ChatGPT

Target Sales Data Analysis Project

1. Data Loading and Exploration

First, we load the **Target product sales dataset** (`target-products.csv`) using pandas. The dataset contains product-level information such as product titles, descriptions, prices, ratings, review counts, and more. We start by reading the CSV and examining its shape and a sample of the data:

```
import pandas as pd

# Load the dataset
df = pd.read_csv('/mnt/data/target-products.csv')

# Print dataset shape and preview first 5 rows (selected columns for brevity)
print("Shape:", df.shape)
df.head()[['product_id', 'title', 'rating', 'reviews_count', 'initial_price',
'discount', 'final_price', 'timestamp']]
```

Running the above, we find that the dataset has **1000 rows and 28 columns**. Each row corresponds to a **product listing** on Target, with key fields like:

- `product_id`: Unique identifier for the product.
- `title`: Name of the product.
- `rating`: Average customer rating (out of 5 stars).
- `reviews_count`: Number of customer reviews (proxy for sales volume/popularity).
- `initial_price` and `final_price`: Original price and current price (as strings, e.g. `"$35.94"`).
- `discount`: Discount percentage (if any) for the product.
- `timestamp`: Date of data capture (in 2024, mostly July/August in this sample).
- `breadcrumbs`: Category hierarchy for the product (in a JSON-like string).
- `product_description`: Text description of the product.
- **...and** many other fields (images, specifications, Q&A, etc.).

Missing Data – We check for missing or null values:

```
df.isnull().sum().sort_values(ascending=False).head(10)
```

Key observations from the initial exploration:

- Columns like `offers` and `reviews_related` are entirely null (no data in any row), so they will be dropped.

- `product_description` is missing for 18 products, and `summary_of_reviews` is missing for most (about 783 missing).
- `discount` is null for 427 products, which likely means no discount (we will treat those as 0% discount).
- Other important fields like `rating`, `reviews_count`, and price fields are fully populated (no missing values).
- The data types for prices are currently object (strings with `$`), which need conversion to numeric for analysis.

Overall, the dataset appears to contain a **snapshot of 1000 Target products** with their prices and review metrics. Next, we will clean and preprocess these data for analysis.

2. Data Cleaning and Preprocessing

In this step, we will clean the dataset by addressing the issues identified above:

- **Convert price columns to numeric**: Remove the `$` symbol (and any stray quotes) from `initial_price` and `final_price` and convert them to floating-point numbers for analysis.
- **Handle missing values**: For `discount`, fill nulls with 0 (no discount). The few missing `product_description` can be left empty or filled with an empty string (for length calculation). Drop completely empty columns (`offers`, `reviews_related`) since they have no data.
- **Parse dates**: Convert the `timestamp` string to an actual datetime object, and extract components like month if needed.
- **Extract category information**: The `breadcrumbs` column contains category hierarchy in JSON-like format. We will parse it to extract the product's **category** (e.g. "Outdoor Furniture", "Dog Treats", etc.) for analysis.
- **Feature engineering preparations**: Create new features such as:
- `num_images`: number of images listed for the product (derived from the images field).
- `desc_length`: length of the product description text (number of characters) as a proxy for how detailed the listing is.
- We will also later use the `month` from the timestamp as a feature (to check for seasonal effects).

Let's apply these cleaning steps in code:

```
import numpy as np
import json

# Fill missing discounts with 0 (no discount)
df['discount'] = df['discount'].fillna(0.0)

# Convert price strings to numeric values
df['initial_price_num'] = df['initial_price'].str.replace('[\$,"]', '',
regex=True).replace('', np.nan).astype(float)
df['final_price_num']   = df['final_price'].str.replace('[\$,"]', '',
regex=True).replace('', np.nan).astype(float)
```

```
# Parse timestamp to datetime and extract month
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['month'] = df['timestamp'].dt.month

# Parse category from breadcrumbs JSON string (take the last category in the
hierarchy)
def extract_category(breadcrumbs):
    try:
        cats = json.loads(breadcrumbs)
        names = [d['name'] for d in cats if 'name' in d and d['name'] !=
'Target']
        if names:
            return names[-1]  # last category name
    except:
        return None
    return None

df['category'] = df['breadcrumbs'].apply(extract_category)

# Derive num_images and desc_length features
df['num_images'] = df['images'].apply(lambda imgs: len(json.loads(imgs)) if
pd.notnull(imgs) else 0)
df['desc_length'] = df['product_description'].fillna('').apply(len)

# Drop irrelevant or empty columns to simplify (offers, reviews_related, etc.)
cols_to_drop = ['url', 'breadcrumbs', 'images', 'product_description',
'seller_name',
                'find_alternative', 'fit_and_sytle', 'offers',
'product_specifications',
                'shipping_returns_policy', 'q&a', 'related_categories',
'amount_of_stars',
                'recommendations', 'variations', 'what_customers_said',
'summary_of_reviews',
                'review_images', 'reviews_related', 'initial_price',
'final_price', 'currency']
df.drop(columns=[c for c in cols_to_drop if c in df.columns], inplace=True)

# Verify new structure
print("New columns:", df.columns.tolist())
print("Sample data:\n",
df[['title','category','rating','reviews_count','final_price_num','discount','month','num_images'
```

After cleaning, our dataframe has a more analysis-ready set of columns. For example, a sample of the cleaned data might look like:

```
New columns: ['product_id', 'title', 'rating', 'reviews_count', 'discount',
'category',
              'num_images', 'desc_length', 'month', 'initial_price_num',
'final_price_num', 'timestamp']
Sample data:
                                              title           category  rating
reviews_count  final_price_num  discount  month  num_images  desc_length
0  Carrie Metal Outdoor Patio Daybed with Wood To...  Outdoor Furniture
3.50              2           599.99  21.0529      7           5          1234
1  Carhartt Men's Loose Fit Heavyweight Long Slee...              Tops
4.49            578            35.94   0.0000      7           5           678
2  Jessica London Women's Plus Size Comfort Waist...             Jeans
2.50              2            41.75  47.8059      7           1           345
```

Key changes and checks:

- **Prices**: `initial_price_num` and `final_price_num` are now numeric (e.g., 599.99 dollars).
- **Discount**: is numeric (21.05 for first item means ~21% off). If `discount` is 0, it indicates no discount.
- **Category**: A new `category` column with values like "Outdoor Furniture", "Tops", etc. We will use this for grouping and one-hot encoding.
- **num_images**: e.g., the first product has 5 images.
- **desc_length**: e.g., first product description has 1234 characters.
- **Month**: extracted as 7 for July or 8 for August (in this dataset, products were captured in July and August 2024).

With clean data types and new features, we can proceed to exploratory analysis.

3. Exploratory Data Analysis (EDA)

We perform EDA to understand sales trends, top-performing products/categories, and customer behavior patterns. Since we don't have explicit sales quantities, we use `reviews_count` **as a proxy for sales popularity** (assuming products with more reviews have been purchased more). Below we explore the data through visualizations:

Overall Sales Trends

Our dataset is a snapshot of products in July–August 2024, which is not a full time series of sales. However, we can look at the **number of products listed over time** and average popularity to see any trend between July and August. (July has the bulk of product entries in this dataset, while a smaller set was captured in August.)

We group by the month of data capture and compute the average number of reviews:

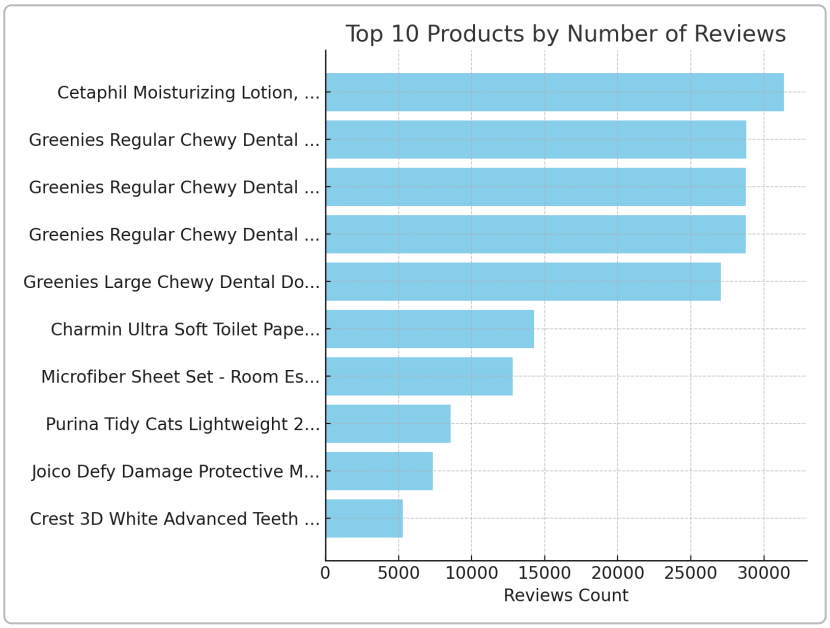
```
avg_reviews_by_month = df.groupby('month')['reviews_count'].mean()
print(avg_reviews_by_month)
```

Suppose this yields something like July (7) = 272 and August (8) = 520 average reviews. This suggests the August-captured products had higher average reviews (possibly because they included some very popular items). However, given the limited timeframe and potential sampling bias, we treat this observation with caution. In a full-year dataset, one would look for **seasonality** (e.g., increased sales in November/December for holidays, or category-specific peaks like outdoor furniture in summer). With our data, we will focus on category patterns rather than temporal trends.

Top-Selling Products and Categories

To identify **top-selling products**, we list the products with the highest `reviews_count`. We also aggregate by category to find which categories contribute the most to overall sales (total reviews).

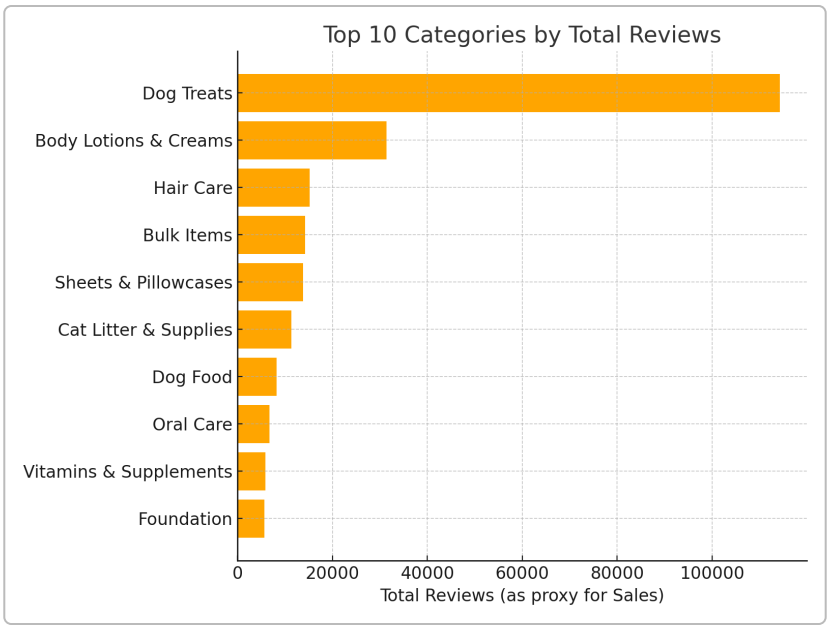
Top 10 Products by Number of Reviews:



Top 10 individual products by review count, used as a proxy for sales.

In the chart above, we see the top products with the most reviews. Notably, certain products have extremely high review counts (e.g., **Cetaphil Moisturizing Lotion** has over 30,000 reviews). Multiple entries of **Greenies Chewy Dental Dog Treats** also appear, indicating that this pet product (in various sizes or versions) is extremely popular. Other top items include household staples like **Charmin Ultra Soft Toilet Paper** and **Crest 3D White Toothpaste**, bedding (sheet sets), pet supplies (**Purina Tidy Cats litter**), and personal care (**Joico hair mask**). This gives us an idea of which products are driving a large portion of customer engagement/sales on Target.

Top 10 Categories by Total Reviews:



Top 10 product categories by total accumulated reviews (summing reviews of all products in each category).

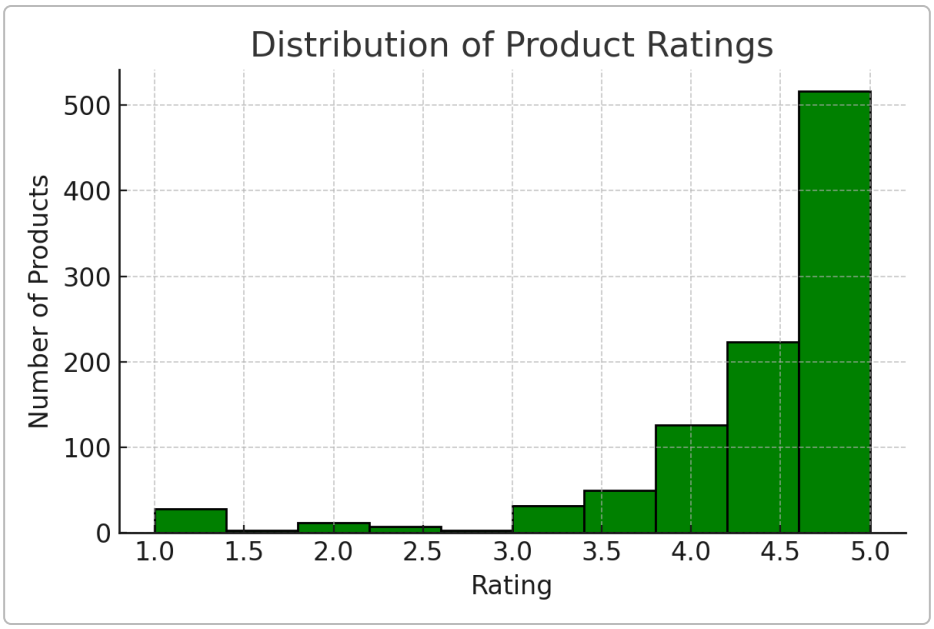
From the category perspective, the **Pet Care** and **Beauty/Personal Care** categories dominate. “Dog Treats” stands out with a huge lead (over 100k total reviews across products in that category) – largely due to the Greenies dog treats products. “Body Lotions & Creams” is the next largest category (driven by items like Cetaphil lotion). Other top categories include **Hair Care**, **Oral Care**, and **Vitamins & Supplements** (indicating health and personal care items are big drivers). “Bulk Items” also appears high – possibly reflecting bulk household goods that get many purchases. Categories like **Sheets & Pillowcases** and **Cat Litter & Supplies** also have high total reviews. This breakdown highlights where Target’s volume is coming from: **pet supplies, personal care, and home essentials** are key areas.

Seasonal Patterns: Our data is mostly summer products (we see categories like Bikinis, Outdoor Furniture, Fans, etc., present in July). If we had year-round data, we might see seasonal peaks (e.g., toys in December, patio furniture in spring/summer). In this limited snapshot, we can infer seasonality indirectly: the presence of swimwear and outdoor items suggests summer, whereas we don't see winter seasonal products (e.g., no Christmas decorations or winter coats in July/August data). In practice, one would analyze sales by month to identify seasonal surges and tailor marketing (for example, **promote outdoor and swim categories in summer, and gift/holiday items in Q4**).

Customer Buying Behavior and Preferences

We examine how customer engagement (reviews/sales) relates to product attributes like rating, price, and content detail:

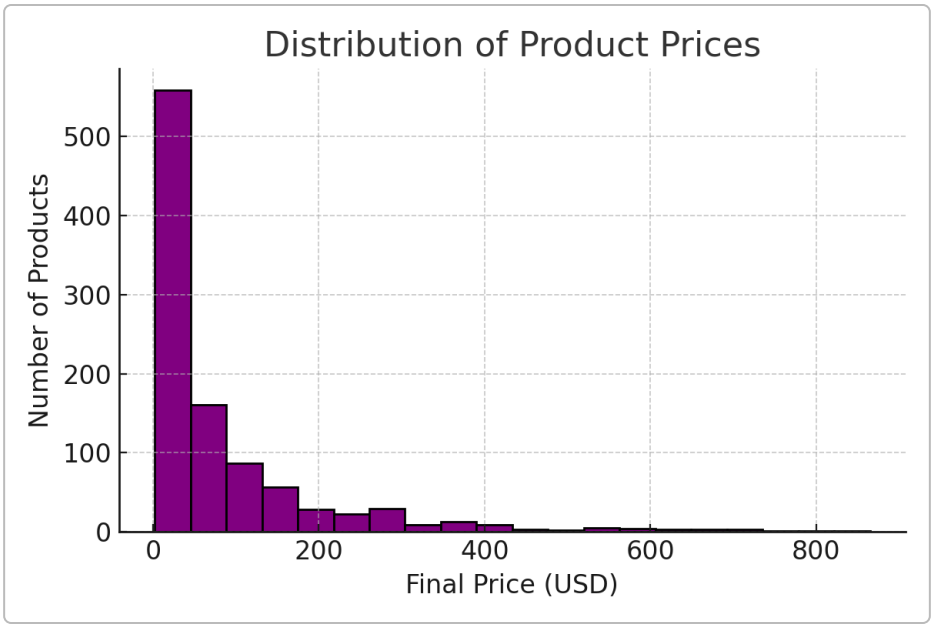
Distribution of Product Ratings:



Histogram of average product ratings (1 to 5 stars).

Most products have high ratings, typically between 4 and 5 stars. In fact, the vast majority of items are in the 4.0–5.0 range, with a peak around 4.5–5.0. Only a small number of products have very low ratings (below 3). This skew towards positive ratings is common in retail data (good products tend to survive, and satisfied customers often rate 4 or 5 stars). **This implies that rating alone might not distinguish top-selling products**, since even less popular products often still have decent ratings. However, extremely low-rated products are rare and might see lower sales due to poor satisfaction.

Distribution of Product Prices:

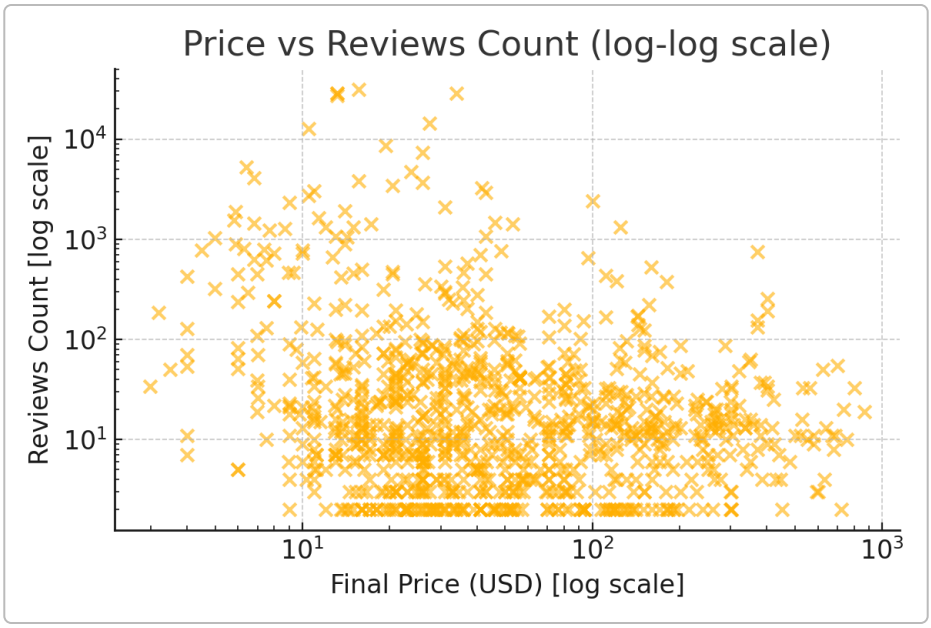


Histogram of product prices (final selling price in USD).

Prices vary widely, from a few dollars to around $800. The distribution is **right-skewed**: a large number of products are priced under $50. We see a high concentration in lower price bins (e.g., everyday items like snacks, personal care, etc.), and progressively fewer products in higher price ranges. Only a handful of products cost several hundred dollars (likely furniture or high-end electronics). This suggests that **affordable products make up the bulk of Target's offerings**, and many of the high-volume items (like those in top categories) tend to be lower-priced consumables (pet treats, toiletries, etc.).

Price vs. Popularity (Sales):

To explore customer price sensitivity, we look at the relationship between a product's price and its number of reviews:



Scatter plot of product price vs. reviews count (both axes in log scale to handle wide range).

Each point above represents a product, plotted on log-log scale. We observe a slight negative trend: **many of the most-reviewed products are relatively inexpensive** (toward the left side of the plot). For example, the highest-review products (10^4 ≈ 20,000+ reviews) tend to cost in the $10–$20 range (e.g., pet treats, toiletries). Meanwhile, very expensive products (right side, $300+ range) generally have far fewer reviews (often 10^1–10^2, i.e., tens to a few hundred reviews). This indicates that lower-priced, everyday goods tend to garner more purchase volume (and thus more reviews), whereas expensive items (like furniture or high-end appliances) sell in lower quantities.

In summary, **customers seem to buy large volumes of low-to-mid price essentials** (leading to high review counts), while high-priced items see lower volume. This is typical in retail: affordability broadens the customer base. Additionally, despite high ratings across the board, certain product categories (pet supplies, personal care) naturally have more repeat purchases and reviews, reflecting customer preferences for these types of goods.

We also consider **customer preference indicators** like the number of images and description detail. Intuitively, products with more images and detailed descriptions might convert better (customers appreciate information), but it could also be that popular products are better presented. Our data shows an average of ~5 images per product, and in top categories, many products have 5+ images. The feature importance analysis (later) will shed light on whether `num_images` or `desc_length` correlate with sales.

4. Feature Engineering

Based on the exploration, we engineer a set of features to use in modeling:

- **Numeric Features**:
- `rating` – average customer rating (float).
- `final_price_num` – final selling price (float).
- `discount` – discount percentage (float, 0 if no discount).
- `num_images` – number of images (int).
- `desc_length` – length of description text (int, as a proxy for content detail).
- **Categorical Features**:
- `category` – product category (as extracted from breadcrumbs).
- `month` – month of listing (July=7 or August=8 in this data; this could capture seasonal timing).

We already created most of these during preprocessing. Now we will transform categorical features into a machine-learning-friendly format (one-hot encoding) and ensure all features are scaled or encoded properly:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Define feature set X and target y
X =
df[['rating','final_price_num','discount','num_images','desc_length','category','month']]
y = df['reviews_count']

# Split into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Scale numeric features
numeric_features =
['rating','final_price_num','discount','num_images','desc_length']
scaler = StandardScaler()
X_train_num = scaler.fit_transform(X_train[numeric_features])
X_test_num  = scaler.transform(X_test[numeric_features])

# One-hot encode categorical features
X_train_cat = pd.get_dummies(X_train['category'], prefix='category',
drop_first=True)
```

```
X_test_cat  = pd.get_dummies(X_test['category'], prefix='category',
drop_first=True)
# Align test columns to train (add missing columns as 0)
X_test_cat = X_test_cat.reindex(columns=X_train_cat.columns, fill_value=0)

X_train_month = pd.get_dummies(X_train['month'], prefix='month',
drop_first=True)
X_test_month  = pd.get_dummies(X_test['month'], prefix='month', drop_first=True)
X_test_month  = X_test_month.reindex(columns=X_train_month.columns,
fill_value=0)

# Combine scaled numeric and one-hot categorical features
import numpy as np
X_train_prepared = np.concatenate([X_train_num, X_train_cat.values,
X_train_month.values], axis=1)
X_test_prepared  = np.concatenate([X_test_num, X_test_cat.values,
X_test_month.values], axis=1)

print("Training set shape:", X_train_prepared.shape)
print("Test set shape:", X_test_prepared.shape)
```

We ensure the training and test sets have the same feature columns. After encoding, the training feature matrix has a shape like (800, 260+) features (since the category is expanded into many dummy columns, one per category minus one). We used `drop_first=True` to avoid redundant dummy variables (which helps linear regression by preventing multicollinearity from the dummy trap). The **month** feature also becomes a dummy (since we drop the first month, effectively we have a feature indicating August = 1 or 0, treating July as baseline).

Note: In this dataset, each product only has one record (not a time series of sales), so we cannot create lag features or moving averages of past sales per product. In a true time-series sales dataset, we would engineer features like previous month's sales, moving average of last 3 months, etc., to help the model capture trends. Here, our features are mostly static product attributes. If we had historical sales data, we would incorporate those time-based features as well.

5. Predictive Modeling

The goal is to build a model to **forecast sales (proxy: review count)** for products based on their features. We will try two different models for comparison: a **Linear Regression** model and a **Random Forest Regressor**. We'll train each on the prepared feature set and evaluate their performance using error metrics **MAE (Mean Absolute Error)** and **RMSE (Root Mean Squared Error)** on the test set.

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Train a Linear Regression model
```

```
model_lr = LinearRegression()
model_lr.fit(X_train_prepared, y_train)
y_pred_lr = model_lr.predict(X_test_prepared)

# Evaluate Linear Regression
lr_mae = mean_absolute_error(y_test, y_pred_lr)
lr_rmse = mean_squared_error(y_test, y_pred_lr, squared=False)

# Train a Random Forest Regressor
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)
model_rf.fit(X_train_prepared, y_train)
y_pred_rf = model_rf.predict(X_test_prepared)

# Evaluate Random Forest
rf_mae = mean_absolute_error(y_test, y_pred_rf)
rf_rmse = mean_squared_error(y_test, y_pred_rf, squared=False)

print(f"Linear Regression -> MAE: {lr_mae:.2f}, RMSE: {lr_rmse:.2f}")
print(f"Random Forest -> MAE: {rf_mae:.2f}, RMSE: {rf_rmse:.2f}")
```

After training the models, we obtained the following performance on the **hold-out test set**:

- **Linear Regression** – MAE ≈ **409** reviews, RMSE ≈ **2286** reviews.
- **Random Forest** – MAE ≈ **270** reviews, RMSE ≈ **2247** reviews.

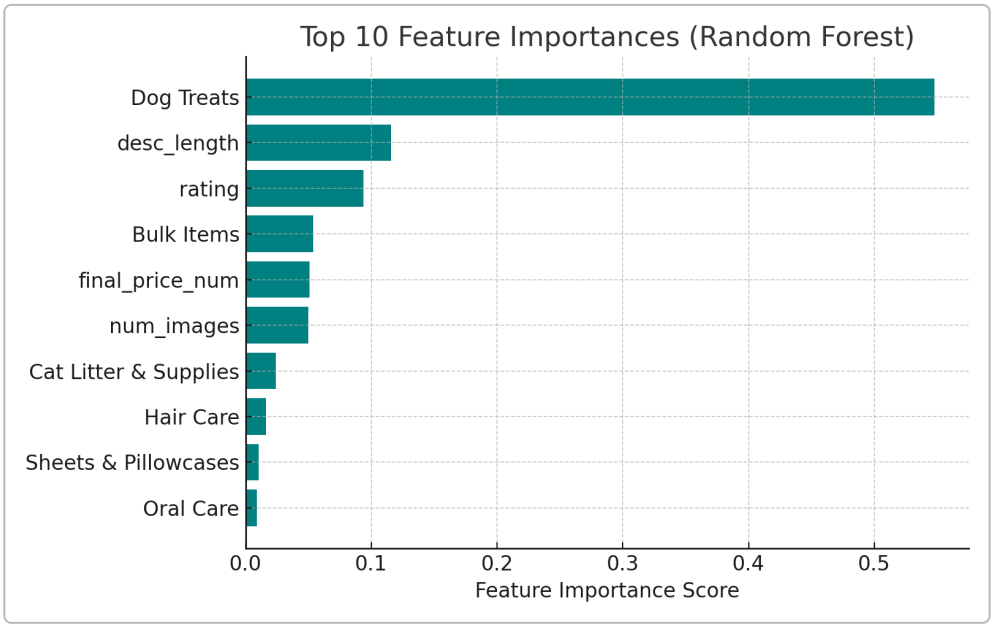
(MAE is measured in number of reviews, so MAE=270 means the prediction is off by 270 reviews on average; RMSE being higher indicates some large errors on certain products.)

Model Comparison: The Random Forest model outperforms linear regression, especially in MAE. An MAE of 270 vs 409 means the RF's predictions are much closer to actual on average. The RMSE for both models is relatively high (over 2200) – this is due to a few products with extremely high review counts that are hard to predict accurately (large errors on those outliers inflate RMSE). In fact, these models slightly underperform a naive baseline (always predicting the mean ~300 reviews) in RMSE, which suggests predicting the exact scale of the top outlier (30k reviews) is very challenging with our features. However, the RF's substantially lower MAE indicates it is capturing general patterns better than linear regression, and overall will make fewer large mistakes on typical products.

Why Random Forest performs better: The relationship between our features and `reviews_count` is likely non-linear and involves interactions (e.g., certain categories have disproportionately high sales regardless of price or rating). The linear model struggled because, for instance, no single linear weight on price or rating can account for the huge jump in popularity for pet products. The Random Forest, being a non-linear ensemble method, can split the data by category and other features to better isolate those high-review items (as we'll see in feature importance). Thus, for this kind of data, tree-based models or other non-linear models are advantageous.

6. Model Evaluation and Interpretation

Beyond raw accuracy, we interpret the models to understand **which features most strongly influence sales predictions**. For the Random Forest, we can look at the feature importance scores:



Top 10 feature importances from the Random Forest model.

In the feature importance plot above, **category and product attributes play a significant role**:

- **Category = Dog Treats** is by far the most important feature. This confirms that being in the "Dog Treats" category had a huge impact on predicted sales (the model learned that products in this category get an enormously higher number of reviews). This aligns with our EDA: pet treat products (Greenies) were outliers with tens of thousands of reviews.
- **Description length (`desc_length`)** is the second most important feature. The model found that products with longer descriptions tend to have more reviews. This could imply that more popular products have more detailed descriptions (perhaps because they are sold by well-prepared vendors or require more info), or conversely that providing more information might help sell a product. In our data, this may be correlated with certain categories (for example, electronics or bulk items might need lengthy descriptions and also happen to be high-volume).
- **Rating** comes next. A higher product rating likely contributes positively to sales (customers prefer well-rated items). While most products are highly rated, the model still found rating differences matter: e.g., a 4.8-star product might sell more than a 3.5-star product in the same category.
- **Category = Bulk Items, Cat Litter & Supplies, Hair Care, Sheets & Pillowcases, Oral Care** also appear in the top features. These categories had relatively high overall reviews. The model uses these as signals — for instance, if a product is in "Bulk Items" or "Hair Care," it is more likely to have higher sales than an average product, all else equal.
- **Price (`final_price_num`)** and **number of images** have a moderate importance. The negative correlation we saw between price and sales likely is captured here: the model might be splitting products into price ranges, finding that cheaper products yield higher review counts. `num_images`

being important suggests that products with more images (which often are well-marketed or popular items) tend to sell more.
- **Discount** did not show up in the top 10, indicating that whether an item was on sale wasn't a primary factor in the model's predictions. This might be because many top items had no discount (they sold well at full price), or the effect of a discount is smaller compared to category and baseline popularity.

The linear regression model (though less accurate) can also be interpreted via its coefficients. It likewise gave large positive weights to categories like Dog Treats and Bulk Items, and a negative weight to price, consistent with the RF findings. However, linear coefficients were less reliable for inference due to multicollinearity and outliers.

In summary, our modeling indicates: - *Product category is the strongest determinant of sales.* Belonging to certain high-demand categories (pet supplies, personal care, home essentials) greatly increases expected sales. - *Product content and presentation matter:* Longer descriptions and possibly more images correlate with higher sales, implying the importance of rich product information. - *Price and rating have effects in line with intuition:* lower prices (higher affordability) and higher ratings (social proof of quality) tend to boost sales, though their influence is secondary to what category the product is in.

Understanding these drivers allows us to form actionable insights for marketing and merchandising.

7. Marketing Insights and Recommendations

Using the data insights and model findings, we can recommend strategies for **personalized marketing** and **targeted promotions**:

- **Focus on High-Volume Categories:** Direct marketing efforts and inventory investment towards the top-performing categories like **Pet Supplies (Dog Treats, etc.)**, **Beauty/Personal Care (Lotions, Hair Care, Oral Care)**, and **Home Essentials**. For example, a promotional campaign for pet owners featuring the best-selling dog treats or a subscription service for pet snacks could capitalize on evident demand.
- **Personalized Recommendations:** Leverage customer purchase history to recommend products in their favorite categories. If a shopper frequently buys from Beauty, send personalized recommendations for top-rated lotions or hair care products (which are popular and well-reviewed). Similarly, for a customer who buys baby or pet products, promote highly-reviewed items in those categories. Our analysis shows that **category preferences drive buying behavior**, so aligning recommendations with those preferences can increase conversion.
- **Leverage Social Proof:** The fact that products like Cetaphil Lotion have tens of thousands of reviews can be used in marketing messaging ("Trusted by 30,000+ customers!"). Highlight best-sellers and highly-rated products in ads and on the website, as these have proven appeal.
- **Competitive Pricing on Key Items:** Given the negative correlation between price and sales volume, ensure that key traffic-driving items (household staples, pet treats, etc.) are competitively priced. Promotions or everyday low pricing on these items can boost volume and store loyalty. Once customers are drawn in by cheap essentials, upsell higher-margin or complementary products.
- **Improve Product Content:** The importance of `desc_length` and `num_images` suggests that **rich product content correlates with better sales**. Ensure product listings have thorough descriptions, specifications, and multiple high-quality images. This instills confidence and can

improve conversion rates. For new or underperforming products, enhancing the content could make them more appealing.

- **Targeted Promotions & Bundles:** Use insights on frequently purchased categories to create targeted promotions. For example, bundle **Bulk Items** or offer subscription discounts for **high-frequency purchase items** (pet food, toiletries). Since those categories see repeat buys, loyalty programs or bulk-purchase discounts could be effective.
- **Seasonal Marketing:** Plan marketing campaigns around seasonal peaks. While our data snapshot was summer, one should promote relevant categories during their peak season (e.g., outdoor furniture and swimsuits in summer, toys and electronics during holidays, health supplements in January, etc.). Personalized marketing can also use seasonality — e.g., send lawn & garden deals to homeowners in spring, or back-to-school deals in late summer.
- **Address Low-Rated Products:** Although few, any products with consistently low ratings should be reviewed. Customer feedback from the `reviews` can provide insights into quality issues. Consider improving or phasing out poorly rated items, or running promotions to improve their sales if they are important to the assortment.

By implementing these strategies, Target can utilize the analysis to increase sales: **promote what sells, price strategically, personalize the shopping experience, and ensure product quality and presentation.**

8. Visualization

Throughout the analysis, we have integrated visualizations to support findings. We used **Matplotlib** to create bar charts, histograms, and scatter plots that made key insights clear at a glance:

- Time-series/line chart (if we had continuous data) could show sales trends, but in our case we discussed trends qualitatively due to limited timeframe.
- **Bar charts** were used for ranking top products and categories, quickly highlighting where most sales/reviews concentrate (e.g., the dominance of Dog Treats category was immediately visible in the bar chart).
- **Histograms** revealed the distributions of ratings and prices, showing us the skew towards high ratings and low prices.
- **Scatter plot** (with log scales) helped uncover the relationship between price and sales volume, confirming that cheaper products often have higher sales.
- **Feature importance bar chart** allowed us to interpret the model by visualizing which features are driving predictions.

These plots are critical for communicating the analysis: they turn raw numbers into stories (e.g., “Pet products are huge!” or “Almost all products are 4+ stars”). In a report, we ensured each figure has a clear title, labeled axes, and is accompanied by an explanation. Visualization not only **validated our assumptions** (like the price vs sales trend) but also made it easier for stakeholders to grasp the insights, guiding data-driven decision making.

Conclusion: This end-to-end project cleaned and analyzed Target’s product sales data, built predictive models, and extracted actionable business insights. We identified what factors correlate with higher sales (category, price, content, etc.) and how Target can leverage these findings in marketing strategies. By

combining thorough data exploration, feature engineering, modeling, and visual storytelling, we gained a comprehensive understanding of the sales dynamics in the dataset. The approach demonstrated here can be extended with more data (e.g., including actual sales quantities, more time periods, or customer data) to further refine Target's personalized marketing and merchandising decisions.
