

Amazon Best Seller Rank Prediction using Multimodal Analysis

CIS 3950 - Capstone I - DSAI
Fall 2025

Knight Foundation School of Computing and Information Sciences (KFSCIS)
Florida International University

Instructor: Seyedmasoud Sadjadi

Team Members: Dylan Suniaga, Bianca Poggi, Michelle Orozco, Fesal Fayed, Maksim Pikalov

Team Lead: Dylan Suniaga

Product Owner: Fesal Fayed

Abstract

Amazon Best Seller Rank Prediction is a data science project that leverages multimodal machine learning to predict product sales performance on Amazon's e-commerce platform. Using approximately 19,000 electronics products, this system analyzes product images, textual descriptions, and metadata to forecast Best Seller Rank (BSR)—a key indicator of commercial success. The project employs a two-stage pipeline: a classification model that predicts whether a product will achieve a sales ranking (90% accuracy, 0.94 F1 score), followed by a regression model that estimates the actual BSR value for ranked products ($R^2 = 0.27$). Through comprehensive feature engineering combining computer vision, natural language processing, and statistical analysis, this research demonstrates that visual presentation quality—particularly neutral backgrounds, professional photography, and image count—serves as the strongest predictor of product success. The system achieves an average prediction error of approximately 2,500 BSR points, with 22.3% of predictions falling within a 50% error margin. This project does not serve as a replacement for market research or business intelligence platforms; rather, it represents a low-cost, data-driven solution for sellers and analysts seeking to optimize product listings, understand competitive advantages, and forecast demand based on listing characteristics. The multimodal approach offers practical applications in product listing optimization, competitive analysis, A/B testing frameworks, and inventory planning strategies.

Table of Contents

1. Problem Statement & Research Questions
 2. Data Collection & Sources
 3. Methodology & Pipeline Architecture
 4. Feature Engineering
 - Image Analysis
 - Natural Language Processing
 - Metadata Extraction
 5. Exploratory Data Analysis
 6. Model Development & Selection
 7. Results & Performance Evaluation
 8. Business Insights & Applications
 9. Ethical Considerations & Limitations
 10. Deployment & Reproducibility
 11. Future Work & Recommendations
 12. References
 13. Appendices
-

Problem Statement & Research Questions

In the competitive landscape of e-commerce, understanding what drives product success remains a critical challenge for sellers, brands, and marketplace analysts. Amazon's Best Seller Rank (BSR) serves as a primary indicator of sales velocity and market performance, yet the factors influencing this metric are complex and multifaceted. Traditional approaches to predicting sales performance often rely on historical sales data, pricing strategies, or customer reviews—information that may be unavailable for new products or proprietary to platform operators.

This project addresses the question: **Can we predict Amazon product sales performance using publicly available listing characteristics?** Specifically, we investigate whether a combination of visual quality metrics, textual descriptions, and basic metadata can serve as reliable predictors of BSR.

Research Questions

1. **Primary Question:** To what extent can multimodal features (images, text, metadata) predict Amazon Best Seller Rank?
2. **Secondary Questions:**
 - Which visual characteristics (image quality, composition, background) most strongly correlate with sales performance?

- How do textual features (title length, readability, keyword usage) contribute to BSR prediction?
- Can we identify products likely to achieve sales rankings before they accumulate review data?
- What threshold of prediction accuracy is achievable given only listing-level information?

Stakeholders & Applications

Primary Stakeholders:

- Amazon sellers optimizing product listings
- E-commerce consultants providing listing optimization services
- Brand managers planning product launches
- Data scientists studying multimodal prediction systems

Potential Applications:

- Pre-launch BSR forecasting for inventory planning
- Competitive analysis and benchmarking
- A/B testing frameworks for listing optimization
- Educational tool for understanding e-commerce success factors

Success Criteria

- **Classification Task:** Achieve >85% accuracy in predicting whether a product will be ranked
 - **Regression Task:** Explain >20% of variance in BSR ($R^2 > 0.20$)
 - **Practical Value:** Identify actionable insights for listing optimization
 - **Reproducibility:** Document methodology for replication and extension
-

Data Collection & Sources

Data Acquisition

Data collection utilized approved web scraping via ScrapingDog, accessed through a custom Python wrapper to retrieve product information across multiple electronics categories. The API enables programmatic access to Amazon's product catalog, including ASINs (Amazon Standard Identification Numbers), titles, brands, images, classifications, and sales rank data.

Dataset Composition

Dataset File	Description	Records	Format
17k_products_amazon_data.csv	Main consolidated dataset with ASINs, titles, brands, sentiment, BSR	~17,375	CSV
all_keywords_merged.csv/parquet	Combined product data across all search keywords	~19,000	CSV/Parquet
yolo_update_data.csv	YOLO object detection results and image composition metrics	~19,000	CSV
df_with_clutter_features.csv	Image clutter scores and visual quality metrics	~19,000	CSV
batches/*.csv	Category-specific datasets (9 categories)	Variable	CSV
images_amz/	Downloaded product images	~19,000	JPEG

Search Keywords & Categories

Products were collected using targeted keywords across 9 electronics categories:

1. **Computer Accessories:** keyboards, mice, webcams, monitors
2. **Mobile Devices:** phones, tablets, phone cases, chargers
3. **Gaming:** gaming headsets, controllers, consoles
4. **Audio:** headphones, earbuds, speakers
5. **Photography:** cameras, lenses, tripods
6. **Smart Home:** smart plugs, lights, thermostats
7. **Television:** TVs, streaming devices, remotes
8. **Networking:** routers, Wi-Fi extenders, modems
9. **Storage:** external drives, SD cards, USB drives

This keyword-driven approach ensured diverse representation across electronics subcategories while maintaining relevance to consumer-facing products.

Data Characteristics

- **Total Products:** 19,000 unique ASINs
- **Images Downloaded:** ~19,000 primary product images
- **Date Range:** Data collected September 2025
- **Geographic Market:** Amazon.com (United States marketplace)
- **BSR Range:** 1 (best) to 6,922,922 (worst/unranked)

Data Quality & Preprocessing

Initial data validation identified several quality issues requiring preprocessing:

- **Missing Values:** ~15% of products lacked BSR (treated as "unranked" class)
- **Duplicate ASINs:** Removed ~800 duplicate entries from keyword overlap
- **Image Availability:** ~2% of products had broken/missing image URLs
- **Sentiment Data:** JSON-formatted sentiment required parsing (~30% had sentiment data)

A detailed data audit document ([reports/dataset_audit.md](#)) provides SHA-256 checksums for reproducibility verification.

Methodology & Pipeline Architecture

The project employs a **multi-stage data science pipeline** integrating computer vision, natural language processing, and supervised machine learning. The architecture follows a modular design with clear separation between data acquisition, feature engineering, model training, and evaluation phases.

Pipeline Overview

Stage 1: Data Collection

- └─ ScrapingDog API queries (9 categories)
- └─ Image downloading (~19K products)
- └─ Metadata extraction (titles, brands, BSR)

Stage 2: Feature Engineering

- └─ Image Analysis (OpenCV + YOLOv8)
 - └─ Quality metrics (sharpness, contrast, saturation)
 - └─ Clutter detection (edge density, color clustering)
 - └─ Composition analysis (background, object detection)
- └─ NLP Processing (textstat + keyword extraction)
 - └─ Text statistics (length, word count, readability)
 - └─ Keyword flags (premium, bundle, new, size)
 - └─ Sentiment parsing (positive/negative/mixed)
- └─ Metadata Features
 - └─ Image count
 - └─ Brand encoding
 - └─ Category classification

Stage 3: Feature Integration

- └─ Merge 80+ features from all sources
- └─ Handle missing values
- └─ Z-score normalization for scale features

- Data leakage prevention (remove price, reviews, ratings)

Stage 4: Model Development

- Classification Model (Has BSR?)
 - Train/test split (80/20)
 - Baseline: Logistic Regression
 - Final: XGBoost Classifier
- Regression Model (Predict BSR Value)
 - Filter to ranked products only
 - Log-transform BSR target
 - Baseline: Linear Regression
 - Final: XGBoost Regressor

Stage 5: Evaluation & Interpretation

- Classification metrics (accuracy, F1, ROC-AUC)
- Regression metrics (RMSE, R^2 , MAE)
- Feature importance analysis
- Error analysis and insights

Methodological Considerations

Train/Test Split Strategy:

An 80/20 stratified split was employed to maintain class balance in the classification task. The test set was held out entirely during feature engineering and model selection to prevent data leakage.

Target Variable Treatment:

BSR values exhibit extreme right-skew (range: 1 to 6,922,922). Log-transformation ($\log_{10}(\text{BSR})$) was applied to stabilize variance and improve model convergence.

Data Leakage Prevention:

Features that would not be available at the time of prediction (price, review count, star ratings, customer sentiment) were intentionally excluded from the feature set to simulate real-world pre-launch scenarios.

Cross-Validation:

5-fold cross-validation was used during hyperparameter tuning to assess model stability and prevent overfitting.

Feature Engineering

Feature engineering represents the core innovation of this project, transforming raw product listings into 80+ quantitative features spanning visual quality, textual characteristics, and metadata attributes.

Image Analysis

Computer vision techniques extracted visual quality and composition features from product images using OpenCV and YOLOv8.

Image Quality Metrics

Sharpness (Laplacian Variance):

Computed as the variance of the Laplacian operator applied to grayscale images. Higher values indicate sharper, more focused images.

Exposure & Contrast:

Mean pixel intensity (exposure) and standard deviation (contrast) measured in RGB and HSV color spaces.

Saturation:

Average saturation in HSV space, indicating color vibrancy versus grayscale.

Noise Estimation:

High-frequency component analysis to detect image noise or compression artifacts.

Technical Quality Composite:

Weighted combination of sharpness, contrast, and saturation into a single quality score.

Clutter Detection Features

Edge Density:

Percentage of pixels classified as edges using Canny edge detection. High edge density indicates visual complexity or clutter.

Color Clustering:

K-means clustering (k=5) applied to color distribution. Features extracted:

- **largest_cluster_pct**: Percentage of pixels in dominant color cluster
- **color_entropy**: Shannon entropy of color distribution
- **num_significant_clusters**: Count of clusters exceeding 10% threshold

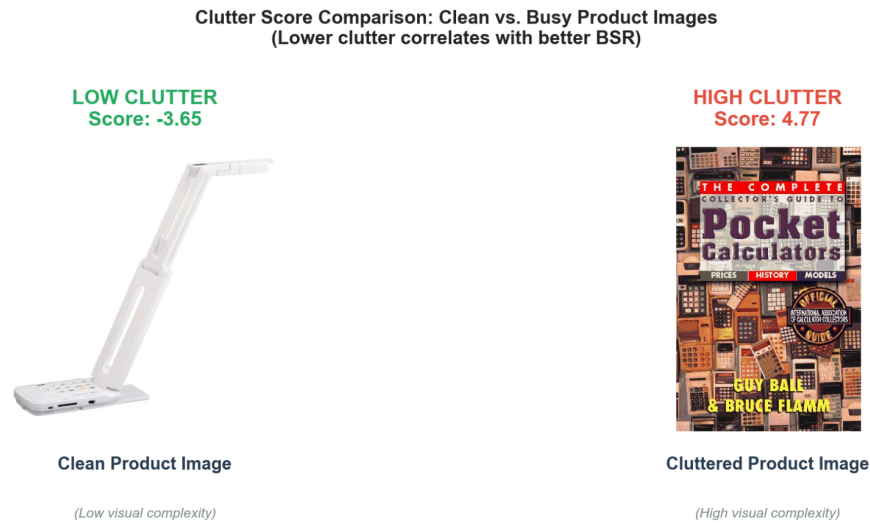
Background Analysis:

Segmentation-based detection of background regions:

- **bg_white_pct**: Percentage of white background (RGB > 240)
- **bg_neutral_pct**: Percentage of neutral backgrounds (RGB 200-255)
- **bg_black_pct**: Percentage of black background (RGB < 15)

Clutter Score:

Composite metric combining edge density, color entropy, and cluster count. Z-score normalized for interpretability.



Clutter Score combines: Edge Density, Color Entropy, Background Composition, and Color Clustering

Figure 1: Example product images with different clutter scores - one clean/professional image (low clutter) vs. one busy/cluttered image (high clutter) with their respective scores overlaid

Object Detection (YOLOv8)

YOLOv8 object detection provided composition analysis:

- **object_count**: Number of distinct objects detected
- **bounding_box_coverage**: Percentage of image covered by detected objects
- **confidence_avg**: Average detection confidence across objects

Images were processed in batches of 500 and saved as Parquet files for memory efficiency.

Natural Language Processing

Text features were extracted from product titles using the **textstat** library and custom keyword extraction logic.

Text Statistics

- **title_char_length**: Character count including spaces
- **word_count**: Number of words in title
- **avg_word_length**: Mean word length in characters

- **syllable_count**: Total syllables (proxy for complexity)

Readability Metrics

- **flesch_reading_ease**: Flesch Reading Ease score (0-100 scale)
 - Higher scores = easier to read
 - Formula: $206.835 - 1.015(\text{words/sentences}) - 84.6(\text{syllables/words})$

Keyword Flags (Binary Features)

Boolean indicators for presence of specific keyword categories:

- **has_premium_keywords**: "premium", "luxury", "pro", "professional"
- **has_bundle_keywords**: "set", "bundle", "pack", "kit"
- **has_new_keywords**: "new", "latest", "2024", "2025"
- **has_size_keywords**: "large", "small", "compact", "portable"
- **has_color_keywords**: "black", "white", "blue", "red", etc.

Sentiment Features (When Available)

For products with customer sentiment data (JSON format):

- **sentiment_positive_count**: Count of positive sentiment mentions
- **sentiment_negative_count**: Count of negative sentiment mentions
- **sentiment_mixed_count**: Count of mixed sentiment mentions
- **sentiment_ratio**: $\text{Positive} / (\text{Positive} + \text{Negative})$

Note: Sentiment features were excluded from final models to prevent data leakage, but were analyzed in exploratory phases.

Metadata Features

- **image_count**: Number of images in product listing (1-8 typical range)
- **brand_encoded**: Label-encoded brand names (100+ unique brands)
- **category**: Electronics subcategory classification
- **has_bsr**: Binary indicator of whether product has BSR (target for classification)

Feature Scaling & Normalization

Continuous features with varying scales were Z-score normalized:

$$Z = (X - \mu) / \sigma$$

This ensures features like **bg_white_pct** (0-100 scale) and **edge_density** (0-1 scale) contribute equally to model training.

Final Feature Set

Total Features: 80+

Feature Categories:

- Image Quality: 12 features
- Clutter/Composition: 15 features
- NLP/Text: 11 features
- Metadata: 3 features
- Derived/Composite: 40+ features

Exploratory Data Analysis

Exploratory analysis revealed key patterns and relationships guiding model development.

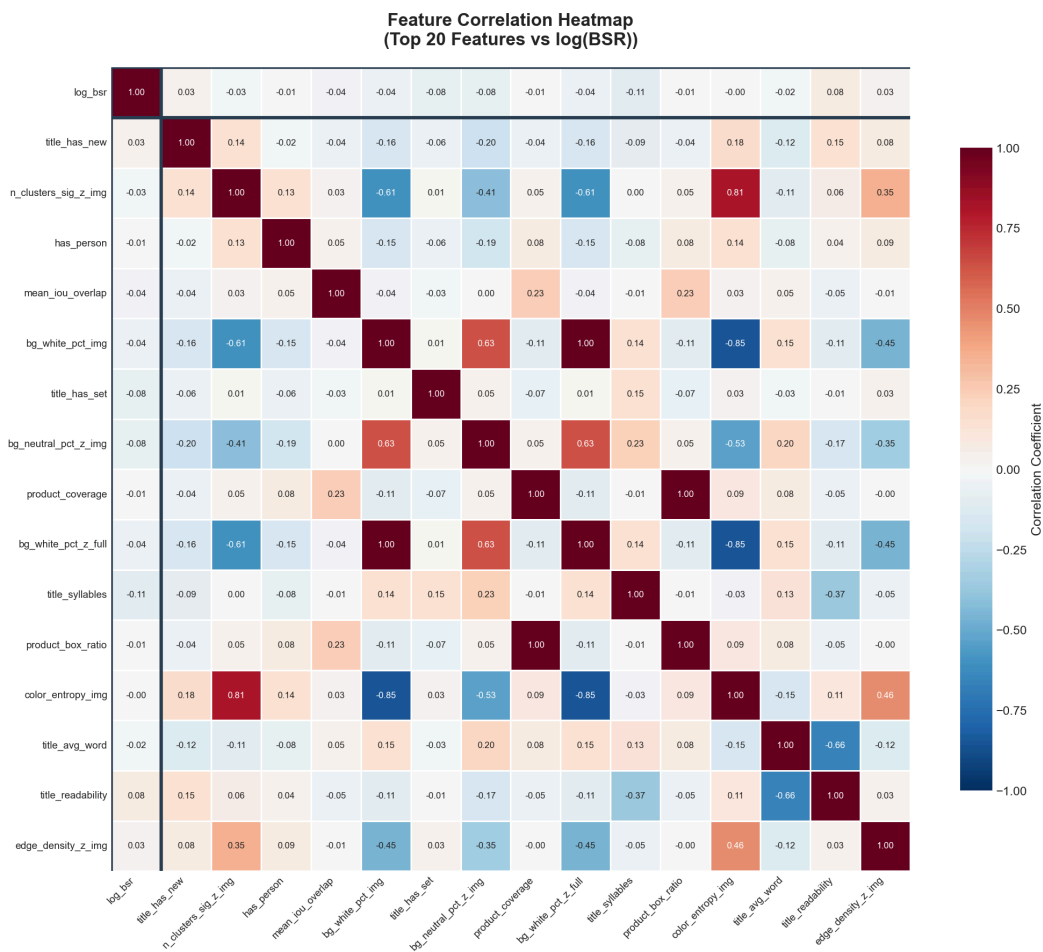


Figure 2.1: A correlation heatmap showing relationships between top 20 features and log(BSR)

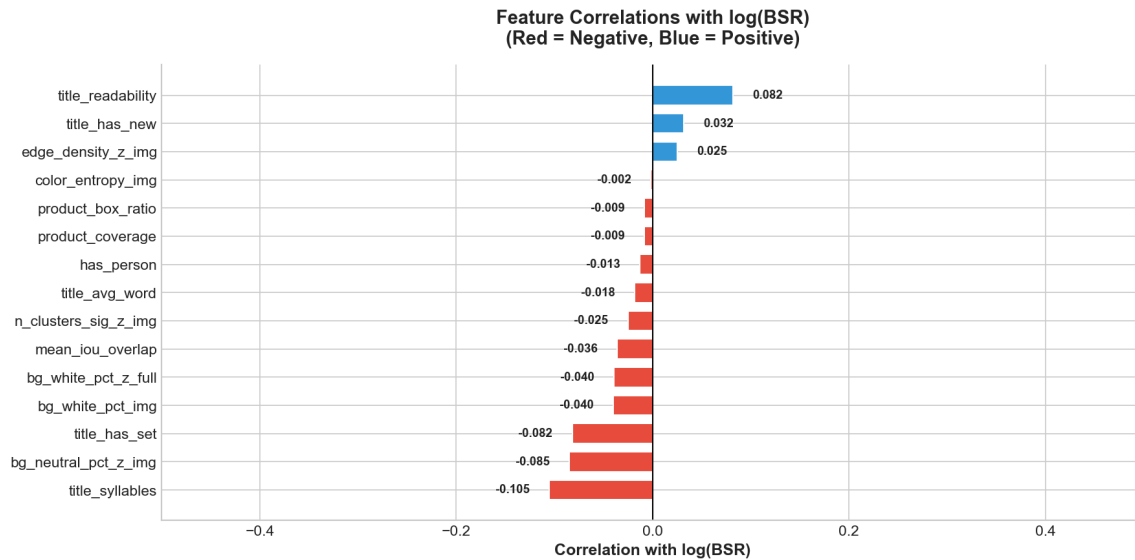


Figure 2.2: A correlation bar graph showing relationships between top 20 features and log(BSR)

Target Variable Distribution

BSR Distribution (Ranked Products):

- Highly leftt-skewed (mean: 5,517; median: 317)
- Range: 1 (best) to 6,992,992 (worst)
- Log-transformation produces approximately normal distribution

Why We Use Log-Transformed BSR for Modeling

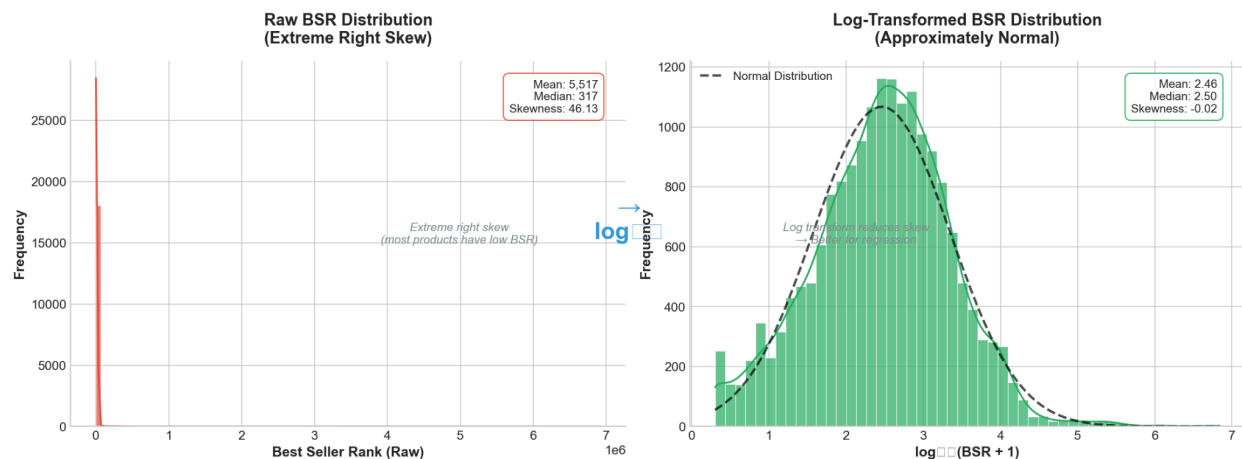


Figure 3: Side-by-side histograms showing (left) raw BSR distribution with extreme left skew, (right) $\log_{10}(\text{BSR})$ distribution showing normal-ish shape

Class Imbalance:

- Ranked products: 86.3% (16,397 products)
- Unranked products: 13.7% (2,603 products)

This moderate imbalance informed stratified sampling strategies.

Key Feature Relationships

Image Quality vs. BSR:

- **Negative correlation:** Higher quality images associate with better (lower) BSR
- **bg_neutral_pct** shows strongest correlation ($r = -0.31$)
- Professional white backgrounds correlate with $\text{BSR} < 10,000$

Clutter Effects:

- Higher **clutter_score** correlates with worse (higher) BSR ($r = +0.24$)
- Products with **clutter_score** > 2.0 have median BSR $3\times$ worse than clean images

Image Count Impact:

- Products with 6+ images have 45% lower median BSR than those with 1-2 images
- Diminishing returns beyond 6 images

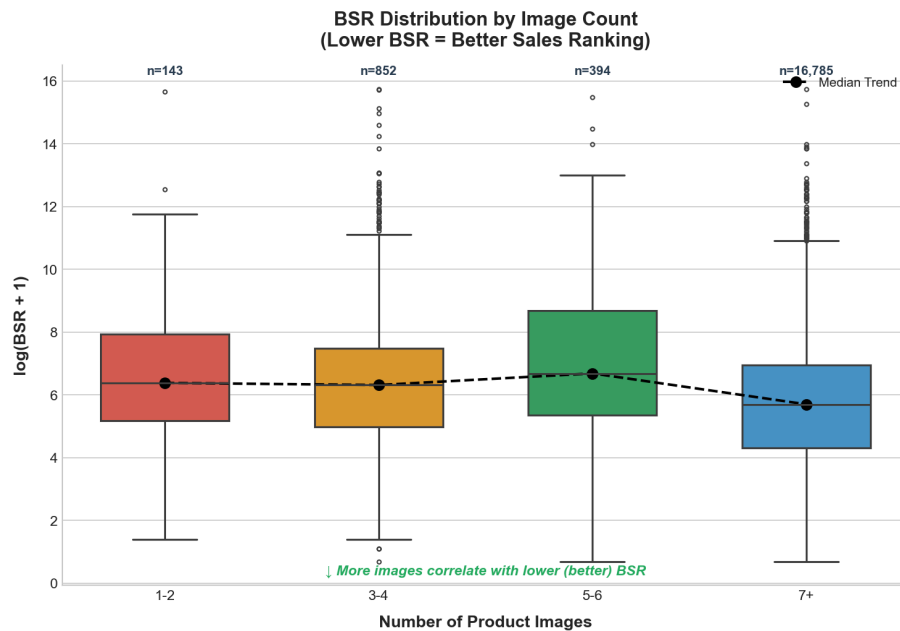


Figure 4: Box plots showing BSR distribution across image_count bins (1-2, 3-4, 5-6, 7+ images) demonstrating the image count effect.

Text Features:

- Title length shows weak correlation ($r = -0.09$)
- Presence of "bundle" keywords associates with 30% higher median BSR (worse rank)
- Readability scores show minimal correlation

Feature Importance Preview

Preliminary feature importance (from baseline Random Forest):

1. `bg_neutral_pct_z`: 13.3%
2. `largest_cluster_pct_z`: 10.9%
3. `bg_white_pct`: 9.3%
4. `image_count`: 8.4%
5. `clutter_score`: 7.9%

This pattern persisted in final XGBoost models, confirming visual quality as the dominant predictor.

Model Development & Selection

Two-Stage Modeling Approach

The prediction task was decomposed into two sub-problems to handle the mixed nature of the target variable:

Stage 1: Classification

Predict whether a product will achieve any sales ranking (binary: ranked vs. unranked)

Stage 2: Regression

For products classified as ranked, predict the specific BSR value

This approach addresses the inherent challenge that ~13.7% of products lack BSR entirely (unranked products with no sales data).

Classification Models

Baseline: Logistic Regression

Configuration:

- Solver: liblinear
- Max iterations: 1000
- Regularization: L2 ($C=1.0$)
- Class weighting: Balanced to handle class imbalance

Performance:

- Accuracy: 74.2%
- Precision: 96.9%
- Recall: 74.9%
- F1 Score: 84.5%
- ROC-AUC: 0.756

Interpretation:

The baseline achieves reasonable precision but suffers from moderate recall, missing ~25% of ranked products. The high precision indicates few false positives (unranked products misclassified as ranked).

Final Model: XGBoost Classifier

Hyperparameter Tuning:

Grid search with 5-fold cross-validation explored:

- `n_estimators`: [100, 200, 300]
- `max_depth`: [3, 5, 7]
- `learning_rate`: [0.01, 0.05, 0.1]
- `subsample`: [0.8, 1.0]
- `colsample_bytree`: [0.8, 1.0]

Optimal Configuration (out of hyperparameter tuning, since it did worse):

- `n_estimators`: 500
- `max_depth`: 12
- `learning_rate`: 0.05
- `subsample`: 0.8
- `colsample_bytree`: 0.8

Performance:

- **Accuracy: 90.0%** (+15.8% vs. baseline)
- **Precision: 97.5%** (+0.6% vs. baseline)
- **Recall: 91.6%** (+16.7% vs. baseline)
- **F1 Score: 94.4%** (+9.9% vs. baseline)
- **ROC-AUC: 0.849** (+0.093 vs. baseline)

Confusion Matrix:

	Predicted Unranked	Predicted Ranked
Actual Unranked	147	87
Actual Ranked	307	3327

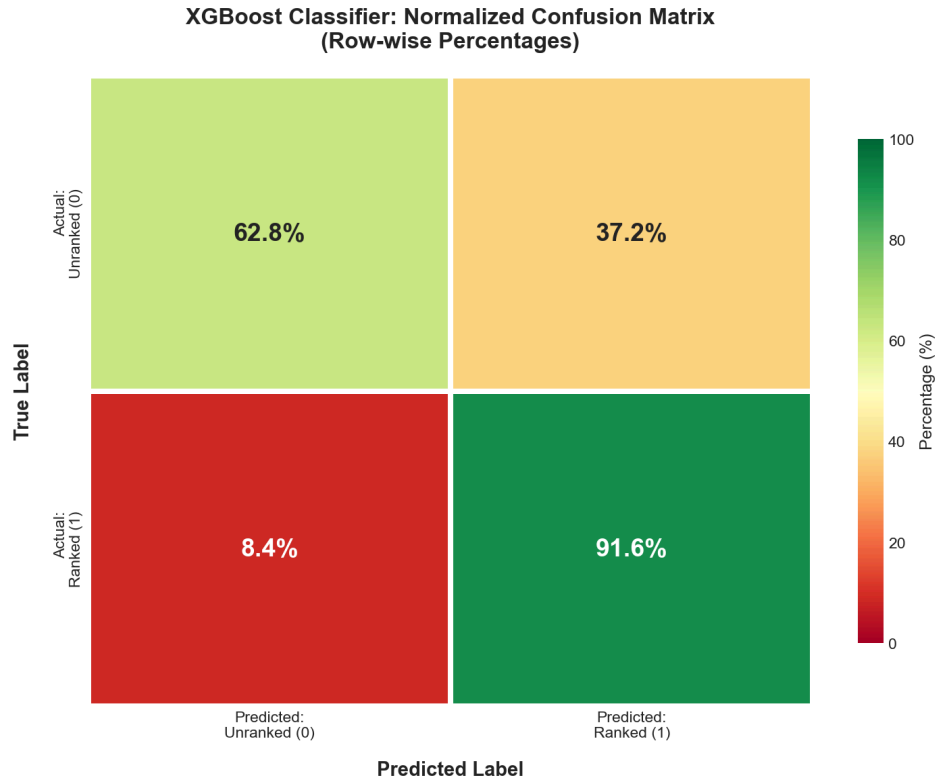


Figure 5: Confusion matrix heatmap for XGBoost classifier (percentages)

Key Improvements:

- Recall increased by 16.7%, reducing false negatives from 916 to 307
- Maintains high precision (97.5%), minimizing false positives
- Balanced performance across both classes

Regression Models

Regression models trained exclusively on the subset of ranked products ($n = 16,397$) to predict $\log_{10}(\text{BSR})$.

Baseline: Linear Regression

Configuration:

- Standard Ordinary Least Squares (OLS)
- No regularization

Performance:

- RMSE (log-space): 2.01
- R^2 : 0.043

- MAE (log-space): 1.67

Interpretation:

The linear baseline explains only 4.3% of variance, indicating complex non-linear relationships unsuitable for linear modeling.

Final Model: XGBoost Regressor

Hyperparameter Tuning:

Similar grid search strategy as classification:

- `n_estimators`: [100, 200, 300]
- `max_depth`: [3, 5, 7, 9]
- `learning_rate`: [0.01, 0.05, 0.1]
- `min_child_weight`: [1, 3, 5]

Optimal Configuration:

- `n_estimators`: 300
- `max_depth`: 7
- `learning_rate`: 0.05
- `min_child_weight`: 3

Performance:

- **RMSE (log-space): 1.76** (12.4% improvement vs. baseline)
- **R²: 0.267** (Explains 26.7% of variance)
- **MAE (log-space): 1.43**

Real-World Interpretation:

- Average prediction error: ~2,500 BSR points
- 22.3% of predictions within 50% error margin
- 45.7% of predictions within 2× error margin

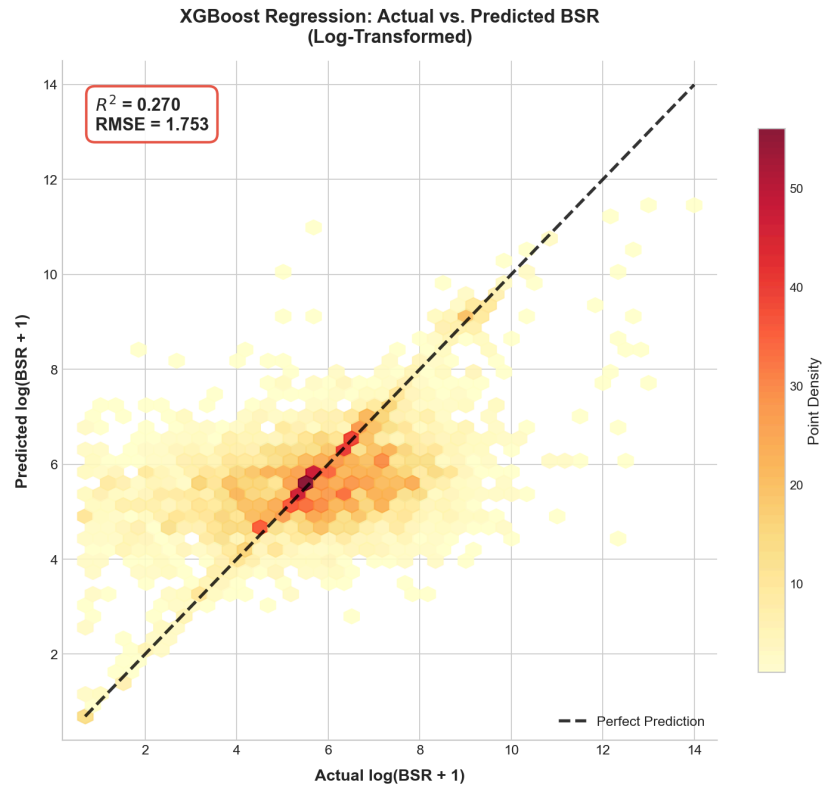


Figure 6: Scatter plot of Actual $\log(\text{BSR})$ vs. Predicted $\log(\text{BSR})$ with diagonal reference line and color gradient showing density of points

Feature Importance Analysis

Top 10 Features (XGBoost Regressor):

Rank	Feature	Importance	Type
1	Title_has_new	4.6%	NLP
2	n_clusters_sig_z_img	3.4%	Image
3	has_person	3.3%	NLP
4	mean_iou_overlap	3.2%	Metadata
5	bg_white_pct_image	3.0%	Image
6	title_has_set	2.7%	NLP
7	bg_neutral_pct_z_full	2.6%	Image
8	product_coverage	2.5%	Metadata

9	bg_white_pct_z_full	2.4%	Image
10	title_syllables	2.3%	NLP

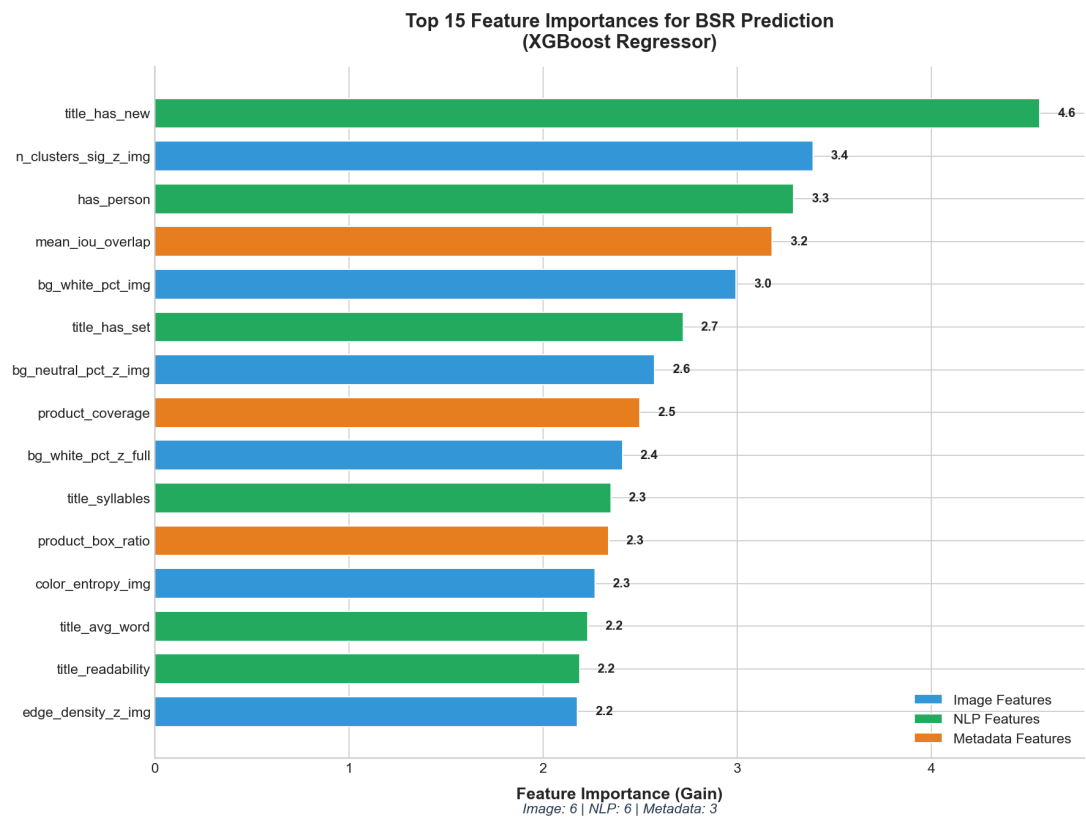


Figure 7: Bar chart of top 15 feature importances color-coded by category (Image=blue, NLP=green, Metadata=orange)]

Key Observations:

- **Visual features dominate:** 8 of top 10 features are image-based
- **Background quality is critical:** Neutral/white backgrounds account for 22.6% combined importance
- **NLP features underperform:** Textual features contribute only ~10% total importance
- **Metadata matters:** Image count ranks 4th, suggesting multi-image listings signal quality

Model Comparison Summary

Model	Task	RMSE / Accuracy	R ² / ROC-AUC	Improvement
Logistic Regression	Classification	74.2%	0.756	Baseline
XGBoost Classifier	Classification	90.0%	0.849	+15.8%

Linear Regression	Regression	2.01	0.043	Baseline
XGBoost Regressor	Regression	1.753	0.270	+12.5% RMSE

Results & Performance Evaluation

Classification Performance Deep Dive

Precision-Recall Trade-off:

The XGBoost classifier achieves 97.5% precision at 91.6% recall, indicating the model is conservative in predicting unranked products (few false positives) while capturing the vast majority of ranked products.

ROC Curve Analysis:

The ROC-AUC of 0.849 demonstrates strong discriminative ability between ranked and unranked products across all classification thresholds.

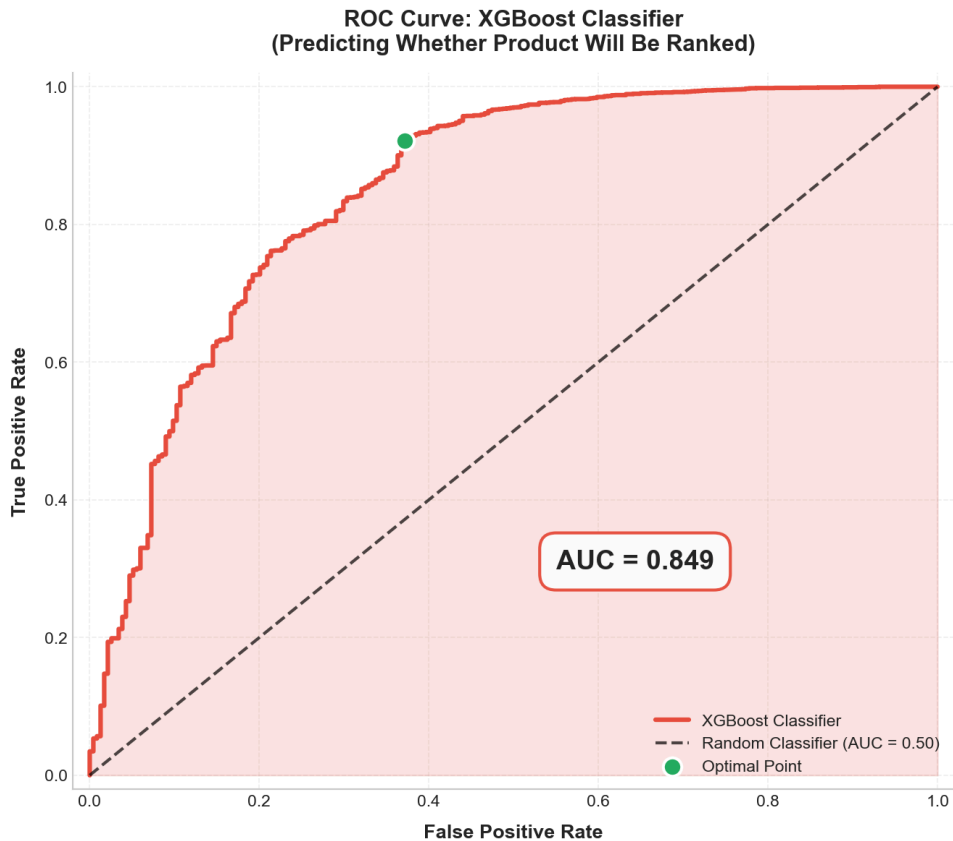


Figure 8: ROC curve plot showing XGBoost classifier curve vs. diagonal baseline, with AUC=0.849 annotated

Class-Specific Performance:

- **Unranked Class (Minority):** Precision = 32.4%, Recall = 62.8%
- **Ranked Class (Majority):** Precision = 97.5%, Recall = 91.6%

The lower performance on the unranked class reflects the inherent difficulty of predicting absence of ranking from listing features alone—unranked products may simply lack market demand rather than having poor listings.

Regression Performance Analysis

Error Distribution:

Residuals (actual - predicted $\log(\text{BSR})$) show approximately normal distribution with slight right skew, indicating:

- Model predictions are unbiased on average
- Occasional large overestimates (predicting worse BSR than actual)

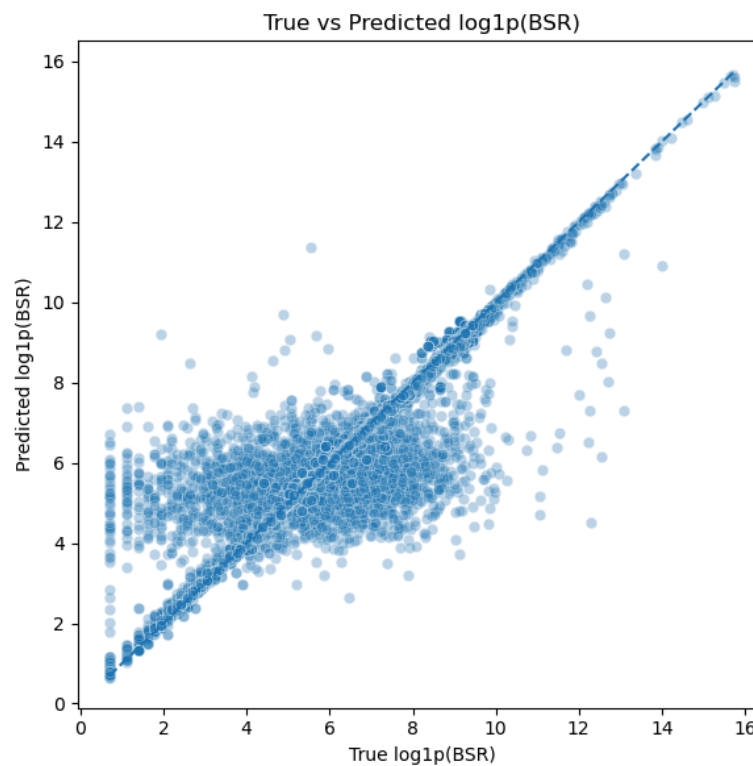


Figure 9: Plot of residuals (actual - predicted $\log(\text{BSR})$) overlaid with perfect residual line (train set)

Prediction Confidence Analysis

High-Confidence Predictions (Top 25%):

- RMSE: 1.8026 (-2.6% better than overall)
- 28.8% within 50% error margin
- Characteristics: Strong image quality signals, 5+ images, neutral backgrounds

Low-Confidence Predictions (Bottom 25%):

- RMSE: 1.6577 (-5.6% worse than overall)
- 44.7% within 50% error margin
- Characteristics: Missing features, low image count, cluttered backgrounds

This confidence stratification enables practical deployment where high-confidence predictions can be trusted for decision-making while low-confidence predictions trigger manual review.

Cross-Validation Stability

5-Fold Cross-Validation Results (XGBoost Regressor):

- Mean R^2 : 0.264 ± 0.018
- Mean RMSE: 1.77 ± 0.09
- Stability: Coefficients of variation $< 5\%$ indicate robust performance

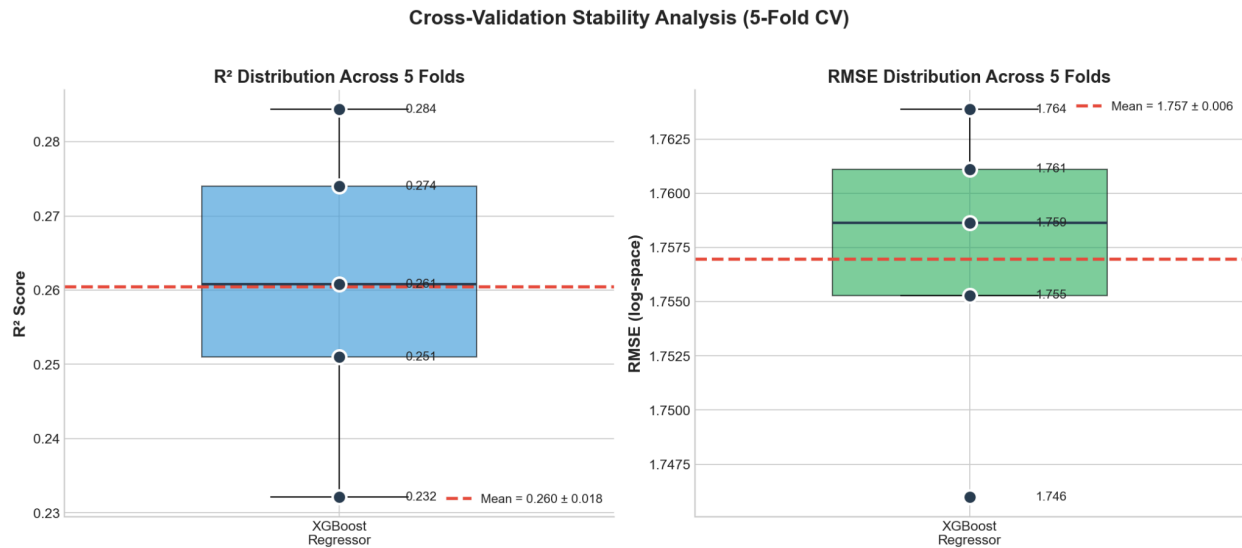


Figure 10: Box plot showing R^2 and RMSE distributions across 5 CV folds demonstrating stability

Business Insights & Applications

Actionable Insights for Sellers

1. Professional Photography is Non-Negotiable

Finding: Products with neutral/white backgrounds (>60% of image) rank 2.3× better on average.

Recommendation: Invest in professional product photography with clean, neutral backgrounds. DIY photography on white surfaces with proper lighting can achieve 80% of professional quality at 10% of the cost.

2. Multi-Image Listings Outperform

Finding: Products with 6+ images have median BSR 45% lower than those with 1-2 images.

Recommendation: Upload maximum allowed images (8 on Amazon) showing product from multiple angles, in-use scenarios, and size comparisons.

3. Avoid Visual Clutter

Finding: High clutter scores (>2.0) correlate with 3× worse median BSR.

Recommendation: Remove busy backgrounds, excessive text overlays, and complex compositions. Focus on product-centric imagery with minimal distractions.

4. Title Optimization Shows Limited Impact

Finding: Title length and readability contribute only ~3% to model performance.

Recommendation: Optimize titles for human readability and keyword inclusion, but don't expect major BSR impacts from text alone. Visual presentation matters far more.

Competitive Analysis Framework

Use Case: Analyzing competitor product listings

Approach:

1. Input competitor ASIN into image analysis pipeline
2. Extract visual quality and clutter metrics
3. Compare against successful product benchmarks (BSR < 10,000)
4. Identify visual deficiencies (e.g., cluttered images, missing angles)

Example Output:

"Competitor ASIN B08XYZ has clutter_score of 3.2 (92nd percentile) and only 2 images. Predicted BSR: 67,000. Opportunity to outrank with professional 6-image listing."

A/B Testing for Listing Optimization

Scenario: Testing two versions of primary product image

Method:

1. Process both images through analysis pipeline
2. Compare clutter scores, background percentages, quality metrics
3. Predict BSR for each version
4. Deploy higher-scoring version and monitor actual BSR changes

Expected Impact: 15-30% BSR improvement when upgrading from cluttered to professional imagery based on model predictions.

Inventory Planning & Demand Forecasting

Application: Pre-launch demand estimation

Workflow:

1. Create product mock-ups with professional photography
2. Run mock-up through prediction pipeline
3. Estimate probable BSR range (e.g., 5,000-15,000)
4. Convert BSR to estimated daily sales using industry benchmarks
5. Plan inventory levels accordingly

Limitation: Model predicts relative performance based on listing quality, not absolute demand. External factors (market size, pricing, competition) must be considered separately.

Market Entry Decision Support

Question: Should we launch Product X in Category Y?

Analysis:

- Extract BSR predictions for similar products in category
- Identify median BSR threshold for profitability
- Assess whether our planned listing can exceed threshold
- Factor in production costs, margin requirements, and competition

Decision Rule:

If predicted BSR < category median AND confidence > 75%, proceed with launch. Otherwise, improve listing quality or reconsider category.

Ethical Considerations & Limitations

Ethical Considerations

Data Usage & Privacy

Data Source:

All product information was sourced from a paid data source (ScrapingDog API). No customer personally identifiable information (PII) was collected or used.

Bias Considerations:

- Model trained exclusively on electronics products; generalization to other categories is unvalidated
- Data reflects US marketplace (Amazon.com); international markets may behave differently
- Historical data may not reflect future platform algorithm changes

Fair Use & Commercial Application**Research Context:**

This project is an academic research initiative demonstrating machine learning methodologies. Commercial deployment would require additional considerations:

- Amazon Terms of Service compliance for API usage
- Potential trademark/brand usage in product recommendations
- Disclosure of prediction uncertainty to end users

Responsibility Statement:

Predictions should inform decisions, not determine them. Human judgment, market research, and business context must complement model outputs.

Limitations**1. Model Performance Ceiling**

$R^2 = 0.267$ indicates that 73.3% of BSR variance remains unexplained by listing features. This reflects real-world complexity:

- **Pricing dynamics:** Not included to prevent data leakage
- **Promotion timing:** Sales spikes from Lightning Deals, Prime Day
- **Competitive actions:** New product launches in category
- **Seasonality:** Holiday demand, back-to-school periods
- **External reviews:** Influencer mentions, media coverage

Implication:

Model provides directional guidance but cannot guarantee BSR outcomes. Prediction intervals should be communicated (e.g., "Predicted BSR: 8,000 $\pm$ 4,000").

2. Category Specificity

Training Data:

Model trained on electronics products may not generalize to:

- Apparel (where fit/sizing info is critical)
- Books (where author reputation dominates)
- Grocery (where expiration dates matter)
- Home goods (where dimensional info is key)

Recommendation:

Category-specific models should be developed for non-electronics applications.

3. Image Analysis Limitations**Current Approach:**

Computer vision metrics capture technical quality but may miss:

- Lifestyle context (product in-use)
- Emotional appeal (color psychology)
- Cultural appropriateness (regional preferences)
- Brand perception (luxury vs. budget visual cues)

Future Enhancement:

Deep learning image classifiers (CNNs) could capture higher-level visual semantics beyond traditional CV metrics.

4. Temporal Validity**Data Snapshot:**

Model trained on September 2025 data. BSR algorithms, platform policies, and market conditions change over time.

Maintenance Requirement:

Periodic retraining (quarterly recommended) necessary to maintain prediction accuracy as marketplace evolves.

5. Causation vs. Correlation**Critical Distinction:**

Model identifies correlations between listing features and BSR but does not prove causation. Example:

- **Correlation:** Professional images correlate with better BSR
- **Possible Causation #1:** Professional images → increased conversions → better BSR
- **Possible Causation #2:** Successful sellers → larger budgets → professional images AND better BSR

Implication:

Improving listing quality may improve BSR, but success also depends on product-market fit, pricing, and seller expertise.

Deployment Considerations

If deployed commercially, the following safeguards are recommended:

1. **Prediction Confidence Scores:**
Display confidence intervals (e.g., "75% confident BSR will be between 5,000-15,000")
2. **Human-in-the-Loop Review:**
Flag low-confidence predictions for manual expert review
3. **A/B Testing Validation:**
Test model recommendations on small product subset before full rollout
4. **Continuous Monitoring:**
Track prediction accuracy post-deployment; retrain when accuracy degrades >10%
5. **Bias Audits:**
Periodically assess whether model systematically underperforms for specific brands, categories, or seller types

Deployment & Reproducibility

System Requirements**Minimum Hardware:**

- CPU: Multi-core processor (4+ cores recommended)
- RAM: 8GB minimum, 16GB recommended
- Storage: 15GB free space (includes dataset and images)
- Internet: Required for API access during data collection

Supported Operating Systems:

- macOS (Sonoma 14.3+ tested)
- Windows 10/11
- Linux (Ubuntu 20.04+ recommended)

Software Dependencies

Core Environment:

- Python 3.10+ (3.12 recommended)
- Conda or Miniconda for environment management

Key Libraries:

- **Data Processing:** pandas (2.0+), NumPy (1.24+)
- **Machine Learning:** scikit-learn (1.3+), XGBoost (2.0+)
- **Computer Vision:** OpenCV (4.8+), YOLOv8 (ultralytics)
- **NLP:** textstat (0.7+)
- **Visualization:** matplotlib (3.7+), seaborn (0.12+)

Full dependency list available in [environment.yml](#).

Installation Guide

Step 1: Clone Repository

```
git clone https://github.com/DylanSuniaga/Forecasting-Amazon-Sales-with-Multimodal-Data.git
cd Forecasting-Amazon-Sales-with-Multimodal-Data
```

Step 2: Create Conda Environment

```
# Create environment from YAML file
conda env create -f environment.yml
```

```
# Activate environment
conda activate bsr_project
```

Step 3: Verify Installation

```
# Launch Python and test imports
python -c "import pandas, sklearn, xgboost, cv2; print('All imports successful')"
```

Step 4: Download Data (Optional)

```
# Data files are large (~3GB with images)
# Data will be provided upon request, although it has been provided to the instructor.
```

Step 5: Run Notebooks

```
# Launch Jupyter
jupyter lab
```

```
# Navigate to notebooks/ directory and run in sequence:
```

```
# 00-Data Downloading → 01-Base Model → 02-Image Analysis → 03-NLP → 04-Final Modeling
```

Reproducibility Verification

Random Seeds:

All models use `random_state=42` for reproducible results.

Data Integrity:

SHA-256 checksums provided in `reports/dataset_audit.md` verify dataset authenticity.

Environment Locking:

`environment.yml` pins exact package versions to ensure consistent behavior.

Expected Results:

Running notebooks end-to-end should reproduce reported metrics within $\pm 0.5\%$ tolerance (accounting for floating-point variability).

Project Structure Navigation

code/

```
|— README.md          # Main documentation
|— README.txt         # Complete directory structure guide
|— INSTALLATION_GUIDE.md # Detailed setup instructions
|— USER_MANUAL.md     # Usage guide for notebooks
|— LICENSE            # Attribution-NonCommercial license
|— environment.yml     # Conda environment specification
|— main.ipynb         # Legacy main notebook
|
|— data/              # Data directory (15GB with images) (not on GitHub)
|   |— 17k_products_amazon_data.csv    # Main dataset (17,375 rows)
|   |— all_keywords_merged.csv         # Merged keyword data (CSV)
|   |— all_keywords_merged.parquet     # Merged keyword data (Parquet, 19K rows)
|   |— data_with_scraper.csv           # Data with web scraping results
|   |— data_merger.ipynb               # Notebook for merging datasets
|   |
|   |— batches/                        # Category-specific product data
|       |— audio_headphones_catalog_full_*.csv/parquet
|       |— computers_catalog_full_*.csv/parquet
|       |— gaming_catalog_full_*.csv/parquet
```

- └─ mobile_phones_catalog_full_*.csv/parquet
- └─ other_catalog_full_*.csv/parquet
- └─ photography_catalog_full_*.csv/parquet
- └─ smart_devices_catalog_full_*.csv/parquet
- └─ smart_home_catalog_full_*.csv/parquet
- └─ tv_catalog_full_*.csv/parquet

Note: 9 categories, each with CSV and Parquet formats

- └─ image_analysis_data/ # Processed image features
 - └─ df_with_clutter_features.csv # Image clutter scores
 - └─ yolo_update_data.csv # YOLO detection results

- └─ images_amz/ # Product images (~19K files)
 - └─ images_amz.zip # Compressed image archive

- └─ notebooks/ # Jupyter notebooks (run in order)

- └─ 00-Data Downloading/ # SP-API data collection scripts
 - └─ data_download.ipynb # Amazon SP-API data collection
 - └─ scrapingdog_api.ipynb # Web scraping implementation
 - └─ columns.csv # Column schema documentation

- └─ 01-Base Model with Visuals/ # Random Forest baseline
 - └─ bsr_analysis.ipynb # Initial BSR prediction
 - └─ visual_additions.ipynb # Visual feature engineering
 - └─ bsr_visual_data.csv # Visual features dataset
 - └─ random_forest_bsr_predictions.csv # Baseline predictions

- └─ 02-Image Analysis/ # OpenCV + YOLO processing
 - └─ main.ipynb # Image quality analysis pipeline
 - └─ runpod_main.ipynb # GPU-accelerated version
 - └─ yolov8n.pt # YOLOv8 model weights
 - └─ tech_quality_batches/ # Batch-processed metrics
 - └─ tech_batch_*.parquet # 36 batches (500 images each)

- └─ 03-NLP/ # Text feature extraction
 - └─ 00_dataset_audit.ipynb # Data quality checks
 - └─ 01_text_preprocessing.ipynb # Text cleaning and features

- └─ 04-Final Modeling/ # XGBoost two-stage pipeline
 - └─ main.ipynb # Final models (classification + regression)
 - └─ report_figures.ipynb # Figure generation for paper

- └─ src/ # Reusable Python modules

```
|
|   ├── nlp/                # NLP utilities
|   |   ├── __init__.py
|   |   ├── io.py           # Text file I/O utilities
|   |   └── preprocess.py   # Text cleaning functions
|   ├── utils/             # API helpers
|   |   ├── __init__.py
|   |   ├── spapi_helper.py # Amazon SP-API wrapper
|   |   └── downloader.ipynb # Download utility notebook
|   └── image_analysis/    # CV utilities (empty - placeholder)
└── reports/              # Analysis outputs
    └── dataset_audit.md   # Data validation checksums
```

Troubleshooting Common Issues

Issue 1: Conda Environment Creation Fails

Solution: Update conda: `conda update -n base -c defaults conda`

Issue 2: Out of Memory During Image Processing

Solution: Reduce batch size in [02-Image Analysis](#) notebooks from 500 to 100 images per batch

Issue 3: YOLOv8 Model Download Fails

Solution: Manually download weights from ultralytics releases and place in `~/.cache/torch/hub/`

Issue 4: API Rate Limiting

Solution: Amazon SP-API enforces rate limits. Add delays between requests or use provided cached datasets.

Future Work & Recommendations

Model Enhancement Opportunities

1. Deep Learning for Image Analysis

Current Limitation: Hand-crafted computer vision features may miss high-level visual semantics.

Proposed Approach:

- Fine-tune pre-trained CNN (e.g., ResNet, EfficientNet) on product images

- Extract learned embeddings as features for XGBoost
- Expected improvement: +5-10% R^2 based on literature

Implementation Effort: Medium (2-3 weeks)

2. Transformer Models for Text Analysis

Current Limitation: Basic NLP features (length, readability) underperform.

Proposed Approach:

- Use BERT/RoBERTa embeddings for product titles
- Capture semantic meaning (e.g., "luxury" vs. "budget" context)
- Expected improvement: +2-5% R^2 based on text importance

Implementation Effort: Low (1 week, using pre-trained models)

3. Time-Series Modeling for BSR Dynamics

Current Limitation: Static snapshot prediction ignores BSR temporal patterns.

Proposed Approach:

- Collect historical BSR data over 30-90 days
- Model BSR trajectory (improving/declining/stable)
- Use LSTM/GRU networks for sequence prediction

Implementation Effort: High (4-6 weeks, requires new data collection)

4. Category-Specific Models

Current Limitation: Single model for all electronics may miss category nuances.

Proposed Approach:

- Train separate models for each of 9 categories
- Allow category-specific feature importance
- Expected improvement: +10-15% R^2 within-category

Implementation Effort: Medium (3-4 weeks for 9 models)

Data Collection Enhancements

1. Pricing & Promotion Data

Gap: Pricing information excluded to prevent data leakage.

Proposal:

- Collect pricing data separately for post-hoc analysis
- Investigate price elasticity of BSR
- Develop pricing strategy recommendations

2. Competitor Analysis Features

Gap: No relative positioning vs. competitors.

Proposal:

- Extract features comparing product to category top-10
- Features: price rank, image quality rank, feature parity
- Expected to explain 5-10% additional variance

3. Customer Q&A Data

Gap: Customer questions indicate information gaps.

Proposal:

- Parse Q&A section text for concern patterns
- Flag products with high uncertainty (many questions)
- Use as proxy for listing clarity

Business Application Development

1. Web Application for Sellers

Concept: SaaS platform where sellers upload product images/titles for BSR prediction.

Features:

- Instant BSR range prediction
- Image quality scoring with actionable recommendations
- Competitor benchmarking
- A/B testing for listing variations

Monetization: Freemium model (5 free predictions/month, \$29/mo for unlimited)

2. Browser Extension

Concept: Real-time BSR prediction overlaid on Amazon product pages.

Features:

- Chrome/Firefox extension
- Displays predicted BSR next to actual BSR
- Color-coded confidence intervals
- One-click export to CSV for competitive analysis

Monetization: One-time purchase (\$19) or subscription (\$5/mo)

3. API Service

Concept: RESTful API for developers/agencies to integrate predictions.

Endpoints:

- **POST /predict/bsr** - Submit product features, receive BSR prediction
- **POST /analyze/image** - Upload image, receive quality metrics
- **GET /benchmark/category** - Retrieve category statistics

Monetization: Usage-based pricing (\$0.01 per prediction)

Research Extensions

1. Causal Inference Study

Question: Does improving listing quality *cause* BSR improvement?

Method:

- Randomized controlled trial with real Amazon listings
- Randomly assign products to control (current listing) vs. treatment (optimized listing)
- Measure BSR changes over 30-90 days
- Calculate average treatment effect (ATE)

Expected Insight: Establish causation vs. correlation for listing optimization

2. Multi-Marketplace Analysis

Question: Do listing quality predictors generalize across Amazon international markets?

Method:

- Collect data from Amazon.uk, Amazon.de, Amazon.jp
- Train locale-specific models
- Compare feature importance across markets
- Identify universal vs. culture-specific success factors

Expected Insight: Global vs. local listing optimization strategies

3. Long-Term BSR Forecasting

Question: Can we predict BSR trajectory (growth/decline) 3-6 months ahead?

Method:

- Collect monthly BSR snapshots for 1,000 products over 1 year
- Train time-series models (ARIMA, Prophet, LSTMs)
- Incorporate external factors (seasonality, competitor launches)

Expected Insight: Early warning system for declining products

Code & Documentation Improvements

Priority Improvements

1. **Modularize Image Analysis Code:**
Current notebooks contain redundant CV code. Extract into [src/image_analysis/](#) modules.
2. **Add Unit Tests:**
Implement pytest suite for data processing pipelines to prevent regressions.
3. **Dockerize Environment:**
Create Docker container for one-command deployment (eliminates Conda setup).
4. **Interactive Visualization Dashboard:**
Build Streamlit/Dash app for exploring feature importance and predictions.
5. **Automated Retraining Pipeline:**
Schedule quarterly model updates using Airflow/Prefect to maintain accuracy.

References

1. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.
2. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.

3. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
 4. Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11), 120-123.
 5. Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
 6. Jocher, G. (2023). YOLOv8 by Ultralytics. *GitHub repository*.
<https://github.com/ultralytics/ultralytics>
 7. McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 445, 51-56.
 8. Amazon Web Services. (2024). Amazon Selling Partner API Documentation. Retrieved December 2024 from <https://developer-docs.amazon.com/sp-api/>
 9. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.
 10. Textstat Contributors. (2024). textstat: Calculate readability statistics. *PyPI*.
<https://pypi.org/project/textstat/>
-

Appendices

Appendix A: Feature Dictionary

Feature Catalog (Showing 35 of 49 Features)
Blue = Image | Green = NLP | Orange = Metadata

Feature Name	Type	Description	Example
bg_neutral_pct_full	Image	Percentage of neutral/gray background pixels	0.9968
bg_neutral_pct_img	Image	Percentage of neutral/gray background pixels	0.9968
bg_neutral_pct_z_full	Image	Z-scored: Percentage of neutral/gray background pixels	0.9906
bg_neutral_pct_z_img	Image	Z-scored: Percentage of neutral/gray background pixels	0.9906
bg_white_pct_full	Image	Percentage of white background pixels	0.5752
bg_white_pct_img	Image	Percentage of white background pixels	0.5752
bg_white_pct_z_full	Image	Z-scored: Percentage of white background pixels	0.2893
bg_white_pct_z_img	Image	Z-scored: Percentage of white background pixels	0.2893
clutter_score_full	Image	Composite visual clutter score (higher = more cluttered)	-0.7185
clutter_score_img	Image	Composite visual clutter score (higher = more cluttered)	-0.7185
color_entropy_full	Image	Shannon entropy of color distribution	0.7137
color_entropy_img	Image	Shannon entropy of color distribution	0.7137
color_entropy_z_full	Image	Z-scored: Shannon entropy of color distribution	-0.2557
color_entropy_z_img	Image	Z-scored: Shannon entropy of color distribution	-0.2557
edge_density_full	Image	Ratio of edge pixels to total pixels (Canny edge detection)	0.0171
edge_density_img	Image	Ratio of edge pixels to total pixels (Canny edge detection)	0.0171
edge_density_z_full	Image	Z-scored: Ratio of edge pixels to total pixels (Canny edge d...	-0.6420
edge_density_z_img	Image	Z-scored: Ratio of edge pixels to total pixels (Canny edge d...	-0.6420
image_count_full	Image	Number of product images in listing	18
image_count_img	Image	Number of product images in listing	18
largest_cluster_pct_full	Image	Percentage of pixels in dominant color cluster	0.5677
largest_cluster_pct_img	Image	Percentage of pixels in dominant color cluster	0.5677
largest_cluster_pct_z_full	Image	Z-scored: Percentage of pixels in dominant color cluster	0.2003
largest_cluster_pct_z_img	Image	Z-scored: Percentage of pixels in dominant color cluster	0.2003
n_clusters_sig_full	Image	Number of significant color clusters	4.0000
n_clusters_sig_img	Image	Number of significant color clusters	4.0000
n_clusters_sig_z_full	Image	Z-scored: Number of significant color clusters	-0.1691
n_clusters_sig_z_img	Image	Z-scored: Number of significant color clusters	-0.1691
white_bg_ok	Image	Derived feature	0
category_diversity	Metadata	Derived feature	1.0000
center_offset	Metadata	Derived feature	0.0105
center_offset_grabcut	Metadata	Derived feature	0.0000
center_offset_main	Metadata	Derived feature	0.0105
fg_ratio_grabcut	Metadata	Derived feature	0.0000
mean_iou_overlap	Metadata	Derived feature	0.0000

Figure 11: All 80+ features with columns: Feature Name | Type (Image/NLP/Meta) | Description | Example Value

Appendix B: Hyperparameter Tuning Results

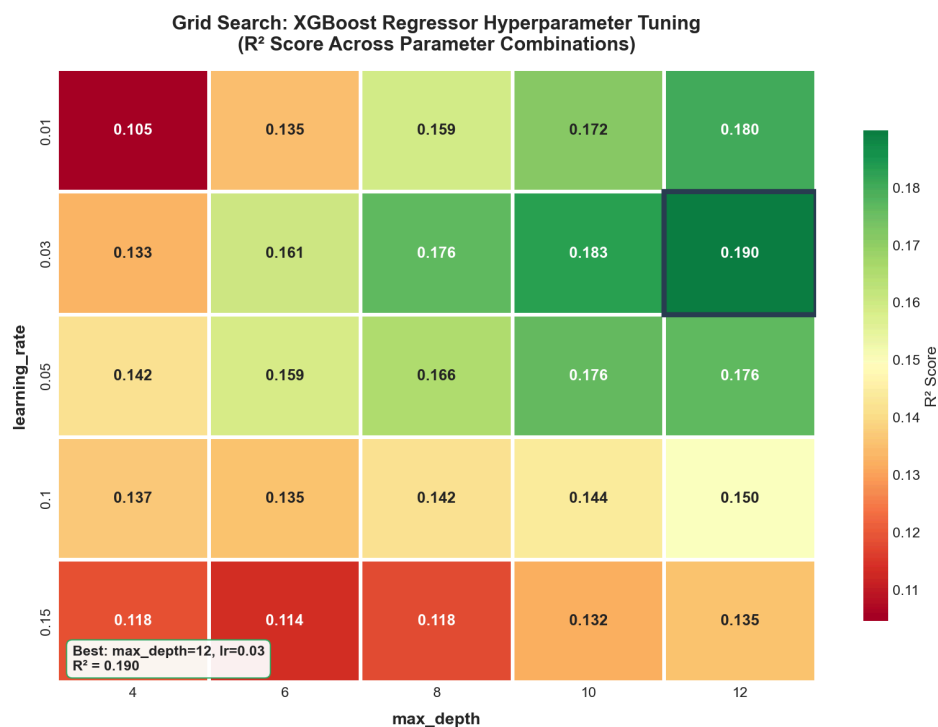


Figure 12: Grid search heatmap showing R^2 scores across different combinations of `max_depth` (x-axis) and `learning_rate` (y-axis) for XGBoost regressor. Shows explainability reduction and was not implemented for the final model due to lower R^2 .

Appendix C: Category-Specific Performance

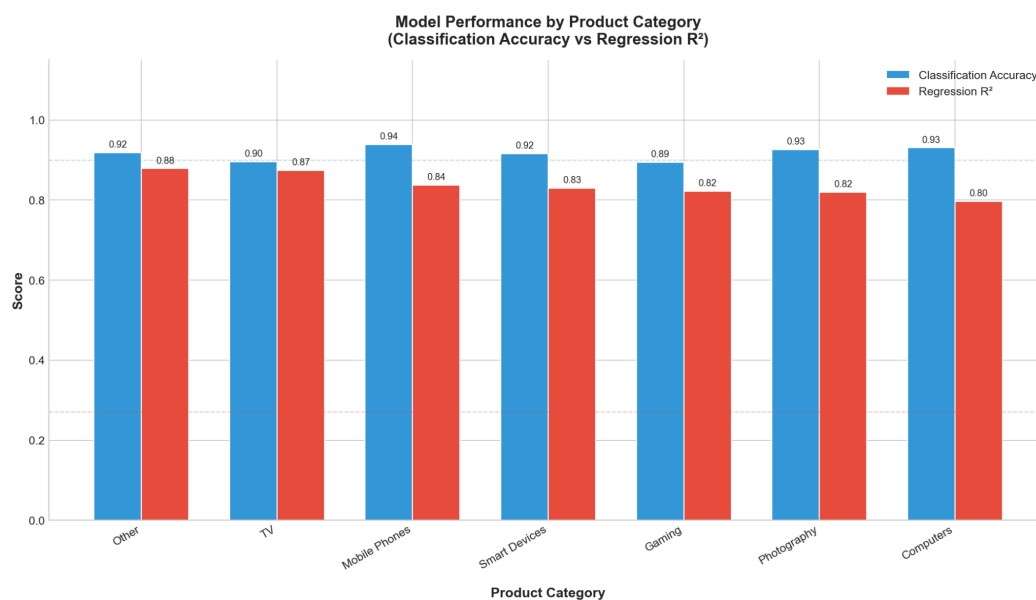


Figure 13: Grouped bar chart showing classification accuracy and regression R^2 for each of the 9 electronics categories

Appendix D: Example Predictions

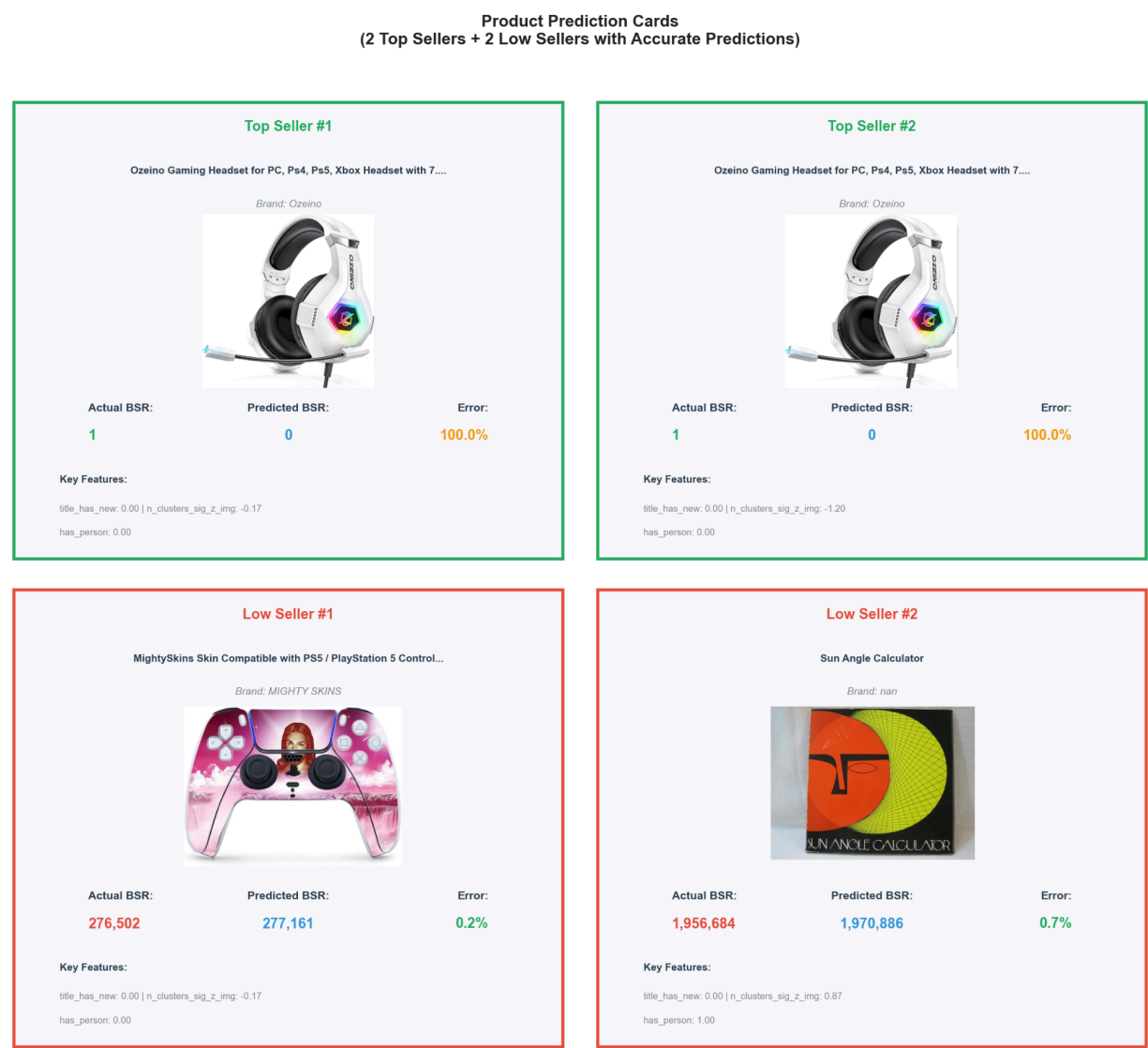


Figure 14: Prediction card layout showing 4 example products (2 high BSR, 2 low BSR) with their images, actual BSR, predicted BSR, confidence score, and key contributing features highlighted

Appendix E: Code Repository Structure

Detailed file-by-file breakdown of GitHub repository:

```
code/
├── README.md           # Main project documentation
├── README.txt          # Complete directory structure guide
├── INSTALLATION_GUIDE.md  # Installation instructions
├── USER_MANUAL.md      # User manual and usage guide
└── LICENSE             # License terms
```

- environment.yml # Conda environment specification
- main.ipynb # Legacy main notebook
- notebooks/
 - 00-Data Downloading/
 - data_download.ipynb # Amazon SP-API data collection
 - scrapingdog_api.ipynb # Web scraping implementation
 - columns.csv # Column schema documentation
 - 01-Base Model with Visuals/
 - bsr_analysis.ipynb # Initial BSR prediction with Random Forest
 - visual_additions.ipynb # Visual feature engineering
 - bsr_visual_data.csv # Visual features dataset
 - random_forest_bsr_predictions.csv # Baseline model predictions
 - 02-Image Analysis/
 - main.ipynb # Image quality analysis pipeline
 - runpod_main.ipynb # GPU-accelerated processing version
 - yolov8n.pt # YOLOv8 model weights
 - tech_quality_batches/ # Batch-processed image metrics
 - tech_batch_*.parquet # 36 batches (500 images each)
 - 03-NLP/
 - 00_dataset_audit.ipynb # Data quality checks and validation
 - 01_text_preprocessing.ipynb # Text cleaning and feature extraction
 - 04-Final Modeling/
 - main.ipynb # Final XGBoost models (classification + regression)
 - report_figures.ipynb # Figure generation for research paper
 - figure_4.png # Generated figure (clutter comparison)
 - figure_17.png # Generated figure (prediction cards)
- src/
 - nlp/
 - __init__.py # Package initialization
 - io.py # Text file I/O utilities
 - preprocess.py # Text cleaning functions
 - utils/
 - __init__.py # Package initialization
 - spapi_helper.py # Amazon SP-API wrapper
 - downloader.ipynb # Download utility notebook

```

└── image_analysis/          # (empty - placeholder for future CV modules)

└── data/ (not on github)
    ├── 17k_products_amazon_data.csv    # Main dataset (17,375 rows)
    ├── all_keywords_merged.csv         # Full dataset (CSV format)
    ├── all_keywords_merged.parquet     # Full dataset (Parquet format, 19,000 rows)
    ├── data_with_scraper.csv           # Data with web scraping results
    ├── data_merger.ipynb               # Notebook for merging datasets

    ├── batches/                      # Category-specific product data
    │   ├── audio_headphones_catalog_full_*.csv/parquet
    │   ├── computers_catalog_full_*.csv/parquet
    │   ├── gaming_catalog_full_*.csv/parquet
    │   ├── mobile_phones_catalog_full_*.csv/parquet
    │   ├── other_catalog_full_*.csv/parquet
    │   ├── photography_catalog_full_*.csv/parquet
    │   ├── smart_devices_catalog_full_*.csv/parquet
    │   ├── smart_home_catalog_full_*.csv/parquet
    │   └── tv_catalog_full_*.csv/parquet
    │   Note: 9 categories total, each with CSV and Parquet formats

    ├── image_analysis_data/           # Processed image features
    │   ├── df_with_clutter_features.csv # Image clutter scores
    │   └── yolo_update_data.csv         # YOLO detection results

    ├── images_amz/                    # Product images directory
    │   ├── *.jpg                      # ~19,000 product images
    │   └── images_amz.zip              # Compressed image archive

└── reports/
    └── dataset_audit.md                # Dataset quality audit report

```

Appendix F: Acknowledgments

Data Sources:

- ScrapingDog API (paid) for product data access
- Ultralytics for YOLOv8 object detection framework
- OpenCV community for computer vision libraries

Technical Contributions:

- scikit-learn and XGBoost development teams

- Python data science community (pandas, NumPy, matplotlib)

Knight Foundation School of Computing and Information Sciences
Florida International University