

2_K-Nearest Neighbors

July 6, 2025

Creado por:

Isabel Maniega

1 K- Nearest Neighbors

```
[1]: import warnings
warnings.filterwarnings("ignore")
```

```
[2]: from sklearn import datasets
```

```
[3]: dataset = datasets.load_breast_cancer()
dataset
```

```
[3]: {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
1.189e-01],
[2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
8.902e-02],
[1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
8.758e-02],
...,
[1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
7.820e-02],
[2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
1.240e-01],
[7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
7.039e-02]], shape=(569, 30)),
'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
```

```

1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 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, 0, 0, 0, 0, 0, 0, 1]),
'frame': None,
'target_names': array(['malignant', 'benign'], dtype='<U9'),
'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer Wisconsin (diagnostic)
dataset\n-----\n\n**Data Set
Characteristics:**\n\nNumber of Instances: 569\n\nNumber of Attributes: 30
numeric, predictive attributes and the class\n\nAttribute Information:\n    -
radius (mean of distances from center to points on the perimeter)\n    - texture
(standard deviation of gray-scale values)\n    - perimeter\n    - area\n    -
smoothness (local variation in radius lengths)\n    - compactness (perimeter^2 /
area - 1.0)\n    - concavity (severity of concave portions of the contour)\n
- concave points (number of concave portions of the contour)\n    - symmetry\n
- fractal dimension ("coastline approximation" - 1)\n\n    The mean, standard
error, and "worst" or largest (mean of the three\n    worst/largest values) of
these features were computed for each image,\n    resulting in 30 features. For
instance, field 0 is Mean Radius, field\n    10 is Radius SE, field 20 is Worst
Radius.\n\n    - class:\n                - WDBC-Malignant\n                - WDBC-
Benign\n\nSummary Statistics:\n\n=====
=====
=====
Min
Max\n=====
=====
=====
\nradius (mean):
6.981  28.11\ntexture (mean):                9.71  39.28\nperimeter
(mean):                43.79  188.5\narea (mean):
143.5  2501.0\nsmoothness (mean):                0.053  0.163\ncompactness
(mean):                0.019  0.345\nconcavity (mean):
0.0    0.427\nconcave points (mean):                0.0    0.201\nsymmetry
(mean):                0.106  0.304\nfractal dimension (mean):
0.05   0.097\nradius (standard error):                0.112  2.873\ntexture
(standard error):                0.36   4.885\nperimeter (standard error):
0.757  21.98\narea (standard error):                6.802  542.2\nsmoothness
(standard error):                0.002  0.031\ncompactness (standard error):
0.002  0.135\nconcavity (standard error):                0.0    0.396\nconcave points
(standard error):                0.0    0.053\nsymmetry (standard error):                0.008

```

```

0.079\nfractal dimension (standard error): 0.001 0.03\nradius (worst):
7.93 36.04\ntexture (worst): 12.02 49.54\nperimeter
(worst): 50.41 251.2\narea (worst):
185.2 4254.0\nsmoothness (worst): 0.071 0.223\ncompactness
(worst): 0.027 1.058\nconcavity (worst):
0.0 1.252\nconcave points (worst): 0.0 0.291\nsymmetry
(worst): 0.156 0.664\nfractal dimension (worst):
0.055 0.208\n===== \n\nMissing
Attribute Values: None\n\nClass Distribution: 212 - Malignant, 357 -
Benign\n\nCreator: Dr. William H. Wolberg, W. Nick Street, Olvi L.
Mangasarian\n\nDonor: Nick Street\n\nDate: November, 1995\n\nThis is a copy of
UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image
of a fine needle\naspirate (FNA) of a breast mass. They
describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating
plane described above was obtained using\nMultisurface Method-Tree (MSM-T) [K.
P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of
the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp.
97-101, 1992], a classification method which uses linear\nprogramming to
construct a decision tree. Relevant features\nwere selected using an exhaustive
search in the space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual
linear program used to obtain the separating plane\nin the 3-dimensional space
is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust
Linear\nProgramming Discrimination of Two Linearly Inseparable
Sets",\nOptimization Methods and Software 1, 1992, 23-34].\n\nThis database is
also available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-
prog/cpo-dataset/machine-learn/WDBC/\n\n.. dropdown:: References\n\n - W.N.
Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction\n for
breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on\n
Electronic Imaging: Science and Technology, volume 1905, pages 861-870,\n San
Jose, CA, 1993.\n - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast
cancer diagnosis and\n prognosis via linear programming. Operations Research,
43(4), pages 570-577,\n July-August 1995.\n - W.H. Wolberg, W.N. Street, and
O.L. Mangasarian. Machine learning techniques\n to diagnose breast cancer
from fine-needle aspirates. Cancer Letters 77 (1994)\n 163-171.\n',
'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean
area',
'mean smoothness', 'mean compactness', 'mean concavity',
'mean concave points', 'mean symmetry', 'mean fractal dimension',
'radius error', 'texture error', 'perimeter error', 'area error',
'smoothness error', 'compactness error', 'concavity error',
'concave points error', 'symmetry error',
'fractal dimension error', 'worst radius', 'worst texture',
'worst perimeter', 'worst area', 'worst smoothness',
'worst compactness', 'worst concavity', 'worst concave points',
'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv',

```

```
'data_module': 'sklearn.datasets.data'}
```

```
[4]: print("Características del dataset:")  
print(dataset.DESCR)
```

Características del dataset:

.. _breast_cancer_dataset:

Breast cancer Wisconsin (diagnostic) dataset

****Data Set Characteristics:****

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter² / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

- class:
 - WDBC-Malignant
 - WDBC-Benign

:Summary Statistics:

```
=====  =====  =====  
                                     Min    Max  
=====  =====  =====  
radius (mean):                      6.981  28.11  
texture (mean):                      9.71   39.28  
perimeter (mean):                   43.79  188.5  
area (mean):                        143.5  2501.0  
smoothness (mean):                   0.053  0.163
```

compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208
=====	=====	=====

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
<https://goo.gl/U2Uwz2>

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society,

pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in:

[K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

```
ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/
```

.. dropdown:: References

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

```
[5]: print("Información en el Dataset:")
      print(dataset.keys())
```

Información en el Dataset:

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

```
[6]: dataset.data
```

```
[7]: array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
          1.189e-01],
          [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
          8.902e-02],
          [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
          8.758e-02],
          ...,
          [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
```

```

7.820e-02],
[2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
1.240e-01],
[7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
7.039e-02]], shape=(569, 30))

```

```
[7]: dataset.feature_names
```

```
[7]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
'mean smoothness', 'mean compactness', 'mean concavity',
'mean concave points', 'mean symmetry', 'mean fractal dimension',
'radius error', 'texture error', 'perimeter error', 'area error',
'smoothness error', 'compactness error', 'concavity error',
'concave points error', 'symmetry error',
'fractal dimension error', 'worst radius', 'worst texture',
'worst perimeter', 'worst area', 'worst smoothness',
'worst compactness', 'worst concavity', 'worst concave points',
'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
[8]: import pandas as pd
```

```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df
```

```
[8]:
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	
..	
564	21.56	22.39	142.00	1479.0	0.11100	
565	20.13	28.25	131.20	1261.0	0.09780	
566	16.60	28.08	108.30	858.1	0.08455	
567	20.60	29.33	140.10	1265.0	0.11780	
568	7.76	24.54	47.92	181.0	0.05263	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	
2	0.15990	0.19740	0.12790	0.2069	
3	0.28390	0.24140	0.10520	0.2597	
4	0.13280	0.19800	0.10430	0.1809	
..	
564	0.11590	0.24390	0.13890	0.1726	
565	0.10340	0.14400	0.09791	0.1752	
566	0.10230	0.09251	0.05302	0.1590	

567	0.27700	0.35140	0.15200	0.2397
568	0.04362	0.00000	0.00000	0.1587

	mean fractal dimension	...	worst radius	worst texture	\
0	0.07871	...	25.380	17.33	
1	0.05667	...	24.990	23.41	
2	0.05999	...	23.570	25.53	
3	0.09744	...	14.910	26.50	
4	0.05883	...	22.540	16.67	
..	
564	0.05623	...	25.450	26.40	
565	0.05533	...	23.690	38.25	
566	0.05648	...	18.980	34.12	
567	0.07016	...	25.740	39.42	
568	0.05884	...	9.456	30.37	

	worst perimeter	worst area	worst smoothness	worst compactness	\
0	184.60	2019.0	0.16220	0.66560	
1	158.80	1956.0	0.12380	0.18660	
2	152.50	1709.0	0.14440	0.42450	
3	98.87	567.7	0.20980	0.86630	
4	152.20	1575.0	0.13740	0.20500	
..	
564	166.10	2027.0	0.14100	0.21130	
565	155.00	1731.0	0.11660	0.19220	
566	126.70	1124.0	0.11390	0.30940	
567	184.60	1821.0	0.16500	0.86810	
568	59.16	268.6	0.08996	0.06444	

	worst concavity	worst concave points	worst symmetry	\
0	0.7119	0.2654	0.4601	
1	0.2416	0.1860	0.2750	
2	0.4504	0.2430	0.3613	
3	0.6869	0.2575	0.6638	
4	0.4000	0.1625	0.2364	
..	
564	0.4107	0.2216	0.2060	
565	0.3215	0.1628	0.2572	
566	0.3403	0.1418	0.2218	
567	0.9387	0.2650	0.4087	
568	0.0000	0.0000	0.2871	

	worst fractal dimension
0	0.11890
1	0.08902
2	0.08758
3	0.17300


```

4          0.07678
..          ...
564        0.07115
565        0.06637
566        0.07820
567        0.12400
568        0.07039

```

[569 rows x 30 columns]

```
[9]: X = dataset.data
```

```
[10]: y = dataset.target
```

```
[11]: from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
[12]: from sklearn.preprocessing import StandardScaler
```

```
[13]: escalator = StandardScaler()
```

```
[14]: X_train = escalator.fit_transform(X_train)
X_test = escalator.fit_transform(X_test)
```

```
[15]: #Defino el algoritmo a utilizar
from sklearn.neighbors import KNeighborsClassifier

algoritmo = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p=2)
```

```
[16]: algoritmo.fit(X_train, y_train)
```

```
[16]: KNeighborsClassifier()
```

```
[17]: y_pred = algoritmo.predict(X_test)
y_pred
```

```
[17]: array([1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
          0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
          1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
          1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
          1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 1, 0, 1])
```

```
[18]: y_test
```

```
[18]: array([1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
          0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,
```

```

1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0,
1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 0, 1])

```

```
[19]: from sklearn.metrics import confusion_matrix
```

```
[20]: matriz = confusion_matrix(y_test, y_pred)
```

```
[21]: matriz
```

```
[21]: array([[44,  3],
          [ 0, 67]])
```

```
[22]: # Calculo de precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred)
print("Precisión del modelo:")
print(precision)
```

```

Precisión del modelo:
0.9571428571428572

```

```
[23]: # calculo para la exactitud del modelo

from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print("Exactitud del modelo:")
print(exactitud)
```

```

Exactitud del modelo:
0.9736842105263158

```

```
[24]: # Calcular la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred)
print("Sensibilidad del modelo")
print(sensibilidad)
```

```

Sensibilidad del modelo
1.0

```

```
[25]: # Calculo el puntaje F1 del modelo
from sklearn.metrics import f1_score

puntajef1 = f1_score(y_test, y_pred)
```

```
print("Puntaje F1 del modelo")  
print(puntajef1)
```

Puntaje F1 del modelo
0.9781021897810219

```
[26]: # Calculo la curva ROC -AUC del modelo  
from sklearn.metrics import roc_auc_score  
  
roc_auc = roc_auc_score(y_test, y_pred)  
print("Curva ROC - AUC del modelo:")  
print(roc_auc)
```

Curva ROC - AUC del modelo:
0.9680851063829787

Creado por:

Isabel Maniega