Dr. D. Y. Patil Vidyapeeth, Pune
Dr. D. Y. Patil School of Science and Technology, Tathwade

Topic: Meesho Data Analysis Using Python

Sohan Mondal
B.Tech (Computer Science & Design)
Second Year (3rd Sem)
Roll: BTCS-2353

PROBLEM STATEMENT

Meesho, as a leading e-commerce platform, offers a wide variety of products across different categories. With the growing scale of operations. It is crucial to understand trends in sales, customer preferences and product performance. This project aims to analyze Meesho's sales and customer data to uncover valuable insights that can drive strategic decision-making.

The key objectives are:

- 1. Identify the most profitable product categories and regions.
- 2. Understand the relationship between pricing, ratings and sales performance.
- 3. Predict future sales trends based on historical data.
- 4. Provide actionable insights for improving customer engagement and optimizing product pricing.

Through Python-based data analysis and visualization, this project seeks to provide meaningful solutions to improve business outcomes and customer satisfaction.

SELECT SUITABLE DATASET

Dataset Requirements:
Columns like: Product_ID, Category, Price, Sales, Ratings, Purchase_Date, Customer_ID, Region, etc.

Sources:
- Search for open datasets on Kaggle or Meesho's insights reports.
- Alternatively, create synthetic data that reflects real-world trends based on Meesho's model.

IMPLEMENT PROJECT USING PYTHON

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('meesho_data.csv')

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(data.head())

# Data cleaning
# Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())

# Drop rows with missing values
data.dropna(inplace=True)

# Convert 'date' to datetime format
data['date'] = pd.to_datetime(data['date'])
```

IMPLEMENT PROJECT USING PYTHON

```
# Basic statistics
print("\nBasic statistics of the dataset:")
print(data.describe())

# Exploratory Data Analysis (EDA)

# 1. Sales trend over time
sales_trend = data.groupby('date')['quantity_sold'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(data=sales_trend, x='date', y='quantity_sold')
plt.title('Sales Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Quantity Sold')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# 2. Distribution of product categories
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='category', order=data['category'].value_counts().index)
plt.title('Product Category Distribution')
plt.xticks(rotation=45)
```

IMPLEMENT PROJECT USING PYTHON

```
plt.ylabel('Number of Products')
plt.tight_layout()
plt.show()

# 3. Average price of products by category
avg_price_by_category = data.groupby('category')['price'].mean().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(data=avg_price_by_category, x='category', y='price', palette='viridis')
plt.title('Average Price of Products by Category')
plt.xticks(rotation=45)
plt.ylabel('Average Price')
plt.tight_layout()
plt.show()

# 4. Customer demographics analysis
plt.figure(figsize=(10, 6))
sns.countplot(data=data, x='customer_gender')
plt.title('Customer Gender Distribution')
plt.ylabel('Number of Customers')
plt.tight_layout()
plt.show()
```

IMPLEMENT PROJECT USING PYTHON

```
# 5. Age distribution of customers
plt.figure(figsize=(10, 6))
sns.histplot(data['customer_age'], bins=10, kde=True)
plt.title('Customer Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# Save cleaned data to a new CSV file
data.to_csv('cleaned_meesho_data.csv', index=False)

print("\nData analysis complete. Cleaned data saved as 'cleaned_meesho_data.csv'.")
```

VISUALIZE DATA WITH BOX PLOT, HISTOGRAM, SCATTER PLOT.

1. Box Plot

```
sns.boxplot(x=data['Price'], color='skyblue’)
plt.title('Price Distribution’)
plt.xlabel('Price’)
plt.show()
```

2. Histogram

```
data['Sales'].hist(bins=20, color='skyblue', alpha=0.7)
plt.title('Sales Distribution')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.show()
```

3. Scatter Plot

```
plt.scatter(data['Price'], data['Sales'], alpha=0.5, color='blue')
plt.title('Price vs Sales')
plt.xlabel('Price')
plt.ylabel('Sales')
plt.show()
```

PLOT PEARSON CORRELATION AND EXPLAIN ABOUT RELATION.

The Pearson correlation matrix helps identify relationships between numeric variables in your dataset.

Code to Plot Correlation Matrix:

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

# Compute the correlation matrix
correlation_matrix = data.corr()

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```

PLOT PEARSON CORRELATION AND EXPLAIN ABOUT RELATION.

Explanation of Relationships:

1. Correlation Coefficients:

- Values range from **-1 to 1**.
- 1: Perfect positive correlation (as one variable increases, the other increases).
- 0: No correlation.
- -1: Perfect negative correlation (as one variable increases, the other decreases).

2. Interpret Relationships:

- Example: If Price has a negative correlation with Sales, it indicates higher prices result in lower sales.
- Example: If Ratings has a positive correlation with Sales, better ratings lead to higher sales.

3. Key Insights:

Look for strong correlations (>0.5 or <-0.5) to identify meaningful relationships.
Weak correlations (<0.3) may not be significant.

IDENTIFY DEPENDENT AND INDEPENDENT FEATURES

In my project, the features (variables) in the dataset can be divided into **Independent** and **Dependent** variables based on their roles in the analysis.

Dependent Feature (Target Variable):
- The dependent feature is the variable we aim to predict or analyze. It is influenced by other features (independent variables) in the dataset.
- In this case:
  - ‘Sales’ is the dependent feature because the project's objective is to analyze and predict sales performance based on other factors like price, ratings, and discounts.

Independent Features (Predictor Variables):
- Independent features are variables that are expected to influence or explain the dependent variable. These are used to predict or analyze the target variable.
- In this case:
  - ‘Price’: The price of the product influences sales as it affects customer purchasing decisions.
  - ‘Ratings’: Higher-rated products are likely to attract more customers.
  - ‘Discount’: Discounts can increase sales by incentivizing purchases.
  - ‘Category’: Different product categories may have varying levels of popularity and sales.
  - ‘Region’: Sales may vary across geographic regions.
  - ‘Purchase_Date’: Temporal trends (e.g., festive seasons) can impact sales.

ANALYSE /PREDICT AS PER PROBLEM STATEMENT

The goal is to analyze the factors influencing **sales** and predict future sales trends based on the independent variables.

1. Exploratory Data Analysis (EDA)

- Correlation Analysis:
- Strong correlations between features like Price, Ratings, and Sales can be identified.
- Positive correlation between Ratings and Sales suggests that higher-rated products tend to sell more.
- Negative correlation between Price and Sales indicates that higher prices may reduce sales.

- Trend Analysis:
- **Monthly Sales Trends**: Group data by Purchase_Date to analyze how sales vary over time (e.g., festive periods, seasons).
- **Category and Region-based Analysis**: Identify which product categories or regions generate the most sales.

2. Predicting Sales Using Regression Model

We will predict **Sales** using independent variables like **Price**, **Ratings**, and **Discount**.

- Linear Regression Model:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Selecting features and target
X = data[['Price', 'Ratings', 'Discount']]  # Independent variables
y = data['Sales']  # Dependent variable
```

ANALYSE /PREDICT AS PER PROBLEM STATEMENT

```
# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Building the regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Evaluating the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R^2 Score: {r2}")
```

Results Interpretation:
MSE (Mean Squared Error): Measures the average squared difference between actual and predicted sales. Lower values indicate better predictions.
R^2 Score: Indicates how well the independent variables explain the variance in sales. A value close to 1 means the model is a good fit.

ANALYSE /PREDICT AS PER PROBLEM STATEMENT

3. Actionable Insights and Predictions

- **Pricing Strategy**: Based on the negative correlation between Price and Sales, consider offering competitive pricing strategies or discounts to increase sales.
- **Rating Improvement**: Since Ratings positively affect sales, improving product quality and customer satisfaction could lead to higher sales.
- **Sales Forecasting**: Use the regression model to forecast future sales trends, helping with inventory planning and marketing strategies.