# 기계학습 3주차 과제

## 재배환경 별 작물 종류 예측

컴퓨터공학과
17011584 정재경

---

## 개요

주어진 데이터셋은 재배환경에 대한 정보가 포함돼있어, 이에 재배하기 적합한 작물의 종류를 나타낸다.
제공된 정보에 따라서 가장 적합한 작물 하나를 선택하는 문제로, 분류문제에 해당한다.
이번 과제는 K-Nearest Neighbors 방식을 사용해서 해결해보았다.

## 변경 가능한 하이퍼파라미터

scikit learn 에서 제공하는 KNeighborsClassifier 공식문서에 따르면 커스터마이징이 가능한 하이퍼 파라미터에는 n_neighbors, weights, algorithm, leaf_size, p, metric, metric_params, n_jobs로 총 7가지이다.
주요한 하이퍼 파라미터의 의미를 살펴보면

1. n_neighbors - KNN알고리즘에서의 K값을 의미한다. 일반적으로 가장 가까운 것들을 투표의 형식으로 뽑은 것이기 때문에 짝수를 사용하면 동점이 나오는 경우가 생길 수 있기 때문에 홀수를 사용하는 것이 좋다
2. weights - 거리를 처리하는 방식이다. 가능한 값들은 uniform과 distance 가 있다. uniform은 거리에 상관없이 모두 같은 영향력을 행사하는 것이고, distance의 경우는 거리가 가까울 수록 더 높은 점수를 주는 방식이다. default 값은 uniform이다.
3. algorithm - 거리를 계산할 때 사용할 알고리즘이다. default 값은 auto로, 들어온 값에 따라서 최적의 알고리즘을 선택한다.
4. leaf_size - 거리계산 알고리즘에 사용될 파라미터 값이다.
5. p - Minkowski 공간에서 사용될 p 값이다. p가 1일 때는 L1거리 (Manhattan Distance) 2일 때는 L2거리 (Euclidean distance)와 같다.

## 실험

이번 과제에서 변경을 해보며 결과값의 차이 추이를 본 하이퍼파라미터는 n_neighbors이다.
우선, 다음과 같은 함수를 작성하여 n값을 번경해가며 모델을 학습시키고, 학습 데이터셋과 테스트 데이터셋의 정확도 점수를 계산하여 result 배열에 저장했다.

```
def _predict(n, train_set, test_set, train_ans, test_ans, size, result:list, weights):
    knn = KNeighborsClassifier(n_neighbors=n, p=2, weights=weights)
    knn.fit(train_set, train_ans)
    train_pred = knn.predict(train_set)
    test_pred = knn.predict(test_set)

    current = {
        'n': n,
        'test size': size,
        'weights': weights,
        'Train': accuracy_score(train_ans, train_pred),
        'Test': accuracy_score(test_ans, test_pred)
        }
    result.append(current)
```

이렇게 계산된 결과를 i값을 2부터 20까지 넣어서 점수를 기준으로 정렬해본 결과, n=3일 때가 가장 높은 점수가 나왔다.

```
result = []
for i in range (2, 21):
    _predict(i, x, x, y, y, 1, result, 'uniform')
pd.DataFrame(result).sort_values(by=['Train', 'Test'], ascending=False)
```

✓ 0.2s

| | n | test size | weights | Train | Test |
|---|---|---|---|---|---|
| 1 | 3 | 1 | uniform | 0.993939 | 0.993939 |
| 3 | 5 | 1 | uniform | 0.992121 | 0.992121 |
| 0 | 2 | 1 | uniform | 0.990303 | 0.990303 |
| 2 | 4 | 1 | uniform | 0.987879 | 0.987879 |
| 5 | 7 | 1 | uniform | 0.987273 | 0.987273 |
| 7 | 9 | 1 | uniform | 0.986061 | 0.986061 |
| 4 | 6 | 1 | uniform | 0.984848 | 0.984848 |
| 9 | 11 | 1 | uniform | 0.981818 | 0.981818 |
| 6 | 8 | 1 | uniform | 0.981212 | 0.981212 |
| 8 | 10 | 1 | uniform | 0.981212 | 0.981212 |
| 11 | 13 | 1 | uniform | 0.981212 | 0.981212 |
| 13 | 15 | 1 | uniform | 0.979394 | 0.979394 |
| 12 | 14 | 1 | uniform | 0.978788 | 0.978788 |
| 10 | 12 | 1 | uniform | 0.978182 | 0.978182 |
| 15 | 17 | 1 | uniform | 0.977576 | 0.977576 |
| 17 | 19 | 1 | uniform | 0.976970 | 0.976970 |
| 14 | 16 | 1 | uniform | 0.975758 | 0.975758 |
| 16 | 18 | 1 | uniform | 0.974545 | 0.974545 |

## 결론

따라서 이 데이터셋은 N=3일 때가 가장 성능이 좋을 것이라 예측하였다. 눈여겨볼 만한 점은 N=3, N=5일 때의 성능이 N=2, N=4일 때 보다 높게 나왔다. 이는 “일반적으로 홀수 K의 성능이 짝수 K의 성능보다 좋다”는 것을 실험적으로 확인할 수 있었다.

실제 Kaggle에 제출할 때도 N=3으로 제출한 점수가 가장 높았다.

노트북 링크: https://github.com/therealjamesjung/ML_2022/blob/master/Assignment%203/Assignment-3.ipynb