```
train_x1, test_x1, train_y1, test_y1 = train_test_split(iris_x_pca2, iris_y,   random_state=0)   #이렇게 네 개로 나누겠다
train_x2, test_x2, train_y2, test_y2 = train_test_split(iris_x_pca3, iris_y,   random_state=0)
#random_state=0 << 이걸 꼭 해줘야 함 계속 분석하려면 / 초기값 설정 seed값 고정함
m_knn1.fit(train_x1, train_y1)          #KNeighborsClassifier()
m_knn1.score(test_x1, test_y1)          #0.8947368421052632    #fit 한 다음 transform 안함 knn은
m_pca2.explained_variance_ratio_  #각 인공변수의 분산 설명력 #array([0.72962445, 0.22850762])
<< 앞에 어레이가 압도적으로 중요한 변수
sum(m_pca2.explained_variance_ratio_)    #0.9581320720000165
m_knn2.fit(train_x2, train_y2)
m_knn2.score(test_x2, test_y2)          #0.9736842105263158
m_pca3.explained_variance_ratio_        #array([0.72962445, 0.22850762, 0.03668922])
sum(m_pca3.explained_variance_ratio_)    #0.9948212908928452
```

! knn 최근접이웃 !
지도 학습 알고리즘 중 하나
굉장히 직관적이고 간단
데이터의 주변 데이터를 살펴본 뒤 더 많은 데이터가 포함되어있는 범주로 분류
k를 어떻게 정하냐에 따라 결과값이 바뀔 수 있음
k가 너무 작아도 커도 안됨 default값은 5(일반적으로 홀수로 씀)
훈련이 따로 필요없음
다른 모델들은 훈련데이터를 기반으로 모델 만들고 테스트 데이터로 테스트
>> 하지만 knn은 훈련 데이터를 저장하는 게 훈련의 전부
데이터와 데이터 사이의 거리를 구해야 함
방식 >> 유클리드 거리 / 맨해튼 거리 (얘네는 나중에 개념정리에서 따로 정리할 예정)