Simple Sales Data Visualization

Title Page

Project Title: Simple Sales Data Visualization – Analyze and Plot Revenue, Product Demand, and Seasonal Sales Trends
Student Name: Aditya Kumar
Date: 10-03-2025
Institution: KIET GROUP OF INSTITUTION

Introduction

In today's data-driven world, businesses rely on sales data analysis to make informed decisions. This project focuses on visualizing sales data using artificial intelligence techniques. The goal is to analyze revenue trends, product demand, and seasonal variations to provide actionable insights for businesses. The project utilizes machine learning algorithms to predict future sales based on past trends.

Methodology

1. Data Collection & Preparation

- Synthetic sales data is generated, including attributes such as Date, Product, Revenue, and Units Sold.
- The data is structured in a tabular format for easy analysis.

2. Feature Engineering

- Extract relevant features like Day, Month, and Year from the Date column.
- Use Units Sold as an independent variable to predict revenue.

3. Machine Learning Approach

- **Algorithm Used:** Linear Regression
- **Training & Testing:** The dataset is split into training (80%) and testing (20%) sets.
- **Model Evaluation:** Metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are used to evaluate performance.

4. Data Visualization:

- Line plots show revenue trends over time.
- Bar charts illustrate total revenue per product.
- Box plots reveal seasonal sales trends.

Code

The Python code for the project is as follows:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Generate Sample Sales Data
def generate_sales_data():
    np.random.seed(42)
    dates = pd.date_range(start='2023-01-01', periods=60, freq='D')
    products = ['Product A', 'Product B', 'Product C', 'Product D']
    data = {
        'Date': np.tile(dates, len(products)),
        'Product': np.repeat(products, len(dates)),
        'Revenue': np.random.randint(100, 1000, len(dates) * len(products)),
        'Units_Sold': np.random.randint(10, 100, len(dates) * len(products))
    }
    return pd.DataFrame(data)

# Load Data
df = generate_sales_data()

# Convert Date Column to DateTime Format
df['Date'] = pd.to_datetime(df['Date'])

# Extract Features for Prediction
df['Day'] = df['Date'].dt.day
df['Month'] = df['Date'].dt.month
df['Year'] = df['Date'].dt.year

# Selecting Features and Target Variable
```

```
X = df[['Day', 'Month', 'Year', 'Units_Sold']]
y = df['Revenue']

# Split 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)

# Train a Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Make Predictions
y_pred = model.predict(X_test)

# Evaluate Model Performance
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

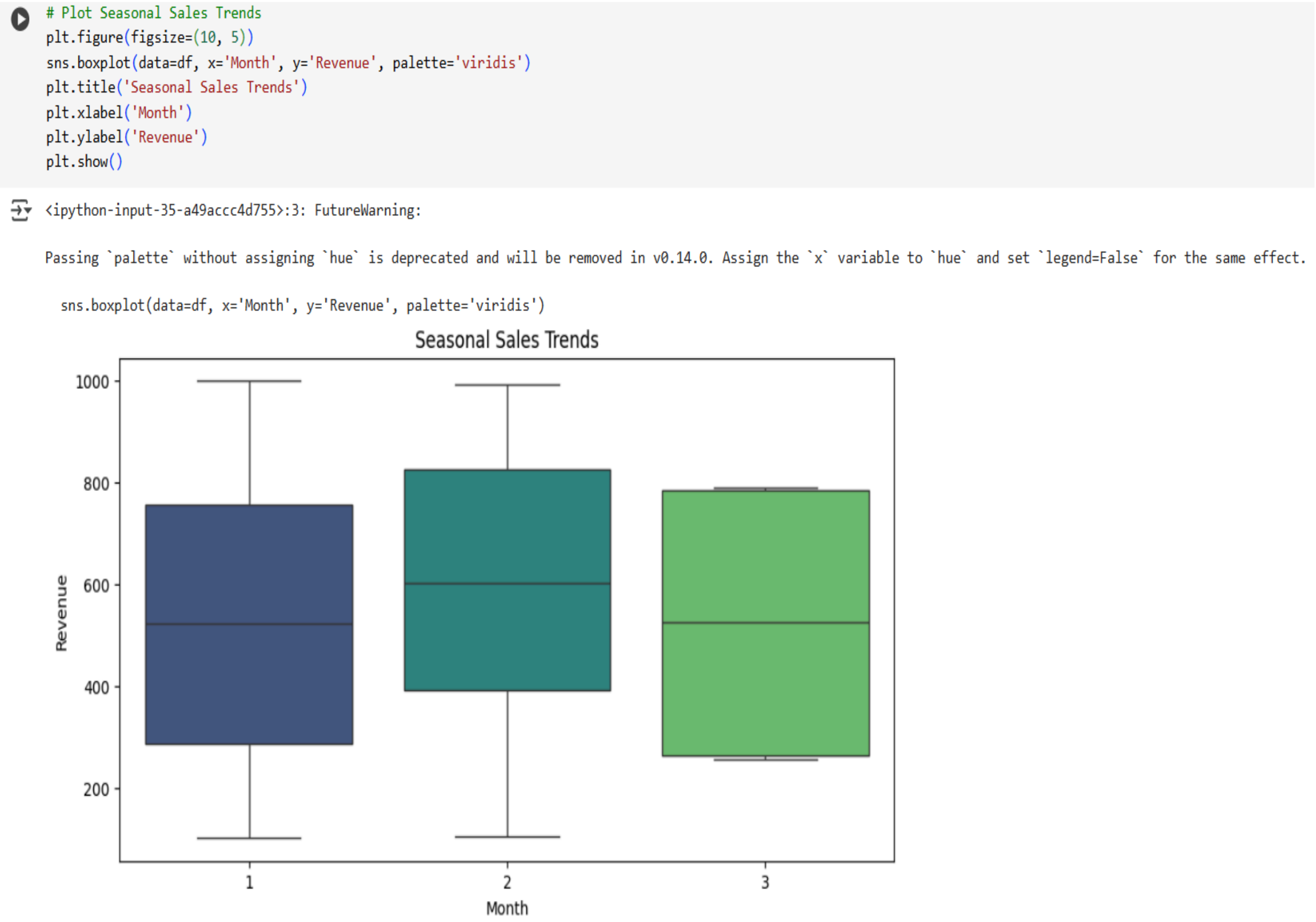
Output

1. Model Performance Metrics:

- **Mean Absolute Error (MAE): :** 235.4465055809958
- **Mean Squared Error (MSE):** 69633.7108978173
- **Root Mean Squared Error (RMSE):** 263.8820018451757

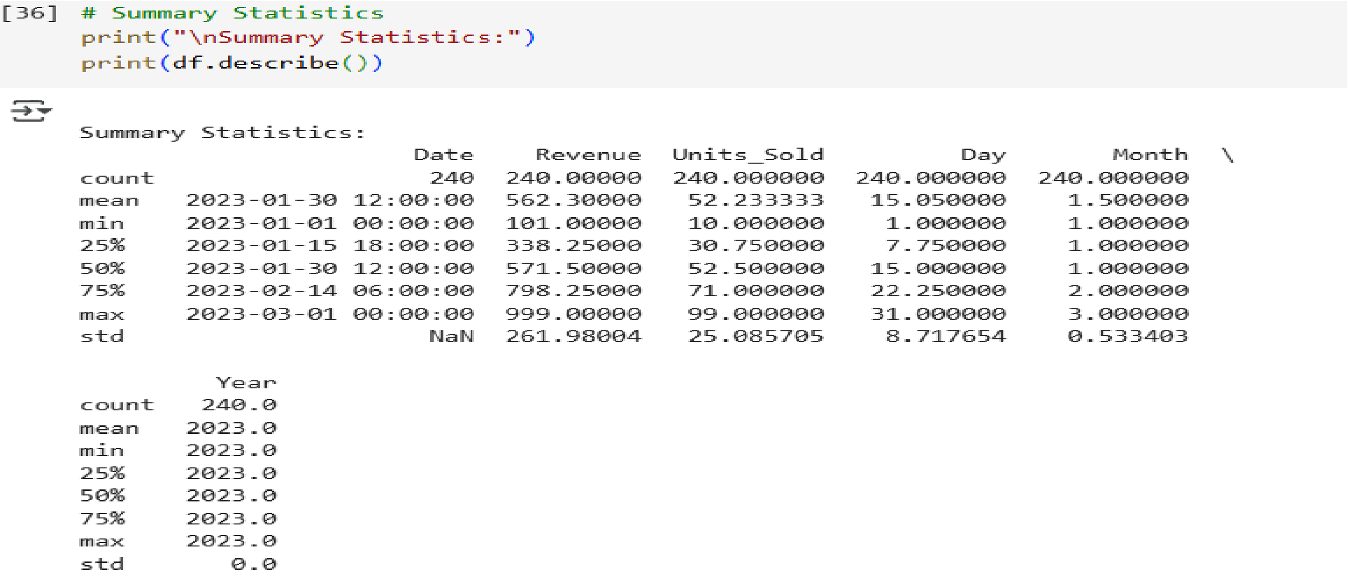
2. Data Visualizations:

- **Revenue Trend Over Time:** A line graph showing revenue fluctuations.
- **Total Revenue Per Product:** A bar chart comparing different products.
- **Seasonal Sales Trends:** A box plot displaying monthly variations.



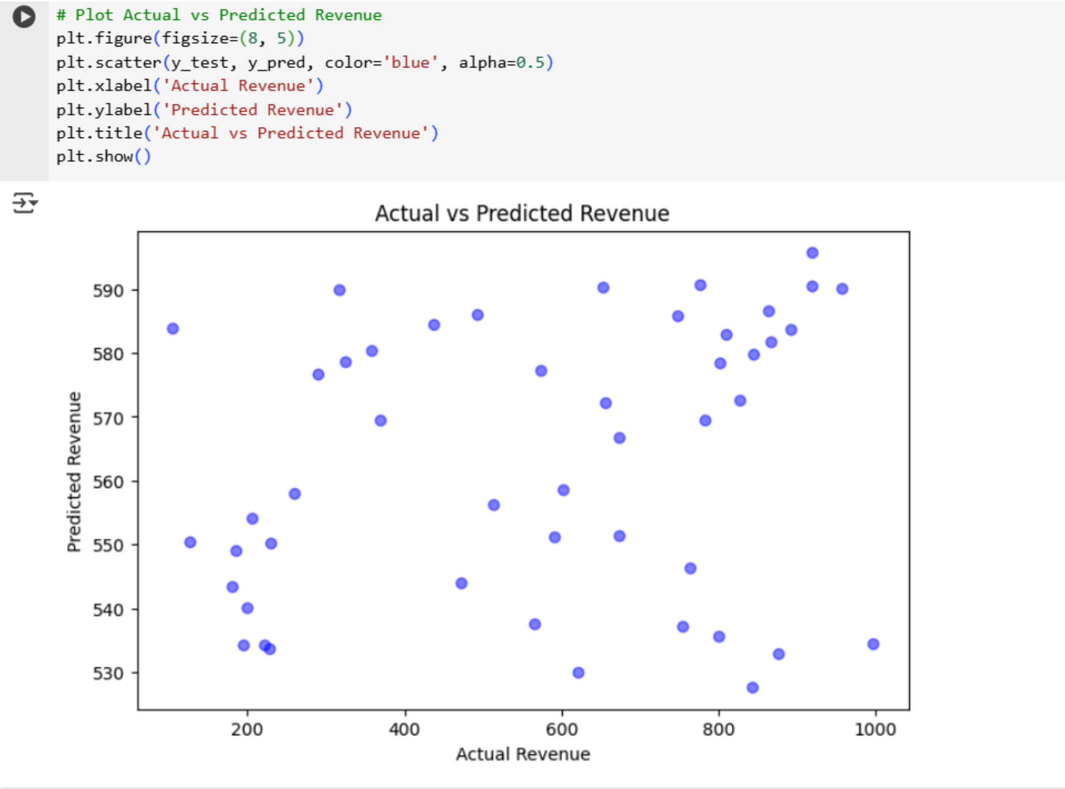
```
# Plot Seasonal Sales Trends
plt.figure(figsize=(10, 5))
sns.boxplot(data=df, x='Month', y='Revenue', palette='viridis')
plt.title('Seasonal Sales Trends')
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.show()
```

```
<ipython-input-35-a49accc4d755>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=df, x='Month', y='Revenue', palette='viridis')
```



```
[36] # Summary Statistics
     print("\nSummary Statistics:")
     print(df.describe())
```

```
Summary Statistics:
                      Date    Revenue  Units_Sold         Day       Month  \
count                  240  240.00000  240.000000  240.000000  240.000000
mean   2023-01-30 12:00:00  562.30000   52.233333   15.050000    1.500000
min    2023-01-01 00:00:00  101.00000   10.000000    1.000000    1.000000
25%    2023-01-15 18:00:00  338.25000   30.750000    7.750000    1.000000
50%    2023-01-30 12:00:00  571.50000   52.500000   15.000000    1.000000
75%    2023-02-14 06:00:00  798.25000   71.000000   22.250000    2.000000
max    2023-03-01 00:00:00  999.00000   99.000000   31.000000    3.000000
std                    NaN  261.98004   25.085705    8.717654    0.533403

         Year
count   240.0
mean   2023.0
min    2023.0
25%    2023.0
50%    2023.0
75%    2023.0
max    2023.0
std       0.0
```



```
# Plot Actual vs Predicted Revenue
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.xlabel('Actual Revenue')
plt.ylabel('Predicted Revenue')
plt.title('Actual vs Predicted Revenue')
plt.show()
```

References

1. Scikit-learn: https://scikit-learn.org/
2. Pandas Documentation: https://pandas.pydata.org/
3. Seaborn Library: https://seaborn.pydata.org/
4. Matplotlib Library: https://matplotlib.org/

This report provides a structured overview of the AI-based sales data visualization project. 🚀 Let me know if you need any modifications!