DABBEL®

EXERCISE: MATCH PREDICTIONS TO SENSOR READINGS

We have a table in which the readings of temperature sensors are stored. There is one row for every timestamp and one column for every sensor.

	Sensor1	Sensor2	Sensor3	Sensor4
2020-12-01 00:00:00	20.0	19.4	21.3	20.5
2020-12-01 00:11:34	20.1	19.4	21.2	20.4
2020-12-01 00:20:00	20.0	19.4	21.0	20.6

Table 1 Temperature sensor readings

We have another table that has temperature predictions. The format is the same as the previous table, rows are timestamps and columns are sensors. However, we can have null (or nan) values for some of the predictions.

Furthermore, the timestamps are not the same for both tables. This means, that if there is a timestamp 2020-12-01 00:00:00 in the readings table, there is not necessarily a 2020-12-01 00:00:00 timestamp in the predictions table.

Also, the number of predictions for a sensor does not have to match the number of readings for a sensor. For example, sensor 1 could have 100 readings and 200 predictions and sensor 2 has 100 readings and 50 predictions.

	Sensor1	Sensor2	Sensor3	Sensor4
2020-12-01 00:00:30	Null	19.5	21.1	20.3
2020-12-01 00:07:24	20.2	Null	Null	20.2
2020-12-01 00:17:15	19.9	19.6	Null	20.4
2020-12-01 00:19:00	20.0	Null	Null	Null
2020-12-01 00:21:30	20.1	Null	Null	20.5

Table 2 Temperature predictions

The goal of the exercise is to match the temperature readings with the closest prediction. Closest here means with the timestamp closest to it.

	Sensor	Temperature	Prediction
2020-12-01 00:00:00	Sensor2	19.4	19.5
2020-12-01 00:00:00	Sensor3	21.3	21.1
2020-12-01 00:00:00	Sensor4	20.5	20.3
2020-12-01 00:11:34	Sensor1	20.1	20.2
2020-12-01 00:11:34	Sensor4	20.4	20.2
2020-12-01 00:20:00	Sensor1	20.0	20.0

2020-12-01 00:20:00	Sensor2	19.4	19.6
2020-12-01 00:20:00	Sensor4	20.6	20.5

Table 3 Expected output

Table 3 shows the expected output. We want to include only the Timestamp + Sensor combination where we could match a prediction to a temperature reading.

Each prediction can only be used for one temperature. For example, the 19.6 prediction at 2020-12-01 00:17:15 of Sensor 2 is the closest to the 2020-12-01 00:11:34 and 2020-12-01 00:20:00 timestamps. Since its timestamp is closer to 2020-12-01 00:20:00, that is what it will be matched to. In the same fashion, one reading can only be matched to one prediction.

Predictions that cannot be matched to a reading are lost, as happens to the 19.9, 20.1 and 20.4 predictions. It's the same for readings that have no prediction. That is why there is only one entry in the output table for sensor 3.

Implement an algorithm that given both tables, returns the expected output table. You can choose if the tables are nested dictionaries, lists or a Pandas data frame.

Why did you choose that particular data structure?

In the example, very few readings and predictions are used, but the algorithm should perform well for very large numbers of them.

What is the computational complexity of the algorithm?

Please also provide the following tests:

- A test that shows that the algorithm works with the provided example data in the tables above
- A test that shows that the algorithm performs well for large numbers (500 sensors * 3000 reading rows * 15000 prediction rows)
    - For this you will have to create the data programmatically
    - Please also include this data creation code

Feel free to also include any additional tests that you deem necessary or that you have used for your development.