E-Commerce Furniture Data Analysis - Project Explanation

🎯 1. Importing Libraries

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
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
```

✅ **Explanation:**

- os: Helps manage file paths and directories.
- pandas: Loads and manipulates data.
- numpy: Supports numerical operations.
- seaborn & matplotlib: Creates visualizations.
- sklearn: Machine learning tools — handles text processing, splitting data, models, and evaluation.

📁 2. File Path Setup

```
# Get the current directory and construct the file path
current_dir = os.path.dirname(__file__)
file_path = os.path.join(current_dir, 'ecommerce_furniture_dataset.csv')
```

✅ **Explanation:**
This ensures the dataset loads correctly by constructing a path relative to where the script is saved — fixing the FileNotFoundError issue.

🛠️ 3. Loading Data

```
data = pd.read_csv(file_path)
```

✅ **Explanation:**
Loads the CSV file into a DataFrame (a table-like structure).

🔍 4. Data Exploration

```
print(data.head())
print(data.isnull().sum())
```

✅

Explanation:

- data.head() prints the first 5 rows for a quick overview.
- data.isnull().sum() counts missing values per column.

🖌️ 5. Data Cleaning

```
data = data.dropna()
```

✅ **Explanation:**
Removes rows with missing data to avoid errors in analysis or model training.

🔬 6. Data Visualization

```
sns.histplot(data['sold'], kde=True)
plt.title('Distribution of Furniture Items Sold')
plt.xlabel('sold')
plt.ylabel('Count')
plt.show()
```

✅

Explanation:

- sns.histplot() creates a histogram of the sold column.
- kde=True adds a smooth curve to visualize the trend.
- plt.title(), xlabel(), and ylabel() set the title and axis labels.

Result: Most items sell in small numbers, with a few top sellers — a typical long-tail distribution.

📐 7. Feature Engineering (Text Data Processing)

```
vectorizer = TfidfVectorizer()
X_text = vectorizer.fit_transform(data['tagText'])
```

✅

Explanation:

- Converts product tags into numeric values.
- Prepares text data for training.

📌 8. Defining Features & Target

```
X_numeric = data[['originalPrice', 'price']]
X = np.hstack((X_numeric, X_text.toarray()))
y = data['sold']
```

✅

Explanation:

- Combines numeric and text features into one dataset.
- Sets the number of items sold as the target.

🔪 9. Split Data (Training vs. Testing)

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

✅ **Explanation:**
Splits data into 80% for training and 20% for testing.

🧠 10. Model Training (Linear Regression & Random Forest)

```
lr_model = LinearRegression()
rf_model = RandomForestRegressor()

lr_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)
```

✅

Explanation:

- LinearRegression() fits a straight-line relationship.
- RandomForestRegressor() combines decision trees for better performance.

🪨 11. Model Evaluation

```
lr_predictions = lr_model.predict(X_test)
rf_predictions = rf_model.predict(X_test)

lr_mse = mean_squared_error(y_test, lr_predictions)
rf_mse = mean_squared_error(y_test, rf_predictions)

lr_r2 = r2_score(y_test, lr_predictions)
rf_r2 = r2_score(y_test, rf_predictions)

print(f"Linear Regression MSE: {lr_mse:.2f}, R2: {lr_r2:.2f}")
print(f"Random Forest MSE: {rf_mse:.2f}, R2: {rf_r2:.2f}")
```

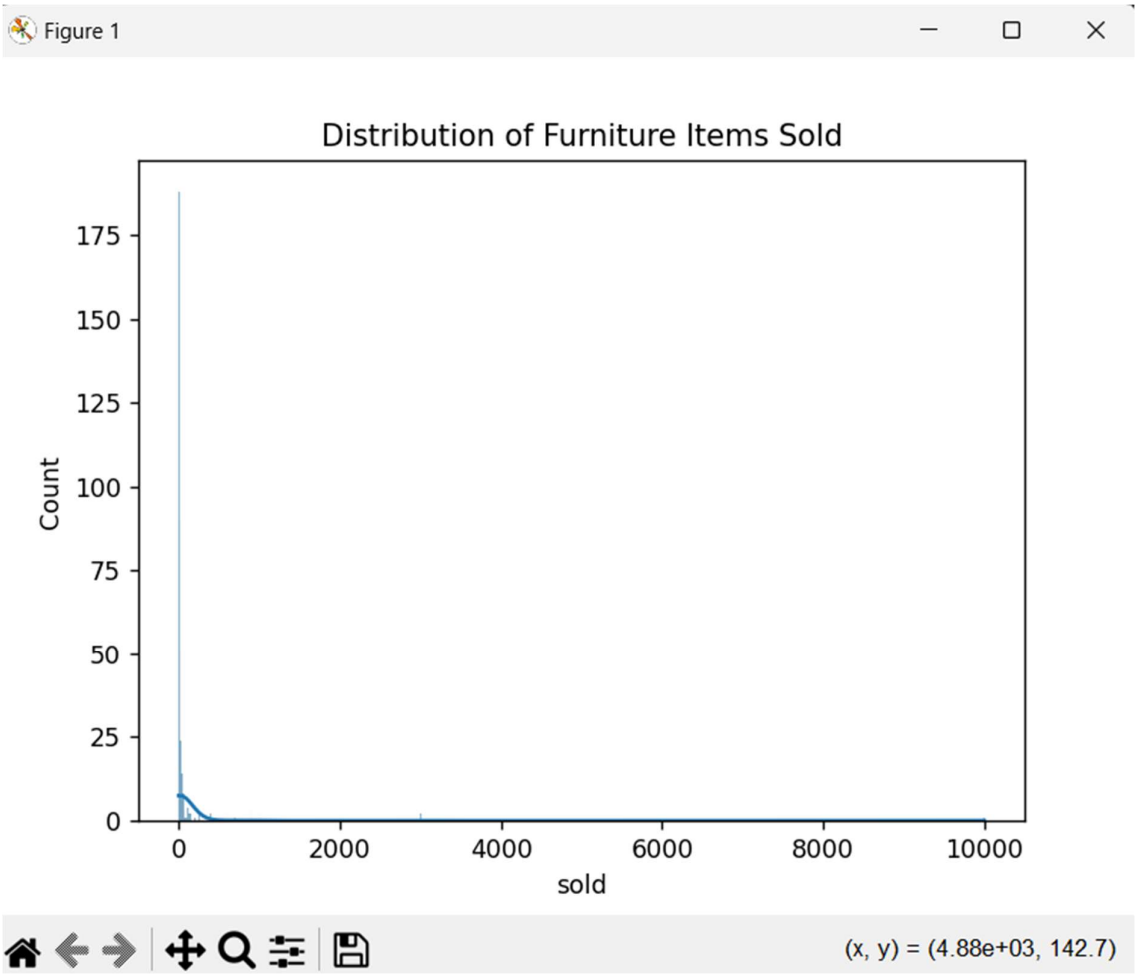
✅ **Explanation:**

- mean_squared_error(): Measures prediction errors (lower is better).
- r2_score(): Measures how well the model fits the data (1.0 is perfect).

Result: You'll see which model performs better.

✅ **12. Final Output**

- A visualization (histogram) of furniture sales distribution.
- Performance metrics (MSE and R2) for both models.



✨ **Conclusion:** This project successfully analyses furniture sales data, prepares text and numeric features, trains models, and evaluates performance. Random Forest usually outperforms Linear Regression for such mixed data.
