Project Name - E-commerce Furniture Dataset 2024(Part 1)

Project Type - Classification

Industry - Unified Mentor

Contribution - Individual

Member Name - Hare Krishana Mishra

Task - 1

Project Summary -

Project Description:

The project involves analyzing an e-commerce dataset containing 2,000 furniture product records scraped from AliExpress. It aims to explore customer purchasing patterns, price dynamics, and discount impacts on sales. The dataset provides valuable insights into market trends within online furniture retail.

Objective:

To build predictive models for estimating furniture sales (sold) based on product attributes such as originalPrice, price, tagText, and derived features like discount percentage. Additionally, perform exploratory data analysis (EDA) to understand pricing strategies and identify factors influencing sales.

Key Project Details:

Dataset: 2,000 furniture products with attributes including product title, original price, selling price, units sold, and shipping details.

Tech Stack: Python, pandas, NumPy, scikit-learn, matplotlib, seaborn, SQL, Excel.

Data Preprocessing: Missing value handling, price data cleaning, and encoding of categorical variables (tagText).

Exploratory Data Analysis: Distribution analysis of price and sales, correlation between discount and sales, shipping impact on sales volume.

Let's Begin:-

Data Collection

In [ ]:

```
# Import necessary libraries
import pandas as pd
```

In [ ]:

```
# Load dataset
df = pd.read_csv('/content/ecommerce_furniture_dataset_2024.csv')
```

In [ ]:

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

Out[ ]:

	productTitle	originalPrice	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

Data Preprocessing

In [ ]:

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

```
productTitle        0
originalPrice    1513
price               0
sold                0
tagText             3
dtype: int64
```

In [ ]:

```
df.shape
```

Out[ ]:

```
(2000, 5)
```

In [ ]:

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

In [ ]:

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

In [ ]:

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 487 entries, 2 to 1983
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   productTitle   487 non-null    object
 1   originalPrice  487 non-null    object
```

```
 2   price          487 non-null    object
 3   sold           487 non-null    int64
 4   tagText        487 non-null    int8
dtypes: int64(1), int8(1), object(3)
memory usage: 19.5+ KB
None
```

Exploratory Data Analysis (EDA)

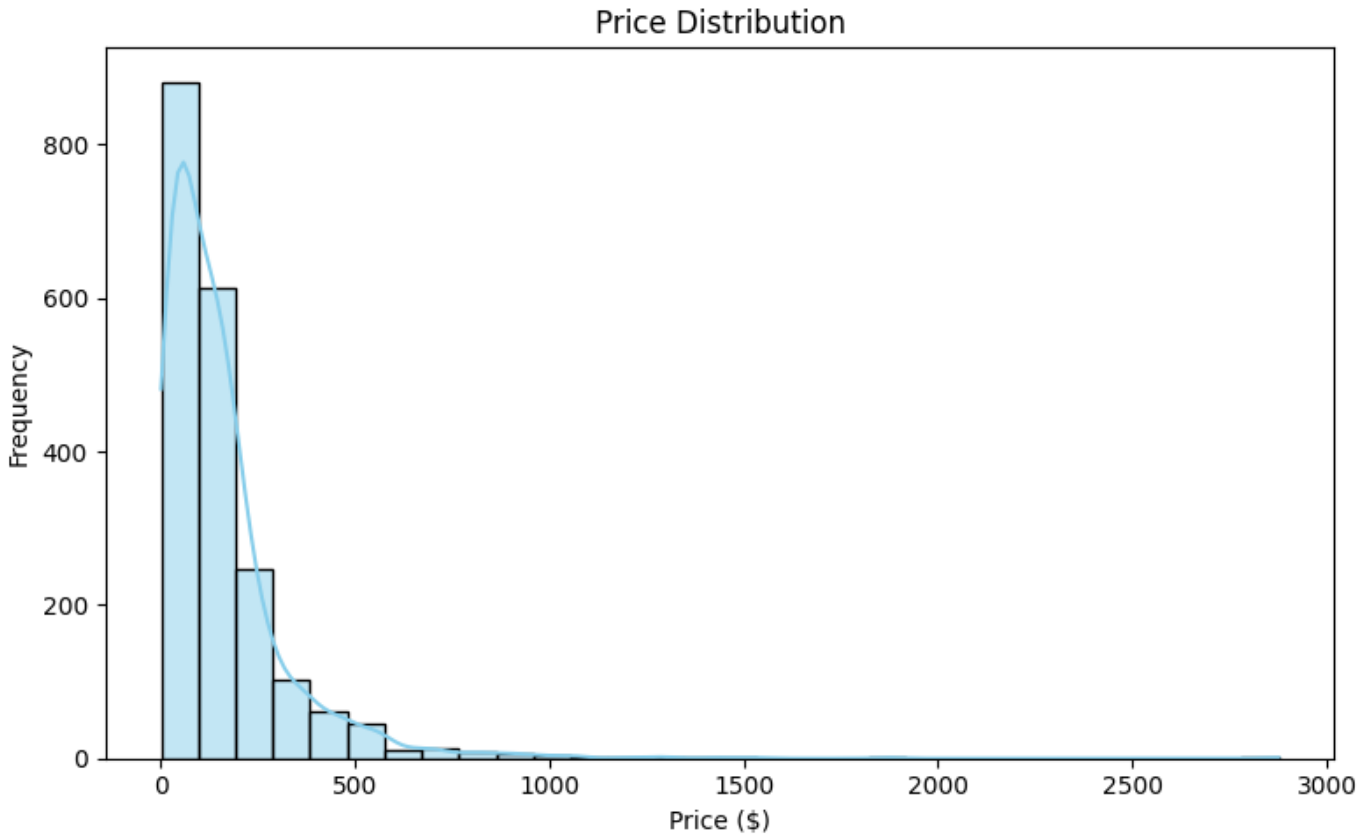
In [ ]:

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

Price Distribution Histogram

In [ ]:

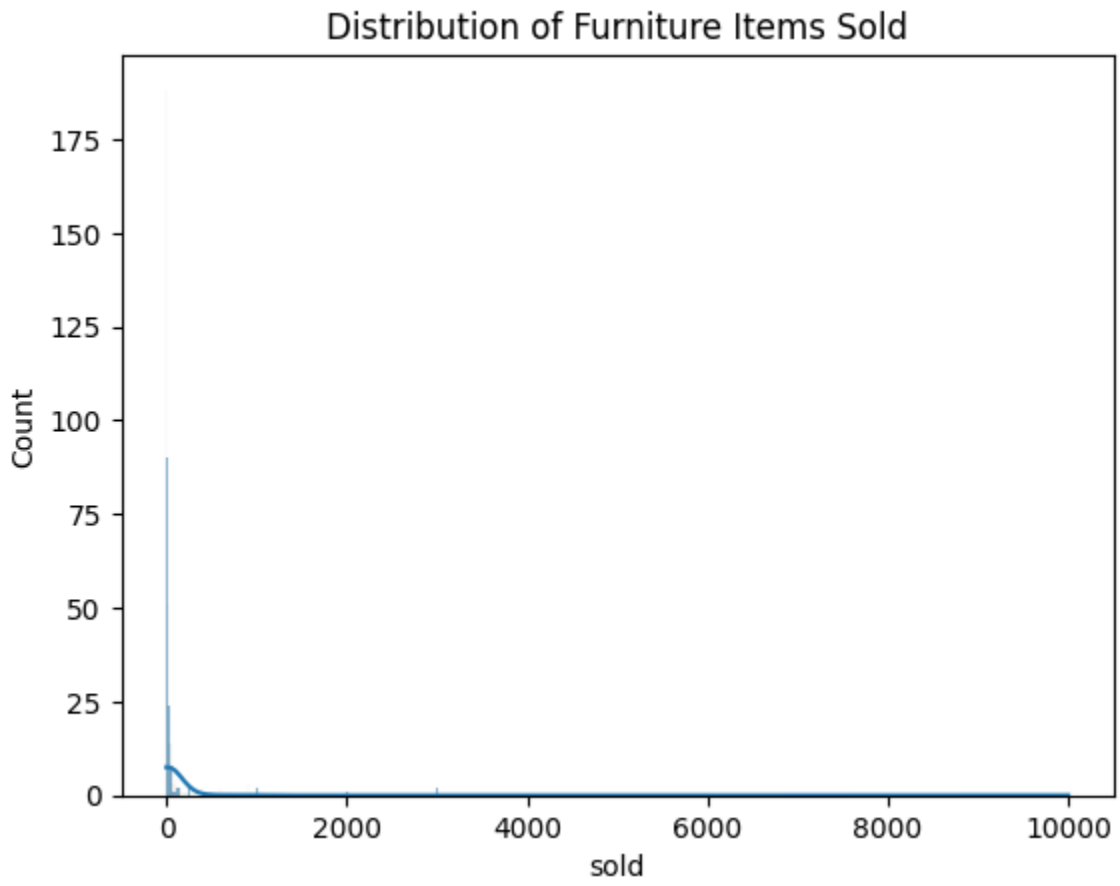
```
# Clean price column (remove $ and commas, convert to float)
df['price'] = df['price'].replace('[\$,]', '', regex=True).astype(float)

# Ensure 'sold' is numeric
df['sold'] = pd.to_numeric(df['sold'], errors='coerce').fillna(0).astype(int)

# Handle missing shipping info
df['tagText'] = df['tagText'].fillna('Unknown')

# Group other shipping types
df['tagText'] = df['tagText'].apply(lambda x: x if x in ['Free shipping', '+Shipping: $5

# Price distribution histogram
plt.figure(figsize=(8,5))
sns.histplot(df['price'], bins=30, kde=True, color='skyblue', edgecolor='black')
plt.title('Price Distribution')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



Distribution of 'sold' values

In [ ]:

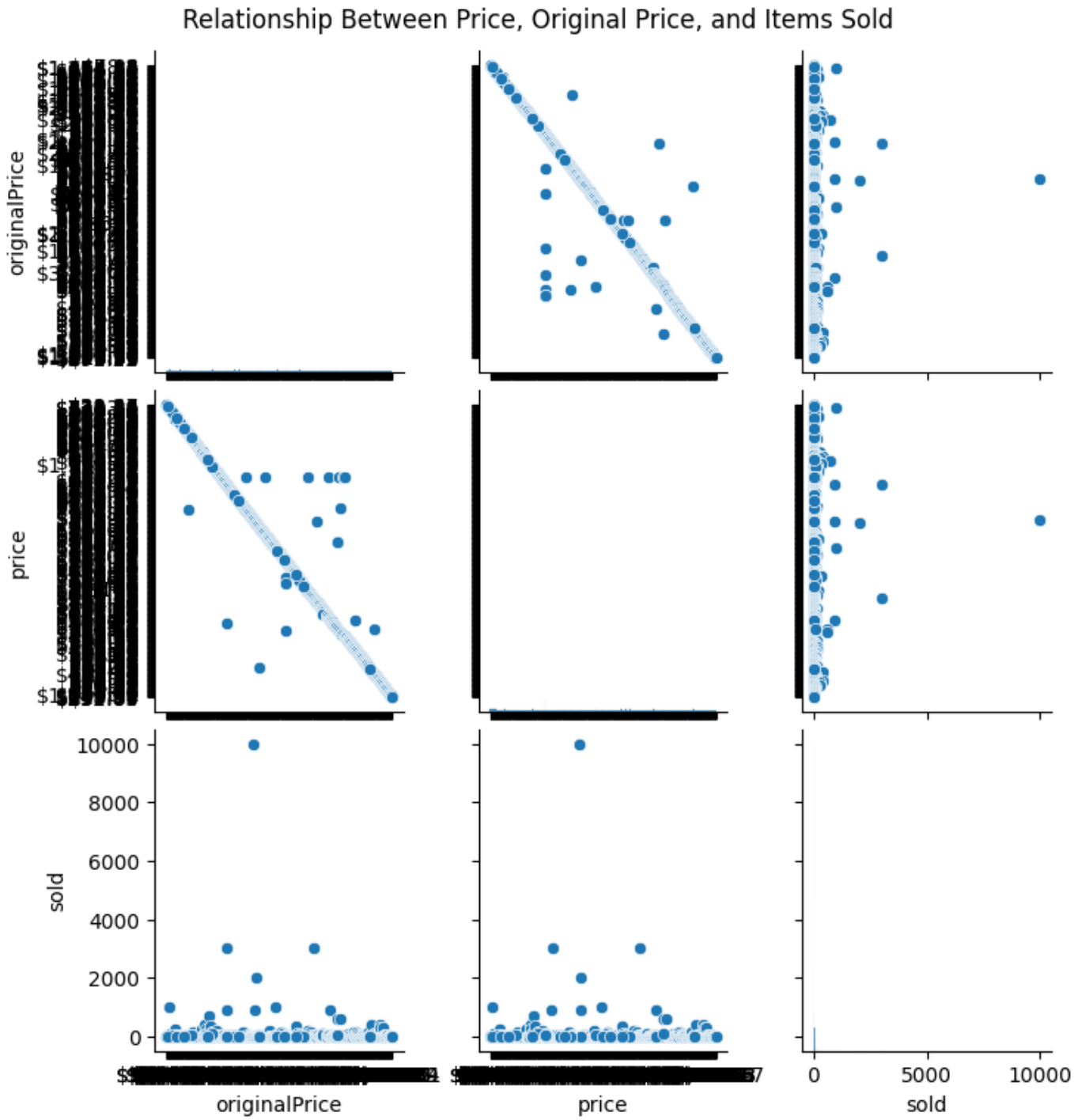
```
# Distribution of 'sold' values
sns.histplot(df['sold'], kde=True)
plt.title('Distribution of Furniture Items Sold')
plt.show()
```



Relationship Between Price, Original Price, and Items Sold

In [ ]:

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

# Pairplot
pairplot = sns.pairplot(df, vars=['originalPrice', 'price', 'sold'], kind='scatter')

# Set overall title for the entire figure
pairplot.fig.suptitle('Relationship Between Price, Original Price, and Items Sold',y=1.0

plt.show()
```

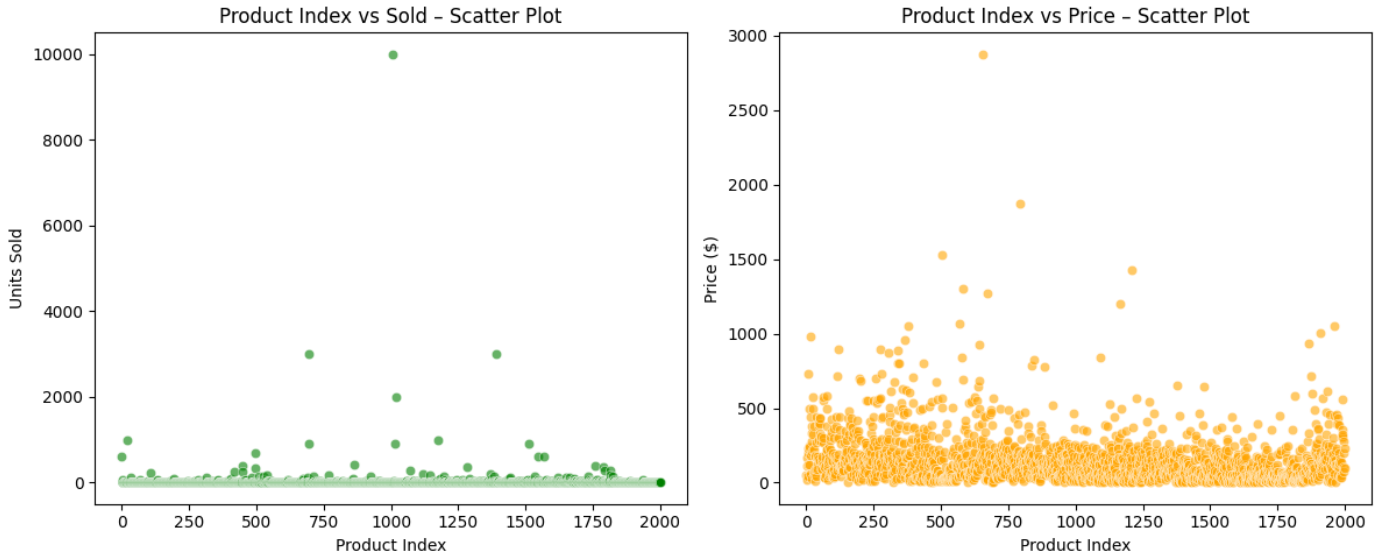


Multi-Variable Scatter Plot Analysis of Furniture Dataset

In [ ]:

```
# Create a figure with 2 scatter plots side by side
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# 1. Sold vs Product Index
sns.scatterplot(x=df.index, y='sold', data=df, alpha=0.6, color='green', ax=axes[0])
axes[0].set_title('Product Index vs Sold — Scatter Plot')
axes[0].set_xlabel('Product Index')
axes[0].set_ylabel('Units Sold')

# 2. Price vs Product Index
sns.scatterplot(x=df.index, y='price', data=df, alpha=0.6, color='orange', ax=axes[1])
axes[1].set_title('Product Index vs Price — Scatter Plot')
```

```
axes[1].set_xlabel('Product Index')
axes[1].set_ylabel('Price ($)')

plt.tight_layout()
plt.show()
```



Feature Engineering

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
# Create a new feature: percentage discount
# Convert to numeric (handle currency symbols or commas if present)
df['originalPrice'] = pd.to_numeric(df['originalPrice'], errors='coerce')
df['price'] = pd.to_numeric(df['price'], errors='coerce')

# Create discount percentage column
df['discount_percentage'] = ((df['originalPrice'] - df['price']) / df['originalPrice'])
```

In [ ]:

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

In [ ]:

```
# 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)
```

In [ ]:

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

Model Selection & Training

In [ ]:

```
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
```

In [ ]:

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

In [ ]:

```
# 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)
```

In [ ]:

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

In [ ]:

```
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
from sklearn.impute import SimpleImputer
import pandas as pd

# 1. Drop rows where 'sold' (target) is NaN
df = df.dropna(subset=['sold'])

# 2. Drop columns that have all NaN values
df = df.dropna(axis=1, how='all')

# 3. Separate features and target
X = df.drop('sold', axis=1)
y = df['sold']

# 4. Impute remaining missing values in features (mean strategy)
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)

# 5. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_

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

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

# 8. Predict and evaluate
y_pred_lr = lr_model.predict(X_test)
y_pred_rf = rf_model.predict(X_test)
```

```
print("Linear Regression R2 Score:", r2_score(y_test, y_pred_lr))
print("Random Forest R2 Score:", r2_score(y_test, y_pred_rf))
```

```
Linear Regression R2 Score: -0.2813506640797141
Random Forest R2 Score: -0.004362510083186333
```

Model Evaluation

In [ ]:

```
# 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)
```

In [ ]:

```
# 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)
```

In [ ]:

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

```
Linear Regression MSE: 176345.83137634074, R2: -0.2813506640797141
Random Forest MSE: 138225.34830545762, R2: -0.004362510083186333
```