Ex.No: 13 Learning – Use Supervised Learning(Miniproject)

DATE:4/11/24

REGISTER NUMBER : 212222040091

AIM:

To write a program to train the classifier for Air Quality.

Algorithm:

```
Step 1 : Start the program.
Step 2 : Import the necessary packages, such as NumPy and Pandas.
Step 3 : Install and import Gradio for creating the user interface.
Step 4 : Load the Air Quality dataset using Pandas.
Step 5 : Split the dataset into input features (x) and target labels (y).
Step 6 : Split the data into training and testing sets using train_test_split.
Step 7 : Standardize the training and testing data using the StandardScaler.
Step 8 : Instantiate the MLPClassifier model with 1000 iterations and train the model on
Step 9 : Print the model's accuracy on both the training and testing sets.
Step 10 : Take input values for Air Quality features and predict the outcome using the t
Step 11 : Stop the program.
```

Program:

Importing Libraries

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline
sns.set()
```

Import Data and Analysis

```
df=pd.read_csv("/content/air-quality-india.csv")
df
```

```
df.head()
```

```
df.tail()
df.shape
```

```
df.info()
```

```
df.isnull().sum()
```

```
df.describe().T.style.background_gradient(cmap="Blues")
```

```
print(df["PM2.5"].describe())
```

```
df.nunique()
```

```
pd.DataFrame(df["Year"].value_counts())
```

```
pd.DataFrame(df["Month"].value_counts().sort_index(ascending=True))
```

```
pd.DataFrame(df["Hour"].value_counts().sort_values(ascending=False))
```

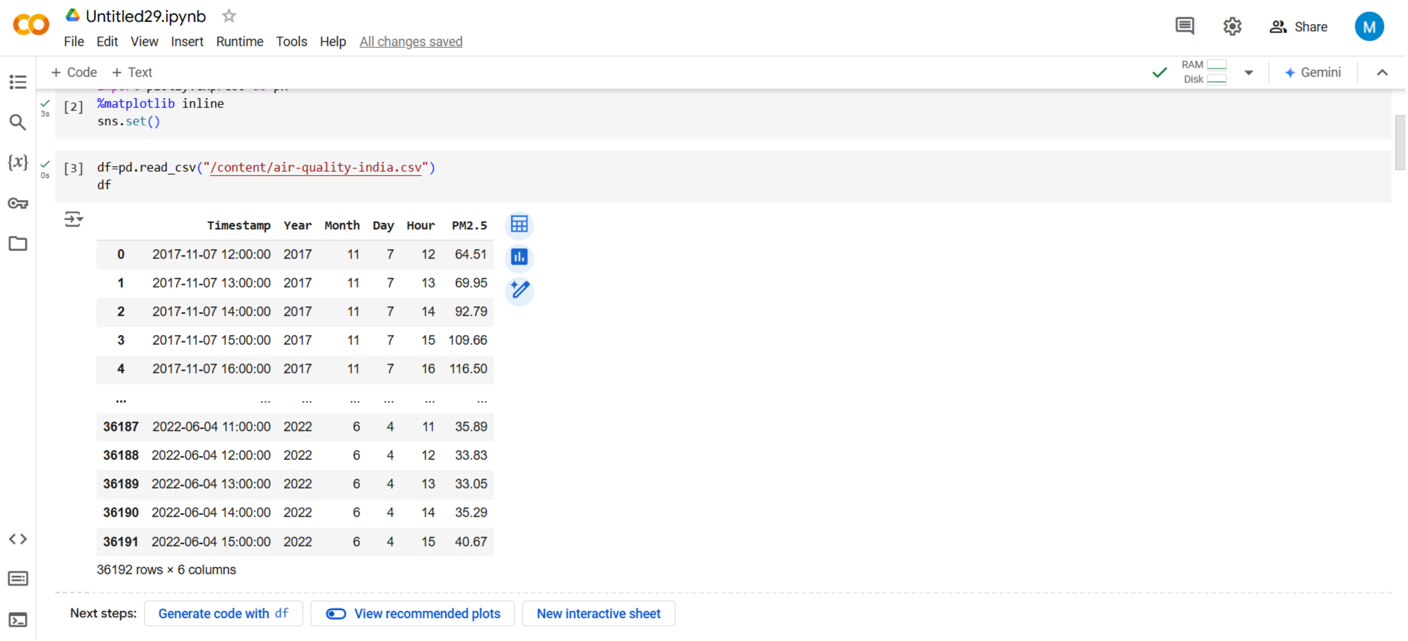
```
pd.DataFrame(df["PM2.5"].sort_values(ascending=False).head(15))
```

```
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

```
df['Day_of_Week'] = df['Timestamp'].dt.dayofweek
df['Season'] = df['Month'].apply(lambda x: 'Winter' if x in [12, 1, 2]
                                                 else 'Summer' if x in [3, 4, 5]
                                                 else 'Monsoon' if x in [6, 7, 8]
                                                 else 'Autumn')


print(df.head())
```

Output:

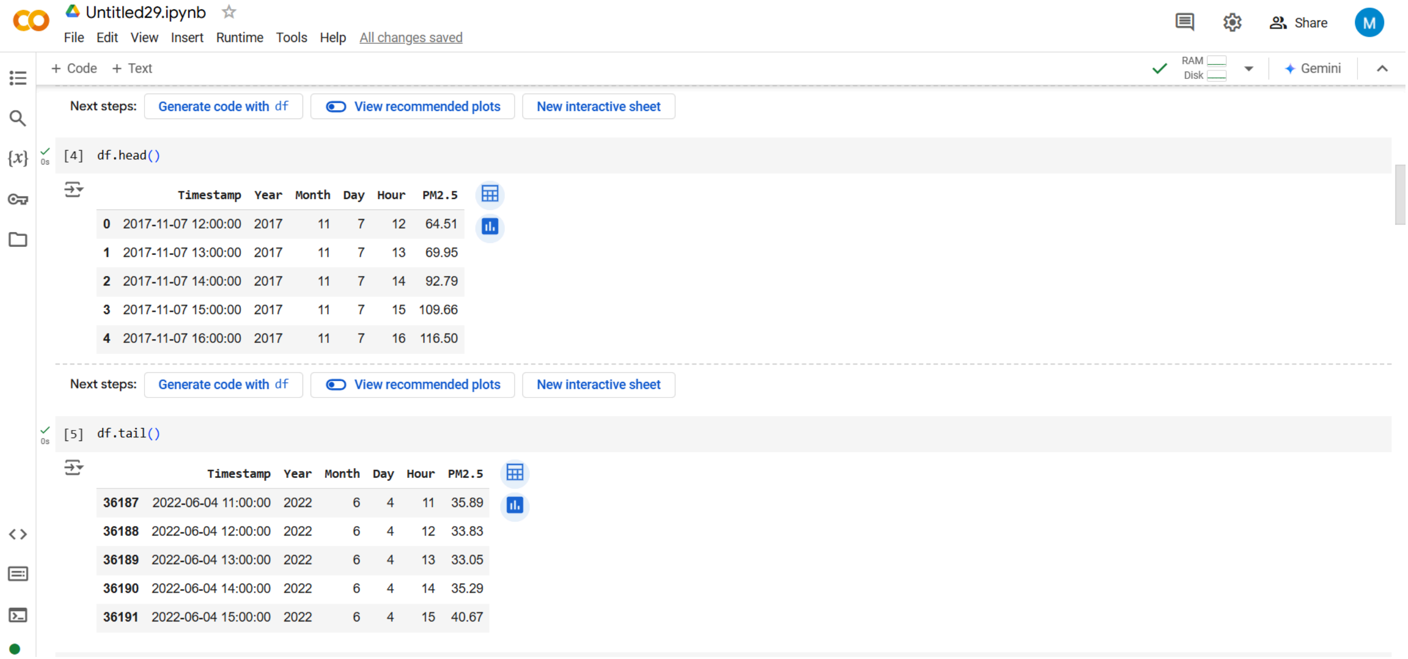


```
[2] %matplotlib inline
    sns.set()
```

```
[3] df=pd.read_csv("/content/air-quality-india.csv")
    df
```

	Timestamp	Year	Month	Day	Hour	PM2.5
0	2017-11-07 12:00:00	2017	11	7	12	64.51
1	2017-11-07 13:00:00	2017	11	7	13	69.95
2	2017-11-07 14:00:00	2017	11	7	14	92.79
3	2017-11-07 15:00:00	2017	11	7	15	109.66
4	2017-11-07 16:00:00	2017	11	7	16	116.50
...	...	...	...	...	...	...
36187	2022-06-04 11:00:00	2022	6	4	11	35.89
36188	2022-06-04 12:00:00	2022	6	4	12	33.83
36189	2022-06-04 13:00:00	2022	6	4	13	33.05
36190	2022-06-04 14:00:00	2022	6	4	14	35.29
36191	2022-06-04 15:00:00	2022	6	4	15	40.67

36192 rows × 6 columns


Untitled29.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM Disk
Gemini
Next steps: Generate code with df View recommended plots New interactive sheet

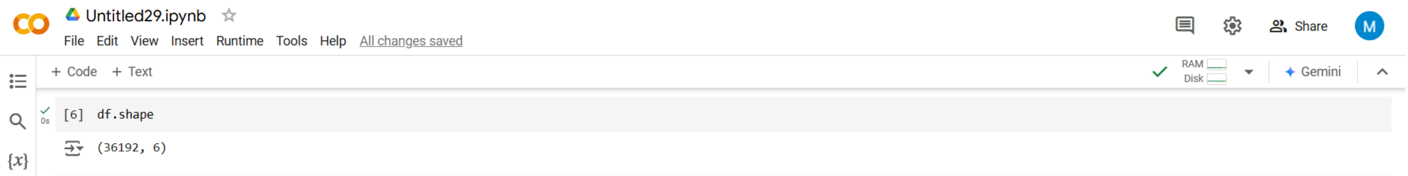
```
[4] df.head()
```

	Timestamp	Year	Month	Day	Hour	PM2.5
0	2017-11-07 12:00:00	2017	11	7	12	64.51
1	2017-11-07 13:00:00	2017	11	7	13	69.95
2	2017-11-07 14:00:00	2017	11	7	14	92.79
3	2017-11-07 15:00:00	2017	11	7	15	109.66
4	2017-11-07 16:00:00	2017	11	7	16	116.50

Next steps: Generate code with df View recommended plots New interactive sheet

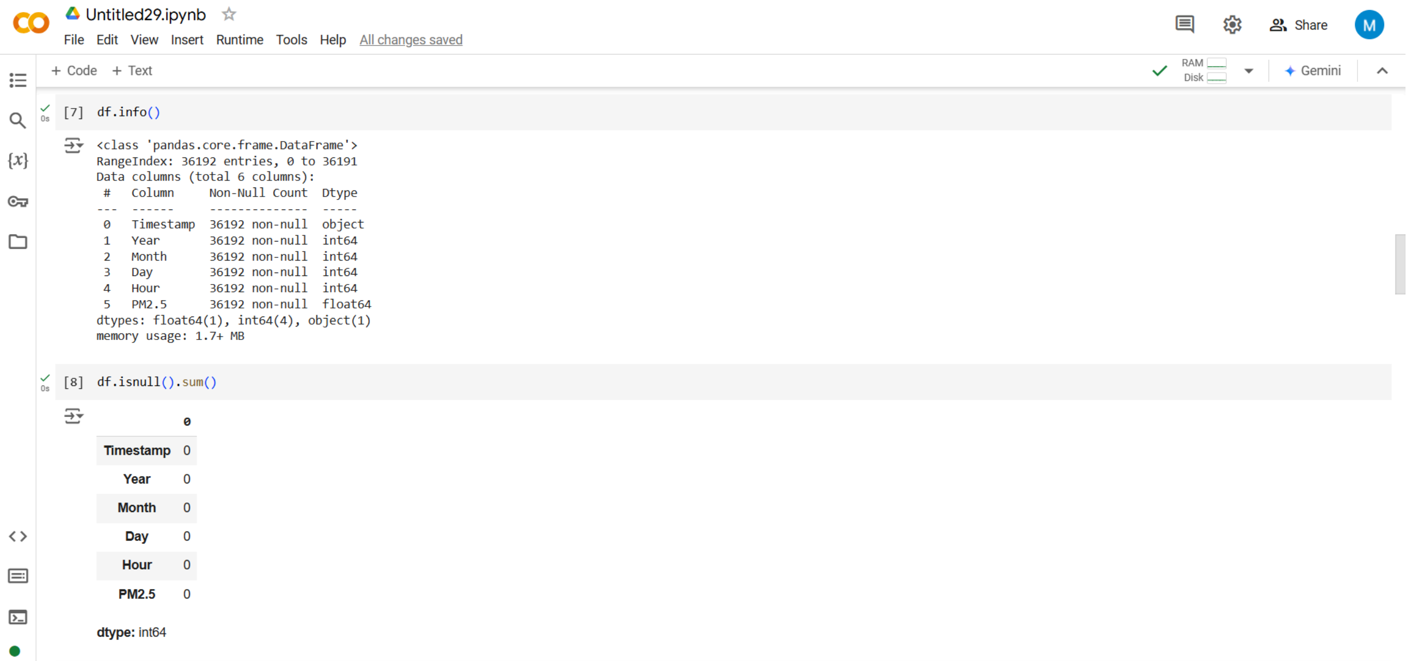
```
[5] df.tail()
```

	Timestamp	Year	Month	Day	Hour	PM2.5
36187	2022-06-04 11:00:00	2022	6	4	11	35.89
36188	2022-06-04 12:00:00	2022	6	4	12	33.83
36189	2022-06-04 13:00:00	2022	6	4	13	33.05
36190	2022-06-04 14:00:00	2022	6	4	14	35.29
36191	2022-06-04 15:00:00	2022	6	4	15	40.67



```
[6] df.shape
```

```
(36192, 6)
```



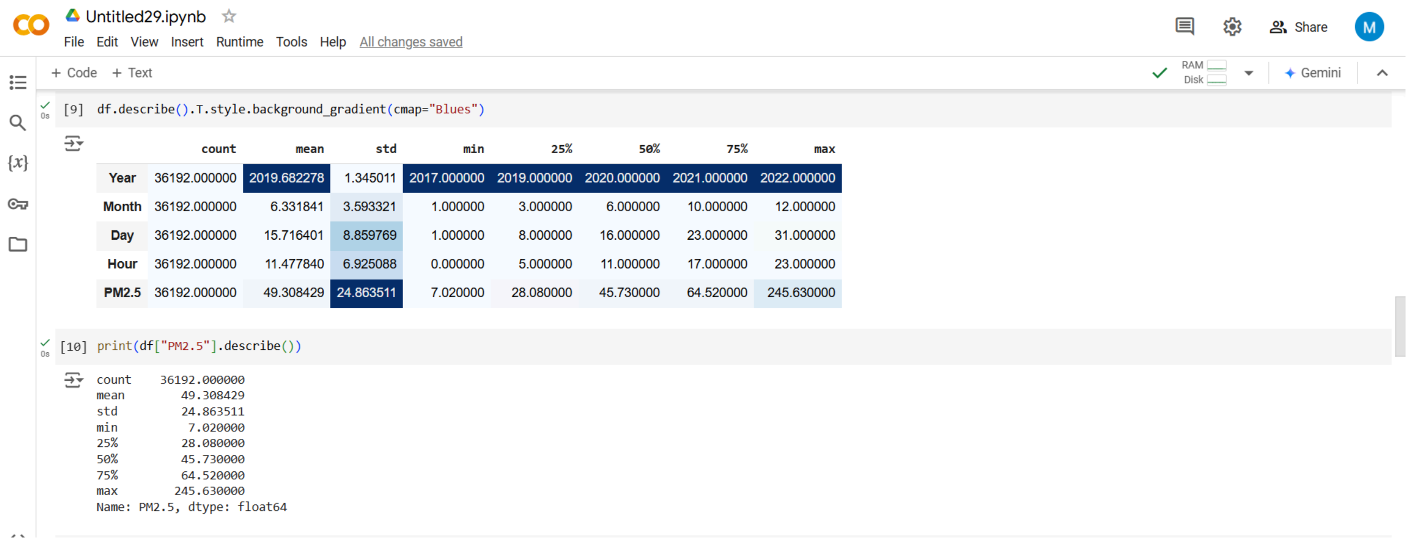
```
[7] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36192 entries, 0 to 36191
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Timestamp  36192 non-null  object
 1   Year       36192 non-null  int64
 2   Month      36192 non-null  int64
 3   Day        36192 non-null  int64
 4   Hour       36192 non-null  int64
 5   PM2.5      36192 non-null  float64
dtypes: float64(1), int64(4), object(1)
memory usage: 1.7+ MB
```

```
[8] df.isnull().sum()
```

	0
Timestamp	0
Year	0
Month	0
Day	0
Hour	0
PM2.5	0

dtype: int64

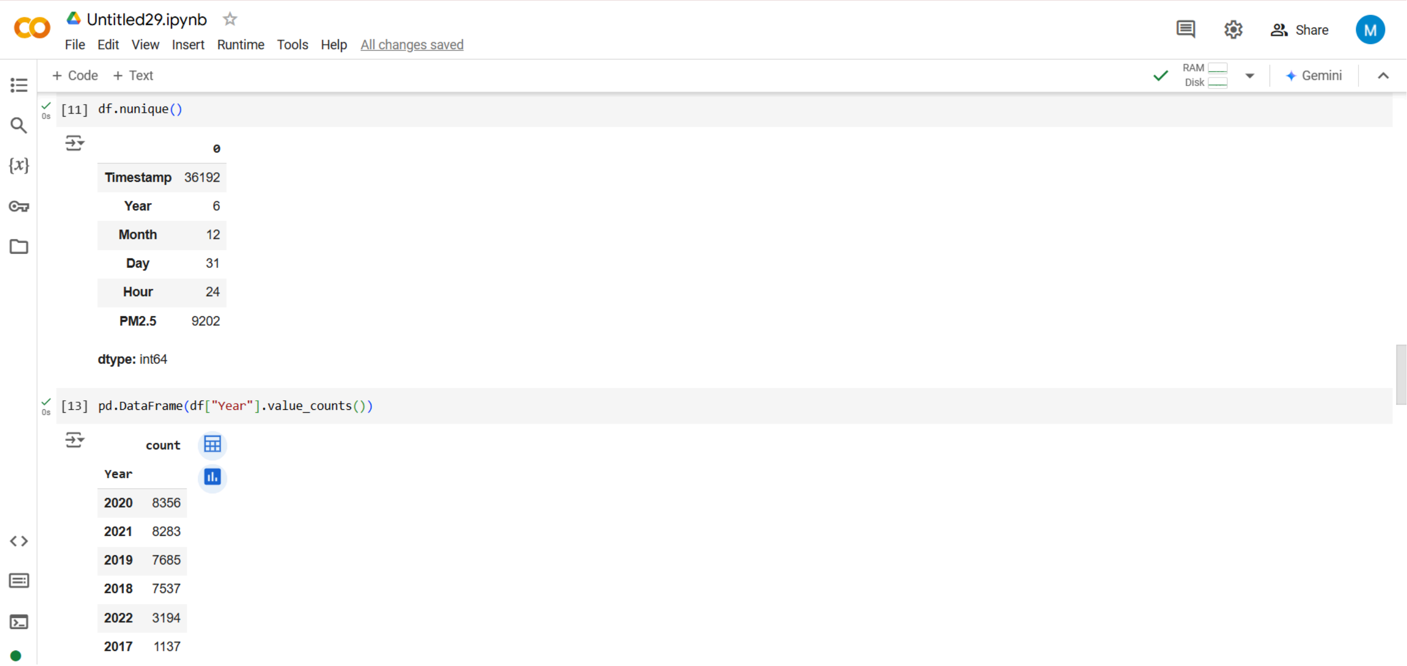


```
[9] df.describe().T.style.background_gradient(cmap="Blues")
```

	count	mean	std	min	25%	50%	75%	max
Year	36192.000000	2019.682278	1.345011	2017.000000	2019.000000	2020.000000	2021.000000	2022.000000
Month	36192.000000	6.331841	3.593321	1.000000	3.000000	6.000000	10.000000	12.000000
Day	36192.000000	15.716401	8.859769	1.000000	8.000000	16.000000	23.000000	31.000000
Hour	36192.000000	11.477840	6.925088	0.000000	5.000000	11.000000	17.000000	23.000000
PM2.5	36192.000000	49.308429	24.863511	7.020000	28.080000	45.730000	64.520000	245.630000

```
[10] print(df["PM2.5"].describe())
```

```
count    36192.000000
mean        49.308429
std         24.863511
min          7.020000
25%         28.080000
50%         45.730000
75%         64.520000
max        245.630000
Name: PM2.5, dtype: float64
```



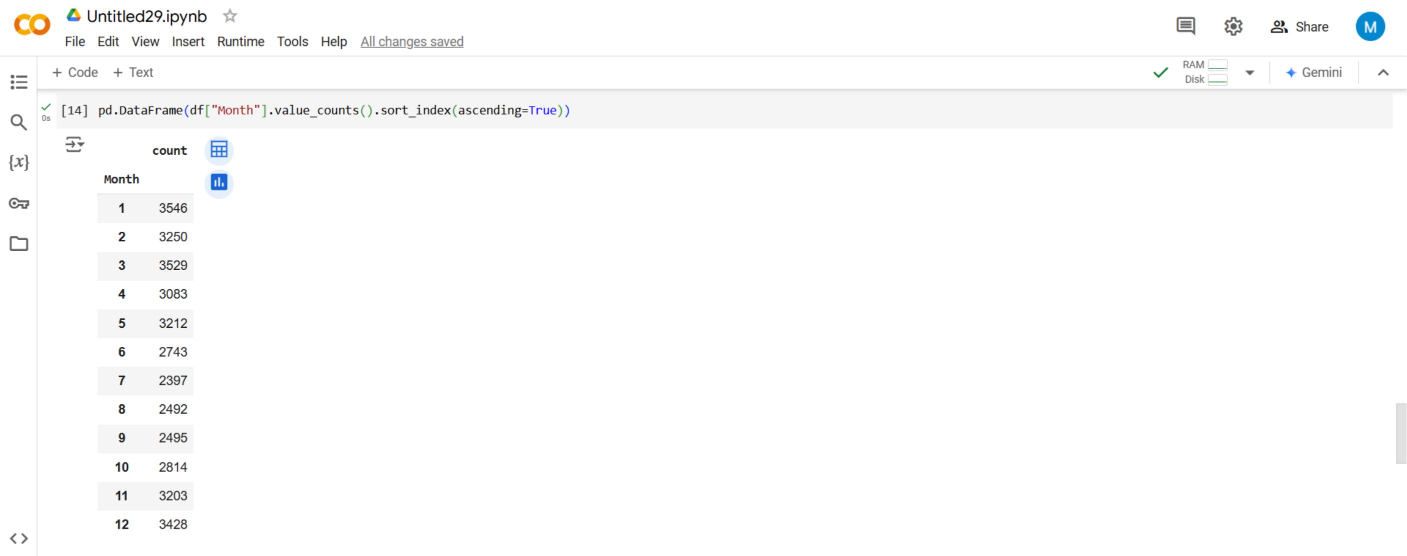
```
[11] df.nunique()
```

```
              0
Timestamp 36192
     Year     6
    Month    12
      Day    31
     Hour    24
    PM2.5  9202

dtype: int64
```

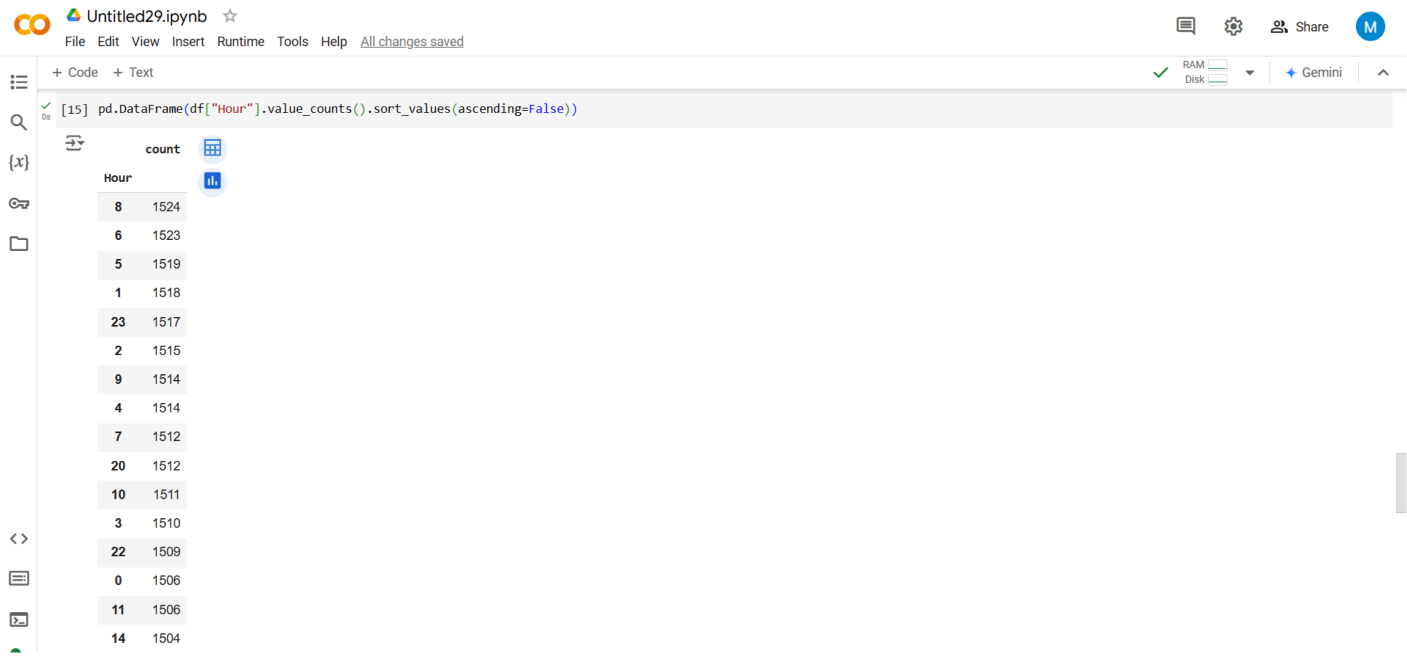
```
[13] pd.DataFrame(df["Year"].value_counts())
```

```
      count
Year
2020   8356
2021   8283
2019   7685
2018   7537
2022   3194
2017   1137
```



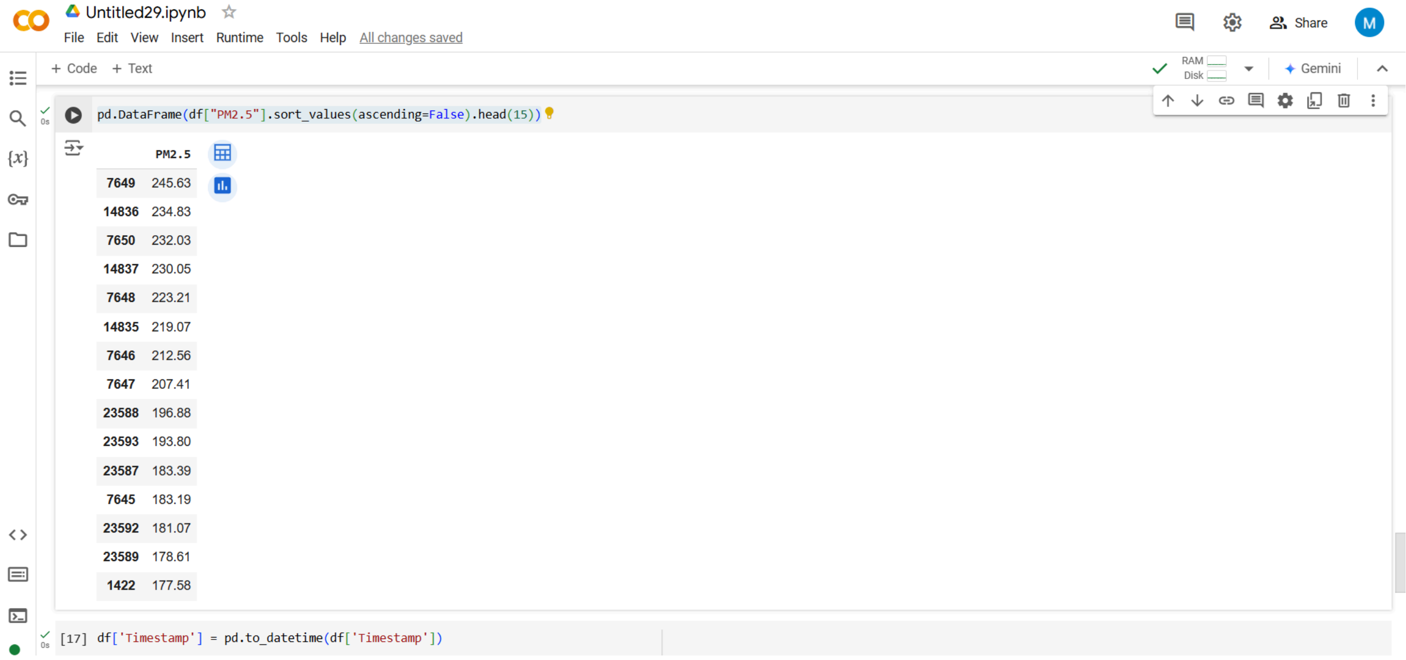
```
[14] pd.DataFrame(df["Month"].value_counts().sort_index(ascending=True))
```

```
       count
Month
    1   3546
    2   3250
    3   3529
    4   3083
    5   3212
    6   2743
    7   2397
    8   2492
    9   2495
   10   2814
   11   3203
   12   3428
```



```
[15] pd.DataFrame(df["Hour"].value_counts().sort_values(ascending=False))
```

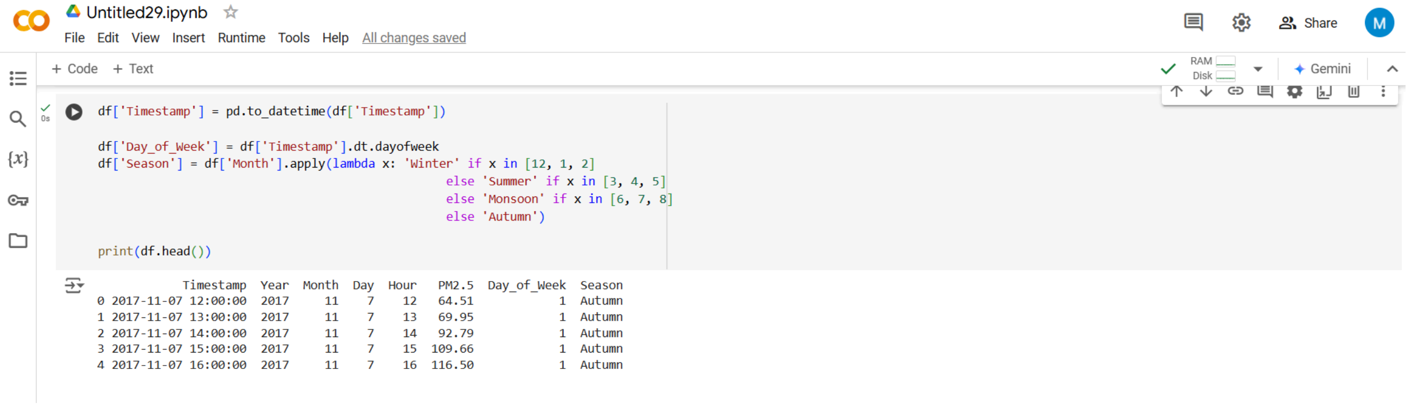
```
      count
Hour
   8   1524
   6   1523
   5   1519
   1   1518
  23   1517
   2   1515
   9   1514
   4   1514
   7   1512
  20   1512
  10   1511
   3   1510
  22   1509
   0   1506
  11   1506
  14   1504
```



```
pd.DataFrame(df["PM2.5"].sort_values(ascending=False).head(15))
```

	PM2.5
7649	245.63
14836	234.83
7650	232.03
14837	230.05
7648	223.21
14835	219.07
7646	212.56
7647	207.41
23588	196.88
23593	193.80
23587	183.39
7645	183.19
23592	181.07
23589	178.61
1422	177.58

```
[17] df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```



```
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

df['Day_of_Week'] = df['Timestamp'].dt.dayofweek
df['Season'] = df['Month'].apply(lambda x: 'Winter' if x in [12, 1, 2]
                                              else 'Summer' if x in [3, 4, 5]
                                              else 'Monsoon' if x in [6, 7, 8]
                                              else 'Autumn')


print(df.head())
```

```
            Timestamp  Year  Month  Day  Hour   PM2.5  Day_of_Week  Season
0 2017-11-07 12:00:00  2017     11    7    12   64.51            1  Autumn
1 2017-11-07 13:00:00  2017     11    7    13   69.95            1  Autumn
2 2017-11-07 14:00:00  2017     11    7    14   92.79            1  Autumn
3 2017-11-07 15:00:00  2017     11    7    15  109.66            1  Autumn
4 2017-11-07 16:00:00  2017     11    7    16  116.50            1  Autumn
```

Result:

Thus the system was trained successfully and the prediction was carried out.