Table Of Contents

ABSTRACT

The Primary objective of this “**SMARTWATCH PRICE PREDICTION USING MACHINE LEARNING AND DATA ANALYSIS**” is to fulfill customer satisfaction and good trust along with challenges of the businesses. It includes data of smartwatch prices and real-time monitoring capabilities.

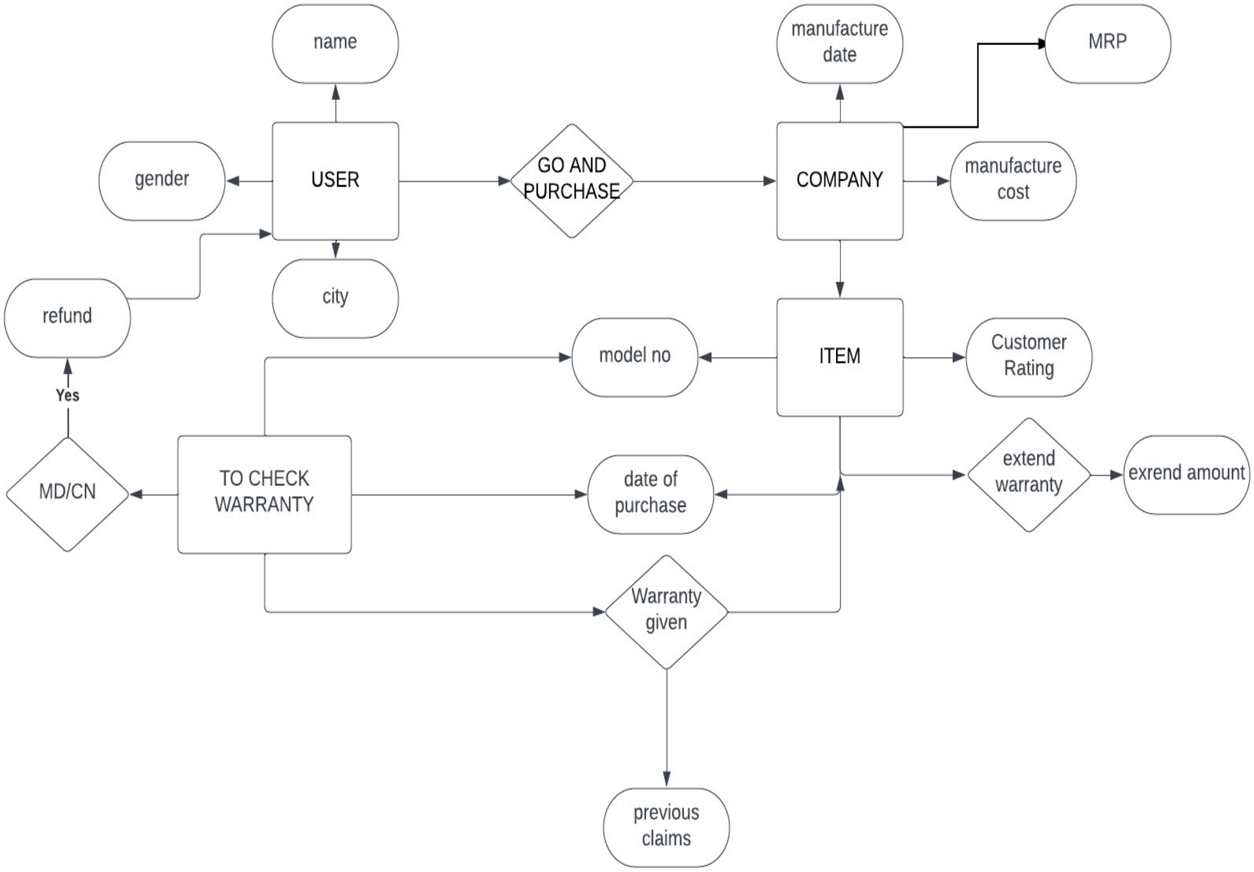
In today's dynamic consumer electronics market, smartwatches have emerged as a popular and rapidly evolving category of wearable technology. Predicting the prices of smartwatches is crucial for both consumers seeking the best deals and retailers optimizing their pricing strategies. This project presents a comprehensive framework for smartwatch price prediction, leveraging advanced machine learning techniques and data-driven insights.

PROBLEM STATEMENT

Smartwatch price prediction is a challenging task due to the rapid evolution of smartwatch technology and the diverse range of models available in the market. However, by using machine learning techniques, it is possible to develop models that can predict smartwatch prices with reasonable accuracy.

In the ever-evolving market of consumer electronics, specifically within the realm of smartwatches, consumers and retailers face a critical challenge: the lack of an efficient and accurate means to predict the prices of smartwatches.

E-R DIAGRAM


name
manufacture date
MRP
gender
USER
GO AND PURCHASE
COMPANY
manufacture cost
refund
city
model no
ITEM
Customer Rating
Yes
MD/CN
TO CHECK WARRANTY
date of purchase
extend warranty
exrend amount
Warranty given
previous claims

REQUIREMENTS

HARDWARE REQUIREMENTS

- Personal Computer / Laptop with minimum RAM (4 GB), ROM (128 GB) and Processor(i3)
- Good latency internet access

SOFTWARE REQUIREMENTS

- Basic Search Engine (Google)
- Google Colaboratory
- MICROSOFT WORD

FUNCTIONAL REQUIREMENTS

- Calculation
- Help in manipulating data and easy process.
- Graphical representation of Datasets

DESCRIPTION

Data Analytics is a process of scrutinizing the data to obtain accurate results. In data analytics the main purpose is extracting the original data from data. In this data analytics we need to perform the major that is data munging.

DATA MUNGING

It is a process of transferring unstructured data into structured format. The goal is to make the data more usable and valuable for analytics or other purposes.

STEPS OF DATA ANALYSIS

1. Defining the Question
2. Collecting the data
3. Cleaning the data
4. Analyzing the data
5. Sharing your results
6. Embracing your failures
7. Summary

<u>CODE</u>

1) DEFINING THE QUESTION

The first step in any data analysis process is to define your objective. In data analytics **jargon**, this is sometimes called the ‘problem statement’. The problem at hand is to develop an efficient and accurate prediction of SmartWatches.

2) COLLECTING THE DATA

Pandas - Helps to create a dataset and it is also a library in python.

Pandas Package – It’s a group of Panal Data’s which are used to analyze the labelled data and relational data.

Series – A series is a method of pandas and labelled data. Series are nothing but columns in Excel sheet.

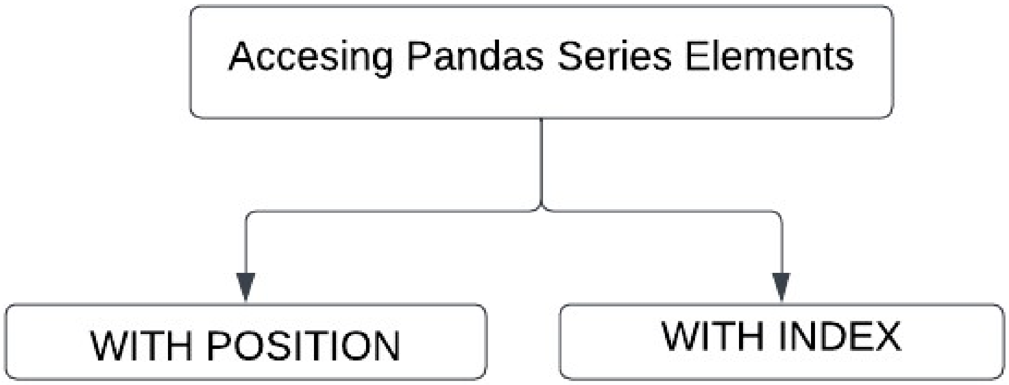
CREATING SERIES-

```
import pandas as pd
a=[10,20,30,'a',50]
b=pd.Series(a) print
(b)
```

```
0    10
1    20
2    30
3    a 4     50 dtype: object
```

```
import pandas as pd
a=[10,20,30,40,50]
b=pd.Series(a)
print(b)
```

```
0    10
1    20
2    30
3    40 4     50 dtype: int64 import pandas as pd a=[10,20,30,40,50]
     b=pd.Series(a) print(b)
```



ACCESSING WITH POSITION-

```
#SLICING import
pandas as pd
a=[10,20,30,40,50]
b=pd.Series(a)
print(b[-2:])
```

```
3    40 4
50 dtype:
```

```
#SLICING import
pandas as pd
a=[10,20,30,40,50]
b=pd.Series(a)
print(b[-4:-2])
```

```
int64
```

```
1    20 2
     30
     dtype:
     int64
```

```
import pandas as pd
a=[10,20,30,40,50]
b=pd.Series(a) print(b[2:])
```

```
2    30
3    40
4    50
dtype: int64
```

ACCESSING WITH INDEX-

```
 import pandas as pd
a=[10,20,'CS',40,50]
 b=pd.Series(a,index=['!','@','#','$','%'])

print(b)  print("-------------------")
 print(b['@'])
```

```
!    10
@    20
#    CS
$    40 %    50
dtype: object -----
--------------
20
```

DATAFRAME

A data frame is a 2D data structure in which we store data in the form of tables. [rows x columns]

We can create a table via Data Frame i.e., known as DATASET.

CREATING A DATASET-

```
#creating empty data set
import pandas as pd
a=pd.DataFrame() print(a)
```

```
Empty DataFrame
Columns: []
Index: []
```

Creating data set using list-

```
 #creating dataframe by using list
import pandas as pd a=[10,20,30,40,50]

 b=pd.DataFrame(a)
print(b)
```

```
    0
0  10
1  20
2  30
3  40
4  50
```

Creating data set using Dict-

```
#creating using DICT
import pandas as pd
x=[{'a':10,'b':20,'c':30}]
y=pd.DataFrame(x) print(y)
```

```
    a   b   c
  0 10  20  30
```

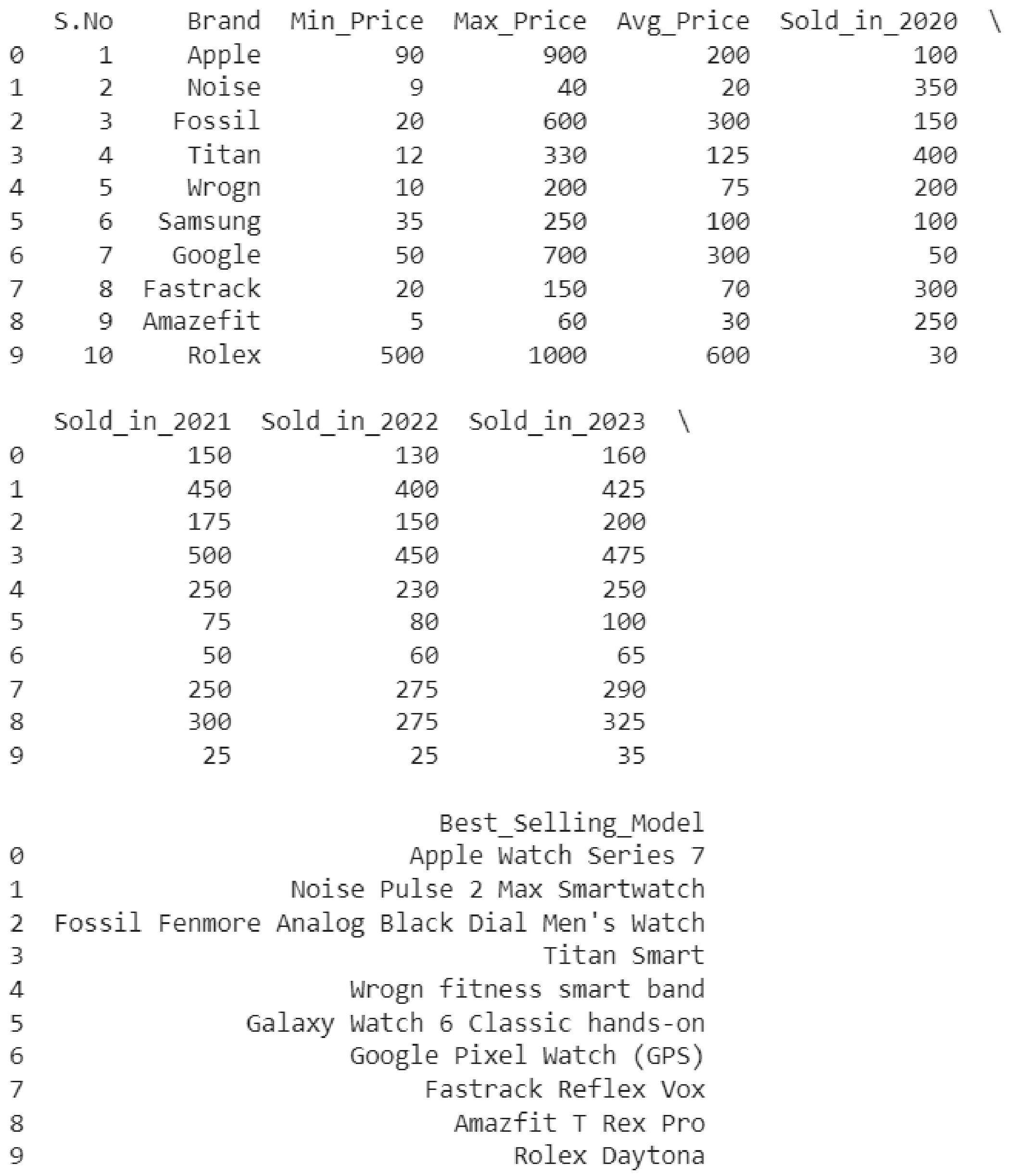
```
#creating using DICT
import pandas as pd
x=[{'a':10,'b':20,'c':30}]
y=pd.DataFrame(x)
print(type(y))
```

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

Creating dataset using Series

```
import pandas as pd
a=[10,20,30,40]
b=pd.Series(a)
c=pd.DataFrame(b) print(c)
```

```
#Import the Required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#Generating the Dataset using Pandas
data = { 'S.No': pd.Series([1,2,3,4,5,6,7,8,9,10]),
        'Brand': pd.Series(['Apple','Noise','Fossil','Titan','Wrogn','Samsung','Google','Fastrack','Amazefit',
         'Rolex']),
        'Min_Price': pd.Series([90,9,20,12,10,35,50,20,5,500]),
        'Max_Price': pd.Series([900,40,600,330,200,250,700,150,60,1000]),
        'Avg_Price': pd.Series([200,20,300,125,75,100,300,70,30,600]),
        'Sold_in_2020': pd.Series([100,350,150,400,200,100,50,300,250,30]),
        'Sold_in_2021': pd.Series([150,450,175,500,250,75,50,250,300,25]),
        'Sold_in_2022': pd.Series([130,400,150,450,230,80,60,275,275,25]),
        'Sold_in_2023': pd.Series([160,425,200,475,250,100,65,290,325,35]),
        'Best_Selling_Model':pd.Series(['Apple Watch Series 7','Noise Pulse 2 Max Smartwatch',
        "Fossil Fenmore Analog Black Dial Men's Watch",'Titan Smart','Wrogn fitness smart band',
        'Galaxy Watch 6 Classic hands-on','Google Pixel Watch (GPS)','Fastrack Reflex Vox','Amazfit T Rex Pro',
        'Rolex Daytona'])}
Data = pd.DataFrame(data)
print(Data)
```



```
   S.No     Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0     1     Apple         90        900        200           100   
1     2     Noise          9         40         20           350   
2     3    Fossil         20        600        300           150   
3     4     Titan         12        330        125           400   
4     5     Wrogn         10        200         75           200   
5     6   Samsung         35        250        100           100   
6     7    Google         50        700        300            50   
7     8  Fastrack         20        150         70           300   
8     9  Amazefit          5         60         30           250   
9    10     Rolex        500       1000        600            30   

   Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0           150           130           160   
1           450           400           425   
2           175           150           200   
3           500           450           475   
4           250           230           250   
5            75            80           100   
6            50            60            65   
7           250           275           290   
8           300           275           325   
9            25            25            35   

                             Best_Selling_Model  
0                         Apple Watch Series 7  
1                  Noise Pulse 2 Max Smartwatch  
2  Fossil Fenmore Analog Black Dial Men's Watch  
3                                  Titan Smart  
4                       Wrogn fitness smart band  
5                  Galaxy Watch 6 Classic hands-on  
6                       Google Pixel Watch (GPS)  
7                            Fastrack Reflex Vox  
8                               Amazfit T Rex Pro  
9                                  Rolex Daytona  
```

OPERATIONS ON DATASET

There are three operations we can perform on a dataset.

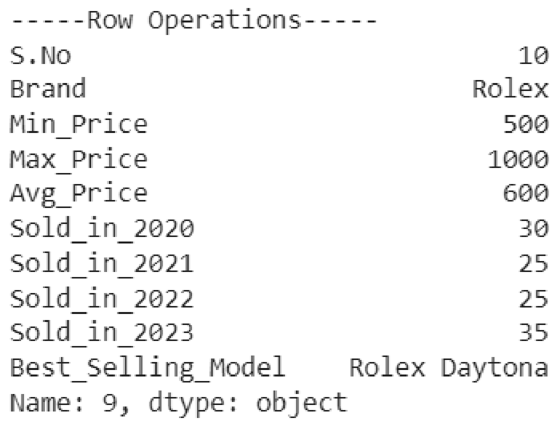
1. Row Operation
2. Column Operation
3. Selection Operation

ROW OPERATIONS-

- row selection
- row addition
- row deletion

ROW SELECTION-

```
#Performing the Row Operations
print('-----Row Operations-----')
#selecting a row and print the
selected row
print(copy.loc[9])
```



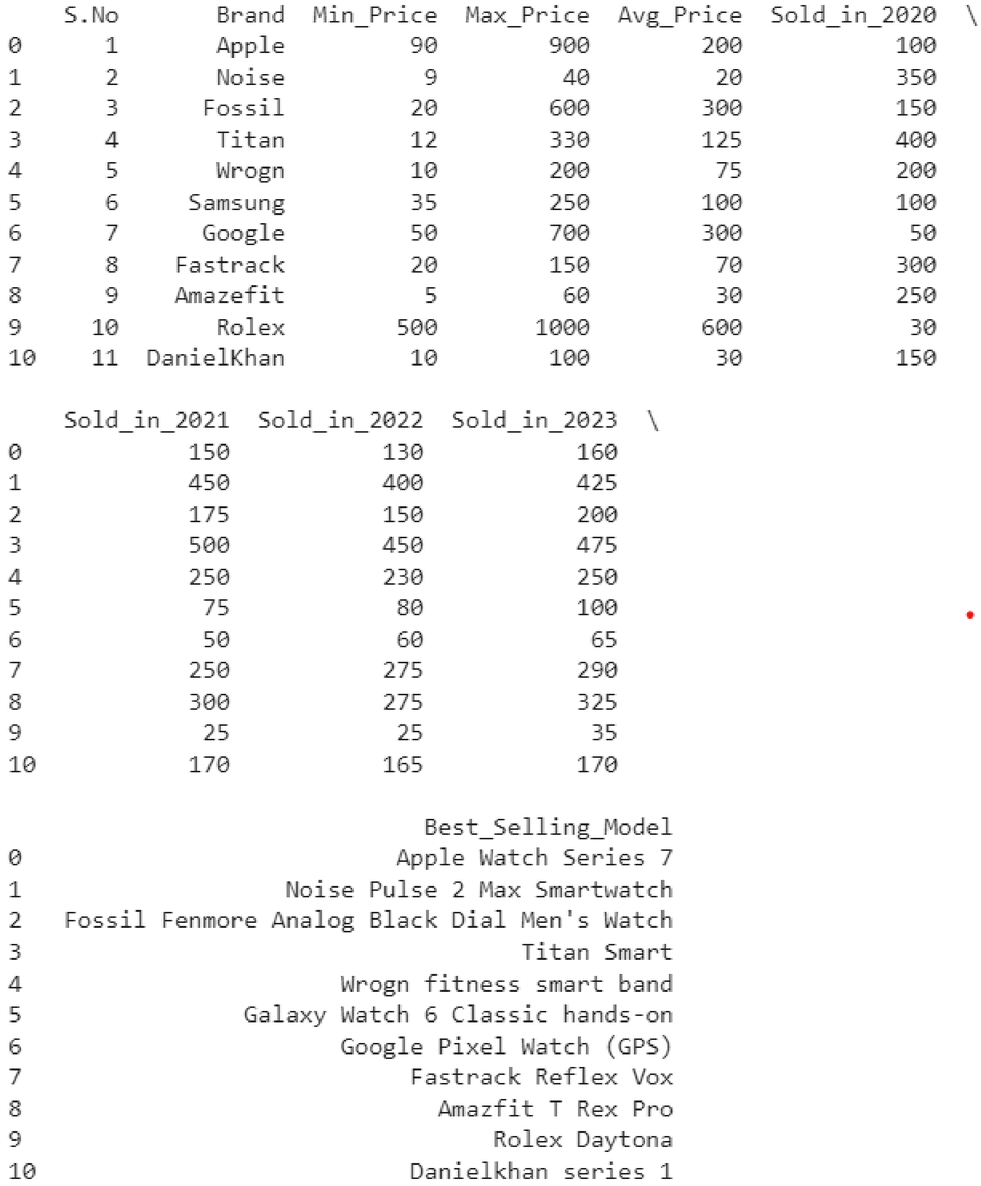
```
-----Row Operations-----
S.No                             10
Brand                         Rolex
Min_Price                       500
Max_Price                      1000
Avg_Price                       600
Sold_in_2020                     30
Sold_in_2021                     25
Sold_in_2022                     25
Sold_in_2023                     35
Best_Selling_Model    Rolex Daytona
Name: 9, dtype: object
```

ROW ADDITION-

We can add the row for the dataset by using ".LOC()" method

But the row data must be the same comparing to the other rows.

```
#Adding a Row and print the DataFrame with including added row
copy.loc[10] = [11,'DanielKhan',10,100,30,150,170,165,170,'Danielkhan
series 1']
print(copy)
```

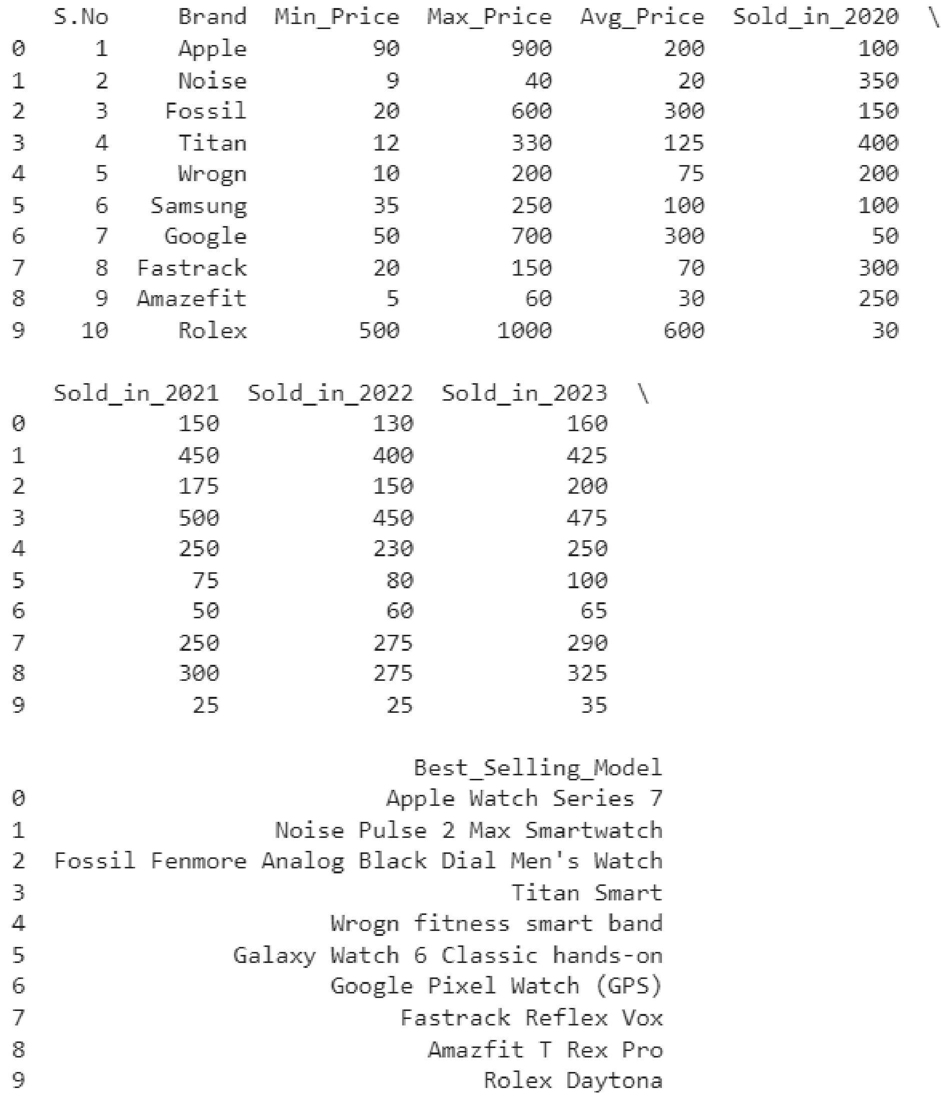


```
    S.No       Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0      1       Apple         90        900        200           100   
1      2       Noise          9         40         20           350   
2      3      Fossil         20        600        300           150   
3      4       Titan         12        330        125           400   
4      5       Wrogn         10        200         75           200   
5      6     Samsung         35        250        100           100   
6      7      Google         50        700        300            50   
7      8    Fastrack         20        150         70           300   
8      9    Amazefit          5         60         30           250   
9     10       Rolex        500       1000        600            30   
10    11  DanielKhan         10        100         30           150   

    Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0            150           130           160   
1            450           400           425   
2            175           150           200   
3            500           450           475   
4            250           230           250   
5             75            80           100   
6             50            60            65   
7            250           275           290   
8            300           275           325   
9             25            25            35   
10           170           165           170   

                             Best_Selling_Model  
0                           Apple Watch Series 7  
1                       Noise Pulse 2 Max Smartwatch  
2   Fossil Fenmore Analog Black Dial Men's Watch  
3                                    Titan Smart  
4                          Wrogn fitness smart band  
5                     Galaxy Watch 6 Classic hands-on  
6                          Google Pixel Watch (GPS)  
7                              Fastrack Reflex Vox  
8                                Amazfit T Rex Pro  
9                                   Rolex Daytona  
10                             Danielkhan series 1  
```

ROW DELETION-

By using drop() method we can delete the row.

```
#Deleting a selected row and print the DataFrame
cop = copy.drop(10)
print(cop)
```



```
   S.No     Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0     1     Apple         90        900        200           100   
1     2     Noise          9         40         20           350   
2     3    Fossil         20        600        300           150   
3     4     Titan         12        330        125           400   
4     5     Wrogn         10        200         75           200   
5     6   Samsung         35        250        100           100   
6     7    Google         50        700        300            50   
7     8  Fastrack         20        150         70           300   
8     9  Amazefit          5         60         30           250   
9    10     Rolex        500       1000        600            30   

   Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0           150           130           160   
1           450           400           425   
2           175           150           200   
3           500           450           475   
4           250           230           250   
5            75            80           100   
6            50            60            65   
7           250           275           290   
8           300           275           325   
9            25            25            35   

                           Best_Selling_Model  
0                       Apple Watch Series 7  
1                Noise Pulse 2 Max Smartwatch  
2  Fossil Fenmore Analog Black Dial Men's Watch  
3                                 Titan Smart  
4                     Wrogn fitness smart band  
5              Galaxy Watch 6 Classic hands-on  
6                    Google Pixel Watch (GPS)  
7                          Fastrack Reflex Vox  
8                            Amazfit T Rex Pro  
9                                Rolex Daytona  
```

COLUMN OPERATIONS

- Column Selection
- Column Addition
- Column deletion

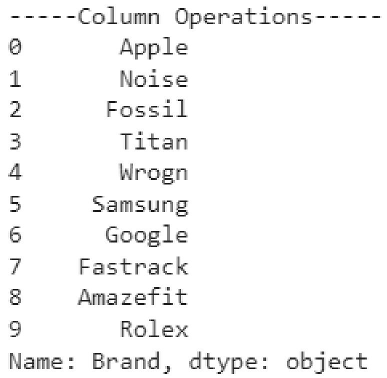
COLUMN SELECTION-

We can select the column by using the column name and the data frame.

DataFrameObj.[‘column name’]

```
#Performing Column Operations
print('-----Column Operations-----')
```

```
#Select a Column in a DataFrame and print
the Selected Column
print(cop['Brand'])
```



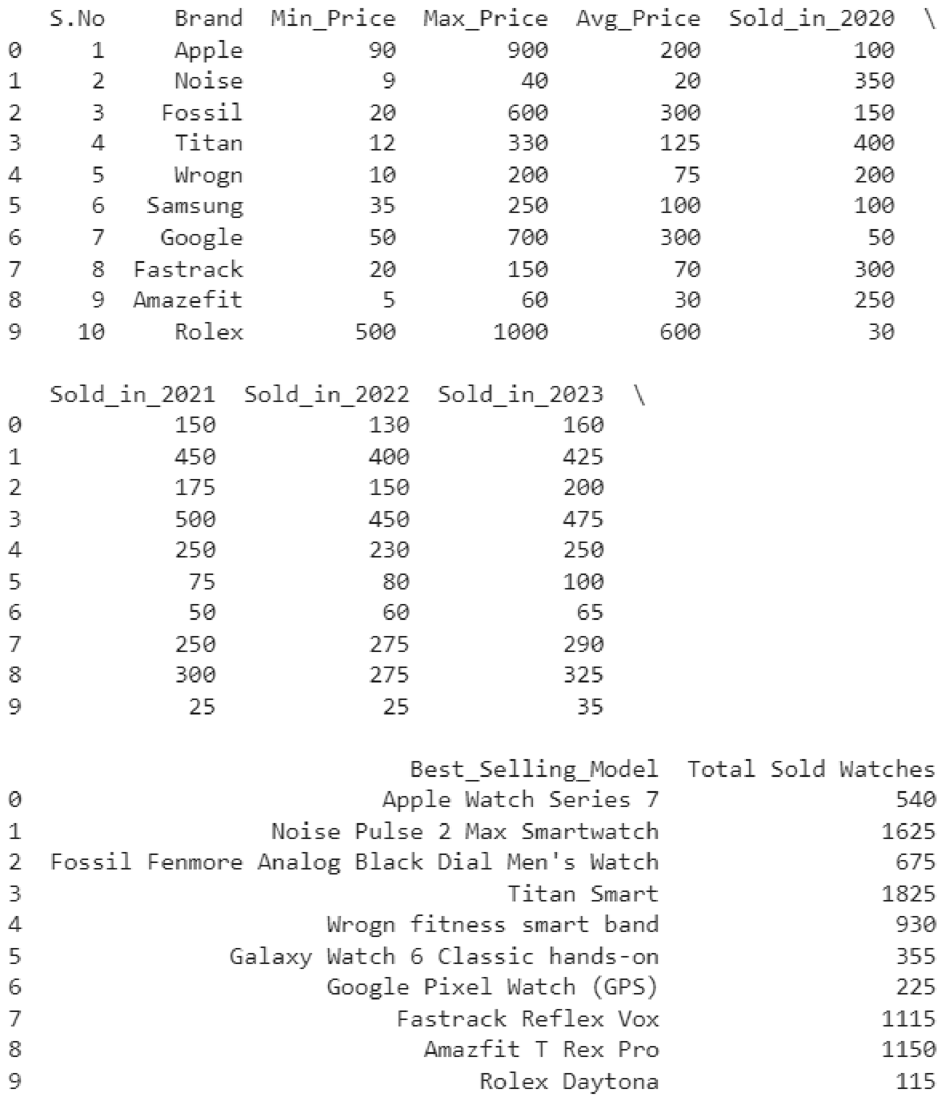
```
-----Column Operations-----
0       Apple
1       Noise
2      Fossil
3       Titan
4       Wrogn
5     Samsung
6      Google
7    Fastrack
8    Amazefit
9       Rolex
Name: Brand, dtype: object
```

COLUMN ADDITION

We can add columns by dataframe[‘ ‘]

I want to add a column Total Sold Watches it can be done by adding 2 columns i.e., sold_in_2022 and sold_in_2023

```
#Adding a Column and print the DataFrame with newly added Column also
cop['Total Sold Watches'] =
cop['Sold_in_2020']+cop['Sold_in_2021']+cop['Sold_in_2022']+cop['Sold_in_2023']
print(cop)
```



```
   S.No     Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0     1     Apple         90        900        200           100   
1     2     Noise          9         40         20           350   
2     3    Fossil         20        600        300           150   
3     4     Titan         12        330        125           400   
4     5     Wrogn         10        200         75           200   
5     6   Samsung         35        250        100           100   
6     7    Google         50        700        300            50   
7     8  Fastrack         20        150         70           300   
8     9  Amazefit          5         60         30           250   
9    10     Rolex        500       1000        600            30   

   Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0           150           130           160   
1           450           400           425   
2           175           150           200   
3           500           450           475   
4           250           230           250   
5            75            80           100   
6            50            60            65   
7           250           275           290   
8           300           275           325   
9            25            25            35   

                          Best_Selling_Model  Total Sold Watches  
0                        Apple Watch Series 7                 540  
1                Noise Pulse 2 Max Smartwatch                1625  
2  Fossil Fenmore Analog Black Dial Men's Watch                 675  
3                                 Titan Smart                1825  
4                       Wrogn fitness smart band                 930  
5                  Galaxy Watch 6 Classic hands-on                 355  
6                       Google Pixel Watch (GPS)                 225  
7                          Fastrack Reflex Vox                1115  
8                            Amazfit T Rex Pro                1150  
9                              Rolex Daytona                 115  
```

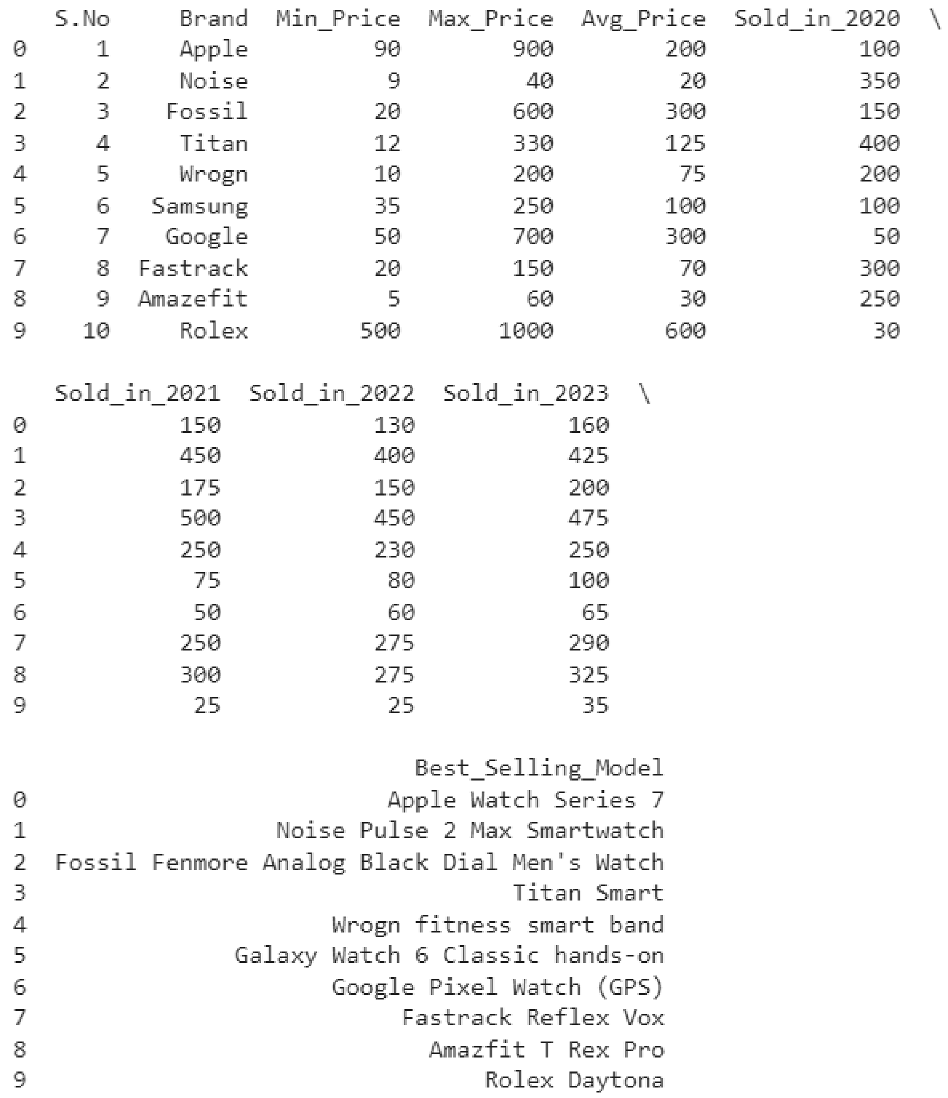
COLUMN DELETION –

To delete an entire column from a Pandas Series in Python, you can use the ’drop’ method or simply select the columns you want to keep.

```
#Deleting a column and print the DataFrame
del cop['Total Sold Watches']
print(cop)
```

The taken dataset ‘Total Sold Watches’ column is removed .

The output of the code is given below.



```
   S.No     Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0     1     Apple         90        900        200           100
1     2     Noise          9         40         20           350
2     3    Fossil         20        600        300           150
3     4     Titan         12        330        125           400
4     5     Wrogn         10        200         75           200
5     6   Samsung         35        250        100           100
6     7    Google         50        700        300            50
7     8  Fastrack         20        150         70           300
8     9  Amazefit          5         60         30           250
9    10     Rolex        500       1000        600            30

   Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0           150           130           160
1           450           400           425
2           175           150           200
3           500           450           475
4           250           230           250
5            75            80           100
6            50            60            65
7           250           275           290
8           300           275           325
9            25            25            35

                             Best_Selling_Model
0                          Apple Watch Series 7
1                  Noise Pulse 2 Max Smartwatch
2  Fossil Fenmore Analog Black Dial Men's Watch
3                                   Titan Smart
4                        Wrogn fitness smart band
5                   Galaxy Watch 6 Classic hands-on
6                        Google Pixel Watch (GPS)
7                             Fastrack Reflex Vox
8                               Amazfit T Rex Pro
9                                   Rolex Daytona
```

Reshaping the Data : In Reshaping of data in the dataset there is no possibility to reshape based on our requirement.

There is only scope we can reshaping the data with the help of stack.

```
#Using stack()
copy = cop.stack()
print(copy)
```

```
0  S.No                    1
   Brand               Apple
   Min_Price              90
   Max_Price                  900
   Avg_Price                  200
                        ...
9  Sold_in_2020                30
   Sold_in_2021                25
   Sold_in_2022                25
   Sold_in_2023                35
   Best_Selling_Model    Rolex Daytona
Length: 100, dtype: object
```

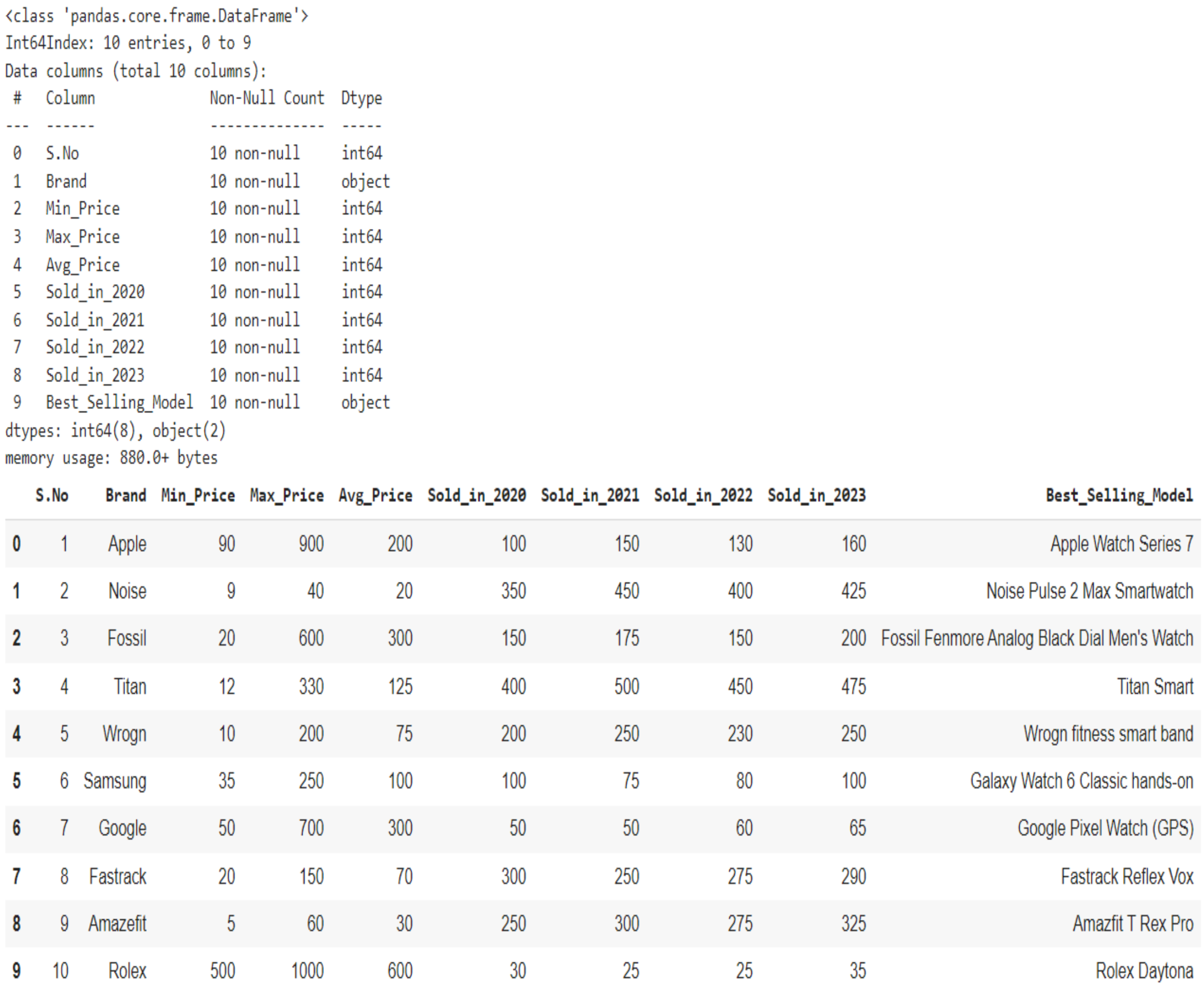
```
#Using Unstack()
cop = copy.unstack()
print(cop)
```

```
S.No     Brand Min_Price Max_Price Avg_Price Sold_in_2020 Sold_in_2021  \
0    1     Apple        90       900       200          100          150
1    2     Noise         9        40        20          350          450
2    3    Fossil        20       600       300          150          175
3    4     Titan        12       330       125          400          500
4    5     Wrogn        10       200        75          200          250
5    6   Samsung        35       250       100          100           75
6    7    Google        50       700       300           50           50
7    8  Fastrack        20       150        70          300          250
8    9  Amazefit         5        60        30          250          300
9   10     Rolex       500      1000       600           30           25

  Sold_in_2022 Sold_in_2023      Best_Selling_Model
0          130          160                   Apple Watch Series 7
1          400          425                   Noise Pulse 2 Max Smartwatch
2          150          200                    Fossil Fenmore Analog Black Dial Men's Watch
3          450          475                    Titan Smart
4          230          250                    Wrogn fitness smart band
5           80          100                    Galaxy Watch 6 Classic hands-on
6           60           65                     Google Pixel Watch (GPS)
7          275          290                      Fastrack Reflex Vox
8          275          325                       Amazfit T Rex Pro
9           25           35

     Rolex Daytona
```

INFO( )

The info method provides a summary of the data including the data types of each column the number of non-null values.

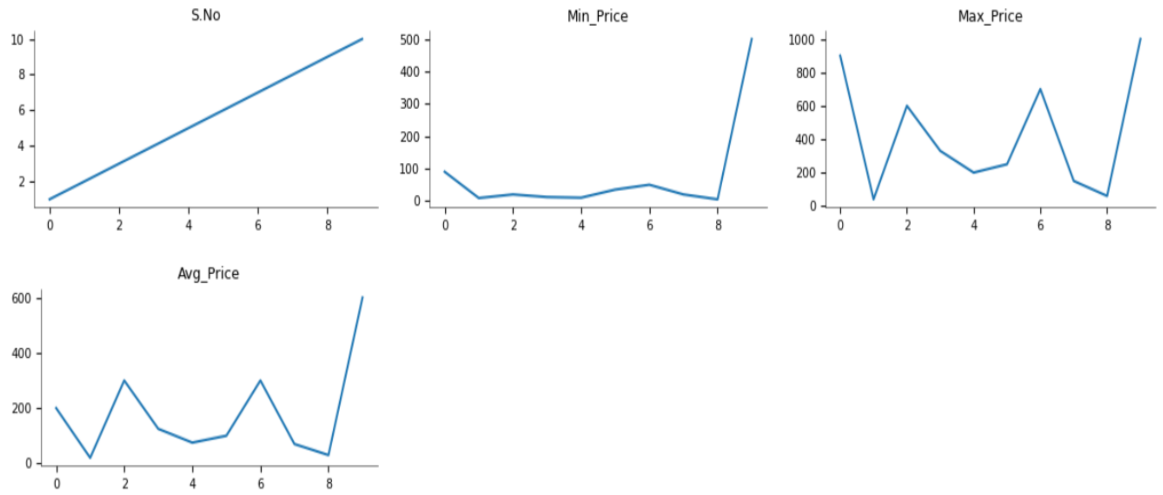
```
#Info of the Data
cop.info()
cop
```



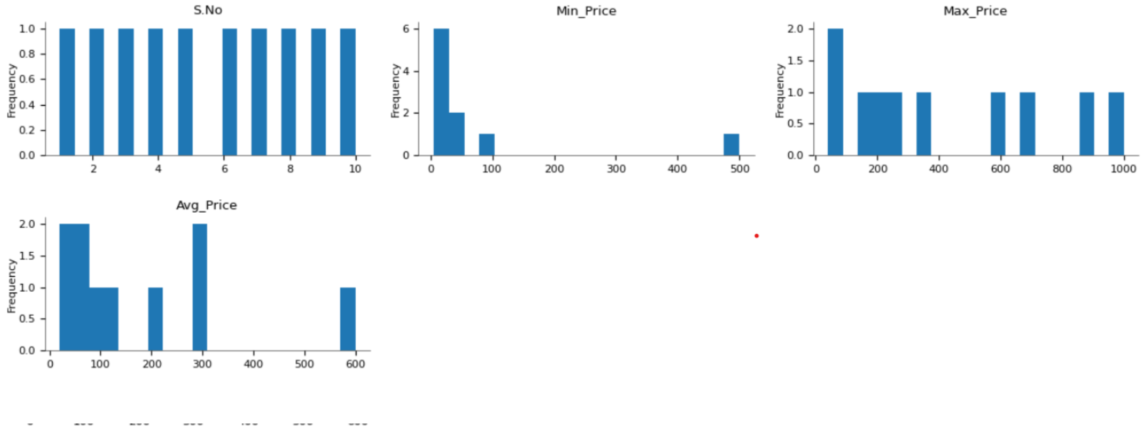
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10 entries, 0 to 9
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   S.No               10 non-null     int64
 1   Brand              10 non-null     object
 2   Min_Price          10 non-null     int64
 3   Max_Price          10 non-null     int64
 4   Avg_Price          10 non-null     int64
 5   Sold_in_2020       10 non-null     int64
 6   Sold_in_2021       10 non-null     int64
 7   Sold_in_2022       10 non-null     int64
 8   Sold_in_2023       10 non-null     int64
 9   Best_Selling_Model 10 non-null     object
dtypes: int64(8), object(2)
memory usage: 880.0+ bytes
```

	S.No	Brand	Min_Price	Max_Price	Avg_Price	Sold_in_2020	Sold_in_2021	Sold_in_2022	Sold_in_2023	Best_Selling_Model
0	1	Apple	90	900	200	100	150	130	160	Apple Watch Series 7
1	2	Noise	9	40	20	350	450	400	425	Noise Pulse 2 Max Smartwatch
2	3	Fossil	20	600	300	150	175	150	200	Fossil Fenmore Analog Black Dial Men's Watch
3	4	Titan	12	330	125	400	500	450	475	Titan Smart
4	5	Wrogn	10	200	75	200	250	230	250	Wrogn fitness smart band
5	6	Samsung	35	250	100	100	75	80	100	Galaxy Watch 6 Classic hands-on
6	7	Google	50	700	300	50	50	60	65	Google Pixel Watch (GPS)
7	8	Fastrack	20	150	70	300	250	275	290	Fastrack Reflex Vox
8	9	Amazefit	5	60	30	250	300	275	325	Amazfit T Rex Pro
9	10	Rolex	500	1000	600	30	25	25	35	Rolex Daytona

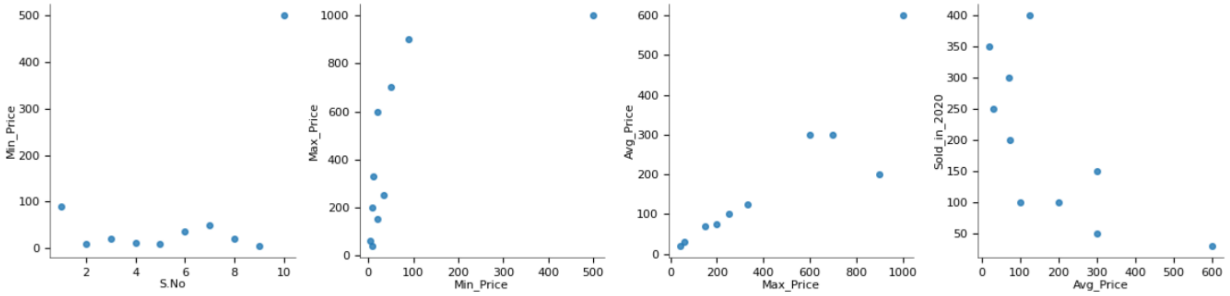
Values

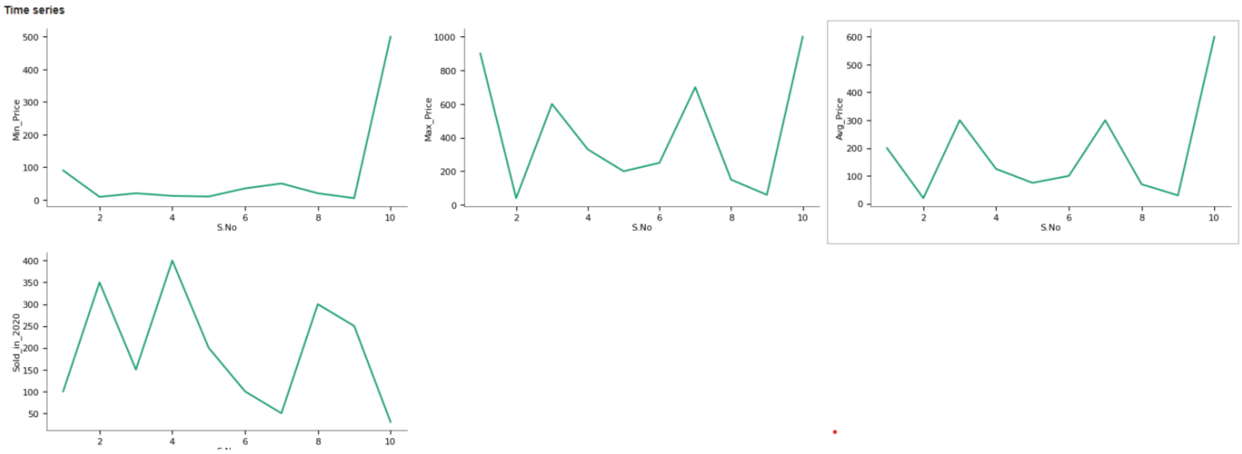


Distributions



2-d distributions





Indexing the Data

```
#Indexing the Data
print(cop.index)
```

Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], dtype='int64')

SIZE OF DATA

```
#Size of the Data
print(Data.size)
```

100

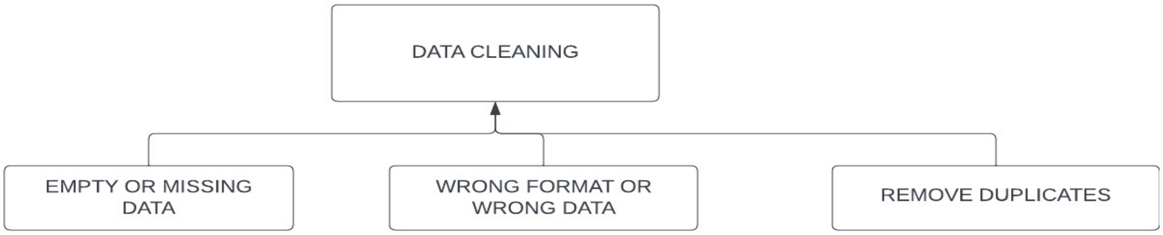
```
#Number of Dimensions in Data
cop.ndim
```

2

3) CLEANING THE DATA

It is the process of remaining or replacing the NAN values.

NAN – Not A Null (or) Not A None



EMPTY OR MISSING DATA

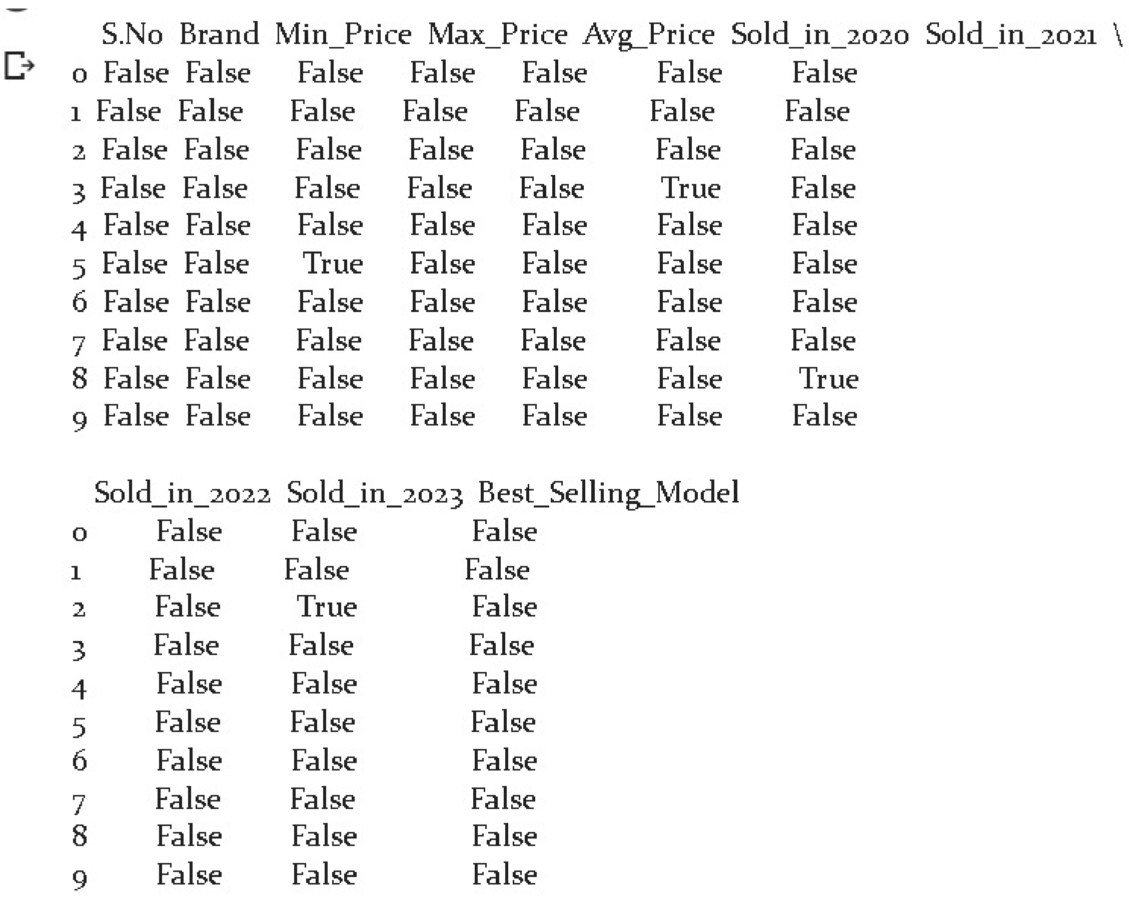
Handling with Null Values :

Here we can perform operations on Dataset by loading numpy’s. In this process we can identify what are the missing values.

We can solve empty cells by using two methods.

- isnull( )
- notnull( )

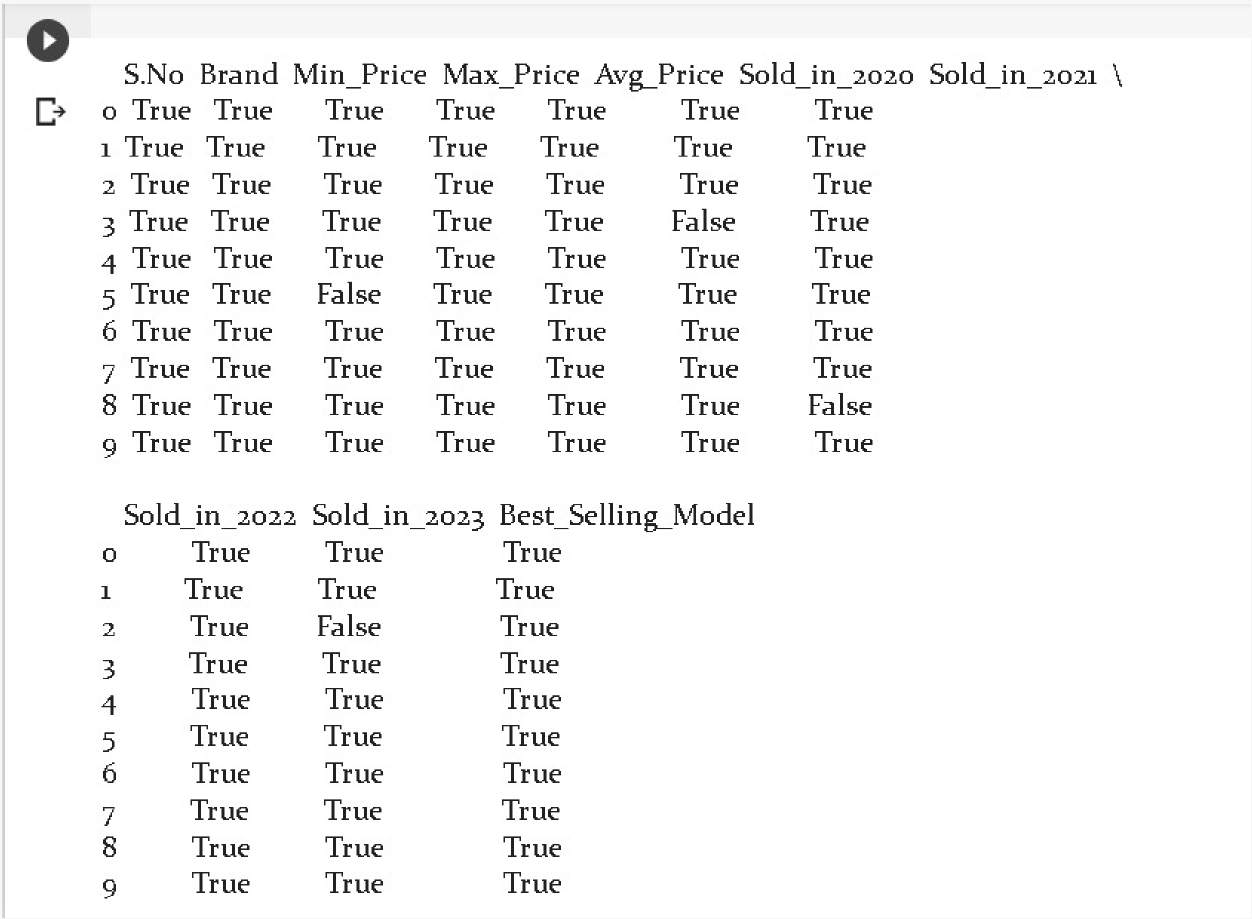
```
#Import the Required libraries
import numpy as np
import pandas as pd
#Generating the Dataset using Pandas
data = { 'S.No': pd.Series([1,2,3,4,5,6,7,8,9,10]),
        'Brand':pd.Series(['Apple','Noise','Fossil','Titan','Wrogn','Samsung','Google','Fa
        strack','Amazefit','Rolex']),
    'Min_Price': pd.Series([90,9,20,12,10,np.nan,50,20,5,500]),
    'Max_Price': pd.Series([900,40,600,330,200,250,700,150,60,1000]),
    'Avg_Price': pd.Series([200,20,300,125,75,100,300,70,30,600]),
    'Sold_in_2020': pd.Series([100,350,150,np.nan,200,100,50,300,250,30]),
    'Sold_in_2021': pd.Series([150,450,175,500,250,75,50,250,np.nan,25]),
    'Sold_in_2022': pd.Series([130,400,150,450,230,80,60,275,275,25]),
    'Sold_in_2023': pd.Series([160,425,np.nan,475,250,100,65,290,325,35]),
    'Best_Selling_Model':pd.Series(['Apple Watch Series 7','Noise Pulse 2 Max
     Smartwatch',"Fossil Fenmore Analog Black Dial Men's Watch",
     'Titan Smart','Wrogn fitness smart band','Galaxy Watch 6 Classic
      handson','Google Pixel Watch (GPS)',
        'Fastrack Reflex Vox','Amazfit T Rex Pro','Rolex Daytona'])}
Data = pd.DataFrame(data)

#Using isnull()
null = Data.isnull()
print(null)
```



```
   S.No  Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  Sold_in_2021 \
0  False  False     False      False      False         False         False
1  False  False     False      False      False         False         False
2  False  False     False      False      False         False         False
3  False  False     False      False      False          True         False
4  False  False     False      False      False         False         False
5  False  False      True      False      False         False         False
6  False  False     False      False      False         False         False
7  False  False     False      False      False         False         False
8  False  False     False      False      False         False          True
9  False  False     False      False      False         False         False

  Sold_in_2022  Sold_in_2023  Best_Selling_Model
0        False         False               False
1        False         False               False
2        False          True               False
3        False         False               False
4        False         False               False
5        False         False               False
6        False         False               False
7        False         False               False
8        False         False               False
9        False         False               False
```

Using Notnull():

```
#Using notnull()
notnull = Data.notnull()
print(notnull)
```

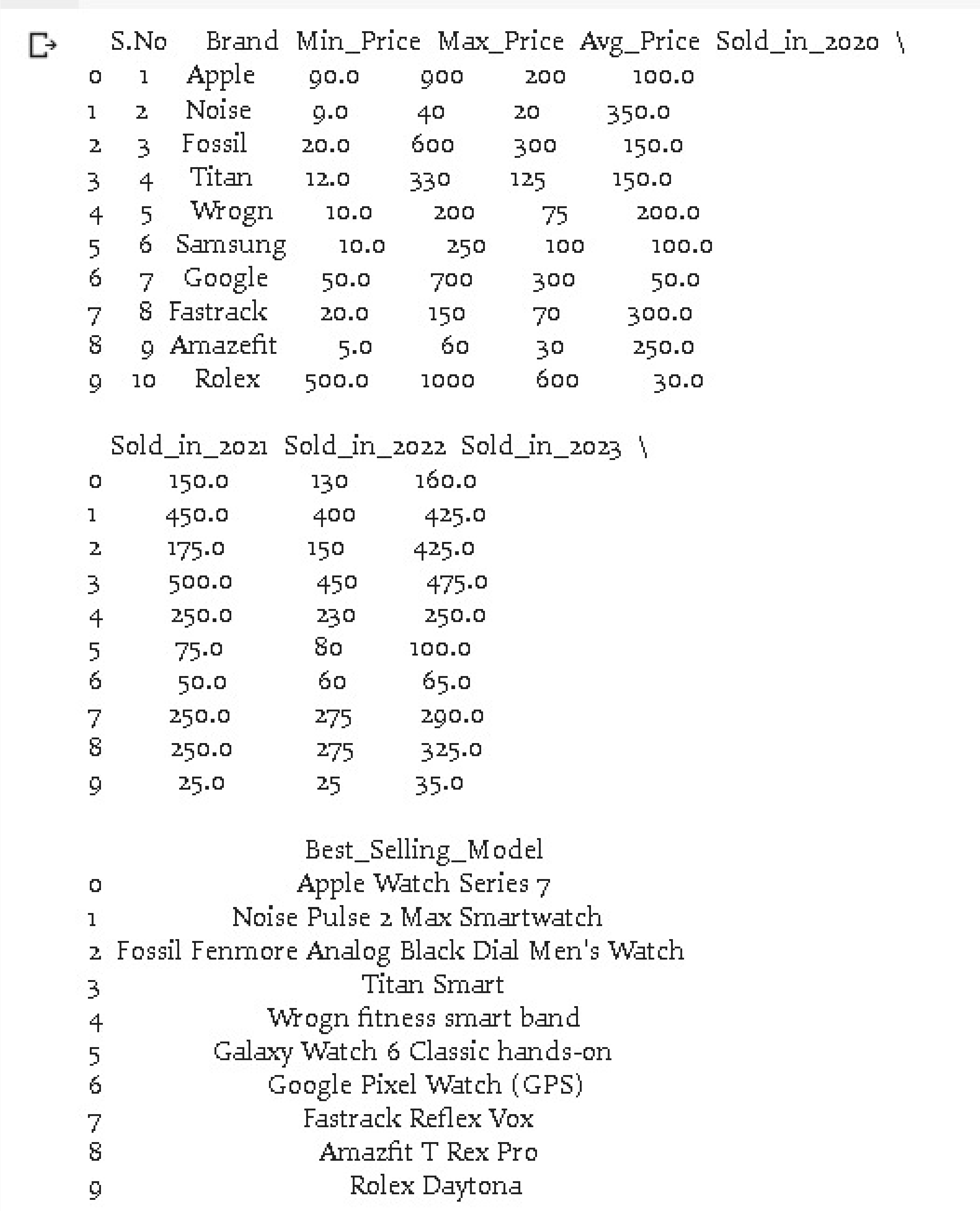


```
  S.No  Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  Sold_in_2021  \
0  True   True      True       True       True          True          True
1  True   True      True       True       True          True          True
2  True   True      True       True       True          True          True
3  True   True      True       True       True         False          True
4  True   True      True       True       True          True          True
5  True   True     False       True       True          True          True
6  True   True      True       True       True          True          True
7  True   True      True       True       True          True          True
8  True   True      True       True       True          True         False
9  True   True      True       True       True          True          True

  Sold_in_2022  Sold_in_2023  Best_Selling_Model
0         True          True                True
1         True          True                True
2         True         False                True
3         True          True                True
4         True          True                True
5         True          True                True
6         True          True                True
7         True          True                True
8         True          True                True
9         True          True                True
```

Fillna():

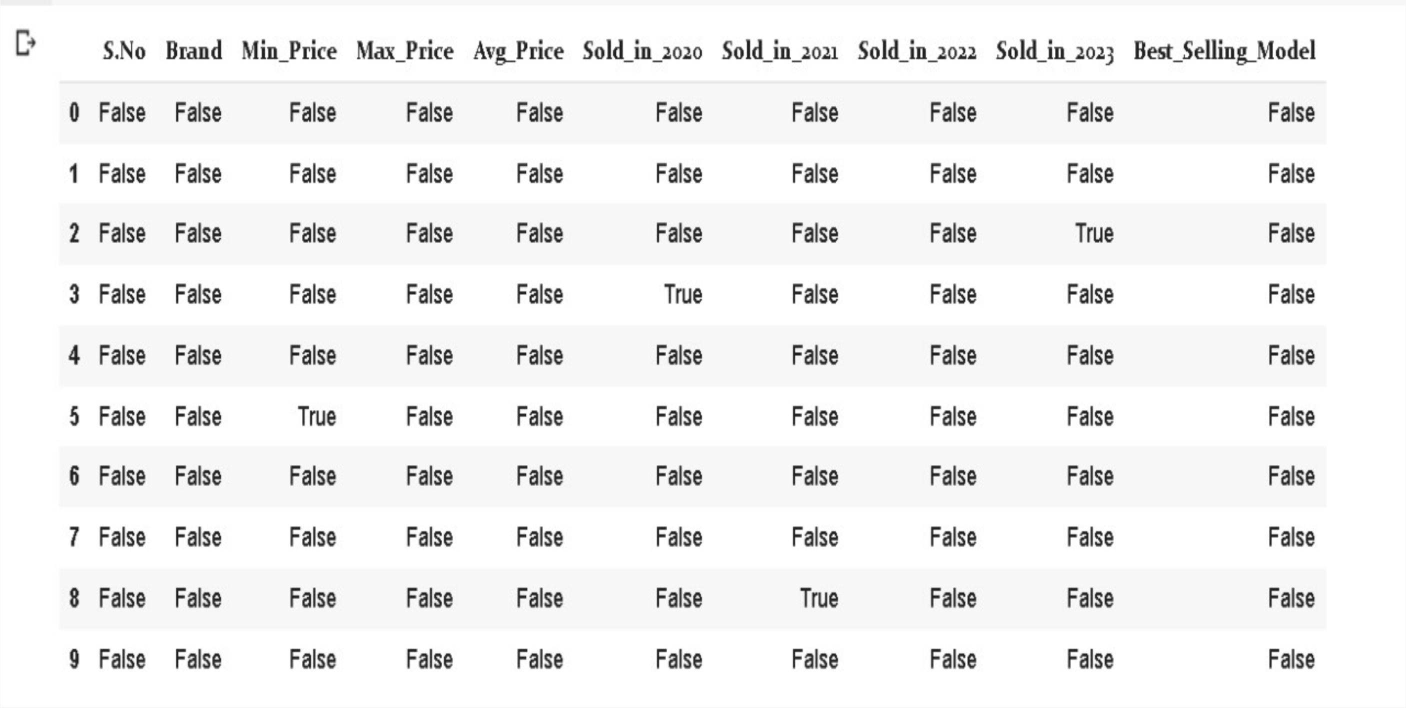
This method is used to fill the missing values with our required data.
We have 2 types in fillna( ) for the parameter method.
1.fillna(method=’pad’)
2.fillna(method=’bfill’)

```
#fillna()using pad parametre
fill = Data.fillna(method='pad')
print(fill)
```

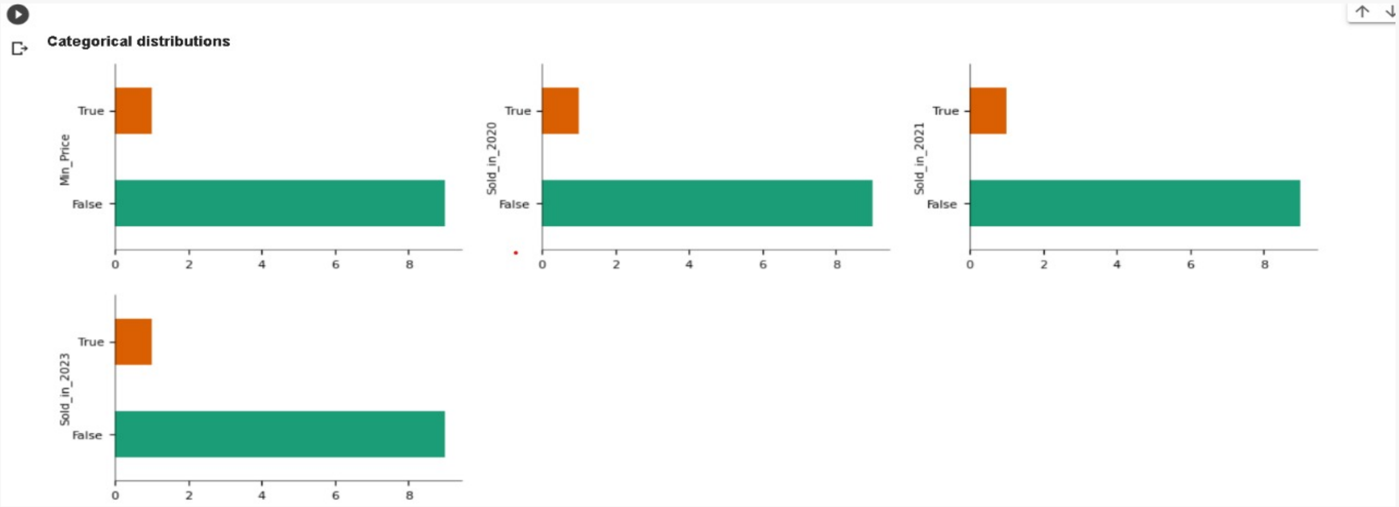


```
   S.No     Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  \
0     1     Apple       90.0        900        200         100.0
1     2     Noise        9.0         40         20         350.0
2     3    Fossil       20.0        600        300         150.0
3     4     Titan       12.0        330        125         150.0
4     5     Wrogn       10.0        200         75         200.0
5     6   Samsung       10.0        250        100         100.0
6     7    Google       50.0        700        300          50.0
7     8  Fastrack       20.0        150         70         300.0
8     9  Amazefit        5.0         60         30         250.0
9    10     Rolex      500.0       1000        600          30.0

   Sold_in_2021  Sold_in_2022  Sold_in_2023  \
0         150.0           130         160.0
1         450.0           400         425.0
2         175.0           150         425.0
3         500.0           450         475.0
4         250.0           230         250.0
5          75.0            80         100.0
6          50.0            60          65.0
7         250.0           275         290.0
8         250.0           275         325.0
9          25.0            25          35.0

                          Best_Selling_Model
0                      Apple Watch Series 7
1               Noise Pulse 2 Max Smartwatch
2  Fossil Fenmore Analog Black Dial Men's Watch
3                               Titan Smart
4                     Wrogn fitness smart band
5             Galaxy Watch 6 Classic hands-on
6                    Google Pixel Watch (GPS)
7                        Fastrack Reflex Vox
8                          Amazfit T Rex Pro
9                             Rolex Daytona
```

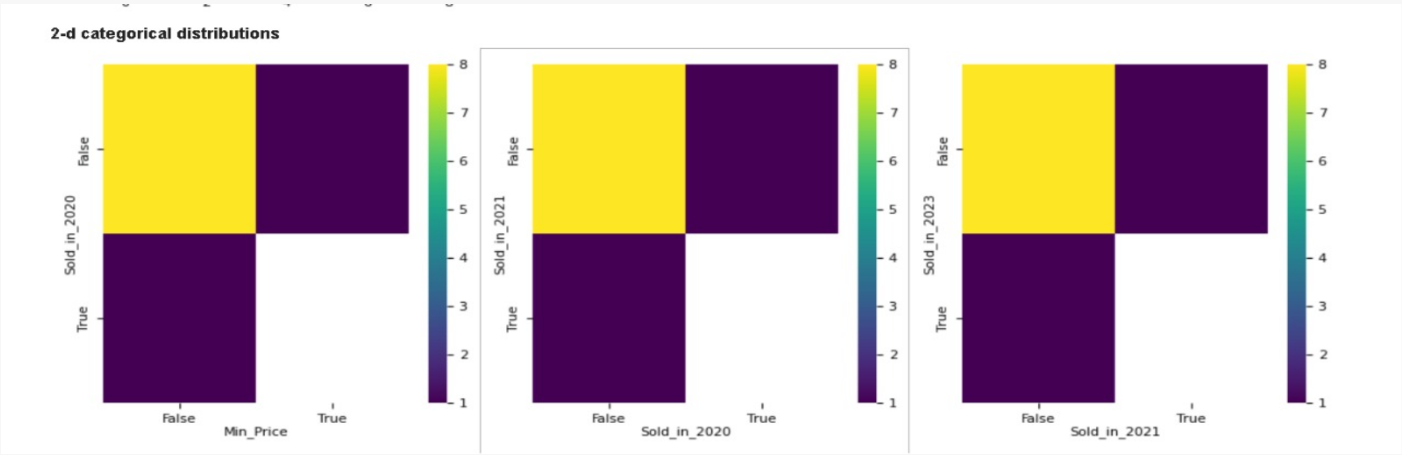
```
#Checking Still any null values present in the Data
check = Data.isnull()
print(check)
```



	S.No	Brand	Min_Price	Max_Price	Avg_Price	Sold_in_2020	Sold_in_2021	Sold_in_2022	Sold_in_2023	Best_Selling_Model
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	True	False
3	False	False	False	False	False	True	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
5	False	False	True	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	True	False	False	False
9	False	False	False	False	False	False	False	False	False	False



Categorical distributions



2-d categorical distributions

Analyzing the Data: In the view of analyzing the data we perform the operations like

1.Statistical Operations
2.Computational Operations

Statistical Operations : In Statistical operations we have perform all operations which are related to mathematics. By performing these operations it gets proved on what dataset we generated.
The following are the Statistical Operations :

- Mean( )
- Mode( )
- Median( )
- Min( )
- Max( )
- Sum( )
- Aggregate( )
- Describe( )

- **Mean( ):** It performs the average operation for particular column in dataset we can perform the mean operation by using “ mean( ) ”.

```
# Mean of Data using mean()
mean = cop['Min_Price'].mean()
print(mean)
```

75.1

```
# Mean of Data using mean()
mean = cop['Min_Price'].mean()
print(mean)
```

423.0

- **Mode( ):** This method is used to return the output most repeated value

```
#Finding the Mode of the Data Using mode()
mode = cop['Avg_Price'].mode()
mode
```

```
0    300
Name: Avg_Price, dtype: object
```

- **Median( ) :** This method is used to return the mid value of the collection.

```
#Median of the Data using median()
median = cop['Max_Price'].median()
median
```

```
290.0
```

- **Min( ) :** This method is used to return the minimum value in Selected column in the Dataset.

```
# Finding the Minimum value using min()
min = cop['Min_Price'].min()
min
```

```
5
```

- **Max( ) :** This method is used to returns the maximum value in selected column of the Dataset.

```
# Finding the Maximum Value using max()
max = cop['Max_Price'].max()
max
```

```
1000
```

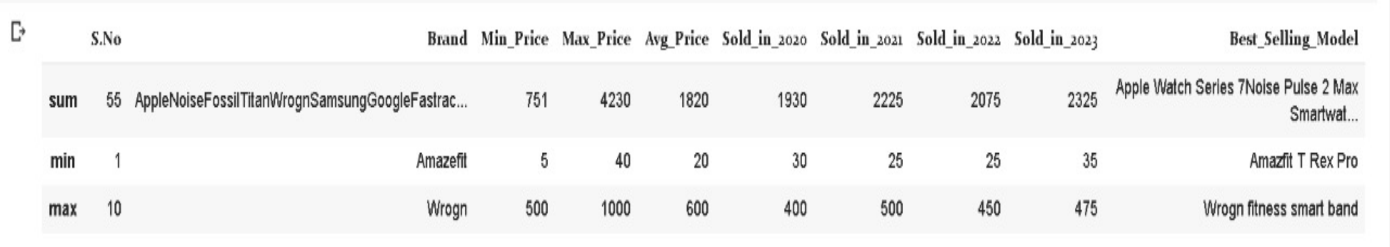
- **Sum( ) :** This method is used to sum all the values in the collection.

```
# Sum all the prices in a column using sum()
sum = cop['Max_Price'].sum()
sum
```

```
4230
```

- **Aggregate( ) :** This method is used perform 2 or more statistical operations at a time.

```
# Aggregating the Data using aggregate()
aggregate = cop.aggregate(['sum','min','max'])
aggregate
```



	S.No	Brand	Min_Price	Max_Price	Avg_Price	Sold_in_2020	Sold_in_2021	Sold_in_2022	Sold_in_2023	Best_Selling_Model
sum	55	AppleNoiseFossilTitanWrognSamsungGoogleFastrac...	751	4230	1820	1930	2225	2075	2325	Apple Watch Series 7Noise Pulse 2 Max Smartwat...
min	1	Amazefit	5	40	20	30	25	25	35	Amazfit T Rex Pro
max	10	Wrogn	500	1000	600	400	500	450	475	Wrogn fitness smart band

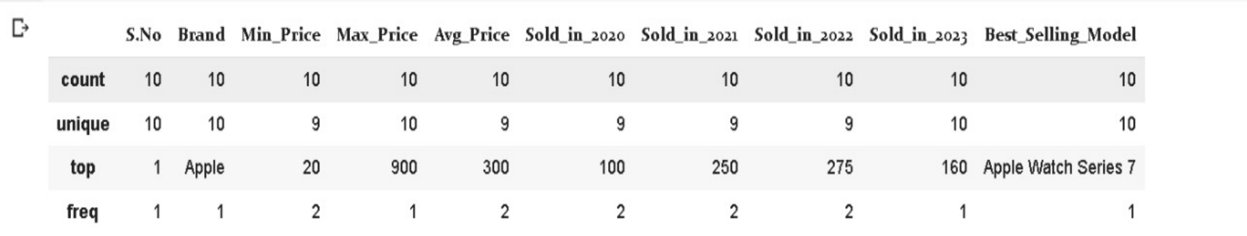
- **COUNT( ) :** This method is used to get the count of values in a column.

```
# Count the Number of Brand of watches using count()
count = cop['Brand'].count()
count
```

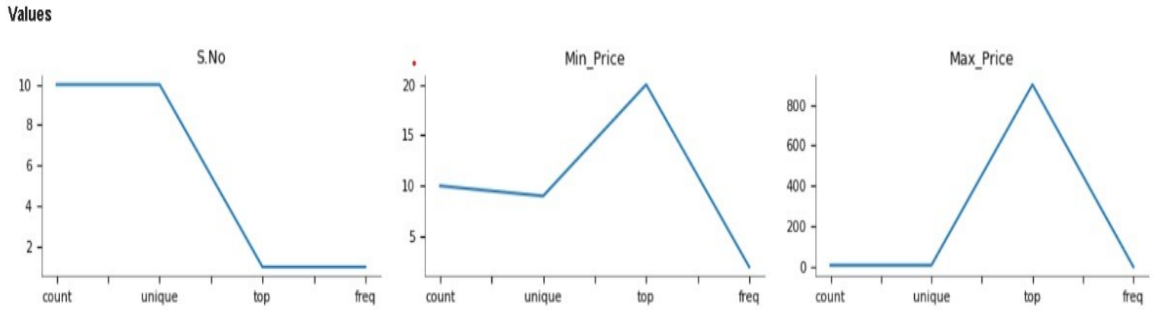
```
10
```

- **Describe( ) :** This method is used to get the details of dataset in mathematical way.

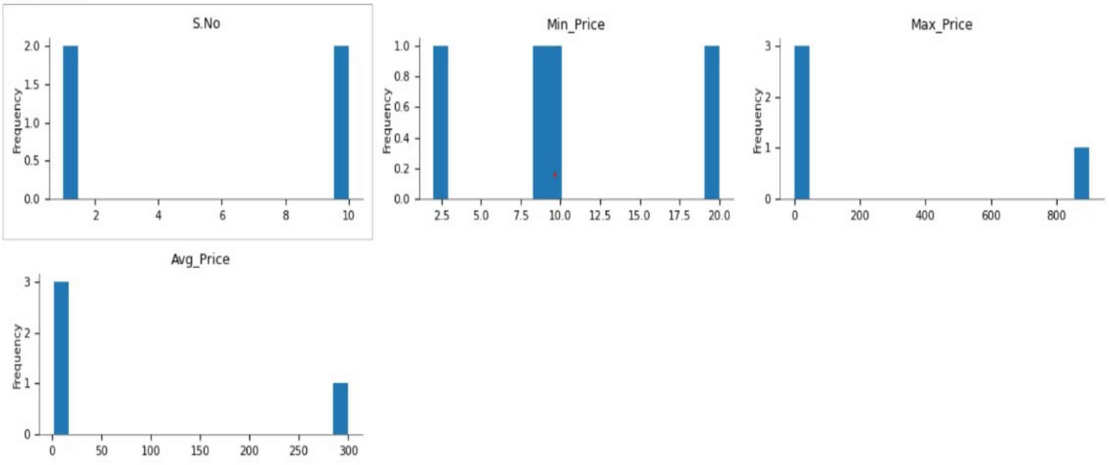
```
#Describing the Dataset using describe()
describe = cop.describe()
describe
```



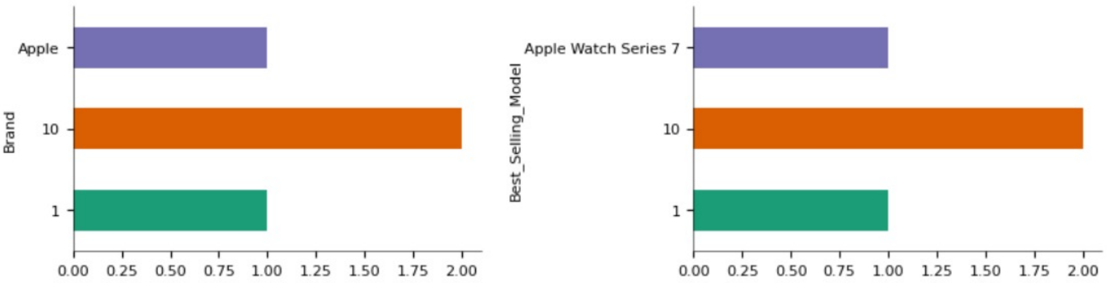
	S.No	Brand	Min_Price	Max_Price	Avg_Price	Sold_in_2020	Sold_in_2021	Sold_in_2022	Sold_in_2023	Best_Selling_Model
count	10	10	10	10	10	10	10	10	10	10
unique	10	10	9	10	9	9	9	9	10	10
top	1	Apple	20	900	300	100	250	275	160	Apple Watch Series 7
freq	1	1	2	1	2	2	2	2	1	1



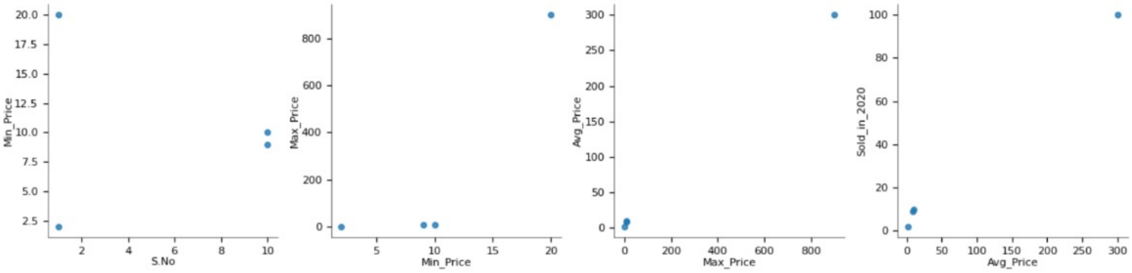
Distributions



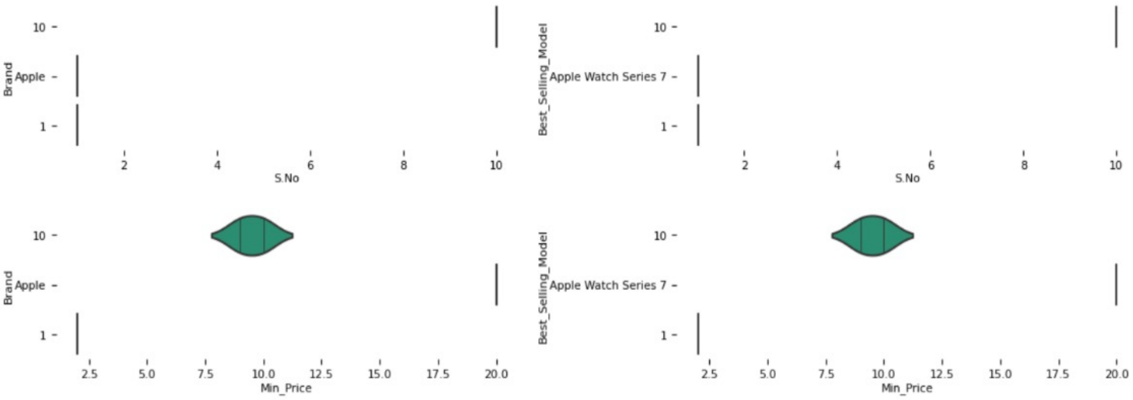
Categorical distributions

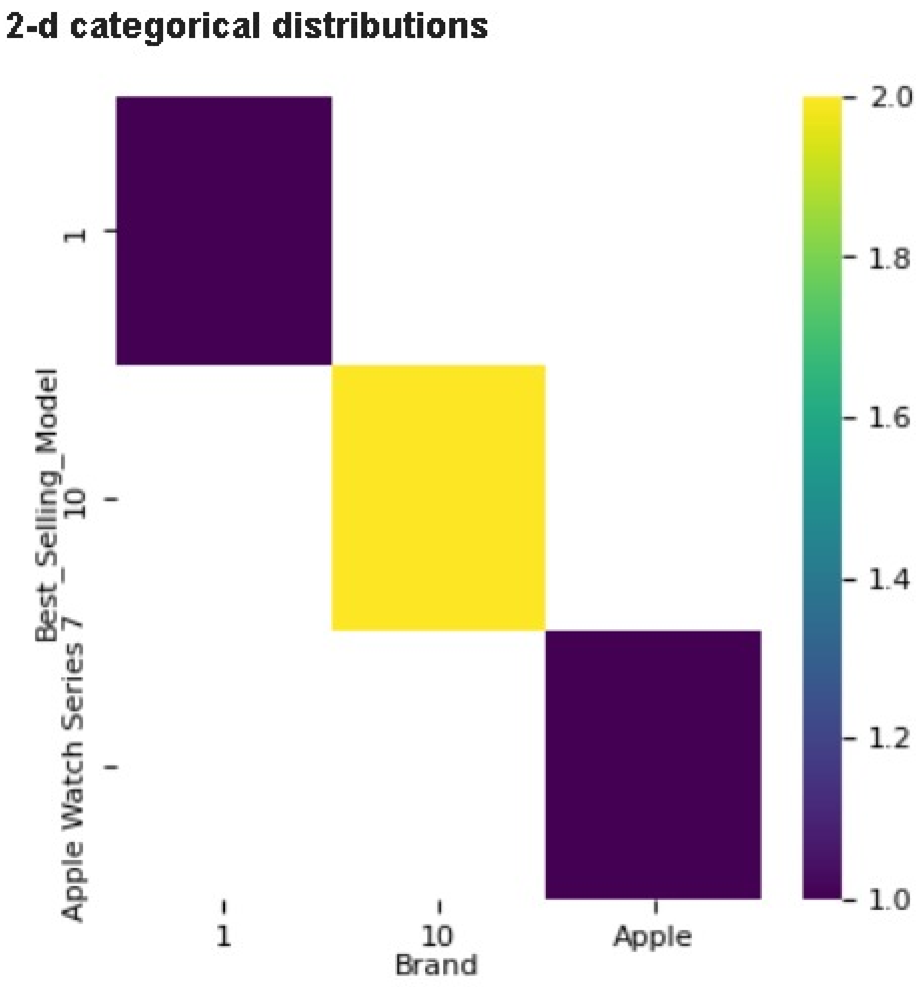


2-d distributions



Faceted distributions





Data Munging :

It is process of filtering the data

Head( ) : This method is used to get the records from top to bottom by default the head method returns the top 5 rows.

```
#Print the first 5 rows of Data using head()
head = cop.head()
print(head)
```

```
S.No   Brand Min_Price Max_Price Avg_Price Sold_in_2020 Sold_in_2021  \
0    1   Apple       90       900       200          100          150
1    2   Noise        9        40        20         350         450
2    3  Fossil       20       600       300          150          175
3    4   Titan       12       330       125          400          500
```

```
4    5   Wrogn       10      200       75         200         250

  Sold_in_2022 Sold_in_2023                         Best_Selling_Model
0          130          160                         Apple Watch Series 7
1          400          425                  Noise Pulse 2 Max Smartwatch
2          150          200  Fossil Fenmore Analog Black Dial Men's Watch
3          450          475                                   Titan Smart
4          230          250                     Wrogn fitness smart band
```

Tail( ) : This method is used to get the records from bottom to up by default the tail method returns the bottom to up 5 rows.

```
#Print the last 5 rows of Data Using tail()
tail = cop.tail()
print(tail)
```

```
S.No     Brand Min_Price Max_Price Avg_Price Sold_in_2020 Sold_in_2021  \
5    6   Samsung        35       250       100          100           75
6    7    Google       50       700       300           50           50
7    8  Fastrack       20       150        70          300          250
8    9  Amazefit        5        60        30          250          300
9   10     Rolex      500      1000       600           30           25

  Sold_in_2022 Sold_in_2023              Best_Selling_Model
5           80          100  Galaxy Watch 6 Classic hands-on
6           60           65        Google Pixel Watch (GPS)
7          275          290              Fastrack Reflex Vox
8          275          325                Amazfit T Rex Pro
9           25           35                   Rolex Daytona
```

Rank( ) :

Gives a rank to the columns or entire data frame according to the ascending order or descending order for Numerical values , for alphabets it follows alphabetical order.

```
#Giving the Priority to values to Data using rank()

rank = cop.rank()
print(rank)
```

```
S.No  Brand  Min_Price  Max_Price  Avg_Price  Sold_in_2020  Sold_in_2021  \
0   1.0    2.0       9.0       9.0       7.0           3.5           4.0
1   2.0    6.0       2.0       1.0       1.0           9.0           9.0
2   3.0    4.0       5.5       7.0       8.5           5.0           5.0
3   4.0    9.0       4.0       6.0       6.0          10.0          10.0
4   5.0   10.0       3.0       4.0       4.0           6.0           6.5
5   6.0    8.0       7.0       5.0       5.0           3.5           3.0
6   7.0    5.0       8.0       8.0       8.5           2.0           2.0
7   8.0    3.0       5.5       3.0       3.0           8.0           6.5
8   9.0    1.0       1.0       2.0       2.0           7.0           8.0
9  10.0    7.0      10.0      10.0      10.0           1.0           1.0

   Sold_in_2022  Sold_in_2023  Best_Selling_Model
0           4.0           4.0                 2.0
1           9.0           9.0                 7.0
2           5.0           5.0                 4.0
3          10.0          10.0                 9.0
4           6.0           6.0                10.0
5           3.0           3.0                 5.0
6           2.0           2.0                 6.0
7           7.5           7.0                 3.0
8           7.5           8.0                 1.0
9           1.0           1.0                 8.0
```

CORRELATION-

- It is a relation between two data column data members.
- We use method called corr( ) • It is scaled form of a covariance.
- Correlation values lies between (-1 to +1)

Attribute1.corr(attribute2)

Types of correlations:

we have three types of correlation.

- Positive – (0 to1)
- Negative – (0 to -1)
- No correlation – (0)

```
# Finding the Correlation between min and max
price of watch using corr()

print(Data['Min_Price'].corr(Data['Max_Price']))
```

```
0.6864875223214113
```

COVARIANCE-

The covariance is the relation between two data members of two different columns.

- It is the measure of a correlation. •
  It lies between (-∞ to ∞).

```
# Finding the Co-Variance between min and max
price of watch using cov()

print(Data['Min_Price'].cov(Data['Max_Price']))
```

```
36521.88888888888
```

5)VISUALIZING THE DATA & SHARING THE RESULT

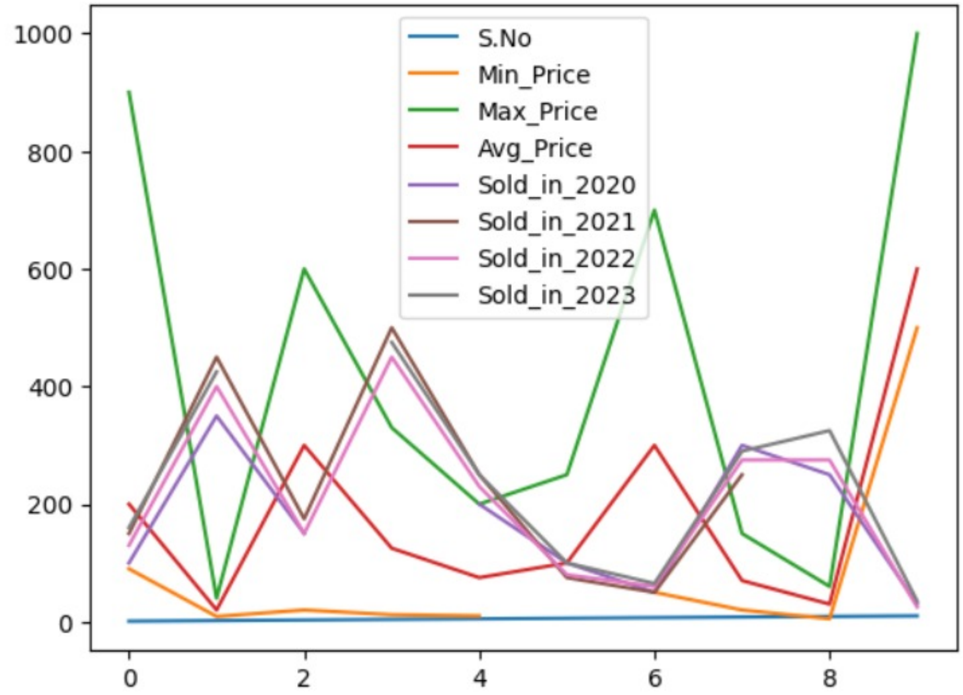
Data visualization is a process of representing data in a graphical way. Here we can represent the below graph formats.

1. Line Graph
2. Bar Graph
3. Box Graph
4. KDE Graph
5. Area Graph
6. Histogram Graph

Here we need to use matplotlib module to represent graphs using the code.

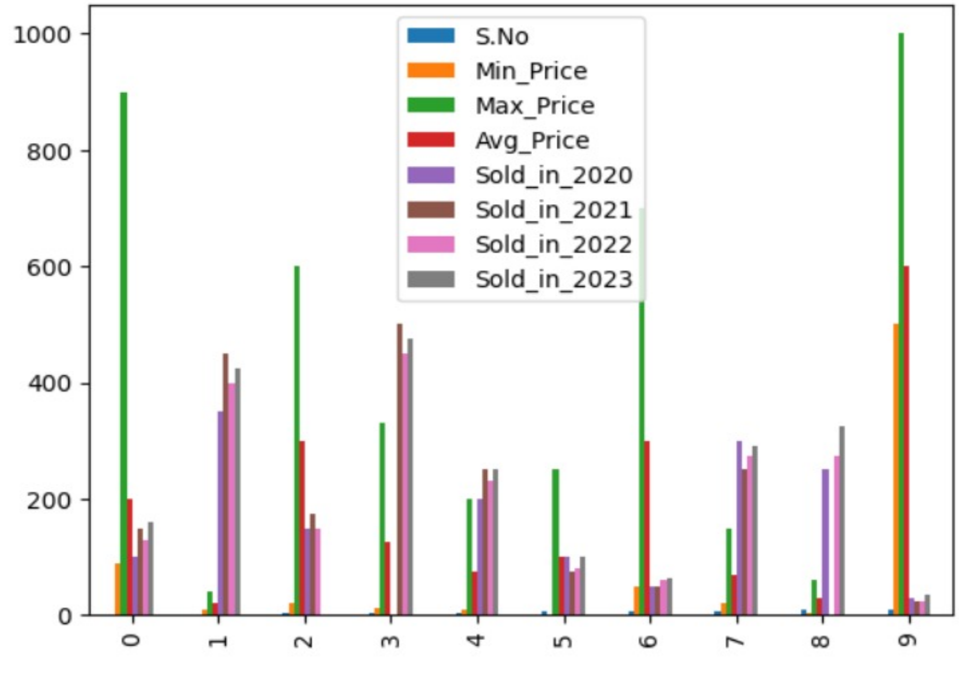
```
# Line Graph
line_Graph = Data.plot.line()
line_Graph
```

```
<Axes: >
```

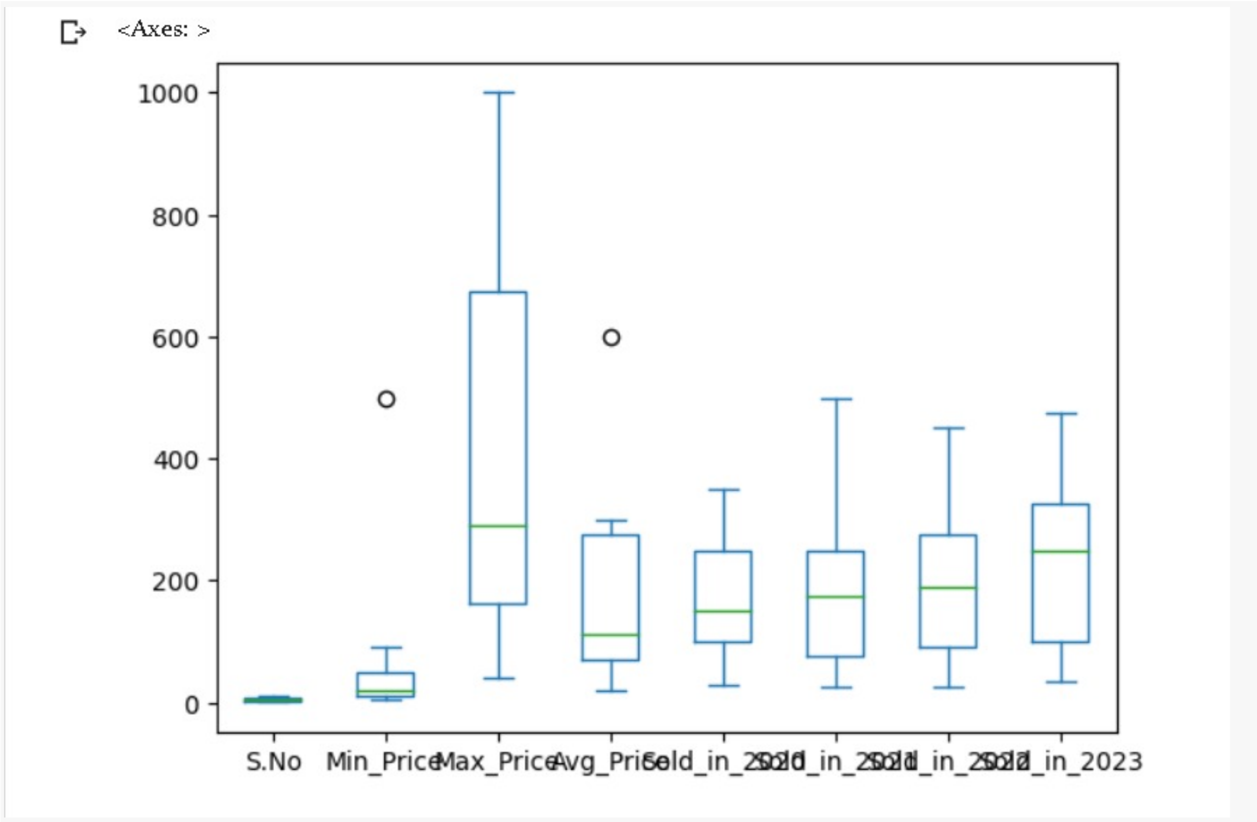


```
# Bar Graph
bar_graph = Data.plot.bar()
bar_graph
```

```
<Axes: >
```



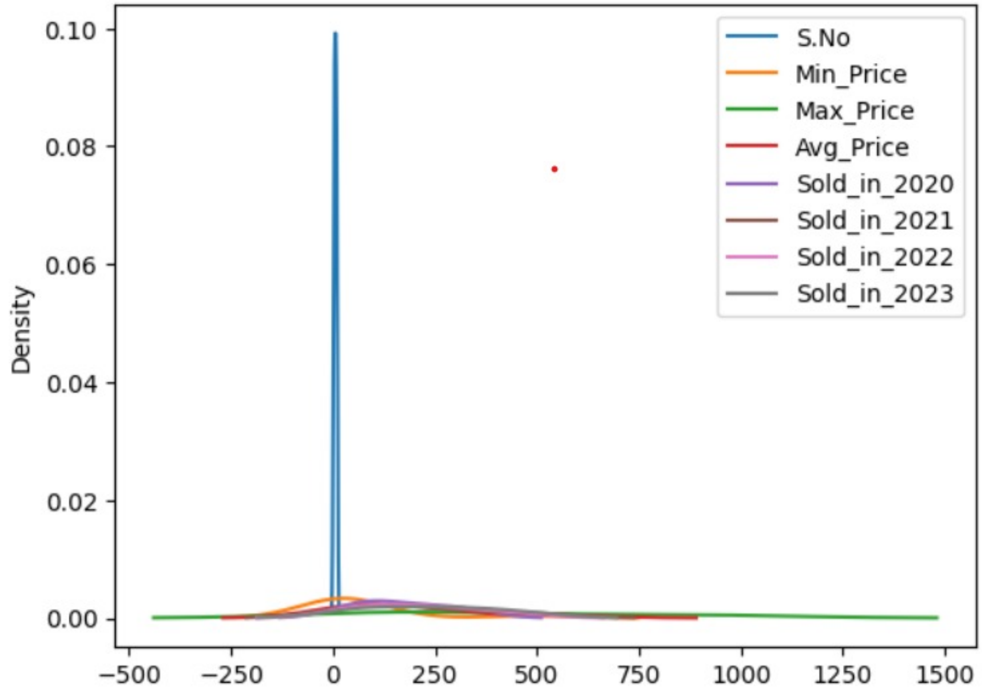
```
 # Box Graph
Box = Data.plot.box()
Box
```



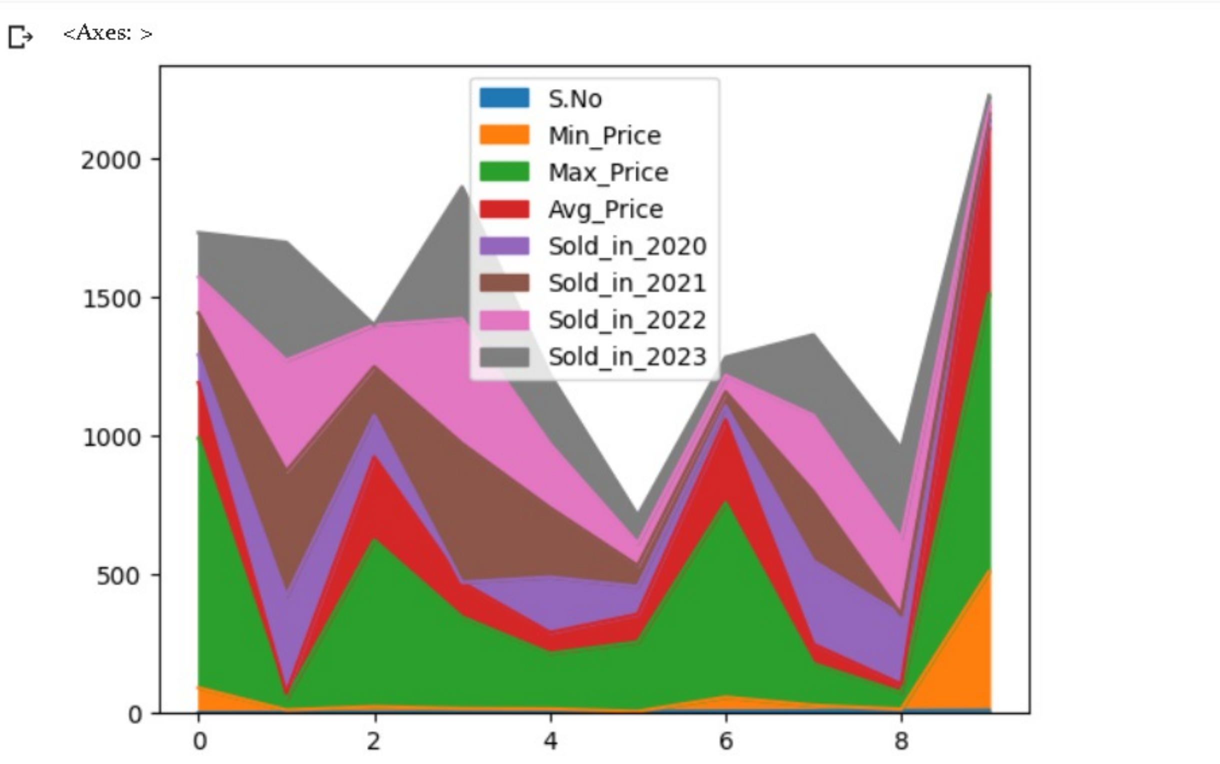
```
<Axes: >
```

```
# kde graph
kde = Data.plot.kde()
kde
```

```
<Axes: ylabel='Density'>
```

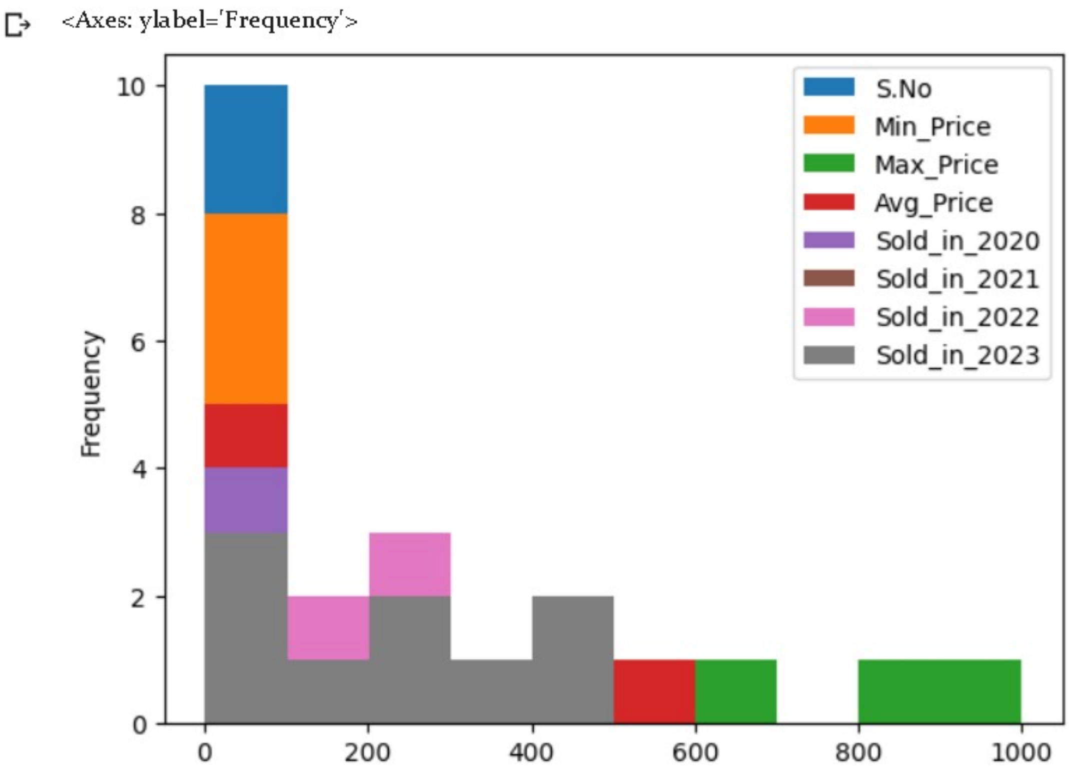


```
# Area Graph
area_graph = Data.plot.area()
area_graph
```



```
<Axes: >
```

```
# Histogram
hist = Data.plot.hist()
hist
```

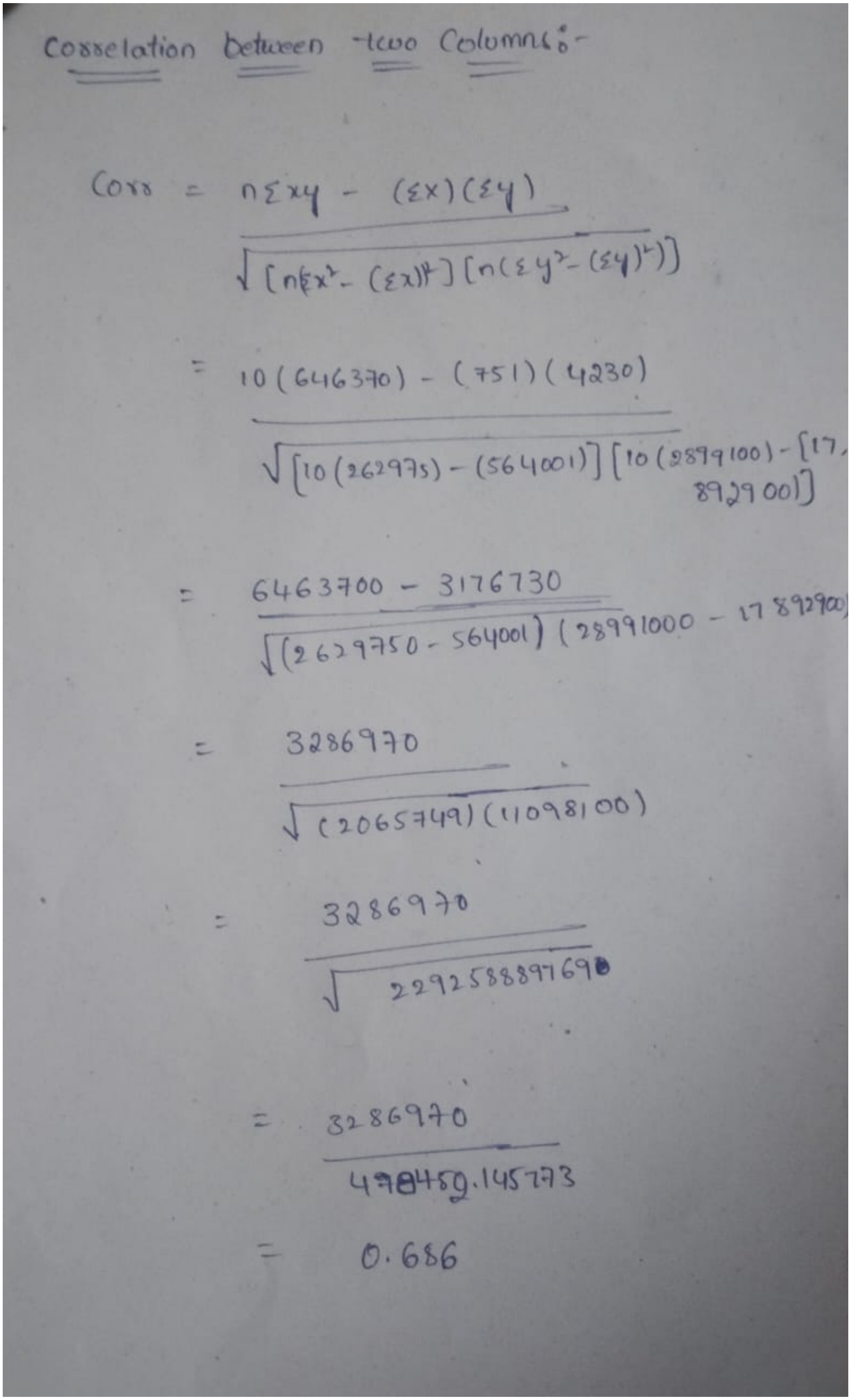


```
<Axes: ylabel='Frequency'>
```

```
# scatter plot
scatter = Data.plot.scatter('Min_Price','Max_Price')
scatter
```

EMBRACING THE RESULTS

CORELATION AND CO-VARIANCE ARE MANUALLY CALCULATED



Correlation between two Columns:-

$$\text{Corr} = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n(\Sigma y^2 - (\Sigma y)^2)]}}$$

$$= \frac{10(646370) - (751)(4230)}{\sqrt{[10(262975) - (564001)][10(2899100) - [17,892900)]}}$$

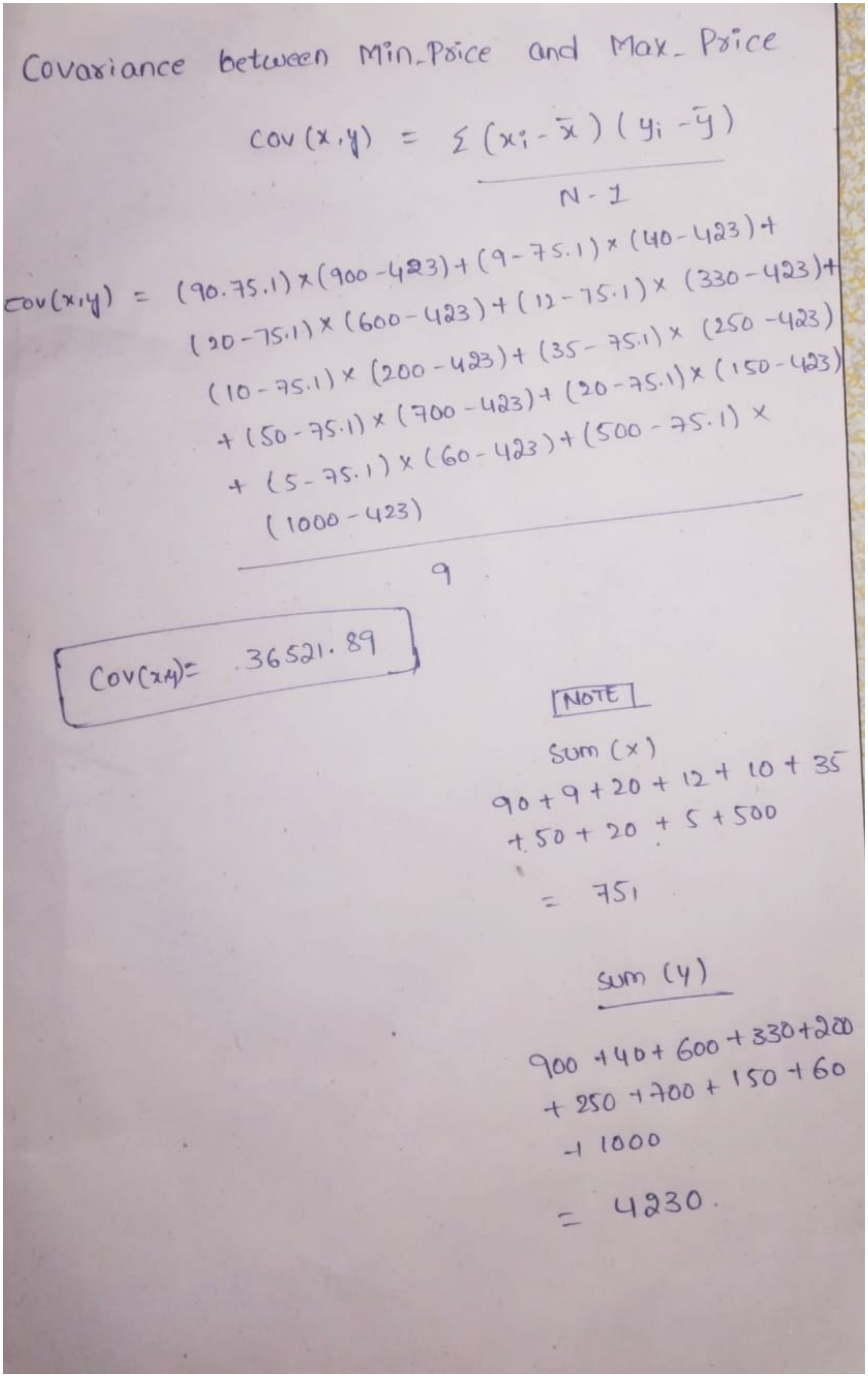
$$= \frac{6463700 - 3176730}{\sqrt{(2629750 - 564001)(28991000 - 17892900)}}$$

$$= \frac{3286970}{\sqrt{(2065749)(1109810,0)}}$$

$$= \frac{3286970}{\sqrt{2292588897690}}$$

$$= \frac{3286970}{478459.145773}$$

$$= 0.686$$



Covariance between Min_Price and Max_Price

$$\text{cov}(x,y) = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

$$\text{cov}(x,y) = \frac{\begin{gathered}(90.75.1)\times(900-423)+(9-75.1)\times(40-423)+\\(20-75.1)\times(600-423)+(12-75.1)\times(330-423)+\\(10-75.1)\times(200-423)+(35-75.1)\times(250-423)\\+(50-75.1)\times(700-423)+(20-75.1)\times(150-423)\\+(5-75.1)\times(60-423)+(500-75.1)\times\\(1000-423)\end{gathered}}{9}$$

$$\boxed{\text{Cov}(x,y) = 36521.89}$$

NOTE

sum (x)

90 + 9 + 20 + 12 + 10 + 35
+ 50 + 20 + 5 + 500

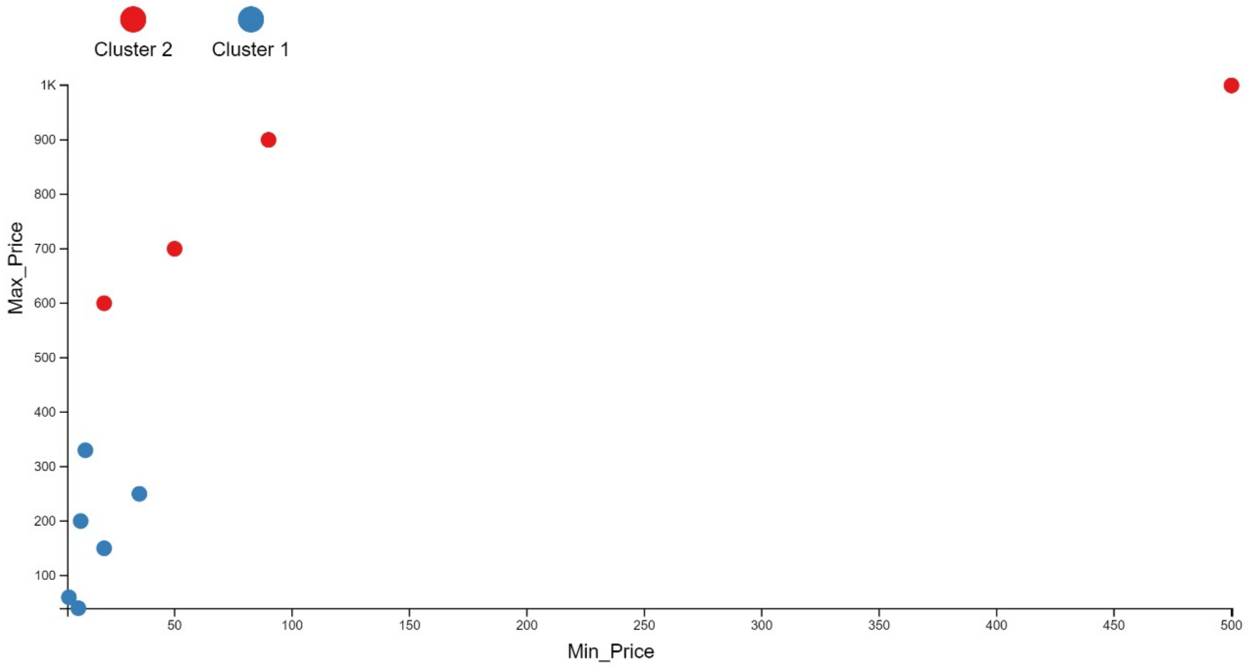
= 751

sum (y)

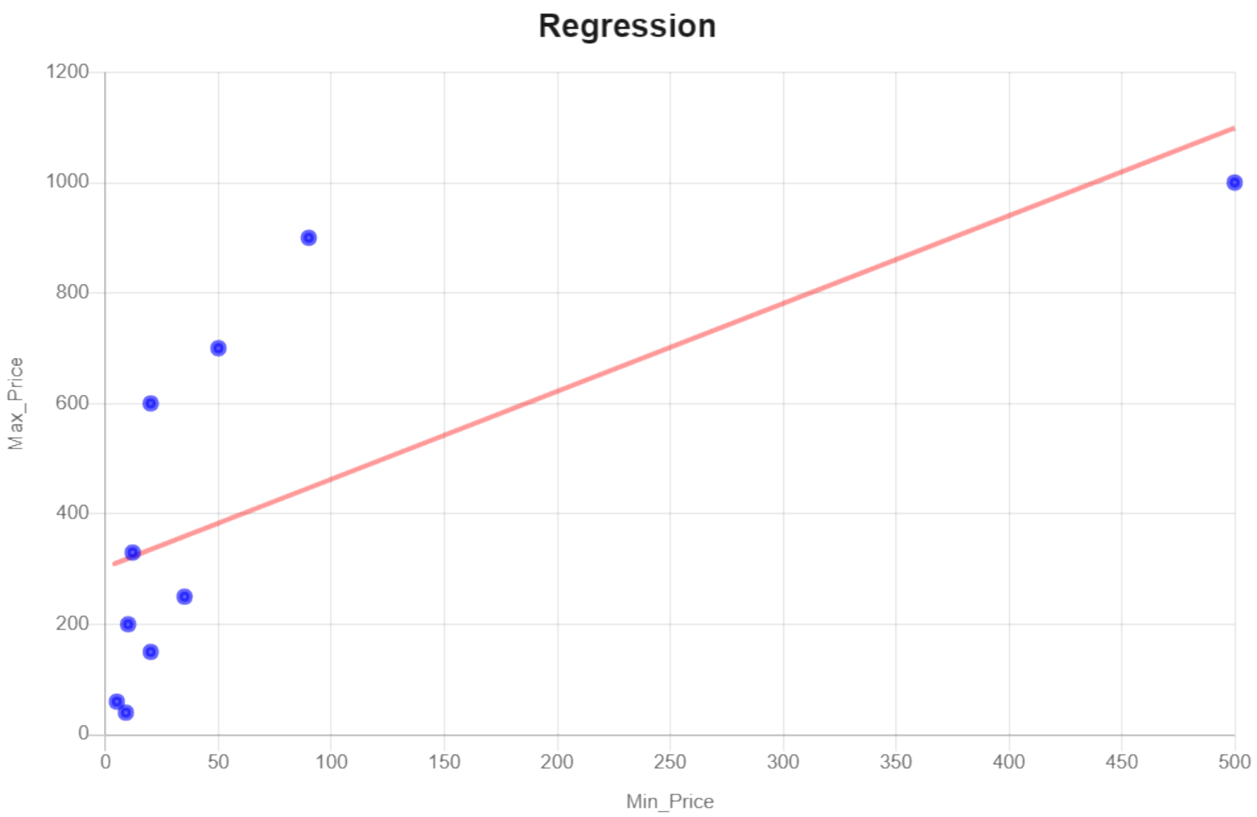
900 + 40 + 600 + 330 + 200
+ 250 + 700 + 150 + 60
+ 1000

= 4230.

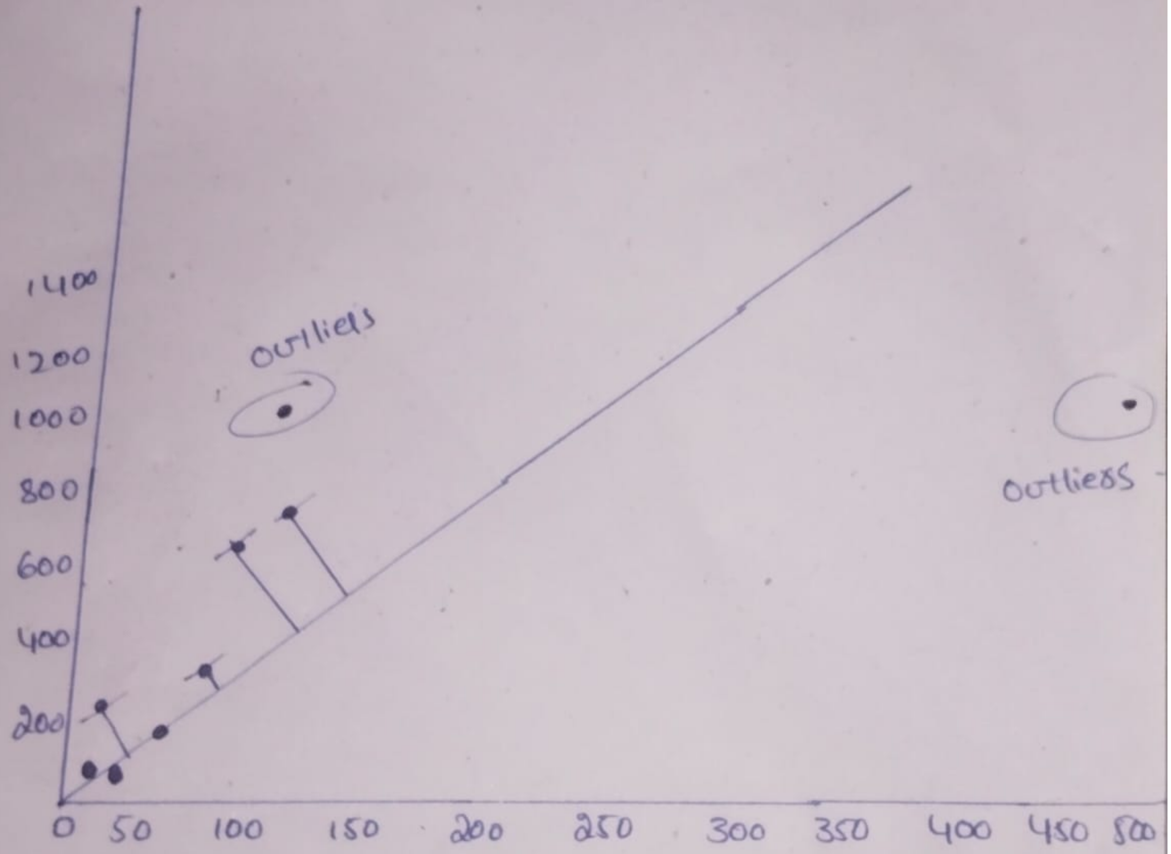
Clustering :



Regression :


Regression
1200
1000
800
600
400
200
0
Max_Price
0
50
100
150
200
250
300
350
400
450
500
Min_Price

Regression :-



1400
1200
1000
800
600
400
200
0
50
100
150
200
250
300
350
400
450
500
outliers
outliers

SUMMARY

In this project, we aimed to develop a smartwatch price prediction using data analytics, with a focus on time series analysis techniques implemented in Python.

In summary, a smartwatch price prediction project involves data collection, preprocessing, modeling, and evaluation to create a predictive tool for estimating smartwatch prices, benefiting both consumers and businesses in the smartwatch industry.

Benefits:

- Consumers can make more informed purchasing decisions by estimating the fair price of a smartwatch.
- Businesses can optimize their pricing strategies to remain competitive in the market.
- Researchers and analysts can gain insights into factors affecting smartwatch prices and market trends.

Challenges:

- The accuracy of price predictions depends on the quality and quantity of data available.
- Smartwatch prices may be influenced by various external factors that are challenging to quantify.
- Ensuring the model's relevance over time requires continuous data updates and monitoring.