E-commerce Furniture Dataset 2024

● 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())
```

E-commerce Furniture Dataset 2024

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']

E-commerce Furniture Dataset 2024

```
# 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.

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.