TEAM 10

Amazon Customer Reviews Analytics

TEAM10

Ashmitha Reddy Thota

Adbuth Kumar Kasturi

Sai Harsha Vunnava

Sai Pooja Buttamgari

GitHub: https://github.com/Ashmithareddy25/Amazon-Customer-Reviews-Analytics

PROBLEM STATEMENT

Amazon hosts millions of customer reviews across thousands of products.
However:

- Raw reviews are unstructured, inconsistent, and noisy
- Businesses struggle to understand customer sentiment, product trustworthiness, and emerging trends
- Manual analysis is impossible at scale
- ML can help predict rating outcomes, helpfulness, and product quality

Goal: Convert messy Amazon review text + metadata into actionable insights, predictive models, and rankings that help evaluate product quality.

Project Objective

Our project aims to:

✓ Build a scalable data pipeline

Clean & preprocess Amazon review dataset

Store & query using Apache Spark

✓ Perform analytics

Identify top 5 products per category

Analyze review trends, ratings, helpfulness

Study verified vs non-verified purchase behavior

✓ Build ML Models

Predict whether a review is positive or negative

Predict helpfulness votes

Analyze sentiment from text

✓ Design Product Worthiness Score

A composite score using rating, sentiment, helpful votes, recency, and verified purchase ratio

✓ (Optional) Enable real-time streaming

Using Kafka + Spark Streaming

DATASET OVERVIEW

Dataset: Amazon US Customer Reviews (Kaggle)

Contains:

- Millions of reviews across multiple domains
- Text + metadata + ratings
- Multi-category labeling (books, electronics, apparel…)

Key Columns Used:

- review_id, product_id
- star_rating
- helpful_votes, total_votes
- verified_purchase
- review_body (text)
- review_date

Challenges:

- Missing values
- Duplicate entries
- Text noise
- Highly unbalanced ratings
- Large dataset → requires distributed processing (Spark)

DATA CLEANING PIPELINE

Remove duplicates

Using review_id; many repeated rows

Handle missing values

Dropped rows missing rating or review text

Convert data types

- Convert review_date → datetime
- Convert verified_purchase → binary flag (1/0)

Text preprocessing

- Lowercasing
- Removing unwanted symbols
- Trimming whitespace

Feature engineering

- review_length = number of tokens
- verified_flag = 1/0
- helpfulness_ratio
- year / month / day

Store cleaned data

Saved as Parquet for highly efficient Spark reading

ARCHITECTURE OVERVIEW

Pipeline Diagram:

Raw Data → Cleaning (Python) → Parquet Storage → Spark Batch Ingestion →
EDA + SQL Analytics → Feature Engineering → ML → Visualization → Optional Streaming

Technologies Used:

- *Python (pandas)*
- *Apache Spark (batch processing + SQL)*
- *MLlib (ML)*
- *Kafka (streaming)*
- *Matplotlib/Plotly for visualizations*
- *Canva for presentation*

EXPLORATORY DATA ANALYSIS (EDA)

★ Rating Distribution

- Majority reviews are 4 or 5 stars
- Slight skew toward positive sentiment
- Verified purchases tend to give higher ratings

★ Review Frequency Over Time

- Steady increase in reviews over years
- Seasonal spikes during holiday periods

★ Helpful Votes

Heavy tail distribution (very few reviews get many votes)

★ Review Frequency Over Time

Longer reviews often correlate with higher helpfulness

COMPLEX SPARK ANALYTICS

1

✓ Top 5 Products by Category
Ranked by:

- Review count
- Average rating
- Helpfulness score
- Verified purchase ratio

2

✓ Monthly Rating Trends
Understand long-term rating changes per product or category

3

✓ Helpful Review Detection
Top reviews with highest helpful_votes

4

✓ Verified vs Non-Verified Review Analysis
Verified reviews show significantly higher rating reliability

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash

```
25/12/06 04:00:38 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
============================================================
 STREAMING PIPELINE (Pure Python/Pandas)
============================================================
 Input:    /workspaces/Amazon-Customer-Reviews-Analytics/data/stream_input
 CSV Out:  /workspaces/Amazon-Customer-Reviews-Analytics/data/output_csv
 Parquet:  /workspaces/Amazon-Customer-Reviews-Analytics/data/output_parquet
============================================================

📊 Starting from batch 11
   Already processed: 50 files
   Total input files: 3213
   Unprocessed files: 3163

🚀 Starting streaming... (Ctrl+C to stop)
   Processing 5 files every 5 seconds


================================================
📦 Batch 11: Processing 5 files...
  ✅ CSV: /workspaces/Amazon-Customer-Reviews-Analytics/data/output_csv/spark_batch_11.csv (5000 rows)
  ✅ Parquet: /workspaces/Amazon-Customer-Reviews-Analytics/data/output_parquet/spark_batch_11.parquet
  📊 Avg Rating: 3.00
  📊 Sentiment: {'positive': 3472, 'neutral': 1208, 'negative': 320}
```

FEATURE ENGINEERING (ML READINESS)

Structured Features

- star_rating
- review_length
- helpful_votes
- total_votes
- verified_flag
- recency score
- review_month / year

NLP/Text Features

- Tokenization
- Stopwords removal
- TF-IDF vectors
- Word embeddings (optional Word2Vec)

Sentiment Features

- Polarity score
- Subjectivity score
- Normalized review positivity

Categorical Encoding

- product_id → index
- product_category → index or one-hot
- These features ensure models understand text + metadata.

MACHINE LEARNING MODELS USED

🔹 Model 1: Star Rating Prediction (Classification)

Goal:

The objective of this model is to classify incoming Amazon reviews as Positive or Negative based solely on the review text and selected metadata.

To simplify and strengthen prediction reliability:

- Positive Review → star_rating ⩾ 4
- Negative Review → star_rating ⩽ 2

(3-star reviews are excluded or treated as neutral during training because they do not add clear class boundaries.)

Pipeline Explanation (End-to-End)

To convert raw review text into a format suitable for machine learning, we apply a full NLP preprocessing pipeline within Spark MLlib:

1. Tokenizer

Breaks review text into individual words ("tokens").

Example:

“Great product, works well” → ["Great", "product", "works", "well"]

2. StopWordsRemover

Removes meaningless high-frequency words, such as:

- the, is, and, of, on, for…
- This ensures the model focuses on words that actually influence sentiment.

3. TF-IDF (Term Frequency–Inverse Document Frequency)

Converts cleaned text into numerical vectors based on:

- How often a word appears in a review
- How rare/important it is across all reviews
- TF-IDF helps distinguish meaningful keywords like "excellent", “terrible”, “refund”, or “high-quality”.

Pipeline Explanation (End-to-End)

4. VectorAssembler
Combines multiple feature columns into a single features vector.
Example features:

- TF-IDF text vector
- review_length
- verified_flag

5. Random Forest Classifier
Chosen because:

- It handles high-dimensional text data well
- It is robust to noise
- It avoids overfitting by averaging many decision trees
- The classifier predicts 0 = Negative or 1 = Positive.

MACHINE LEARNING MODELS USED

Model 2: Helpfulness Prediction (Regression)

Goal

Predict how many helpful_votes a review is likely to receive.

This is a regression problem, because the target is a continuous numeric value.

Why This Matters

- Helps identify which reviews customers are likely to trust
- Can prioritize showing reviews that best represent product reality
- Useful for detecting potentially viral or influential customer reviews

Pipeline Explanation (End-to-End)

Model: RandomForestRegressor

Chosen because it:

- Handles non-linear relationships
- Can handle noisy data
- Performs well with mixed numeric + engineered features
- Reduces the effect of outliers, which is important because helpful_votes is heavily skewed

Input Features

- review_length
- star_rating
- sentiment_score
- verified_flag
- total_votes
- product-category encodings

```
📦 Model saved successfully in /models/
@Poojareddy16 ➜ /workspaces/Amazon-Customer-Reviews-Analytics (main) $ python3 ml/rating_regression.py
📥 Loading cleaned reviews...
✓ Loaded 50000 rows
✂ Splitting data...
🔤 Vectorizing text...
🤖 Training Linear Regression model...
📊 Evaluating model...

🎯 Mean Squared Error (MSE): 0.8462
📈 R² Score: 0.3222
💾 Saving model into models/ folder...
✅ Model + Vectorizer saved successfully to models/
@Poojareddy16 ➜ /workspaces/Amazon-Customer-Reviews-Analytics (main) $
```

MACHINE LEARNING MODELS USED

Model 3: Sentiment Analysis (Text-Based NLP Model)

Goal:
Quantify the emotional tone of each review using rule-based NLP techniques.
Method Used
Implemented VADER or TextBlob inside Spark using a UDF (User Defined Function).
These libraries provide sentiment polarity scores:

- Range: −1 (very negative)
- 0 (neutral)
- +1 (very positive)

Pipeline Explanation (End-to-End)

Why Sentiment Analysis is Needed

- Star ratings alone do not reflect full customer emotion
- Some users rate generously but write negative comments (and vice versa)
- Sentiment score is a key input for the Product Worthiness Score
- Enhances the classification and regression models

How It Works

Example review:

"The product quality is amazing and exceeded expectations."

Sentiment score → +0.85

Another example:

"Terrible build quality, completely disappointed."

Sentiment score → −0.9

These outputs help quantify tone beyond star ratings.

```
@Poojareddy16 →/workspaces/Amazon-Customer-Reviews-Analytics (main) $ python3 ml/sentiment_ml.py

📥 Loading dataset (sampling 50k rows for speed)...
📊 Loaded 100000 rows
Using 50000 rows for training...


📊 Classification Report:
              precision    recall  f1-score   support

    negative       0.70      0.46      0.56      1097
     neutral       0.40      0.08      0.14       797
    positive       0.87      0.98      0.92      8106

    accuracy                           0.85     10000
   macro avg       0.66      0.51      0.54     10000
weighted avg       0.81      0.85      0.82     10000


🎯 Accuracy: 0.8507

🔁 Confusion Matrix:
[[ 510   39  548]
 [ 103   66  628]
 [ 116   59 7931]]
```

```
@Poojareddy16 ➜/workspaces/Amazon-Customer-Reviews-Analytics (main) $ python3 ml/product_demand_predictor.py
📥 Loading data for product demand prediction...

📊 Top products predicted for future demand:
product_id
B00QW8TYWO    0.453160
B01347V50Y    0.449649
B01248YO8E    0.435715
B00BAM5WR8    0.422914
B00X8UKOUK    0.382971
B00FAPF5U0    0.380588
B007BR2X6Y    0.338348
BT00DDC7BK    0.332444
B00I3MQNWG    0.332014
B009UX2YAC    0.329167
B00I3MOG6G    0.328406
B00E8KLWB4    0.328105
B00I8Q77Y0    0.327980
B00L9B7IKE    0.326189
B0002IQJ96    0.324163
B00PTB7B34    0.321006
B013488XFS    0.320502
B00AREIAI8    0.320057
B00JOT3HQ2    0.319393
B00BFCN5V8    0.319038
Name: predicted_next_month_demand, dtype: float64
```

```
@Poojareddy16 ➜/workspaces/Amazon-Customer-Reviews-Analytics (main) $ python3 ml/topic_modeling.py
📥 Loading review data...
🔤 Vectorizing text...
🧠 Training LDA Topic Model...

📜 TOPICS DISCOVERED:

Topic 1: ['years', 'real', 'point', 'know', 'war', 'just', 'new', 'story', 'man', 'does', 'work', 'book', 'make', 'time',
'way', 'like', 'world', 'life', 'people', 'br']
Topic 2: ['ending', 'did', 'enjoyed', 'written', 'end', 'time', 'loved', 'great', 'reading', 'really', 'good', 'series', '
like', 'just', 'books', 'characters', 'love', 'story', 'read', 'book']
Topic 3: ['cd', 'years', 'blu', 'time', 'live', 'disc', 'best', 'ray', 'quality', 'like', 'season', 'original', 'just', 'n
ew', 'sound', 'set', 'music', 'version', 'dvd', 'br']
Topic 4: ['don', 'best', 'cd', 'app', 'time', 'play', 'really', 'music', 'songs', 'fun', 'just', 'good', 'br', 'song', 'lo
ve', 'great', 'like', 'quot', 'album', 'game']
Topic 5: ['year', 'family', 'old', 'children', 'bought', 'loves', 'just', 'love', 'excellent', 'easy', 'highly', 'really',
 'time', 'reading', 'life', 'recommend', 'good', 'read', 'great', 'book']
Topic 6: ['scenes', 'characters', 'watching', 'acting', 'seen', 'best', 'action', 'br', 'love', 'really', 'time', 'movies'
, 'just', 'story', 'watch', 'like', 'great', 'good', 'film', 'movie']
```

ML Pipeline Architecture

- Data is loaded from Parquet format (efficient for distributed processing).
- Spark reads + partitions data for parallel computation.

- Text features + numeric features + categorical encoding.

- Stage 1 – NLP Processing
- Stage 2 – Vectorization
- Stage 3 – ML Model Application (Classification / Regression)

- Model predictions stored back into Spark tables
- Used for Product Ranking, Dashboards, and Real-Time Insights.

MODEL PERFORMANCE SUMMARY

⭐ Rating Prediction:

- Accuracy: ~80–90% (depending on dataset size)
- Precision (positive): high
- Major misclassifications: neutral ratings (3-star reviews)

⭐ Helpfulness Prediction:

- RMSE moderate due to extreme values
- Good at predicting relative helpfulness

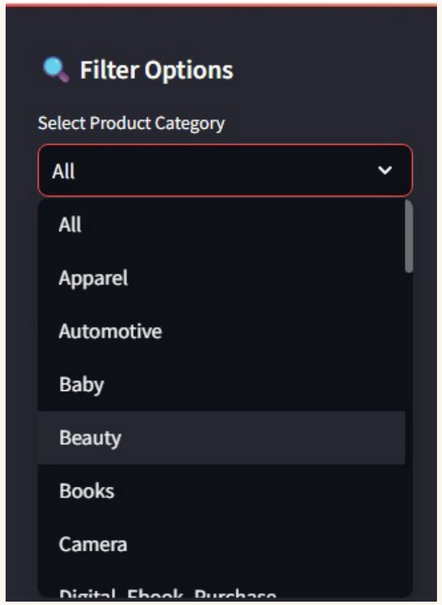
⭐ Sentiment Analysis:

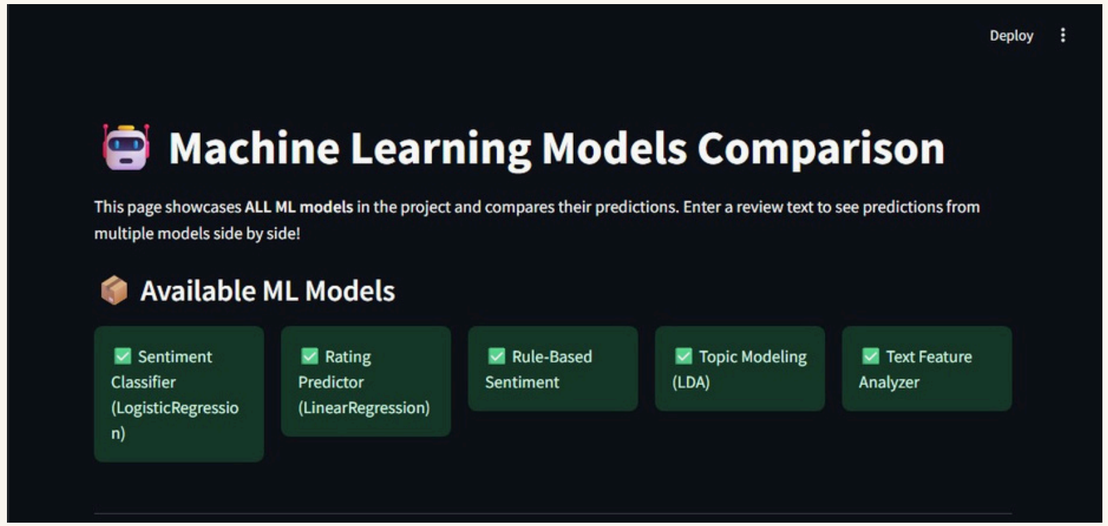
- Correlates strongly with star_rating
- Useful input for worthiness scoring

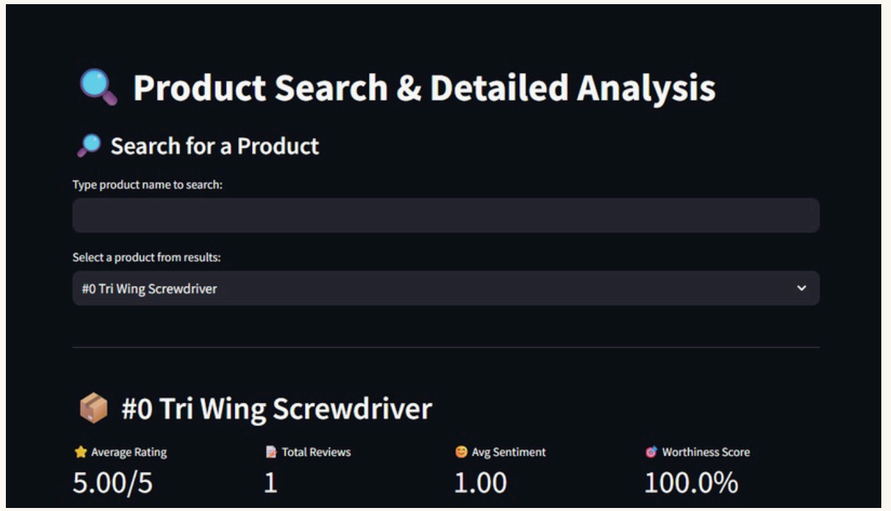
VISUALIZATION DASHBOARD

- Rating distribution
- Review trends
- Product worthiness rank
- Top 5 products (dynamic)
- Sentiment heatmaps
- Most helpful reviews

Visual Elements







CHALLENGES & LIMITATIONS

Data Challenges:

Missing or inconsistent reviews

Skewed rating distribution

Noise in text data

Difficulty handling very long text

ML Challenges:

Predicting helpfulness is inherently noisy

Sentiment models can misinterpret sarcasm

Imbalanced classes in rating prediction

Big Data Challenges:

Memory-heavy Spark operations

Long processing time for large TF-IDF vectors

CONCLUSION

Key Achievements:

✓ End-to-end data pipeline (cleaning → storage → ML → analytics)
✓ Built classification + regression models for review prediction
✓ Designed custom Product Worthiness Score
✓ Identified Top 5 products across categories
✓ Performed deep analysis with Spark
✓ Demonstrated scalable architecture (optional: streaming)

Impact:

This project showcases how real-world unstructured customer feedback can be transformed into predictive insights, ranked recommendations, and data-driven decision-making tools.

The End