import numpy as np  
import pandas as pd  
  
#sample data  
np.random.seed(48)  
df = pd.DataFrame(10\*np.random.rand(4,2).round(2))  
df.columns = ["A","B"]  
df.A[1] = np.nan # adding nan'  
df.isnull() #check missing value  
df.isnull().sum()  
df2 = df.dropna() # delete row  
df3 = df.dropna(axis=1) # delete col  
df4 = df.dropna(how="all") # show all  
df.B[1]= np.nan  
print(df.dropna(how="all")) # 행, 열이 전체 nan이면 삭제 아니면 그대로  
df5 =pd.DataFrame([[np.nan,np.nan,np.nan],[1,np.nan,np.nan],[2,3,np.nan],[4,5,6],[np.nan,7,8]])  
print(df5.dropna(thresh=3,axis=1)) # col에 nan이 아닌 데이터가 3개 이상 존재: 삭제x/ 아닌 경우 col 삭제  
df.fillna(0).head() #nan부분을 0으로 채움  
df.fillna(method="ffill").head() # 위의(이전)값이 들어온다  
df.fillna(method="bfill").head() # 아래(다음)값이 들어온다  
df.fillna(method="bfill", axis=1).head() # 오른쪽 값이 들어온다  
print(df.mean(axis=0).round(2))   
print(df.mean(axis=1).round(2))  
df.fillna(df.mean().round(2)).head()

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
np.random.seed(5)  
n\_sample = 1000  
height = 4\*np.random.randn(n\_sample).round(2) + 170 #mean : 170 표준편차 :4  
weight = 5\*np.random.randn(n\_sample).round(2) + 65  
df\_raw=pd.DataFrame({"height":height,"weight":weight})  
print(df\_raw[:5])  
#copy data  
df=df\_raw.copy()  
sns.distplot(df.height.values)  
  
#standardization  
from sklearn.preprocessing import StandardScaler  
scale = StandardScaler()  
df[['h\_sc','v\_sc']]=scale.fit\_transform(df[['height','weight']])  
print(df[:5])  
#reverse transform  
scale.inverse\_transform(df[['h\_sc','v\_sc']])[:5]  
#히스토다이어그램  
sns.distplot(df.h\_sc.values)  
df[['height','weight']].plot.kde()  
df[['h\_sc','v\_sc']].plot.kde() # 표준화 그래프 비교

**1.Clustering**  
High intra-class 🡪 cohesive

Low inter-class 🡪 distinctive

Connectivity-based (공 모양 연결, density) / Distance-based(유클리드안)

텍스트, 시계, 손목시계이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

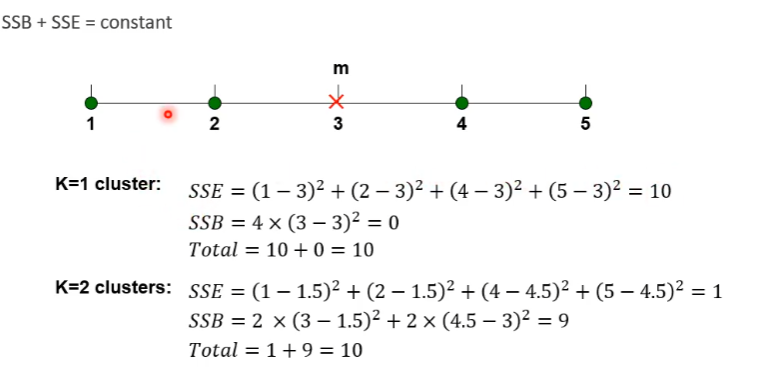
자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



**# calculating SSE**

**## SSE with the initial clustring and centroid**

**print("initial SSE at iteration 0")  
sse = 0  
for i in range(k):  
  dist = np.linalg.norm(pixels[clusters == i] - centroids\_old[i], axis = 1)  
  sse += np.sum(dist\*\*2)**

**print("wrong angser: ", sse)  
## SSE with new centroids before re-assigning   
print("before re-assigning")**

**# SSE solution 1  
sse = 0  
for i in range(k):  
  dist = np.linalg.norm(pixels[clusters == i] - centroids\_new[i], axis = 1)  
  sse += np.sum(dist\*\*2)  
print("sol1: ", sse)**

**# SSE solution 2  
sse = 0  
for i in range(len(pixels)):  
# for i in range(3):  
  diff = pixels[i] - centroids\_new[clusters[i]]  
  sse += np.sum(diff\*\*2)**

**print("sol2: ", sse)**

**### SSE with new centroids after re-assigning (this was the intended solution)  
distances = np.zeros((pixels.shape[0],k))  
for i in range(k):  
        distances[:,i] = np.linalg.norm(pixels - centroids\_new[i], axis=1)  
clusters = np.argmin(distances, axis = 1)**

**print("after re-assigning")**

**# SSE solution 1**

**sse = 0  
for i in range(k):  
  dist = np.linalg.norm(pixels[clusters == i] - centroids\_new[i], axis = 1)  
  sse += np.sum(dist\*\*2)  
print("sol1: ", sse)**

**# SSE solution 2|  
sse = 0  
for i in range(len(pixels)):  
# for i in range(3):  
  diff = pixels[i] - centroids\_new[clusters[i]]  
  sse += np.sum(diff\*\*2)  
print("sol2: ", sse)  
-------------------------------------------------**from sklearn.cluster import KMeans  
from matplotlib.pyplot import imshow  
import cv2  
import numpy as np  
import imutils

# load an image from a url

img\_url = 'https://snappygoat.com/b/5e3c07f2560b6420543d8e2367a70d3cdb08c39e'  
img = imutils.url\_to\_image(img\_url)  
img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
pixels = img.reshape((-1,3))  
pixels = np.float32(pixels)

# TODOs

k = 3  
model = KMeans(n\_clusters = k, random\_state=42)  
model.fit(pixels)  
labels = model.labels\_  
pixels\_new = np.zeros(pixels.shape)  
cluster\_centers = model.cluster\_centers\_  
for i in range(k):  
  pixels\_new[labels==i] = cluster\_centers[i]

pixels\_new = np.uint8(pixels\_new)  
result\_img = pixels\_new.reshape(img.shape)  
imshow(result\_img)

**2. KNN**  
fromsklearn.neighborsimportKNeighborsClassifier  
kn=KNeighborsClassifier(n\_neighbors=3,metric='euclidean')  
kn.fit(fish\_data,fish\_target)  
newFish=[25,200]  
kn.predict([newFish])  
train\_input,test\_input,train\_target,test\_target=train\_test\_split(fish\_data,fish\_target,stratify=fish\_target,random\_state=42)

mean=np.mean(train\_input,axis=0)std=np.std(train\_input,axis=0)train\_scaled=(train\_input-mean)/std  
kn.fit(train\_scaled,train\_target)  
newFish\_scaled=(newFish-mean)/stdprint(kn.predict([newFish\_scaled]))  
kn=KNeighborsClassifier(algorithm='brute',metric='mahalanobis',metric\_params={'V':np.cov(train\_input)})kn.fit(train\_input,train\_target)

<https://colab.research.google.com/drive/1OlCZN9u81DLAte_JnJXnzAQGTxHNYOSt?usp=sharing>

3. Linear Regression

Import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
datum = pd.read\_cvs(‘url’,index\_col = “id”)  
datum.head()  
plt.scatter(datum[‘friends’],datum[‘minutes’])  
plt.show()  
sns.regplot(x=’friends’,y=’minutes’,data=datum[[‘friends’,’minutes’]])  
plt.show()

#calculating Correlation Coeffiecient  
data[‘friends’].corr(data[‘minute’])

1에 가까우면 strong linear relation // -1에 가까울수록 negative //0에가까우면 weak relation

테이블이(가) 표시된 사진

자동 생성된 설명

Simple linear Regression : Regression & Prediction / Numerical

나이즈베이 : Classification / Categorical

Logistic Regression : Classification / Categorical