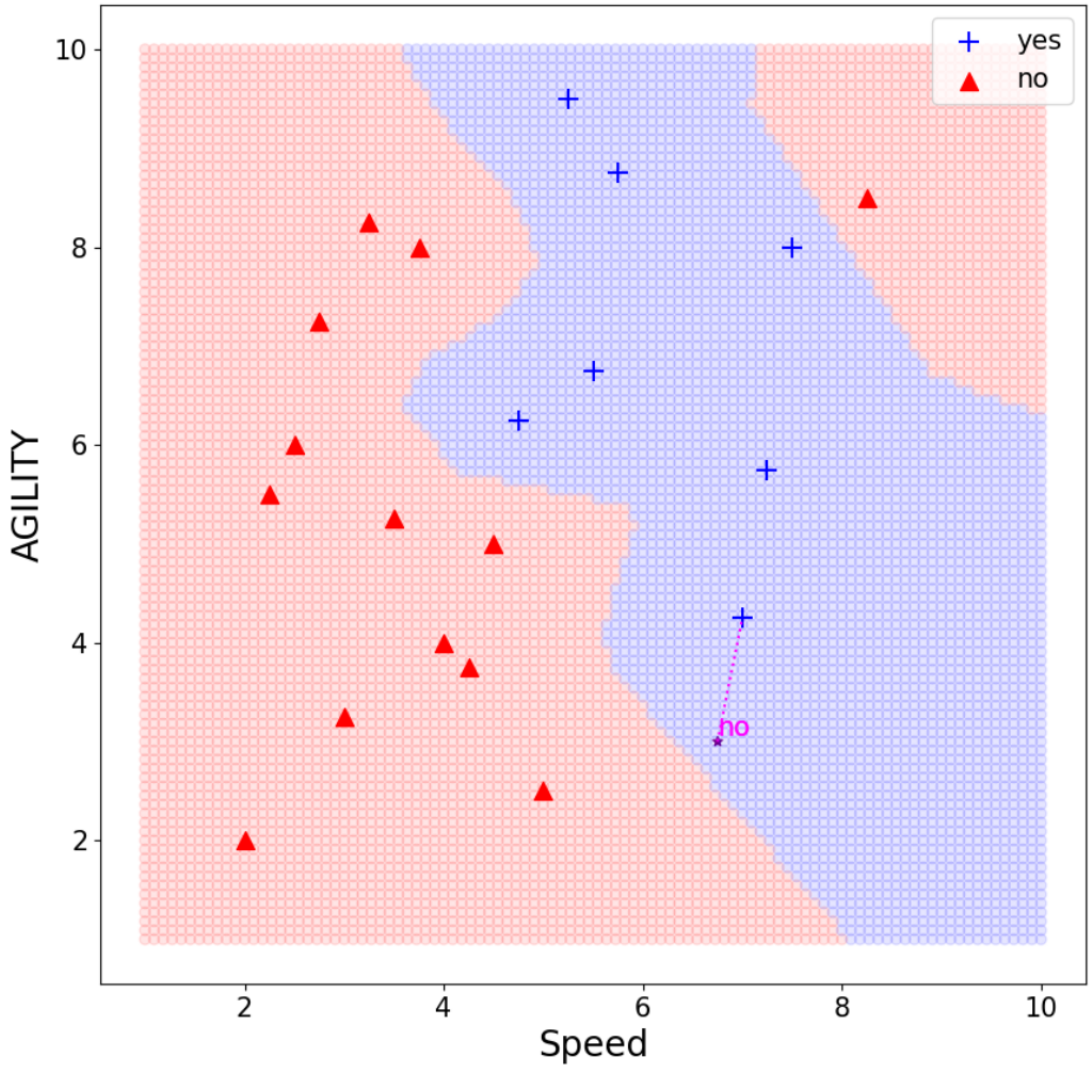
ML Day18 (Matplotlib) (K-Nearest Neighbor)

▼ step(4) : data형태 = DataFrame

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
import pandas as pd

athletes = [[2.50, 6.00, 'no'],
            [3.75, 8.00, 'no'],
            [2.25, 5.50, 'no'],
            [3.25, 8.25, 'no'],
            [2.75, 7.25, 'no'],
            [4.50, 5.00, 'no'],
            [3.50, 5.25, 'no'],
            [3.00, 3.25, 'no'],
            [4.00, 4.00, 'no'],
            [4.25, 3.75, 'no'],
            [2.00, 2.00, 'no'],
            [5.00, 2.50, 'no'],
            [8.25, 8.50, 'no'],
            [5.75, 8.75, 'yes'],
            [4.75, 6.25, 'yes'],
            [5.50, 6.75, 'yes'],
            [5.25, 9.50, 'yes'],
            [7.00, 4.25, 'yes'],
            [7.50, 8.00, 'yes'],
            [7.25, 5.75, 'yes']]

test_data = [6.75, 3.00]

athletes_df = pd.DataFrame(athletes,
                           columns=['SPEED', 'AGILITY', 'DRAFT'])

X = athletes_df[['SPEED', 'AGILITY']].to_numpy()  # to_numpy(): series 또는 DataFrame에서 데이터를 접근 또는 추출할 수 있는 numpy 메서드 - (c
y = athletes_df['DRAFT'].values                   # values: series 또는 DataFrame에서 데이터를 접근 또는 추출할 수 있는 numpy 메서드 - (20,

fig, ax = plt.subplots(figsize=(10, 10))

# 'yes'가 7개인 (7, 2)형태의 X_pos, 'no'가 13개인 (13, 2) 형태의 X_neg 변수 생성
X_pos, X_neg = X[y == 'yes'], X[y == 'no']

# main scatter 표현 (X_pos[:, 0] => 7개의 데이터, 'DRAFT'='yes'값을 가진 speed 데이터)
ax.scatter(X_pos[:, 0], X_pos[:, 1], color='blue',
           marker='+', s=130, label='yes')
ax.scatter(X_neg[:, 0], X_neg[:, 1], color='red',
           marker='^', s=100, label='no')

# test_data scatter 표현
ax.scatter(test_data[0], test_data[1],
           c='purple',
           s=90,
           marker='*')

# test_data와 X data간의 Euclidean distance를 numpy연산을 통해 계산하여 test_X 변수에 대입 후 argsort함수로 오름차순sorting ->정렬된 값들의 index값
test_X = X - test_data
test_X = np.array((test_X[:, 0]**2 + test_X[:, 1]**2)**1/2)
sort_list = np.argsort(test_X)                                  # (20, ) array형태
```

```
K = 1

# sort_list[:K] 만큼 for문으로 plot을 그린다.(x = [test_data[0], X[sort_list[test_line]][0]], y = [test_data[1], X[sort_list[test_line]
for test_line in range(len(sort_list))[:K]:
    ax.plot([test_data[0], X[sort_list[test_line]][0]], [test_data[1], X[sort_list[test_line]][1]],
            color='fuchsia',
            linestyle=':')

# sort_list[:K] 만큼 for문으로 [:K]까지 sort_list[yn]과 가장 근접하는 athletes의 해당하는 draft값('yes', 'no')의 합산을 해준다.
for yn in range(len(sort_list))[:K]:
    yes_sum = 0
    no_sum = 0
    if sort_list[yn] == 'yes':
        yes_sum += 1
    else:
        no_sum += 1

# 위에서 나온 yes_sum, no_sum의 크기를 비교하여 큰 값에 해당하는 draft값을 test_data 위치에 표현
for _ in range(2):
    if yes_sum > no_sum:
        ax.text(x=test_data[0], y=test_data[1],
                va='bottom',
                ha='left',
                s='yes',
                c='fuchsia',
                fontsize=15)
    else:
        ax.text(x=test_data[0], y=test_data[1],
                va='bottom',
                ha='left',
                s='no',
                c='fuchsia',
                fontsize=15)

#### meshgrid
xx = np.linspace(1, 10, 100)
yy = np.linspace(1, 10, 100)
xx, yy = np.meshgrid(xx, yy)

# 2개의 column을 가진 원본 데이터들과 맞추기 위해 reshape을 하여 hstack
xy_data = np.hstack([xx.reshape(-1, 1), yy.reshape(-1, 1)])

preds = []
for xy_data in xy_data:
    dis = np.sum(((X - xy_data)**2), axis=1)        # meshgrid 각 점과 X data들 간의 Euclidean distance, 크기만 비교하면 되기에 루트를 씌우는
    dist_argsort = np.argsort(dis)                  # for문을 돌며 계산된 e_distance를 np.argsort로 정렬 후 변수에 계속 대입
    close_K_indices = dist_argsort[:K]              # 입력받은 K로 distances_argsort를 슬라이싱하여 close_K_indices 대입
    close_yn = y[close_K_indices]                   # index값으로 슬라이싱한 것을 close_K_indices로 y의 index값을 찾은 후 close_yn 변수에 대입
    uniq, cnts = np.unique(close_yn, return_counts=True)   # y[close_K_indices]에서 고유한 원소들('no', 'yes')을 뽑아주고, 그것들의 개수를 c
    pred = uniq[np.argmax(cnts)]                    # argmax: 함수 내 array와 비슷한 형태를 넣어주면 가장 큰 원소의 index를 반환 ('yes'와 'no'
    if pred == 'yes': preds.append(1)
    elif pred == 'no': preds.append(0)

ax.scatter(xx, yy, c=preds, cmap='bwr_r', alpha=0.1)   # cmap='bwr_r' : 빨간색과 파란색으로 구성된 tab

ax.set_xlabel("Speed", fontsize=20)
ax.set_ylabel("AGILITY", fontsize=20)
ax.legend(fontsize=15)
ax.tick_params(labelsize=15)

plt.show()
```


yes
no
AGILITY
Speed
10
8
6
4
2
2
4
6
8
10
no