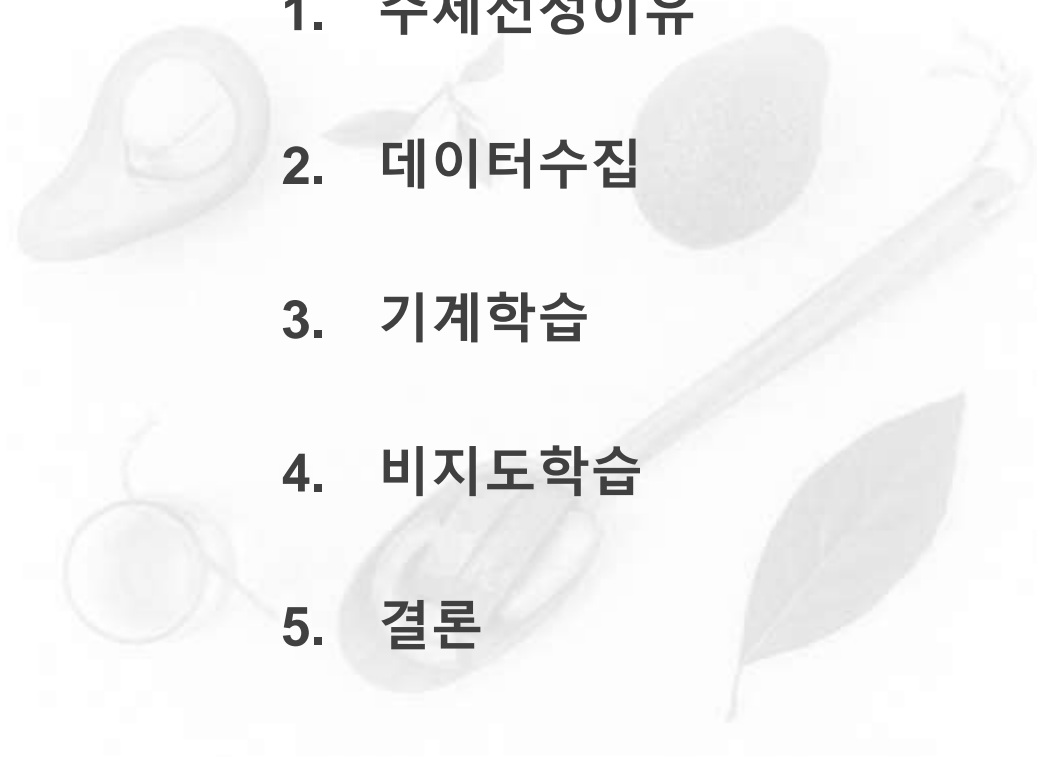


# 빅데이터

학번 : 20172048

이름 : 박주형

# CONTENTS

- 
1. 주제선정이유
  2. 데이터수집
  3. 기계학습
  4. 비지도학습
  5. 결론

# 1. 주제선정이유

본인 뿐만 아니라 많은사람들이 카페를 자주 이용합니다.

요즘은 식사를 마치고 카페를 방문하는것 이 거의 당연시 되어가는 문화 입니다.

학생들은 공부를 독서실,도서관이 아닌 카페에서 공부를 자주합니다.

사람들이 어느 카페를 많이가고 왜 그 카페를 선호하는지 알기위해서 주제를 선정하였습니다.

또 서울에서 핫플레이스인 곳을 알기위해 매출이랑 아주 근접하게 연결되어있다고 생각해서 서울시의 매출을 조사하면서 핫플레이스 TOP3를 알아보겠습니다.

# 2. 데이터수집(파일데이터)

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

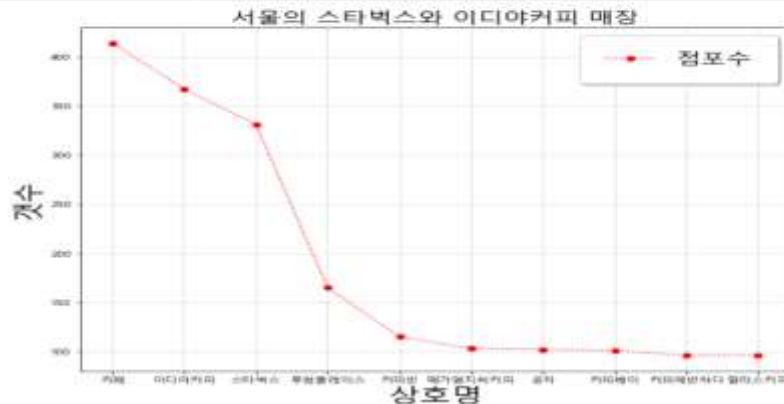
```
df = pd.read_csv('서울시태깅정보.csv')
```

	상자업소번호	상호명	상권업종중분류명	시도명	시군구명	경도	위도	지점명
0	17174175	비지토	한식	서울특별시	서초구	126.991394	37.488375	NaN
1	17174119	스리애드	한식	서울특별시	종로구	126.969952	37.481133	NaN
2	17174096	채움	한식	서울특별시	영등포구	126.961794	37.572387	NaN
3	17174052	포구인주방	가정주방인테리어	서울특별시	영등포구	126.897832	37.536700	NaN
4	17174040	다방	한식	서울특별시	서초구	127.009392	37.483436	NaN
...	...	...	...	...	...	...	...	...
316073	16982165	김치대우방	한식	서울특별시	서대문구	126.951473	37.578889	NaN
316074	16983995	한올라미스	한식	서울특별시	마포구	126.921671	37.562230	NaN
316075	20861429	강인일침사	침물난방/간접가열매	서울특별시	강동구	127.131765	37.529776	NaN
316076	16991480	솔시카고피자	패스트푸드	서울특별시	종로구	126.908321	37.481893	NaN
316077	16990348	고자바거울리스	패스트푸드	서울특별시	종로구	127.000989	37.525913	NaN

```
df[['상호명']] = df[['상호명']].value_counts()
```

```
상호명
상호명_업종명_시도명_시군구명_경도_위도_지점명
16981627 커피 커피점/카페 서울특별시 구로구 129.576483 37.491889 동대문 1
17125539 커피 커피점/카페 서울특별시 서초구 127.063614 37.455071 강남1 1
17171898 커피 한식 서울특별시 강남구 127.040004 37.521996 도산대 1
17111889 커피 커피점/카페 서울특별시 관악구 128.574629 37.538833 오수동 1
17120164 커피 한식 서울특별시 관악구 128.528623 37.601851 관동동관동관 1
16986434 커피 한식 서울특별시 강남구 127.049811 37.514088 기루면대이코 1
16988990 커피 커피점/카페 서울특별시 강릉구 127.090108 37.542518 연개간 1
16988937 커피 한식 서울특별시 영등포구 127.082988 37.592859 시그니처 1
16988995 커피 커피점/카페 서울특별시 강남구 127.023883 37.510813 김복논현정 1
17526406 커피 커피점/카페 서울특별시 강남구 127.080253 37.491045 유달로 1
```

```
plt.figure(figsize=(10,8))
plt.plot(ar,marker='o',linestyle='-',label='점포수',color='red')
plt.legend(fontsize=20,shadow=True,borderpad=1)
plt.title('서울의 스타벅스와 이디야커피 매장')
plt.xlabel('상호명',fontsize='25')
plt.ylabel('점포수',fontsize='25')
plt.grid(alpha=0.5)
```



## 2. 데이터수집(파일데이터)

서울시매장정보 파일데이터를 사용해서 정보를 불러왔습니다.

서울시 매장중에서 업종이름이 “카페”인 상호명들을 상위 10개의 데이터만 출력해보았습니다.

데이터를 그래프로 출력해보니 가장높은순위에 있는 것은 “카페”였습니다

상호명 “카페”에 대한 데이터들을 조사해보니, 이름이 없는 카페들이기에 “카페”는 제외하겠습니다.

저는 제가 자주가는 카페이기도하는 “스타벅스”, “이디야커피”를 기반으로 데이터를 출력하겠습니다.

## 2. 데이터수집(파일데이터)

저는 친구들을 만나러 서울에 있는  
카페를 자주가기에 서울시 어디에  
위치하고 있는지 출력해보겠습니다.

```
from matplotlib import font_manager, rc
import matplotlib
font_location="c:/Windows/fonts/malgun.ttf"
font_name=font_manager.FontProperties(fname=font_location).get_name()
matplotlib.rc('font',family=font_name)
```

```
df_star=df[df['상호명']=='스타벅스']
df_star=df_star.reset_index()
df_star
```

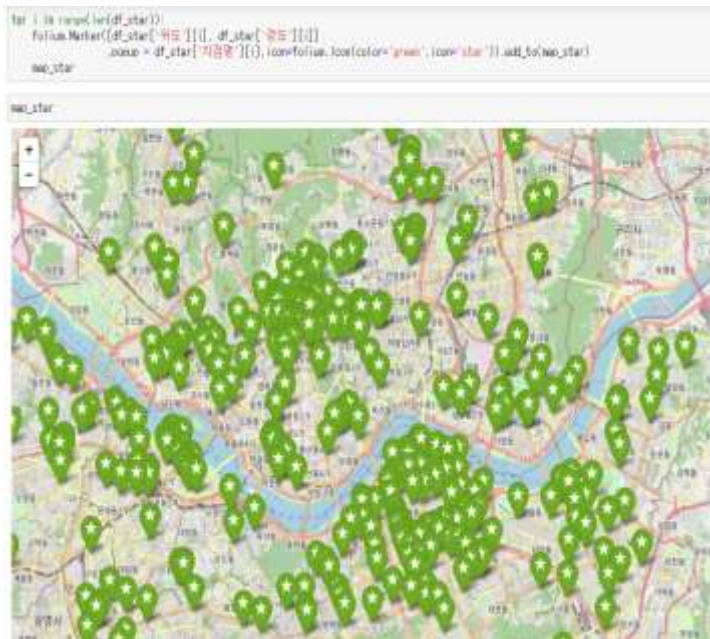
```
df_ed=df[df['상호명']=='이디야커피']
df_ed=df_ed.reset_index()
df_ed
```

```
!pip install folium import folium
```

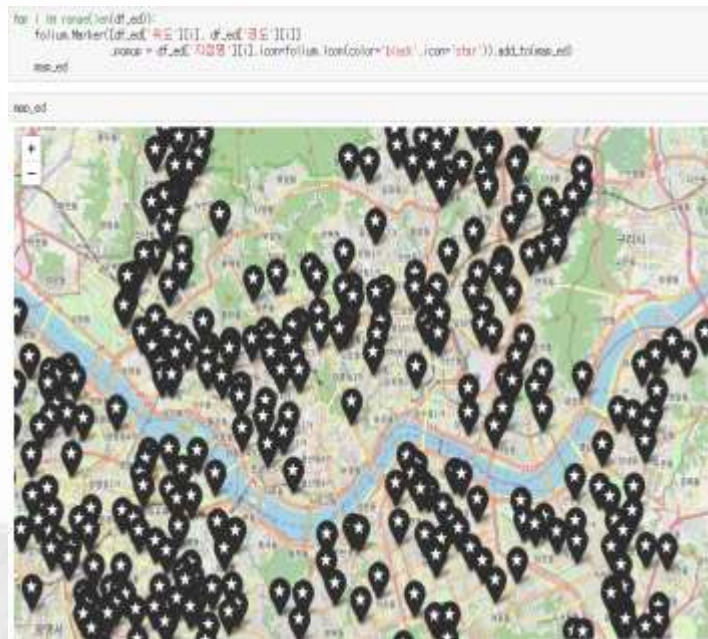
데이터안에 있는 경도와 위도를 사  
용해서 데이터를 출력하기 위해서는  
왼쪽에 있는 사진들과같이 명령을  
해주어야합니다.



## 2. 데이터수집(파일데이터)



위 사진과 같이 “스타벅스”와 “이디야커피”는  
다음과 같이 위치하고 있었습니다



“스타벅스”와 “이디야커피”는 주로 한강주변인 올림픽대로와 강  
변북로 주변에 많이 위치하고 있었습니다. 하지만 스타벅스매장  
의 수가 많은곳에는 이디야커피의 매장수가 상대적으로 적었고  
반대로 이디야커피의 매장수가 많은곳은 스타벅스의 매장수가 상  
대적으로 적은것을 알 수 있었습니다.

## 2. 데이터수집(웹)

## 이디야 커피

사람들이 왜 다른 카페들 보다 스타벅스와 이디야커피를 많이 이용하는지 궁금해서 후기를 조사해보았습니다.

```
url='https://mm-melier.tistory.com/20'
```



스타벅스

```
url='https://www.jbsori.com/news/articleView.html?idxno=65'
```





## 2. 데이터수집 (오픈API)

서울에서 핫플레이스인 곳을 알기위해 매출이랑 아주 근접하게 연결되어있습니다.  
서울시의 매출을 조사하면서 핫플레이스 TOP3를 알아보겠습니다.

url='http://openapi.seoul.go.kr:8088/7a4d5158486a516833337a6557436e/m1/wonTrdh

```
html=requests.get(uri).text
```

## Introduction

```
xmltodict.parse(requests.get(url).text)
```

```
json.dumps(xltdict.parse(requests.get(url).text))
```

```
data=json.loads(json.dumps(xlntodict.parse(requests.get(url).text)))
```

data

```
requests.get(url).status_code
```

200

```
data.keys()
```

```
dict_keys(['YwamTrdh|NonCnsapOa'])
```

```
data["VignTcdh11NonOncOa"]
```

```
data['YwsmTrdhIncmCnsmPQq']
```

```
data['YwsmTrdh|NcmCnsmPq']['row']
```

```
df=pd.DataFrame(data['VwsMTrdhINcmCnsmPQq']
```

df

[illegible]

## 2. 데이터수집 (오픈API)

월평균소득으로 소득분위를 정해보았습니다.

```
result=[]
for i in df['월_평균_소득_금액']:
    if 1000000>i>0:
        i=1;
    else:
        if 2000000>i>1000000:
            i=2;
        else:
            if 3000000>i>2000000:
                i=3;
            else:
                if 4000000>i>3000000:
                    i=4;
                else:
                    if 5000000>i>4000000:
                        i=5;
                    else:
                        if 6000000>i>5000000:
                            i=6;
                        else:
                            if 7000000>i>6000000:
                                i=7;
result.append(i)
```

```
df['등급']=result
```

등급라인이 생성되었습니다.

최상위 등급은 8등급,7등급으로 출력되었습니다.

```
df['등급'].value_counts(ascending=True)
```

8	19
7	203
6	1421
5	5060
3	8207
4	14380

```
df[df['등급']==8]['상권_코드명']
```

10191	삼성로57길
11201	삼성로57길
12211	삼성로57길
13221	삼성로57길
14231	삼성로57길
15241	삼성로57길
16251	삼성로57길
17261	삼성로57길
18271	삼성로57길
19281	삼성로57길
20291	삼성로57길
21301	삼성로57길
22311	삼성로57길
23321	삼성로57길
24331	삼성로57길
25341	삼성로57길
26351	삼성로57길
27361	삼성로57길
28371	삼성로57길

```
df[df['등급']==7]['상권_코드명'].value_counts().head(2)
```

고무재로9길	20
세민당로17길	19

8분위,7분위 등급의 코드명은 위 사진과같습니다.

```
df[df['등급']==7]['상권_구분_코드명']
```

91	골목상권
163	골목상권
1101	골목상권
1173	골목상권
2111	골목상권
...	
28425	골목상권
28443	골목상권
28766	골목상권
29231	골목상권
29240	골목상권

Name: 상권\_구분\_코드명, Length: 203, dtype: object

## 2. 데이터수집 (오픈API)

8분위, 7분위의 상권구분코드명은 "골목상권" 으로 출력되었습니다.

```
df[df['등급'] == 7]['상권_구분_코드_명']
```

```
91      골목상권  
163     골목상권  
1101    골목상권  
1173    골목상권  
2111    골목상권
```

```
...  
28425   골목상권  
28443   골목상권  
28766   골목상권  
29231   골목상권  
29240   골목상권
```

Name: 상권\_구분\_코드\_명, Length: 203, dtype: object

```
df[df['등급'] == 8]['상권_구분_코드_명']
```

```
10191   골목상권  
11201   골목상권  
12211   골목상권  
13221   골목상권  
14231   골목상권  
15241   골목상권  
16251   골목상권  
17261   골목상권  
18271   골목상권  
19281   골목상권  
20291   골목상권  
21301   골목상권  
22311   골목상권  
23321   골목상권  
24331   골목상권  
25341   골목상권  
26351   골목상권  
27361   골목상권  
28371   골목상권
```

## 2. 데이터수집 (오픈API)

TOP1: 삼성로57길



TOP2: 고무래로8길



TOP3: 사임당로17길



TOP3순위로 선정해서 네이버 지도에서 출력해보았습니다.

## 2. 데이터수집 분석결과

TOP1 : 삼성로57길 - 검색결과로는 대치동으로 확인되었습니다. 대치동은 학구열이 매우 높은 곳으로 유명한 곳입니다.

따라서 학생들이 많기에 대치동의 수입이 많은것으로 예측되기에 TOP1으로 선정된것으로 예측됩니다.

TOP2 : 고무래로8길 - 서초구로 검색되었습니다. 서울의 대표 명산 청계산은 다양한 산행 코스가 구성되어 있어 서울 시민들뿐만아니라 타지역사람들도 많이 찾습니다. 프랑스인들이 많이 거주하고 있는 서래마을은 트렌디한 맛집과 디저트 카페들이 가득합니다. 몽마르트 공원이 구성되어 있어 뽀띠프랑스라고 불립니다. 관광거리가 잘 관리되어 있어서 사람들의 많은 사람들이 방문하기에 TOP2로 출력된것으로 예측됩니다.

TOP3 : 사임당로17길 - TOP3주소도 서초구로 검색됩니다. TOP2와 거리상으로는 버스로 15분 남짓이면 도착하는 거리로 출력됩니다. 가까운거리 이기에 TOP2와 비슷한 이유인것으로 예측됩니다.



### 3. 기계학습

[illegible]

각 지출금액으로 소득분위를 추출하겠습니다.

```
x=df[['월_평균_소득_금액', '지출_총금액', '식료품_지출_총금액', '의류_신발_지출_총금액', '의료비_지출_총금액', '교통_지출_총금액', '여가_지출_총금액', '문화_지출_총금액', '유출_지출_총금액']]
y=df['소득']
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y)
```

# 3. 기계 학습(Regressor)

```
knn1= KNeighborsRegressor(n_neighbors=1)
knn1.fit(X_train,y_train)
]: KNeighborsRegressor(n_neighbors=1)
knn1.predict(X_test)
]: array([3., 4., 4., ..., 4., 5])
knn1.score(X_test,y_test)
]: 0.8746295087371968

knn1= KNeighborsRegressor(n_neighbors=3)
knn1.fit(X_train,y_train)
]: KNeighborsRegressor(n_neighbors=3)
knn1.predict(X_test)
]: array([3., 4., 4., ..., 4., 5])
knn1.score(X_test,y_test)
]: 0.8235595930514674

knn1= KNeighborsRegressor(n_neighbors=5)
knn1.fit(X_train,y_train)
]: KNeighborsRegressor(n_neighbors=5)
knn1.predict(X_test)
]: array([3., 4., 4., ..., 4., 5])
knn1.score(X_test,y_test)
]: 0.77070130703150770

knn1= KNeighborsRegressor(n_neighbors=7)
knn1.fit(X_train,y_train)
]: KNeighborsRegressor(n_neighbors=7)
knn1.predict(X_test)
]: array([3.14285714, 4., 5.])
knn1.score(X_test,y_test)
]: 0.6945138008173227

knn1= KNeighborsRegressor(n_neighbors=9)
knn1.fit(X_train,y_train)
]: KNeighborsRegressor(n_neighbors=9)
knn1.predict(X_test)
]: array([3.33333333, 4.22222222, 5.77777778, ..., 4., 4.11111111, 5., 1])
knn1.score(X_test,y_test)
]: 0.6403325710199759
```

n\_neighbors의 숫자가 상승할수록  
KNeighborsRegressor의 score는 낮아지는 것을 확인할 수 있습니다.

### 3. 기계학습(Regressor)

```
rf1=RandomForestRegressor(n_estimators=1)
```

```
rf1.fit(X_train,y_train)
```

```
: RandomForestRegressor(n_estimators=1)
```

```
rf1.predict(X_test)
```

```
: array([3., 4., 4., ..., 4., 5., 5.])
```

```
rf1.score(X_test,y_test)
```

```
: 1.0
```

```
rf1=RandomForestRegressor(n_estimators=50)
```

```
rf1.fit(X_train,y_train)
```

```
: RandomForestRegressor(n_estimators=50)
```

```
rf1.predict(X_test)
```

```
: array([3., 4., 4., ..., 4., 5., 5.])
```

```
rf1.score(X_test,y_test)
```

```
: 0.9999588604615497
```

```
rf1=RandomForestRegressor(n_estimators=100)
```

```
rf1.fit(X_train,y_train)
```

```
: RandomForestRegressor()
```

```
rf1.predict(X_test)
```

```
: array([3., 4., 4., ..., 4., 5., 5.])
```

```
rf1.score(X_test,y_test)
```

```
: 0.9999526107050242
```

**n\_estimators의 숫자가 상승할수록  
RandomForestRegressor의 score는  
낮아지는 것을 확인할 수 있습니다.**

### 3. 기계학습(Regressor)

```
dt=DecisionTreeClassifier(random_state=1)
```

```
dt.fit(X_train,y_train)
```

```
] DecisionTreeClassifier(random_state=1)
```

```
dt.predict(X_test)
```

```
] array([3, 4, 4, ..., 4, 5, 5], dtype=int64)
```

```
dt.score(X_test,y_test)
```

```
] 1.0
```

```
dt=DecisionTreeClassifier(random_state=100)
```

```
dt.fit(X_train,y_train)
```

```
] DecisionTreeClassifier(random_state=100)
```

```
dt.predict(X_test)
```

```
] array([3, 4, 4, ..., 4, 5, 5], dtype=int64)
```

```
dt.score(X_test,y_test)
```

```
] 1.0
```

DecisionTreeRegressor는  
random\_state 값을 변경해도 100프로의  
확률을 가지고있다는것을 알수있습니다.

### 3. 기계학습(Classifier)

```
▶ knn1= KNeighborsRegressor(n_neighbors=1)
```

```
▶ knn1.fit(X_train,y_train) ▶ knn1= KNeighborsRegressor(n_neighbors=5)
```

```
: KNeighborsRegressor(n_neighbors=1) ▶ knn1.fit(X_train,y_train) ▶ knn1= KNeighborsRegressor(n_neighbors=10)
```

```
▶ knn1.predict(X_test)]: KNeighborsRegressor() ▶ knn1.fit(X_train,y_train)
```

```
: array([3., 4., 4., ..., 4., 5.]) ▶ knn1.predict(X_test): KNeighborsRegressor(n_neighbors=10)
```

```
▶ knn1.score(X_test,y_test)]: array([3. , 4. , 4. , ..., 4. , 5. ]) ▶ knn1.predict(X_test)
```

```
: 0.8746295087371968 ▶ knn1.score(X_test,y_test): array([3.4, 4.3, 3.7, ..., 4. , 4.1, 5. ])
```

```
]: 0.7570139706165779
```

```
▶ knn1.score(X_test,y_test)
```

```
: 0.6158858349728991
```

**KNeighborsRegressor는  
n\_neighbors상승할수록 낮은확률을  
가진다는것을 알수있습니다.**



### 3. 기계학습(Classifier)

```
rf=RandomForestClassifier(n_estimators=1)
```

```
rf.fit(X_train,y_train)
```

```
: RandomForestClassifier(n_estimators=1)
```

```
rf.predict(X_test)
```

```
: array([3, 4, 4, ..., 4, 5, 5], dtype=int64)
```

```
rf.score(X_train,y_train)
```

```
: 0.9930350070560386
```

```
rf=RandomForestClassifier(n_estimators=8)
```

```
rf.fit(X_train,y_train)
```

```
: RandomForestClassifier(n_estimators=8)
```

```
rf.predict(X_test)
```

```
: array([3, 4, 4, ..., 4, 5, 5], dtype=int64)
```

```
rf.score(X_train,y_train)
```

```
: 0.9998634315109027
```

```
rf=RandomForestClassifier(n_estimators=15)
```

```
rf.fit(X_train,y_train)
```

```
: RandomForestClassifier(n_estimators=15)
```

```
rf.predict(X_test)
```

```
: array([3, 4, 4, ..., 4, 5, 5], dtype=int64)
```

```
rf.score(X_train,y_train)
```

```
: 1.0
```

RandomForestClassifier는 n\_estimators 숫자가 15까지만 상승해도 100프로의 확률을 가질수있다는것을 알수있습니다.

### 3. 기계학습(Classifier)

```
dt1=DecisionTreeRegressor(random_state=1)
```

```
dt1.fit(X_train,y_train)
```

```
DecisionTreeRegressor(random_state=1)
```

```
dt1.predict(X_test)
```

```
array([3., 4., 4., ..., 4., 5., 5.])
```

```
dt1.score(X_test,y_test)
```

```
1.0
```

```
dt1=DecisionTreeRegressor(random_state=100)
```

```
dt1.fit(X_train,y_train)
```

```
DecisionTreeRegressor(random_state=100)
```

```
dt1.predict(X_test)
```

```
array([3., 4., 4., ..., 4., 5., 5.])
```

```
dt1.score(X_test,y_test)
```

```
1.0
```

**DecisionTreeRegressor는 random\_state 값을 변경해도 100프로의 확률을 가지고있다는것을 알수있습니다.**

# 4. 비지도 학습

```
kmeans=KMeans(n_clusters=5)
```

```
kmeans.fit(X)
```

```
KMeans(n_clusters=5)
```

```
kmeans.predict(X)
```

```
array([1, 4, 0, ..., 2, 2, 2])
```

```
df['클러스터번호']=kmeans.predict(X)  
df
```

예측한 클러스터 값을 데이터  
프레임에 추가시켜줍니다

생활용품\_지출\_총금액    의료비\_지출\_총금액    교통\_지출\_총금액    여가\_지출\_총금액    문화\_지출\_총금액    교육\_지출\_총금액    유흥\_지출\_총금액    클러스터번호

```
df.groupby('클러스터번호')['상권_코드', '월_평균_소득_금액'].mean()  
  
<ipython-input-231-26db27756adb>:11: FutureWarning: Indexing with a  
labeled Series is deprecated. Use a list instead.  
df.groupby('클러스터번호')['상권_코드', '월_평균_소득_금액'].mean()
```

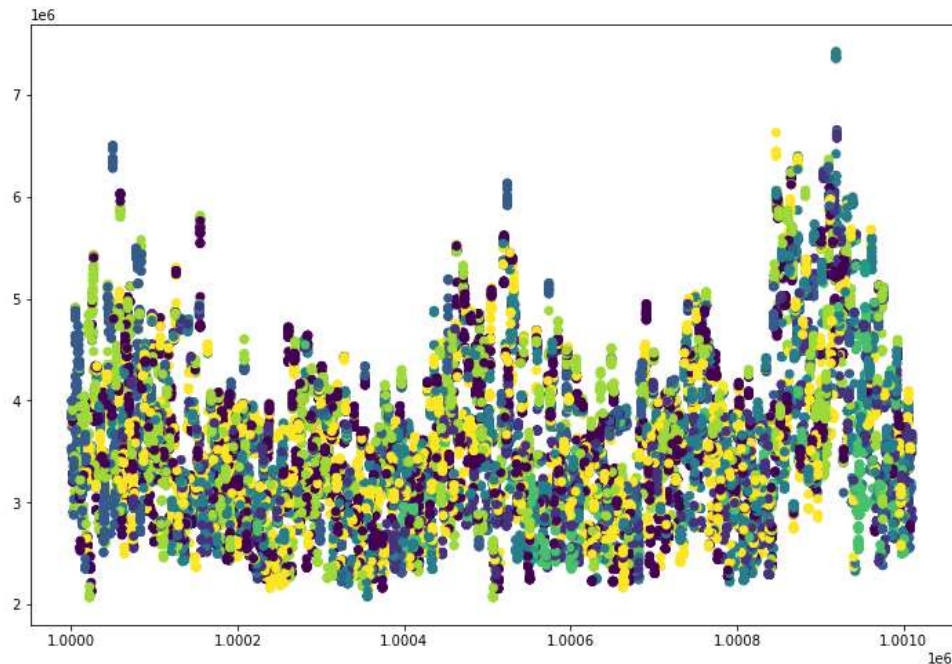
	상권_코드	월_평균_소득_금액
클러스터번호		
0	1.000461e+06	3.534696e+06
1	1.000598e+06	3.431527e+06
2	1.000358e+06	3.760763e+06
3	1.000561e+06	3.488138e+06
4	1.000716e+06	3.446421e+06
5	1.000678e+06	3.202375e+06
6	1.000439e+06	3.675284e+06
7	1.000486e+06	3.477458e+06

클러스터 번호 기준으로 상권코드와  
월평균소득금액의 평균을 구해봅니다.

# 4. 비지도 학습

```
plt.figure(figsize=(12,8))  
plt.scatter(df['상권_코드'],df['월_평균_소득_금액'],c=df['클러스터번호'])
```

<matplotlib.collections.PathCollection at 0x15f613c820>



- 그래프의 결과로는 상권코드중에서 가장 높은 월평균소득을 나타내는 상권코드와 가장 적은 월평균소득을 나타내는 상권코드를 알수있었습니다.
- 그리고 클러스터 범위 2 ~ 4이 가장 많은 월평균소득분위를 차지하고 있다는 것을 그래프를 통해 알 수 있었습니다.

## 5. 결론(분석결과)

- 서울시에서 가장 사람들이 많이가는 카페는 “스타벅스”, “이디야커피”인것을 알수있었습니다.
  - “스타벅스”, “이디야커피”가 인기가 많은 이유를 알 수 있었습니다.
- 스타벅스”와 “이디야커피 ” 는 주로 한강주변인 올림픽대로와 강변북로 주변에 많이 위치하고 있었습니다. 하지만 스타벅스매장의 수가 많은곳에는 이디야커피의 매장수가 상대적으로 적었고 반대로 이디야커피의 매장수가 많은곳은 스타벅스의 매장수가 상대적으로 적은것을 알 수 있었습니다.
- 서울의 핫플레이스TOP3를 알게 되었고 TOP3의 위치 정보와 그 곳에 위치한 정보를 간단하게 예측을 해볼수있었습니다.
  - 7등급,8등급의 상권구분코드명은 “골목상권”으로 출력된것을 알수있었습니다.
- 기계학습으로는 KNeighborsRegressor 의 n\_neighbors , RandomForestClassifier 의 n\_estimators 을 변경하면 확률이 달라지는것을 알수있었습니다 하지만DecisionTreeRegressor는 random\_state 값을 변경해도 확률이 일정하다는것을 알수있었습니다.
- 비지도학습을 통해서는 가장높고,낮은 월평균소득을 나타내는 상권코드를 알수있었고, 클러스터 범위 2 ~ 4이 가장 많은 월평균소득분위를 차지하고 있다는 것을 그래프를 통해 알 수 있었습니다.
- 분석결과를 통해 서울의 핫플레이스가 어디인지 모르는 친구들에게 데이터 자료를 통해 핫플레이스가 어디인곳을 편하게 알려줄 수 있습니다.