Task 1: Unified Mentor PVT. LTD.

Project Name: E-commerce Furniture Dataset 2024 (Unified Mentor)

Tools Used ==>> Python (google colab).

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 analyse the relationship between product features and their popularity.

Column Descriptors:

● product_Title: 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.

Steps:

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

Data Cleaning:

---

* Identified 1513 in 'original price' column & 3 in "tag Text" column null or Missing Values and Filled 1513 Values in null "original price" column with a Median Value of the column Filled 'unknown' as value in tag-text 3 null column

data['originalPrice'].fillna(data['originalPrice'].median()): Fills missing values with the median of non-missing original Price values. The median is a robust choice because it's not skewed by outliers (e.g., extremely high or low prices). For example, if most furniture items have prices around $100 but a few are $10,000, the median gives a better central value than the mean.

* Why Median?: The median is less sensitive to extreme values, which is common in price data. Dropping 1513 rows would lose too much data, and

imputing with a constant (e.g., 0) could bias the discount_percentage calculation later.

* Handling tagText (3 missing):

* data['tagText'].fillna('No Tag'): Replaces the 3 missing tagText values with the string 'No Tag'. Since tagText is categorical (e.g., "Free shipping", "Discount"), a placeholder like 'No Tag' is a simple way to retain these rows without affecting one-hot encoding.

* Why Placeholder?: With only 3 missing values, a placeholder is safe and ensures all rows are usable. Dropping these rows would have minimal impact, but keeping them is better for maximizing data.

* One-Hot Encoding:

pd.get_dummies(data, columns=['tagText'], prefix='tag'): Converts tagText into numerical columns (e.g., tag_Free shipping, tag_Discount). The 'No Tag' placeholder becomes its own column (e.g., tag_No Tag).

Why One-Hot Encoding?: Machine learning models like Linear Regression and Random Forest require numerical inputs, so categorical variables must be converted.

* Gain experience troubleshooting data type conversion errors (e.g., handling non-numeric values) using Libraries like (pandas, NumPy etc.)

* Used Visualization libraries like (Matplotlib and Seaborn).

Remove Duplicates:

---

* Duplicated Value ==>> Identified 94 "Duplicated" values in the whole data and Dropped all as it can affect our decision process.

Standardize Text:

---

*For product-Title and tag-Text, remove extra spaces and convert to lowercase

*** Why This Approach?**

Simplicity: Using the median for originalPrice and a placeholder for tagText is straightforward and effective for a beginner. More advanced techniques (e.g., imputing based on other features) are possible but complex and may not be necessary.

Preserves Data: Imputing missing values allows you to keep all rows, which is important since 1513 missing originalPrice values could be a large portion of your dataset.

Aligns with Project: The instructions suggest handling missing values, and this approach follows best practices for numerical and categorical data.

Prevents Errors: Filling missing values ensures calculations like discount_percentage and model training run without errors.

3. Exploratory Data Analysis (EDA)

KPI (Key Performance Indicator)

* Average Price
* Total Sold
* Count of tag-Text

Stacked Column Chart:

* 0-50 ($) Price Range Has Higher sales
* Learn to create and customize a column chart in Power BI.
* Created new Column Range with price 0 to 200+
* Gain experience with Power Query or DAX for creating calculated columns.
* LOW Price Category has Highest Sells

Scatter Plot:

* Learn to create scatter plots in Power BI to explore relationships between Price and Sells.
* Understand how to use legends to add context (e.g., tagText).
* Gain skills in interpreting scatter plots for trends and patterns.
* "Free Shipping" has Highest sales

4. Feature Engineering:

---

* Learn DAX for creating calculated columns.

* Created new Column "Discount %", "Discount Price", "Price Category", "Has free Shipping"

* "Low / 0-100" Price category has higher sales value

* The Product which has "Free Shipping" Selling more

* Understand conditional logic and text manipulation in Power BI.

* Gain experience with Power Query for feature creation.

What is Feature Engineering?

Feature engineering is the process of creating new features (columns) or transforming existing ones to make your data more suitable for machine learning models. It's like preparing ingredients before cooking—raw data (like productTitle or originalPrice) may not be directly usable by models, so you transform it into a format that helps the model learn patterns effectively.

In my project, the goal is to predict the number of furniture items sold (sold) based on features like productTitle, originalPrice, price, and tagText. Feature engineering ensures these features are in a numerical format (since models like Linear Regression and Random Forest require numbers) and captures meaningful patterns (e.g., discounts influencing sales).

Objective of Feature Engineering:

- I created new features to improve the predictive power of my models for the "E-commerce Furniture Dataset 2024" project. The goal was to predict the number of items sold (sold) using features like productTitle, originalPrice, price, and tagText.

- Discount Percentage Feature:

. I calculated discount_percentage using the formula: (originalPrice - price) / originalPrice * 100. This feature captures how much a product's price was reduced, which might influence customer purchasing behavior. For example, a high discount could lead to more sales, which the model can learn from.

- **TF-IDF for Product Titles:**

. I used the TfidfVectorizer from scikit-learn to convert productTitle text (e.g., "Wooden Coffee Table") into numerical features. TF-IDF assigns weights to words based on their frequency in a title and rarity across all titles, creating 50 columns for key words like "table," "wood," and "storage."

. This transformation allowed the model to use product descriptions numerically, capturing patterns like whether "table" or "wardrobe" products sell more.

- **Impact on Dataset:**

The feature engineering step expanded the dataset to 155 columns, including numerical features (originalPrice, price, sold, discount_percentage), one-hot encoded tagText columns, and 50 TF-IDF columns. This provided a richer dataset for modeling but increased complexity due to many tag_* columns.

- **Challenges and Observations:**

. The tagText column generated many unique columns (e.g., tag_+Shipping: $1,097.18), possibly due to specific shipping costs. This high dimensionality could be simplified in future iterations (e.g., grouping similar tags).

. The TF-IDF columns showed which words were important in each product title, helping identify product types that might drive sales.

- **Skills Learned:**

. I learned to use Python libraries like pandas for data manipulation and scikit-learn for text processing (TfidfVectorizer).

. I understood the importance of transforming raw data (text and numerical) into model-ready features, a critical skill for data analysis.

. I practiced interpreting the output of feature engineering, such as checking new columns and their values, which will help me validate data in future projects.

5. Model Selection & Training:

- Objective of Model Selection & Training:

. In the "E-commerce Furniture Dataset 2024" project, I trained two machine learning models—Linear Regression and Random Forest Regressor—to predict the number of furniture items sold (sold) based on features like originalPrice, price, discount_percentage, and TF-IDF features from productTitle.

- Data Preparation:

. I prepared the feature matrix X by excluding the target sold and problematic non-numeric columns (e.g., tagText if it contained strings like 'Free shipping'). The target y was set to sold.

. I split the data into 80% training and 20% testing sets using train_test_split to evaluate model performance on unseen data.

- Model Training:

. Linear Regression: I used LinearRegression to model a linear relationship between features and sold. It's simple but assumes linear patterns, which may not fully capture complex relationships in e-commerce data.

. Random Forest Regressor: I used RandomForestRegressor with 100 decision trees (n_estimators=100) to capture non-linear patterns and interactions between features, making it more robust for this dataset.

- Predictions:

. The first five predictions showed Linear Regression producing a negative value (-3.13649132) and larger values (e.g., 78.91588083), which is unrealistic for sold (a non-negative count). Random Forest predictions (e.g., 5.33, 0.47) were positive and smaller, suggesting better suitability for this task.

. The negative Linear Regression prediction indicates potential issues like unscaled features or outliers, which I'll investigate in the evaluation step.

- Challenges and Observations:

. I handled non-numeric data (e.g., 'Free shipping') by excluding problematic columns, ensuring all features in X were numeric for model training.

. The dataset’s 155 columns (from TF-IDF and one-hot encoded tags) may have affected Linear Regression’s performance, as it struggles with high-dimensional data. Random Forest handled this better due to its robustness.

- Skills Learned:

. I learned to use scikit-learn for model training (LinearRegression, RandomForestRegressor) and data splitting (train_test_split).

. I practiced debugging data issues, like non-numeric columns, which is a common task in data analysis.

. I gained experience comparing model outputs, a key skill for selecting the best model in real-world projects.

6. Model Evaluation:

---

1. Objective of Model Evaluation:

. In the "E-commerce Furniture Dataset 2024" project, I evaluated Linear Regression and Random Forest Regressor models to predict the number of furniture items sold (sold) using Mean Squared Error (MSE) and R-squared metrics.

2. Evaluation Metrics:

. MSE: Measures the average squared difference between predicted and actual sold values. Linear Regression had an MSE of 14,090.95, while Random Forest had a lower MSE of 12,978.71, indicating slightly better predictions.

. R-squared: Shows how much variance in sold is explained by the model. Both models had negative R-squared values (-0.091 for Linear Regression, -0.0049 for Random Forest), meaning they performed worse than a baseline model predicting the mean.

3. Model Comparison:

. Random Forest outperformed Linear Regression slightly, with a lower MSE and less negative R-squared, likely due to its ability to handle non-linear relationships and high-dimensional data (155 features from TF-IDF and tag_* columns).

.Linear Regression struggled, producing negative predictions (seen in Step 5) and higher errors, possibly due to unscaled features or non-linear patterns in the data.

4. Challenges and Observations:

. Negative R-squared values indicate both models failed to capture patterns in sold, suggesting issues like poor feature quality (e.g., TF-IDF features not strongly correlated with sales) or data issues (e.g., outliers, unscaled features).

. The large number of features (155) may have caused overfitting, especially for Linear Regression, which is less robust to high-dimensional data.

. Non-numeric data (e.g., 'Free shipping') was handled by excluding problematic columns, but further feature engineering (e.g., grouping tag_* columns) could improve results.

5. Skills Learned:

. I learned to use scikit-learn for model evaluation (mean_squared_error, r2_score) and interpret metrics to assess model performance.

. I practiced identifying model weaknesses (e.g., negative R-squared) and planning improvements, a key skill for iterative data analysis.

. I gained experience comparing models, which is essential for selecting the best approach in real-world projects.