ai_assignment1

October 20, 2022

[25]:
```
import pandas as pd
import requests
from tqdm import tqdm
import json
from collections import Counter
import random

import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#(1,5)->1~5        .

#     1035
num=1035

#Task 1
#df1_1=getInfo1(1,num)
df1_1=pd.read_csv('C:/Users/dlwld/Downloads/lottery-2022.09-2.csv')
totalNumlist = list(df1_1['first']) + list(df1_1['second']) +␣
 ↪list(df1_1['third']) + list(df1_1['fourth']) + list(df1_1['fifth']) +␣
 ↪list(df1_1['sixth'])+list(df1_1['bonus'])
count=Counter(totalNumlist)
countSet=count.most_common(45)
df1_2=pd.DataFrame(countSet,columns=['num','times'])
df1_2.to_csv("getTimes1.csv",index=False)
pd.read_csv("getTimes1.csv")
```

[25]:

	num	times
0	43	180
1	34	178
2	1	175
3	27	175
4	12	174
5	17	174
6	13	174
7	18	171

```
8    33    171
9    39    170
10   20    169
11    4    168
12   14    166
13   26    165
14   10    165
15   24    165
16    2    164
17   11    164
18   37    164
19   38    164
20   40    164
21    3    162
22   31    162
23    7    161
24   16    161
25   45    161
26   15    160
27   21    160
28   35    160
29    6    159
30   44    159
31   36    158
32   42    158
33    5    156
34    8    155
35   19    155
36   30    152
37   25    151
38   32    146
39   28    145
40   29    144
41   41    144
42   23    142
43    9    135
44   22    132
```

[26]:
```python
#Task 2
def getInfo2(minNo, maxNo):
    round=[]
    date=[]
    num_winners=[]
    reward=[]
    first = []
    second = []
    third = []
```

```
    fourth = []
    fifth = []
    sixth = []
    bonus = []
    win=[]
    #tqdm           .
    for i in tqdm(range(minNo, maxNo, 1)): #(1,5)  -> 1  5  1     i
        #No=
        # win=1
        round.append(i)
        i=i-1;
        date.append(list(df1_1['date'])[i])
        num_winners.append(list(df1_1['num_winners'])[i])
        reward.append(list(df1_1['reward'])[i])
        first.append(list(df1_1['first'])[i])
        second.append(list(df1_1['second'])[i])
        third.append(list(df1_1['third'])[i])
        fourth.append(list(df1_1['fourth'])[i])
        fifth.append(list(df1_1['fifth'])[i])
        sixth.append(list(df1_1['sixth'])[i])
        bonus.append(list(df1_1['bonus'])[i])
        win.append("1")
    ␣
↪real=[list(df1_1['first'])[i],list(df1_1['second'])[i],list(df1_1['third'])[i],
        ␣
↪list(df1_1['fourth'])[i],list(df1_1['fifth'])[i],list(df1_1['sixth'])[i]]
        # win=0
        fake=random.sample(range(1,46),7)
        round.append(i)
        date.append(list(df1_1['date'])[i])
        num_winners.append(list(df1_1['num_winners'])[i])
        first.append(fake[0])
        second.append(fake[1])
        third.append(fake[2])
        fourth.append(fake[3])
        fifth.append(fake[4])
        sixth.append(fake[5])
        bonus.append(fake[6])
        if(real==fake):
            win.append("1")
            reward.append(list(df1_1['reward'])[i])
        else:
            win.append("0")
            reward.append(0)
        dataset = { "round":round,"date":date, "num_winners":num_winners,␣
↪"reward":reward,"first":first, "second":second, "third":third, "fourth":
↪fourth, "fifth":fifth, "sixth":sixth,
```

```
                        "bonus":bonus,"win":win}
    df = pd.DataFrame(dataset)

    return df
#Task 2
df2_1=getInfo2(1,num)
df2_1.to_csv("getInfo2.csv",index=False)
pd.read_csv("getInfo2.csv")
```

```
100%|
 | 1034/1034 [00:03<00:00, 316.47it/s]
```

[26]:

	round	date	num_winners	reward	first	second	third	fourth	fifth	sixth	bonus	win
0	1	2022.09.24	9	2868856209	26	31	32	33	38	40	11	1
1	0	2022.09.24	9	0	7	32	3	43	23	16	25	0
2	2	2022.09.17	13	1913414943	3	11	15	20	35	44	10	1
3	1	2022.09.17	13	0	19	32	12	43	7	26	21	0
4	3	2022.09.10	10	2675257538	1	6	12	19	36	42	28	1
…	…	…	…	…	…	…	…	…	…	…	…	…
2063	1031	2002.12.21	1	0	9	43	44	13	8	23	42	0
2064	1033	2002.12.14	1	2002006800	9	13	21	25	32	42	2	1
2065	1032	2002.12.14	1	0	22	31	17	20	35	36	8	0
2066	1034	2002.12.07	0	0	10	23	29	33	37	40	16	1
2067	1033	2002.12.07	0	0	25	37	42	1	23	24	31	0

[2068 rows x 12 columns]

[21]:
```
#Task 3
totalFakeList=list(df2_1['first']) + list(df2_1['second']) +␣
 ↪list(df2_1['third']) + list(df2_1['fourth']) + list(df2_1['fifth']) +␣
 ↪list(df2_1['sixth'])+list(df2_1['bonus'])
totalFakeList=totalFakeList[1::2]
count2=Counter(totalFakeList)
countSet2=count2.most_common(45)
df2_2=pd.DataFrame(countSet2,columns=['num','times'])
```

```
df2_2.to_csv("getTimes2.csv",index=False)
pd.read_csv("getTimes2.csv")
```

[21]:

	num	times
0	11	186
1	23	177
2	27	177
3	44	177
4	42	173
5	3	172
6	36	172
7	26	171
8	2	171
9	30	170
10	28	170
11	8	170
12	45	169
13	7	169
14	19	169
15	9	169
16	18	167
17	14	166
18	37	166
19	21	166
20	35	166
21	1	165
22	15	165
23	40	163
24	39	163
25	32	162
26	4	162
27	38	161
28	43	159
29	25	156
30	5	155
31	6	155
32	10	154
33	24	153
34	22	152
35	12	148
36	16	146
37	41	145
38	33	144
39	34	143
40	17	142
41	13	141
42	31	141

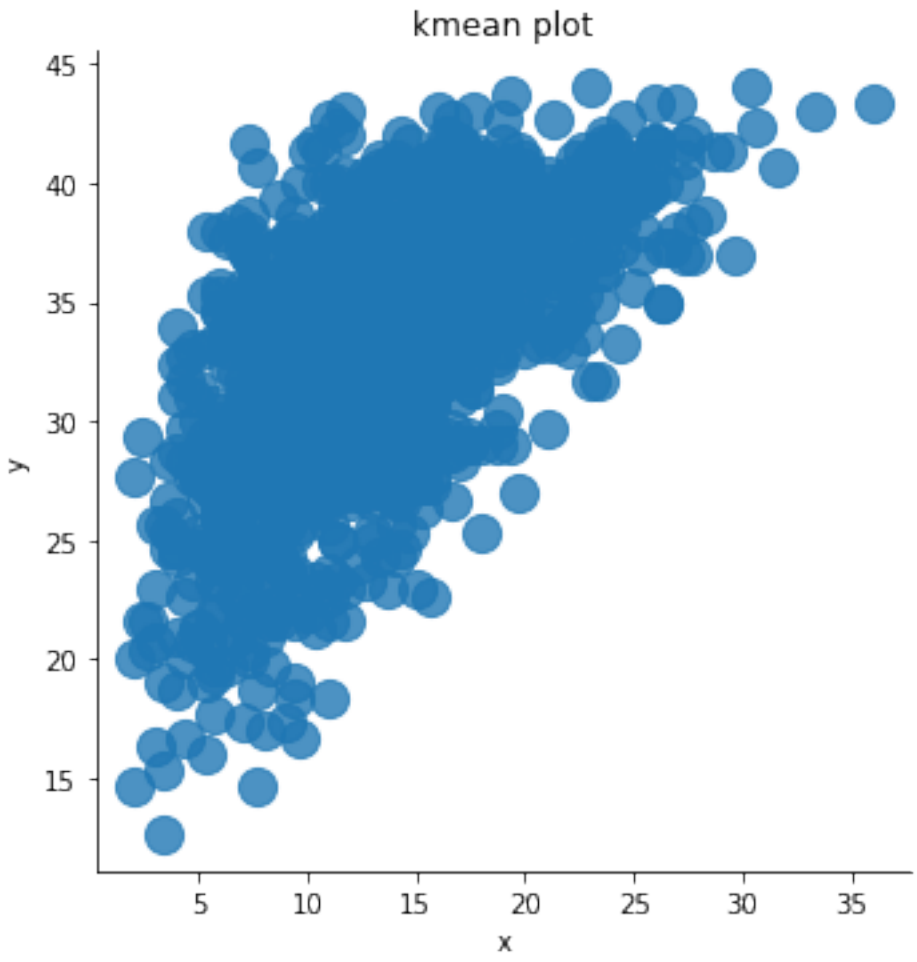
```
43   20    135
44   29    135
```

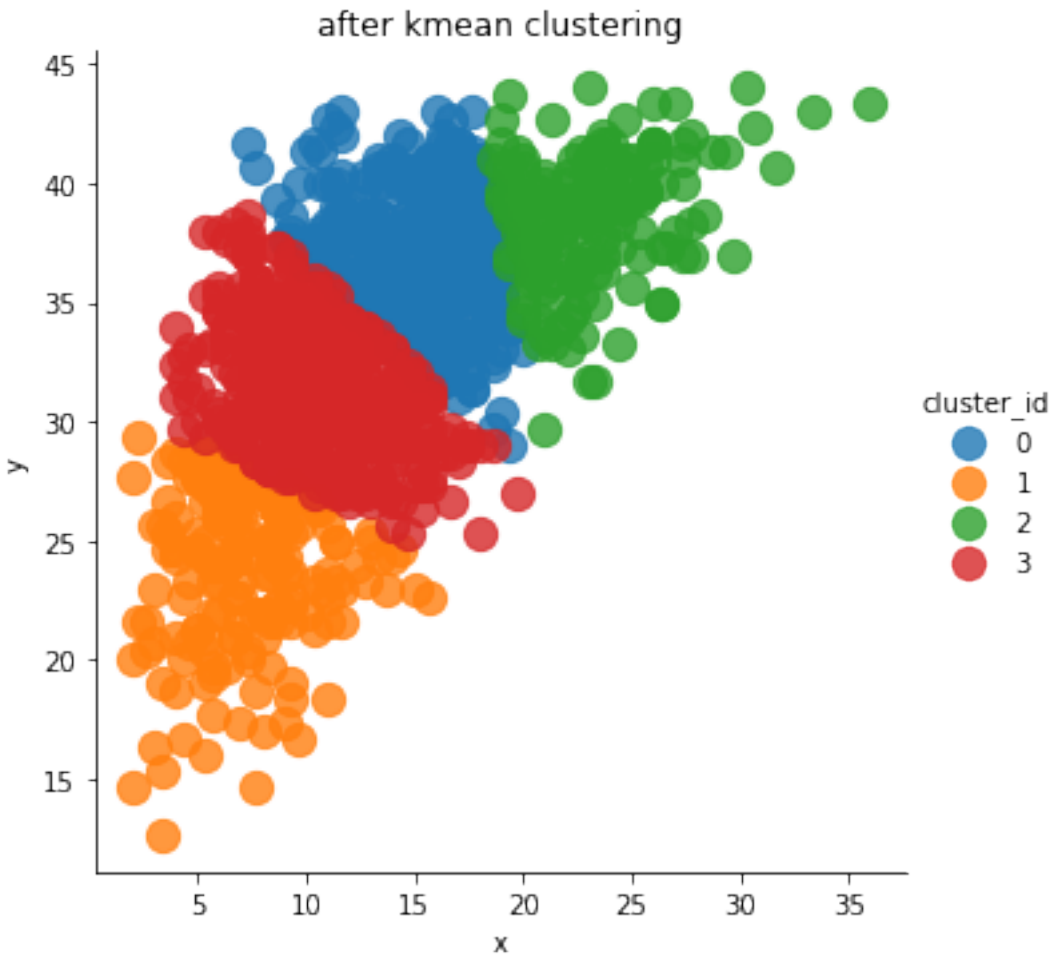
[24]:
```
# #Task 4
x=[]
y=[]
df4=pd.DataFrame(columns=['x','y'])
for i in range(num):
    # 1~3       x, 4~6       y    .
    i=i-1
    x.
 ↪append((list(df1_1['first'])[i]+list(df1_1['second'])[i]+list(df1_1['third'])[i])/
 ↪3)
    y.
 ↪append((list(df1_1['fourth'])[i]+list(df1_1['fifth'])[i]+list(df1_1['sixth'])[i])/
 ↪3)
for i in range(num):
    df4.loc[i]=[x[i],y[i]]

# visualize data point
sns.lmplot('x', 'y', data=df4, fit_reg=False, scatter_kws={"s": 200}) # x-axis,␣
 ↪y-axis, data, no line, marker size
# title
plt.title('kmean plot')
# x-axis label
plt.xlabel('x')
# y-axis label
plt.ylabel('y')
# convert dataframe to numpy array
data_points = df4.values
kmeans = KMeans(n_clusters=4).fit(data_points)
# cluster id for each data point
kmeans.labels_
# this is final centroids position
kmeans.cluster_centers_
df4['cluster_id'] = kmeans.labels_
df4.head(12)
sns.lmplot('x', 'y', data=df4, fit_reg=False,  # x-axis, y-axis, data, no line
           scatter_kws={"s": 150}, # marker size
           hue="cluster_id") # color

# title
plt.title('after kmean clustering')
```

```
C:\Users\dlwld\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
```

```
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
C:\Users\dlwld\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[24]: `Text(0.5, 1.0, 'after kmean clustering')`




after kmean clustering
45
40
35
30
25
20
15
y
5
10
15
20
25
30
35
x
cluster_id
0
1
2
3

[ ]:

Task-5. Using Pandas, the table was made in the order of the winning numbers that appeared the most among the 1,036 winning numbers. In Task2, Task1's getInfo function was given my own six random numbers for each round so that I could check whether I won. (win=1,lose=0) In Task3, I made a table in the order of the most number of my lottoes. In Task4, the average of the first, second, and third winning numbers was set to x, and the average of the fourth, fifth, and sixth winning numbers was set to y, and the correlation was sought using K-mean clustering. However, it was not easy to find a correlation because the lottery number was definitely random. I hope there is a task that Java can do instead of Python. This is because there must be students who are not the main language of Python as well as me.

Pandas를 이용하여 1036회차의 당첨번호 중 가장 많이 나온 당첨번호 순으로 표를 만들었다. Task2에서는 Task1의 getInfo함수에 각 회차마다 나만의 6개의 랜덤번호를 부여해서 당첨이 되었는지를 확인할 수 있게끔 했다. (win=1,lose=0)

Task3에서는 나의 로또 중 가장 많이 나온 번호 순으로 표를 만들었다.

Task4에서는 1,2,3번째 당첨번호의 평균을 x로 놓고, 4,5,6번째 당첨번호의 평균을 y로 놓은 후 K-mean clustering을 이용하여 상관관계를 구하고자 했다. 하지만 역시 로또번호는 확실히 랜덤이라 상관관계를 구하기가 쉽지 않았다. (집단이 기울어진 것은 로또번호가 오름차순으로 정렬되어있기 때문이었다.) 파이썬 말고 자바로도 할 수 있는 과제가 있으면 좋겠다고 생각한다. 파이썬이 주언어가 아닌 학 생도 나 뿐만 아니라 분명 있을 것이기 때문이다