Dataset Description and Project Overview

Project Title: E-Commerce Product Rating Prediction and Recommendation System

Project Objective:

The goal of this project is to predict product ratings for various e-commerce products based on features such as product price, rating count, and other derived features. This prediction model can be used to suggest products with high ratings to customers, thus assisting e-commerce platforms in generating personalized recommendations.

Dataset Description:

The dataset contains information about products available on various e-commerce platforms such as Shophive, HomeShopping, and PriceOye. The dataset includes several features that describe the product attributes, such as product price, rating, rating count, and other relevant details. Below is a description of the columns in the dataset:

product\_name:

Type: String

Description: The name or title of the product. This column is used to identify the product and will not be used in model training (though it can be useful for generating recommendations).

product\_price:

Type: Float (numeric)

Description: The price of the product. It contains values in a currency format, which are cleaned by removing currency symbols like 'Rs.' and commas. After cleaning, it is normalized for model use.

product\_ratings:

Type: Float (numeric)

Description: The average rating of the product, typically on a scale of 1 to 5 stars. This is the target variable for prediction in the project.

rating\_count:

Type: Integer

Description: The number of ratings given to the product. It is cleaned to extract the numerical value (e.g., "8 Ratings" becomes 8). This feature is used to assess the popularity or credibility of the product.

original\_price (if available):

Type: Float (numeric)

Description: The original price of the product before any discounts. This is used to calculate a new feature, the discount\_rate, which indicates the percentage discount on the product.

discount\_rate:

Type: Float (numeric, derived)

Description: A calculated feature representing the percentage discount on the product. It is derived by subtracting the cleaned product\_price from original\_price and dividing by original\_price. This feature helps assess the attractiveness of the product from a price perspective.

rating\_interaction:

Type: Float (numeric, derived)

Description: A calculated feature that is the product of product\_ratings and rating\_count. This interaction feature represents product popularity — the more ratings a product has and the higher the rating, the more likely it will be recommended.

price\_zscore:

Type: Float (numeric, derived)

Description: A Z-score that represents how far the product\_price is from the mean, indicating whether a price is unusually high or low compared to others.

rating\_count\_zscore:

Type: Float (numeric, derived)

Description: A Z-score that indicates whether the rating\_count is unusually high or low.

Data Cleaning Process:

Missing Values: Missing data in columns like product\_price and rating\_count was imputed using the median value.

Outliers: Outliers in product\_price and rating\_count were identified using the Interquartile Range (IQR) method. Any value beyond 1.5 times the IQR from the first and third quartile was replaced with the median value.

Text Cleaning: The product\_price was cleaned by removing the currency symbol 'Rs.' and commas, ensuring the value is ready for numerical processing. The rating\_count was stripped of any non-numeric characters to retain only the number.

Feature Engineering:

Several new features were created to enhance the model's performance:

discount\_rate: A measure of how much discount the product has compared to its original price.

rating\_interaction: A feature that captures the interaction between the number of ratings and the average rating to represent product popularity.

price\_zscore: Z-scores were calculated to identify outliers in the product price.

rating\_count\_zscore: Similarly, Z-scores were calculated for rating\_count to highlight outlier values.

Modeling:

The project used a deep learning approach with a neural network model built in TensorFlow/Keras. The model architecture consists of the following layers:

Input Layer: The input layer accepts features such as product\_price, product\_ratings, rating\_count, discount\_rate, and rating\_interaction.

Hidden Layers: There are two hidden layers with 128 and 64 neurons respectively, using the ReLU activation function.

Dropout Layers: Dropout layers were added after each hidden layer with a 30% dropout rate to prevent overfitting.

Output Layer: The output layer consists of a single neuron since this is a regression problem (predicting a numerical rating).

Loss Function: The model uses Mean Squared Error (MSE) as the loss function, which is commonly used for regression tasks.

The model was trained for 10 epochs, and the training data was fed in batches of 32 samples.

Evaluation Metrics:

Root Mean Squared Error (RMSE) was used to evaluate the performance of the model. RMSE is a commonly used metric for regression tasks, and it indicates the average difference between predicted and actual values.

Recommendation System:

Once the model was trained, it was used to generate recommendations based on predicted ratings for products. The top N products with the highest predicted ratings were recommended to users. This is a simple recommendation system that relies on product ratings predicted by the model, along with other features such as discount rate and interaction score.

Project Enhancements and Future Work:

Model Improvements:

We could explore more advanced models such as XGBoost or Random Forest for comparison.

Hyperparameter tuning can be done using Grid Search or Randomized Search to improve model accuracy.

Feature Engineering:

Additional features could be derived from textual data like product\_name, such as product categories or sentiment analysis of product descriptions.

Time-related features such as product listing date or seasonality could be incorporated.

Recommendation System:

A more sophisticated recommendation system could be implemented using collaborative filtering, matrix factorization, or content-based filtering methods.

User-specific recommendations could be added by incorporating user profiles, browsing history, and past purchases.

Scalability:

To scale this project for large e-commerce datasets, tools like Apache Spark or Dask for parallel processing and distributed computing could be integrated.

Real-time Predictions:

For a production environment, deploying the model to make real-time product recommendations using a REST API framework like Flask or FastAPI would be beneficial.

Conclusion:

This project demonstrates the use of machine learning, specifically deep learning models, to predict product ratings and create a recommendation system for e-commerce platforms. By utilizing product attributes and derived features, we can recommend products based on predicted ratings, enhancing user experience and potentially driving sales on e-commerce platforms.