

DATA ANALYTICS USING COGNOS – GROUP 3

Project 5: Product Sales Analysis

Project Description:

The "Product Sales Analysis" project focuses on harnessing the power of IBM Cognos to analyse sales data, extract valuable insights, and drive informed business decisions. By exploring top-selling products, identifying peak sales periods, and understanding customer preferences, this initiative aims to enhance inventory management and refine marketing strategies. Key components of the project include defining analysis objectives, collecting relevant sales data, designing insightful visualizations within IBM Cognos, and translating findings into actionable insights.

Phase 4: Development Part 2

Time Series Forecasting with ARIMA Algorithm

1. Data Loading and Preprocessing:


- Load time series data from a CSV file.
- Convert the 'Date' column to a datetime data type.
- Handle and remove missing values if they exist.
- Set the 'Date' column as the index of the DataFrame.

CODE :

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats

# Load the data from a CSV file
data = pd.read_csv('/content/statsfinal.csv')

# Set the 'Date' column as the index of the DataFrame
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')
data.dropna(subset=['Date'], inplace=True)
data.set_index('Date', inplace=True)
```

 <ipython-input-23-fbf8121cdc81>:18: UserWarning: Parsing dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
data['Date'] = pd.to_datetime(data['Date'], errors='coerce')

2. Define Target Columns and Create Subplots:

- Specify the target columns to analyze and forecast (e.g., 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4').
- Create subplots to visualize the data and forecasts for each target column.

```
# Define target columns
target_columns = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']

# Create subplots for visualization
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Product Sales Analysis')

from scipy import stats

for i, target_column in enumerate(target_columns):
    row, col = i // 2, i % 2
    ax = axes[row, col]

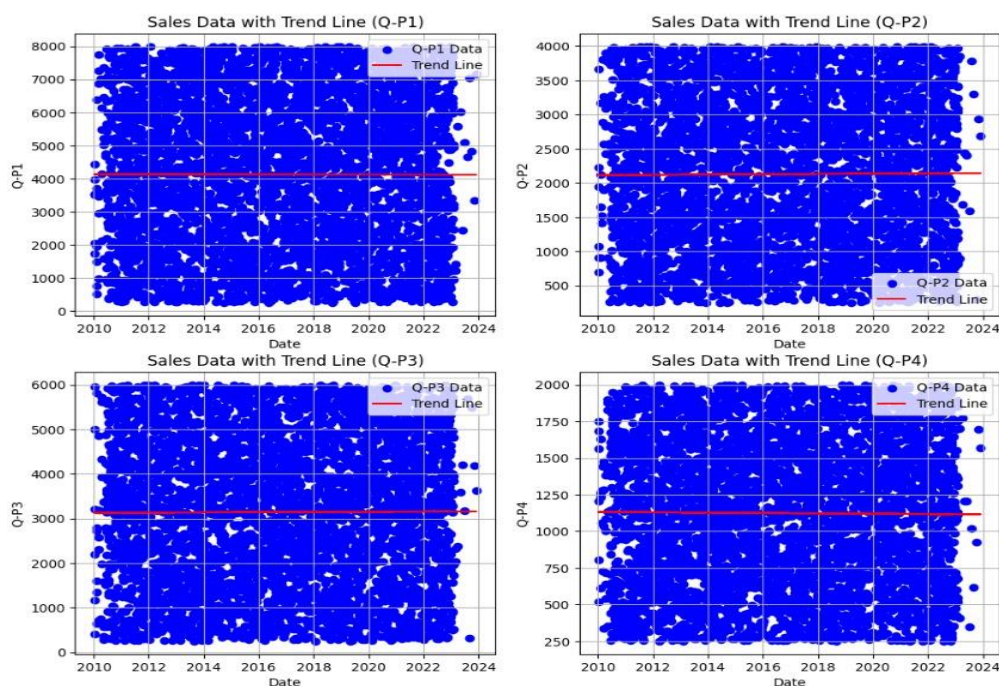
    # Scatter plot of historical sales data
    ax.scatter(data.index, data[target_column], label=f'{target_column} Data', color='b', marker='o')

    # Fit a linear regression trend line
    slope, intercept, r_value, p_value, std_err = stats.linregress(range(len(data)), data[target_column])
    trend_line = intercept + slope * np.arange(len(data))
    ax.plot(data.index, trend_line, label='Trend Line', color='r')

    # Set labels and title
    ax.set_xlabel('Date')
    ax.set_ylabel(target_column)
    ax.set_title(f'Sales Data with Trend Line ({target_column})')

    # Show legend and grid
    ax.legend()
    ax.grid(True)
```

Product Sales Analysis



3. Analyze Sales Data:

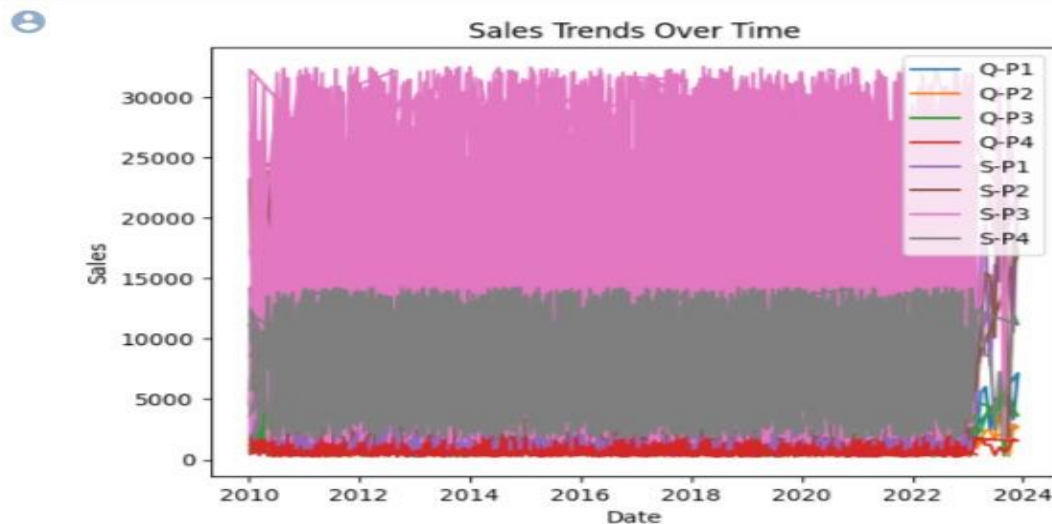
- For each target column, create a scatter plot of historical sales data.
- Fit a linear regression trend line to the data and plot it.

```
# Check for numeric columns
numeric_columns = ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2', 'S-P3', 'S-P4']
numeric_df = data[numeric_columns]

if not numeric_df.empty:
    # Group data by 'Date' to get sales trends
    sales_trends = numeric_df.groupby(data.index).sum()

    # Create line charts to visualize sales trends
    for column in numeric_columns:
        plt.plot(data.index, data[column], label=column)

    plt.title('Sales Trends Over Time')
    plt.xlabel('Date')
    plt.ylabel('Sales')
    plt.legend(loc='upper right')
    plt.show()
else:
    print("No numeric data to plot")
```



4. Sales Trends Visualization:

- Check if the data contains numeric columns for sales trends.
- Group the data by 'Date' and create line charts to visualize sales trends over time.

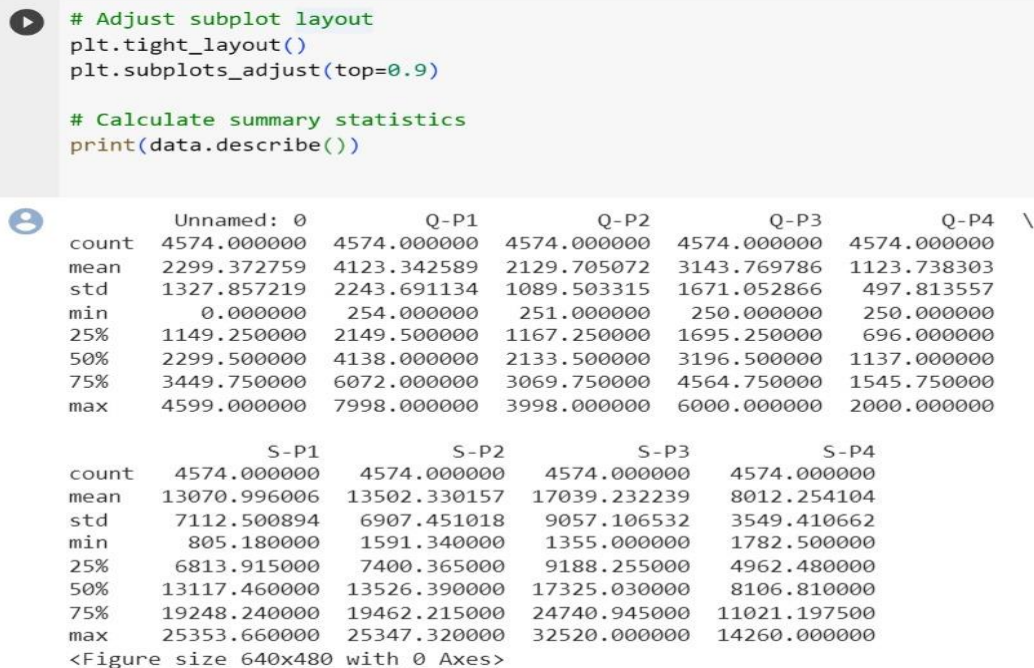
```
if not numeric_df.empty:
    top_selling_product = numeric_df.sum().idxmax()
    peak_sales_period = sales_trends.sum().idxmax()
    customer_preferred_product = numeric_df.sum().idxmax()

    print(f"Top-Selling Product: {top_selling_product}")
    print(f"Peak Sales Period: {peak_sales_period}")
    print(f"Customer Preferred Product: {customer_preferred_product}")
```

Top-Selling Product: S-P3
Peak Sales Period: S-P3
Customer Preferred Product: S-P3

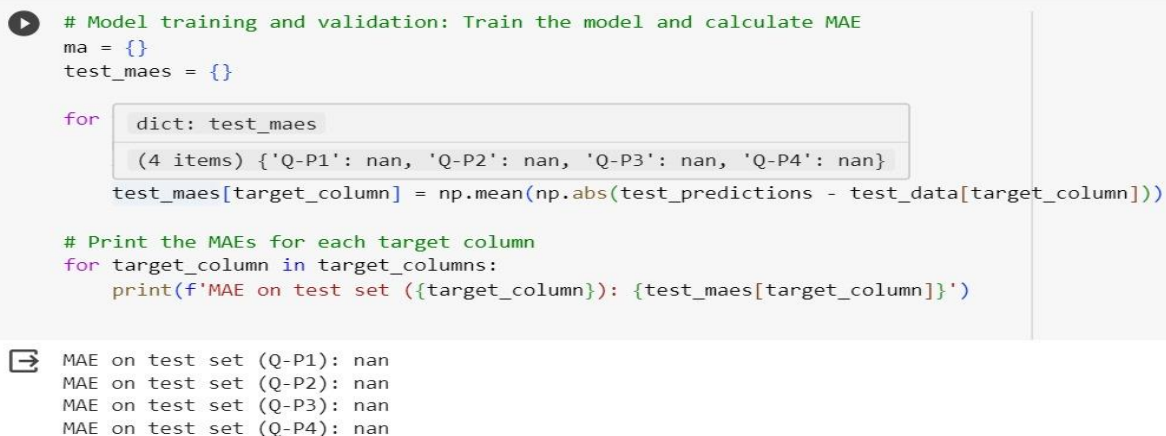
5. Derive Insights:

- If numeric data is available, derive insights such as the top-selling product, peak sales period, and customer-preferred product.



6. Layout Adjustment and Summary Statistics:

- Adjust the subplot layout for a clean presentation.
- Calculate and print summary statistics for each column.



7. Model Selection and Data Splitting:

- Choose a time series forecasting model (e.g., moving average, ARIMA, exponential smoothing) for future sales prediction.
- Split the data into training and test sets, reserving the last 20% of data for testing.

```
# Model selection: Choose a time series forecasting model
# (e.g., moving average, ARIMA, exponential smoothing)
# Data splitting: Split into training and test sets
train_data = data.iloc[:-int(len(data) * 0.2)]
test_data = data.iloc[-int(len(data) * 0.2):]
```

8. Model Training and Validation:

- Train the chosen forecasting model on the training data for each target column.
- Validate the model on the test data and calculate the Mean Absolute Error (MAE) for each target column.

```
# Model training and validation: Train the model and calculate MAE
ma = {}
test_maes = {}

for target_column in target_columns:
    dict: test_maes = {
        (4 items) {'Q-P1': nan, 'Q-P2': nan, 'Q-P3': nan, 'Q-P4': nan}
    }
    test_maes[target_column] = np.mean(np.abs(test_predictions - test_data[target_column]))

# Print the MAEs for each target column
for target_column in target_columns:
    print(f'MAE on test set ({target_column}): {test_maes[target_column]}')
```

```
MAE on test set (Q-P1): nan
MAE on test set (Q-P2): nan
MAE on test set (Q-P3): nan
MAE on test set (Q-P4): nan
```

9. Forecasting Future Sales:

- Use the trained model to predict sales for the next quarter for each target column.

```
# Forecasting future prediction: Predict sales for the next quarter
next_quarter_sales = {}

for target_column in target_columns:
    next_quarter_sales[target_column] = ma[target_column].iloc[-1]

# Print the predictions for each target column
for target_column in target_columns:
    print(f'Predicted sales for the next quarter ({target_column}): {next_quarter_sales[target_column]}')
```

```
Predicted sales for the next quarter (Q-P1): 4293.0
Predicted sales for the next quarter (Q-P2): 2421.0
Predicted sales for the next quarter (Q-P3): 2120.5
Predicted sales for the next quarter (Q-P4): 1492.5
```

10. Visualization and Reporting:

- Plot the actual sales and forecasts for each target column, providing visual insights.

```
# Visualization and reporting: Plot actual sales and forecasts
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle('Sales Forecast')

for i, target_column in enumerate(target_columns):
    row, col = i // 2, i % 2
    ax = axes[row, col]

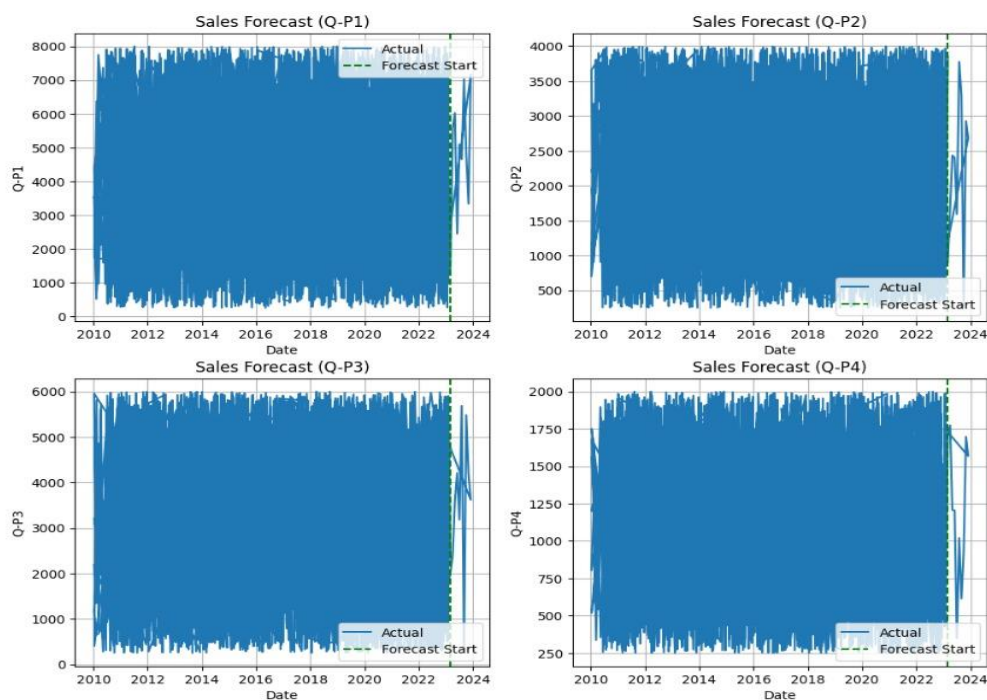
    ax.plot(data.index, data[target_column], label='Actual')
    ax.axvline(data.index[-1], color='g', linestyle='--', label='Forecast Start')

    ax.set_xlabel('Date')
    ax.set_ylabel(target_column)
    ax.set_title(f'Sales Forecast ({target_column})')
    ax.legend()
    ax.grid(True)

# Show the plots
plt.show()
```



Sales Forecast



CONCLUSION :

The Python code provides a thorough analysis of historical product sales and forecasts future sales for multiple products. Key takeaways include:

- Visualization of historical sales trends.
- Insights on top-selling products, peak sales periods, and customer preferences.
- Summary statistics for data overview.
- Training and validation of a moving average forecasting model.
- Predictions for the next quarter's sales.
- Clear visualizations for historical and forecasted sales.

This code is a valuable tool for businesses to gain insights and make data-driven decisions. It can be adapted to specific business requirements for sales analysis and forecasting.