Python | Imputation using KNNimputer()

KNNimputer is a scikit-learn class used to fill out or predict the missing values in a dataset. It is a more useful method which works on the basic approach of the KNN algorithm rather than the naive approach of filling all the values with mean or the median. In this approach, we specify a distance from the missing values which is also known as the K parameter. The missing value will be predicted in reference to the mean of the neighbours.

It is implemented by the **KNNimputer()** method which contains the following arguments:

n_neighbors: *number of data points to include closer to the missing value.*

metric: *the distance metric to be used for searching.*
values – {nan_euclidean. callable} by default – nan_euclidean

weights: *to determine on what basis should the neighboring values be treated*
values -{uniform , distance, callable} by default- uniform.

Maths behind KNN Imputation Technique

- KNN or K-Nearest Neighbour is one of the simplest and easiest technique of imputation in Machine Learning.
- KNN works on Euclidean distance between the neighbour coordinates.
- KNN can used for both Classification and Regression problems.
- KNN is often used as benchmark for more complex classifiers such Artificial Neural Network (ANN) and Support Vector Machines (SVM).,

Steps to be followed

1. Choose the first column with missing value to fill in the data.
2. Select the values in a row
3. Choose the number of neighbours you want to work with (ideally 2-5)
4. Calculate Euclidean distance from all other data points corresponding to each other in the row.
5. Select the smallest 2 if number of neighbours is 2 and average out.

In [4]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
df1 = pd.DataFrame()
df1['Age'] = [20, 40, 30]
df1['Salary'] = [3, 4.5, np.nan]
```

In [8]:
```
print(df1)
```

```
   Age  Salary
0   20     3.0
1   40     4.5
2   30     NaN
```

In [9]:
```
from sklearn.impute import KNNImputerputer
```

In [10]:
```
knnimp = KNNImputer(n_neighbors=2)
```

In [11]:
```
knnimp.fit_transform(df1)
```

Out[11]:
```
array([[20.  ,  3.  ],
       [40.  ,  4.5 ],
       [30.  ,  3.75]])
```

In [11]:
```
knnimp.fit_transform(df1)
```

Out[11]:
```
array([[20.  ,  3.  ],
       [40.  ,  4.5 ],
       [30.  ,  3.75]])
```

In [12]:
```
(3 + 4.5)/2
```

Out[12]:
```
3.75
```

In [14]:
```
df = pd.DataFrame()
df['Maths'] = [80,90,np.nan, 92]
df['Chemistry'] = [60,65,58, np.nan]
df['Physics'] = [np.nan,57,84,78]
df['Biology'] = [78, 89, 67, np.nan]
```

In [15]:
```
df
```

Out[15]:

	Maths	Chemistry	Physics	Biology
0	80.0	60.0	NaN	78.0
1	90.0	65.0	57.0	89.0
2	NaN	58.0	84.0	67.0
3	92.0	NaN	78.0	NaN

```
In [15]: df
```

Out[15]:

	Maths	Chemistry	Physics	Biology
0	80.0	60.0	NaN	78.0
1	90.0	65.0	57.0	89.0
2	NaN	58.0	84.0	67.0
3	92.0	NaN	78.0	NaN

```
In [25]: knnimp = KNNImputer(n_neighbors=3)
```

```
In [26]: knnimp.fit_transform(df)
```

```
Out[26]: array([[80.        , 60.        , 73.        , 78.        ],
                [90.        , 65.        , 57.        , 89.        ],
                [87.33333333, 58.        , 84.        , 67.        ],
                [92.        , 61.        , 78.        , 78.        ]])
```

```
In [27]: (60 + 65 + 58)/3
```

```
Out[27]: 61.0
```