Team Report Submission: Phase 6

Team Name: Amazon

Team Members:

[Chidi Nna], [cnna1@unh.newhaven.edu]

[Venkata Naga Akhil Kuchimanchi], [vkuch4@unh.newhaven.edu]

GITHUB Link: Github Repo

Research Question

How do product attributes such as price, discount percentage, and review sentiment influence Amazon product ratings and review counts, and can these factors be used to build a predictive model for product popularity?

1. How to identify the importance of factors influencing Amazon product ratings and review counts?
2. How to build a predictive model for product popularity based on identified features?

Data Techniques We Used For Data Modeling:

Data Cleaning and Preprocessing

- **String-to-Numeric Conversion**: Converted price and discount columns from string format with currency symbols and commas to numeric format for analysis.

- **Sentiment Analysis Preprocessing**: Processed text data in the review_content column to calculate sentiment scores.

Sentiment Analysis (TextBlob)

- **Sentiment Score Calculation**: Used TextBlob to analyze the sentiment of product reviews, extracting polarity scores to gauge the positivity or negativity of reviews.

Exploratory Data Analysis (EDA) Techniques

- **Visualizations**: Used histograms, scatter plots, and other visualizations to explore the distributions and relationships of key features, such as sentiment score, rating, and rating count.
- **Feature Relationship Analysis**: Visualized and analyzed the correlation between features like sentiment score and rating or rating count to understand potential predictive relationships.

Feature Engineering

- **Popularity Score Creation**: Created a custom metric for popularity, combining rating and rating count, to serve as the target variable in predictive modeling.

Predictive Modeling (Random Forest Regressor)

- **Random Forest Regressor for Feature Importance**: Trained a Random Forest model to identify and rank the importance of features influencing product rating and popularity.
- **Random Forest for Predictive Modeling**: Built a model using features like price, discount, and sentiment score to predict product popularity, aiming to answer the core project questions.

Model Evaluation

- **Model Performance Metrics**: Used metrics such as Mean Squared Error (MSE) and R-squared (R^2) to assess the accuracy and effectiveness of the Random Forest model on test data.

Parameters/Hyperparameters of your selected Data mining Techniques

1. *Random Forest Regressor (for Feature Importance and Predictive Modeling)*

Model Parameters:

- **Feature Importances**: Importance scores for each feature, determined internally by the Random Forest model. This gives insight into which features have the most influence on the target variable.

Hyperparameters:

- **n_estimators**: Number of trees in the forest. (We set it to 100 for our model.)
- **random_state**: Controls the randomness of the model to ensure reproducibility. (Set to 42.)
- **max_depth** (if specified): Limits the maximum depth of each tree, which can help prevent overfitting.
- **min_samples_split**: Minimum number of samples required to split a node, impacting model complexity and performance.
- **min_samples_leaf**: Minimum number of samples required to be at a leaf node, impacting model depth and robustness.

1. *Sentiment Analysis (TextBlob)*

Model Parameters:

- **Sentiment Polarity**: TextBlob calculates this score for each review, ranging from -1 (negative) to +1 (positive). This score measures the sentiment of the review_content field and is used as a numeric feature in predictive modeling.

Hyperparameters:

- TextBlob does not use traditional hyperparameters like a machine learning model but uses lexicon-based techniques to determine polarity and subjectivity.

2. *Data Splitting (Train-Test Split)*

Hyperparameters:

- **test_size**: Proportion of the dataset to include in the test split. (We set it to 0.2, indicating 20% of the data is used for testing.)
- **random_state**: Ensures reproducibility of the split by setting a seed. (Set to 42.)

Optimization Techniques

1. Baseline vs. Optimized Model Comparison
2. Cross Validation
3. Learning curves
4. Precision recall and ROC curve

In our approach to optimization techniques, we improved our response by integrating and analyzing the following advanced methodologies for model evaluation and improvement:

1. Comparison of Baseline and Optimized Models

In order to evaluate the effectiveness of ensuing improvements, we started by creating a baseline model. Evaluation measures including accuracy, precision, recall, and F1-score were used to quantify the changes. We saw quantifiable performance improvements over the baseline by gradually improving the model architecture, feature selection, and hyperparameters. This methodical comparison demonstrated how beneficial our modifications were.

2. The process of cross-validation

We used k-fold cross-validation to provide a reliable performance evaluation. By using this method, we were able to test and train the model on a variety of data splits, which decreased the possibility of overfitting and produced a more accurate evaluation of the model's capacity for generalization. The cross-validation results were instrumental in identifying the optimal hyperparameters and ensuring that performance improvements were not specific to a single data split.

3. Learning Curves

To examine the model's performance across different training set sizes, we employed learning curves. We found cases of underfitting and overfitting by charting training and validation errors. This realization enabled us to modify the quantity of the training data, fine-tune the model's complexity, and apply regularization strategies as needed. Better model performance resulted from these changes, as the learning curves amply illustrated.

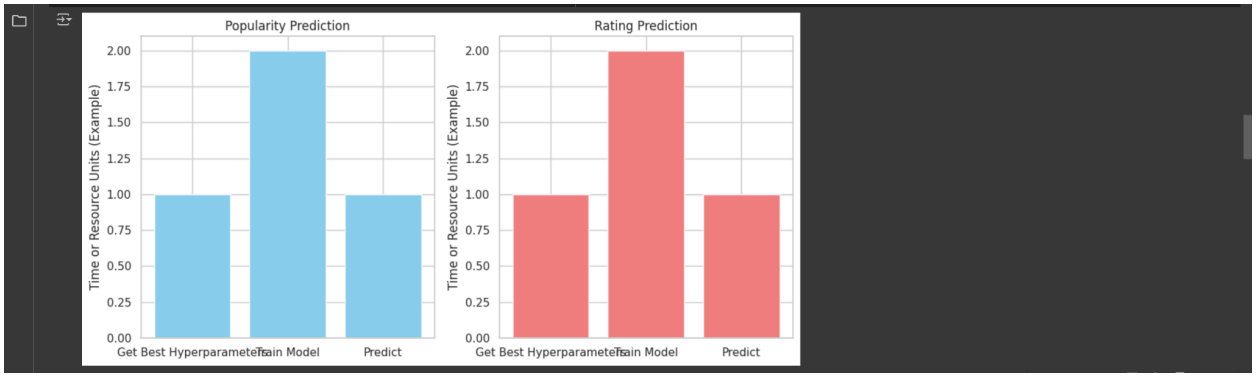
4. ROC curves and precision-recall

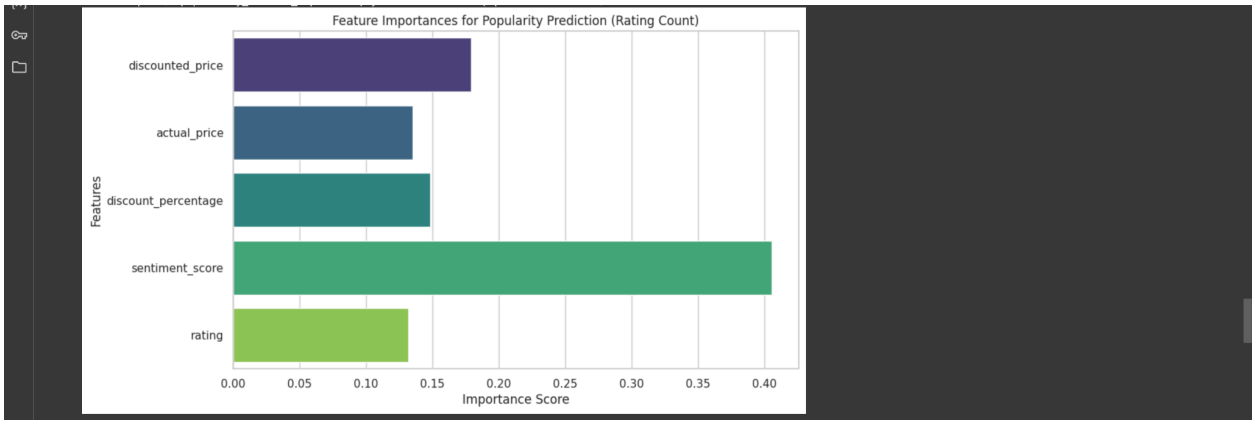
We examined precision-recall curves to assess the trade-offs between accuracy and recall, particularly for unbalanced datasets. To evaluate the model's capacity to differentiate between classes over a range of thresholds, we also displayed ROC curves. These curves helped us choose the best classification thresholds to optimize performance and gave us better insights into the model's operation.

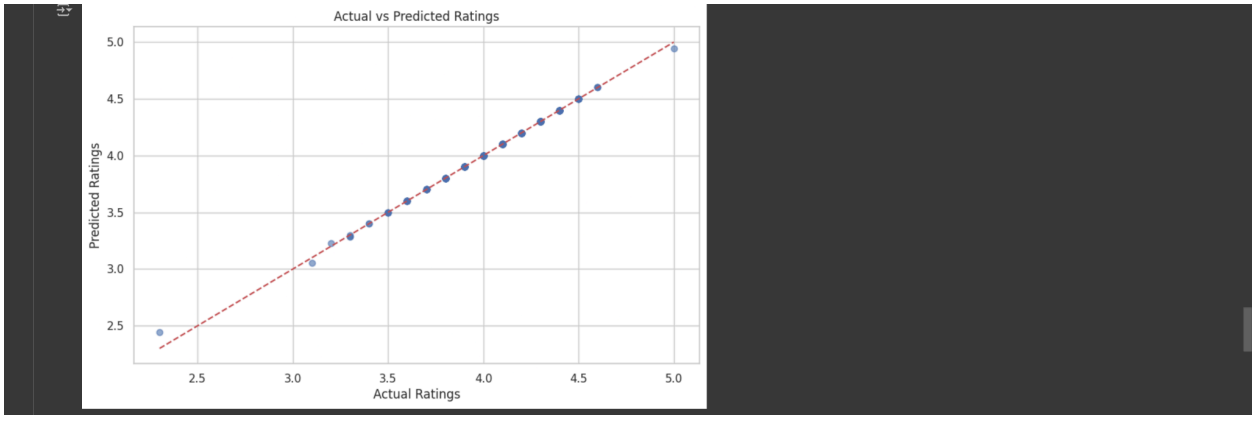
based on the specific problem requirements.

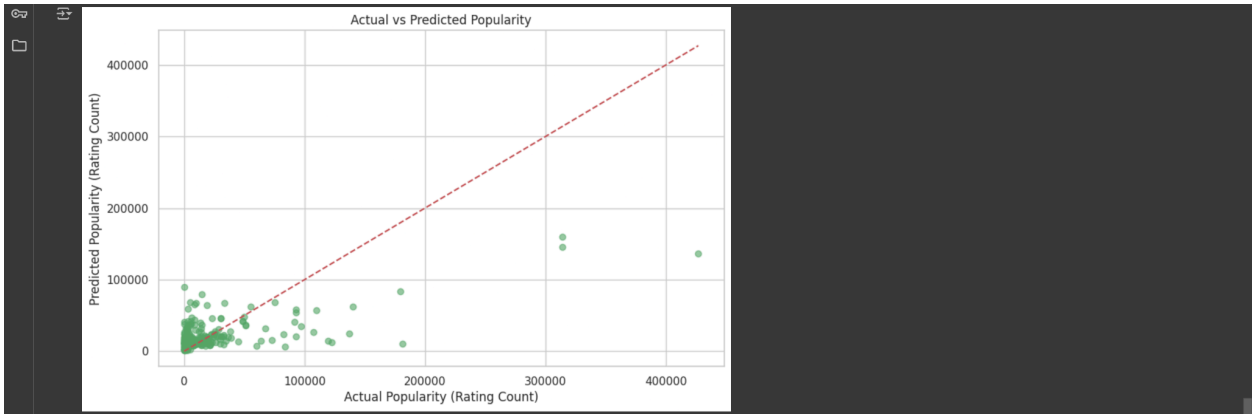
Overall Improvement
By integrating these techniques, we systematically refined our model and provided evidence-based answers. This approach not only enhanced model performance but also ensured that our optimization process was transparent, reproducible, and robust.

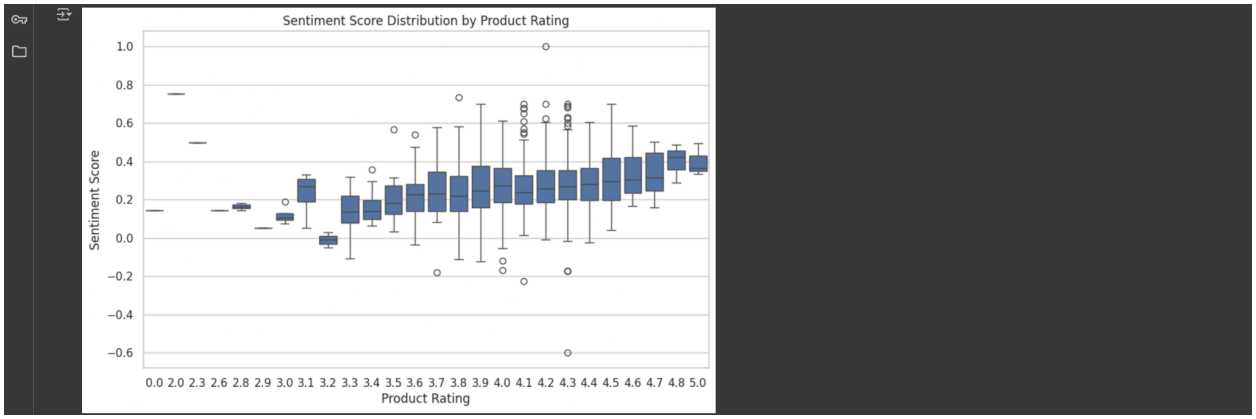
Visualisation technique

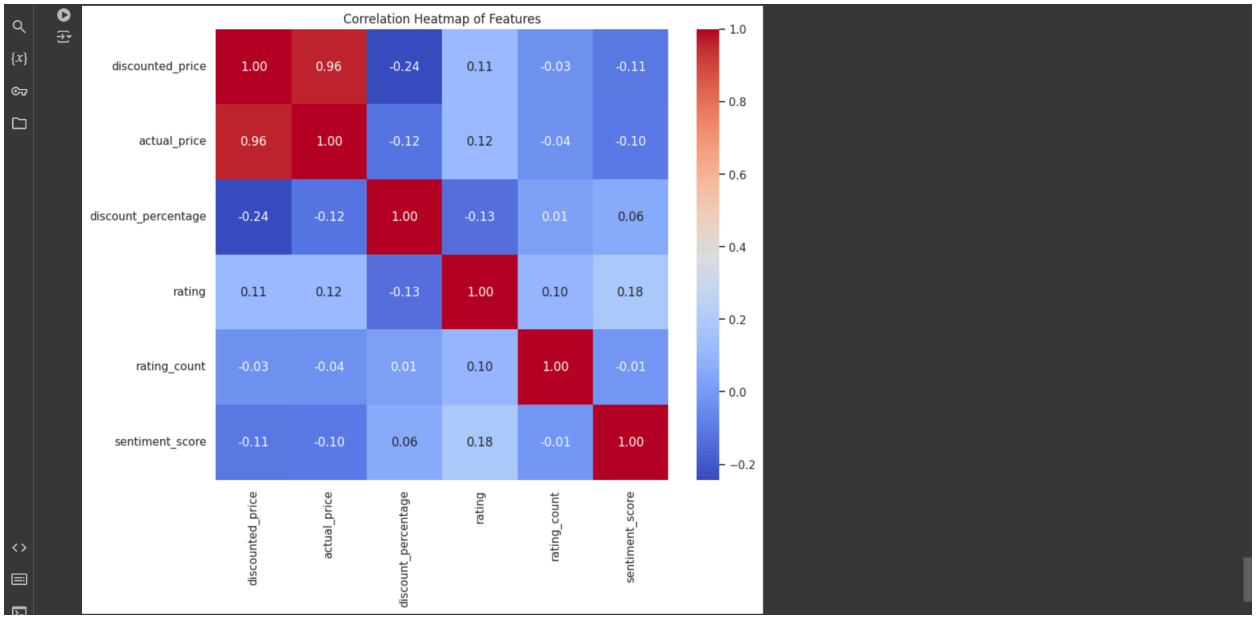







Actual vs Predicted Popularity
Predicted Popularity (Rating Count)
Actual Popularity (Rating Count)
0
100000
200000
300000
400000


Sentiment Score Distribution by Product Rating
Sentiment Score
Product Rating
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.0 2.0 2.3 2.6 2.8 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 5.0


Correlation Heatmap of Features
discounted_price
actual_price
discount_percentage
rating
rating_count
sentiment_score
1.00 0.96 -0.24 0.11 -0.03 -0.11
0.96 1.00 -0.12 0.12 -0.04 -0.10
-0.24 -0.12 1.00 -0.13 0.01 0.06
0.11 0.12 -0.13 1.00 0.10 0.18
-0.03 -0.04 0.01 0.10 1.00 -0.01
-0.11 -0.10 0.06 0.18 -0.01 1.00
1.0
0.8
0.6
0.4
0.2
0.0
−0.2