UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

Project Title	**E-commerce Furniture Dataset 2024**
Tools	Python, ML, SQL, Excel
Domain	Data Analyst
Project Difficulties level	Beginner

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Dataset Overview:

This dataset comprises 2,000 entries scraped from AliExpress, detailing a variety of furniture products. It captures key sales metrics and product details, offering a snapshot of consumer purchasing patterns and market trends in the online furniture retail space.

Data Science Applications:

The dataset is ripe for exploratory data analysis, market trend analysis, and price optimization studies. It can also be used for predictive modeling to forecast sales,

understand the impact of discounts on sales volume, and analyze the relationship between product features and their popularity.

Column Descriptors:

- `productTitle`: The name of the furniture item.
- `originalPrice`: The original price of the item before any discounts.
- `price`: The current selling price of the item.
- `sold`: The number of units sold.
- `tagText`: Additional tags associated with the item (e.g., "Free shipping").

Ethically Collected Data:
The data was collected in compliance with ethical standards, ensuring respect for user privacy and platform terms of service.

Acknowledgements:
This dataset was created with data sourced from AliExpress, using Apify for scraping. The thumbnail image was generously provided by Spacejoy on Unsplash. We extend our gratitude to these parties for their contributions to this dataset.

Photo by Spacejoy on Unsplash.

Example: You can get the basic idea how you can create a project from here

Project Overview:

- **Objective**: Predict the number of furniture items sold (`sold`) based on product attributes such as `productTitle`, `originalPrice`, `price`, and `tagText`.
- **Tech Stack**: Python, pandas, scikit-learn, matplotlib, seaborn

Steps:

1. **Data Collection**
2. **Data Preprocessing**
3. **Exploratory Data Analysis (EDA)**
4. **Feature Engineering**
5. **Model Selection & Training**
6. **Model Evaluation**
7. **Conclusion**

---

1. Data Collection

In this step, we assume that the dataset is available in CSV format. We can load it using pandas.

```
# Import necessary libraries
import pandas as pd

# Load dataset
df = pd.read_csv('ecommerce_furniture_dataset.csv')
```

```
# View the first few rows of the dataset
print(df.head())
```

2. Data Preprocessing

We will clean the data by handling missing values, converting categorical variables, and removing irrelevant columns.

```
# Check for missing values
print(df.isnull().sum())

# Dropping any rows with missing values (if applicable)
df = df.dropna()

# Converting tagText into a categorical feature (if necessary)
df['tagText'] = df['tagText'].astype('category').cat.codes

# Checking for data types and conversions if necessary
print(df.info())
```

3. Exploratory Data Analysis (EDA)

Visualize the relationships between features and the target variable (`sold`).
Understand the distribution and trends in the data.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of 'sold' values
sns.histplot(df['sold'], kde=True)
plt.title('Distribution of Furniture Items Sold')
plt.show()

# Plot the relationship between originalPrice, price and sold
sns.pairplot(df, vars=['originalPrice', 'price', 'sold'],
kind='scatter')
plt.title('Relationship Between Price, Original Price, and
Items Sold')
plt.show()
```

4. Feature Engineering

1. **Handling Product Titles**: We will convert `productTitle` to numerical features using techniques like **TF-IDF**.
2. **Price and Discount Feature**: Create a new feature to calculate the percentage discount from `originalPrice` and `price`.

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Create a new feature: percentage discount
df['discount_percentage'] = ((df['originalPrice'] -
df['price']) / df['originalPrice']) * 100

# Convert productTitle into a numeric feature using TF-IDF
Vectorizer
tfidf = TfidfVectorizer(max_features=100)
productTitle_tfidf = tfidf.fit_transform(df['productTitle'])

# Convert to DataFrame and concatenate to original df
productTitle_tfidf_df =
pd.DataFrame(productTitle_tfidf.toarray(),
columns=tfidf.get_feature_names_out())
df = pd.concat([df, productTitle_tfidf_df], axis=1)

# Drop original productTitle as it's now encoded
df = df.drop('productTitle', axis=1)
```

5. Model Selection & Training

We will use **Linear Regression** and **Random Forest Regressor** as models to predict the number of items sold.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Split the dataset into features (X) and target (y)
X = df.drop('sold', axis=1)
y = df['sold']

# Train-test split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize models
lr_model = LinearRegression()
rf_model = RandomForestRegressor(n_estimators=100,
random_state=42)

# Train models
lr_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
```

6. Model Evaluation

We evaluate the model's performance using **mean squared error (MSE)** and **R-squared** metrics.

```
# Predict with Linear Regression
y_pred_lr = lr_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# Predict with Random Forest
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# Print model evaluation results
print(f'Linear Regression MSE: {mse_lr}, R2: {r2_lr}')
print(f'Random Forest MSE: {mse_rf}, R2: {r2_rf}')
```

7. Conclusion

After evaluating the models, we can conclude which model performed better and further tune hyperparameters if needed. Random Forest tends to perform better on complex datasets with high variance, while Linear Regression might work well if relationships are linear.

Output:

1. **Linear Regression Model**: MSE and R-squared score.
2. **Random Forest Model**: MSE and R-squared score.

NOTE :

1. this project is only for your guidance, not exactly the same you have to create. Here I am trying to show the way or idea of what steps you can follow and how your projects look. Some projects are very advanced (because it will be made with the help of flask, nlp, advance ai, advance DL and some advanced things) which you can not understand .

2. You can make or analyze your project with yourself, with your idea, make it more creative from where we can get some information and understand about our business. make sure what overall things you have created all things you understand very well.

Example: You can get the basic idea how you can create a project from here

Sample code with output

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create
```

```
a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they
won't be saved outside of the current session
```

```
/kaggle/input/e-commerce-furniture-dataset-2024/ecommerce_furni
ture_dataset_2024.csv
```

In [2]:

```
df =
pd.read_csv('/kaggle/input/e-commerce-furniture-dataset-2024/ec
ommerce_furniture_dataset_2024.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	productTitle	original Price	price	sold	tagText

0	Dresser For Bedroom With 9 Fabric Drawers Ward...	NaN	$46.79	600	Free shipping
1	Outdoor Conversation Set 4 Pieces Patio Furnit...	NaN	$169.72	0	Free shipping
2	Desser For Bedroom With 7 Fabric Drawers Organ...	$78.4	$39.46	7	Free shipping
3	Modern Accent Boucle Chair,Upholstered Tufted ...	NaN	$111.99	0	Free shipping
4	Small Unit Simple Computer Desk Household Wood...	$48.82	$21.37	1	Free shipping

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
productTitle        0
```

```
originalPrice    1513
price               0
sold                0
tagText             3

dtype: int64
```

In [5]:

```
df.shape
```

Out[5]:

```
(2000, 5)
```

In [6]:

```
df.drop(['originalPrice'],axis=1,inplace=True)
```

In [7]:

```
df.head()
```

Out[7]:

	productTitle	price	so	tagText

			ld	
0	Dresser For Bedroom With 9 Fabric Drawers Ward...	$46.79	600	Free shipping
1	Outdoor Conversation Set 4 Pieces Patio Furnit...	$169.72	0	Free shipping
2	Desser For Bedroom With 7 Fabric Drawers Organ...	$39.46	7	Free shipping
3	Modern Accent Boucle Chair,Upholstered Tufted ...	$111.99	0	Free shipping
4	Small Unit Simple Computer Desk Household Wood...	$21.37	1	Free shipping

In [8]:

```
df['tagText'].nunique()
```

Out[8]:

```
100

In [9]:
df['tagText'].value_counts()
```

```
Out[9]:
tagText
Free shipping           1880
+Shipping: $5.09           9
+Shipping: $239.64         2
+Shipping: $97.54          2
+Shipping: $64.56          2
                         ...
+Shipping: $88.26          1
+Shipping: $170.31         1
+Shipping: $1,097.18       1
+Shipping: $106.13         1
+Shipping: $171.49         1

Name: count, Length: 100, dtype: int64

In [10]:
# Replace all values except 'Free shipping' and '+Shipping:
$5.09' with 'others'
df['tagText'] = df['tagText'].apply(lambda x: x if x in ['Free
```

```
shipping', '+Shipping: $5.09'] else 'others')


# Display the modified value counts
print(df['tagText'].value_counts())
```

```
tagText
Free shipping       1880
others               111
+Shipping: $5.09       9
Name: count, dtype: int64
```

In [11]:

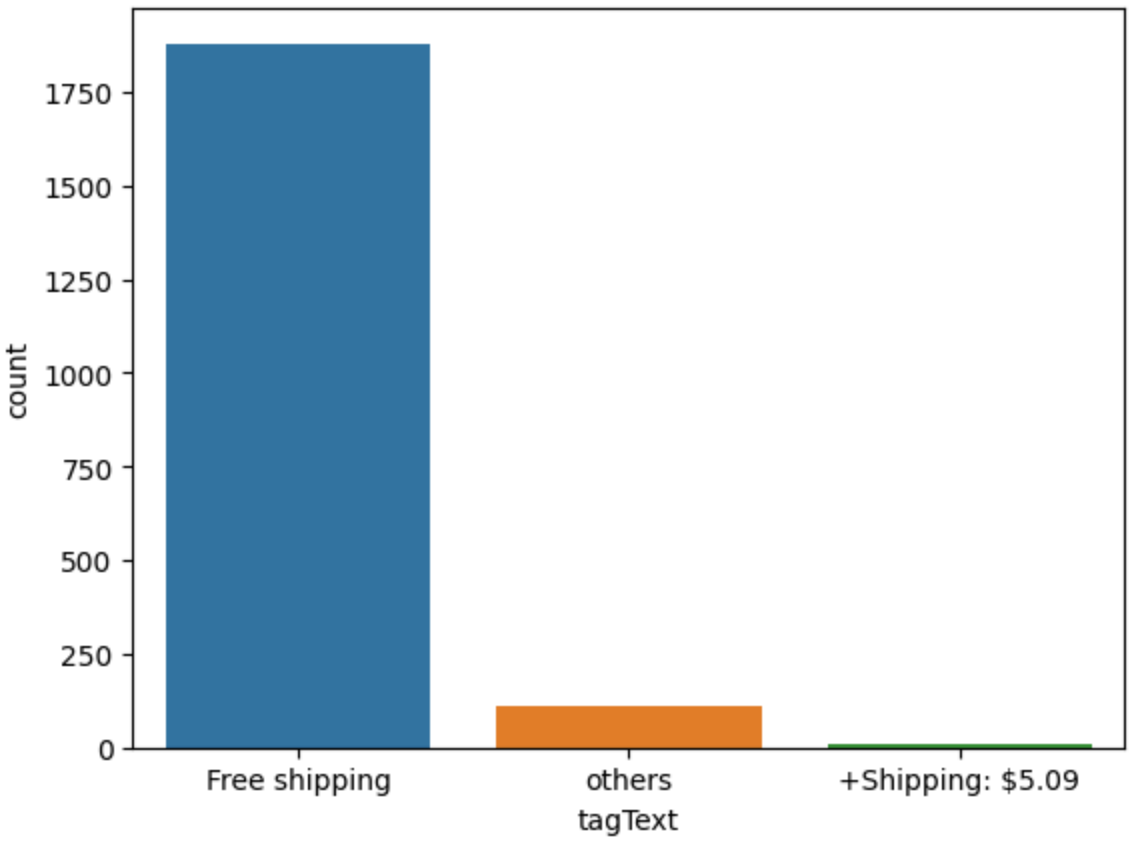
```
import seaborn as sns
```

In [12]:

```
sns.countplot(x='tagText',data=df)
```

Out[12]:

```
<Axes: xlabel='tagText', ylabel='count'>
```



In [13]:

```
df['price'] = df['price'].replace('[\$,]', '',
regex=True).astype(float)
```

In [14]:

```
df.head()
```

Out[14]:

	productTitle	price	sold	tagText
0	Dresser For Bedroom With 9 Fabric Drawers Ward...	46.79	600	Free shipping
1	Outdoor Conversation Set 4 Pieces Patio Furnit...	169.72	0	Free shipping
2	Desser For Bedroom With 7 Fabric Drawers Organ...	39.46	7	Free shipping
3	Modern Accent Boucle Chair,Upholstered Tufted ...	111.99	0	Free shipping
4	Small Unit Simple Computer Desk Household Wood...	21.37	1	Free shipping

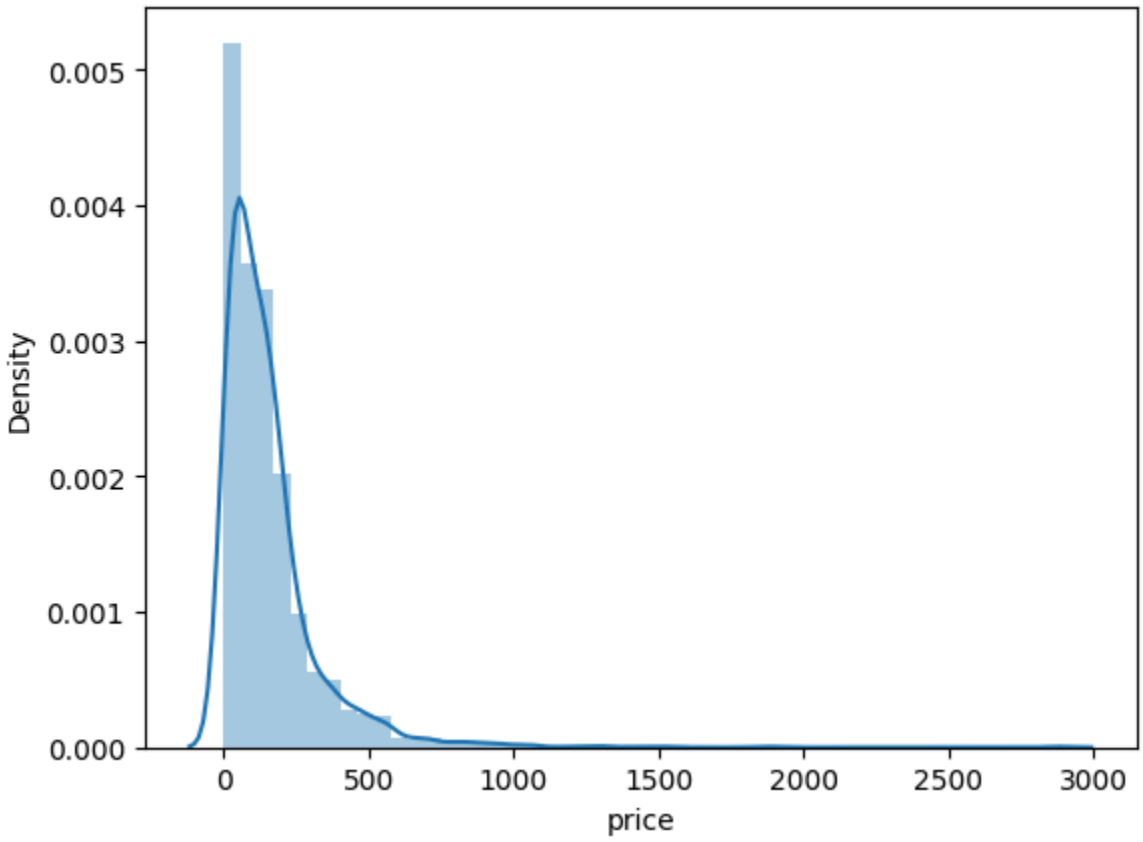
In [15]:

```
sns.distplot(df['price'])
```

```
/tmp/ipykernel_18/444587821.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in
seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions,
please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe575
1


  sns.distplot(df['price'])
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[15]:

<Axes: xlabel='price', ylabel='Density'>



In [16]:

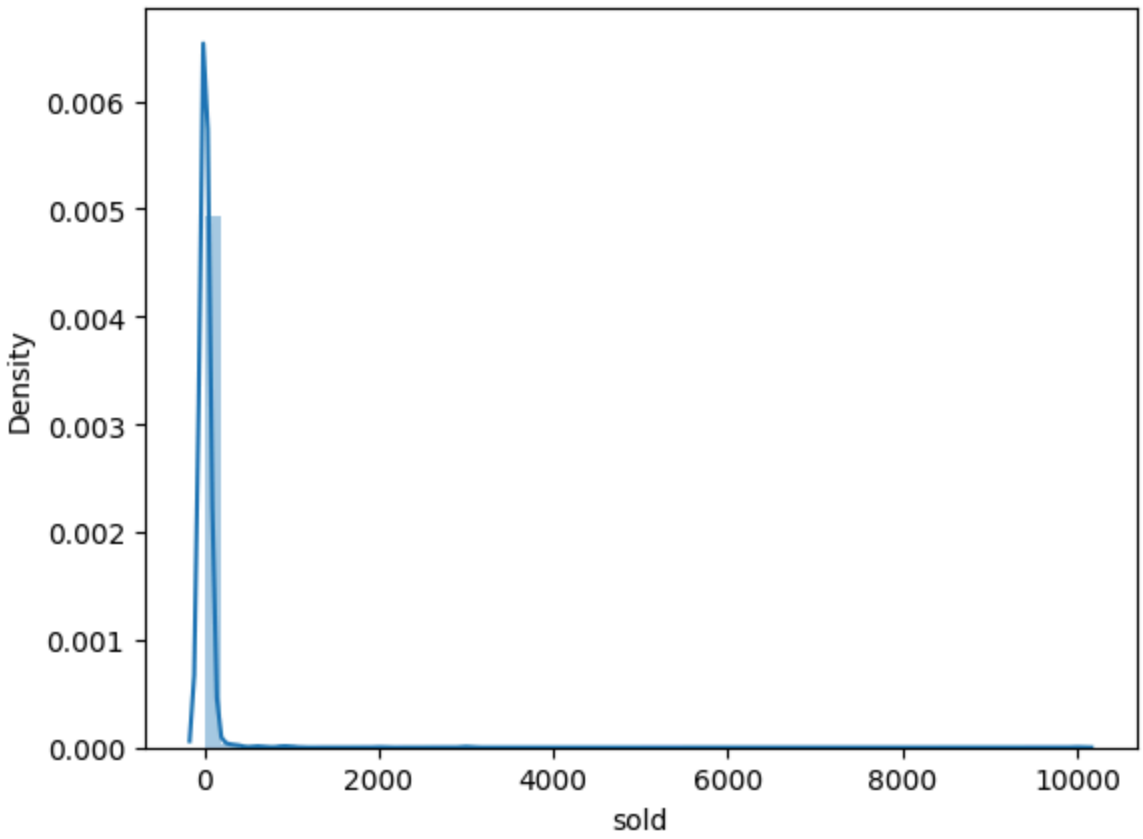
```
sns.distplot(df['sold'])
```

```
/tmp/ipykernel_18/2507294489.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in
seaborn v0.14.0.
```

```
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751


  sns.distplot(df['sold'])
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[16]:

```
<Axes: xlabel='sold', ylabel='Density'>
```

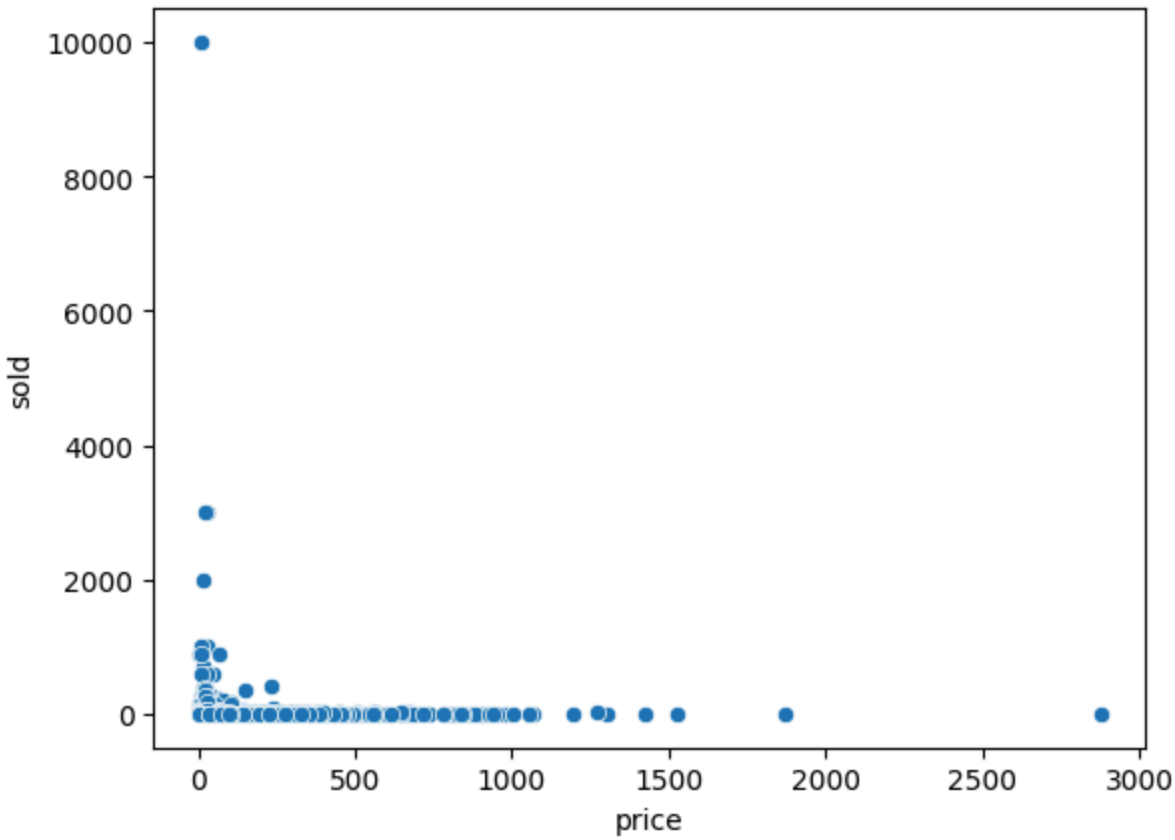


In [17]:

```
sns.scatterplot(x='price', y='sold', data=df)
```

Out[17]:

```
<Axes: xlabel='price', ylabel='sold'>
```



In [18]:

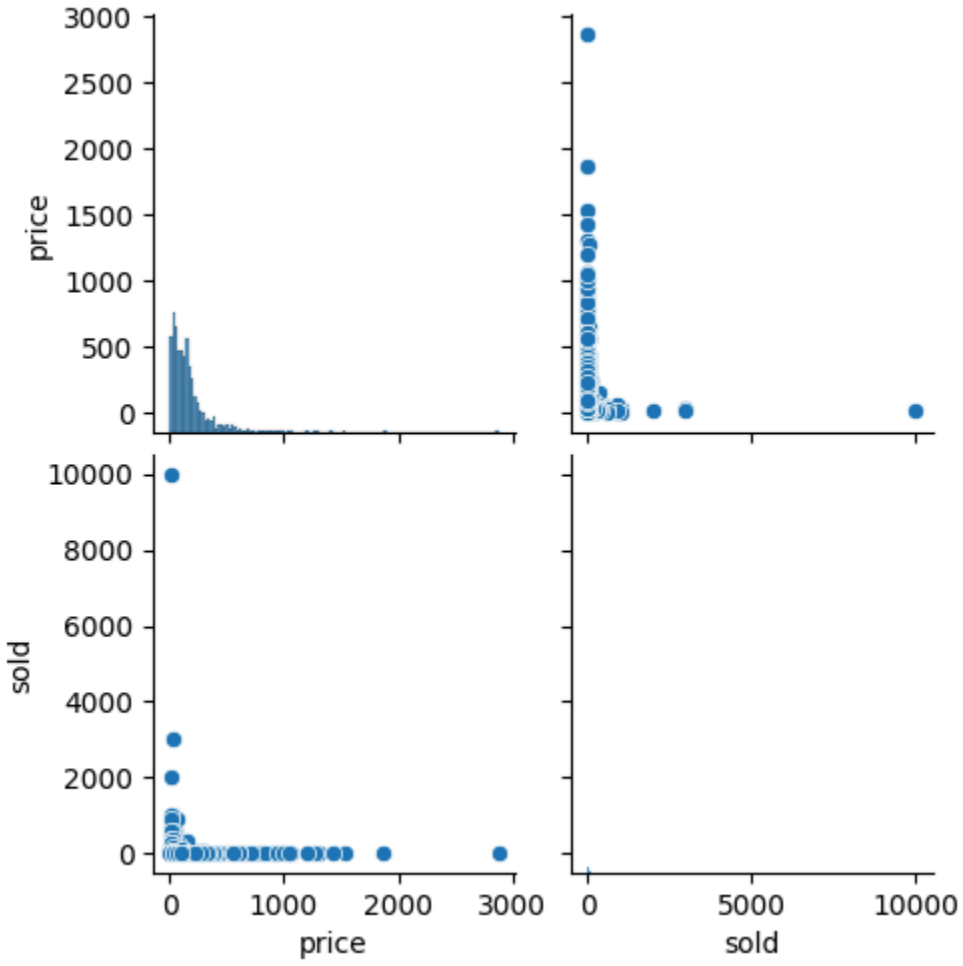
```
filtered_df = df[df['tagText'] == 'Free shipping']

# Create a pairplot including the 'sold' column and other
relevant columns
sns.pairplot(filtered_df[['price', 'sold']])
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
before operating instead.
```

```
  with pd.option_context('mode.use_inf_as_na', True):
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:111
9: FutureWarning: use_inf_as_na option is deprecated and will
be removed in a future version. Convert inf values to NaN
before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[18]:

```
<seaborn.axisgrid.PairGrid at 0x7b8ab4b8bd00>
```



In [19]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['tagText']=le.fit_transform(df['tagText'])
```

In [20]:

```
df.head()
```

Out[20]:

	productTitle	price	sold	tagText
0	Dresser For Bedroom With 9 Fabric Drawers Ward...	46.79	600	1
1	Outdoor Conversation Set 4 Pieces Patio Furnit...	169.72	0	1
2	Desser For Bedroom With 7	39.4	7	1

	Fabric Drawers Organ...	6		
3	Modern Accent Boucle Chair,Upholstered Tufted ...	111.99	0	1
4	Small Unit Simple Computer Desk Household Wood...	21.37	1	1

In [21]:

```
df['tagText'].value_counts()
```

Out[21]:

```
tagText
1    1880
2     111
0       9

Name: count, dtype: int64
```

In [ ]:

[Reference link]