5_Imputation

January 14, 2025

[ ]:
```
'''
    Imputation [Fill Missing Data] -->

    Imputation is the process of replacing missing data (NaN values) in a␣
 ↪dataset with substituted values.
    Missing data can occur due to various reasons such as errors during data␣
 ↪collection, sensor malfunctions,
    or incomplete data entries. Imputation helps maintain the dataset's␣
 ↪integrity, allowing machine learning
    algorithms to work effectively without being disrupted by missing values.
'''
```

[ ]:
```
'''
    Why Imputation is Important ?

    Incomplete Data: Many machine learning algorithms cannot handle missing␣
 ↪values
    and will raise errors or produce inaccurate results.
    Preserving Dataset Size: Dropping rows or columns with missing values can␣
 ↪significantly
    reduce the amount of data, which may negatively impact model performance.
    Bias Avoidance: Proper imputation can reduce bias that might result from␣
 ↪ignoring or dropping missing data.
'''
```

[1]:
```
#   To Impute data we will use Simple Imputer From sklearn
```

[1]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
```

[2]:
```
data = pd.read_csv('Data/Data.csv')
data
```

[2]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes

```
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
```

[7]:
```
#   Selecting Features -->

x_data = data.iloc[:, :-1].values
y_data = data.iloc[:, -1].values
```

[ ]:
```
#   There are many ways to fill the missing values -->

#       Strategy                                        Description        Usage␣
 ↪                                          Example

#         mean                Replaces missing values with the mean of the␣
 ↪column.                            SimpleImputer(strategy='mean')
#        median               Replaces missing values with the median of the␣
 ↪column.                            SimpleImputer(strategy='median')
#     most_frequent           Replaces missing values with the most frequent␣
 ↪(mode) value in the column.        SimpleImputer(strategy='most_frequent')
#       constant            Replaces missing values with a constant value (e.g.
 ↪, 0 or another value).        SimpleImputer(strategy='constant',␣
 ↪fill_value=0)
#   knn (KNN Imputer)       Replaces missing values using K-Nearest Neighbors␣
 ↪to find similar data.            KNNImputer(n_neighbors=5)
#      iterative            Replaces missing values by modeling each feature as␣
 ↪a function of others.        IterativeImputer()
```

[8]:
```
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')
imputer.fit(x_data[:, 1:3])
x_data[:, 1:3] = imputer.transform(x_data[:, 1:3])
print(x_data)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
 ['Spain' 38.77777777777778 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
```

```
['France' 37.0 67000.0]]
```

[ ]:
```
'''
    Now you can see that missing values are replaced by mean values
    as the strategy is set to mean and missing_values is defined for nan values
    and is fitted on 1st to 3rd coloumn as there aren't any missing values in␣
 ↪4th coloumn
    We can also use dataset in place of x[:, 1:3] to apply operation on whole␣
 ↪data
    transform applies this operation and return modified values
'''
```