**막대도표** - Series 데이터,

ax = data.plot.bar()

ax.set\_xlabel(‘’)

plt.show()

----------------------------------------------------------------------------------------------------

**matplotlib** 활용 – Series 데이터

plt.bar(data.index , data, color=’’)

plt.xlabel('')

plt.show()

----------------------------------------------------------------------------------------------------

fig, ax = plt.subplots()

ax.bar(data.index , data , color='')

ax.set\_xlabel('')

plt.show()

----------------------------------------------------------------------------------------------------

**seaborn** 활용 – Series 데이터

sns.countplot(x='범주형1', hue='범주형', data=data, order =data[범주형1].value\_counts().index)

plt.show() # hue\_order도 있다.

----------------------------------------------------------------------------------------------------

**범주별 중심위치측도의 barplot- 신뢰구간 그리기**

sns.barplot(x='범주형1', y=’수치형1', data = data, ci=95, capsize=0.5,

dodge=False, # dodge: 간격, 안하면 hue 가 0 일때 1일때 **간격**이 띄어져있다)

plt.show() #ci=confidence interval

신뢰구간의 너비를 보면 좋은 점

- 분산의 크기를 알 수 있다.

(신뢰구간이 좁을수록 분산이 작다)

----------------------------------------------------------------------------------------------------

**가로로 barplot그리기**

data.plot(kind='barh', grid=True, figsize=(10,10)) # grid: 범위 별 경계선

plt.title(' ')

plt.show()

===========================================================

**원그래프** - Series데이터

ax=data.plot.pie()

ax.set\_xlabel('')

plt.show()

--------------------------------------------------------------------------------------------------------

**sunburst그래프 그리기**

import plotly.express as px

fig = px.sunburst(df,path=['continent', 'country'], values='pop', color='lifeExp', hover\_data=[‘’], color\_continuous\_scale='RdBu', color\_continuous\_midpoint=np.average(df['lifeExp'], weights=df['pop']), range\_color=[0,250000])

fig.show()

===========================================================

**그래프 두개 그리는 법**

**Barplot 과 pieplot 그리기**

fig, axes = plt.subplots(1,2, figsize=(12,6))

axes[0].bar(data.index, data , color=’’)

axes[0].set\_xlabel(‘’)

axes[1].pie(data, labels = data.index, autopct = "%.1f%%", colors=’’) #"%.1f%%" 백분율

plt.show()

----------------------------------------------------------------------------------------------------

**Countplot 두개 그리기**

fig, axes = plt.subplots(1,2, figsize=(12,6))

sns.countplot(x=열1, hue='열2', data=data, ax = axes[0])

sns.countplot(x= 열1, hue='열2', data=data, ax =axes[1], order = data[열1].value\_counts().index)

plt.show()

===========================================================

**두 범주형 변수의 관계 요약**

pd.crosstab(data.열1, data.열2, margins=True, normalize = 'index') #count만 가능한 것

**범주 별 수치형 변수 피벗테이블**

1. data.pivot\_table(index='범주1',

columns='범주2', values='수치1',aggfunc='sum', margins=True)’

2. pd.pivot\_table(data, index=’범주형’,aggfunc=)

#sum, mean, max, min, count 다 가능하다.

----------------------------------------------------------------------------------------------------

**xtab= crosstab, pivot\_table데이터시각화**

fig, axes = plt.subplots(1,2, figsize=(12,5))

xtab.plot.bar(ax=axes[0], legend=True, ylabel=' ')

xtab.T.plot.bar(stacked = True, ax = axes[1], ylabel = ' ')

axes[1].legend(bbox\_to\_anchor=(1.0, 1.0))

plt.show()

# legend 유무, label, stacked 유무에 관한 것들

==============================================================

**히스토그램** – Series

ax = data.plot.hist(density=True, xlim=[0, 12], bins=range(1,12), figsize=(4, 4))

data.plot.density(ax=ax)

ax.set\_xlabel('Murder Rate (per 100,000)')

plt.show()

#density는 정규화, xlim은 보고싶은 범위, density는 밀도곡선을 그려주는 함수

----------------------------------------------------------------------------------------------------

**정규분포에 얼마나 근접한지 판단**

-점들이 직선에 가까울수록 정규분포에 근사

stats.probplot(data, dist="norm", plot=plt)

plt.show()

# 밑으로 볼록한 모양은 right skewed모형으로 왼쪽으로 치우친, 오른쪽 늘어진 모형이 된다.

===========================================================

**Boxplot**

fig, ax = plt.subplots(figsize=(5, 5))

sns.boxplot(x="범주형열", y="수치형열", data = data, color='white')

ax.set\_xlabel('')

ax.set\_ylabel('')

plt.show() #violinplot도 같은 방법으로 가능

----------------------------------------------------------------------------------------------

plt.figure(figsize=(12,10))

plt.boxplot((state.Population/1000000))

plt.show()

===========================================================

**산점도(pairplot)**

# Pariplot은 변수들 간의 상관관계를 보기 위해 사용한다(= corr()을 시각화한 것)

sns.pairplot(data, corner=True, kind='reg', vars=[보고싶은 수치형들])

sns.pairplot(data=data,y\_vars='reviews\_per\_month',x\_vars=['review\_scores\_rating','review\_scores\_accuracy','review\_scores\_cleanliness'],kind='reg')

plt.show() #corner 대각선을 기준 한쪽 없애기

y\_vars=보고싶은 범주형1개, x\_vars=[보고싶은 범주형들]-한줄로 보여지는 pairplot이다

===========================================================

**Scatter plot**

plt.figure(figsize=(14,10))

plt.scatter(data[수치형1], data[수치형2], s=50) # scatterplot

for n in range(data.shape[0]):

plt.text(data[수치형1][n]\*1.02, data [수치형2'][n]\*0.98, data.index[n], fontsize=15)

# scatterplot위에 각 점에 해당하는 index표시

plt.xlabel(), plt.ylabel(), plt.grid(), plt.show()

--------------------------------------------------------------------------------------------------------------

**Scatter plot (Bubble chart)**

import plotly.express as px

px.scatter(data\_frame=data, x=수치형열1,

y=수치형열2, color=범주형열1,

hover\_name=범주형열2, size=수치형열3, size\_maxs=50)

===========================================================

**Facet Grid**

grid1 = sns.FacetGrid(data, col=범주형1, # col 이 가로로 row가 세로로

row=범주형2)

grid1.map\_dataframe(sns.histplot, x=’’, y=’’)

plt.show()

-------------------------------------------------------------------------------------------------------------- grid1 = sns.FacetGrid(data, col=범주형1)

grid1.map\_dataframe(sns.barplot, x=범주형, y=수치형, palette='hls', ci=95, capsize=0.2)

grid1.set\_xticklabels(범주형1의 index, rotation='vertical') #작은 그래프들이 라벨들

grid1.set\_axis\_labels(x축 라벨, y축 라벨) #작은 그래프들의 x라벨과 y라벨

grid1.set\_titles(col\_template="{col\_name}", fontsize=15, fontweight='bold',y=1.1) #작은그래프 제목

plt.show()

==============================================================

**범주형 변수 종류 순서를 정하고 싶다면 …**

sort=data.범주형1.value\_counts().index.to\_list()

data.groupby(범주형)[수치형].mean().reset\_index().sort\_values(by=수치형,ascending=False).범주형.values.to\_list()

data.범주형1=data.범주형1.astype(‘category’)

data.범주형1=data.범주형1.cat.reorder\_categories(sort) 하면 된다.

===========================================================

**상관관계 시각화**

import matplotlib.font\_manager as fm

plt.rc('font', family=fm.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get\_name())

colormap = plt.cm.PuBu

plt.figure(figsize=(10, 8))

plt.title('review scores 와 월별 리뷰수 관계', y = 1.05, size = 15)

sns.heatmap(data.corr(), linewidths = 0.1, vmax = 1.0, square = True, cmap = colormap, linecolor = "white", annot = True, annot\_kws = {"size" : 16})

fig,ax=plt.subplots(figsize=(10,8))

sns.heatmap(data.corr(), cmap='RdYlGn\_r', annot=True, vmin=-0.5, vmax=1, linewidths=.5, annot\_kws = {"size" : 15})

#annot: 각셀에 숫자 입력

#vmin: 히트맵의 제일 작은 숫자

#vmax: 히트맵의 제일 큰 숫자

#linewidth: 흰선

#annot\_kws: 안에 숫자크기

================================================================

**클러스터 맵**

sns.clustermap(df.corr(),annot=True, vmin=-1, vmax=1)

----------------------------------------------------------------------------------------------------

**스피어만 상관계수 시각화**

spear=df.corr(method='spearman')

sns.heatmap(spear,vmin=-1,vmax=1,annot=True)

**# 스피어만 분석**

상관성이 선형적인 상관성만 존재하는 것이 아닌지 확인하기위해

스피어만 상관계수를 통한 heatmap도 그려보았는데 거의 유사한것으로 보아

현재의 데이터에서의 상관성들은 대부분 선형성을 띄는 상관성일 가능성이 높다.

--------------------------------------------------------------------------------------------------------------

#Pearson 상관계수 (correlation coefficient)

import scipy.stats as stats

stats.pearsonr(data.수치형1, data.수치형2)

- 표준편차로 나누어 주어 언제나 -1과 1 사이의 값을 가진다.

- 직선관계의 정도를 나타낸다.

cor>0: 같은 방향으로 움직이는 경향

cor<0: 반대 방향으로 움직이는 경향

±1 에 가까울 수록 (즉, 절대값이 클수록) 강한 상관관계

cor=1: 완벽한 직선관계를 의미

cor=0에 가까울 수록 관계없음

===========================================================

**# 잡것들**

palette='hls' #색깔 바꾸는 것

color=['black', 'lightgrey'] # 색깔 정해주는 것 – 출력되는 종류가 두가지면 두가지 정하기

ax.set\_xticklabels(rotation='vertical')

#하나의 그래프만 적용되는 코드#수직

plt.xticks(fontsize=12, rotation='vertical') 그래프 두개라면 두개 중 뒤에껀 이걸로 바꿀 수 있음

#두개의 그래프 모두 적용되는 코드 x축 돌리기#수직

fig.autofmt\_xdate(rotation='vertical')

시각화하였는데 분석하기 어렵다면 log변환을 해보자. – data[수치형]=np.log(data[수치형])

===========================================================

**아노바 분석**

import statsmodels.api as sm

import statsmodels.formula.api as smf

from statsmodels.formula.api import ols

from statsmodels.stats.anova import anova\_lm

model = smf.ols('reviews\_per\_month ~ most\_freq\_aspect', data = data).fit()

display(model.summary().tables[1])

display(anova\_lm(model))

================================================================

np.sum( data , axis=1), data.sum(axis=1) 🡪 행을 기준으로 모든 열을 더하는 것

subway.loc[:,subway.columns.str.contains('승차')] 🡪 특정단어 포함된 열 뽑기

data.reset\_index(inplace=True) 🡪 행 이름을 열로 빼고 싶을 때

plt.title('제목달기',FontSize=20) #제목

plt.legend(bbox\_to\_anchor=(1,1)) #범례(bbox는 위치조정(0,1)(1,0) 등도 할수 있음)

plt.xlabel('x라벨',FontSize=15) #x라벨 (fontsize: 글자크기)

plt.ylabel('y라벨',FontSize=15) #y라벨

plt.xticks(rotation='vertical') #x눈금자 (rotation: 글자를 돌림) 수직

plt.grid() #눈금칸

plt.show()

ucb.pivot\_table(fill\_value=None, observed=False)

pd.crosstab(normalize=False)

#pd.crosstab 과 pivot\_table 의 차이는 normalize와 fill\_value, observed 이다. 굳이 pivot\_table 을 사용해야하나...?

================================================================모자이크

from statsmodels.graphics.mosaicplot import mosaic

plt.rcParams['figure.figsize'] = [15, 8]

mosaic(lc\_loans.sort\_values('grade'), ['grade', 'status'],axes\_label=True, gap=0.01)

plt.show()

================================================================ **MinMaxscaling**

from sklearn import preprocessing

min\_max\_scaler = preprocessing.MinMaxScaler()

x\_scaled = min\_max\_scaler.fit\_transform(data\_result)

data\_result\_norm =

pd.DataFrame(x\_scaled, columns = data\_result.columns, index = data\_result.index)

subway\_merge = pd.merge(subway\_raw, subwayfee, on="역ID")

**bar그래프 모양이 이상하다면 dodge=False 를 해보기**

**선 그래프**

sns.lineplot(x=population.Year, y=population.Population,marker='o')

plt.ylim(0,7) # y의 범위를 정해주는 것

plt.text(1955,7.2,'(단위: 인구 10억)') #단위를 달 때 x값과 y을 주면서 위치를 조정한 것

**계단식 그래프**

sns.lineplot(x='Year', y='Price', data=data, **drawstyle**='steps-post')

steps-post : 처음에 낮게 시작하는 것, steps-pre : 시작부터 올라가는 것

plt.xticks(np.arange(1991, 2011), rotation=90) # x범위 지정, x변수 이름 수직

================================================================**평활기법**

**이동평균 – 계절성이 없는 시계열에 사용**

**차분하기 – 시계열에서 추세 또는 계절적 패턴을 제거하는 단순하면서 대중적 방법**

data.x.diff()

data.x.diff(periods=12) # 월별 자료의 연간 패턴을 없애기 위해 시차12의 차분

**시계열 분해**

* **추세(Tt): 전체적으로 증가 혹은 감소하는 큰 흐름**
* **계절성(St): 짧은 기간에 규칙적으로 반복되는 사이클**
* **오차(Rt): random variation**

from statsmodels.tsa.seasonal import seasonal\_decompose

result = seasonal\_decompose(data.x, model='additive', period=12)

result.plot()

plt.show()

#종류별 그래프

result.seasonal(계절성의 데이터), result.trend(추세 데이터)

result.resid(추세와 계절성을 제거한 데이터)

**단순지수평활기법**

**지수가중이동평균, 평활상수 a**

**평활상수 a 가 1에 가까울수록 최근 관측치 예측이 큰 영향, 원래 값과 비슷해진다.**

**a가 0에 가까울수록 과거 관측치 예측이 큰 영향, 스무스해짐**

### pandas의 ewm 사용

fit\_ewm1 = data.x[-50:].ewm(alpha=0.2, adjust=False).mean()

fit\_ewm2 = data.x[-50:].ewm(alpha=0.6, adjust=False).mean()

원리 : alpha \* x(t=3) + (1- alpha) \* ewm(t=2)

-----------------------------------------------

**### statsmodels의 SimpleExpSmoothing 사용**

# 지수가중이동평균 두번째 방법, 위의 방법에서 하나씩 밀려나있다.

from statsmodels.tsa.api import SimpleExpSmoothing

fit1 = SimpleExpSmoothing(data.x[-50:]).fit(smoothing\_level=0.2) # optimized

fit2 = SimpleExpSmoothing(data.x[-50:]).fit(smoothing\_level=0.6)

fit1.fittedvalues – 수치를 확인할 수 있는 것

**SimpleExpSmoothing을 사용하면 예측(forecast) 을 할 수 있다.**

fcast1 = fit1.forecast(10), fcast2 = fit2.forecast(10)

모두 동일한 flat 형태를 지닌다.

----------------------------------------------------

# 종류별로 3가지 그래프 그리기

plt.figure(figsize=(8,5))

line1, = plt.plot(data.x[-50:], marker='o', color='black') ## 원래 그래프

line2, = plt.plot(fit1.fittedvalues, marker='o', color='blue') ## 0.2 그래프

line3, = plt.plot(fit2.fittedvalues, marker='o', color='red') ## 0.6 그래프

plt.plot(fcast1, marker='o', color='blue') # 0.2 로 예측한 그래프

plt.plot(fcast2, marker='o', color='red') # 0.6 으로 예측한 그래프

plt.legend([line1, line2, line3],['original', r'$\alpha=0.2$', r'$\alpha=0.6$'])

plt.show()

================================================================

**ARMA 모형**

* 변수의 과거 값의 선형 조합을 이용하여 관심있는 변수를 예측하는 AR모형과 과거 예측오차의 선형 조합을 이용하는 MA 모형을 결합한 형태의 모형

**ARMA 모형의 식별**

* 자기상관함수(Autocorrelation function; ACF): 𝑦𝑡와 𝑦𝑡−𝑘 간의 상관계수
* 편자기상관함수(Partial autocorrelation function; PACF): 다른 모든 짧은 시차 항에 따라 조정한 후 𝑦𝑡와 𝑦𝑡−𝑘 간의 상관계수
* ACF와 PACF의 모양을 참고하여 ARMA 모형의 차수(𝑝,𝑞)를 식별

|  | **AR(p)** | **MA(q)** | **ARMA(p, q)** |
| --- | --- | --- | --- |
| ACF | Tails off | Cuts off after lag q | Tails off |
| PACF | Cuts off after lag p | Tails off | Tails off |

## 몇번째에서 Cuts off 를 했는지 보고 p 와 q를 정하면 된다.

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

fig, axes = plt.subplots(1,2, figsize=(12,3))

plot\_acf(data.x, ax=axes[0])

plot\_pacf(sse.x, ax=axes[1] )

plt.show() ## na 값이 있으면 그래프가 그려지지 않기 때문에 data.x.dropna()를 해야한다.

------------------------------------------------------------------

from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(data.x, order=(1,숫자,0)).fit() ## 숫자: 몇번 차분을 할 것이냐

**#### ARIMA 는 숫자만큼 차분을 하고 나서 ARMA를 돌리는 것을 말한다.**

model.summary().tables[0] ## 분석을 위한 표 (잘 모른다.)

fcast\_arima = model.forecast(10) ## 이 모델을 이용하여 예측도 할 수 있다.

-------------------------

# 그래프 그리기

plt.figure(figsize=(8,5))

line1, = plt.plot(sse.Close[-50:], marker='o', color='black')

line2, = plt.plot(model.fittedvalues[-50:], marker='o', color='blue')

line3, = plt.plot(fcast\_arima, marker='o', color='red')

plt.legend([line1, line2, line3],['original', 'ARIMA', 'Forecast'])

plt.show()

**Pandas Timestamp**

import pandas as pd

pd.Timestamp(year=2021, month=1, day=1, hour=5, minute=10, second=8, microsecond=99)

pd.Timestamp('2021/1/10'), pd.Timestamp('2021-1/10'), pd.Timestamp('Jan 10, 2021 20:45.56')

# **pd.to\_datetime**을 통해 dataframe의 column을 timestamp로 변환할 수 있음

pd.to\_datetime('2015-5-13'), pd.to\_datetime('2015-13-5', dayfirst=True) #일자를 먼저 입력할 때

data['REPORTED\_DATE'] = pd.to\_datetime(data.REPORTED\_DATE)

-------------------------------------------------------------

**시계열을 이용한 슬라이스**

**# Timestamp를 인덱스로 지정하여 효율적으로 데이터 슬라이싱**

data2= data.set\_index('REPORTED\_DATE')

data2.loc['2016-05-12 16:45:00'] / data2.loc['2016-05-12'] / data2.loc['2016-05'] / data2.loc['2016']

data2.loc['Dec 2015'] / data2.loc['2015-3-4':'2016-1-1'] / data2.loc['2015-03-04 22':'2015-03-05 22']

**# 인덱스로 지정하지 않았을 때 작동이 안될 때 있다. Between 대부분 활용할 수 있다.**

data[data.REPORTED\_DATE.between('2016-05-12', '2016-05-13')]

data[data.REPORTED\_DATE.between('2016-05', '2016-06')]

--------------------------------------------------------------------

## 날짜와 상관없이 시간 기준으로 행 선택 – 인덱스로 지정해야 가능하다

data2.between\_time('2:00', '5:00', include\_end=False) ## include\_end False : 5시 포함하지 않는 것

data2.at\_time('5:47')

------------------------------------------------------------------

## .first: 인덱스에서 정렬해야 사용 가능, pd.offsets 모듈에 있는 DataOffset 객체로 표시

data\_sort = data2.sort\_index() # 인덱스 정렬

data\_sort.first(pd.offsets.MonthBegin(6, normalize=True)) ## normalize 는 첫 시간을 0으로 할 것인가에 대한 것, True 로 하면 첫 시간을 0으로 보고 1월부터6월마지막날까지의 데이터 추출

data\_sort.first('6MS') ## 위에서 normalize=True 제외한 것과 동일한 것 - 6 month start

data\_sort.first('5D') # 5 days

data\_sort.first('5B') # 5 business days # 영업일 기준 5일 주말빼고

data\_sort.first('7W') # 7 weeks, with weeks ending on Sunday

data\_sort.first('3QS') # 3rd quarter start # 3분기 자료

data\_sort.first('A') # one year end

## .resample: 가능한 시간 범위별로 그룹화 (groupby와 비슷)

crime\_sort.resample('W').size() # 주별 요약: 일요일-토요일

crime\_sort.resample('W-THU').size() # 주별 요약: 목요일-수요일

## resample 을 이용하여 다른 자료도 집계를 할 수 있음

crime2.resample('Q')['IS\_CRIME', 'IS\_TRAFFIC'].sum()

# 사용 가능 함수 확인

r = crime\_sort.resample('W')

[attr for attr in dir(r) if attr[0].islower()]

---------------------------------------------------------------------------------------

**# .dt 속성을 사용해 요일과 연도를 추출할 수 있다.**

data.REPORTED\_DATE.dt.day\_of\_week #0:월 1:화 ... 6:일 요일 숫자

crime.REPORTED\_DATE.dt.day\_name() # 요일이름 영어로 나옴

crime.REPORTED\_DATE.dt.year # 연도 추출

cf)

#sub: 빼기, round: 소수자리, mul: 곱하기

data.div(crime\_begin).sub(1).round(2).mul(100)

# 인덱스 요일이 숫자로 되어있을 때 문자로 바꾸는 것

crime\_week.index = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',

'Friday', 'Saturday', 'Sunday']

# .unstack('day')) 의 효과

crime.groupby([crime.REPORTED\_DATE.dt.year.rename('year'),

crime.REPORTED\_DATE.dt.day\_of\_week.rename('day')]).size()

(crime.groupby([crime.REPORTED\_DATE.dt.year.rename('year'),

crime.REPORTED\_DATE.dt.day\_of\_week.rename('day')]).size()**.unstack('day'))**

**-------------------------------------------------------------------------------------------------------------**

# 20분 이내에 이루어진 연속적인 것을 찾는 것( True, False 형태로)

data['NewSession']=data['Time Stamp'].diff()>pd.Timedelta(minutes=20)

#Timestamp 시각, Timedelta 시간

# 이를 이용해 20분 이내에 이루어진 것들을 같은 값으로 보기( 20분 내의 클릭한 것을 하나로)

data['SessionID']=data['NewSession'].cumsum()

data\_session=data.groupby('SessionID').first() # 첫번째 접속한 것만 남기기

# 이를 함수를 만들어 사용하기( 처음에 나오는 것이 Falsgde로 나오는 것을 해결 한 것)

def add\_newsession(df):

df['Newsession'] = df['Time Stamp'].diff()>pd.Timedelta(minutes=20)

df['Newsession'].iloc[0] = True

return df

data = data.groupby('KMID').apply(add\_newsession)

data['SessionID'] = data['Newsession'].cumsum()

# 체류시간 구하기

data.groupby('SessionID')['Time Stamp'].max()-data.groupby('SessionID')['Time Stamp'].min()

# 시간을 초단위로

data\_session['SessionTime']=data\_session['SessionTime'].dt.total\_seconds()

# 특정 단어 포함하는지 확인하는 방법

data.loc[data['Full Title'].str.contains('쇼핑')]

--------------------------------------------------------------------------------------------------------------

한글 깨질 때

import matplotlib.font\_manager as fm

plt.rc('font', family=fm.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get\_name())

색 정할 때

from matplotlib import cm

colors1=cm.get\_cmap('Set2')(np.arange(7))

================================================================

**구글 맵스**

import googlemaps

gmaps\_key = "AIzaSyDcfUXo-ILaA8jg6pUxFbYH8kgLAiIX4SE" # 자신의 key를 사용

gmaps = googlemaps.Client(key=gmaps\_key)

## 전체 주소와 각 위도 경도 정보 저장하기 ##

station\_address = []

station\_lat = []

station\_lng = []

for name in station\_name:

tmp = gmaps.geocode(name, language='ko')

station\_address.append(tmp[0].get("formatted\_address")) ## 전체 주소 정보

tmp\_loc = tmp[0].get("geometry") ## 위치 정보, 타입

station\_lat.append(tmp\_loc['location']['lat']) ## 위도

station\_lng.append(tmp\_loc['location']['lng']) ## 경도

print(name + '-->' + tmp[0].get("formatted\_address"))

**Folium**

import folium

gmaps.geocode("국민대학교", language='ko') ## 국민대학교 정보

folium.Map(location=[37.6108694, 126.9972889], zoom\_start=15, tiles= 지정안하면 기본 지도)

## tiles에 'Stamen Toner': 검정색으로 된 지도, 'Stamen Terrain': 약간 컬러감 있음

## 구름마커를 map에 추가하는 것

folium.Marker([37.612216, 127.008154], popup='북한산보국문역',

icon=folium.Icon(icon='cloud')).add\_to(map) ## 팝업이 뜨는 아이콘 마커를 map에 추가하는 것

## 사이즈가 30인 원 마커를 map에 추가하는 것

folium.CircleMarker([37.6108117, 126.9976303], radius=30,

popup='국민대학교 경영관', color='#3186cc', fill\_color='#3186cc',).add\_to(map)

## sides가 5인 다각형 마커를 map에 추가하는 것

folium.RegularPolygonMarker([37.602667, 127.0133464],

popup='정릉역', fill\_color='#132b5e',

number\_of\_sides=5, radius=10).add\_to(map)

import json

geo\_path = '../Tamdata/02. folium\_us-states.**json'**

geo\_path=json.load(open(geo\_path, encoding = 'utf-8'))

import folium

map = folium.Map(location=[37.5502,126.982], zoom\_start=11)

folium.Choropleth(geo\_data=geo\_path,

data=crime\_anal\_norm['범죄'],

columns=[crime\_anal\_norm.index, crime\_anal\_norm['범죄']],

fill\_color='PuRd',

key\_on='feature.id').add\_to(map) ## feature.id 와 index가 같음

json파일에서 feature 안의 id를 뜻한다.

for n in range(len(data)): ##index가 번호로 되어있어서 그렇다.(= data[0].shape)

folium.CircleMarker([crime\_anal\_raw['lat'][n],crime\_anal\_raw['lng'][n]],

radius=crime\_anal\_raw['검거'][n]\*10,

color='#3186cc',fill\_color='#3186cc',fill=True).add\_to(map)

map ## 지도에 지형 색칠하고 위치마다 원을 그리는데 원의 크기는 검거율에 맞춘다.

# 문자 붙어있는거 나눠서 첫번째꺼를 숫자로 바꿔주는 함수

eq['위도']=eq['위도'].str.split().str[0].astype('float64')

# 회귀 문제 그래프

# 라인 그래프(line)

sns.regplot(x='새순max', y='착과량(int)', data=df\_train\_v3, dropna=True)

plt.show()

# 전체적인 분포 등고선

sns.regplot(x='엽록소mean2', y='착과량(int)', data=df\_train\_v3\_1.iloc[:])

sns.kdeplot(x='엽록소mean2', y='착과량(int)', data=df\_train\_v3\_1.iloc[:])

plt.show()

# 분포 색깔로 구분(색으로 구분)

sns.swarmplot(x='엽록소mean2', y='착과량(int)', data=df\_train\_v3\_1.iloc[:])

plt.show()