|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211020 |
| 교육장소 | YGL 학과장 및 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. 결정트리   캡처.JPG  캡처.JPG  설명하기 쉬운 모델. 질문을 추가해서 데이터를 계속 나누며 분류 정확도를 높이는 모델이다.  사이킷런의 DecisionTreeClassfier 클래스로 만들 수 있음.  예)  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  wine = pd.read\_csv('https://bit.ly/wine\_csv\_data')  wine.head()  wine.info()  data = wine[["alcohol", "pH"]].to\_numpy()  target = wine["class"].to\_numpy()  data[:10]  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target = train\_test\_split(  data, target, stratify= target)  print(train\_input.shape)  print(test\_input.shape)  from sklearn.tree import DecisionTreeClassifier  dt = DecisionTreeClassifier()  dt.fit(train\_input, train\_target)  print(dt.score(train\_input, train\_target))  print(dt.score(test\_input, test\_target))  from sklearn.tree import plot\_tree  plt.figure(figsize= (10,7))  plot\_tree(dt)  plt.show()  plot\_tree(dt, max\_depth = 1, filled=True, feature\_names=['alcohol', 'sugar', 'pH'])  plt.show()  dt = DecisionTreeClassifier(max\_depth=3, random\_state=42)  dt.fit(train\_input, train\_target)  print(dt.score(train\_input, train\_target))  print(dt.score(test\_input, test\_target))  plt.figure(figsize= (20, 15))  plot\_tree(dt, feature\_names=['alchol', 'sugar', 'pH'])  plt.show()  dt.feature\_importances\_  feature\_names = ['alcohol','sugar','pH']  data = wine[['alcohol','sugar','pH']].to\_numpy()  target = wine["class"].to\_numpy()  print(data.shape)  # 데이터 나누기  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target = train\_test\_split(  data, target, stratify=target, random\_state=42)  from sklearn.tree import DecisionTreeClassifier  dt = DecisionTreeClassifier(random\_state= 42) # 데이터 샘플링에 대한 랜덤스테이츠  from sklearn.model\_selection import cross\_validate  from sklearn.model\_selection import StratifiedKFold # 데이터 비율이 맞게 뽑히게  splitter = StratifiedKFold(n\_splits= 10, shuffle=True, random\_state= 42)  score = cross\_validate(dt, train\_input, train\_target, cv = splitter)  score['test\_score'] #cv = 10이라서 10개나옴  score['test\_score'].mean()  from sklearn.model\_selection import GridSearchCV  dt = DecisionTreeClassifier()  papam = {'max\_depth': np.arange(4,20,1),  'min\_impurity\_decrease': np.arange(0.0001, 0.001, 0.001)}  gs = GridSearchCV(dt, param\_grid= papam, cv = 10, n\_jobs= -1)  # n\_job = -1 : 모든자원 활용  gs.fit(train\_input, train\_target)  gs.cv\_results\_['mean\_test\_score']  # max\_depth가 5개  #가장 좋은 결과를 가지고있는 모형  dt = gs.best\_estimator\_ # depth = 5(베스트라서)  print(dt.score(train\_input, train\_target))  gs.best\_params\_  plt.plot(np.arange(4,20,1),gs.cv\_results\_['mean\_test\_score'])  plt.show()   1. 랜덤 포레스트   from sklearn.model\_selection import cross\_validate  from sklearn.ensemble import RandomForestClassifier  rf = RandomForestClassifier(n\_estimators=100,  criterion= 'gini',  n\_jobs= -1,  )  scores = cross\_validate(rf, train\_input, train\_target, cv = 10,  return\_train\_score=True)  print(np.mean(scores['test\_score']))  print(np.mean(scores['train\_score']))  rf = RandomForestClassifier(n\_estimators=100,  criterion= 'gini',  n\_jobs= -1)  rf.fit(train\_input, train\_target)  print(rf.feature\_importances\_)  print(wine.columns)   1. K-Means   예)  ! python -m wget [https://bit.ly/fruit\_300 -o fruits\_300.npy](https://bit.ly/fruit_300%20-o%20fruits_300.npy)  import numpy as np  import matplotlib.pyplot as plt  fruits = np.load('fruits\_300.npy')  plt.imshow(fruits[0], cmap = 'gray\_r') # cmap은 나중에 배움  plt.show() # 100x100짜리 사진 한 개  print(fruits.shape)  # 2D로 바꾸기  fruits\_2d = fruits.reshape(-1, 10000) #array가 가로방향 일렬로 퍼짐  print(fruits\_2d.shape)  fig, axs = plt.subplots(1,3)  axs[0].imshow(fruits[0], cmap = 'gray\_r')  axs[1].imshow(fruits[100], cmap = 'gray\_r')  axs[2].imshow(fruits[200], cmap = 'gray\_r')  plt.show()  from sklearn.cluster import KMeans  km = KMeans(n\_clusters= 3, random\_state=42) #cluster 정해줘야함  km.fit(fruits\_2d)  print(km.labels\_) # 2222.....222, 000....000, 111...111 이어야하는디?  # centroid  km\_center = km.cluster\_centers\_.reshape(-1, 100, 100)  print(km.cluster\_centers\_.shape) # 3 cluster, 1만개  print(km.cluster\_centers\_.reshape(-1, 100, 100))  print(km\_center.shape)  # centroid가 어떻게생겼지?  fig, axs = plt.subplots(1,3)  axs[0].imshow(km\_center[0], cmap = 'gray\_r')  axs[1].imshow(km\_center[1], cmap = 'gray\_r')  axs[2].imshow(km\_center[2], cmap = 'gray\_r')  plt.show()  inertia = []  for k in range(2, 7):  km = KMeans(n\_clusters= k, random\_state= 42)  km.fit(fruits\_2d)  inertia.append(km.inertia\_)    plt.plot(range(2, 7), inertia)  plt.show()  # elbow method, 3에서 꺾인다.  km.predict(fruits\_2d[0:1]) | |