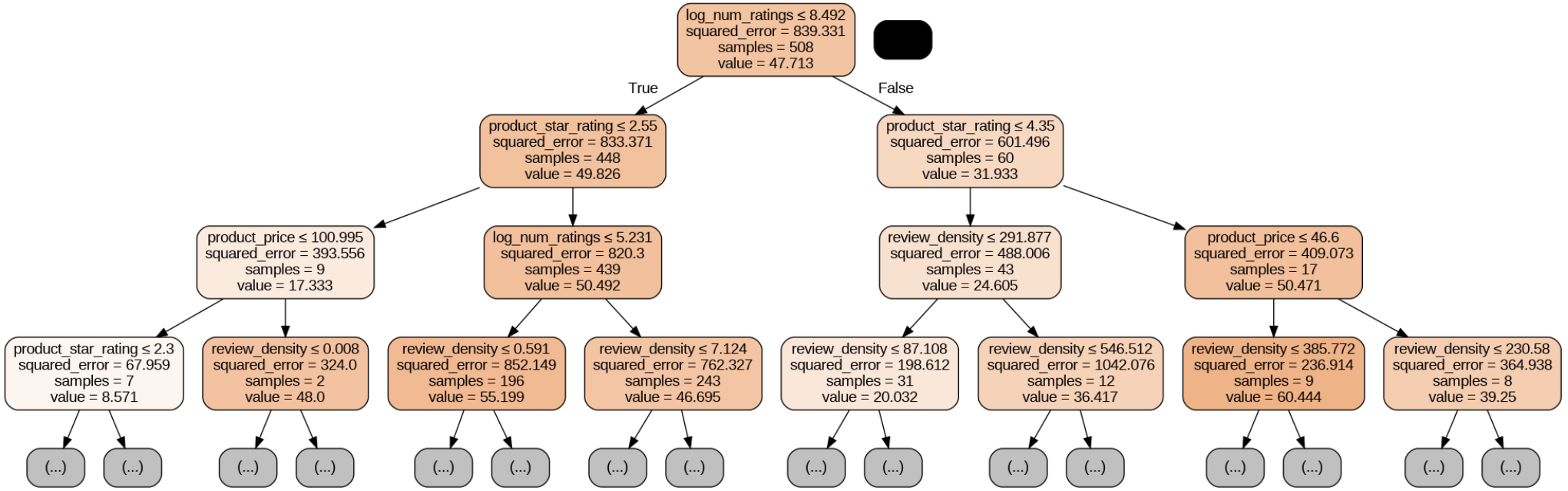
Random Forest performed best, with the highest R^2 and lowest error values, showing it captures the data's complexity better than the others.

Linear regression



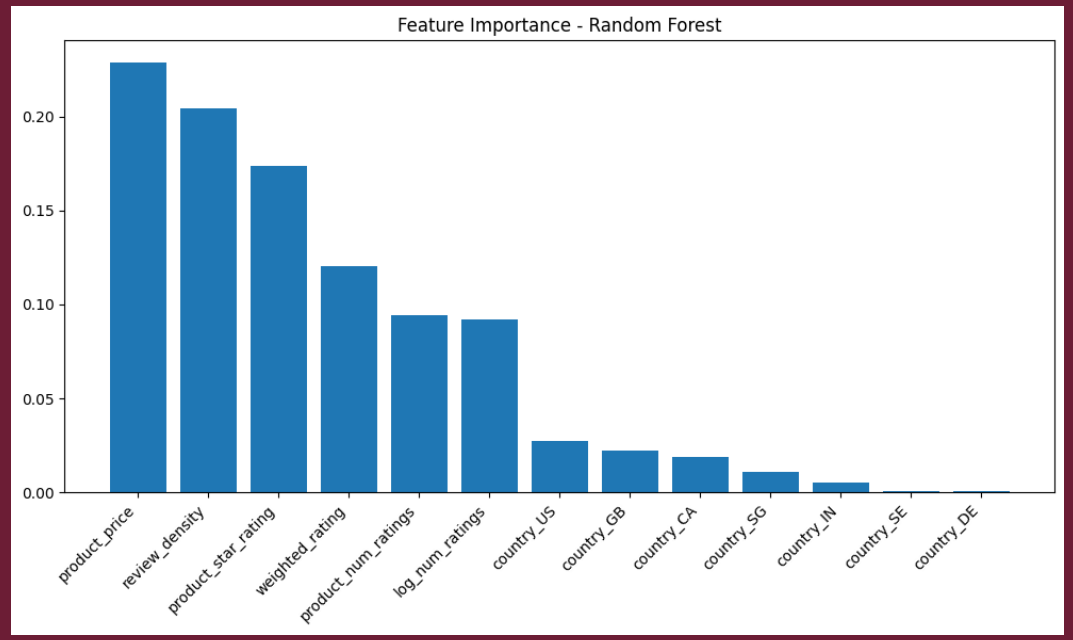
Linear Regression had poor performance — it couldn't capture complex patterns.

Decision tree



Random Forest

Feature Importance



- Top features:
  A. product_price – higher or lower prices have a significant impact on BSR
  B. review_density
  C. product_star_rating– has moderate impact
- Price and review dynamics drive BSR more than rating alone
- Customers seem to prefer well-reviewed, well-priced products over just highly rated ones.

Application

```
new_product = pd.DataFrame([{
    'product_price': 49.99,
    'product_star_rating': 4.5,
    'product_num_ratings': 350,
    'review_density': 350 / 49.99,
    'weighted_rating': 4.5 * 350,
    'log_num_ratings': np.log1p(350),
    'country_BE':1,
    'country_CA':0, 'country_DE':0,
    'country_ES':0,
    'country_FR':0,
    'country_GB':0,
    'country_IN':0,
    'country_IT':0,
    'country_JP':0,
    'country_MX':0,
    'country_NL':0,
    'country_PL':0,
    'country_SE':0,
    'country_SG':0,
    'country_US':0

}])
```

```
predicted_bsr = rf_model.predict(new_product)
print(f"Predicted BSR: {predicted_bsr[0]:.0f}")
```

```
Predicted BSR: 52
```

This section illustrates how the trained Random Forest model can be used to predict BSR for new product listings based on their price, rating, and review metrics.

Summary

- **Feature importance analysis** showed that product_price, review_density, and star_rating were the top predictors of BSR.
- **EDA** revealed skewed rating distributions and imbalanced country data.
- **Three models** were tested: Linear Regression, Decision Tree, and Random Forest.
- **Random Forest performed best** with the highest R^2 and lowest prediction error.
- Results confirm that **non-linear models** better capture the complexity of Amazon Best Seller Rank.