1 확인학습

- 유방암(breast cancer) 진단 데이터셋
- 유방암 진단 사진으로부터 측정한 종양(tumar)의 특정값을 사용하여 종양이 양성인지 음성인지
- 악성(malignant)인지를 판별하는 데이터

In [11]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#한글패치
import platform
from matplotlib import font_manager , rc

if platform.system() == 'Darwin':
  rc('font' , family = 'AppleGothic')
elif platform.system() == 'Windows':
  path = 'C:/Windows/Fonts/malgun.ttf'
  font_name = font_manager.FontProperties(fname = path).get_name()
  rc('font' , family = font_name)
else:
  print('모름')
plt.rcParams['axes.unicode_minus'] = False
import warnings
warnings.filterwarnings('ignore')
```

executed in 2.00s, finished 17:31:38 2023-10-23

In [6]:
```
# 와인 데이터 로드
from sklearn.datasets import load_wine
y = load_wine()
y
```
executed in 25ms, finished 17:26:24 2023-10-23

Out[6]:
```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
         1.065e+03],
        [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
         1.050e+03],
        [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
         1.185e+03],
        ...,
        [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
         8.350e+02],
        [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
         8.400e+02],
        [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
         5.600e+02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
        2, 2]),
 'frame': None,
 'target_names': array(['class_0', 'class_1', 'class_2'], dtype='<U7'),
 'DESCR': '.. _wine_dataset:₩n₩nWine recognition dataset₩n------------------------₩n₩n**Data Set Characteristi
cs:**₩n₩n    :Number of Instances: 178₩n    :Number of Attributes: 13 numeric, predictive attributes and the c
lass₩n    :Attribute Information:₩n ₩t₩t- Alcohol₩n ₩t₩t- Malic acid₩n ₩t₩t- Ash₩n₩t₩t- Alcalinity of ash  ₩n
₩t₩t- Magnesium₩n₩t₩t- Total phenols₩n ₩t₩t- Flavanoids₩n ₩t₩t- Nonflavanoid phenols₩n ₩t₩t- Proanthocyanins₩n
₩t₩t- Color intensity₩n ₩t₩t- Hue₩n ₩t₩t- OD280/OD315 of diluted wines₩n ₩t₩t- Proline₩n₩n    - class:₩n
- class_0₩n            - class_1₩n            - class_2₩n₩t₩t₩n    :Summary Statistics:₩n    ₩n    ===========
================== ==== ===== ======= =====₩n                                         Min   Max   Mean     SD₩n    =
============================ ==== ===== ======= =====₩n    Alcohol:                          11.0  14.8    13.0
0.8₩n    Malic Acid:                    0.74  5.80    2.34  1.12₩n    Ash:                              1.36  3.23
2.36  0.27₩n    Alcalinity of Ash:          10.6  30.0    19.5   3.3₩n    Magnesium:                        70.0
162.0    99.7  14.3₩n    Total Phenols:              0.98  3.88    2.29  0.63₩n    Flavanoids:
0.34  5.08    2.03  1.00₩n    Nonflavanoid Phenols:       0.13  0.66    0.36  0.12₩n    Proanthocyanins:
0.41  3.58    1.59  0.57₩n    Colour Intensity:            1.3  13.0     5.1   2.3₩n    Hue:
0.48  1.71    0.96  0.23₩n    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71₩n    Proline:
278  1680     746   315₩n    ============================ ==== ===== ======= =====₩n₩n    :Missing Attribute
Values: None₩n    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)₩n    :Creator: R.A. Fisher₩n
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)₩n    :Date: July, 1988₩n₩nThis is a copy of UCI ML Win
e recognition datasets.₩nhttps://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data₩n₩nThe data
is the results of a chemical analysis of wines grown in the same₩nregion in Italy by three different cultivato
rs. There are thirteen different₩nmeasurements taken for different constituents found in the three types of₩nw
ine.₩n₩nOriginal Owners: ₩n₩nForina, M. et al, PARVUS - ₩nAn Extendible Package for Data Exploration, Classifi
cation and Correlation. ₩nInstitute of Pharmaceutical and Food Analysis and Technologies,₩nVia Brigata Salern
o, 16147 Genoa, Italy.₩n₩nCitation:₩n₩nLichman, M. (2013). UCI Machine Learning Repository₩n[https://archive.i
cs.uci.edu/ml]. Irvine, CA: University of California,₩nSchool of Information and Computer Science. ₩n₩n.. topi
c:: References₩n₩n  (1) S. Aeberhard, D. Coomans and O. de Vel, ₩n  Comparison of Classifiers in High Dimensio
nal Settings, ₩n  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of  ₩n  Mathematics and St
atistics, James Cook University of North Queensland. ₩n  (Also submitted to Technometrics). ₩n₩n  The data was
used with many others for comparing various ₩n  classifiers. The classes are separable, though only RDA ₩n  ha
s achieved 100% correct classification. ₩n  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
₩n  (All results using the leave-one-out technique) ₩n₩n  (2) S. Aeberhard, D. Coomans and O. de Vel, ₩n  "THE
CLASSIFICATION PERFORMANCE OF RDA" ₩n  Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of ₩n
Mathematics and Statistics, James Cook University of North Queensland. ₩n  (Also submitted to Journal of Chemo
metrics).₩n',
 'feature_names': ['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline']}
```

In [9]:
```
y.keys()
```
executed in 14ms, finished 17:30:44 2023-10-23

Out[9]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names'])
```

In [13]:
```
## iris를 데이터 프레임 형태로 변환
df = pd.DataFrame(y.data , columns = y.feature_names)
df
```
executed in 32ms, finished 17:31:47 2023-10-23

Out[13]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.2
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.2
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.8
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.1
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.8
...	...	...	...	...	...	...	...	...	..
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.0
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.4
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.3
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.4
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.3

178 rows × 13 columns

In [14]:
```
from sklearn.model_selection import train_test_split
```
executed in 96ms, finished 17:34:47 2023-10-23

In [33]:
```
y.data[:,:5]
```
executed in 26ms, finished 17:41:41 2023-10-23

```
       [ 12.25,   4.72,   2.54,  21.  ,  89.  ],
       [ 12.53,   5.51,   2.64,  25.  ,  96.  ],
       [ 13.49,   3.59,   2.19,  19.5 ,  88.  ],
       [ 12.84,   2.96,   2.61,  24.  , 101.  ],
       [ 12.93,   2.81,   2.7 ,  21.  ,  96.  ],
       [ 13.36,   2.56,   2.35,  20.  ,  89.  ],
       [ 13.52,   3.17,   2.72,  23.5 ,  97.  ],
       [ 13.62,   4.95,   2.35,  20.  ,  92.  ],
       [ 12.25,   3.88,   2.2 ,  18.5 , 112.  ],
       [ 13.16,   3.57,   2.15,  21.  , 102.  ],
       [ 13.88,   5.04,   2.23,  20.  ,  80.  ],
       [ 12.87,   4.61,   2.48,  21.5 ,  86.  ],
       [ 13.32,   3.24,   2.38,  21.5 ,  92.  ],
       [ 13.08,   3.9 ,   2.36,  21.5 , 113.  ],
       [ 13.5 ,   3.12,   2.62,  24.  , 123.  ],
       [ 12.79,   2.67,   2.48,  22.  , 112.  ],
       [ 13.11,   1.9 ,   2.75,  25.5 , 116.  ],
       [ 13.23,   3.3 ,   2.28,  18.5 ,  98.  ],
       [ 12.58,   1.29,   2.1 ,  20.  , 103.  ],
       [ 13.17,   5.19,   2.32,  22.  ,  93.  ],
```

In [40]:
```
X_train , X_test , y_train , y_test = train_test_split(y.data , y.target , test_size = 0.3, random_state = 2)
```
executed in 5ms, finished 17:44:04 2023-10-23

In [18]:
```
from sklearn.neighbors import KNeighborsClassifier
```
executed in 179ms, finished 17:36:25 2023-10-23

In [21]:
```
# 최적의 K 찾기
for i in range(1,113,2):
    knnn = KNeighborsClassifier(n_neighbors = i)
    knnn.fit(X_train , y_train)
    score = knnn.score(X_train , y_train)
    print(score)
```
executed in 623ms, finished 17:36:49 2023-10-23

```
1.0
0.8629032258064516
0.782258064516129
0.7338709677419355
0.7580645161290323
0.7338709677419355
0.717741935483871
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7419354838709677
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7338709677419355
0.7258064516129032
0.7096774193548387
0.7258064516129032
0.7096774193548387
0.7096774193548387
0.7096774193548387
0.717741935483871
0.7016129032258065
0.7016129032258065
0.7016129032258065
0.6774193548387096
0.6774193548387096
0.6693548387096774
0.6774193548387096
0.6693548387096774
0.6935483870967742
0.6693548387096774
0.6693548387096774
0.6612903225806451
0.6612903225806451
0.6532258064516129
0.6532258064516129
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6370967741935484
0.6532258064516129
0.6532258064516129
0.6209677419354839
```

- K = 3일 때 , 점수가 가장 높음

In [41]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
```
executed in 4ms, finished 17:44:14 2023-10-23

In [23]:
```
df
```
executed in 24ms, finished 17:37:21 2023-10-23

Out[23]:

malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensit
1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.6
1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.3
2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.6
1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.8
2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.3
...	...	...	...	...	...	...	...	.
5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.7
3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.3
4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.2
2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.3
4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.2

columns

In [24]:
```
X_new = np.array([[14,1.7,2.4,15,126,2.79,3.05,0.27,2.28,5.6,1.02,3.9,1064]])
```
executed in 13ms, finished 17:38:02 2023-10-23

In [42]:
```
knn.fit(X_train , y_train)
```
executed in 21ms, finished 17:44:19 2023-10-23

Out[42]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [27]:
```
yhat = knn.predict(X_new)
```
executed in 14ms, finished 17:38:27 2023-10-23

In [28]:
```
y.target_names[yhat]
```
executed in 6ms, finished 17:38:39 2023-10-23

Out[28]:
```
array(['class_0'], dtype='<U7')
```

In [43]:
```
knn.score(X_test , y_test)
```
executed in 16ms, finished 17:44:29 2023-10-23

Out[43]:
```
0.6666666666666666
```

- 점수가 너무 낮다.

2 컬럼의 개수가 많으므로 , 5개만 뽑아서 해보기

In [44]:
```
X_train , X_test , y_train , y_test = train_test_split(y.data[:,:5] , y.target , test_size = 0.3, random_state =
```
executed in 6ms, finished 17:45:13 2023-10-23

In [38]:
```
for i in range(1,113,2):
    knnn = KNeighborsClassifier(n_neighbors = i)
    knnn.fit(X_train , y_train)
    score = knnn.score(X_train , y_train)
    print(score)
```
executed in 508ms, finished 17:43:06 2023-10-23

```
1.0
0.8951612903225806
0.7983870967741935
0.7741935483870968
0.7338709677419355
0.7258064516129032
0.717741935483871
0.7258064516129032
0.7258064516129032
0.7096774193548387
0.7016129032258065
0.6854838709677419
0.6854838709677419
0.6532258064516129
0.6612903225806451
0.6532258064516129
0.6532258064516129
0.6532258064516129
0.6451612903225806
0.6451612903225806
0.6451612903225806
0.6451612903225806
0.6209677419354839
0.6129032258064516
0.6209677419354839
0.6209677419354839
0.6209677419354839
0.6129032258064516
0.6048387096774194
0.6048387096774194
0.6048387096774194
0.6048387096774194
0.5887096774193549
0.5725806451612904
0.5725806451612904
0.5645161290322581
0.5564516129032258
0.5403225806451613
0.5403225806451613
0.5483870967741935
0.5564516129032258
0.5645161290322581
0.5564516129032258
0.5645161290322581
0.5564516129032258
0.5483870967741935
0.5403225806451613
0.5403225806451613
0.5403225806451613
0.5161290322580645
0.5241935483870968
0.5241935483870968
0.5
0.49193548387096775
0.49193548387096775
0.5080645161290323
```

- k = 3일때 가장 높음

In [46]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
```
executed in 6ms, finished 17:45:35 2023-10-23

In [47]:
```
knn.fit(X_train , y_train)
```
executed in 31ms, finished 17:45:36 2023-10-23

Out[47]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [48]:
```
X_new = np.array([[14,1.7,2.4,15,126]])
```
executed in 4ms, finished 17:45:58 2023-10-23

In [49]:
```
yhat = knn.predict(X_new)
```
executed in 14ms, finished 17:46:06 2023-10-23

In [50]:
```
y.target_names[yhat]
```
executed in 10ms, finished 17:46:12 2023-10-23

Out[50]:
```
array(['class_0'], dtype='<U7')
```

In [51]:
```
knn.score(X_test , y_test)
```
executed in 8ms, finished 17:46:18 2023-10-23

Out[51]:
```
0.5925925925925926
```

- 5개만 뽑았는데 낮음. 그러면 train으로 score를 내봤을 때 , 값이 1인 k = 1을 선택해보자

In [52]:
```
knn = KNeighborsClassifier(n_neighbors = 1)
```
executed in 17ms, finished 17:46:58 2023-10-23

In [53]:
```
X_train , X_test , y_train , y_test = train_test_split(y.data , y.target , test_size = 0.3, random_state = 2)
```
executed in 25ms, finished 17:47:17 2023-10-23

In [54]:
```
knn.fit(X_train , y_train)
```
executed in 17ms, finished 17:47:24 2023-10-23

Out[54]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [60]:
```
df
```
executed in 31ms, finished 17:49:06 2023-10-23

Out[60]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82
...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35

178 rows × 13 columns

In [57]:
```
X_new = np.array([[14,1.7,2.4,15,126,2.79,3.05,0.27,2.28,5.6,1.02,3.9,1064]])
```
executed in 5ms, finished 17:48:57 2023-10-23

In [58]:
```
yhat = knn.predict(X_new)
```
executed in 7ms, finished 17:48:59 2023-10-23

In [59]:
```
y.target_names[yhat]
```
executed in 16ms, finished 17:48:59 2023-10-23

Out[59]:
```
array(['class_0'], dtype='<U7')
```

In [55]:
```
knn.score(X_test , y_test)
```
executed in 17ms, finished 17:47:45 2023-10-23

Out[55]:
```
0.7037037037037037
```

In [56]:
```
knn.score(X_train , y_train)
```
executed in 20ms, finished 17:47:58 2023-10-23

Out[56]:
```
1.0
```

- train끼리 했을 땐 1.0 인데 test로 검증해봤을 때는 0.7로 , 차이가 상당히 나는 것을 볼 수 있다.

In [61]:
```
from sklearn.metrics import accuracy_score
```
executed in 15ms, finished 17:49:20 2023-10-23

In [62]:
```
n = 100
acc = np.zeros([n-1])

for i in range(1,n):
    clf = KNeighborsClassifier(n_neighbors = i).fit(X_train , y_train)
    yhat = clf.predict(X_test)
    acc[i-1] = accuracy_score(y_test , yhat)

print(acc)
```
executed in 754ms, finished 17:49:25 2023-10-23

```
[0.7037037  0.7037037  0.66666667 0.77777778 0.66666667 0.77777778
 0.74074074 0.7037037  0.68518519 0.75925926 0.74074074 0.74074074
 0.74074074 0.7037037  0.68518519 0.68518519 0.74074074 0.72222222
 0.7037037  0.7037037  0.7037037  0.7037037  0.7037037  0.7037037
 0.72222222 0.72222222 0.72222222 0.72222222 0.7037037  0.7037037
 0.74074074 0.7037037  0.7037037  0.68518519 0.7037037  0.7037037
 0.68518519 0.7037037  0.7037037  0.7037037  0.7037037  0.72222222
 0.72222222 0.72222222 0.72222222 0.72222222 0.74074074 0.74074074
 0.74074074 0.74074074 0.74074074 0.72222222 0.72222222 0.72222222
 0.72222222 0.72222222 0.72222222 0.72222222 0.7037037  0.74074074
 0.72222222 0.72222222 0.74074074 0.77777778 0.77777778 0.77777778
 0.75925926 0.77777778 0.75925926 0.7962963  0.75925926 0.74074074
 0.74074074 0.74074074 0.7037037  0.7037037  0.7037037  0.7037037
 0.7037037  0.72222222 0.72222222 0.72222222 0.72222222 0.72222222
 0.72222222 0.72222222 0.72222222 0.72222222 0.72222222 0.74074074
 0.72222222 0.72222222 0.7037037  0.7037037  0.7037037  0.7037037
 0.7037037  0.7037037  0.7037037 ]
```

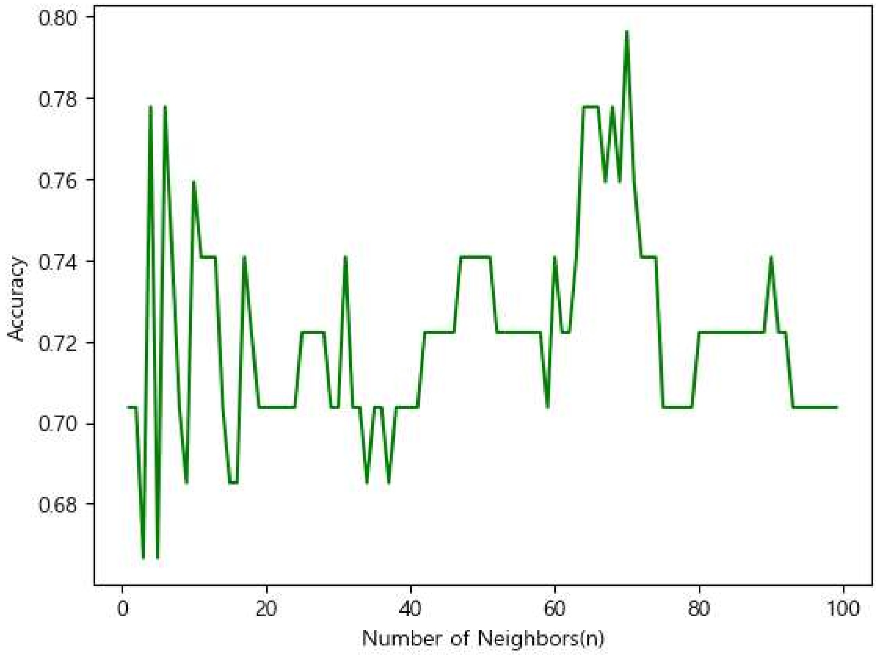
In [69]:
```

```
executed in 32ms, finished 17:51:13 2023-10-23

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[69], line 1
----> 1 acc.index(acc.max())

AttributeError: 'numpy.ndarray' object has no attribute 'index'
```

In [63]:
```
plt.plot(range(1,n) , acc , color = 'g')
plt.xlabel('Number of Neighbors(n)')
plt.ylabel('Accuracy')
plt.show()
```
executed in 137ms, finished 17:49:31 2023-10-23



- 그래프를 보니 , k가 70일 때 점수가 가장 높다.

In [64]:
```
knn = KNeighborsClassifier(n_neighbors = 70)
```
executed in 14ms, finished 17:50:20 2023-10-23

In [65]:
```
knn.fit(X_train , y_train)
```
executed in 6ms, finished 17:50:28 2023-10-23

Out[65]:
```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=70)
```

In [66]:
```
knn.score(X_train , y_train)
```
executed in 25ms, finished 17:50:36 2023-10-23

Out[66]: 0.6774193548387096

In [70]:
```
knn.score(X_test , y_test)
```
executed in 30ms, finished 17:51:48 2023-10-23

Out[70]: 0.7962962962962963

- 이건 train 점수가 오히려 낮다.