|  |  |
| --- | --- |
|  | |
| **Team 6 Programing homework #2 report** | |
|  | |
| **member** |  |
|  | 민우혁(Min Woohyuck) software  이민수(Lee Minsoo) software  양소영(Yang Soyoung) electronic engineering |
|  | |

|  |  |
| --- | --- |
|  | |
| **Index** |  |
|  | 1. Python codes. |
| 1. Explaination and Interpretation of 1st result. |
| 1. Explaination and Interpretation of 2nd result. |
| 1. Each members opinions and culclusion. |
| 1. Refferance. |
|  | |

|  |
| --- |
|  |
| **Python codes.** |
|  |

|  |
| --- |
|  |
| Lab2.py |
| from matplotlib import pyplot as plt  from sklearn.metrics import accuracy\_score  import pandas as pd  import numpy as np  from sklearn import preprocessing  from sklearn.mixture import GaussianMixture  from sklearn.cluster import DBSCAN  from sklearn.cluster import estimate\_bandwidth  from sklearn.cluster import MeanShift  import math  import time  import warnings ; warnings.filterwarnings('ignore')  import seaborn as sns  from sklearn.metrics import accuracy\_score, silhouette\_samples, silhouette\_score  from sklearn import preprocessing  from sklearn.model\_selection import cross\_val\_score, StratifiedKFold  skf = StratifiedKFold(n\_splits=10)  from sklearn.model\_selection import KFold  from sklearn.model\_selection import cross\_val\_score  from sklearn.model\_selection import GridSearchCV  import sys  from sklearn.cluster import KMeans  ################  #make result table  score\_sample = {'Group':["Group"],'Scaler':["Sample"], 'Encoder':["Sample"], 'Model':["Sample"],'Best\_para':["Sample"], "Score":[1]}  score\_results = pd.DataFrame(score\_sample)  score\_sample2 = {'type':["eeror"],'info':["info"]}  error\_data = pd.DataFrame(score\_sample2)    #for scale and encorde  class PreprocessPipeline():      def \_\_init\_\_(self, num\_process, cat\_process, verbose=False):          #super(PreprocessPipeline, self).\_\_init\_\_()          self.num\_process = num\_process          self.cat\_process = cat\_process          #for each type          if num\_process == 'standard':              self.scaler = preprocessing.StandardScaler()          elif num\_process == 'minmax':              self.scaler = preprocessing.MinMaxScaler()          elif num\_process == 'maxabs':              self.scaler = preprocessing.MaxAbsScaler()          elif num\_process == 'robust':              self.scaler = preprocessing.RobustScaler()          else:              raise ValueError("Supported 'num\_process' : 'standard','minmax','maxabs','robust'")          if cat\_process == 'onehot':              self.encoder = preprocessing.OneHotEncoder(sparse=False, handle\_unknown='ignore')          elif cat\_process == 'ordinal':              self.encoder = preprocessing.OrdinalEncoder()          else:              raise ValueError("Supported 'cat\_process' : 'onehot', ordinal'")          self.verbose=verbose            #do Preprocess      def process(self, X):          X\_cats = X.select\_dtypes(np.object).copy()          X\_nums = X.select\_dtypes(exclude=np.object).copy()          #Xt\_cats = Xt.select\_dtypes(np.object).copy()          #Xt\_nums = Xt.select\_dtypes(exclude=np.object).copy()          if self.verbose:              print(f"Categorica Colums : {list(X\_cats)}")              print(f"Numeric Columns : {list(X\_nums)}")          if self.verbose:              print(f"Categorical cols process method : {self.cat\_process.upper()}")          X\_cats = self.encoder.fit\_transform(X\_cats)          #Xt\_cats = self.encoder.transform(Xt\_cats)          if self.verbose:              print(f"Numeric columns process method : {self.num\_process.upper()}")          X\_nums = self.scaler.fit\_transform(X\_nums)          #Xt\_nums = self.scaler.transform(Xt\_nums)          X\_processed = np.concatenate([X\_nums, X\_cats],1)          #Xt\_processed = np.concatenate([Xt\_nums, Xt\_cats], axis=-1)          return X\_processed  # do process on I want  class AutoProcess():      def \_\_init\_\_(self, verbose=False):            self.pp = PreprocessPipeline          self.verbose= verbose        def run(self, X,group):          methods = []          scores = []          print(X.shape)            for num\_process in ['standard','robust','minmax','maxabs']:              for cat\_process in ['ordinal','onehot']:                  if self.verbose:                      print("\n------------------------------------------------------\n")                      print(f"Numeric Process : {num\_process}")                      print(f"Categorical Process : {cat\_process}")                  methods.append([num\_process, cat\_process])                  pipeline = self.pp(num\_process=num\_process, cat\_process=cat\_process)                    X\_processed= pipeline.process(X)                    #print(X\_processed.shape)                  #Classifier part                  for model in ['k-mean','em','clarans','dbscan','mean-shift']:                      if self.verbose:                          print(f"\nCluster model: {model}")                      if model =='k-mean':                          k\_num = {2,3,4,5,7,10}                          for k in k\_num:                              c\_mdel = KMeans(n\_clusters=k)                              #print(X\_processed)                              c\_mdel.fit(X\_processed)                              sample = X.copy()                              sample['cluster'] = c\_mdel.labels\_                              sample\_score = silhouette\_samples(X\_processed,sample['cluster'] )                              sample['silhouette\_'] = sample\_score                              score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,'k='+str(k), str(sample['silhouette\_'].mean())]                      elif model == 'em':                          k\_num ={2,3,4,5,7,10}                          for k in k\_num:                              c\_mdel = GaussianMixture(n\_components=k, random\_state=0).fit(X\_processed)                              sample = X.copy()                              c\_mdel\_cluster\_labels = c\_mdel.predict(X\_processed)                              sample['cluster'] = c\_mdel\_cluster\_labels                              sample\_score = silhouette\_samples(X\_processed,sample['cluster'])                              sample['silhouette\_'] = sample\_score                              score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,'k='+str(k), str(sample['silhouette\_'].mean())]                        elif model == 'dbscan':                          esp = {0.01, 0.1, 0.2, 0.3, 0.5, 0.75 }                          ms = {2,3,5,7,10}                          for e in esp:                              for m in ms:                                  try:                                      c\_mdel = DBSCAN(eps = e, min\_samples=m)                                      sample = X.copy()                                      sample['cluster'] = pd.DataFrame(c\_mdel.fit\_predict(X\_processed))                                      sample\_score = silhouette\_samples(X\_processed,sample['cluster'])                                      sample['silhouette\_'] = sample\_score                                      score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,'eps: '+str(e)+'  m: '+str(m)+'  cluster: '+str(len(sample['cluster'].value\_counts())), str(sample['silhouette\_'].mean())]                                  except ValueError:                                      error\_data.loc[len(error\_data)]=['ValueError','eps: '+str(e)+'  ms: '+str(m)+'only one cluster']                      elif model == 'mean-shift':                          best\_bandwidth = estimate\_bandwidth(X\_processed)                          c\_mdel = MeanShift(bandwidth=best\_bandwidth)                          c\_mdel\_cluster\_labels = c\_mdel.fit\_predict(X\_processed)                          sample = X.copy()                          sample['cluster'] = c\_mdel\_cluster\_labels                          print('cluster labels type: ', np.unique(c\_mdel\_cluster\_labels))                          print('bandwidth값 : ',best\_bandwidth)                          sample\_score = silhouette\_samples(X\_processed,sample['cluster'])                          sample['silhouette\_'] = sample\_score                          print('aver sihouette\_: ' +str(sample['silhouette\_'].mean()))                          print(sample.groupby('cluster')['silhouette\_'].mean())                          score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,'bandwidth: '+str(best\_bandwidth), str(sample['silhouette\_'].mean())]            return  kfold = KFold(5, True, 1)  pd.set\_option('display.max\_row', 10000)  # Import the data file  df = pd.read\_csv('E:\PythonWorkSpace\s2\lab2\housing.csv', encoding='utf-8')  print(df.dtypes)  print(df.isna().sum())  ##setting data set  #ex = df.iloc[:,8]  df = df.drop('median\_house\_value',axis=1)  #df['median\_house\_value'] = ex.to\_numpy()  #fill na in total\_bedrooms  #I thought it was marked Na because the value is 0.  df = df.fillna(0)  print(df.isna().sum())  #group 1 room  X1 = df[['total\_rooms','total\_bedrooms']]  #group 2 where  X2 = df[['longitude','latitude','ocean\_proximity']]  #group 3 spec  X3 = df[['housing\_median\_age','total\_rooms','total\_bedrooms','ocean\_proximity']]  #group 4 eviroment  X4 = df[['population','households']]  #group 5 all  X5 = df  autoprocess = AutoProcess(verbose=True)  autoprocess.run(X1,'room')  autoprocess.run(X2,'where')  autoprocess.run(X3,'spec')  autoprocess.run(X4,'eviroment')  autoprocess.run(X5,'all')  print(score\_results)  sys.stdout = open('E:\PythonWorkSpace\score result.txt', 'w')  print(score\_results)  sys.stdout.close() |
| Explain |
| Output and store silhouette scores for all conditions (data group, scaler, encoder, parameter) situations. |

|  |
| --- |
|  |
| Lab2\_p.y |
| from matplotlib import pyplot as plt  from sklearn.metrics import accuracy\_score  import pandas as pd  import numpy as np  from sklearn import preprocessing  import warnings ; warnings.filterwarnings('ignore')  import seaborn as sns  from sklearn.metrics import accuracy\_score, roc\_auc\_score, f1\_score  from sklearn import preprocessing  from sklearn.model\_selection import cross\_val\_score, StratifiedKFold  from sklearn.model\_selection import KFold  from sklearn.model\_selection import cross\_val\_score  from sklearn.model\_selection import GridSearchCV  import sys  from pyclustering.cluster.clarans import clarans  from sklearn.cluster import KMeans, estimate\_bandwidth, MeanShift  from sklearn.metrics import silhouette\_score, silhouette\_samples  from sklearn.mixture import GaussianMixture  from sklearn.cluster import DBSCAN  ################  #make result table  score\_sample = {'Group':["Group"],'Scaler':["Sample"], 'Encoder':["Sample"], 'Model':["Sample"],'Best\_para':["Sample"], "Score":[1]}  score\_results = pd.DataFrame(score\_sample)  score\_sample2 = {'type':["eeror"],'info':["info"]}  error\_data = pd.DataFrame(score\_sample2)  #for scale and encorde  class PreprocessPipeline():  def \_\_init\_\_(self, num\_process, cat\_process, verbose=False):  # super(PreprocessPipeline, self).\_\_init\_\_()  self.num\_process = num\_process  self.cat\_process = cat\_process  # for each type  if num\_process == 'standard':  self.scaler = preprocessing.StandardScaler()  elif num\_process == 'minmax':  self.scaler = preprocessing.MinMaxScaler()  elif num\_process == 'maxabs':  self.scaler = preprocessing.MaxAbsScaler()  elif num\_process == 'robust':  self.scaler = preprocessing.RobustScaler()  else:  raise ValueError("Supported 'num\_process' : 'standard','minmax','maxabs','robust'")  if cat\_process == 'onehot':  self.encoder = preprocessing.OneHotEncoder(sparse=False, handle\_unknown='ignore')  elif cat\_process == 'ordinal':  self.encoder = preprocessing.OrdinalEncoder()  else:  raise ValueError("Supported 'cat\_process' : 'onehot', ordinal'")  self.verbose = verbose  # do Preprocess  def process(self, X):  X\_cats = X.select\_dtypes(np.object).copy()  X\_nums = X.select\_dtypes(exclude=np.object).copy()  # Xt\_cats = Xt.select\_dtypes(np.object).copy()  # Xt\_nums = Xt.select\_dtypes(exclude=np.object).copy()  if self.verbose:  print(f"Categorica Colums : {list(X\_cats)}")  print(f"Numeric Columns : {list(X\_nums)}")  if self.verbose:  print(f"Categorical cols process method : {self.cat\_process.upper()}")  X\_cats = self.encoder.fit\_transform(X\_cats)  # Xt\_cats = self.encoder.transform(Xt\_cats)  if self.verbose:  print(f"Numeric columns process method : {self.num\_process.upper()}")  X\_nums = self.scaler.fit\_transform(X\_nums)  # Xt\_nums = self.scaler.transform(Xt\_nums)  X\_processed = np.concatenate([X\_nums, X\_cats], 1)  # Xt\_processed = np.concatenate([Xt\_nums, Xt\_cats], axis=-1)  return X\_processed  # do process on I want  class AutoProcess():  def \_\_init\_\_(self, verbose=False):  self.pp = PreprocessPipeline  self.verbose = verbose  def run(self, X, group):  methods = []  scores = []  # print(X.shape)  # need dataframe, list of label, number of clusters  def makePlt23(df, label, k, count, title):  if(len(df.columns)==3):  enc=preprocessing.OrdinalEncoder()  op = enc.fit\_transform(df['ocean\_proximity'].to\_numpy().reshape(-1, 1))  df['ocean\_proximity'] = op  # list for store feature data for each cluster  store = [[[] for col in range(len(df.columns))] for row in range(k)]  for m in range(len(label)):  for n in range(k):  if (label[m] == n):  for o in range(len(df.columns)):  store[n][o].append(df.iloc[m:m + 1, o:o + 1].values[0][0])  c = ['b.', 'r.', 'g.', 'y.', 'c.', 'm.', 'k.']  if (len(df.columns) == 2):  plt.subplot(120 + count, title=title)  plt.xlabel(df.columns[0])  plt.ylabel(df.columns[1])  for p in range(k):  plt.plot(store[p][0], store[p][1], c[p])  if (len(df.columns) == 3):  plt.subplot(120 + count, projection='3d', title=title)  plt.xlabel(df.columns[0])  plt.ylabel(df.columns[1])  for p in range(k):  plt.plot(store[p][0], store[p][1], store[p][2], c[p])  def makePltBig(df, label, k):  if (len(df.columns) == 3):  enc = preprocessing.OrdinalEncoder()  op = enc.fit\_transform(df['ocean\_proximity'].to\_numpy().reshape(-1, 1))  df['ocean\_proximity'] = op  # list for store feature data for each cluster  store = [[[] for col in range(len(df.columns))] for row in range(k)]  for m in range(len(label)):  for n in range(k):  if (label[m] == n):  for o in range(len(df.columns)):  store[n][o].append(df.iloc[m:m + 1, o:o + 1].values[0][0])  if(len(df.columns)==2):  for j in range(int(k/9)+1):  for i in range(j\*9, (j+1)\*9, 1):  if(i<k):  plt.subplot(330+(i-(j\*9)+1), title='Cluster N.'+str(i+1))  plt.xlabel(df.columns[0])  plt.ylabel(df.columns[1])  plt.plot(store[i][0], store[i][1], '.')  plt.show()  if (len(df.columns) == 3):  for j in range(int(k / 9) + 1):  for i in range(j \* 9, (j + 1) \* 9, 1):  if (i < k):  plt.subplot(330 + (i - (j \* 9) + 1), projection='3d', title='Cluster N.' + str(i + 1))  plt.xlabel(df.columns[0])  plt.ylabel(df.columns[1])  plt.plot(store[i][0], store[i][1], store[i][2], '.')  plt.show()  for num\_process in ['maxabs']:  for cat\_process in ['ordinal']:  if self.verbose:  print("\n------------------------------------------------------\n")  print(f"Numeric Process : {num\_process}")  print(f"Categorical Process : {cat\_process}")  methods.append([num\_process, cat\_process])  pipeline = self.pp(num\_process=num\_process, cat\_process=cat\_process)  X\_processed = pipeline.process(X)  # print(X\_processed.shape)  # Classifier part  for model in ['k-mean', 'em', 'clarans', 'dbscan', 'mean-shift']:  if self.verbose:  print(f"\nCluster model: {model}")  if model == 'k-mean':  if group == 'room':  k\_num = {3, 5}  elif group == 'where':  k\_num = {4, 7}  elif group == 'eviroment':  k\_num = {5, 7}  countPlt=0  for k in k\_num:  countPlt=countPlt+1  c\_mdel = KMeans(n\_clusters=k)  # print(X\_processed)  c\_mdel.fit(X\_processed)  sample = X.copy()  sample['cluster'] = c\_mdel.labels\_  sample\_score = silhouette\_samples(X\_processed, sample['cluster'])  sample['silhouette\_'] = sample\_score  score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,  'k=' + str(k), str(sample['silhouette\_'].mean())]  print(group, num\_process, cat\_process, model, 'k=' + str(k),  str(sample['silhouette\_'].mean()))  print(sample.groupby('cluster')['silhouette\_'].mean())  makePlt23(X, sample['cluster'], k, countPlt, 'k='+str(k))  plt.show()  if model == 'em':  if group == 'room':  k\_num = {3, 5}  elif group == 'where':  k\_num = {5, 7}  elif group == 'eviroment':  k\_num = {3, 5}  countPlt = 0  for k in k\_num:  countPlt+=1  c\_mdel = GaussianMixture(n\_components=k, random\_state=0).fit(X\_processed)  sample = X.copy()  c\_mdel\_cluster\_labels = c\_mdel.predict(X\_processed)  sample['cluster'] = c\_mdel\_cluster\_labels  sample\_score = silhouette\_samples(X\_processed, sample['cluster'])  sample['silhouette\_'] = sample\_score  score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,  'k=' + str(k), str(sample['silhouette\_'].mean())]  print(group, num\_process, cat\_process, model, 'k=' + str(k),  str(sample['silhouette\_'].mean()))  print(sample.groupby('cluster')['silhouette\_'].mean())  makePlt23(X, sample['cluster'], k, countPlt, 'k='+str(k))  plt.show()  if model == 'clarans':  if group == 'room':  k\_num = {3, 5}  elif group == 'where':  k\_num = {5, 7}  elif group == 'eviroment':  k\_num = {3, 5}  countPlt=0  for k in k\_num:  countPlt+=1  sample = X[1400:2400].copy()  # make list to store each rows label  label = [0 for l in range(len(X[1400:2400]))]  # data, number of cluster, num local, max neighbor  clarans\_instance = clarans(X\_processed[1400:2400], k, 6, 4)  clarans\_instance.process()  clusters = clarans\_instance.get\_clusters()  # make label  for j in range(0, len(clusters), 1):  for i in range(0, len(clusters[j]), 1):  label[clusters[j][i]] = j  sample['cluster']=label  sample\_score = silhouette\_samples(X\_processed[1400:2400], sample['cluster'])  # print(k, 'clusters silhouette score :', score)  sample['silhouette\_']=sample\_score  score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model, 'k=' + str(k), str(sample['silhouette\_'].mean())]  print(group, num\_process, cat\_process, model, 'k=' + str(k), str(sample['silhouette\_'].mean()))  makePlt23(X[1400:2400], label, k, countPlt, 'k='+str(k))  plt.show()  if model == 'dbscan':  if group == 'room':  esp = {0.01}  ms = {3, 5}  elif group == 'where':  esp = {0.01, 0.75}  ms = {7, 10}  elif group == 'eviroment':  esp = {0.01}  ms = {3, 5}  for e in esp:  countPlt=0  for m in ms:  countPlt+=1  try:  c\_mdel = DBSCAN(eps=e, min\_samples=m)  sample = X.copy()  sample['cluster'] = pd.DataFrame(c\_mdel.fit\_predict(X\_processed))  sample\_score = silhouette\_samples(X\_processed, sample['cluster'])  sample['silhouette\_'] = sample\_score  score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,  'eps: ' + str(e) + ' m: ' + str(  m) + ' cluster: ' + str(  len(sample['cluster'].value\_counts())),  str(sample['silhouette\_'].mean())]  print(group, num\_process, cat\_process, model,  'eps: ' + str(e) + ' m: ' + str(m) + ' cluster: ' + str(  len(sample['cluster'].value\_counts())), str(sample['silhouette\_'].mean()))  sample['cluster'] = sample['cluster'] + 1  print(sample.groupby('cluster')['silhouette\_'].mean())  print(sample['cluster'].value\_counts())  # sample['cluster']=sample['cluster']+1  k=len(sample[['cluster']].groupby('cluster').count())  if(k<=7):  makePlt23(X, sample['cluster'], k, countPlt, 'eps='+str(e)+' m='+str(m))  else:  makePltBig(X, sample['cluster'], k)  except ValueError:  error\_data.loc[len(error\_data)] = ['ValueError', 'eps: ' + str(e) + ' ms: ' + str(  m) + 'only one cluster']  plt.show()  elif model == 'mean-shift':  best\_bandwidth = estimate\_bandwidth(X\_processed)  c\_mdel = MeanShift(bandwidth=best\_bandwidth)  c\_mdel\_cluster\_labels = c\_mdel.fit\_predict(X\_processed)  sample = X.copy()  sample['cluster'] = c\_mdel\_cluster\_labels  print('cluster labels type: ', np.unique(c\_mdel\_cluster\_labels))  print('bandwidth값 : ', best\_bandwidth)  sample\_score = silhouette\_samples(X\_processed, sample['cluster'])  sample['silhouette\_'] = sample\_score  print('aver sihouette\_: ' + str(sample['silhouette\_'].mean()))  print(sample.groupby('cluster')['silhouette\_'].mean())  score\_results.loc[len(score\_results)] = [group, num\_process, cat\_process, model,  'bandwidth: ' + str(best\_bandwidth),  str(sample['silhouette\_'].mean())]  print(group, num\_process, cat\_process, model, 'bandwidth: ' + str(best\_bandwidth),  str(sample['silhouette\_'].mean()))  print(sample.groupby('cluster')['silhouette\_'].mean())  k = len(sample[['cluster']].groupby('cluster').count())  if(k<=7):  makePlt23(X, sample['cluster'], k, 1, 'bandwidth='+str(best\_bandwidth))  else:  makePltBig(X, sample['cluster'], k)  plt.show()  return  # Import the data file  df = pd.read\_csv('C:/Users/leeminsu/PycharmProjects/mlPHW2CaliforniaHousing/housing.csv', encoding='utf-8')  # print(df.dtypes)  # print(df.isna().sum())  ##setting data set  # separate median house value feature  mhv=df['median\_house\_value']  df.drop('median\_house\_value',axis=1, inplace=True)  # fill nan value in total\_bedrooms  df.fillna(0, inplace=True)  # print(df.isna().sum())  #group 1 room  X1 = df[['total\_rooms','total\_bedrooms']]  #group 2 where  X2 = df[['longitude','latitude','ocean\_proximity']]  #group 4 eviroment  X4 = df[['population','households']]  autoprocess = AutoProcess(verbose=True)  autoprocess.run(X1,'room')  autoprocess.run(X2,'where')  autoprocess.run(X4,'eviroment')  print(score\_results) |
| Explain |
| lab2.py selects a situation under certain conditions based on the criteria we set and shows the "total silhouette average," "silhouette average for each cluster," and "visualization of results with fewer than 7 clusters." |

|  |
| --- |
|  |
| Lab2\_clarans.py |
| !pip install pyclustering  !pip install chart\_studio  !pip install cufflinks  #plotly imports  import plotly as py  from plotly.subplots import make\_subplots  import plotly.express as px  import plotly.graph\_objs as go  from plotly.offline import download\_plotlyjs, init\_notebook\_mode, plot, iplot  import chart\_studio  chart\_studio.tools.set\_credentials\_file(username='username', api\_key='api\_key')  from matplotlib import pyplot as plt  from sklearn.metrics import accuracy\_score  import pandas as pd  import numpy as np  from sklearn import preprocessing  import warnings ; warnings.filterwarnings('ignore')  import seaborn as sns  from sklearn.metrics import accuracy\_score, roc\_auc\_score, f1\_score  from sklearn import preprocessing  from sklearn.model\_selection import cross\_val\_score, StratifiedKFold  from sklearn.model\_selection import KFold  from sklearn.model\_selection import cross\_val\_score  from sklearn.model\_selection import GridSearchCV  import sys  from sklearn.metrics import silhouette\_score  from matplotlib import pyplot as plt  from pyclustering.cluster.clarans import clarans  # To render in colab  def configure\_plotly\_browser\_state():  import IPython  display(IPython.core.display.HTML('''  <script src="/static/components/requirejs/require.js"></script>  <script>  requirejs.config({  paths: {  base: '/static/base',  plotly: 'https://cdn.plot.ly/plotly-latest.min.js?noext',  },  });  </script>  '''))  ################################################################################  df = pd.read\_csv('/content/drive/MyDrive/머신러닝/housing.csv')  mhv=df['median\_house\_value']  df.drop('median\_house\_value',axis=1, inplace=True)  # fill nan value in total\_bedrooms  df.fillna(0, inplace=True)  roomDF=df[['total\_rooms', 'total\_bedrooms']]  whereDF=df[['longitude', 'latitude', 'ocean\_proximity']]  specDF=df[['housing\_median\_age', 'total\_rooms', 'total\_bedrooms', 'ocean\_proximity']]  enviDF=df[['population', 'households']]  sc=preprocessing.MinMaxScaler()  enc=preprocessing.OrdinalEncoder()  op=enc.fit\_transform(whereDF['ocean\_proximity'].to\_numpy().reshape(-1, 1))  whereDF['ocean\_proximity']=op  tdf=whereDF[0:2000]  #tdf=tdf.drop('ocean\_proximity', axis=1)  # print(tdf)  tdfs=sc.fit\_transform(tdf)  tdfs=tdfs.tolist()  # need dataframe, list of label, number of clusters  # def makePlt23(df, label, k):  # # list for store feature data for each cluster  # store = [[[] for col in range(len(df.columns))] for row in range(k)]  # for m in range(len(label)):  # for n in range(i):  # if(label[m]==n):  # for o in range(len(tdf.columns)):  # store[n][o].append(tdf.iloc[m:m+1, o:o+1].values[0][0])  # plt.subplot(230 + (k - 1))  # plt.xlabel(df.columns[0])  # plt.ylabel(df.columns[1])  # c = ['b.', 'r.', 'g.', 'y.', 'c.', 'm.']  # if(len(df.columns)==2):  # plt.subplot(230 + (k - 1))  # plt.xlabel(df.columns[0])  # plt.ylabel(df.columns[1])  # for p in range(k):  # plt.plot(store[p][0], store[p][1], c[p])  # if(len(df.columns)==3):  # plt.subplot(230+(k-1), projection='3d')  # plt.xlabel(df.columns[0])  # plt.ylabel(df.columns[1])  # for p in range(k):  # plt.plot(store[p][0], store[p][1], store[p][2], c[p])  ########## clarans ###########  # try three cluster numbers  for i in (2,3,4,5,6):  # make list to store each rows label  label=[0 for l in range(len(tdf))]  # data, number of cluster, num local, max neighbor  clarans\_instance=clarans(tdfs, i, 6, 4)  clarans\_instance.process()  clusters=clarans\_instance.get\_clusters()  # make label  for j in range(0, len(clusters), 1):  for k in range(0, len(clusters[j]), 1):  label[clusters[j][k]]=j  print(label)  score=silhouette\_score(tdfs, label)  print(i, 'clusters silhouette score :', score)    tdf['clu'] = np.array(label)  print(tdf)  title = f'Visualizing Clusters with,longitude, latitude, ocean\_proximity\n\ Number of Clusters :{i}'  fig = px.scatter\_3d(tdf, x='longitude', y='latitude',z='ocean\_proximity',color = 'clu', title = title)  fig.update\_traces(marker=dict(size=3))  #z='ocean\_proximity',color = 'clu'  configure\_plotly\_browser\_state()  init\_notebook\_mode(connected=False)  iplot(fig)  # makePlt23(tdf, label, i)  #plt.show() |
| Explain |
| Code performed separately because too many costs occurred when using the clarans model. |

|  |
| --- |
|  |
| **Explaination and Interpretation of 1st result.** |
|  |

|  |
| --- |
| Screenshot of the results. |
|  |

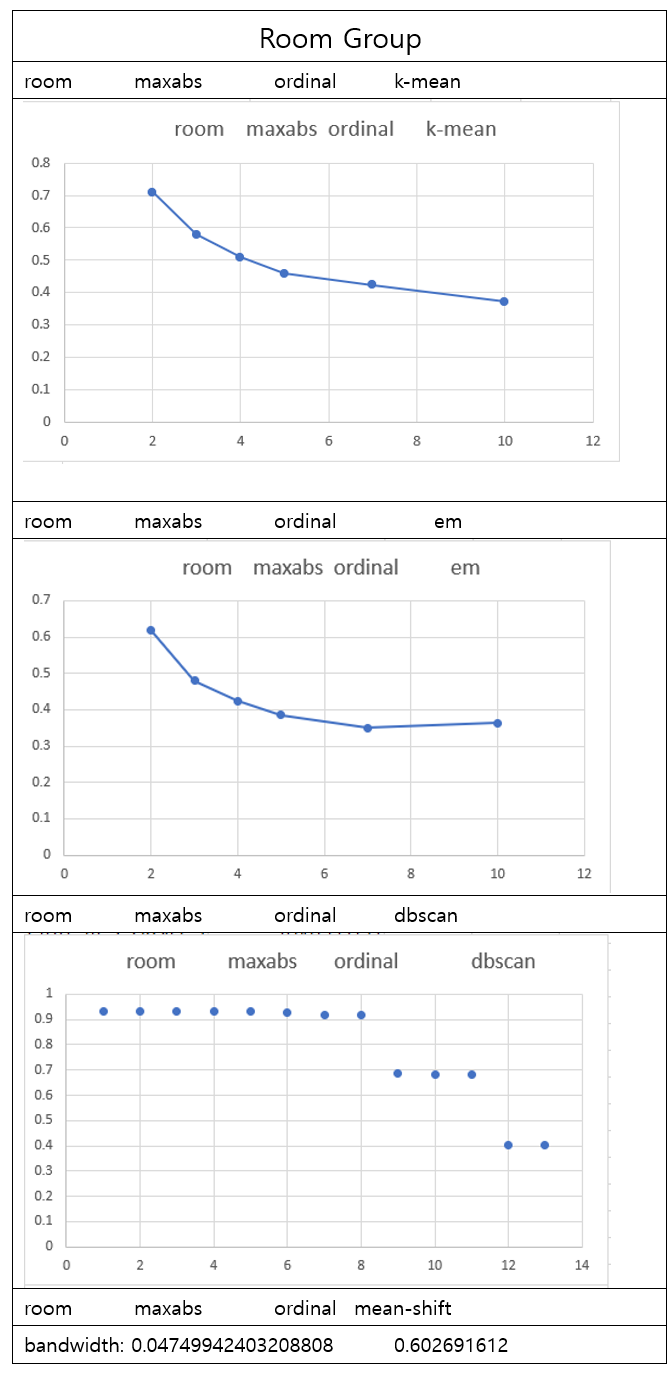
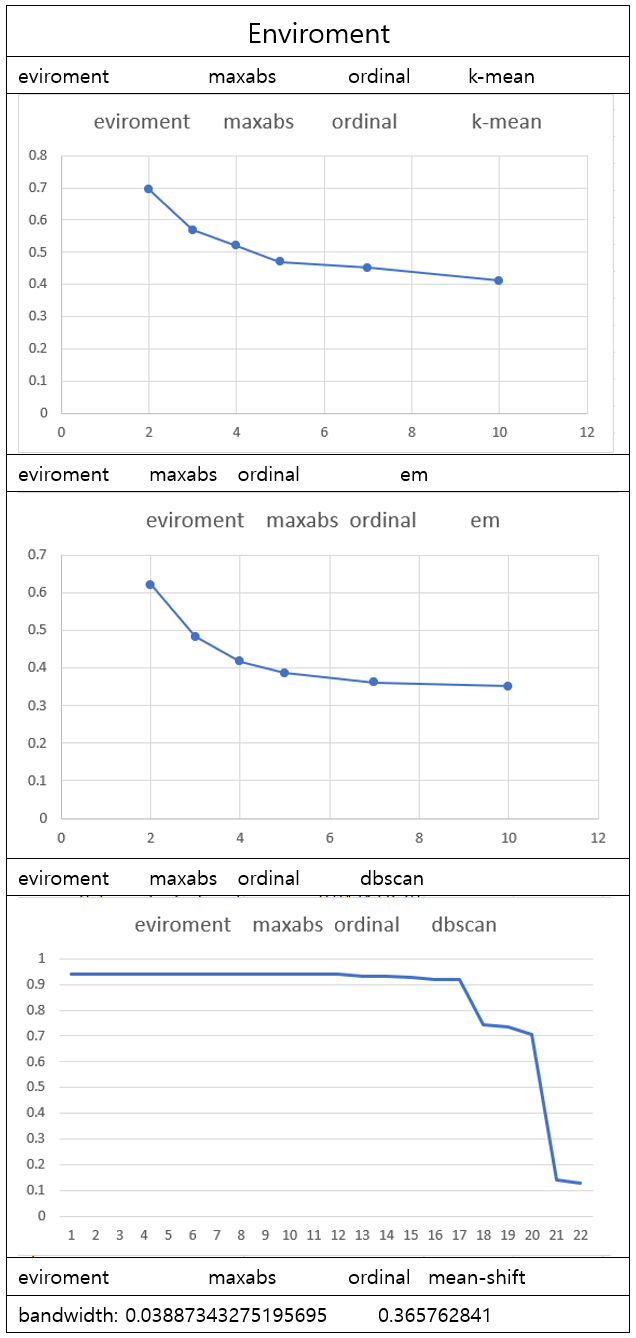
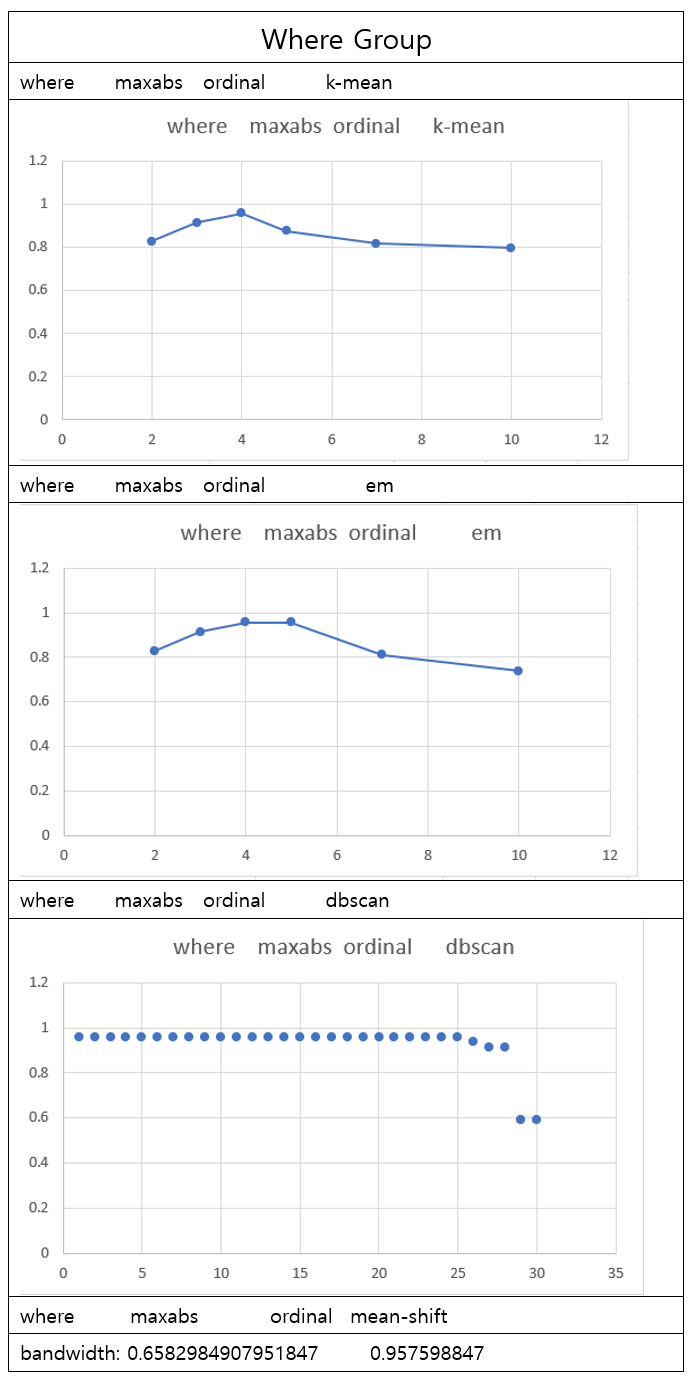
There are 1,571 results.

In fact, there is no part in the code that deals with clarans. When learning the model, we consumed a lot of cost, wrote a separate code, and included it in the result.

We calculated the average silhouette score for each condition by dividing by group and model.

|  |  |
| --- | --- |
| Group by | Model\_by |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Group by | Aver\_sil | Scaler\_by | Encord\_by | Aver\_sil | | room | 0.517697 | standard | ordinal | 0.460647 | | onehot | 0.460906 | | robust | ordinal | 0.430401 | | onehot | 0.430509 | | minmax | ordinal | 0.636876 | | onehot | 0.636931 | | maxabs | ordinal | 0.636771 | | onehot | 0.637056 | | where | 0.649162 | standard | ordinal | 0.412534 | | onehot | 0.386558 | | robust | ordinal | 0.539907 | | onehot | 0.538443 | | minmax | ordinal | 0.737253 | | onehot | 0.7598 | | maxabs | ordinal | 0.912397 | | onehot | 0.906406 | | enviroment | 0.51242 | standard | ordinal | 0.381648 | | onehot | 0.38173 | | robust | ordinal | 0.338579 | | onehot | 0.338553 | | minmax | ordinal | 0.699532 | | onehot | 0.699458 | | maxabs | ordinal | 0.699598 | | onehot | 0.699501 | | spec | 0.317284 | standard | ordinal | 0.009753 | | onehot | 0.008174 | | robust | ordinal | 0.043809 | | onehot | 0.034709 | | minmax | ordinal | 0.571629 | | onehot | 0.607528 | | maxabs | ordinal | 0.572759 | | onehot | 0.608092 | | all | 0.181696 | standard | ordinal | -0.14788 | | onehot | -0.15394 | | robust | ordinal | -0.13212 | | onehot | -0.14304 | | minmax | ordinal | 0.406339 | | onehot | 0.444012 | | maxabs | ordinal | 0.465649 | | onehot | 0.508848 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Model\_by | Aver\_sil | Scaler\_by | Encord\_by | 평균 | | k-mean | 0.533272 | minmax | ordinal | 0.599859 | | onehot | 0.586356 | | maxabs | ordinal | 0.623412 | | onehot | 0.607352 | | em | 0.453443 | minmax | ordinal | 0.540883 | | onehot | 0.518253 | | maxabs | ordinal | 0.562424 | | onehot | 0.535185 | | dbscan | 0.405787 | minmax | ordinal | 0.622484 | | onehot | 0.663704 | | maxabs | ordinal | 0.689994 | | onehot | 0.723941 | | mean-shift | 0.603025 | minmax | ordinal | 0.639282 | | onehot | 0.653399 | | maxabs | ordinal | 0.665322 | | onehot | 0.699247 | |

Let's look at the average silhouette score for each condition.  
We can see that the silhouette scores are relatively low in the spec group and all group in Group\_by.  
What they have in common is that they have a lot of features. I found it difficult to cluster if there are many unrelated features.  
Therefore, we chose to focus on the room, where, and environment groups with high average scores.  
We found that there are many differences in scores in scaler selection.  
Standard and robust confirmed that the scores were significantly lower than that of minmax and maxabs.  
However, there was no significant difference between the two encorders.  
Therefore, we chose to focus on the results from the room, where, environment group, minmax, max scaler.  
However, there are still too many results.  
We decided to use model\_by to deduce one more time.  
If you look at model\_by, you can see that the score is generally high when using the maxabs scaler.  
Encoder found that there was no significant difference in silhouette scores. We personally prefer the original encoder, so we decided to visualize it by selecting parameter\_set, which recorded a high silhouette score among each of the values.  
The table below is a silhouette score graph of the conditions we described above.

We chose to check and analyze the results by separately executing only the knee part and the relatively high score situation in each situation..

|  |
| --- |
|  |
| **Explaination and Interpretation of 2nd result.** |
|  |

Room Group

|  |  |  |
| --- | --- | --- |
| Cluster model: k-mean  room maxabs ordinal k-mean k=3 0.5823095637184951  cluster  0 0.382061  1 0.648942  2 0.364064 | Cluster model: k-mean  room maxabs ordinal k-mean k=5 0.4574803704425634  cluster  0 0.364014  1 0.528689  2 0.357499  3 0.368930  4 0.428195 | |
| Explain | | |
| When K is 3, the standard deviation of the silhouette score is 0.159.  When K is 5, the standard deviation of the silhouette score is 0.07.  The overall average silhouette score is higher in "k=3", but the silhouette score is higher in "k=5".  When looking at the graph representing the actual cluster, it seems that the graph when "k=5" is more distinguished.  This seems to be a better cluster if the silhouette score between each cluster is even though the overall silhouette is different. | | |
| Cluster model: em  room maxabs ordinal em k=3 0.4792379717378938  cluster  0 0.586756  1 0.099020  2 0.255223 | Cluster model: em  room maxabs ordinal em k=5 0.3926376932157858  cluster  0 0.507062  1 0.334710  2 0.296321  3 -0.360172  4 0.400864 |
| Explain | |
| When K is 3, the standard deviation of the silhouette score is 0.25.  When K is 5, the standard deviation of the silo score is 0.34.  The overall silhouette score and the standard deviation between clusters are also better when "k=3".  However, if you remove cluster 1 of "k=3" and cluster 3 of "k=5," which adversely affect here, you can see that "k=5" is a much better result.  Even when "k=3", I think that too much data loss occurs except for cluster 1, which adversely affects the results.  Simply looking at the results, the result of "k=3" is evaluated as better, but I think "k=5" is better if the value is removed too much. | |
| Cluster model: clarans  room maxabs ordinal clarans k=3 0.20471528070924733 | Cluster model: clarans  room maxabs ordinal clarans k=5 0.4317456222688951 | |
| Explain | | |
| It can be seen that both "k=3" and "k=5" are clearly distinguished.  However, it can be seen that the score is more than twice as high when "k=5", and that the graph shows better distinction. | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| environment maxabs ordinal mean-shift bandwidth : 0.03887343275195695 silhouette score : 0.365762841372087 | | | | | |
| 0.364918 | 0.174931 | | 0.313840 | | |
|  |  | |  | | |
| 0.155682 | 0.513323 | | 0.360562 | | |
|  |  | |  | | |
| 0.487906 | 0.198101 | | 0.000000 | | |
|  |  | |  | | |
| 0.566688 | 0.699361 | | 0.637004 | | |
|  |  | |  | | |
| 0.240607 | 0.518996 | | 0.246039 | | |
|  |  | |  | | |
| 0.602876 | 0.026550 | | 0.932737 | | |
|  |  | |  | | |
| N18~23 : 0.0000 | N24 : 0.0147739 | | 0.000000 | | |
|  |  | |  | | |
| 0.000000 | 0.317765 | | 0.616874 | | |
|  |  | |  | | |
| 테이블이(가) 표시된 사진  자동 생성된 설명테이블이(가) 표시된 사진  자동 생성된 설명 | | It can be seen that the standard deviation of all clusters in this case is 0.27 and 0.22 when 9 noise with a double score of 0 is removed. In addition, better data can be obtained at 0.21 if clusters 16 and 24 are also removed. However, there is too much data such as noise, and the deviation of the score is too large to say that this case is appropriate. | | | |
| Cluster model : mean-shift  where maxabs ordinal mean-shift badwidth : 0.6582954907951847  silhouette score : 0.9575988470645481 | | | | |
| 그룹별 mean-shift에서 where그룹의 데이터가 가장 좋다고 할 수 있다.  먼저 score가 약 0.96으로 다른 그룹에 비해 상당히 높은 점수를 갖고 있고, 그래프를 관찰했을 때 5개의 클러스터로 확연히 클러스터링이 되는 것을 확인할 수 있다. | | | |

|  |  |
| --- | --- |
| **Where group** | |
| Cluster model: k-mean  where maxabs ordinal k-mean k=4 0.9571457474740559  cluster  0 0.938343  1 0.939160  2 0.966425  3 0.994833 | Cluster model: k-mean  where maxabs ordinal k-mean k=7 0.816251173470932  cluster  0 0.897217  1 0.791532  2 0.651128  3 0.994812  4 0.718734  5 0.764116  6 0.801174 |
| Explain | |
| When K is 4, the standard deviation of the silhouette score is 0.02.  When K is 7, the standard deviation of the silhouette score is 0.11.  When "k=4", all results are superior, even when viewed on a graph, the characteristics and shapes of each cluster are expressed very clearly.  It seems that the feature groups have very close relationships and harmonized with each other, and in addition, model selection and preprocessing are very good. | |
| Cluster model: em  where maxabs ordinal em k=5 0.9575988470645439  cluster  0 0.998854  1 0.939160  2 0.966399  3 0.994833  4 0.939760 | where maxabs ordinal em k=7 0.8795665367519526  cluster  0 0.998854  1 0.820539  2 0.906154  3 0.994813  4 0.939608  5 0.613242  6 0.594003 |
| Explain | |
| When K is 5, the standard deviation of the silhouette score is 0.028.  When K is 7, the standard deviation of the silhouette score is 0.17.  When "k=5", all results are superior, even when viewed on a graph, the characteristics and shapes of each cluster are expressed very clearly.  It seems that the feature groups have very close relationships and harmonized with each other, and in addition, model selection and preprocessing are very good.  However, in the case of "k=5", it is difficult to observe the results of one cluster, so this should be referred to. | |
| |  |  |  | | --- | --- | --- | | room maxabs ordinal dbscan eps: 0.01 m: 3 cluster: 33 0.40383335624345845 | | | | -0.689489 | 0.411851 | 0.891278 | |  |  |  | | 0.767720 | 0.494258 | 0.195297 | |  |  |  | | 0.404919 | 0.306962 | 0.944980 | |  |  |  |  |  |  |  | | --- | --- | --- | | room maxabs ordinal dbscan eps: 0.01 m: 3 cluster: 9 0.6814441937393674 | | | | -0.576357 | 0.699347 | 0.595643 | |  |  |  | | 0.770366 | 0.820736 | 0.706225 | |  |  |  | | 0.683556 | 0.736536 | 0.675933 | |  |  |  | | |
| Explain | |
| The deviation of the score according to the cluster is very large.  Also, the score of this case is not high.  If the scores of cluster 22 and cluster 4 are less than 0.2, and this data is judged as noise and removed, better data can be obtained. However, this case is not appropriate because there is a possibility of major data loss | |
| Cluster model: dbscan  where maxabs ordinal dbscan eps: 0.75 m: 10 cluster: 5 0.9575988470645439  cluster  -1 0.998854  0 0.994833  1 0.966399  2 0.939760  3 0.939160 | where maxabs ordinal dbscan eps: 0.01 m: 7 custer: 6 0.912274512601629  cluster  -1 -0.359544  0 0.994824  1 0.966431  2 0.939774  3 0.949120  4 0.576307 |
| Explain | |
| When "eps: 0.75 m: 10", the standard deviation of the silhouette score is 0.028.  When "eps: 0.01 m: 7", the standard deviation of the silhouette score is 0.53.  When "eps: 0.75 m: 10", all results are superior, even when viewed on a graph, the features and shapes of each cluster are expressed very clearly.  It seems that the feature groups have very close relationships and harmonized with each other, and in addition, model selection and preprocessing are very good. | |

|  |
| --- |
| Cluster model : mean-shift  where maxabs ordinal mean-shift badwidth : 0.6582954907951847  silhouette score : 0.9575988470645481 |
| It can be said that the data of where group is the best in mean-shift by group. First, the score is about 0.96, which is significantly higher than that of other groups, and when the graph is observed, it can be seen that it is clearly clustered into five clusters. |

|  |  |
| --- | --- |
| **Eviroment group** | |
| Cluster model: k-mean  eviroment maxabs ordinal k-mean k=5 0.4697027820276445  cluster  0 0.542011  1 0.392455  2 0.395372  3 0.439716  4 0.395087 | eviroment maxabs ordinal k-mean k=7 0.44741401957527105  cluster  0 0.394563  1 0.510049  2 0.391423  3 0.386507  4 0.450594  5 0.383685  6 0.454844 |
| Explain | |
| When K is 5, the standard deviation of the silhouette score is 0.064.  When K is 7, the standard deviation of the silhouette score is 0.048.  When "k=5", the overall silhouette is higher, but the standard deviation between clusters is slightly larger.  Both of these data should be in a meaningful form, and efforts should be made to evaluate and analyze both.  If you look at the yellow cluster with an exception in the "K=7" situation, you can see that it is a much better result.  In addition, there are few elements of the yellow cluster, so it is thought that it will not significantly affect the decline in reliability. | |
| Cluster model: em  eviroment maxabs ordinal em k=3 0.4835862406085093  cluster  0 0.582823  1 0.255770  2 0.221467 | Cluster model: em  eviroment maxabs ordinal em k=5 0.38593647803168346  cluster  0 0.480076  1 0.338862  2 0.393994  3 0.039374  4 0.292055 |
| Explain | |
| In this case, the difference is clearly revealed when checked with a visualized graph.  The overall silhouette score is better when k=3, but in the case of green clusters, the silhouette is not good because of the far value, even though the dense part stands out.  However, if the yellow cluster is excluded from "K=5", it can be seen that it is a very clean type of cluster. | |
| Cluster model: clarans  eviroment maxabs ordinal clarans k=3 0.5460664190220929 | Cluster model: clarans  eviroment maxabs ordinal clarans k=5 0.4132568956388239 |
| Explain | |
| In the case of CLARANS, it is difficult to know the exact distribution because a lot of data is not available, but it is expected that the result will be similar to that of EM.  Currently, the overall silhouette score seems to be at a good level around 0.5, but if data is added, the value is likely to drop. | |

|  |  |
| --- | --- |
| Cluster model: dbscan  eviroment maxabs ordinal dbscan eps: 0.1 m: 3 cluster: 3 0.9195233403632125  cluster  -1 0.351373  0 0.919578   1. 0.920340 | Cluster model: dbscan  eviroment maxabs ordinal dbscan eps: 0.1 m: 5 cluster: 2 0.9307135267320606  cluster  -1 0.682885  0 0.930774 |
| Explain | |
| The scores for each cluster are similar, so the deviation is not large. In addition, the scores are all 0.5 or higher, which is better data than the case with 33 clusters. In addition, the overall score is 0.68, so it is more appropriate than the 33 cluster cases with a score of 0.4. | |

|  |
| --- |
| eviroment maxabs ordinal dbscan eps: 0.01 m: 3 cluster: 14 0.13881267601581548  cluster  -1 -0.755226  0 0.143349  1 0.668145  2 0.926291  3 0.812058  4 0.726993  5 0.743603  6 0.633752  7 0.646256  8 0.871860  9 0.516373  10 0.710146  11 0.764415  12 0.895464 |
| Explain |
|  |

|  |
| --- |
|  |
| **Each members opinions and culclusion.** |
|  |

We made five groups of feature in the following form.

테이블이(가) 표시된 사진

자동 생성된 설명

Since we are a group that we personally thought about and shared, there were groups that showed good results and relatively low results.

Where, environment, room, spec, and all groups that received good reviews in order are in order.

When clustering into Where groups, the results came out so well that the silhouette score was close to 1.

And when using environment and room groups, they scored relatively high between 0.6 and 0.7.

However, the all group using all features and the spec group using 4 features scored poorly.

Therefore, we found that where, environment, and room groups were closely related between features, and we selected where, environment, and room for a more detailed analysis.

Since then, the scaler has confirmed that the scaler, which limits the difference between the maximum and minimum values to 1 and 2, is much more effective. Therefore, we selected minmax and maxabs as data to compare.

There was no significant difference between the two encorders. As a result of consulting with our team member, we chose to use the original encoder as an opinion that it would be more convenient to interpret.

In the same way as above, all clustering models were used to cluster.

Among the results, we visualized the conditions of the Knee part and the conditions with a high silhouette score when graphing the silhouette score, and found that the visualization results were not clustered properly and compared.

Through this direction, we evaluated each graph in the "Explanation and Intervention of 2nd result."

Explain why we got these results when we were divided into groups.

The Room group seems to be proportional to total\_rooms if the total\_rooms are high, but the farther away from 0, the larger the distribution, adversely affecting the silhouette and affecting the quality of the cluster.

Where group also had an overwhelmingly good average silhouette score. The latitude longitude is a fair value, and the coastal and distance features are thought to have added information to produce better results.

In fact, if you look at the graph after preprocessing, you can see that each personality is united.

I think it's a very good example.

Unfortunately, the Spec group did not include recording and visualizing low silhouette scores. The reason for the low evaluation is thought to be that each factor is a factor that determines the value of a house, but the relationship itself between each factor is very low and widely distributed, resulting in results.

The Environment Group chose a feature about the population environment around the house, not information about the house itself as an environmental factor. In fact, I didn't think this would produce much good results, but it seems to have a high convenience relationship beyond expectations.

When clustering was done using all features, it showed very disastrous results. This is expected to be too much feature, but there were features that had nothing to do with each other, which was found to have been a major obstacle to clustering.

Through this, we learned how much it affects the results by dividing features into groups and comparing the results when clustering data or learning models in the future.

**Room group (total\_rooms, total\_bedrooms)**

We think K-means with k=5 that preprocessed by maxabs scaler and ordinal encoder makes the best result for the room group. Because its average silhouette score(0.46) is on the high side compared to other algorithm’s scores and the standard deviation of its silhouette score is 0.07, which means that the silhouette score of each cluster is even.

**Where group (longitude, latitude, ocean\_proximity)**

Where group gave good results overall. Therefore, it seems that the three features are closely related. They showed similar scores, but we think mean-shift with bandwidth=0.66 that preprocessed by maxabs scaler and ordinal encoder is slightly better among them. Because its average silhouette score(0.96) and standard deviation of its silhouette score is 0.02. Above all, since the mean-shift does not have to directly determine the number of clusters, there is little difficulty in setting parameter values.

**Environment group (population, households)**

Environment group did not score well overall, but EM with k=3 that preprocessed by maxabs scaler and ordinal encoder showed the best result among them. EM has a relatively high average silhouette score(0.48). Compared to K-means, the standard deviation of its silhouette score is a little higher, but if we handle outliers, we will get better results.