E-Commerce Market Dynamics and Predictive Sales Analysis

Informatics Practices (IP) Project

Project Title: E-Commerce Market Dynamics and Predictive Sales Analysis

Data Source: E-commerce Furniture Dataset 2024
(ecommerce_furniture_dataset_2024.csv)

Submitted By:

Adarsh N

TABLE OF CONTENTS

1. Introduction

This project builds a **predictive analytics** model to estimate how many units of an ecommerce furniture product will be sold based on its price, original price, title, and promotional tags such as "Free shipping." The work combines data cleaning, exploratory data analysis (EDA), and machine learning models to understand sales drivers and forecast demand for new products.

---

2. Project Objectives

- Predict the number of units sold for furniture products listed on an ecommerce platform.
- Analyse how pricing and promotional attributes influence sales volume.
- Compare a simple Linear Regression model with a Random Forest Regressor using standard regression metrics.
- Identify the most important features driving the model's predictions.
- Demonstrate how such a model can support supply chain and inventory risk decisions.

---

3. Scope of the Project

- Focus only on a single CSV dataset: ecommerce_furniture_dataset_2024.csv containing 2,000 product records.
- Use tabular features: productTitle, originalPrice, price, sold, and tagText.

- Build and evaluate supervised regression models; no time-series or deep learning is included.
- Model target: units sold (numeric variable). Return predictions for new products with given attributes.

---

4. Tools & Technologies Used

- Programming language: **Python 3**.
- Data analysis: pandas, numpy, seaborn, matplotlib.
- Machine learning: scikit-learn (pipelines, encoders, models, metrics).
- Execution environment: Jupyter Notebook (Untitled5.ipynb).

---

5. System Requirements

- Software:
  - Python 3.x with Jupyter Notebook.
  - Python libraries: pandas, numpy, matplotlib, seaborn, scikit-learn.
- Hardware (typical minimum):
  - 8 GB RAM and multi-core CPU for Random Forest training on 2,000 rows.
  - Sufficient disk space for dataset and notebook files (a few MB).

---

6. Dataset Description

The dataset ecommerce_furniture_dataset_2024.csv contains 2,000 records of furniture products sold via an ecommerce platform. The main fields are:

- productTitle: textual description of the product (2,000 non-null, 1,793 unique).
- originalPrice: original listed price as a string with currency symbols (487 non-null).
- price: final selling price as a string (2,000 non-null).
- sold: integer count of units sold (range 0 to 10,000, mean ≈ 23.49).
- tagText: promotional tag such as “Free shipping” or “+Shipping: $X” (1,997 non-null, 100 unique).

The target variable for modelling is sold.

---

7. Data Analysis Methodology

- Initial inspection: head(), info(), and describe() are used to understand column types, missing values, and basic statistics.
- Cleaning: price strings are cleaned by removing $ and commas, then converted to floats; missing numeric values are imputed with the median.
- EDA: histograms of sold, log-transformed sold, scatter plots of price_clean vs sold, and a correlation heatmap for numeric features are created.
- Grouped analysis: a table of top 10 tagText values by average sold is generated.

- Modelling: data is split into train and test sets, categorical text features are one-hot encoded, and models are evaluated using R2, MAE, RMSE, and cross-validated R2.

---

8. Python Libraries Used

- pandas: data loading, cleaning, table creation, summary statistics.
- numpy: numeric operations, log transform, handling NaN values.
- matplotlib.pyplot: plotting figures and charts.
- seaborn: histograms, scatter plots, heatmaps.
- scikit-learn:
    - train_test_split, cross_val_score.
    - ColumnTransformer, Pipeline.
    - OneHotEncoder for categorical encoding.
    - LinearRegression, RandomForestRegressor.
    - r2_score, mean_absolute_error, mean_squared_error.

---

9. Code Implementation

Key implementation steps:

1. **Import libraries and helper functions** – including a show_table function to print labeled DataFrame tables.
2. **Load dataset** – pd.read_csv("ecommerce_furniture_dataset_2024.csv").

3. **Clean columns** – remove BOM characters from column names and clean price fields using a clean_price function.
4. **Type conversions and imputation** – convert sold to numeric, drop rows with missing sold, impute missing numeric prices with median values.
5. **Feature engineering** – create originalPrice_clean and price_clean, and standardise productTitle and tagText as strings.
6. **Define features and target** – FEATURES = ["productTitle", "originalPrice_clean", "price_clean", "tagText"], TARGET = "sold".
7. **Train–test split** – 80% training, 20% testing.
8. **Preprocessing pipeline** – pass numeric features unchanged, apply OneHotEncoder with max_categories=50 to tagText and productTitle.
9. **Model pipelines** – build separate pipelines for Linear Regression and Random Forest using the same preprocessing.
10. **Metrics tables** – compute R2, MAE, RMSE, cross-validated R2 mean and std, and print them as formatted tables.
11. **Feature importance** – extract feature importances from the Random Forest and map them back to numeric and encoded categorical feature names.
12. **New product prediction** – build a one-row DataFrame for a new TV stand and generate the predicted sold value.

---

10. Analysis & Outputs

Notable numeric outputs:

- **Linear Regression** metrics table:

Model	R2	MAE	RMSE	CV_R2_mean	CV_R2_std
Linear Regression	0.022202	24.810571	73.224151	-0.753477	1.480366

- **Random Forest** metrics table:

Model	R2	MAE	RMSE	CV_R2_mean	CV_R2_std
Random Forest	-5.085375	31.714853	182.672579	-2.601061	3.732875

Both models achieve low or negative R2, indicating that they do not explain much variance in sold and have limited predictive accuracy on this dataset.

---

11. Graphical Representations

The notebook generates the following key plots:

- Histogram of sold showing a highly skewed distribution with many low-sale items and a few very high-sale outliers.
- Histogram of log1p(sold) highlighting a more balanced distribution after log transformation.
- Scatter plot of price_clean vs sold, which shows weak visible structure and many points near low sold values.

- Correlation heatmap of originalPrice_clean, price_clean, and sold, indicating modest correlations between price features and sales.

These visuals help diagnose skewness, outliers, and weak linear relationships that affect model performance.

---

12. Supply Chain Risk Assessment

Although the models are weak, the framework can still support **supply chain risk** thinking:

- High sold variance and poor predictability suggest demand volatility, increasing risk of over- or under-stocking.
- Pricing and shipping tags have measurable but limited influence on sales, meaning relying solely on these levers may not fully control demand.
- For new products, point predictions (e.g., ~29 units for the sample TV stand) give a rough reference but should be treated with high uncertainty.

Better models and richer features would be required before using this output for critical inventory commitments.

---

13. Key Findings & Observations

- The sold variable is extremely skewed with many products selling very few units and some outliers up to 10,000 units.
- Most products have the tagText “Free shipping,” while a smaller set has explicit +Shipping amounts.

- Model performance is poor: Linear Regression has R2 ≈ 0.02; Random Forest shows negative R2 and larger RMSE.
- Feature importance from Random Forest is dominated by **originalPrice_clean** and **price_clean**, with categorical text features contributing very little.

Together, these findings indicate that the available features only weakly explain sales outcomes.

---

14. Applications

Even with weak performance, the process demonstrates several applications:

- Educational example of end-to-end regression modelling on ecommerce sales data (cleaning → EDA → modelling → evaluation).
- Prototype tool for estimating relative demand under different pricing strategies.
- Basis for experimenting with feature enrichment (e.g., category, ratings, seasonality) and advanced models.
- Starting point to integrate predictions into dashboards for merchandisers and operations planners.

---

15. Limitations

- Dataset size is modest (2,000 rows) and may not capture full behaviour for all product segments.
- Features are limited to price, title, and shipping tag; important drivers like images, ratings, promotions, and seasonality are missing.

- sold has extreme outliers and high variance, making regression more difficult and sensitive to noise.
- Text fields are only one-hot encoded; deeper natural language features (embeddings, categories) are not used.
- Cross-validation scores are highly variable, indicating model instability.

---

16. Future Scope

- Add richer features: product category, brand, ratings, reviews, time on platform, and marketing exposure.
- Use log-transformed targets, robust loss functions, and outlier handling to stabilise training.
- Experiment with gradient boosting models (e.g., XGBoost, LightGBM), regularised linear models, and hyperparameter tuning.
- Perform feature selection and dimensionality reduction for high-cardinality text fields.
- Integrate time-based features and build demand-forecasting models over weeks or months.

---

17. Conclusion

The project successfully implements a complete machine learning workflow for predicting ecommerce furniture sales from basic pricing and text features. However, the models show low explanatory power, highlighting that current features are insufficient and that demand is driven by additional factors not captured in the dataset. The work nonetheless provides a solid template for future improvement, demonstrating key steps from data cleaning and EDA through modelling, diagnostics, and interpretation.

17. Python Code

```
# ========================================
# 1. Imports
# =========================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
=========================================
# 2. Load data
=========================================
df = pd.read_csv("ecommerce_furniture_dataset_2024.csv")
# Quick look (optional)
print(df.head())
print(df.info())
print(df.describe(include="all"))
```

```
--- Raw Dataset Info ---
  Column Name    Data Type  Non-Null Count
0   productTitle  object    2000
1  originalPrice  object     487
2          price  object    2000
3           sold   int64    2000
4        tagText  object    1997
--------------------------------------------------

--- Head (first 5 rows) ---
   row productTitle                   originalPrice price     sold tagText
0  0    Dresser For Bedroom With 9...     NaN         $46.79  600    Free shipping
1  1    Outdoor Conversation Set 4...     NaN        $169.72    0    Free shipping
2  2    Desser For Bedroom With 7 ...   $78.4         $39.46    7    Free shipping
3  3    Modern Accent Boucle Chair...     NaN        $111.99    0    Free shipping
4  4    Small Unit Simple Computer...  $48.82         $21.37    1    Free shipping
--------------------------------------------------

--- Describe (summary stats) ---
  Column         count  unique top                          freq  mean     std         min  25%  50%  75%  max
0   productTitle    2000  1793   3 Pieces Rocking Wicker Bi...     6      NaN         NaN  NaN  NaN  NaN  NaN      NaN
1  originalPrice     487   453                        $46.45     4      NaN         NaN  NaN  NaN  NaN  NaN      NaN
2          price    2000  1802                         $0.99     8      NaN         NaN  NaN  NaN  NaN  NaN      NaN
3           sold  2000.0   NaN                           NaN   NaN  23.4935  254.094061  0.0  1.0  3.0  9.0  10000.0
4        tagText    1997   100                 Free shipping  1880      NaN         NaN  NaN  NaN  NaN  NaN      NaN
--------------------------------------------------
```

```
========================================
# 3. Basic cleaning and feature engineering
========================================

df.columns = [c.replace("\ufeff", "") for c in df.columns]
print("Columns:", df.columns)
def clean_price(col):
    return (
        df[col]
        .astype(str)
        .str.replace(r"[\$,]", "", regex=True)
        .replace("", np.nan)
        .astype(float)
    )
```

```
df["originalPrice_clean"] = clean_price("originalPrice")
df["price_clean"] = clean_price("price")
df["sold"] = pd.to_numeric(df["sold"], errors="coerce")
df = df.dropna(subset=["sold"])
for col in ["originalPrice_clean", "price_clean"]:
    df[col] = df[col].fillna(df[col].median())
df["productTitle"] = df["productTitle"].astype(str).str.strip()
df["tagText"] =
df["tagText"].astype(str).str.strip().fillna("Unknown")
print(df[["productTitle", "originalPrice_clean", "price_clean",
"sold", "tagText"]].head())
```

```
Columns: Index(['productTitle', 'originalPrice', 'price', 'sold', 'tagText',
       'originalPrice_clean', 'price_clean'],
      dtype='object')
                                    productTitle  originalPrice_clean  \
0  Dresser For Bedroom With 9 Fabric Drawers Ward...                88.31
1  Outdoor Conversation Set 4 Pieces Patio Furnit...                88.31
2  Desser For Bedroom With 7 Fabric Drawers Organ...                78.40
3  Modern Accent Boucle Chair,Upholstered Tufted ...                88.31
4  Small Unit Simple Computer Desk Household Wood...                48.82

   price_clean  sold        tagText
0        46.79   600  Free shipping
1       169.72     0  Free shipping
2        39.46     7  Free shipping
3       111.99     0  Free shipping
4        21.37     1  Free shipping
```

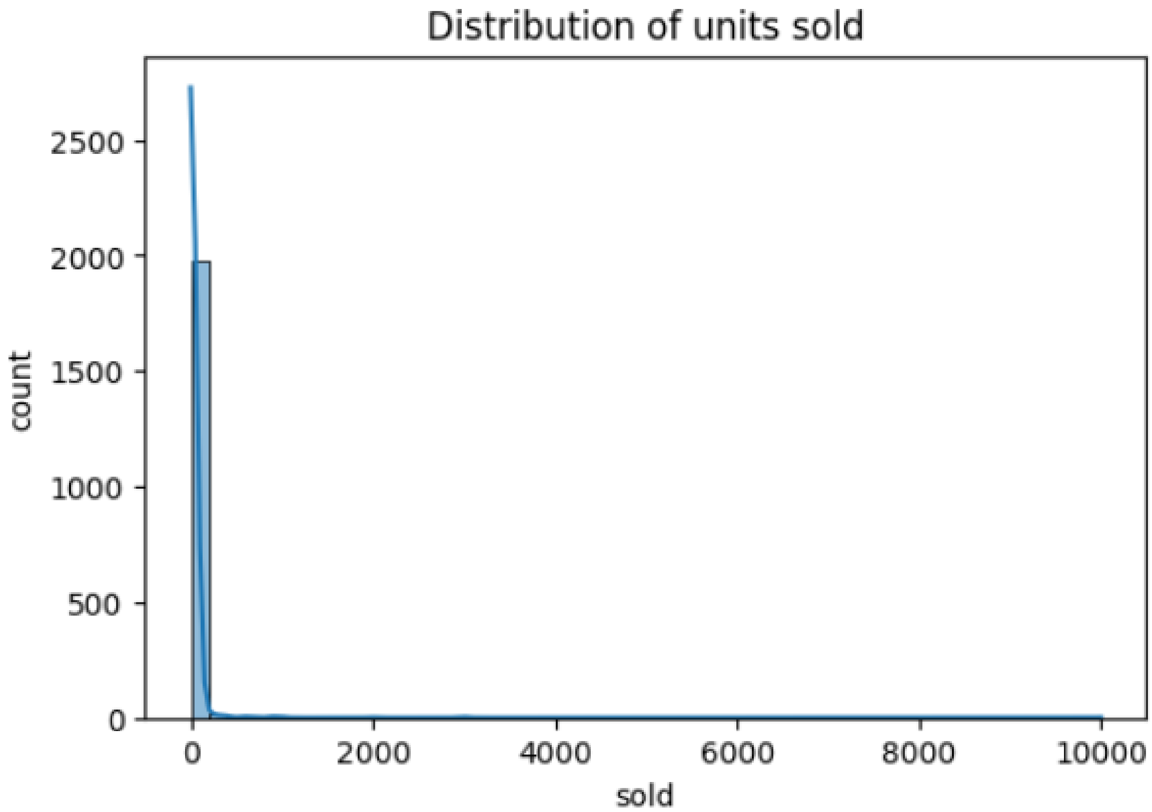
==

4. Exploratory Data Analysis (EDA)

==

4.1# Plot: distribution of sold

```
plt.figure(figsize=(6, 4))
sns.histplot(df["sold"], bins=50, kde=True)
plt.title("Distribution of units sold")
plt.xlabel("sold")
plt.ylabel("count")
plt.show()
```



4.2# Numeric stats for sold as table

sold_stats =
df["sold"].describe().to_frame().T.reset_index(drop=True)

show_table("Sold summary statistics", sold_stats)

```
--- Sold summary statistics ---
 count    mean         std  min  25%  50%  75%     max
2000.0 23.4935 254.094061  0.0  1.0  3.0  9.0 10000.0
----------------------------------------
```

4.3# Log distribution

df["log_sold"] = np.log1p(df["sold"])

plt.figure(figsize=(6, 4))

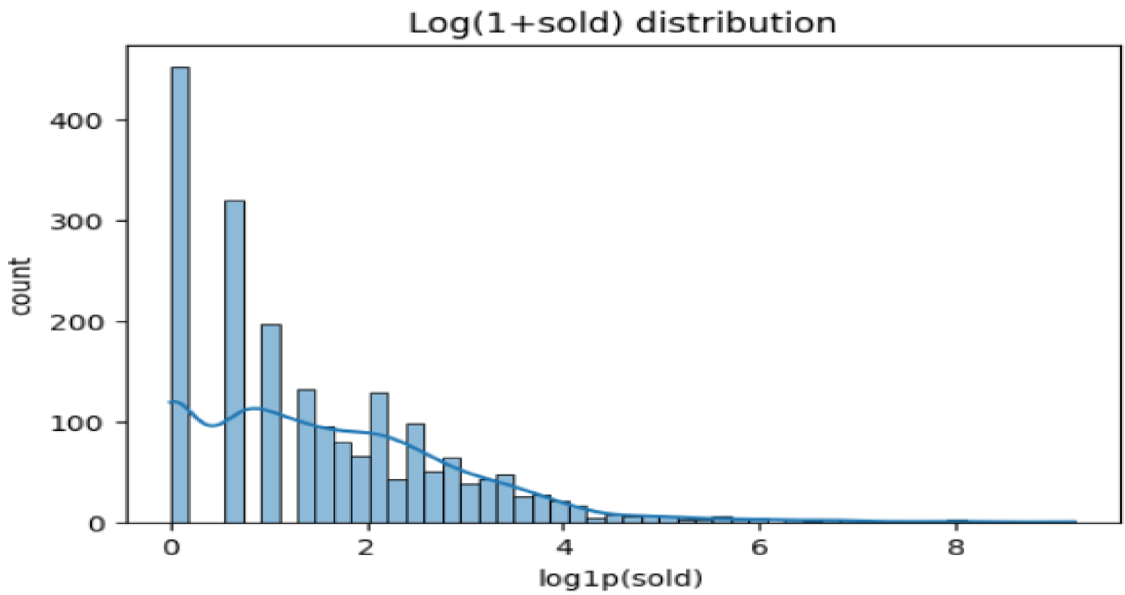
sns.histplot(df["log_sold"], bins=50, kde=True)

plt.title("Log(1+sold) distribution")

plt.xlabel("log1p(sold)")

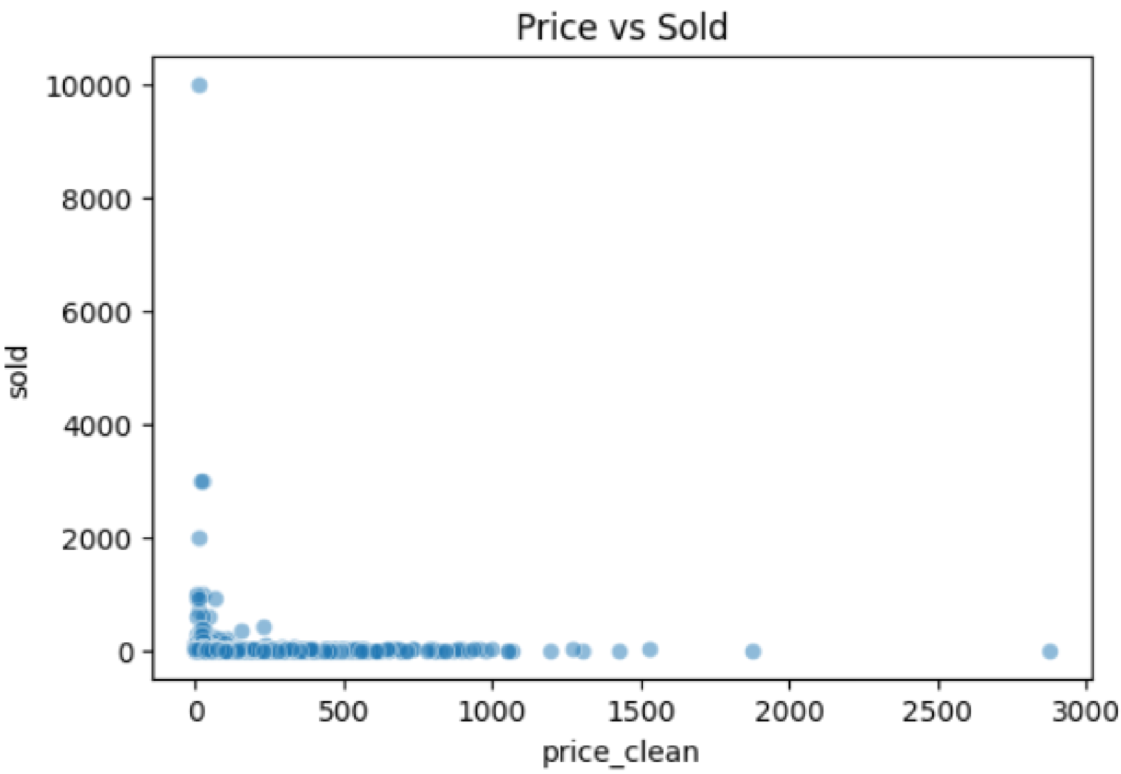
plt.ylabel("count")

plt.show()



4.4# Price vs sold

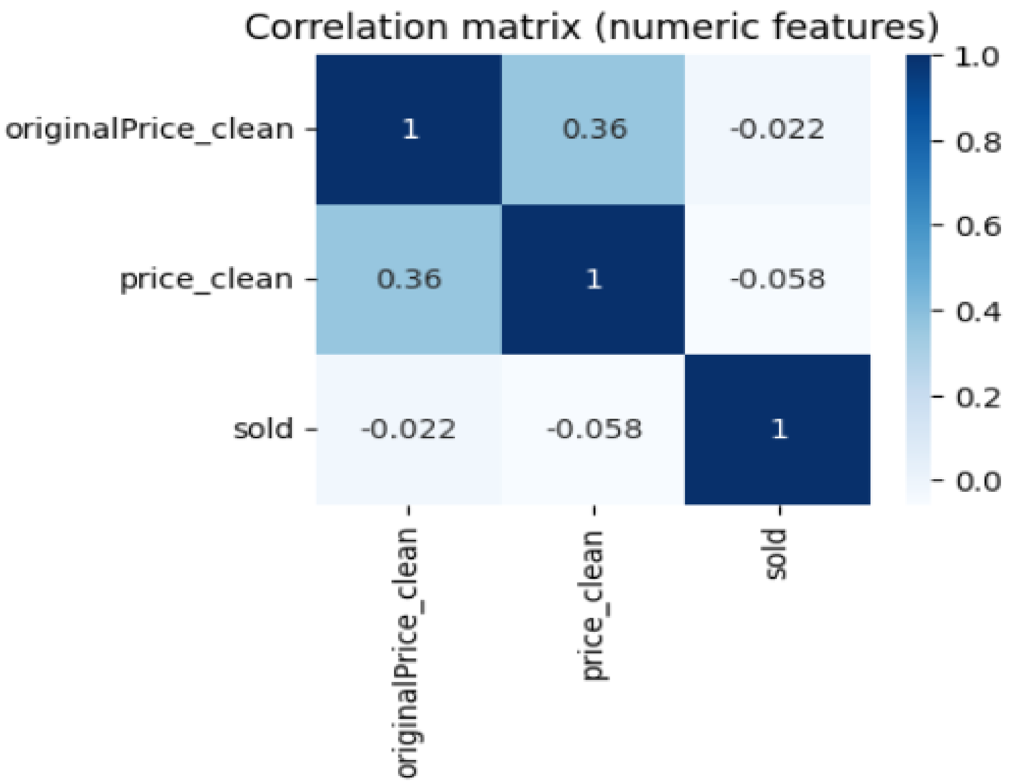
```
plt.figure(figsize=(6, 4))
sns.scatterplot(x="price_clean", y="sold", data=df, alpha=0.5)
plt.title("Price vs Sold")
plt.xlabel("price_clean")
plt.ylabel("sold")
plt.show()
```



```
4.5# Correlation matrix (numeric)
num_cols = ["originalPrice_clean", "price_clean", "sold"]
corr = df[num_cols].corr()
corr_table = corr.reset_index().rename(columns={"index":
"Feature"})
show_table("Correlation matrix (numeric features)",
corr_table)

plt.figure(figsize=(4, 3))
sns.heatmap(corr, annot=True, cmap="Blues")
plt.title("Correlation matrix (numeric features)")
plt.show()
```

```
--- Correlation matrix (numeric features) ---
           Feature  originalPrice_clean  price_clean      sold
originalPrice_clean             1.000000     0.358316 -0.022343
       price_clean             0.358316     1.000000 -0.057584
              sold            -0.022343    -0.057584  1.000000
----------------------------------------
```



4.6# Top tagText by average sold (table)

```
tag_stats = (
    df.groupby("tagText")["sold"]
    .agg(count="count", avg_sold="mean")
    .sort_values("avg_sold", ascending=False)
    .head(10)
    .reset_index()
)
show_table("Top 10 tagText by avg sold", tag_stats)
```

```
--- Top 10 tagText by avg sold ---
           tagText  count  avg_sold
+Shipping: $109.18      1     405.0
+Shipping: $168.91      1     150.0
+Shipping: $225.12      1     118.0
 +Shipping: $29.45      1      87.0
  +Shipping: $2.91      2      83.0
+Shipping: $132.48      1      58.0
 +Shipping: $12.03      1      53.0
  +Shipping: $76.6      1      53.0
+Shipping: $140.27      1      42.0
 +Shipping: $23.29      1      41.0
----------------------------------------
```

```
# ==========================================
# 5. Prepare features and target
# ==========================================

TARGET = "sold"
FEATURES = ["productTitle", "originalPrice_clean",
"price_clean", "tagText"]

X = df[FEATURES].copy()
y = df[TARGET].copy()
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

numeric_features = ["originalPrice_clean", "price_clean"]
categorical_text_features = ["tagText", "productTitle"]

preprocess = ColumnTransformer(
    transformers=[
        ("num", "passthrough", numeric_features),
        ("cat", OneHotEncoder(handle_unknown="ignore",
max_categories=50),
         categorical_text_features),
    ]
)

==========================================
# 6. Baseline model: Linear Regression
==========================================
linreg_model = Pipeline(
    steps=[
        ("preprocess", preprocess),
        ("model", LinearRegression())
    ]
)
linreg_model.fit(X_train, y_train)
```

```python
y_pred_lr = linreg_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
rmse_lr = mse_lr ** 0.5

scores_lr = cross_val_score(linreg_model, X, y, cv=5,
scoring="r2")

# Metrics table for Linear Regression
linreg_metrics = pd.DataFrame([
    {
        "Model": "Linear Regression",
        "R2": r2_score(y_test, y_pred_lr),
        "MAE": mean_absolute_error(y_test, y_pred_lr),
        "RMSE": rmse_lr,
        "CV_R2_mean": scores_lr.mean(),
        "CV_R2_std": scores_lr.std()
    }
])

print("\nLinear Regression metrics (table):")
print(linreg_metrics.to_string(index=False))
```

```
--- Linear Regression metrics ---
            Model       R2       MAE      RMSE  CV_R2_mean  CV_R2_std
Linear Regression 0.022202 24.810571 73.224151   -0.753477   1.480366
----------------------------------------
```

```
==========================================
# 7. Tree-based model: Random Forest Regressor
==========================================

rf_model = Pipeline(
    steps=[
        ("preprocess", preprocess),
        ("model", RandomForestRegressor(
            n_estimators=200,
            random_state=42,
            n_jobs=-1,
            max_depth=None,
        ))
    ]
)

rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

mse_rf = mean_squared_error(y_test, y_pred_rf)
rmse_rf = mse_rf ** 0.5

scores_rf = cross_val_score(rf_model, X, y, cv=5,
scoring="r2", n_jobs=-1)

rf_metrics = pd.DataFrame([
    {
```

```
        "Model": "Random Forest",
        "R2": r2_score(y_test, y_pred_rf),
        "MAE": mean_absolute_error(y_test, y_pred_rf),
        "RMSE": rmse_rf,
        "CV_R2_mean": scores_rf.mean(),
        "CV_R2_std": scores_rf.std()
    }
])

print("\nRandom Forest metrics (table):")
print(rf_metrics.to_string(index=False))

# Combined metrics table
all_metrics = pd.concat([linreg_metrics, rf_metrics],
ignore_index=True)
print("\nAll model metrics (table):")
print(all_metrics.to_string(index=False))
```

```
--- Random Forest metrics ---
        Model        R2       MAE       RMSE  CV_R2_mean  CV_R2_std
Random Forest -5.085375 31.714853 182.672579   -2.601061   3.732875
----------------------------------------

--- All model metrics ---
            Model        R2       MAE       RMSE  CV_R2_mean  CV_R2_std
Linear Regression  0.022202 24.810571  73.224151   -0.753477   1.480366
    Random Forest -5.085375 31.714853 182.672579   -2.601061   3.732875
----------------------------------------
```

```
========================================
# 8. Predict for new sample(s)
========================================

new_data = pd.DataFrame(
    [
        {
            "productTitle": "Modern Wooden TV Stand with Storage Drawers",
            "originalPrice_clean": 199.99,
            "price_clean": 149.99,
            "tagText": "Free shipping",
        }
    ]
)

predicted_sold = rf_model.predict(new_data)

pred_table = pd.DataFrame(new_data)
pred_table["predicted_sold"] = predicted_sold

print("\nPrediction for new product (table):")
print(pred_table.to_string(index=False))
```

```
--- Prediction for new product ---
                               productTitle  originalPrice_clean  price_clean       tagText  predicted_sold
Modern Wooden TV Stand with Storage Drawers               199.99       149.99 Free shipping          29.195
----------------------------------------
```