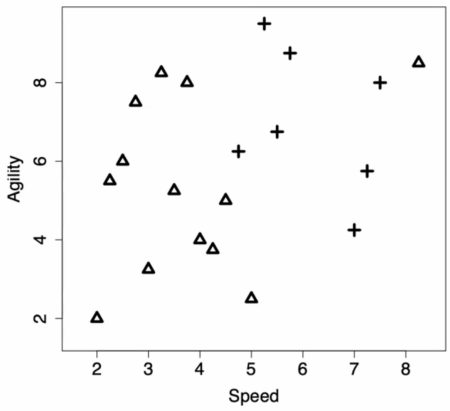
ML Day11 (Euclidean distance, Manhattan distance)

Feature Space

ID	Speed	Agility	Draft
1	2.50	6.00	no
2	3.75	8.00	no
3	2.25	5.50	no
4	3.25	8.25	no
5	2.75	7.50	no
6	4.50	5.00	no
7	3.50	5.25	no
8	3.00	3.25	no
9	4.00	4.00	no
10	4.25	3.75	no

ID	Speed	Agility	Draft
11	2.00	2.00	no
12	5.00	2.50	no
13	8.25	8.50	no
14	5.75	8.75	yes
15	4.75	6.25	yes
16	5.50	6.75	yes
17	5.25	9.50	yes
18	7.00	4.25	yes
19	7.50	8.00	yes
20	7.25	5.75	yes



<Id 5, Id 12>

▼ **ID 5, 12 간의 Euclidean distance, Manhattan distance(for구문으로 값을 구함)**

```python
# Euclidean distance
v1, v2 = [2.75, 7.50], [5.00, 2.50]

diff_square_sum = 0
for dim_idx in range(len(v1)):
    diff_square_sum += (v1[dim_idx] - v2[dim_idx])**2
e_distance = diff_square_sum ** 0.5
print("Euclidean distance: ", e_distance)

# Manhattan distance
v1, v2 = [2.75, 7.50], [5.00, 2.50]
m_distance = 0
for dim_idx in range(len(v1)):
    sub = v1[dim_idx] - v2[dim_idx]
    if sub < 0:
        m_distance += -sub
    else:
        m_distance += sub
print("Manhattan distance: ", m_distance)

# list in list
instance = [[2.75, 7.50],
            [5.00, 2.50]]

diff = 0
for dim_idx in range(len(instance[0])):
    diff += (instance[0][dim_idx] - instance[1][dim_idx])**2
e_distance = diff ** 0.5
print("E_distance:", e_distance)

abs_sum = 0
for dim_idx in range(len(instance[0])):
    diff = instance[0][dim_idx] - instance[1][dim_idx]
    if diff < 0:
        abs_sum += -diff
    else:
        abs_sum += diff
m_distance = abs_sum
print("M_distance:", m_distance)
```

▼ [7]: for 구문을 이용하여 e_distance를 구함. v1 list의 길이만큼 v1, v2에 해당하는 dim_idx의 차에 제곱을 하여 합산해준 값(diff_square_sum)의 제곱근이e_distance

▼ [15]: v1, v2 list의 같은 index에 해당하는 값들의 차를 sub 변수로 지정해주고 그 값(sub)이 음수인 경우 -sub 를 해주어 양수로 변환해준 후 합산하여 준다. (m_distance)

▼ [23]: list 안에 list가 들어가는 형태의 행렬을 for구문으로 e_distance와 m_distance를 구함.

▼ [28]: instance[0]의 길이만큼(0~1) instance의 [0] index에 해당하는 list의 dim_idx에서 instance[1]에 해당하는 dim_idx를 빼준 후 제곱을 하여 diff 변수에 합산하여 대입해준다.(e_distance)

▼ [33]: abs_sum 변수(instance의 원소 간 차의 절대값의 합산)를 0으로 지정

▼ [34]: for문을 이용하여 instance[0]의 길이만큼(0~1) instance[0]에 해당하는 dim_idx에서 instance[1]에 해당하는 dim_idx를 빼준 값들을 diff에 대입. (diff값이 0보다 작은 경우 -diff을 하여 합산해서 대입, 그렇지 않은 경우 그대로 합산하여 diff에 대입)

```
Euclidean distance:  5.482928049865327
Manhattan distance:  7.25
E_distance: 5.482928049865327
M_distance: 7.25
```

▼ **Numpy를 이용하여 e_distance, m_distance 구하기**

```python
# numpy를 이용한 E_distance
d5 = np.array([2.75, 7.50])
d12 = np.array([5.00, 2.50])

e_distance = np.sqrt(np.sum((d5 - d12)**2))
print("E_distance: ", e_distance)

# numpy를 이용한 M_distance
d5 = np.array([2.75, 7.50])
d12 = np.array([5.00, 2.50])

m_distance = np.sum(np.abs(d5 - d12))
print("M_distance: ", m_distance)

# list in list numpy를 이용
instance = [[2.75, 7.50],
            [5.00, 2.50]]

d_v = np.array([[2.75, 7.50], [5.00, 2.50]])

e_distance = np.sqrt(np.sum((d_v[0] - d_v[1])**2))
print("E_distance:", e_distance)

m_distance = np.sum(np.abs(d_v[0] - d_v[1]))
print("M_distance:", m_distance)
```

▼ [4]: numpy의 array함수를 이용해 list를 행렬로 변환하여 준다.

▼ [7]: d5행렬과 d12행렬의 차에 제곱을 하여준 값을 sum함수로 합산하여주고, 계산된 값을 sqrt함수로 제곱근을 구하여준다.(e_distance)

▼ [14]: d5행렬과 d12행렬의 차이의 절대값을 sum함수로 합산하여 준다. (m_distance)

```
E_distance:  5.482928049865327
M_distance:  7.25
E_distance: 5.482928049865327
M_distance: 7.25
```

▼ **각 행렬의 원소와 test_instance 간의 Euclidean distance 구하기(row)**

```python
import numpy as np
instances = np.array([[5., 2.5, 3],
                      [2.75, 7.50, 4],
                      [9.10, 4.5, 5],
                      [8.9, 2.3, 6]])
test_instance = instances[0]

distances = []
for instance in instances:
    distance = np.sqrt(np.sum((instance - test_instance)**2))
    distances.append(distance)
print(distances)
```

▼ [6]: test_instance는 instances의 [0]index로 지정해 준다. (5., 2.5, 3)

▼ [9]: array행렬로 변환된 instances를 for문으로 test_instance와 각 instance간의 Euclidean distance를 구하고 distances list에 append를 해준다.

```
[0.0, 5.573374202401989, 4.980963762164908, 4.924428900898053]
```

▼ **각 행렬의 원소와 test_instance 간의 Euclidean distance 구하기(column)**

```python
# 각 행렬의 원소와 test_instance 간의 Euclidean distance 구하기(column)
import numpy as np
instances = np.array([[5., 2.5, 3],
                      [2.75, 7.50, 4],
                      [9.10, 4.5, 5],
                      [8.9, 2.3, 6]])
test_instance = instances[:, 0]

n_cols = instances.shape[1]
distances = []
for col_idx in range(n_cols):
    instance = instances[:, col_idx]
    distance = np.sqrt(np.sum((instance - test_instance) ** 2))
    distances.append(distance)
print(distances)
```

▼ [7]: test_instance = 각 instance의 행이 모두 포함된 0번 column을 test_instance로 지정 (5., 2.75, 9.10, 6.9)

▼ [9]: instances.shape = (4, 3) 에서 [1]index 값 = 3 을 n_cols로 지정

▼ [11]: for문으로 n_cols만큼 instance는 순서대로 instances의 모든 행과 col_idx에 해당하는 값으로 지정하고, distance는 numpy로 e_distance(column index에 해당하는 값을 기준으로)를 구한다.

```
[0.0, 9.671220191888922, 5.548197905626655]
```

<Id 12, Id 17>

▼ **Euclidean Distance 와 Manhattan Distance (numpy 이용)**

```python
import numpy as np

# numpy를 이용한 E_distance
d17 = np.array([5.25, 9.50])
d12 = np.array([5.00, 2.50])

e_distance = np.sqrt(np.sum((d17 - d12)**2))
print("E_distance: ", e_distance)

# numpy를 이용한 M_distance
d17 = np.array([5.25, 9.50])
d12 = np.array([5.00, 2.50])

m_distance = np.sum(np.abs(d17 - d12))
print("M_distance: ", m_distance)

# list in list numpy를 이용
instance = [[2.75, 7.50],
            [5.00, 2.50]]

d_v = np.array([[5.25, 9.50], [5.00, 2.50]])

e_distance = np.sqrt(np.sum((d_v[0] - d_v[1])**2))
print("E_distance:", e_distance)

m_distance = np.sum(np.abs(d_v[0] - d_v[1]))
print("M_distance:", m_distance)
```

▼ id 12와 id 17 간의 거리(euclidean distance, manhattan distance)를 구함

```
E_distance:  7.00446286306095
M_distance:  7.25
E_distance: 7.00446286306095
M_distance: 7.25
```