Jewelry Price Optimization with ML: Pricing Data to Refine Pricing Strategies

1. Importing and Cleaning Data

Data Import:

- Imported a dataset with 13 columns, including order details, product characteristics, and pricing information.

Initial Data Cleaning:

- Identified columns to eliminate: SKU_quantity, Category_ID, Brand_ID, Order_Id, and User_ID.
- Processed missing values using SimpleImputer with strategy='most_frequent'.

2. Data Preprocessing Pipelines

Pipeline Creation:

- Defined a function generate_pipeline to create preprocessing pipelines for different models.
- Pipelines included transformations for imputing missing values, one-hot encoding, and ordinal encoding for categorical features.

Pipeline Components:

- SimpleImputer for handling missing values.
- OneHotEncoder for encoding categorical features as one-hot vectors.
- OrdinalEncoder for encoding categorical features as ordinal values.

3. Model Selection and Training

Models Used:

- Linear Regression
- AdaBoost Regressor
- ExtraTrees Regressor
- CatBoost Regressor

Pipeline for Each Model:

- Constructed pipelines for each model incorporating preprocessing steps and model instantiation.
- Added TransformedTargetRegressor with PowerTransformer(method='box-cox') to handle target variable transformation.

4. Handling Missing Values in Model Training

- Integrated the preprocessing steps into the pipelines to ensure models could handle missing values in the dataset.

5. Model Training and Evaluation

Model Fitting:

- Trained each model pipeline on the training dataset X_train and y_train.

Error Handling:

- Implemented a try-except block to catch NotFittedError during model training, ensuring that the model fitting errors were appropriately handled.

Evaluation Function:

- Created functions generate_model_score and test_model to evaluate model performance using specified metrics (e.g., RMSE).

6. Error Fixes and Corrections

Parameter Corrections:

- Corrected parameter names and values to match the expectations of the models (e.g., corrected los_function to loss_function for CatBoostRegressor).

Function and Syntax Fixes:

- Fixed function names and syntax issues to ensure proper execution of code.

Final Adjustments:

- Ensured that the TransformedTargetRegressor instance was properly fitted before making predictions.
- Adjusted the error-handling code block to maintain correct indentation and functionality.

Conclusion

By implementing various machine learning models and preprocessing pipelines, this project aimed to optimize the pricing strategies for jewelry products. The steps included importing and cleaning the data, setting up preprocessing pipelines, training different models, handling missing values, and evaluating model performance. The integration of error handling and parameter corrections ensured robust model training and evaluation.