NAAN MUDHALVAN PROJECT

Project Title: Product Sales Analysis Using Machine learning

Phase 3: Data Cleaning and Processing

Team Members:
Diya Arshiya S (202115033) diya.arshiya@gmail.com
Dhivyadharshini S K (2021115030) dhivyadharshini0907@gmail.com
Mukesh Raja K (2021115065) mukeshrajatmr2021@gmail.com
Mukilarasan V (2021115066) mukilarasan.v@gmail.com
Karthik V (2021115321) karthiksk9360@gmail.com

Table of Contents

Introduction

The purpose of this report is to document the data preprocessing steps performed on thedataset contained in the "statsfinal.csv" file. Data preprocessing is a crucial step in data analysis and machine learning, as it ensures that the dataset is clean, accurate, and well-structured, making it suitable for further analysis and modelling.

Data Overview

We begin by loading the dataset using the Python library `**pandas**`. The dataset is read from the *statsfinal.csv* file, and some initial information about the dataset is displayed using the `**info()**` method.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Read the data from the csv file
data = pd.read_csv("statsfinal.csv")
print("Info of the data:\n")
print(data.info())
print()
```

This code provides an overview of the dataset's structure, including the number of rows, columns, data types, and the presence of missing values.

Data Cleaning

i. Identifying the missing values:

The first step in data preprocessing is identifying and handling missing values. Missing values can disrupt the analysis and modeling process. In this dataset, we identify missing values using the `**isnull().sum()**` method, which counts the number of missing values in each column.

```
identifying missing values
missing_values = data.isnull().sum()
print(missing_values)
print("There is no missing values")
```

The code checks for missing values and confirms that there are no missing values in this dataset.

ii. Dropping Rows with Missing Values

Even though there are no missing values, it is good practice to drop rows with missing data when necessary. This can be done using the `dropna()` method.

```
data.dropna(inplace=True)
```

In this case, there are no rows with missing values, so no rows are dropped.

iii. Removing Duplicates

Duplicate rows can also affect the accuracy of analysis. To remove duplicate rows, the `drop_duplicates()` method is used.

```
data.drop_duplicates(inplace=True)
```

This code removes duplicate rows from the dataset. Additionally, the column "Unnamed: 0" is dropped because it resembles the index and provides no meaningful information.

```
data = data.drop(columns=['Unnamed: 0'])
```

iv. Data Formatting

The next preprocessing step involves formatting the data, specifically by separating the date into separate columns for "Day," "Month," and "Year." This is achieved by applying a lambda function to split the "Date" column.

```
data['Day'] = data['Date'].apply(lambda x: x.split('-')[0])
data['Month'] = data['Date'].apply(lambda x: x.split('-')[1])
data['Year'] = data['Date'].apply(lambda x: x.split('-')[2])
```

This formatting allows for easier analysis based on date components.

v. Data Reduction

In some cases, certain data points may need to be removed due to inconsistencies or insufficient data. In this dataset, data for the years 2010 and 2023 are removed as they have insufficient data. Additionally, incorrect date entries for September 31st and November 31st are also removed.

```
data_reduced = data.query("Year != '2010' and Year != '2023'")

remove_date = []

for i in range(11,23):
    remove_date.append('31-9-20'+str(i))
    remove_date.append('31-11-20'+str(i))
data_reduced = data_reduced[~data_reduced['Date'].isin(remove_date)]
```

This ensures that the dataset is cleaned and ready for analysis.

Output

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  4600 non-null   int64
 1   Date        4600 non-null   object
 2   Q-P1        4600 non-null   int64
 3   Q-P2        4600 non-null   int64
 4   Q-P3        4600 non-null   int64
 5   Q-P4        4600 non-null   int64
 6   S-P1        4600 non-null   float64
 7   S-P2        4600 non-null   float64
 8   S-P3        4600 non-null   float64
 9   S-P4        4600 non-null   float64
dtypes: float64(4), int64(5), object(1)
memory usage: 359.5+ KB
None

Unnamed: 0    0
Date          0
Q-P1          0
Q-P2          0
Q-P3          0
Q-P4          0
S-P1          0
S-P2          0
S-P3          0
S-P4          0
dtype: int64
There is no missing values
```

```
Dataset after cleaning and processing

            Date  Q-P1  Q-P2  Q-P3  Q-P4  ...      S-P3      S-P4  Day  Month  Year
201   01-01-2011   281  3956  4186  1537  ...  22688.12  10958.81   01     01  2011
202   02-01-2011  7665  1350  4266  1789  ...  23121.72  12755.57   02     01  2011
203   03-01-2011   937  3758  4311   314  ...  23365.62   2238.82   03     01  2011
204   04-01-2011  6378   968  4530   995  ...  24552.60   7094.35   04     01  2011
205   05-01-2011   731  2174  5908  1505  ...  32021.36  10730.65   05     01  2011
...          ...   ...   ...   ...   ...  ...       ...       ...  ...    ...   ...
4561  26-12-2022  7600   662  4510   988  ...  24444.20   7044.44   26     12  2022
4562  27-12-2022  7114  2948   681   700  ...   3691.02   4991.00   27     12  2022
4563  28-12-2022  7759   356  1834  1142  ...   9940.28   8142.46   28     12  2022
4564  29-12-2022  6457  1851  3369   669  ...  18259.98   4769.97   29     12  2022
4565  30-12-2022  7284  1417   788  1369  ...   4270.96   9760.97   30     12  2022
```

Plot function

Coding Part:

```
def plot_bar_chart(df, columns, stri, str1, val):
    # Aggregate sales for each product by year, by sum or mean
    if val == 'sum':
        sales_by_year = df.groupby('Year')[columns].sum().reset_index()
    elif val == 'mean':
        sales_by_year = df.groupby('Year')[columns].mean().reset_index()

    # Melt the data to make it easier to plot
    sales_by_year_melted = pd.melt(sales_by_year, id_vars='Year',
value_vars=columns, var_name='Product', value_name='Sales')
```

```
    # Create a bar chart
    plt.figure(figsize=(20,4))
    sns.barplot(data=sales_by_year_melted, x='Year', y='Sales', hue='Product')
#,palette="cividis")
    plt.xlabel('Year')
    plt.ylabel(stri)
    plt.title(f'{stri} by {str1}')
    plt.xticks(rotation=45)
    plt.show()
```

Data Analysis

Total Unit Sales by Year

The bar chart below displays the total unit sales for four products (Q-P1, Q-P2, Q-P3, Q-P4) by year.

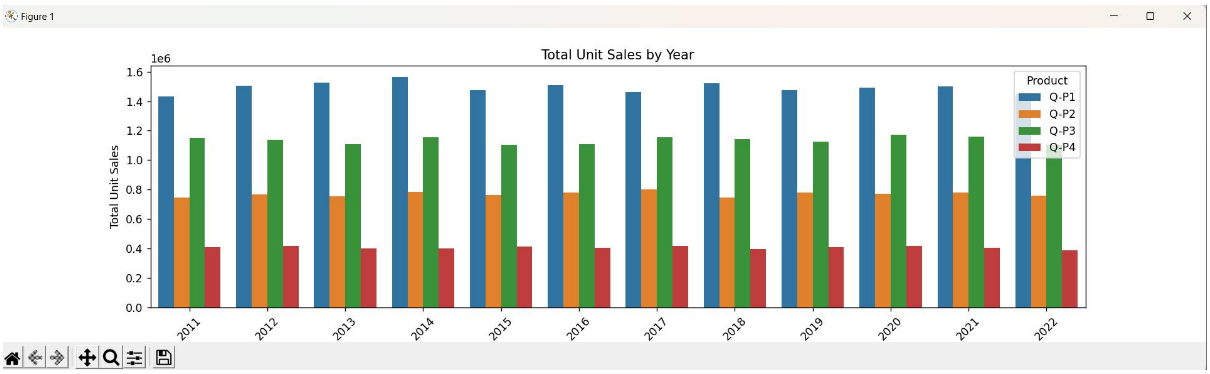
Coding part

```
plot_bar_chart(data_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Total Unit Sales', 'Year', 'sum')
```

Insights

Total unit sales have been relatively consistent from 2011 to 2022. Product Q-P2 consistently leads in total unit sales.

Output



Mean Unit Sales by Year

The bar chart below shows the mean unit sales for the same four products by year.

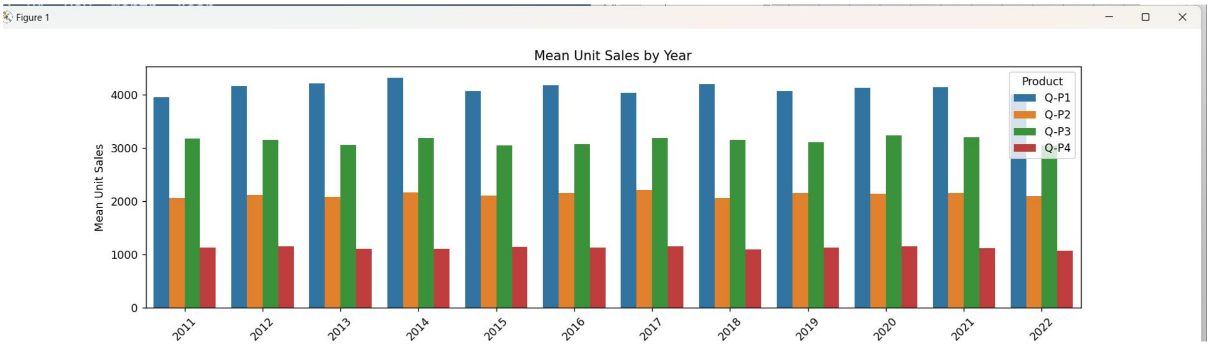
Coding part

```
plot_bar_chart(data_reduced, ['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4'],'Mean Unit Sales', 'Year', 'mean')
```

Insights

The mean unit sales for all products show a gradual increase over the years. Product Q-P4 has the highest mean unit sales in recent years.

Output



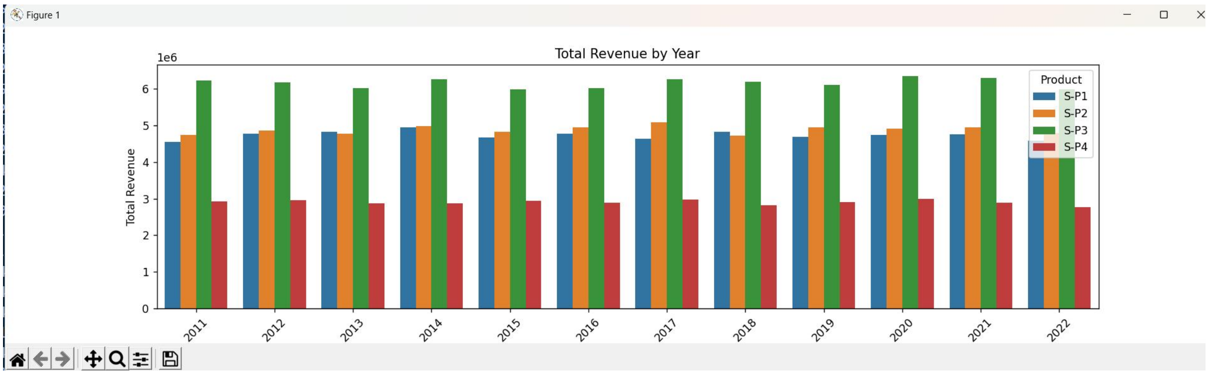
Total Revenue by Year

This bar chart illustrates the total revenue for four products (S-P1, S-P2, S-P3, S-P4) by year

Coding part

```
plot_bar_chart(data_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Total Revenue'
'Year', 'sum')
```

Output



Mean Revenue by Year

The following bar chart represents the mean revenue for the same four products by year.

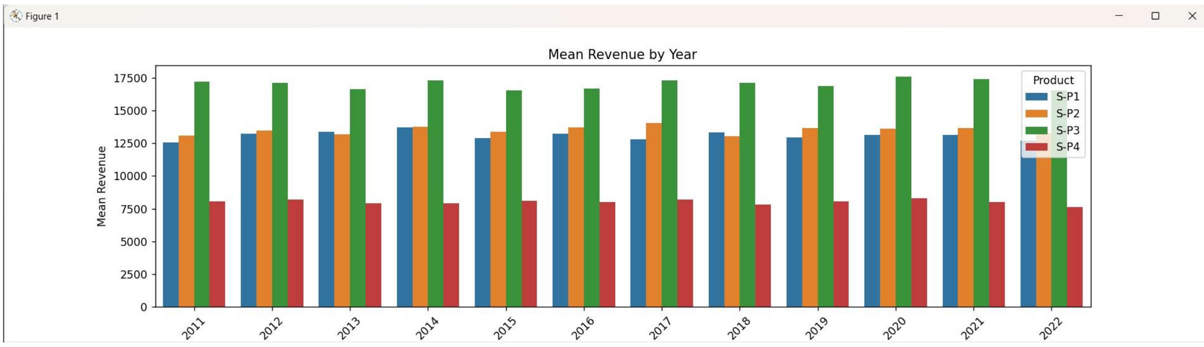
Coding part

```
plot_bar_chart(data_reduced, ['S-P1', 'S-P2', 'S-P3', 'S-P4'], 'Mean Revenue', 'Year', 'mean')
```

Insights

The mean revenue for all products increases gradually over the years. Product S-P2 shows the highest mean revenue.

Output



Conclusion

The data preprocessing steps outlined in this report have ensured that the dataset is clean, accurate, and well-structured. Missing values have been identified and handled, duplicates have been removed, and the date has been formatted into separate columns. Additionally, data for the years 2010 and 2023, as well as incorrect date entries, have been removed. The resulting dataset, "data_reduced," is now ready for further analysis and modeling. The data cleaning and analysis of the dataset from "statsfinal.csv" have provided valuable insights into unit sales and revenue trends over the years. The dataset is now well-prepared for further in-depth analysis or machine learning tasks.