Nearest Neighbors

Understand the purpose of finding nearest neighbors for data points.

Chapter Goals:

- Learn how to find the nearest neighbors for a data observation

A. Finding the nearest neighbors

In Chapter 1, we mentioned that clustering is a method for grouping together similar data observations. Another method for finding similar data observations is the *nearest neighbors* approach. With this approach, we find the *k* most similar data observations (i.e. neighbors) for a given data observation (where *k* represents the number of neighbors).

In scikit-learn, we implement the nearest neighbors approach with the `NearestNeighbors` object (part of the `neighbors` module).

The code below finds the 5 nearest neighbors for a new data observation (`new_obs`) based on its fitted dataset (`data`).

```
data = np.array([
  [5.1, 3.5, 1.4, 0.2],
  [4.9, 3. , 1.4, 0.2],
  [4.7, 3.2, 1.3, 0.2],
  [4.6, 3.1, 1.5, 0.2],
  [5. , 3.6, 1.4, 0.2],
  [5.4, 3.9, 1.7, 0.4],
  [4.6, 3.4, 1.4, 0.3],
  [5. , 3.4, 1.5, 0.2],
  [4.4, 2.9, 1.4, 0.2],
  [4.9, 3.1, 1.5, 0.1]])

from sklearn.neighbors import NearestNeighbors
nbrs = NearestNeighbors()
nbrs.fit(data)
new_obs = np.array([[5. , 3.5, 1.6, 0.3]])
dists, knbrs = nbrs.kneighbors(new_obs)

# nearest neighbors indexes
print('{}\n'.format(repr(knbrs)))
# nearest neighbor distances
print('{}\n'.format(repr(dists)))
```

```
only_nbrs = nbrs.kneighbors(new_obs,

                            return_distance=False)
print('{}\n'.format(repr(only_nbrs)))
```

The `NearestNeighbors` object is fitted with a dataset, which is then used as the pool of possible neighbors for new data observations. The `kneighbors` function takes in new data observation(s) and returns the *k* nearest neighbors along with their respective distances from the input data observations. Note that the nearest neighbors are the neighbors with the smallest distances from the input data observation. We can choose not to return the distances by setting the `return_distance` keyword argument to `False`.

The default value for *k* when initializing the `NearestNeighbors` object is 5. We can specify a new value using the `n_neighbors` keyword argument.

```
data = np.array([
  [5.1, 3.5, 1.4, 0.2],
  [4.9, 3. , 1.4, 0.2],
  [4.7, 3.2, 1.3, 0.2],
  [4.6, 3.1, 1.5, 0.2],
  [5. , 3.6, 1.4, 0.2],
  [5.4, 3.9, 1.7, 0.4],
  [4.6, 3.4, 1.4, 0.3],
  [5. , 3.4, 1.5, 0.2],
  [4.4, 2.9, 1.4, 0.2],
  [4.9, 3.1, 1.5, 0.1]])

from sklearn.neighbors import NearestNeighbors
nbrs = NearestNeighbors(n_neighbors=2)
nbrs.fit(data)
new_obs = np.array([
  [5. , 3.5, 1.6, 0.3],
  [4.8, 3.2, 1.5, 0.1]])
dists, knbrs = nbrs.kneighbors(new_obs)

# nearest neighbors indexes
print('{}\n'.format(repr(knbrs)))
# nearest neighbor distances
print('{}\n'.format(repr(dists)))
```

In the code above, the first row of `knbrs` and `dists` correspond to the first data observation in `new_obs`, while the second row of `knbrs` and `dists` correspond to the second observation in `new_obs`.