**PREDICTING PRODUCT SALES USING MACHINE LEARNING**

|  |  |
| --- | --- |
| Date | 26.10.2023 |
| Team ID | 8941 |
| Project Name | Product Sales Analysis |

**Phase 3**: Development Part-1

**Topic**: Start building the sales prediction model by loading and pre-processing the dataset.



**Introduction:**

Building a sales prediction model is a data-driven process that involves harnessing the power of machine learning to analyse historical sales data and make informed product sales predictions. This journey begins with the fundamental steps of data loading and preprocessing.

This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the product sales dataset, and perform critical preprocessing steps. Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis.

Necessary step to follow in the phase:

1.Loading dataset

2.Data Preprocessing

**Import libraries:**

Start by importing the necessary libraries:

**Program:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

pd.options.display.max\_columns=50

sns.set(style="darkgrid")

**Load the Dataset:-**

Load your dataset into a Pandas DataFrame. You can typically find product sales datasets in CSV format,but you can adapt this code to other formats as needed.

**Program:**

df=pd.read\_csv(‘D://NanMudhalvanProject/Sales.csv’)

df.head(5)

| **Unnamed: 0** | **Date** | | **Q-P1** | | **Q-P2** | | **Q-P3** | | **Q-P4** | | **S-P1** | **S-P2** | **S-P3** | **S-P4** | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 13-06-2010 | 5422 | | 3725 | | 576 | | 907 | | 17187.74 | | 23616.50 | 3121.92 | | 6466.91 | |  |
| **1** | 14-06-2010 | 7047 | | 779 | | 3578 | | 1574 | | 22338.99 | | 4938.86 | 19392.76 | | 11222.62 | |  |
| **2** | 15-06-2010 | 1572 | | 2082 | | 595 | | 1145 | | 4983.24 | | 13199.88 | 3224.90 | | 8163.85 | |  |
| **3** | 16-06-2010 | 5657 | | 2399 | | 3140 | | 1672 | | 17932.69 | | 15209.66 | 17018,80 | | 11921.36 | |  |
| **4** | 17-06-2010 | 3668 | | 3207 | | 2184 | | 708 | | 11627.58 | | 20332.38 | 118337.28 | | 5048.04 | |  |

**Understanding the data:**

# Fethcing rows and columns

df.shape

**Output:**

(4600, 10)

# fetching column names

df.columns

**Output:**

Index(['Unnamed: 0', 'Date', 'Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P1', 'S-P2',

'S-P3', 'S-P4'],dtype='object')

# basic info

df.info()

**Output:**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4600 entries, 0 to 4599

Data columns (total 10 columns):

# 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

# Checking null values

df.isnull().sum()

**Output:**

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

# Checking Dtypes

df.dtypes

**Output:**

Unnamed: 0 int64

Date object

Q-P1 int64

Q-P2 int64

Q-P3 int64

Q-P4 int64

S-P1 float64

S-P2 float64

S-P3 float64

S-P4 float64

dtype: object

df.duplicated().sum()

**Output:**

0

df.describe().T

**Output:**

|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Unnamed: 0** | 4600.0 | 2299.500000 | 1328.049949 | 0.00 | 1149.750 | 2299.500 | 3449.250 | 4599.00 |
| **Q-P1** | 4600.0 | 4121.849130 | 2244.271323 | 254.00 | 2150.500 | 4137.000 | 6072.000 | 7998.00 |
| **Q-P2** | 4600.0 | 2130.281522 | 1089.783705 | 251.00 | 1167.750 | 2134.000 | 3070.250 | 3998.00 |
| **Q-P3** | 4600.0 | 3145.740000 | 1671.832231 | 250.00 | 1695.750 | 3202.500 | 4569.000 | 6000.00 |
| **Q-P4** | 4600.0 | 1123.500000 | 497.385676 | 250.00 | 696.000 | 1136.500 | 1544.000 | 2000.00 |
| **S-P1** | 4600.0 | 13066.261743 | 7114.340094 | 805.18 | 6817.085 | 13114.290 | 19248.240 | 25353.66 |
| **S-P2** | 4600.0 | 13505.984848 | 6909.228687 | 1591.34 | 7403.535 | 13529.560 | 19465.385 | 25347.32 |
| **S-P3** | 4600.0 | 17049.910800 | 9061.330694 | 1355.00 | 9190.965 | 17357.550 | 24763.980 | 32520.00 |
| **S-P4** | 4600.0 | 8010.555000 | 3546.359869 | 1782.50 | 4962.480 | 8103.245 | 11008.720 | 14260.00 |

**Data Preprocessing**

**Data cleansing**

#Data Cleansing

df.sample(2)

**Output:**

| **Unnamed: 0** | **Date** | **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1466** | 24-06-2014 | 7663 | 2307 | 5094 | 261 | 24291.71 | 14626.3 | 27609.48 | 1860.93 |  |
| **1136** | 28-07-2013 | 4383 | 679 | 3011 | 902 | 13894.11 | 4304.86 | 16319.62 | 6431.26 |  |

# Changing dtype

from datetime import datetime as dt

df[df["Date"]=="31-9-2010"]

**Output:**

| **Unnamed: 0** | **Date** | **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **109** | 31-9-2010 | 4986 | 342 | 4978 | 558 | 15805.62 | 2168.28 | 26980.76 | 3978.54 |  |

df['Date'] = pd.to\_datetime(df['Date'], errors='coerce')

df[df['Date'].isnull()]

**Output:**

| **Unnamed: 0** | **Date** | **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **109** | 109 | NaT | 4986 | 342 | 4978 | 558 | 15805.62 | 2168.28 | 26980.76 | 3978.54 |
| **170** | 170 | NaT | 4632 | 3930 | 523 | 1581 | 14683.44 | 24916.20 | 2834.66 | 11272.53 |
| **473** | 473 | NaT | 2242 | 401 | 5926 | 789 | 7107.14 | 2542.34 | 32118.92 | 5625.57 |
| **534** | 534 | NaT | 325 | 3476 | 4588 | 1771 | 1030.25 | 22037.84 | 24866.96 | 12627.23 |
| **836** | 836 | NaT | 1003 | 256 | 1346 | 1449 | 3179.51 | 1623.04 | 7295.32 | 10331.37 |
| **897** | 897 | NaT | 2509 | 2666 | 4146 | 593 | 7953.53 | 16902.44 | 22471.32 | 4228.09 |
| **1200** | 1200 | NaT | 597 | 709 | 5470 | 1994 | 1892.49 | 4495.06 | 29647.40 | 14217.22 |
| **1261** | 1261 | NaT | 7681 | 1235 | 347 | 1087 | 24348.77 | 7829.90 | 1880.74 | 7750.31 |
| **1564** | 1564 | NaT | 5333 | 833 | 3494 | 618 | 16905.61 | 5281.22 | 18937.48 | 4406.34 |
| **1625** | 1625 | NaT | 3870 | 2779 | 3246 | 1290 | 12267.90 | 17618.86 | 17593.32 | 9197.70 |
| **1928** | 1928 | NaT | 3583 | 2111 | 4225 | 1401 | 11358.11 | 13383.74 | 22899.50 | 9989.13 |
| **1989** | 1989 | NaT | 7516 | 3423 | 3116 | 458 | 23825.72 | 21701.82 | 16888.72 | 3265.54 |
| **2291** | 2291 | NaT | 7891 | 741 | 2280 | 1068 | 25014.47 | 4697.94 | 12357.60 | 7614.84 |
| **2352** | 2352 | NaT | 2457 | 3144 | 533 | 1184 | 7788.69 | 19932.96 | 2888.86 | 8441.92 |
| **2655** | 2655 | NaT | 3512 | 2851 | 4072 | 1597 | 11133.04 | 18075.34 | 22070.24 | 11386.61 |
| **2716** | 2716 | NaT | 6094 | 3798 | 5849 | 881 | 19317.98 | 24079.32 | 31701.58 | 6281.53 |
| **3019** | 3019 | NaT | 1727 | 2645 | 5715 | 1295 | 5474.59 | 16769.30 | 30975.30 | 9233.35 |
| **3080** | 3080 | NaT | 7360 | 2974 | 2717 | 1127 | 23331.20 | 18855.16 | 14726.14 | 8035.51 |
| **3383** | 3383 | NaT | 3195 | 2525 | 5918 | 1003 | 10128.15 | 16008.50 | 32075.56 | 7151.39 |
| **3444** | 3444 | NaT | 2660 | 2674 | 2732 | 934 | 8432.20 | 16953.16 | 14807.44 | 6659.42 |
| **3746** | 3746 | NaT | 4713 | 1227 | 4065 | 403 | 14940.21 | 7779.18 | 22032.30 | 2873.39 |
| **3807** | 3807 | NaT | 870 | 3463 | 798 | 851 | 2757.90 | 21955.42 | 4325.16 | 6067.63 |
| **4110** | 4110 | NaT | 3511 | 2609 | 1543 | 853 | 11129.87 | 16541.06 | 8363.06 | 6081.89 |
| **4171** | 4171 | NaT | 506 | 3333 | 3897 | 574 | 1604.02 | 21131.22 | 21121.74 | 4092.62 |
| **4474** | 4474 | NaT | 6964 | 1873 | 5481 | 1336 | 22075.88 | 11874.82 | 29707.02 | 9525.68 |
| **4535** | 4535 | NaT | 4600 | 2006 | 3796 | 1426 | 14582.00 | 12718.04 | 20574.32 | 10167.38 |

## Filling the NaT values with average of time

df["Date"].fillna(df["Date"].mean(),inplace=True)

df['Date'].isnull().sum()

**Output:**

0

df.dtypes

**Output:**

Unnamed: 0 int64

Date datetime64[ns]

Q-P1 int64

Q-P2 int64

Q-P3 int64

Q-P4 int64

S-P1 float64

S-P2 float64

S-P3 float64

S-P4 float64

dtype: object

#fetching month,day of week, weekday

df["month"]=df["Date"].dt.month\_name()

df["day"]=df["Date"].dt.day\_name()

df["dayoftheweek"]=df["Date"].dt.weekday

df["year"]=df["Date"].dt.year

df.sample()

**Output:**

| **Unnamed: 0** | **Date** | **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** | **month** | **day** | **dayoftheweek** | **year** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3015** | 2018-09-27 | 6899 | 3933 | 4515 | 1447 | 21869.83 | 24935.22 | 24471.3 | 10317.11 | September | Thursday | 3.0 | 2018.0 |  |

df.drop(columns=["Unnamed: 0"],inplace=True)

df.sample()

**Output:**

| **Date** | **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** | **month** | **day** | **dayoftheweek** | **year** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **2021-09-18** | 1302 | 1005 | 3356 | 1430 | 4127.34 | 6371.7 | 18189.52 | 10195.9 | September | Saturday | 5.0 | 5.0 | 2021. |

df.corr().T

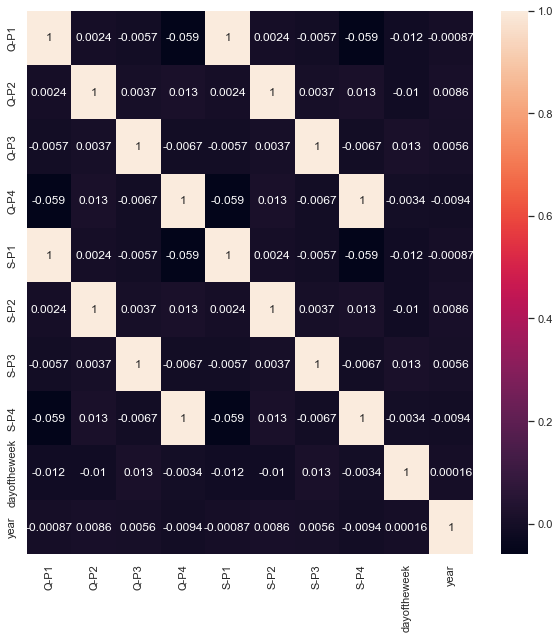
**Output:**

| **Q-P1** | **Q-P2** | **Q-P3** | **Q-P4** | **S-P1** | **S-P2** | **S-P3** | **S-P4** | **dayoftheweek** | **year** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **yearyyeQ-P1** | 1.000000 | 0.002422 | -0.005650 | -0.059365 | 1.000000 | 0.002422 | -0.005650 | -0.059365 | -0.011935 | -0.000917 |
| **Q-P2** | 0.002422 | 1.000000 | 0.003729 | 0.013082 | 0.002422 | 1.000000 | 0.003729 | 0.013082 | -0.010340 | 0.008621 |
| **Q-P3** | -0.005650 | 0.003729 | 1.000000 | -0.006693 | -0.005650 | 0.003729 | 1.000000 | -0.006693 | 0.012007 | 0.005736 |
| **Q-P4** | -0.059365 | 0.013082 | -0.006693 | 1.000000 | -0.059365 | 0.013082 | -0.006693 | 1.000000 | -0.003121 | -0.009489 |
| **S-P1** | 1.000000 | 0.002422 | -0.005650 | -0.059365 | 1.000000 | 0.002422 | -0.005650 | -0.059365 | -0.011935 | -0.000917 |
| **S-P2** | 0.002422 | 1.000000 | 0.003729 | 0.013082 | 0.002422 | 1.000000 | 0.003729 | 0.013082 | -0.010340 | 0.008621 |
| **S-P3** | -0.005650 | 0.003729 | 1.000000 | -0.006693 | -0.005650 | 0.003729 | 1.000000 | -0.006693 | 0.012007 | 0.005736 |
| **S-P4** | -0.059365 | 0.013082 | -0.006693 | 1.000000 | -0.059365 | 0.013082 | -0.006693 | 1.000000 | -0.003121 | -0.009489 |
| **dayoftheweek** | -0.011935 | -0.010340 | 0.012007 | -0.003121 | -0.011935 | -0.010340 | 0.012007 | -0.003121 | 1.000000 | 0.000364 |
| **year** | -0.000917 | 0.008621 | 0.005736 | -0.009489 | -0.000917 | 0.008621 | 0.005736 | -0.009489 | 0.000364 | 1.000000 |

plt.figure(figsize=(10,10))

sns.heatmap(df.corr(),annot=True)

**Output:**



for i in df.columns:

print(i,"---------",df[i].unique())

**Output:**

Date Q-P1 Q-P2 Q-P3 Q-P4 S-P1 S-P2 S-P3

0 2010-06-13 5422 3725 576 907 17187.74 23616.50 3121.92

1 2010-06-14 7047 779 3578 1574 22338.99 4938.86 19392.76

2 2010-06-15 1572 2082 595 1145 4983.24 13199.88 3224.90

3 2010-06-16 5657 2399 3140 1672 17932.69 15209.66 17018.80

4 2010-06-17 3668 3207 2184 708 11627.56 20332.38 11837.28

... ... ... ... ... ... ... ... ...

4595 2023-01-30 2476 3419 525 1359 7848.92 21676.46 2845.50

4596 2023-01-31 7446 841 4825 1311 23603.82 5331.94 26151.50

4597 2023-01-02 6289 3143 3588 474 19936.13 19926.62 19446.96

4598 2023-02-02 3122 1188 5899 517 9896.74 7531.92 31972.58

4599 2023-03-02 1234 3854 2321 406 3911.78 24434.36 12579.82

S-P4 month day dayoftheweek year

0 6466.91 June Sunday 6 2010

1 11222.62 June Monday 0 2010

2 8163.85 June Tuesday 1 2010

3 11921.36 June Wednesday 2 2010

4 5048.04 June Thursday 3 2010

... ... ... ... ... ...

4595 9689.67 January Monday 0 2023

4596 9347.43 January Tuesday 1 2023

4597 3379.62 January Monday 0 2023

4598 3686.21 February Thursday 3 2023

4599 2894.78 March Thursday 3 2023

**Conclusion:**

In the quest to build a sales prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset.We have traversed through essential steps, starting with importing the necessary

libraries to facilitate data manipulation and analysis.

Data preprocessing emerged as a pivotal aspect of this process. It

involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms.

We will discuss for future work such as visualization,analysis,

Accuracy and reliability.