참고 : github/jiwoo35/HoML/1_머신러닝-개요 및 KNN.ipynb
핸즈온 머신러닝 2판(Hands-on Machine Learning)

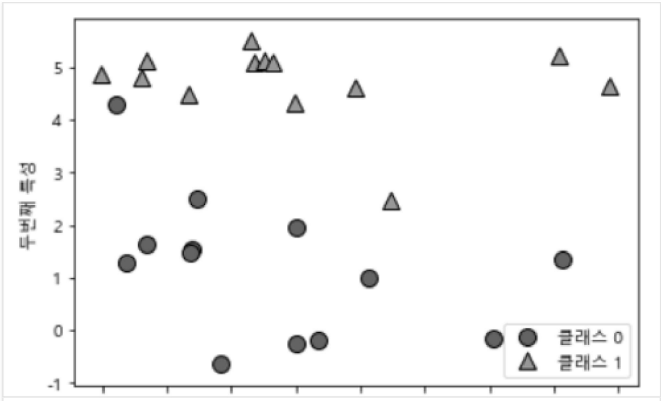
이번 시간에는 지도학습 중에서도 분류에 활용되는 대표적인 알고리즘, KNN에 대해 알아보자!

KNN (K-Nearest Neighbors)

1) 특징
- 이해하기 쉽고 직관적인 모델
- 더 복잡한 알고리즘을 적용하기 전에 시도해 볼 수 있는 알고리즘
- 훈련 데이터 셋이 너무 크면 예측이 느려진다.

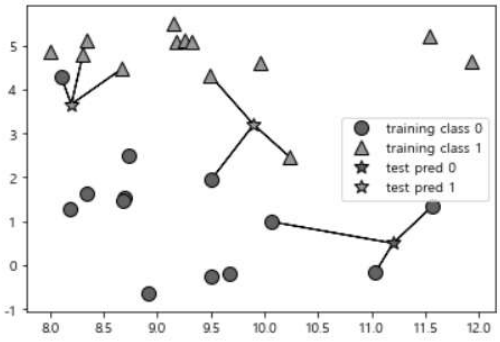
2) 파라미터(hyperparameter)
- 이웃의 개수 (K) : n_neighbors 파라미터로 지정
- 데이터 사이의 거리를 재는 방법 : L1 norm, L2 norm

이름에서 알 수 있듯이 K개의 이웃으로 분류를 한다 !
한번 mglearn 패키지에서 제공하는 knn 분류 알고리즘 사용 시 어떤 결과를 보여주는지 살펴보자!
(※ 아래 링크, 1. 머신러닝 개요 및 샘플데이터 코드 중에서 맨 앞에 필요한 라이브러리 임포트 하는 것 필수!)



[Machinelearning with Sci-Kit learn] 1. 머신러닝 개요 및 …
참고 : ../ai/1_머신러닝-개요 및 KNN.ipynb핸즈온 머신러닝 2판(Hand…
blog.naver.com

```
mglearn.plots.plot_knn_classification(n_neighbors=3)
```



knn 분류 예시

KNN 2) 파라미터에서 *n_neighbors*는 이웃의 개수를 지정하는 파라미터 라는 것은 위의 예시에서 알 수 있고, L1 norm과 L2 norm은 무엇인지 알아보자!

L1 norm, L2 norm

Norm (노름) 이란?
- 벡터의 길이 혹은 크기를 측정하는 방법(함수)
- Norm이 측정한 벡터의 크기는 원점에서 벡터 좌표까지의 거리 혹은 Magnitude

$$L_p = (\sum_{i}^{n} |x_i|^p)^{\frac{1}{p}}$$

- p: Lorm의 차수, 즉 p가 1이면 L1 Norm, p가 2이면 L2 Norm
- n: 대상 벡터의 요소 수

L1 norm
- 단순한 거리의 절대값의 합
- 맨하탄 거리(Manhattan norm)

$$L_1 = (\sum_{i}^{n} |x_i|)$$
$$= |x_1| + |x_2| + |x_3| + \ldots + |x_n|$$

L2 norm
- 거리의 절대값의 합에 루트 적용
- 피타고라스의 정리, 유클리드 거리(Euclidean norm)

$$L_2 = \sqrt{\sum_{i}^{n} x_i^2}$$
$$= \sqrt{x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2}$$

판다스 데이터 프레임 형식의 예시 데이터를 활용하여 좌표에 찍힌 점들 간의 최단 거리를 계산해보자

```
# 데이터 포인트의 최단 거리 : L2 norm으로 계산
df = pd.DataFrame(np.random.randint(low=1, high=10, size=(3,2)))
df.columns=["x", "y"]
df
```

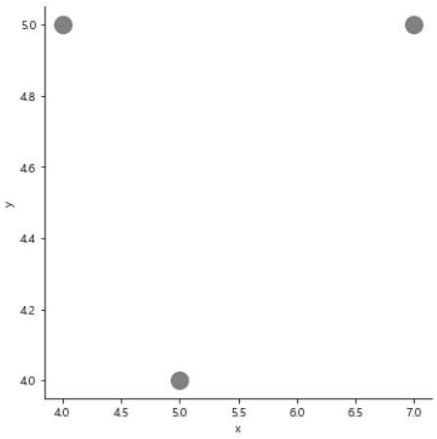
numpy의 randint 함수를 이용해서 1부터 10까지의 숫자로 3행 2열 형식의 랜덤한 df 변수를 만든다. 각열의 이름은 x와 y로 한다. 아래와 같은 데이터 프레임이 만들어졌다.

	x	y
0	4	5
1	7	5
2	5	4

seaborn 패키지의 **lmplot**(말 그대로 linear model을 그려주는 함수)를 사용하여 위의 좌표를 그래프에 표시한다.

```
sns.lmplot('x', 'y', data=df, fit_reg=False, scatter_kws={"s":200})
```

```
<seaborn.axisgrid.FacetGrid at 0x1acaa830d08>
```



이제 L2 norm을 활용하여 원점에서 제일 가까운 거리에 위치한 점이 어디인지 찾아보자.

```
from numpy import linalg

dt_point = df.values
dt_point

# 원점(0, 0)으로부터 어디가 제일 가까울까?
linalg.norm(dt_point, ord=2, axis=1)
```

```
array([6.40312424, 8.60232527, 6.40312424])
```

위 데이터에서는 첫번째(4, 5)와 세번째 데이터(5, 4)의 유클리디안 거리(L2 norm)을 계산하면 같은 값이므로 둘다 원점(0, 0)에 가장 가깝다. 당연히 randint를 반복 실행하면 결과 값은 달라진다.

다음은 L1 norm을 활용하여 걸음 수를 측정해 보자.

```
# L1 norm으로 계산
# 3명이 주사위를 3번 던진 후 걸음 수를 측정한다.

df = pd.DataFrame(np.random.randint(low=-3, high=3, size=(3,3)))
df.columns = ["move_1", "move_2", "move_3"]
print(df)

# 누가 가장 많이 걸었을까?
walk = df.values
linalg.norm(walk, ord=1, axis = 1)
```

마찬가지로 numpy의 randint 함수를 이용해서 -3부터 3까지의 숫자로 3행 3열 형식의 랜덤한 df 변수를 만든다. 각 열의 이름은 move_1, move_2, move_3로 한다. 아래와 같은 데이터 프레임이 만들어졌다.

	move_1	move_2	move_3
0	0	0	-3
1	-2	-2	-2
2	-3	-1	0

이제 L1 norm을 활용하여 누가 가장 많이 걸었는지 찾아보자.

```
array([3., 6., 4.])
```

L1 norm은 단순한 거리의 절대값의 합이므로 첫번째 행 0+0+|-3|=3, 두번째 행 |-2|+|-2|+|-2|=6, 세번째 행 |-3|+|-1|+0=4 와 같은 결과가 나온다. 즉, 두번째 행의 사람이 가장 많이 걸었다.

다음 시간에는 KNN을 본격적으로 구현해보자.