In [1]:
```
#浏览量和访问量跟文章阅读量的关系
```

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns


# 모든 데이터를 볼 수 있는 ( head랑 tail만 보여주는게 아니라, 다 보여주는 방법)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

## 중국어 폰트 설정 하는 방법

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False
f_path = "C:\Windows\Fonts/simhei.ttf"
font_name = font_manager.FontProperties(fname=f_path).get_name()
rc('font', family=font_name)

# 폰트를 선명하게 하기
# retina설정
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")

data_name = f"浏览量和访问量跟文章阅读量的关系.csv"
df = pd.read_csv(data_name) #Encoding cp949는 못함

# Nan 값 0으로 만들어주기
df = df.fillna(value=0)
df.head()
```

```
C:\Users\toyou\AppData\Local\Temp/ipykernel_11688/3850729006.py:25: DeprecationWarning: `set_matplotlib_formats` is deprecated since IPython 7.23, directly use `matplotlib_inline.backend_inline.set_matplotlib_formats()`
  set_matplotlib_formats("retina")
```

Out[2]:

日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量	淘宝特点活动	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16

0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0	2022.3.4~2022.3.8：情人节	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	02-25	3.0	1.0	102.0	31.0	0.0	137.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

데이터처리

In [3]:

```
#df =df.drop(['Unnamed: 9'],axis=1)
df.drop(df.columns[8:], axis=1 , inplace=True)
df
```

Out[3]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量
0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0
4	02-25	3.0	1.0	102.0	31.0	0.0	137.0	835.0
5	02-26	13.0	4.0	127.0	28.0	0.0	172.0	1290.0
6	02-27	4.0	2.0	79.0	22.0	0.0	107.0	835.0
7	02-28	73.0	12.0	123.0	28.0	40.0	276.0	423.0
8	03-01	33.0	18.0	143.0	33.0	6.0	233.0	3199.0
9	03-02	20.0	4.0	158.0	32.0	0.0	214.0	2791.0
10	03-03	45.0	14.0	142.0	33.0	1.0	235.0	1470.0
11	03-04	28.0	10.0	154.0	34.0	0.0	226.0	701.0
12	03-05	12.0	6.0	50.0	17.0	0.0	85.0	567.0
13	03-06	2.0	2.0	46.0	15.0	0.0	65.0	835.0
14	03-07	193.0	27.0	34.0	9.0	1.0	264.0	1780.0
15	03-08	115.0	35.0	91.0	16.0	0.0	257.0	732.0

16	03-09	221.0	35.0	150.0	29.0	0.0	435.0	479.0
17	03-10	188.0	32.0	124.0	20.0	4.0	368.0	1202.0
18	03-11	158.0	28.0	779.0	62.0	0.0	1027.0	835.0
19	03-12	66.0	31.0	50.0	8.0	1.0	156.0	835.0
20	03-13	107.0	37.0	31.0	5.0	1.0	181.0	1188.0
21	03-14	40.0	9.0	133.0	28.0	0.0	210.0	125.0
22	03-15	77.0	13.0	162.0	29.0	1.0	282.0	1357.0
23	03-16	429.0	35.0	117.0	35.0	0.0	616.0	835.0
24	03-17	95.0	11.0	131.0	33.0	3.0	273.0	366.0
25	03-18	136.0	13.0	273.0	45.0	12.0	479.0	835.0
26	03-19	98.0	17.0	209.0	52.0	0.0	376.0	702.0
27	03-20	3.0	2.0	395.0	110.0	0.0	510.0	137.0
28	03-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29	03-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30	03-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31	03-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0
32	03-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
33	03-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0
34	03-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0
35	03-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0
36	03-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0
37	03-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0
38	03-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0
39	04-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40	04-02	0.0	0.0	0.0	0.0	0.0	0.0	0.0
41	04-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0
42	04-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
43	04-05	0.0	0.0	0.0	0.0	0.0	0.0	0.0
44	04-06	0.0	0.0	0.0	0.0	0.0	0.0	0.0
45	04-07	0.0	0.0	0.0	0.0	0.0	0.0	0.0

46	04-08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
47	04-09	0.0	0.0	0.0	0.0	0.0	0.0	0.0
48	04-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0
49	04-11	0.0	0.0	0.0	0.0	0.0	0.0	0.0
50	04-12	0.0	0.0	0.0	0.0	0.0	0.0	0.0
51	04-13	0.0	0.0	0.0	0.0	0.0	0.0	0.0
52	04-14	0.0	0.0	0.0	0.0	0.0	0.0	0.0
53	04-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
54	04-16	0.0	0.0	0.0	0.0	0.0	0.0	0.0
55	04-17	0.0	0.0	0.0	0.0	0.0	0.0	0.0
56	04-18	0.0	0.0	0.0	0.0	0.0	0.0	0.0
57	04-19	0.0	0.0	0.0	0.0	0.0	0.0	0.0
58	04-20	0.0	0.0	0.0	0.0	0.0	0.0	0.0
59	04-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0
60	04-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0
61	04-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0
62	04-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0
63	04-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
64	04-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0
65	04-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0
66	04-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0
67	04-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0
68	04-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69	05-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0
70	05-02	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71	05-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72	05-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0
73	05-05	0.0	0.0	0.0	0.0	0.0	0.0	0.0
74	05-06	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75	05-07	0.0	0.0	0.0	0.0	0.0	0.0	0.0

76	05-08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
77	05-09	0.0	0.0	0.0	0.0	0.0	0.0	0.0
78	05-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79	05-11	0.0	0.0	0.0	0.0	0.0	0.0	0.0
80	05-12	0.0	0.0	0.0	0.0	0.0	0.0	0.0
81	05-13	0.0	0.0	0.0	0.0	0.0	0.0	0.0
82	05-14	0.0	0.0	0.0	0.0	0.0	0.0	0.0
83	05-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0
84	05-16	0.0	0.0	0.0	0.0	0.0	0.0	0.0
85	05-17	0.0	0.0	0.0	0.0	0.0	0.0	0.0
86	05-18	0.0	0.0	0.0	0.0	0.0	0.0	0.0
87	05-19	0.0	0.0	0.0	0.0	0.0	0.0	0.0
88	05-20	0.0	0.0	0.0	0.0	0.0	0.0	0.0
89	05-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0
90	05-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0
91	05-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0
92	05-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0
93	05-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0
94	05-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0
95	05-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0
96	05-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0
97	05-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0
98	05-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0
99	05-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0
100	06-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [4]:
```
df.columns
```

Out[4]:
```
Index(['日期', '小程序（浏览量）', '小程序（访客量）', '淘宝（浏览量）', '淘宝（访客量）', '掌上大学', '总计',
       '文章的阅读量'],
      dtype='object')
```

In [5]:
```
# 각 컬럼끼리의 상관계수를 보는 법!
df.corr()
```

Out[5]:

	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量
小程序（浏览量）	1.000000	0.856721	0.460295	0.457011	0.186627	0.762425	0.396717
小程序（访客量）	0.856721	1.000000	0.503833	0.475004	0.191237	0.746636	0.534300
淘宝（浏览量）	0.460295	0.503833	1.000000	0.859718	0.170927	0.922250	0.485815
淘宝（访客量）	0.457011	0.475004	0.859718	1.000000	0.214085	0.847181	0.544294
掌上大学	0.186627	0.191237	0.170927	0.214085	1.000000	0.235852	0.146179
总计	0.762425	0.746636	0.922250	0.847181	0.235852	1.000000	0.543838
文章的阅读量	0.396717	0.534300	0.485815	0.544294	0.146179	0.543838	1.000000

In [6]:
```
#grid_ndf = sns.pairplot(df)
#plt.show()
#plt.close()
```

In [7]:
```
#컬럼 삭제
df_new = df
df_new = df_new.drop(['文章的阅读量'],axis=1)
df_new = df_new.drop(['总计'],axis=1)
df_new
```

Out[7]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学
0	02-21	16.0	4.0	184.0	23.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0
4	02-25	3.0	1.0	102.0	31.0	0.0
5	02-26	13.0	4.0	127.0	28.0	0.0
6	02-27	4.0	2.0	79.0	22.0	0.0
7	02-28	73.0	12.0	123.0	28.0	40.0
8	03-01	33.0	18.0	143.0	33.0	6.0
9	03-02	20.0	4.0	158.0	32.0	0.0

10	03-03	45.0	14.0	142.0	33.0	1.0
11	03-04	28.0	10.0	154.0	34.0	0.0
12	03-05	12.0	6.0	50.0	17.0	0.0
13	03-06	2.0	2.0	46.0	15.0	0.0
14	03-07	193.0	27.0	34.0	9.0	1.0
15	03-08	115.0	35.0	91.0	16.0	0.0
16	03-09	221.0	35.0	150.0	29.0	0.0
17	03-10	188.0	32.0	124.0	20.0	4.0
18	03-11	158.0	28.0	779.0	62.0	0.0
19	03-12	66.0	31.0	50.0	8.0	1.0
20	03-13	107.0	37.0	31.0	5.0	1.0
21	03-14	40.0	9.0	133.0	28.0	0.0
22	03-15	77.0	13.0	162.0	29.0	1.0
23	03-16	429.0	35.0	117.0	35.0	0.0
24	03-17	95.0	11.0	131.0	33.0	3.0
25	03-18	136.0	13.0	273.0	45.0	12.0
26	03-19	98.0	17.0	209.0	52.0	0.0
27	03-20	3.0	2.0	395.0	110.0	0.0
28	03-21	0.0	0.0	0.0	0.0	0.0
29	03-22	0.0	0.0	0.0	0.0	0.0
30	03-23	0.0	0.0	0.0	0.0	0.0
31	03-24	0.0	0.0	0.0	0.0	0.0
32	03-25	0.0	0.0	0.0	0.0	0.0
33	03-26	0.0	0.0	0.0	0.0	0.0
34	03-27	0.0	0.0	0.0	0.0	0.0
35	03-28	0.0	0.0	0.0	0.0	0.0
36	03-29	0.0	0.0	0.0	0.0	0.0
37	03-30	0.0	0.0	0.0	0.0	0.0
38	03-31	0.0	0.0	0.0	0.0	0.0
39	04-01	0.0	0.0	0.0	0.0	0.0

40	04-02	0.0	0.0	0.0	0.0	0.0
41	04-03	0.0	0.0	0.0	0.0	0.0
42	04-04	0.0	0.0	0.0	0.0	0.0
43	04-05	0.0	0.0	0.0	0.0	0.0
44	04-06	0.0	0.0	0.0	0.0	0.0
45	04-07	0.0	0.0	0.0	0.0	0.0
46	04-08	0.0	0.0	0.0	0.0	0.0
47	04-09	0.0	0.0	0.0	0.0	0.0
48	04-10	0.0	0.0	0.0	0.0	0.0
49	04-11	0.0	0.0	0.0	0.0	0.0
50	04-12	0.0	0.0	0.0	0.0	0.0
51	04-13	0.0	0.0	0.0	0.0	0.0
52	04-14	0.0	0.0	0.0	0.0	0.0
53	04-15	0.0	0.0	0.0	0.0	0.0
54	04-16	0.0	0.0	0.0	0.0	0.0
55	04-17	0.0	0.0	0.0	0.0	0.0
56	04-18	0.0	0.0	0.0	0.0	0.0
57	04-19	0.0	0.0	0.0	0.0	0.0
58	04-20	0.0	0.0	0.0	0.0	0.0
59	04-21	0.0	0.0	0.0	0.0	0.0
60	04-22	0.0	0.0	0.0	0.0	0.0
61	04-23	0.0	0.0	0.0	0.0	0.0
62	04-24	0.0	0.0	0.0	0.0	0.0
63	04-25	0.0	0.0	0.0	0.0	0.0
64	04-26	0.0	0.0	0.0	0.0	0.0
65	04-27	0.0	0.0	0.0	0.0	0.0
66	04-28	0.0	0.0	0.0	0.0	0.0
67	04-29	0.0	0.0	0.0	0.0	0.0
68	04-30	0.0	0.0	0.0	0.0	0.0
69	05-01	0.0	0.0	0.0	0.0	0.0

70	05-02	0.0	0.0	0.0	0.0	0.0
71	05-03	0.0	0.0	0.0	0.0	0.0
72	05-04	0.0	0.0	0.0	0.0	0.0
73	05-05	0.0	0.0	0.0	0.0	0.0
74	05-06	0.0	0.0	0.0	0.0	0.0
75	05-07	0.0	0.0	0.0	0.0	0.0
76	05-08	0.0	0.0	0.0	0.0	0.0
77	05-09	0.0	0.0	0.0	0.0	0.0
78	05-10	0.0	0.0	0.0	0.0	0.0
79	05-11	0.0	0.0	0.0	0.0	0.0
80	05-12	0.0	0.0	0.0	0.0	0.0
81	05-13	0.0	0.0	0.0	0.0	0.0
82	05-14	0.0	0.0	0.0	0.0	0.0
83	05-15	0.0	0.0	0.0	0.0	0.0
84	05-16	0.0	0.0	0.0	0.0	0.0
85	05-17	0.0	0.0	0.0	0.0	0.0
86	05-18	0.0	0.0	0.0	0.0	0.0
87	05-19	0.0	0.0	0.0	0.0	0.0
88	05-20	0.0	0.0	0.0	0.0	0.0
89	05-21	0.0	0.0	0.0	0.0	0.0
90	05-22	0.0	0.0	0.0	0.0	0.0
91	05-23	0.0	0.0	0.0	0.0	0.0
92	05-24	0.0	0.0	0.0	0.0	0.0
93	05-25	0.0	0.0	0.0	0.0	0.0
94	05-26	0.0	0.0	0.0	0.0	0.0
95	05-27	0.0	0.0	0.0	0.0	0.0
96	05-28	0.0	0.0	0.0	0.0	0.0
97	05-29	0.0	0.0	0.0	0.0	0.0
98	05-30	0.0	0.0	0.0	0.0	0.0
99	05-31	0.0	0.0	0.0	0.0	0.0

100	06-01	0.0	0.0	0.0	0.0	0.0

In [8]:

```
#인덱스 삭제
df_new = df_new.drop(df.index[30:]) # 03-22기준
df_new
```

Out[8]:

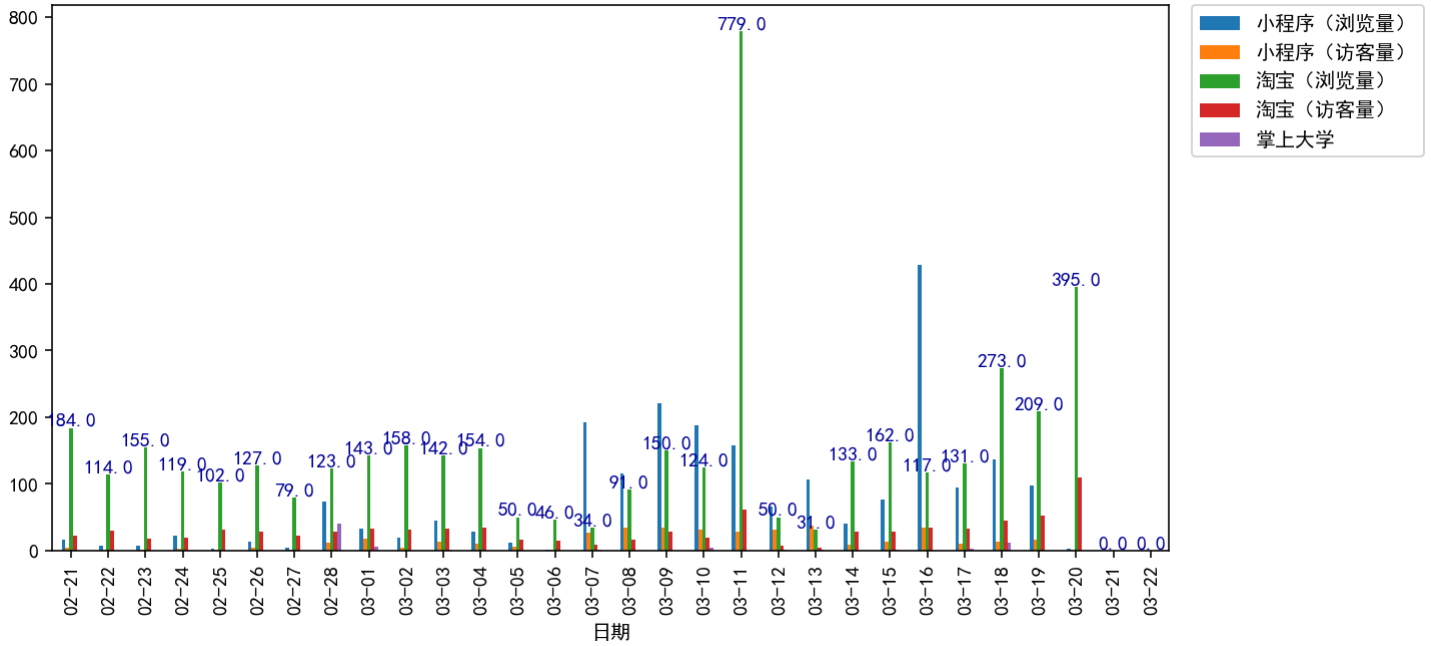
	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学
0	02-21	16.0	4.0	184.0	23.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0
4	02-25	3.0	1.0	102.0	31.0	0.0
5	02-26	13.0	4.0	127.0	28.0	0.0
6	02-27	4.0	2.0	79.0	22.0	0.0
7	02-28	73.0	12.0	123.0	28.0	40.0
8	03-01	33.0	18.0	143.0	33.0	6.0
9	03-02	20.0	4.0	158.0	32.0	0.0
10	03-03	45.0	14.0	142.0	33.0	1.0
11	03-04	28.0	10.0	154.0	34.0	0.0
12	03-05	12.0	6.0	50.0	17.0	0.0
13	03-06	2.0	2.0	46.0	15.0	0.0
14	03-07	193.0	27.0	34.0	9.0	1.0
15	03-08	115.0	35.0	91.0	16.0	0.0
16	03-09	221.0	35.0	150.0	29.0	0.0
17	03-10	188.0	32.0	124.0	20.0	4.0
18	03-11	158.0	28.0	779.0	62.0	0.0
19	03-12	66.0	31.0	50.0	8.0	1.0
20	03-13	107.0	37.0	31.0	5.0	1.0
21	03-14	40.0	9.0	133.0	28.0	0.0
22	03-15	77.0	13.0	162.0	29.0	1.0
23	03-16	429.0	35.0	117.0	35.0	0.0

24	03-17	95.0	11.0	131.0	33.0	3.0
25	03-18	136.0	13.0	273.0	45.0	12.0
26	03-19	98.0	17.0	209.0	52.0	0.0
27	03-20	3.0	2.0	395.0	110.0	0.0
28	03-21	0.0	0.0	0.0	0.0	0.0
29	03-22	0.0	0.0	0.0	0.0	0.0

판다스로 데이터 시각화

In [9]:

```
# plt 크기 설정!
plt.rcParams["figure.figsize"] = (10, 5)

#막대 그래프!
df_new.plot(x='日期',kind='bar')
plt.legend(bbox_to_anchor=(1.02,1), loc=2, borderaxespad=0.)  #레전드값을 바깥에 그릴수 있게 하는 방법

# 그래프에 수치 표시하는 방법!
for i , value in enumerate(df_new['淘宝（浏览量）']):
    plt.text(i,value,'%s'% value, fontsize=10, color='#000099',ha='center',va='bottom')
```

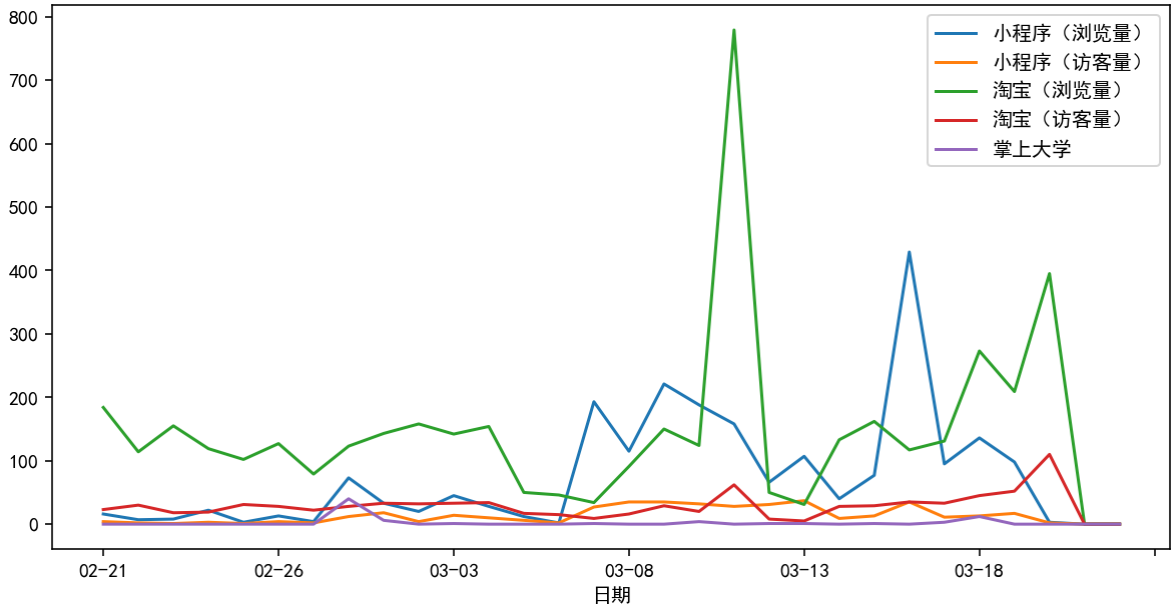


In [10]:

```
# 선그래프
```

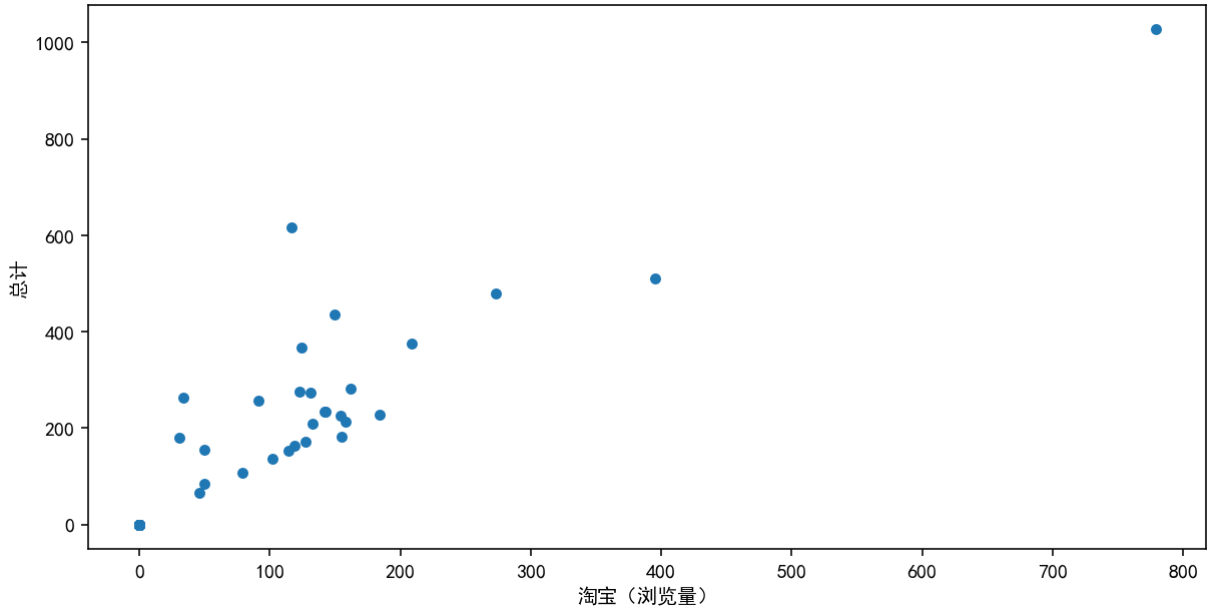
```
plt.rcParams["figure.figsize"] = (10, 5)
df_new.plot(x='日期')
```

Out[10]: <AxesSubplot:xlabel='日期'>



In [11]:
```
#산점도
df.plot(x='淘宝（浏览量）',y='总计', kind='scatter')
```

Out[11]: <AxesSubplot:xlabel='淘宝（浏览量）', ylabel='总计'>

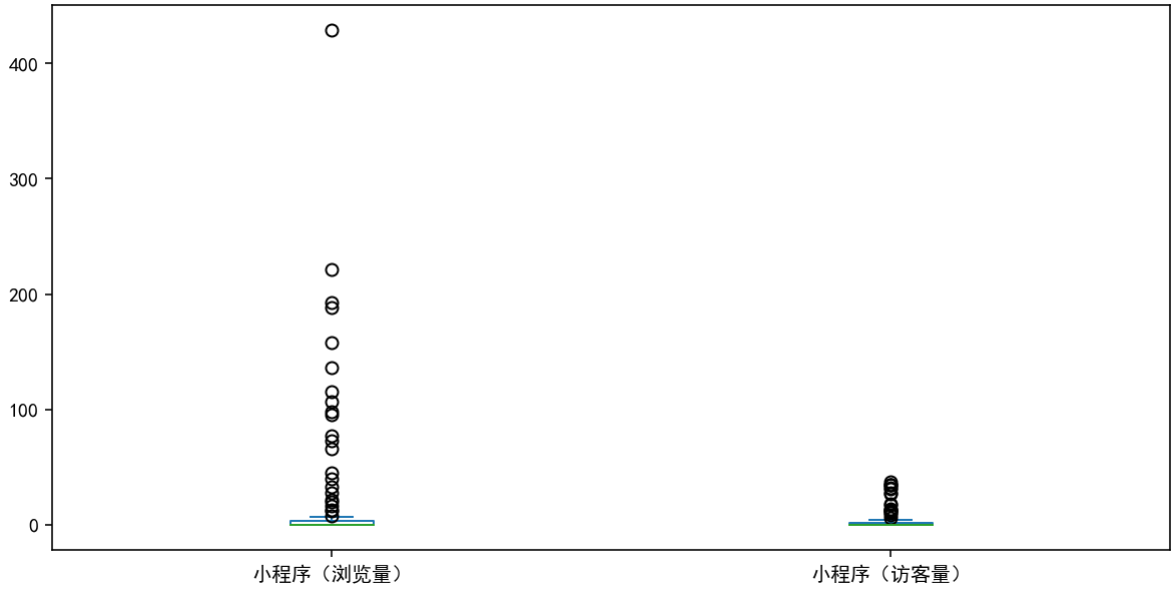


In [12]:

```
# 박스플롯
df[['小程序（浏览量）','小程序（访客量）']].plot(kind='box')
```

Out[12]:

```
<AxesSubplot:>
```

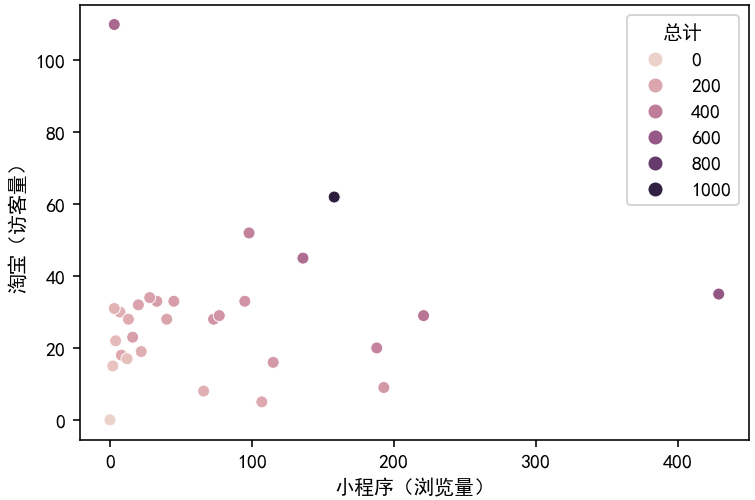


Seaborn 데이터 시각화

In [13]:
```
# regplot은 못 그리겠음...
```

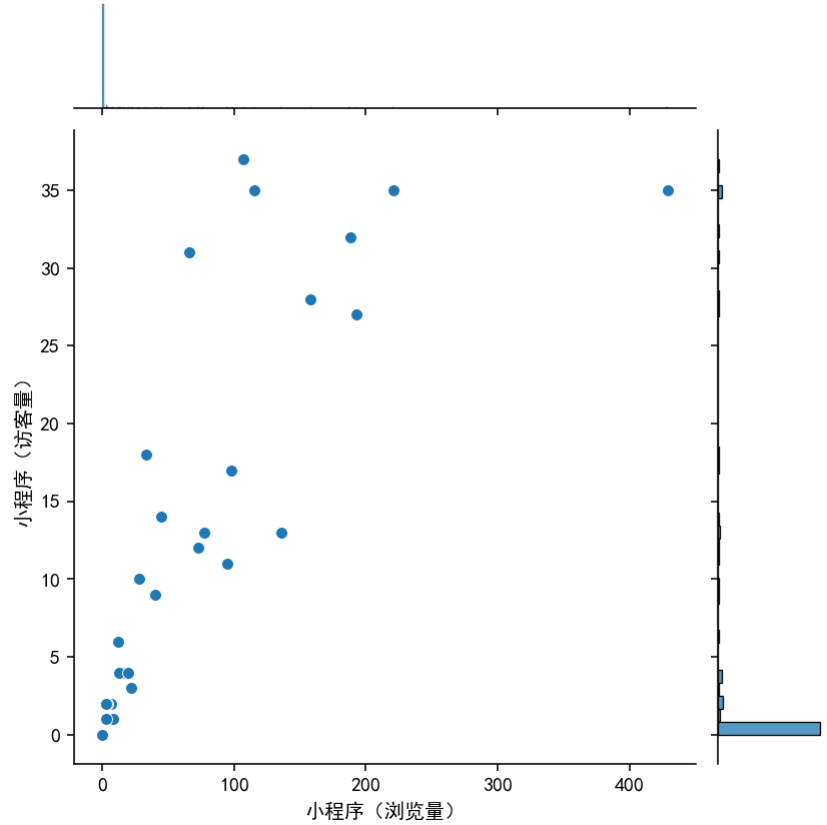
In [61]:
```
#Scatterplot
sns.scatterplot(data=df, x='小程序（浏览量）', y='淘宝（访客量）', hue='总计', sizes=(50,330))
```

Out[61]:
```
<AxesSubplot:xlabel='小程序（浏览量）', ylabel='淘宝（访客量）'>
```



In [14]:
```
#seaborn 데이터 시각화
#jointplot
ax = sns.jointplot(x='小程序（浏览量）', y='小程序（访客量）', data =df , kind='scatter')
ax.set_axis_labels(xlabel='小程序（浏览量）',ylabel='小程序（访客量）')
```

Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x2a300e7bd60>
```

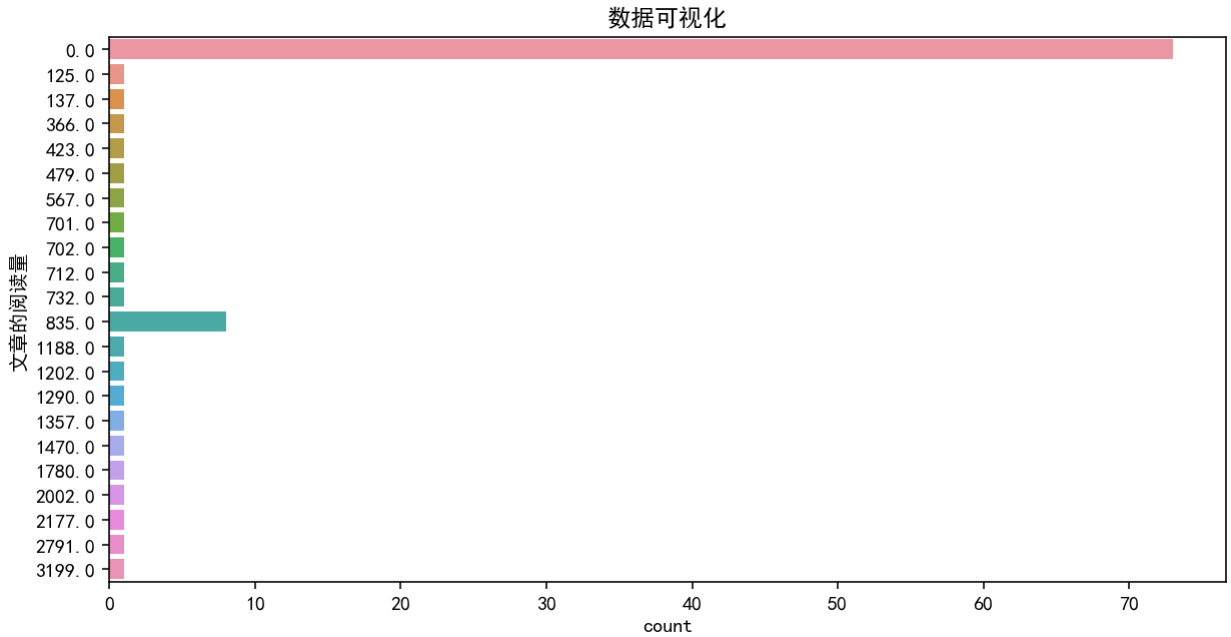


In [ ]:
```
# pointplot
plt.figure(figsize=(18,7))
sns.pointplot(data=df, x='小程序（浏览量）', y='淘宝（访客量）', hue='总计')
```

Out[ ]:
```
<AxesSubplot:xlabel='小程序（浏览量）', ylabel='淘宝（访客量）'>
```

In [15]:
```
#seaborn
# countplot은 적합하지 않다.
ax =sns.countplot(y='文章的阅读量', data =df)
ax.set_title('数据可视化')
```

Out[15]:
```
Text(0.5, 1.0, '数据可视化')
```

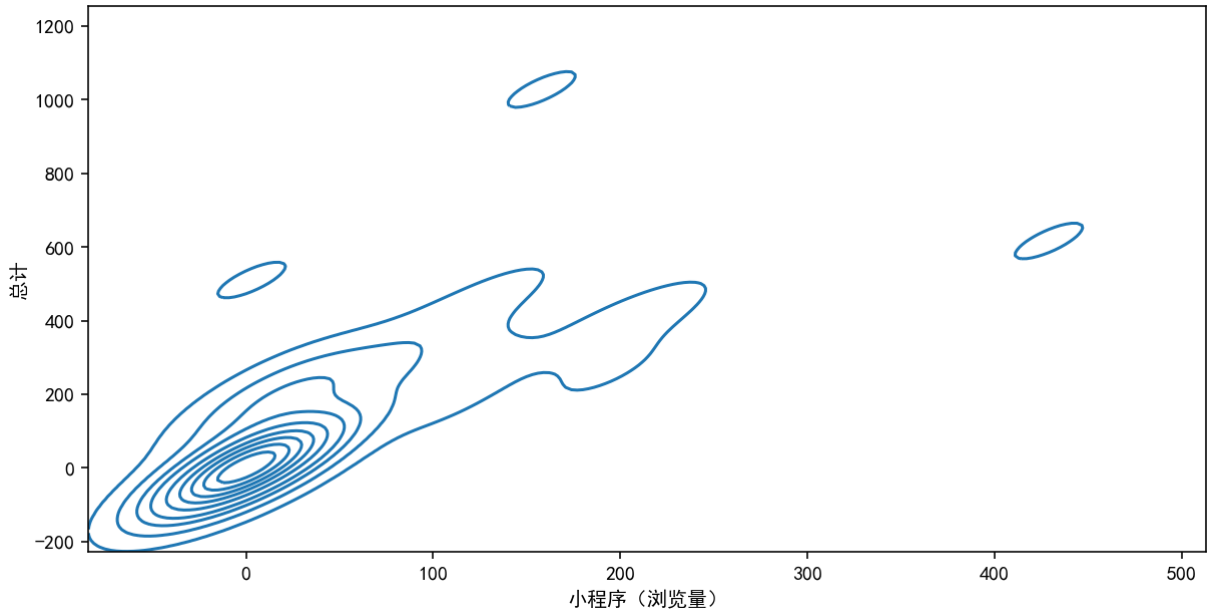


In [16]:

```
#seaborn kdeplot()
sns.kdeplot(x='小程序（浏览量）', y='总计', data =df)
```

Out[16]:

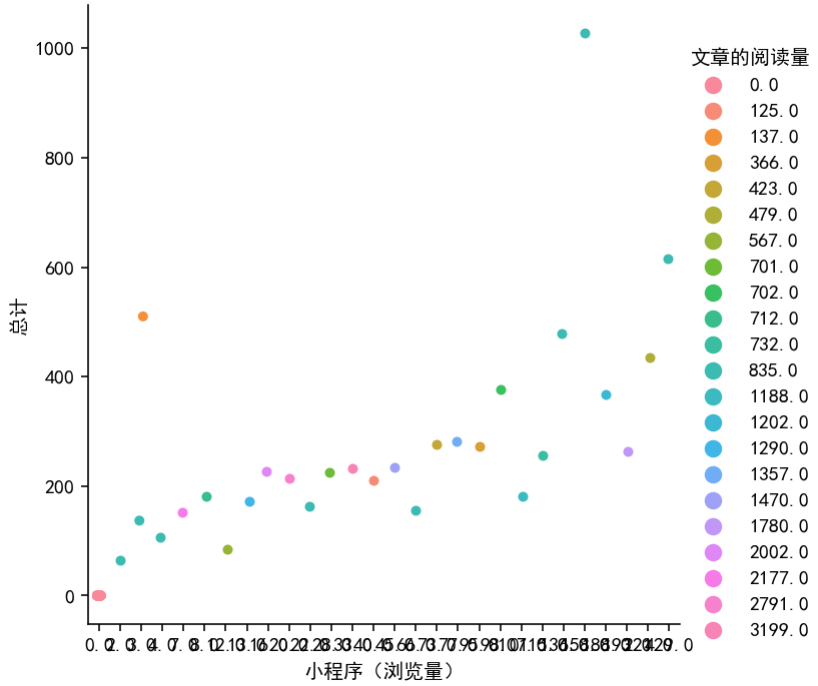
```
<AxesSubplot:xlabel='小程序（浏览量）', ylabel='总计'>
```



In [17]:

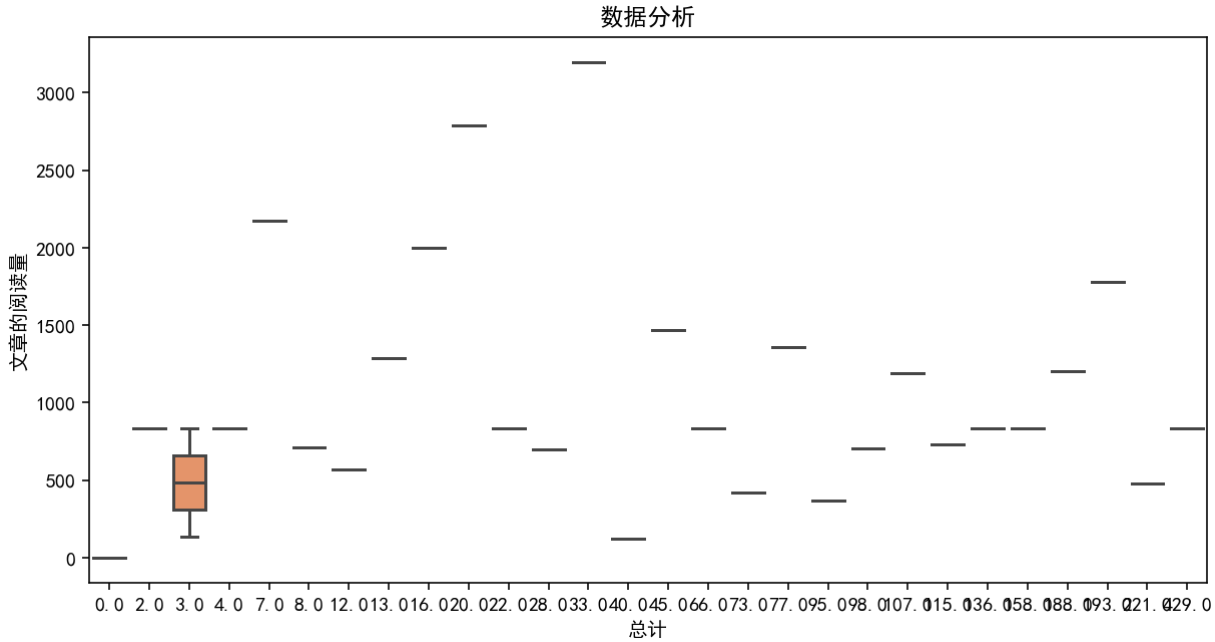
```
# 3개의 컬럼을 비교가능
# catplot
sns.catplot(x='小程序（浏览量）', y='总计',hue='文章的阅读量', data =df)
```

Out[17]:
```
<seaborn.axisgrid.FacetGrid at 0x2a3021abf10>
```



In [18]:
```
#boxplot

ax = plt.subplots()
ax = sns.boxplot(x='小程序（浏览量）', y='文章的阅读量',  data=df)
ax.set_title('数据分析')
ax.set_xlabel('总计')
ax.set_ylabel('文章的阅读量')
```

Out[18]:
```
Text(0, 0.5, '文章的阅读量')
```

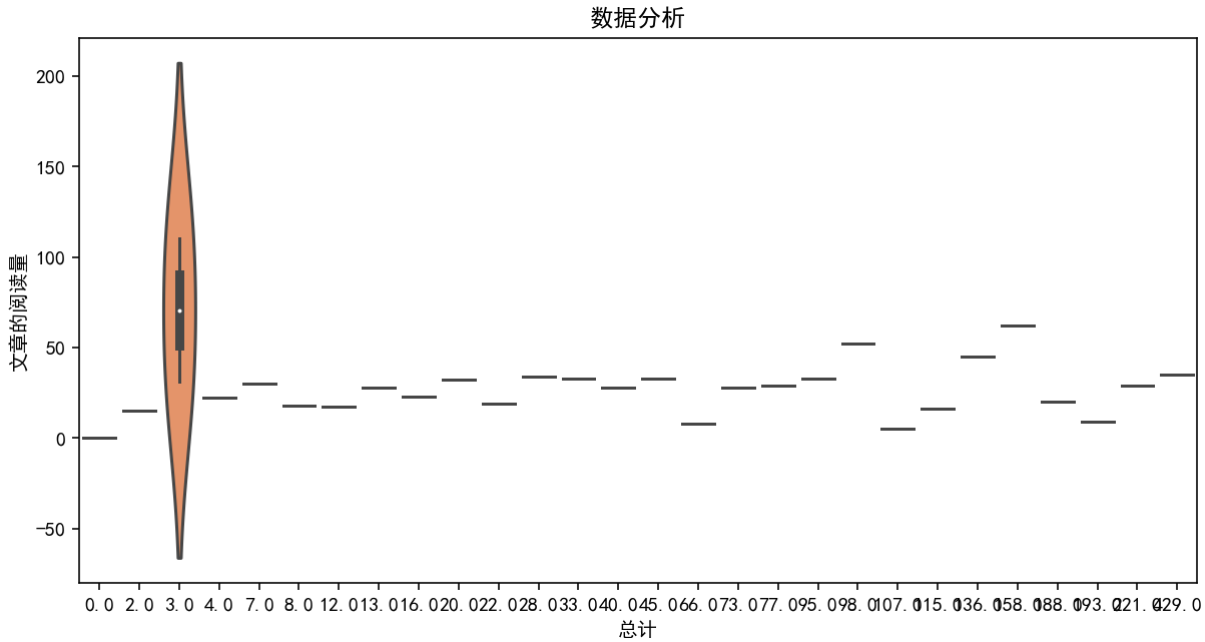


In [19]:

```
# violinplot

ax=plt.subplots()
ax=sns.violinplot(x='小程序（浏览量）', y = '淘宝（访客量）', data=df)
ax.set_title('数据分析')
ax.set_xlabel('总计')
ax.set_ylabel('文章的阅读量')
```

Out[19]:

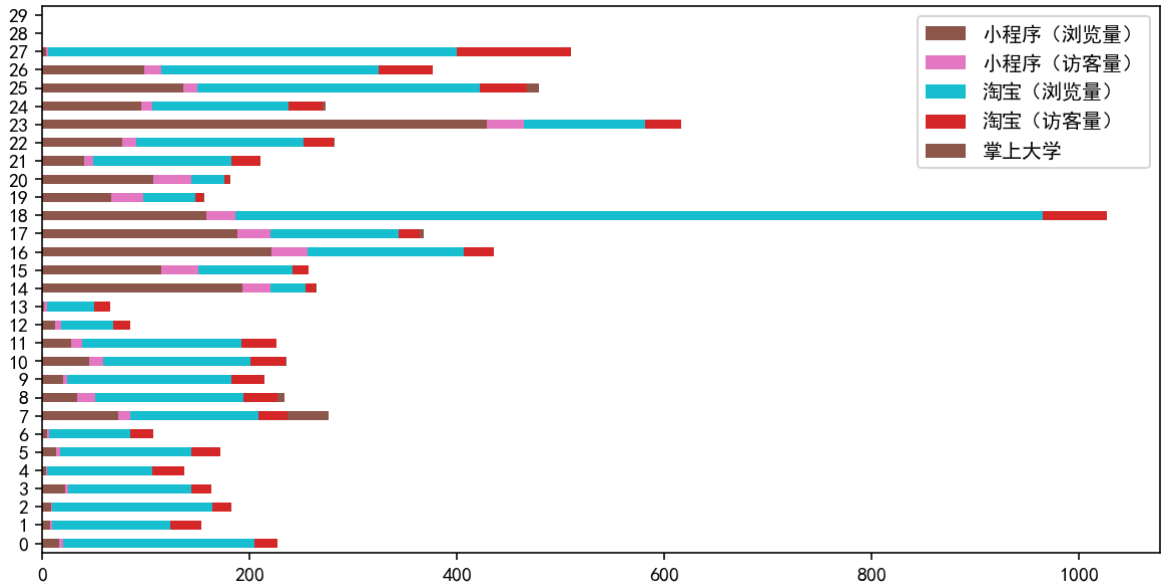
```
Text(0, 0.5, '文章的阅读量')
```



Matplotlib 시각화

In [20]:
```
df_new.plot(kind='barh',color=['tab:brown','tab:pink','tab:cyan','tab:red'],stacked=True)
```

Out[20]:
```
<AxesSubplot:>
```

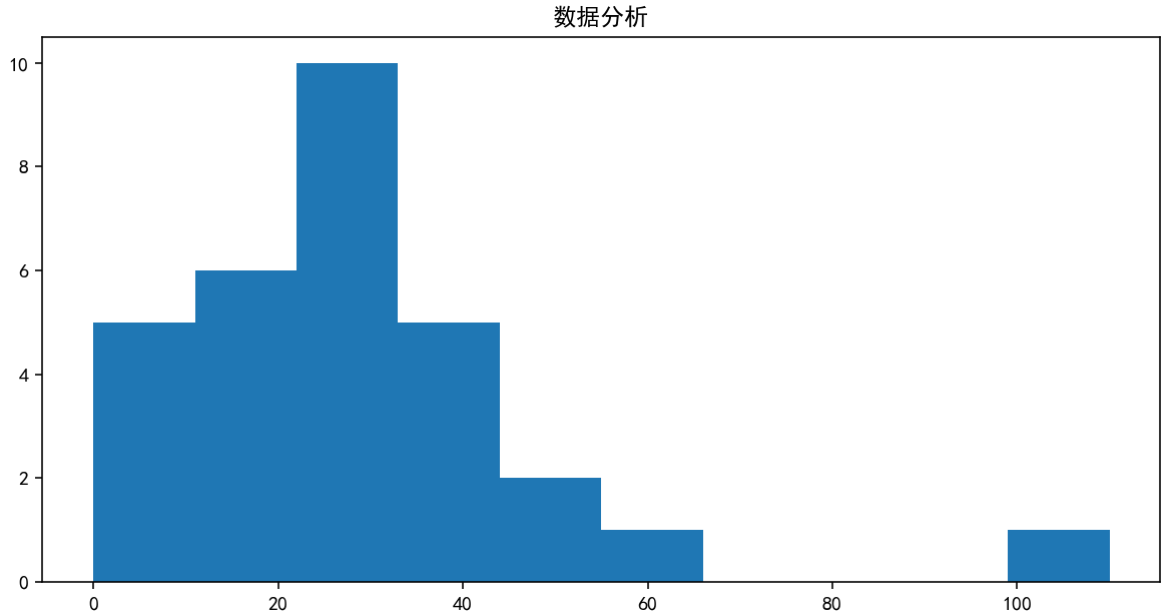


In [21]:

```
# Histogram
fig = plt.figure()
axes1 = fig.add_subplot(1,1,1)
axes1.hist(df_new['淘宝（访客量）'], bins =10)
axes1.set_title('数据分析')
```

Out[21]:

```
Text(0.5, 1.0, '数据分析')
```

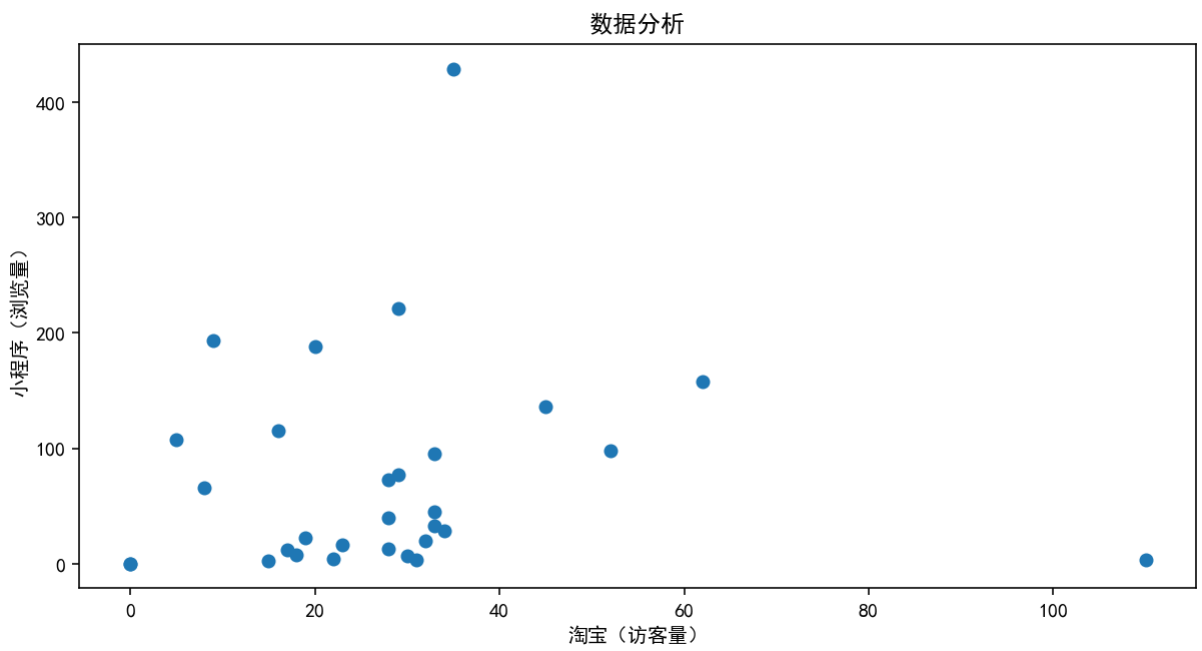


In [22]:

```
# Scatterplot
scatter_plot = plt.figure()
axes1 = scatter_plot.add_subplot(1,1,1)
axes1.scatter(df_new['淘宝（访客量）'], df_new['小程序（浏览量）'])
axes1.set_title('数据分析')
axes1.set_xlabel('淘宝（访客量）')
axes1.set_ylabel('小程序（浏览量）')
```

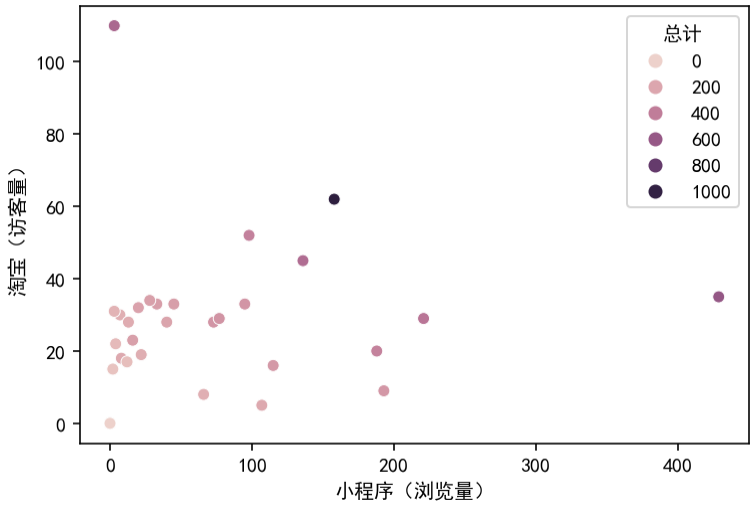
Out[22]:

```
Text(0, 0.5, '小程序（浏览量）')
```



In [60]:

Out[60]: <AxesSubplot:xlabel='小程序（浏览量）', ylabel='淘宝（访客量）'>



In [23]:
```
#axes1.boxplot(df['淘宝（访客量）'],df['小程序（浏览量）'])
```

단순회귀로 상관관계 분석해보기

利用机器学习单纯回归分析的数据相关关系分析

In [24]:

```
#import pandas as pd
#import matplotlib.pyplot as plt
#import numpy as np
#import seaborn as sns
#import matplotlib.pyplot as plt
#%matplotlib inline
# 모든 데이터를 볼 수 있는 ( head랑 tail만 보여주는게 아니라, 다 보여주는 방법)
#pd.set_option('display.max_columns', None)
#pd.set_option('display.max_rows', None)

## 중국어 폰트 설정 하는 방법

#import matplotlib.pyplot as plt
#%matplotlib inline

#from matplotlib import font_manager, rc
#plt.rcParams['axes.unicode_minus'] = False
#f_path = "C:\Windows\Fonts/simhei.ttf"
#font_name = font_manager.FontProperties(fname=f_path).get_name()
#rc('font', family=font_name)

# 폰트를 선명하게 하기
# retina설정
#from IPython.display import set_matplotlib_formats
#set_matplotlib_formats("retina")

#data_name = f"浏览量和访问量跟文章阅读量的关系.csv"
#df = pd.read_csv(data_name) #Encoding cp949는 못함

# Nan 값 0으로 만들어주기
#df = df.fillna(value=0)
#df.head()
```

In [25]:

```
#df =df.drop(['Unnamed: 9'],axis=1)
#df =df.drop(['Unnamed: 10'],axis=1)
#df =df.drop(['Unnamed: 11'],axis=1)
#df =df.drop(['Unnamed: 12'],axis=1)
#df =df.drop(['Unnamed: 13'],axis=1)
#df =df.drop(['Unnamed: 14'],axis=1)
#df =df.drop(['Unnamed: 15'],axis=1)
#df =df.drop(['Unnamed: 16'],axis=1)
#df =df.drop(['Unnamed: 17'],axis=1)
#df =df.drop(['Unnamed: 18'],axis=1)
#df =df.drop(['Unnamed: 19'],axis=1)
```

```
#df =df.drop(['Unnamed: 20'],axis=1)
#df =df.drop(['Unnamed: 21'],axis=1)
#df =df.drop(['Unnamed: 22'],axis=1)
#df =df.drop(['Unnamed: 23'],axis=1)
#df =df.drop(['Unnamed: 24'],axis=1)
#df =df.drop(['淘宝特点活动'],axis=1)
#df
# df = df_new.drop(df.index[30:]) # 03-22기준
```

In [26]:

```
df.drop(df.index[28:])
```

Out[26]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量
0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0
4	02-25	3.0	1.0	102.0	31.0	0.0	137.0	835.0
5	02-26	13.0	4.0	127.0	28.0	0.0	172.0	1290.0
6	02-27	4.0	2.0	79.0	22.0	0.0	107.0	835.0
7	02-28	73.0	12.0	123.0	28.0	40.0	276.0	423.0
8	03-01	33.0	18.0	143.0	33.0	6.0	233.0	3199.0
9	03-02	20.0	4.0	158.0	32.0	0.0	214.0	2791.0
10	03-03	45.0	14.0	142.0	33.0	1.0	235.0	1470.0
11	03-04	28.0	10.0	154.0	34.0	0.0	226.0	701.0
12	03-05	12.0	6.0	50.0	17.0	0.0	85.0	567.0
13	03-06	2.0	2.0	46.0	15.0	0.0	65.0	835.0
14	03-07	193.0	27.0	34.0	9.0	1.0	264.0	1780.0
15	03-08	115.0	35.0	91.0	16.0	0.0	257.0	732.0
16	03-09	221.0	35.0	150.0	29.0	0.0	435.0	479.0
17	03-10	188.0	32.0	124.0	20.0	4.0	368.0	1202.0
18	03-11	158.0	28.0	779.0	62.0	0.0	1027.0	835.0
19	03-12	66.0	31.0	50.0	8.0	1.0	156.0	835.0
20	03-13	107.0	37.0	31.0	5.0	1.0	181.0	1188.0

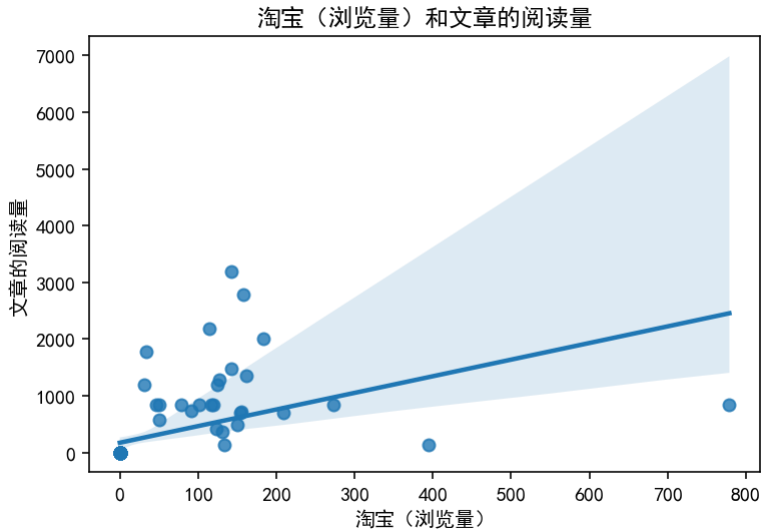
21	03-14	40.0	9.0	133.0	28.0	0.0	210.0	125.0
22	03-15	77.0	13.0	162.0	29.0	1.0	282.0	1357.0
23	03-16	429.0	35.0	117.0	35.0	0.0	616.0	835.0
24	03-17	95.0	11.0	131.0	33.0	3.0	273.0	366.0
25	03-18	136.0	13.0	273.0	45.0	12.0	479.0	835.0
26	03-19	98.0	17.0	209.0	52.0	0.0	376.0	702.0
27	03-20	3.0	2.0	395.0	110.0	0.0	510.0	137.0

In [57]:

```
ax=plt.subplot()
ax=sns.regplot(x='淘宝（浏览量）', y='文章的阅读量', data = df)
ax.set_title('淘宝（浏览量）和文章的阅读量')
ax.set_xlabel('淘宝（浏览量）')
ax.set_ylabel('文章的阅读量')
```

Out[57]:

```
Text(0, 0.5, '文章的阅读量')
```

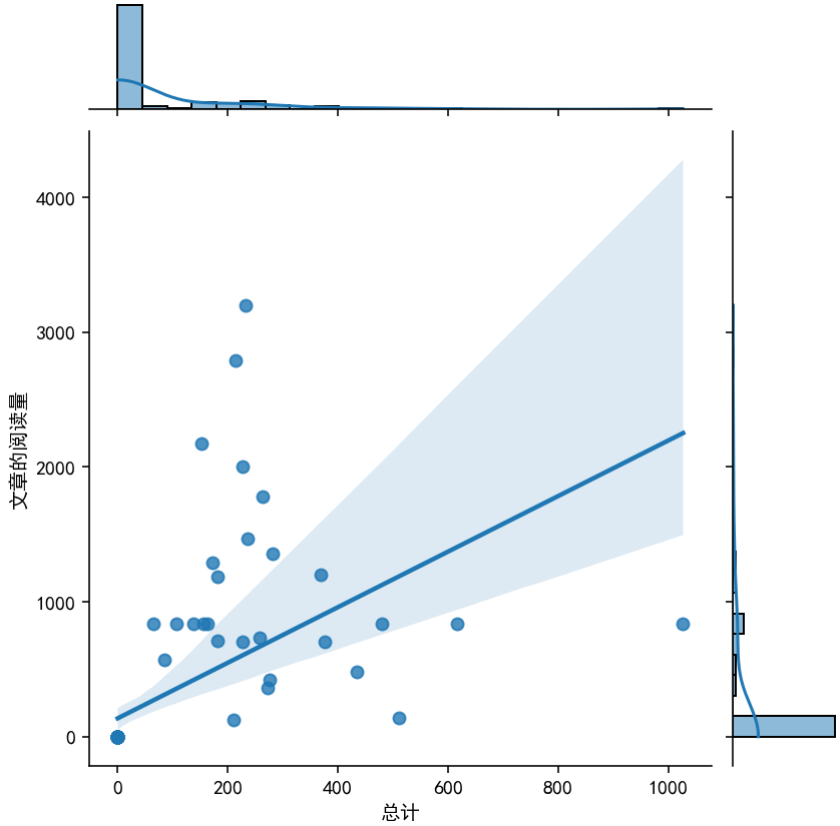


In [28]:

```
# 산점도
sns.jointplot(x='总计',y='文章的阅读量',data=df,kind='reg')
```

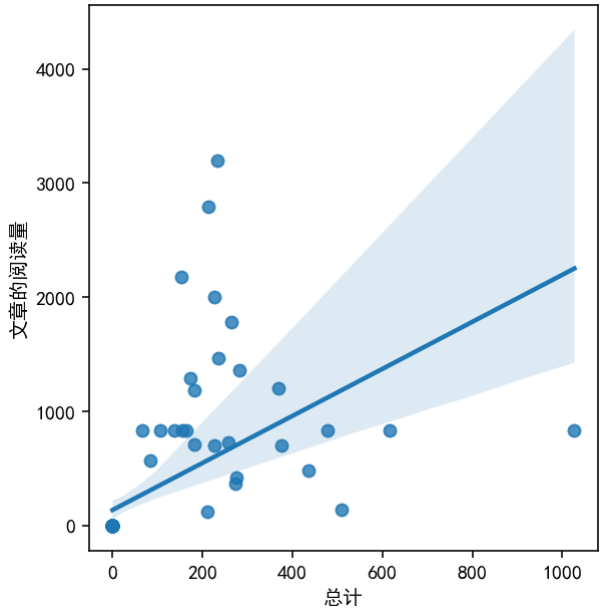
Out[28]:

```
<seaborn.axisgrid.JointGrid at 0x2a302b52040>
```



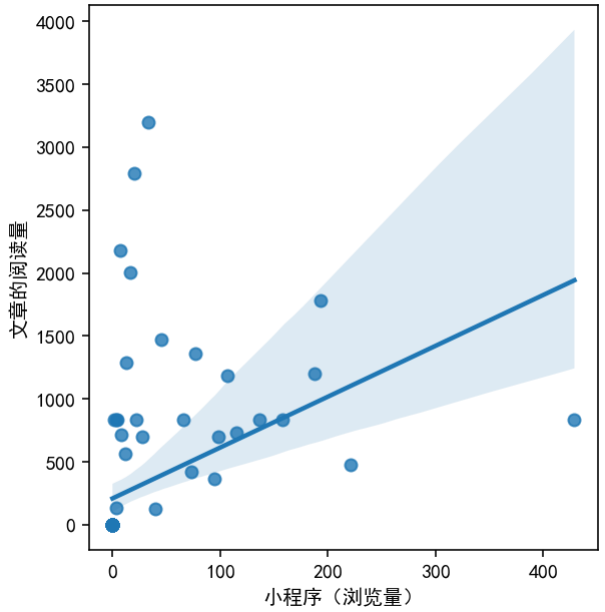
In [29]:

```
# Seaborn 라이브러리의 regplot() 함수를 이용하여 두 변수에 대한 그래프(산점도) 그리기.

fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(1,2,1)
#ax2 = fig.add_subplot(1,2,2)
sns.regplot(x='总计', y='文章的阅读量', data=df, ax=ax1)  # 회귀선 표시
#sns.regplot(x='总计', y='文章的阅读量', data=df, ax=ax2) # 회귀선 표시 안함 표시는 , fit_reg = False

plt.show()
plt.close()
```



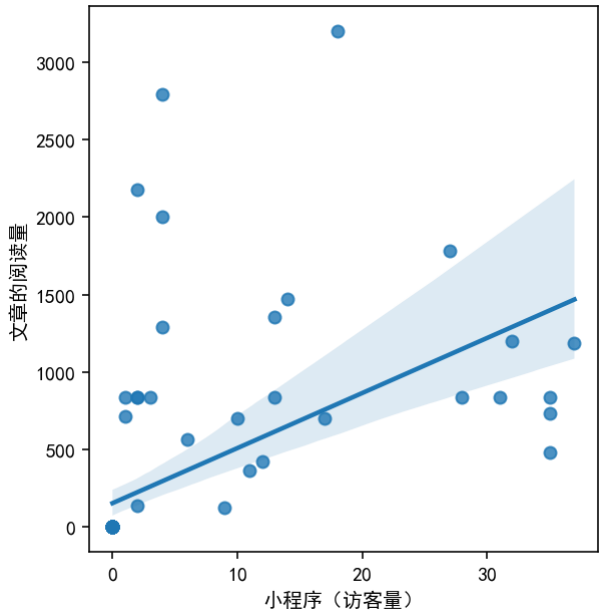
In [30]:

```
# Seaborn 라이브러리의 regplot() 함수를 이용하여 두 변수에 대한 그래프(산점도) 그리기.

fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(1,2,1)
#ax2 = fig.add_subplot(1,2,2)
sns.regplot(x='小程序（浏览量）', y='文章的阅读量', data=df, ax=ax1)  # 회귀선 표시
#sns.regplot(x='总计', y='文章的阅读量', data=df, ax=ax2) # 회귀선 표시 안함 표시는 , fit_reg = False

plt.show()
plt.close()
```



In [31]:

```
# Seaborn 라이브러리의 regplot() 함수를 이용하여 두 변수에 대한 그래프(산점도) 그리기.

fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(1,2,1)
#ax2 = fig.add_subplot(1,2,2)
sns.regplot(x='小程序（访客量）', y='文章的阅读量', data=df, ax=ax1)  # 회귀선 표시
#sns.regplot(x='总计', y='文章的阅读量', data=df, ax=ax2) # 회귀선 표시 안함 표시는 , fit_reg = False

plt.show()
plt.close()
```



In [32]:

```
ndf =df
```

In [33]:

```
X=ndf[['文章的阅读量']] # 독립 변수 X
y=ndf[['总计']]    # 종속 변수 Y

# train data와 test data로 구분(7:3 비율)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,             # 독립 변수
                                                    y,             # 종속 변수
                                                    test_size=0.3,  # 검증 30%
                                                    random_state=10 # 랜덤 추출 값
                                                    )

print('train data 개수 : ', len(X_train))
print('testdata 개수 : ', len(X_test))
```

```
train data 개수 :  70
testdata 개수 :  31
```

In [34]:

```
# sklearn 라이브러리에서 선형회귀분석 모듈 가져오기
from sklearn.linear_model import LinearRegression

# 단순 회귀 분석 모형 객체 생성
lr = LinearRegression()
```

```
# train data를 가지고 모형 학습
lr.fit(X_train, y_train)

# 학습을 마친 모형에 test data를 적용하여 결정계수(R-제곱) 계산
r_square = lr.score(X_test, y_test)

print(r_square)
```

```
0.41815098377398474
```

In [35]:

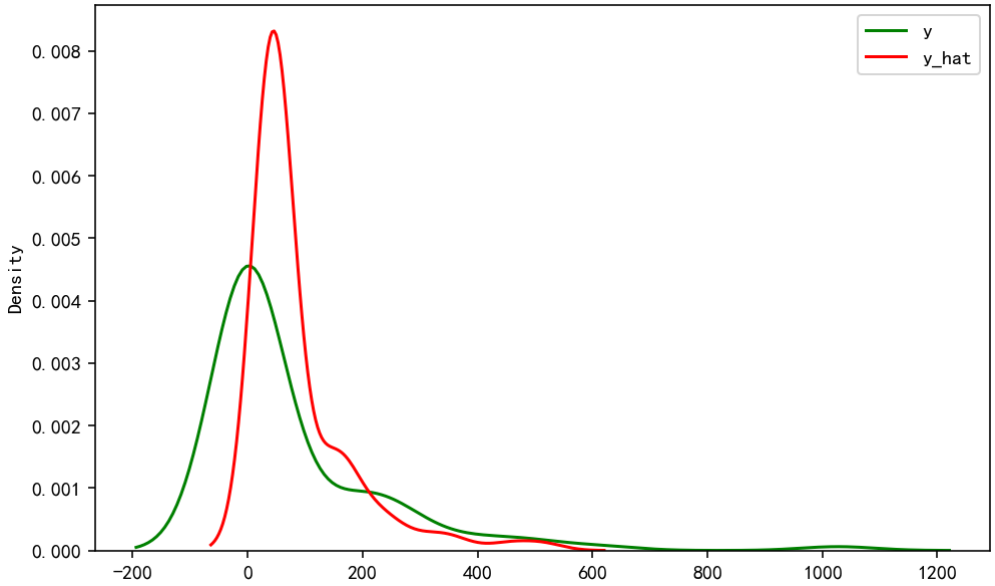
```
# 회귀식의 기울기
print('기울기 : ', lr.coef_)
print('\n')

# 회귀식의 y절편
print('y절편 :', lr.intercept_)
```

```
기울기 :  [[0.14656045]]


y절편 : [43.74416742]
```

In [36]:

```
# 모형에 전체 X 데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교
y_hat = lr.predict(X)

plt.figure(figsize=(8, 5))
ax1 = sns.distplot(y, hist=False, color = 'green',label="y")

ax2 = sns.distplot(y_hat, hist=False, label="y_hat", color ='red', ax=ax1)

plt.legend()

plt.show()
plt.close()
```

```
C:\Users\toyou\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\toyou\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



다항회귀로 상관 관계 분석해보기

In [37]:

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

# 모든 데이터를 볼 수 있는 ( head랑 tail만 보여주는게 아니라, 다 보여주는 방법)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

## 중국어 폰트 설정 하는 방법

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False
f_path = "C:\Windows\Fonts/simhei.ttf"
font_name = font_manager.FontProperties(fname=f_path).get_name()
rc('font', family=font_name)

# 폰트를 선명하게 하기
# retina설정
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")
```

```
#data_name = f"浏览量和访问量跟文章阅读量的关系.csv"
df = pd.read_csv(data_name) #Encoding cp949는 못함

# Nan 값 0으로 만들어주기
df = df.fillna(value=0)
df.head()
```

```
C:\Users\toyou\AppData\Local\Temp/ipykernel_11688/2084237805.py:24: DeprecationWarning: `set_matplotlib_formats` is deprecated since IPython 7.23, directly use `matplotlib_inline.backend_inline.set_matplotlib_formats()`
  set_matplotlib_formats("retina")
```

Out[37]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量	淘宝特点活动	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0	2022.3.4~2022.3.8：情人节	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	02-25	3.0	1.0	102.0	31.0	0.0	137.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [38]:
```
#df = df.reset_index()
```

In [39]:
```
ndf = df
```

In [40]:
```
X = ndf[['总计']]
y = ndf[['文章的阅读量']]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=10)

print('훈련 데이터:', X_train.shape)
print('검증 데이터:', y_test.shape)
```

```
훈련 데이터: (70, 1)
검증 데이터: (31, 1)
```

In [41]:

```python
from sklearn.linear_model import LinearRegression # 선형회귀분석
from sklearn.preprocessing import PolynomialFeatures # 다항식 변환

# 다항식 변환
poly = PolynomialFeatures(degree=2) # 2차항 적용
X_train_poly = poly.fit_transform(X_train) # X_train 데이터를 2차항으로 변형

print('원 데이터: ', X_train.shape)
print('2차항 변환 데이터: ', X_train_poly.shape)
```

```
원 데이터:  (70, 1)
2차항 변환 데이터:  (70, 3)
```

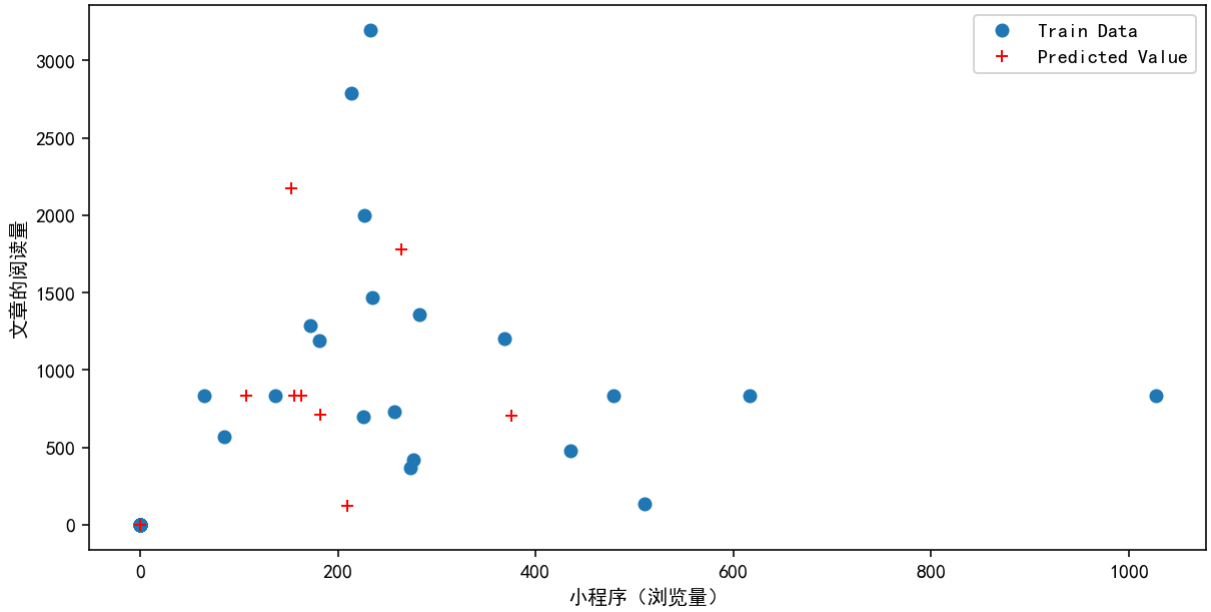
In [42]:

```python
# train data를 가지고 모형 학습
pr = LinearRegression()
pr.fit(X_train_poly, y_train)

# 학습을 마친 모형에 test data를 적용하여 결정계수(R-제곱) 계산
X_test_poly = poly.fit_transform(X_test) # X_test 데이터를 2차항으로 변형
r_square = pr.score(X_test_poly, y_test)
print(r_square)
```

```
0.5773029974960575
```

In [43]:

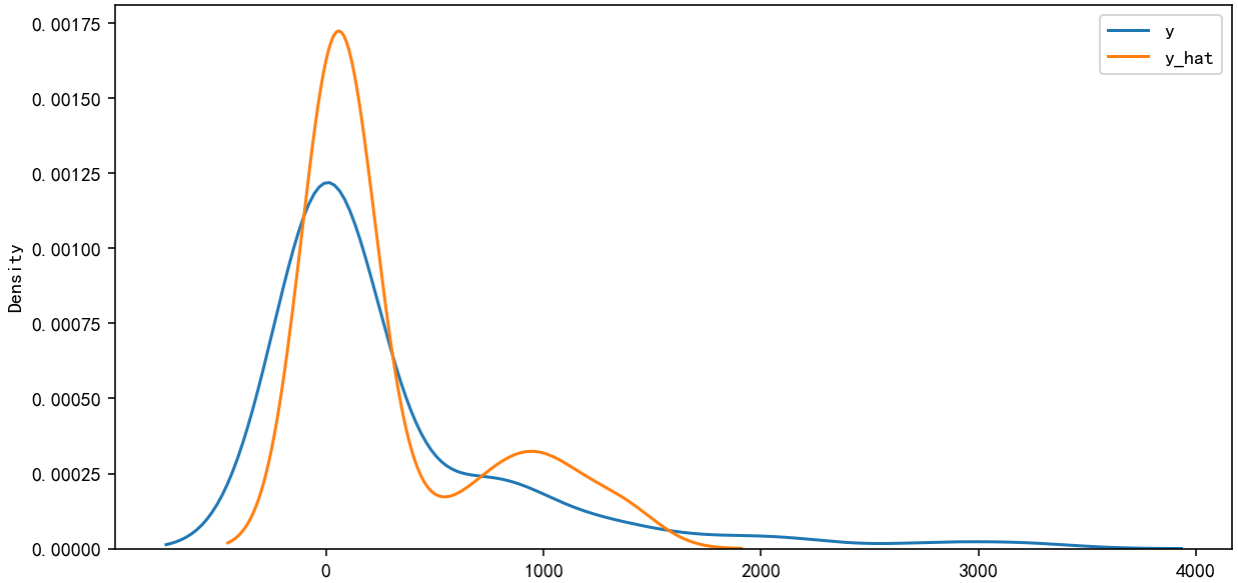
```python
y_hat_test = pr.predict(X_test_poly)
# train data의 산점도와 test data로 예측한 회귀선을 그래프로 출력
y_hat_test = pr.predict(X_test_poly)

fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(1,1,1)
ax.plot(X_train, y_train, 'o', label='Train Data') # 데이터 분포
ax.plot(X_test, y_test, 'r+', label='Predicted Value') # 모형이 학습한 회귀선

ax.legend(loc='best')
plt.xlabel('小程序（浏览量）')
plt.ylabel('文章的阅读量')
plt.show()
plt.close()
```



In [44]:

```
# 모형에 전체 X 데이터를 입력하여 예측한 값 y_hat을 실제 값 y와 비교
X_ploy = poly.fit_transform(X)
y_hat = pr.predict(X_ploy)

plt.figure(figsize=(10, 5))
ax1 = sns.distplot(y, hist=False, label="y")
ax2 = sns.distplot(y_hat, hist=False, label="y_hat", ax=ax1)
plt.legend()
plt.show()
plt.close()
```

```
C:\Users\toyou\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\toyou\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



다중회귀분석으로 상관 관계 분석하기

利用机器学习多重回归分析的数据相关关系分析

In [45]:

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

# 모든 데이터를 볼 수 있는 ( head랑 tail만 보여주는게 아니라, 다 보여주는 방법)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

## 중국어 폰트 설정 하는 방법

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib import font_manager, rc
plt.rcParams['axes.unicode_minus'] = False
f_path = "C:\Windows\Fonts/simhei.ttf"
font_name = font_manager.FontProperties(fname=f_path).get_name()
rc('font', family=font_name)

# 폰트를 선명하게 하기
# retina설정
```

```
from IPython.display import set_matplotlib_formats
set_matplotlib_formats("retina")

#data_name = f"浏览量和访问量跟文章阅读量的关系.csv"
df = pd.read_csv(data_name) #Encoding cp949는 못함

# Nan 값 0으로 만들어주기
df = df.fillna(value=0)
df.head()
```

```
C:\Users\toyou\AppData\Local\Temp/ipykernel_11688/2084237805.py:24: DeprecationWarning: `set_matplotlib_formats` is deprecated since IPython 7.23, directly use `matplotlib_inline.backend_inline.set_matplotlib_formats()`
  set_matplotlib_formats("retina")
```

Out[45]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量	淘宝特点活动	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unnamed: 16
0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0	2022.3.4~2022.3.8：情人节	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	02-25	3.0	1.0	102.0	31.0	0.0	137.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [46]:

```
df.columns
```

Out[46]:

```
Index(['日期', '小程序（浏览量）', '小程序（访客量）', '淘宝（浏览量）', '淘宝（访客量）', '掌上大学', '总计',
       '文章的阅读量', '淘宝特点活动', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11',
       'Unnamed: 12', 'Unnamed: 13', 'Unnamed: 14', 'Unnamed: 15',
       'Unnamed: 16', 'Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19',
       'Unnamed: 20', 'Unnamed: 21', 'Unnamed: 22', 'Unnamed: 23',
       'Unnamed: 24'],
      dtype='object')
```

In [47]:

```
# 속성(변수)선택
X = df[['小程序（浏览量）','小程序（访客量）','淘宝（访客量）']] #독립변수 X1,X2,X3
y = df['文章的阅读量']
```

```
# train data와 test data로 구분(7:3비율)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state=10)

print('훈련 데이터: ', X_train.shape)
print('검증 데이터: ', X_test.shape)
```

```
훈련 데이터:  (70, 3)
검증 데이터:  (31, 3)
```

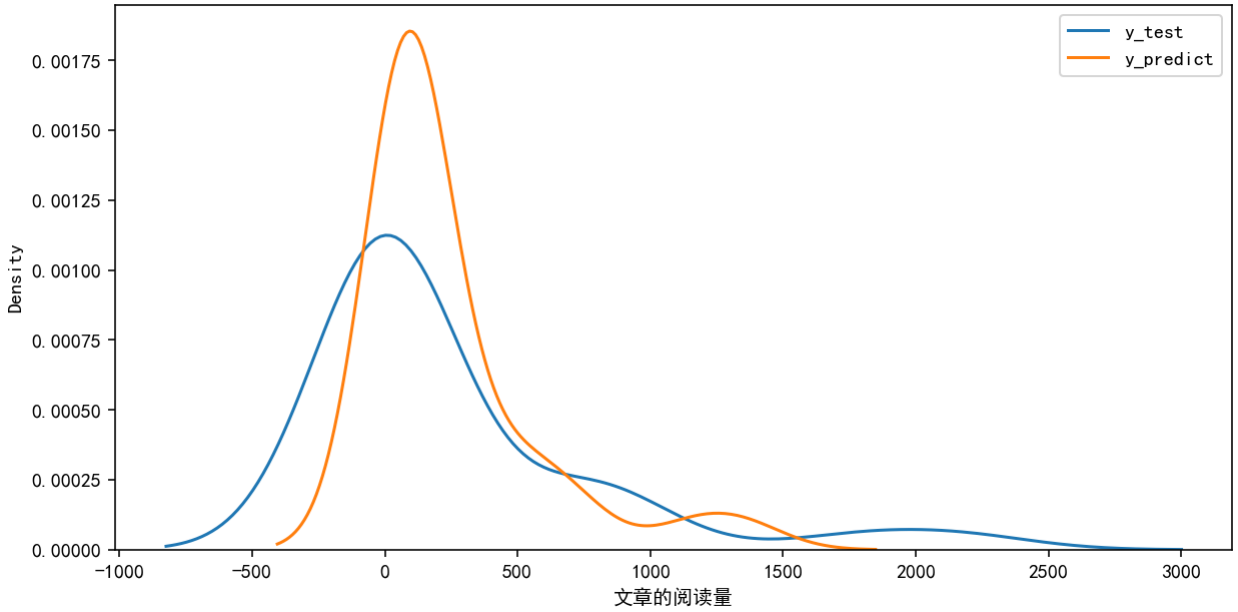
In [48]:

```
# sklearn 라이브러리에서 선형회귀분석 모듈 가져오기
from sklearn.linear_model import LinearRegression

# 단순회귀분석 모형 객체 생성
lr = LinearRegression()

# train data를 가지고 모형 학습
lr.fit(X_train, y_train)

# 학습을 마친 모형에 test data를 적용하여 결정계수 (R-제곱) 계산
r_square = lr.score(X_test, y_test)
print(r_square)
print('\n')

# 회귀식의 y절편
print('상수항 b', lr.intercept_)
```

```
0.3982073049947016


상수항 b 86.78301073479616
```

In [49]:

```
# train data의 산점도와 test data로 예측한 회귀선을 그래프로 출력
y_hat = lr.predict(X_test)

plt.figure(figsize=(10,5))
ax1 = sns.kdeplot(y_test, label="y_test")
ax2 = sns.kdeplot(y_hat, label="y_predict", ax=ax1)
plt.legend()
plt.show()
```



KNN 분류 알고리즘을 이용하여 상관관계 분석하기

KNN : KNeighborsClassifier()

In [50]:

```
# 만약에 컬럼에 성별이 있으면 범주형 데이터를 숫자형으로 변환하는 원핫코딩을 이용한다(one-hot-encoding)
# 속성(변수)선택
X = df[['小程序（浏览量）','小程序（访客量）','淘宝（访客量）']] #설명변수 X
y = df['文章的阅读量'] #예측 변수 Y

# 설명 변수 데이터를 정규화(normalization)
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

# train data와 test data로 구분(7:3 비율)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=10)

print('train data 개수: ', X_train.shape)
print('test data 개수: ', X_test.shape)
```

```
train data 개수:  (70, 3)
test data 개수:  (31, 3)
```

In [51]:

```
# sklearn 라이브러리에서 KNN 분류 모형 가져오기
from sklearn.neighbors import KNeighborsClassifier
```

```
#모형 객체 생성(K=5로 설정)
knn = KNeighborsClassifier(n_neighbors=5)

# train data를 가지고 모형 학습
knn.fit(X_train, y_train)

# test data를 가지고 y_hat을 예측(분류)
y_hat = knn.predict(X_test)

print(y_hat[0:10])
print(y_test.values[0:10])
```

```
[  0.   0.   0.   0.   0. 567. 732.   0.   0. 479.]
[   0.    0.    0.    0.    0.  835.  835.    0.    0. 1780.]
```

In [52]:

```
# 모형 성능 평가 - Confusion Matrix 계산
from sklearn import metrics
knn_matrix = metrics.confusion_matrix(y_test, y_hat)
print(knn_matrix)
```

```
[[23  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  1  1  0  0]
 [ 0  0  0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0]]
```

In [53]:

```
#모형 성능 평가 - 평가 지표 계산
knn_report = metrics.classification_report(y_test, y_hat)
print(knn_report)
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        23
       125.0       0.00      0.00      0.00         1
       366.0       0.00      0.00      0.00         0
       423.0       0.00      0.00      0.00         0
       479.0       0.00      0.00      0.00         0
       567.0       0.00      0.00      0.00         0
       702.0       0.00      0.00      0.00         1
       712.0       0.00      0.00      0.00         1
```

```
       732.0       0.00      0.00      0.00         0
       835.0       0.50      0.33      0.40         3
      1780.0       0.00      0.00      0.00         1
      2177.0       0.00      0.00      0.00         1

    accuracy                           0.77        31
   macro avg       0.12      0.11      0.12        31
weighted avg       0.79      0.77      0.78        31
```

```
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\toyou\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

SVM

Support Vector Machine

In [54]:

```
# 속성(변수)선택
X = df[['小程序（浏览量）','小程序（访客量）','淘宝（访客量）']] #설명변수 X
y = df['文章的阅读量'] #예측 변수 Y

# 설명 변수 데이터를 정규화(normalization)
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)

# train data와 test data로 구분(7:3 비율)
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=10)

print('train data 개수: ', X_train.shape)
print('test data 개수: ', X_test.shape)
```

```
train data 개수:  (70, 3)
test data 개수:  (31, 3)
```

In [55]:

```
# sklearn라이브러리에서 SVM 분류 모형 가져오기
from sklearn import svm

# 모형 객체 생성(kernel='rbf' 적용)
svm_model = svm.SVC(kernel='rbf')

# train data를 가지고 모형 학습
svm_model.fit(X_train, y_train)

# test data를 가지고 y_hat예측(분류)
y_hat = svm_model.predict(X_test)

print(y_hat[0:10])
print(y_test.values[0:10])
```

```
[   0.    0.    0.    0.    0.  835. 1188.    0.    0.  835.]
[   0.    0.    0.    0.    0.  835.  835.    0.    0. 1780.]
```

In [56]:

```
df.sort_values(by='总计',ascending=False) #总计순으로 sort_values()정렬하여 어느날에 가장많은 값을 가진지 확인.
```

Out[56]:

	日期	小程序（浏览量）	小程序（访客量）	淘宝（浏览量）	淘宝（访客量）	掌上大学	总计	文章的阅读量	淘宝特点活动	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12	Unnamed: 13	Unnamed: 14	Unnamed: 15	Unn
18	03-11	158.0	28.0	779.0	62.0	0.0	1027.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
23	03-16	429.0	35.0	117.0	35.0	0.0	616.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
27	03-20	3.0	2.0	395.0	110.0	0.0	510.0	137.0	加购即赠眼霜棒	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
25	03-18	136.0	13.0	273.0	45.0	12.0	479.0	835.0	加购即赠眼霜棒	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
16	03-09	221.0	35.0	150.0	29.0	0.0	435.0	479.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

26	03-19	98.0	17.0	209.0	52.0	0.0	376.0	702.0	加购即赠眼霜棒	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	03-10	188.0	32.0	124.0	20.0	4.0	368.0	1202.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	03-15	77.0	13.0	162.0	29.0	1.0	282.0	1357.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	02-28	73.0	12.0	123.0	28.0	40.0	276.0	423.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	03-17	95.0	11.0	131.0	33.0	3.0	273.0	366.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	03-07	193.0	27.0	34.0	9.0	1.0	264.0	1780.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	03-08	115.0	35.0	91.0	16.0	0.0	257.0	732.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	03-03	45.0	14.0	142.0	33.0	1.0	235.0	1470.0	2022.3.4~2022.3.8：3.8节	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	03-01	33.0	18.0	143.0	33.0	6.0	233.0	3199.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0	02-21	16.0	4.0	184.0	23.0	0.0	227.0	2002.0	2022.3.4~2022.3.8：情人节	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	03-04	28.0	10.0	154.0	34.0	0.0	226.0	701.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	03-02	20.0	4.0	158.0	32.0	0.0	214.0	2791.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	03-14	40.0	9.0	133.0	28.0	0.0	210.0	125.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	02-23	8.0	1.0	155.0	18.0	0.0	182.0	712.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	03-13	107.0	37.0	31.0	5.0	1.0	181.0	1188.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	02-26	13.0	4.0	127.0	28.0	0.0	172.0	1290.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	02-24	22.0	3.0	119.0	19.0	0.0	163.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	03-12	66.0	31.0	50.0	8.0	1.0	156.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	02-22	7.0	2.0	114.0	30.0	0.0	153.0	2177.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	02-	3.0	1.0	102.0	31.0	0.0	137.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	25															
6	02-27	4.0	2.0	79.0	22.0	0.0	107.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
12	03-05	12.0	6.0	50.0	17.0	0.0	85.0	567.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	03-06	2.0	2.0	46.0	15.0	0.0	65.0	835.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
79	05-11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
78	05-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
74	05-06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
80	05-12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
77	05-09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
76	05-08	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
75	05-07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
68	04-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
73	05-05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72	05-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71	05-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
70	05-02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69	05-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
67	04-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
66	04-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
82	05-14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

81	05-13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
95	05-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
83	05-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
84	05-16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
99	05-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
98	05-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
97	05-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
96	05-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
64	04-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
94	05-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
93	05-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
92	05-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
91	05-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
90	05-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
89	05-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
88	05-20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
87	05-19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
86	05-18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
85	05-17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
65	04-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

50	04-12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
63	04-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
62	04-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
42	04-04	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
41	04-03	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
40	04-02	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
39	04-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
38	03-31	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
37	03-30	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
36	03-29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
35	03-28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
34	03-27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
33	03-26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
32	03-25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
31	03-24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
30	03-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
29	03-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
28	03-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
43	04-05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
44	04-	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	06																
45	04-07	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
55	04-17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
61	04-23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
60	04-22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
59	04-21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
58	04-20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
57	04-19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
56	04-18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
54	04-16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
46	04-08	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
53	04-15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
52	04-14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
51	04-13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
49	04-11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
48	04-10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
47	04-09	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
100	06-01	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	