Sales Data Analysis

Project Report

By: Siddhant Prakash Satote

Date: 18–01-2025

Table of Contents

1. Summary

This report provides a comprehensive analysis of sales data using the Sample Superstore dataset. It aims to identify trends, patterns, and key performance indicators (KPIs) to optimize sales strategies and improve revenue generation. The report also includes a sales forecasting model to predict future sales. The insights gained from this analysis can help businesses make data-driven decisions to improve profitability and customer satisfaction.

2. Introduction

Sales analysis is a fundamental part of business intelligence. It provides valuable insights into customer preferences, market trends, and business performance. By analyzing historical sales data, businesses can optimize inventory, improve customer targeting, and enhance sales strategies. This project leverages the Sample Superstore dataset to perform a detailed sales analysis, focusing on total sales, profit margins, and product performance across different categories. Additionally, a sales forecasting model is developed to predict future sales trends, helping the business plan for upcoming demand.

3. Data Preparation and Cleaning

Objective: Ensure the data is clean and structured for analysis to avoid inaccuracies.

Steps

1. Load the dataset using Pandas.
2. Check for duplicate entries and handle missing values.
3. Convert date columns to datetime objects for time-series analysis.

Code

```
import pandas as pd

# Load the dataset
data = pd.read_excel('Sample_Superstore.xlsx')
print(data.isnull().sum())  # Check for missing values

# Remove duplicate entries
data = data.drop_duplicates()

# Convert 'Order Date' and 'Ship Date' to datetime
data['Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])
```

Explanation

- Duplicate entries can distort analysis, leading to inaccurate results. The `drop_duplicates` method ensures data integrity.
- Date conversion allows grouping data by time periods, essential for trend analysis.

Output

```
Row ID                   0
Order Priority           0
Discount                 0
Unit Price               0
Shipping Cost            0
Customer ID              0
Customer Name            0
```

```
Ship Mode                0
Customer Segment         0
Product Category         0
Product Sub-Category     0
Product Container        0
Product Name             0
Product Base Margin     16
Country                  0
Region                   0
State or Province        0
City                     0
Postal Code              0
Order Date               0
Ship Date                0
Profit                   0
Quantity ordered new     0
Sales                    0
Order ID                 0
dtype: int64
```

4. Exploratory Data Analysis (EDA)

EDA helps uncover initial patterns, relationships, and anomalies within the data.

Code

```
# Summary statistics
print(data[['Sales', 'Profit', 'Discount', 'Quantity ordered
new']].describe())
```

Explanation

- `describe()` provides statistical insights, including mean, standard deviation, and minimum/maximum values for key numeric fields.

Output

```
              Sales        Profit     Discount  Quantity ordered new
count   1952.000000   1952.000000  1952.000000           1952.000000
mean     985.828832    114.793859     0.048975             12.944672
std     2559.900167   1141.112387     0.031378             13.871565
min        2.250000 -16476.838000     0.000000              1.000000
25%       58.807500    -84.485400     0.020000              5.000000
50%      202.395000      1.476450     0.050000             10.000000
75%      802.945000    116.201575     0.080000             16.000000
max    45737.330000   9228.225600     0.210000            167.000000
```

5. Key Performance Indicators (KPIs)

Total Sales

Objective: Calculate and display the total revenue generated.

Code

```
total_sales = data['Sales'].sum()
print(f'Total Sales: ${total_sales:,.2f}')
```

Explanation

- Summing the `Sales` column gives the overall revenue.

Output

```
Total Sales: $1,924,337.88
```

Profit Margin

Objective: Assess profitability by calculating profit margin.

Code

```
total_profit = data['Profit'].sum()
profit_margin = total_profit / total_sales
print(f'Profit Margin: {profit_margin * 100:.2f}%')
```

Explanation

- The profit margin indicates how much profit is made for every dollar of sales, a key profitability metric.

Output

```
Profit Margin: 11.64%
```

6. Visualizations

Visualizations are crucial for understanding trends and making data actionable.

Monthly Sales Trend

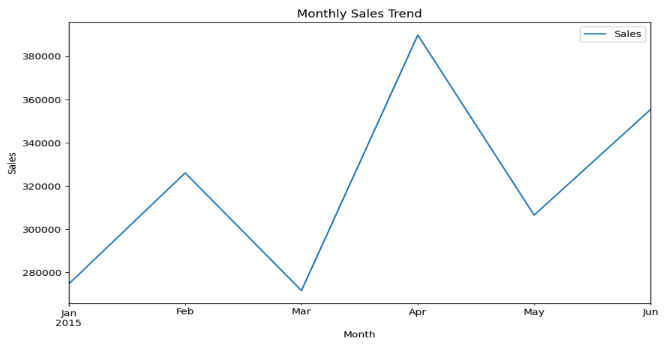
Objective: Analyze and plot monthly sales trends to identify patterns.

Code

```
import matplotlib.pyplot as plt

monthly_sales = data.groupby(data['Order
Date'].dt.to_period('M')).agg({'Sales': 'sum'})
monthly_sales.index = monthly_sales.index.to_timestamp()

plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales['Sales'], label='Monthly
Sales Trend')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()
plt.savefig('images/monthly_sales_trend.png')
plt.show()
```

Explanation

- The plot displays monthly sales trends, revealing peak sales periods and seasonality.



Profit by Product Category

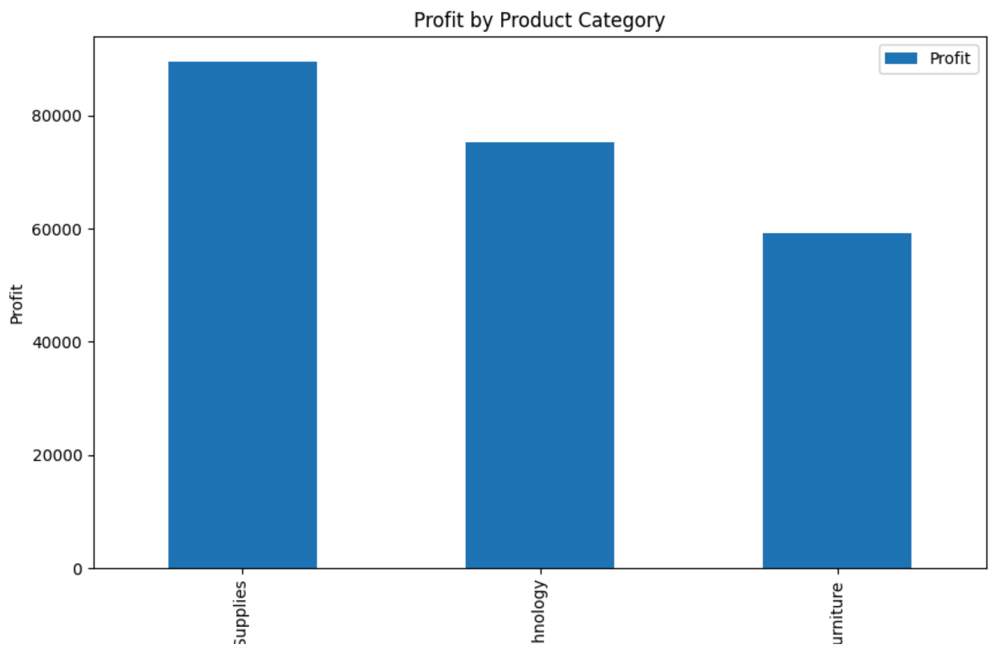
Objective: Visualize profit distribution by product category.

Code

```
category_profit = data.groupby('Product Category').agg({'Profit':
'sum'})
category_profit.plot(kind='bar')
plt.title('Profit by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Profit')
plt.savefig('images/profit_by_category.png')
plt.show()
```

Explanation

- Bar plots make it easy to compare profits across different product categories.



7. Sales Forecasting

Forecasting Model

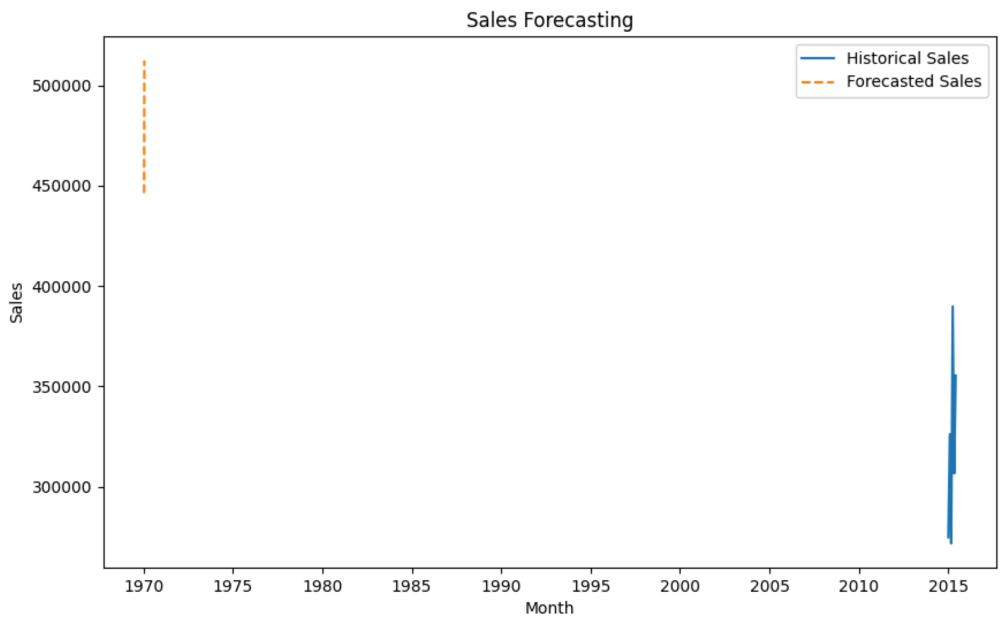
Objective: Use Linear Regression to predict future sales trends.

Code

```
from sklearn.linear_model import LinearRegression
import numpy as np

monthly_sales['Month'] = monthly_sales.index.month
X = np.array(monthly_sales['Month']).reshape(-1, 1)
y = monthly_sales['Sales'].values

model = LinearRegression()
model.fit(X, y)

future_months = np.array(range(13, 19)).reshape(-1, 1)
predictions = model.predict(future_months)

plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales['Sales'],
label='Historical Sales')
plt.plot(range(13, 19), predictions, label='Forecasted Sales',
linestyle='--')
plt.title('Sales Forecasting')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.legend()
plt.savefig('images/sales_forecasting.png')
plt.show()
```

Explanation

- This linear regression model uses historical sales data to project future trends.


Sales Forecasting
Historical Sales
Forecasted Sales
Sales
500000
450000
400000
350000
300000
Month
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015

8. Conclusion

This analysis provides critical insights into sales performance, including key trends, profit drivers, and forecasted sales. These insights can inform strategic decisions to boost revenue, improve customer satisfaction, and optimize inventory management. The forecasting model serves as a foundation for more sophisticated predictive models in the future.

---

9. References

1. Python Data Analysis Library (Pandas) Documentation: https://pandas.pydata.org/pandas-docs/stable/
2. Matplotlib for Data Visualization: https://matplotlib.org/stable/contents.html
3. Scikit-Learn Documentation: https://scikit-learn.org/stable/
