Solution for Exercise M1.02

The goal of this exercise is to fit a similar model as in the previous notebook to get familiar with manipulating scikit-learn objects and in particular the `.fit/.predict/.score` API.

Let’s load the adult census dataset with only numerical variables

```
import pandas as pd

adult_census = pd.read_csv("../datasets/adult-census-numeric.csv")
data = adult_census.drop(columns="class")
target = adult_census["class"]
```

In the previous notebook we used `model = KNeighborsClassifier()`. All scikit-learn models can be created without arguments. This is convenient because it means that you don’t need to understand the full details of a model before starting to use it.

One of the `KNeighborsClassifier` parameters is `n_neighbors`. It controls the number of neighbors we are going to use to make a prediction for a new data point.

What is the default value of the `n_neighbors` parameter?

Hint: Look at the documentation on the scikit-learn website or directly access the description inside your notebook by running the following cell. This opens a pager pointing to the documentation.

```
from sklearn.neighbors import KNeighborsClassifier

KNeighborsClassifier?
```

We can see that the default value for `n_neighbors` is 5.

Create a `KNeighborsClassifier` model with `n_neighbors=50`

```
# solution
model = KNeighborsClassifier(n_neighbors=50)
```

Fit this model on the data and target loaded above

Skip to main content

```
# solution
model.fit(data, target)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=50)
```

Use your model to make predictions on the first 10 data points inside the data. Do they match the actual target values?

```
# solution
first_data_values = data.iloc[:10]
first_predictions = model.predict(first_data_values)
first_predictions
```

```
array([' <=50K', ' <=50K', ' <=50K', ' <=50K', ' <=50K', ' <=50K',
       ' <=50K', ' >50K', ' <=50K', ' <=50K'], dtype=object)
```

```
first_target_values = target.iloc[:10]
first_target_values
```

```
0     <=50K
1     <=50K
2     <=50K
3     <=50K
4     <=50K
5     <=50K
6     <=50K
7      >50K
8     <=50K
9      >50K
Name: class, dtype: object
```

```
number_of_correct_predictions = (
    first_predictions == first_target_values
).sum()
number_of_predictions = len(first_predictions)
print(
    f"{number_of_correct_predictions}/{number_of_predictions} "
    "of predictions are correct"
)
```

```
9/10 of predictions are correct
```

Skip to main content

Compute the accuracy on the training data.

```
# solution
model.score(data, target)
```

```
0.8290379545978042
```

Now load the test data from `"../datasets/adult-census-numeric-test.csv"` and compute the accuracy on the test data.

```
# solution
adult_census_test = pd.read_csv("../datasets/adult-census-numeric-test.csv")

data_test = adult_census_test.drop(columns="class")
target_test = adult_census_test["class"]

model.score(data_test, target_test)
```

```
0.8177909714402702
```

Looking at the previous notebook, the accuracy seems slightly higher with `n_neighbors=50` than with `n_neighbors=5` (the default value).