Understanding the Data

My approach to the star prediction model was straightforward yet thorough. First, I analyzed each data field to understand its nuances, identify potential pitfalls, and determine its usefulness for the model. I then examined how each field interacted with others to develop a comprehensive picture of the data's structure and relationships. Let me provide an example to illustrate this approach.

The ProductID field, described as a "unique identifier for the product," suggested each value would be unique. To verify, I used ChatGPT to generate code that confirmed each ProductID was indeed unique. However, I wanted to see if multiple users were reviewing the same product, so I asked ChatGPT for additional code, which revealed that while some products had multiple reviews, most had only one. Knowing this helped me understand that ProductID could introduce potential biases in my model if not carefully managed.

Similarly, the "Score" field represents ratings between 1 and 5. I considered how this data could be misleading if a user consistently gives high or low scores. For example, if a user rates every show a 5, we need to weigh that differently compared to a user who is more stringent with their ratings. This discrepancy is akin to the difference between an amateur movie critic and a professional reviewer. To ensure the data is as reliable and objective as possible, such factors need to be accounted for. Failing to do so could bias the model towards predicting 5-star ratings due to their higher frequency.

Another factor is brand reputation; popular movies are often rated highly by fans. Conversely, a show that is objectively good but not popular might have only one bad review, skewing the perception of its quality. There's also temporal bias depending on seasons or expectations—for instance, Christmas movies are more likely to be rated during the holiday season, and the festive mood might influence higher ratings. Nostalgia bias can have a similar effect. Recognizing these biases, I aimed to examine the best, worst, and average cases of each field, understand their relationships, and train my model to mitigate these biases as much as possible.

Implementing the Model

Libraries and File Loading

Libraries were divided into data manipulation, machine learning, model selection, training, evaluation, and visualization categories. Loading the data files followed standard procedures, with initial visualizations of the columns we'd be working with.

Feature Engineering

Feature engineering involved filling missing values with defaults to avoid errors and manipulating or creating fields that would improve the model's performance. Here's a summary of the added features:

- **ProductID_encoded and UserID_encoded**: Converted ProductID and UserID to integer values to be used in machine learning, allowing the model to learn from these identifiers.
- **Helpfulness**: Calculated based on HelpfulnessDenominator, managing cases where the denominator could be 0 or show irregular patterns.
- **User_avg_score**: Calculated each user's average rating, helping to weigh their individual rating patterns.
- **Year**: Converted to a standard datetime format to capture temporal data accurately.

After feature engineering, I generated X_train.csv and X_submission.csv by processing the text and separating scores from the training data.

Sampling & Splitting the Data

For sampling and splitting into training and testing sets, I ensured that everything in X_train looked appropriate before performing the split.

Feature Selection and Modeling

For feature selection, I created a new split and developed models using:

- **Gradient Boosting for Structured Data:** This model helps with including class weights and addressing over- and under-sampling issues present in the data. It can also handle time fields like year, season, and product age, capturing shifts in rating patterns over time and helping the model recognize seasonal or lifecycle-related rating variations.
- **Transformers (BERT) for Text Data:** BERT is used for the text data in the "Summary" and "Text" fields. It captures the full context of phrases, allowing the model to interpret whether extreme language aligns with the user's actual rating. I utilized get_bert_embeddings to create the BERT embeddings in batches to prevent memory errors.

Model Evaluation

After feature engineering and model selection, I proceeded with model evaluation to measure its performance and accuracy based on the data transformations applied.