Clustering Basic;
k-Means Clustering & EM Algorithm

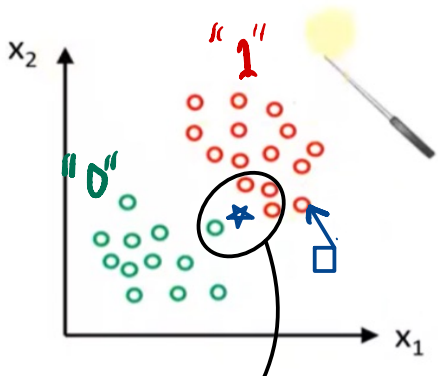
Hanwool Jeong

hwjeong@kw.ac.kr

classify 하는 과정이 다른 것임 , 직관적이고 단순한데 성능은 괜찮음

Classification based on Nearest Neighbors

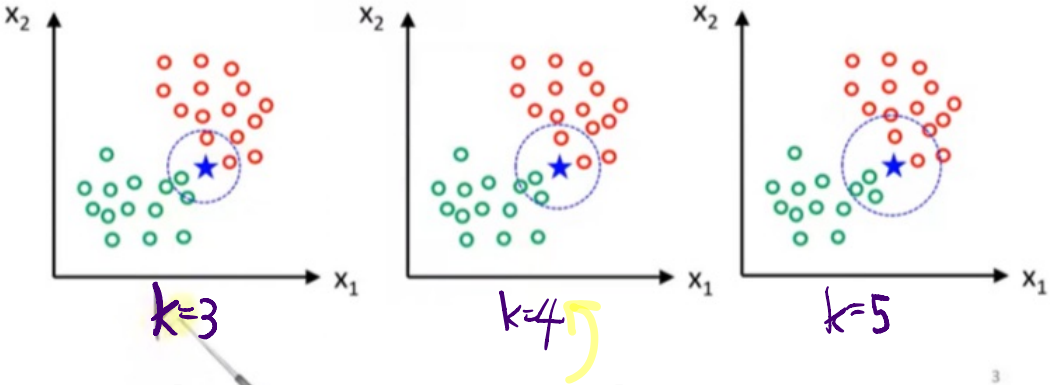
- The simplest method to determine the class of new data is based on the nearest neighbor.
- Is there any problem?



이미 정답이 있는 데이터셋이 있는 상황에서 0/1로 classifiaction 한 상황에서 데이터셋에 없는 새로운 데이터가 들어왔을 때 K-NN은 제일 가까운 것이 어느 클래스에 있는지 찾는 것임
그런데 별모양 데이터같은 데이터는 아무래도 빨강색클래스에 속하는 것이 맞는 것 같은데 가까운놈이 초록색이라고 하여서 0으로 classify되는 경우가 발생함, 따라서 이런 상황을 방지하기 위해서 하나만 보는게 아니라 k개를 보는 것
" 빨강색(1)의 개수가 3개 초록색(0)의 개수가 1개니깐 빨강색으로 보자"
와 같은 방식으로 정확도를 개선하는 것

Check k Nearest Neighbors (k-NN)

- k-NN is to perform classification based on comparing how k nearest neighbors are composed of.
- The result can be different according to k values
- Odd k is preferred. Why?



여기서 k는 주로 홀수로 채택함 , 짝수이면 비기는 상황이 발생, 홀수로 해서 한쪽이 무조건 우세한 경우가 나오도록 하는게 k-nn알고리즘임 그리고 위의 상황과 같이 k=3일때와 k=5일때의 결과가 다른 경우도 발생하는 문제점을 가지고 있음.

그런데 굉장히 단순한 알고리즘이어서 빠르게 동작할 것 같지만, 데이터셋이 1000개 10000개...모든 각각의 데이터와의 거리를 구한 다음에 제일 짧은 거리를 몇개(k개)를 보는 것인데,2차원이 아니라 3차원이상에서 100만개 1000만개에 클래스의 개수도 늘어나게 되어도 다 구해야함

Steps for k-NN Algorithm

1) Calculating distance to training set data from the input
2) Examining the top "k"s nearest neighbors' distance
3) Decide the class according to majority of the class.

그리고 k=5와 같은 경우처럼 조금만 원이 더 넓으면 빨강색이 더 가까워보이는 케이스도 있어서 애매해 보이는 경우도 생김 그래서 k-nn알고리즘을 좀더 개선할 수 있는 방법이....

가까운 샘플한테는 가중치를 좀 더 주고 멀리있는 것에는 적게주는 방식을 사용할 수 있는데 여기에서 Gaussian 분포가 들어 갈 수 있음. 이제부터는 파이썬 코드로 한땀한땀 작성하지 않고 라이브러리를 사용할 것임

Can you Refine k-NN?

- We can give more importance to the closer one!

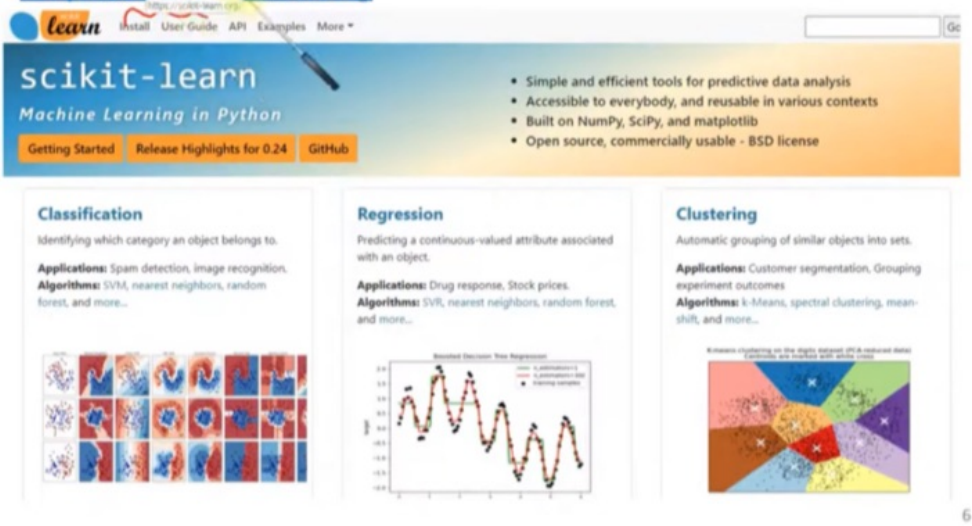
머신러닝 알고리즘 라이브러리 중에서 가장 대표적인게

Machine Learn

Python Library

Scikit-learn Library

- From now on, we will exploit scikit-learn library
- https://scikit-learn.org



9:00~

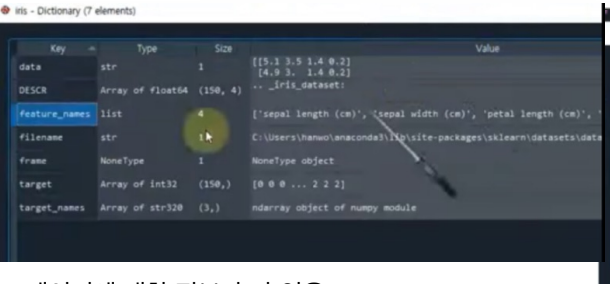
Try This!

```
1  from sklearn.datasets import load_iris
2  iris = load_iris()
3
4  from sklearn.datasets import load_breast_cancer
5  bCancer = load_breast_cancer()
6
```

Scikit-learn Provides Various Sample Datasets

label이 된 데이터셋을 사용해볼 수 있음

	Explanation
load_boston([return_X_y])	Load and return the boston house-prices dataset (regression). 보스톤의 특정 시점의 집 값의 데이터셋
load_iris([return_X_y])	Load and return the iris dataset (classification).
load_diabetes([return_X_y])	Load and return the diabetes dataset (regression).
load_digits([n_class, return_X_y])	Load and return the digits dataset (classification).
load_linnerud([return_X_y])	Load and return the linnerud dataset (multivariate regression).
load_wine([return_X_y])	Load and return the wine dataset (classification).
load_breast_cancer([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).



데이터에 대한 정보가 다 있음
target에 classification 되어 있음

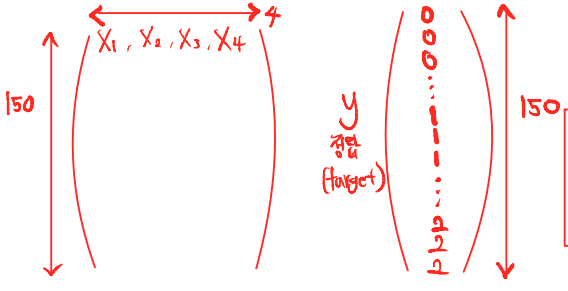
12:43~

Split Train Data vs. Test Data

- After loading datasets, the data for training/test should be split as:

```
from sklearn.model_selection import train_test_split
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 42)
```

이제 iris_flower를 KNN으로 구현 해볼 것임
4차원 150개 데이터중에 90개정도만 training하는데 사용하고 나머지 60개를 new data처럼 사용해서 정답을 맞추는지 판단해볼것임



이것을 전부 training해버리면 다 맞출 수 도 있음
그러면 의미가 없음

90개와 60개로 트레이닝과 테스트로 랜덤하게 분류해줘 라고하는 방법이 sklearn.model_selection 모듈 내부에 train_test_split이라는 메소드로 구현이 되어있음

test_size = 0.4 : 테스트 사이즈가1이면 전부 다 하는것 , 150중에 40%를 테스트에 사용하겠다.
random_state = 42 : 90, 60을 어떻게 split할 것인지 설정하는 것 (42번째 규칙으로 쪼개겠다는 뜻)

16:56

Name	Type	Size	Value
bCancer	utils.Bunch	7	Bunch object of sklearn.utils module
iris	utils.Bunch	7	Bunch object of sklearn.utils module
irisDataInput	Array of float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
X	Array of float64	(150, 4)	[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2]
X_test	Array of float64	(60, 4)	[[6.1 2.8 4.7 1.2] [5.7 3.8 1.7 0.3]
X_train	Array of float64	(90, 4)	[[6.3 2.7 4.9 1.8] [4.8 3.4 1.9 0.2]
y	Array of int32	(150,)	[0 0 0 ... 2 2 2]
y_test	Array of int32	(60,)	[1 0 2 ... 0 0 1]
y_train	Array of int32	(90,)	[2 0 0 ... 0 1 2]

x_train ,y_train이 90개로 랜덤하게 분류됨

Perform k-NN!

그런데 사이킥 런에 knn이 이미 구현되어 있음

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
scores = metrics.accuracy_score(y_test, y_pred)
```

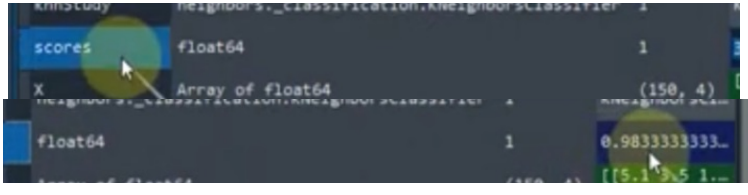
5-NN

metrics : 나눈 데이터셋을 이용해서 test를 할 때, 그것의 성능을 나타내는 지표를 나타내는 다양한 메소드가 들어 있는 모듈임

training을 할때 직접 while 문이나 for문을 사용할 필요 없이 knn.fit(데이터셋, 정답셋) 하면 끝임
이걸로 모델은 만들어 진 것임

y_pred : 만들어진 모델이 우리가 원하는 test 데이터를 넣어서 test set을 모델에대해서 predict 해준 것
metrics.accuracy_score(y_test, y_pred) : x_test의 진짜 정답인 y_test 와 y_pred를 비교하여서 정확도를 판별해보는 기능

돌려보면 y_pred , y_test가 거의 차이가 없는 것을 확인 할 수 있음



scores 도 0.983333...으로 98.33%의 정확도를 보여줌

왠만하면 정확함 n_neihbors값을 조금 씩 바꿔보면 정확도도 조금씩 변화하는 것을 확인 할 수 있음

이렇게 기존의 정답을 아는 애매한 데이터들을 knn에 때려박아서 우리가 맞추고자 하는 정답을 판단할 수 있는 알고리즘을 다른사람이 만든 라이브러리로 구현한 것임