k-nearest neighbors

Paulo S. A. Sousa

2023-03-21

Abstract

K-nearest neighbors model is based on a very simple idea, but, in spite of that and astonishingly, performs very competitively in many cases.

This simple model is introduced here.

1 Introduction

The k-nearest neighbors model is based on a very simple idea: The predictions will consider only the information of the closest neighbors to the case we want to predict. For instance, in a problem of classification, if all closest neighbors are of `red` class, the new case will be predicted as being also of `red` class. In a regression problem, the prediction of a new case outcome variable will be the mean of the outcome variable of its closest neighbors.

We need first to define the number of neighbors to consider. This is a hyperparameter of the model, usually represented by k. Furthermore, we need to define a measure of distance, to determine the closest neighbors.

In spite of the simplicity of this model, its predictive performance is astonishingly quite competitive in many prediction problems.

2 Distance measure

There are several distance measures, but the most popular is the Euclidean distance, as it is computationally cheap to calculate.

The Euclidean distance between points $P_1 = (x_1, x_2, \cdots, x_n)$ and $P_2 = (y_1, y_2, \cdots, y_n)$, d, is given by the following formula:

$$d\left(P_1, P_2\right) = \sqrt{\left(x_1 - y_1\right)^2 + \left(x_2 - y_2\right)^2 + \cdots + \left(x_n - y_n\right)^2}.$$

Example 2.1 (Euclidean distance)**.** Given points $P_1 = (2, 4, 3)$ and $P_2 = (8, 3, 5)$, the Euclidean distance between them is:

$$d\left(P_1, P_2\right) = \sqrt{(2-8)^2 + (4-3)^2 + (3-5)^2} \approx 6.403.$$

3 KNN in classification problems

To produce a prediction of a new data point, P, KNN algorithm goes to find the k closest points to P in the dataset (these closest points are called *neighbors*), and uses the majority of class in the set of the neighbors as the prediction of the outcome variable corresponding to P. Clearly, the number of neighbors to consider, k, must be set beforehand.

Example 3.1 (KNN in classification)**.** Consider the following dataset, which corresponds to a classification problem, where the outcome variable can only assume A or B as values (there are only two classes, A and B; that is a binary classification problem):

	X1	X2	Y
Data points			
0	1	4	B
1	6	2	B
2	5	3	A
3	3	1	A
4	2	9	B
5	1	2	A

Considering $k = 3$, to determine the prediction of the outcome variable of a new data point, $P = (3, 5)$, by using KNN, we need to compute the Euclidean distances between the new data point and all data points of the dataset:

	Distance
Data points	
0	2.236
1	4.243
2	2.828
3	4.000
4	4.123
5	3.606

The 3 neighbors of the new data point ($k = 3$) are the closest points to P:

- Data point 0 (class B);

- Data point 2 (class A);
- Data point 5 (class A).

Since the 3 neighbors have class A as majoritary, the prediction for P is class A.

When the chosen value for k is even, ties may occur. In case od ties, a class is randomly selected from the classes ofthe neighbors as the prediction.

4 KNN in regression problems

Likewise in classification, to produce a prediction of a new data point, P, KNN algorithm goes to find the k closest points to P in the dataset. The prediction is the mean of the outcome variable of the k-nearest neighbors. Again, the number of neighbors to consider, k, must be set beforehand.

Example 4.1 (KNN in regression)**.** Consider the following dataset, which corresponds to a regression problem:

Data points	X1	X2	Y
0	1	4	8
1	6	2	5
2	5	3	7
3	3	1	10
4	2	9	3
5	1	2	6

Considering $k = 3$, to determine the prediction of the outcome variable of a new data point, $P = (3, 5)$, by using KNN, we need to compute the Euclidean distances between the new data point and all data points of the dataset:

Data points	Distance
0	2.236
1	4.243
2	2.828
3	4.000
4	4.123
5	3.606

The 3 neighbors of the new data point ($k = 3$) are the closest points to P:

- Data point 0 (Y=8);

- Data point 2 (Y=7);
- Data point 5 (Y=6).

The prediction for P is the mean of 8, 7 and 6: 7.

5 Python implementation of the examples

For classification, we need `KNeighborsClassifier` function:

```
from sklearn.neighbors import KNeighborsClassifier
```

We will use a pipeline as usual. Therefore, we need to load the respetive function:

```
from sklearn.pipeline import Pipeline
```

We can now run the model and get the wanted prediction:

```
df = pd.DataFrame({
  'X1': [1, 6, 5, 3, 2, 1],
  'X2': [4, 2, 3, 1, 9, 2],
  'Y': ['B', 'B', 'A', 'A', 'B', 'A']
})

X = df.drop('Y', axis=1)
y = df['Y']

pipe = Pipeline([
    ('knn', KNeighborsClassifier(n_neighbors=3))
])

X_new = pd.DataFrame({
    'X1': [3],
    'X2': [5]
})
pipe.fit(X, y)
pipe.predict(X_new)
```

```
array(['A'], dtype=object)
```

For regression, we need `KNeighborsRegressor` function:

```
from sklearn.neighbors import KNeighborsRegressor
```

The Pipeline function is already load. Hence, we can run the model and get the prediction for the new case:

```python
df = pd.DataFrame({
  'X1': [1, 6, 5, 3, 2, 1],
  'X2': [4, 2, 3, 1, 9, 2],
  'Y': [8, 5, 7, 10, 3, 6]
})

X = df.drop('Y', axis=1)
y = df['Y']

pipe = Pipeline([
    ('knn', KNeighborsRegressor(n_neighbors=3))
])

X_new = pd.DataFrame({
    'X1': [3],
    'X2': [5]
})
pipe.fit(X, y)
pipe.predict(X_new)
```

```
array([7.])
```