#EngineeringPlus Online Course Series

ADAMAS
SCHOOL OF ENGINEERING AND TECHNOLOGY

AI & ML

Module: 03
Lecture:

Course Instructor

Roneeta Purkayastha
Assistant Professor
Computer Science and Engineering
roneeta.purkayastha@adamasuniversity.ac.in

Index

Introduction to Data Preprocessing

- Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model.
- It is the first and crucial step while creating a machine learning model.

Why Data Preprocessing?

ADAMAS SCHOOL OF ENGINEERING AND TECHNOLOGY

Why do we need Data Preprocessing?

Why Data Preprocessing? contd.

- Real world data is not directly usable in ML models due to:
  - Noises,
  - Missing values,
  - Unusable format

Steps of Data Preprocessing

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Get the dataset

- The collected data for a particular problem in a proper format is known as the **dataset**.
- Most commonly, it is a Comma separated values (CSV) file.
- It may be a HTML or excel file according to requirements.

Importing Libraries

- There are three specific libraries that we will use for data preprocessing, which are:
  - numpy
  - matplotlib
  - pandas

Importing the Datasets

- **read_csv() function:**
  - In this step, we will use read_csv() function of pandas library, which is used to read a csv file and performs various operations on it.
  - Using this function, we can read a csv file locally as well as through an URL.
- Code snippet:

```
data_set= pd.read_csv('Dataset.csv')
```

Extracting independent variable:

- x= data_set.iloc[:,:-1].values
- Output:

```
[['India' 38.0 68000.0]
 ['France' 43.0 45000.0]
 ['Germany' 30.0 54000.0]
 ['France' 48.0 65000.0]
 ['Germany' 40.0 nan]
 ['India' 35.0 58000.0]
 ['Germany' nan 53000.0]
 ['France' 49.0 79000.0]
 ['India' 50.0 88000.0]
 ['France' 37.0 77000.0]]
```

Extracting dependent variable:

- y= data_set.iloc[:,3].values
- **Output:**

  array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
  dtype=object)

Handling Missing data

- **By deleting the particular row:**
  - This is not effective or advisable as it leads to data loss.
- **By calculating the mean:**
  - In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value.
  - This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Handling Missing data

- Code snippet

```
#handling missing data (Replacing missing data with the mean value)
from sklearn.preprocessing import Imputer
imputer= Imputer(missing_values ='NaN', strategy='mean', axis = 0)
#Fitting imputer object to the independent variables x.
imputerimputer= imputer.fit(x[:, 1:3])
#Replacing missing data with the calculated mean value
x[:, 1:3]= imputer.transform(x[:, 1:3])
```

Output

- array([['India', 38.0, 68000.0],
['France', 43.0, 45000.0],
 ['Germany', 30.0, 54000.0],
 ['France', 48.0, 65000.0],
['Germany', 40.0, 65222.22222222222],
   ['India', 35.0, 58000.0],
 ['Germany', 41.111111111111114, 53000.0],
 ['France', 49.0, 79000.0],
 ['India', 50.0, 88000.0],
['France', 37.0, 77000.0]], dtype=object

Encoding Categorical data

- Categorical data is data which has some categories such as, in our dataset; there are two categorical variable, Country, and Purchased.

- Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers

Encoding Categorical data: contd.

- For Country variable
- We will use LabelEncoder() class from preprocessing library.

```
#Catgorical data
#for Country Variable
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

Output:

```
array([[2, 38.0, 68000.0],
          [0, 43.0, 45000.0],
       [1, 30.0, 54000.0],
       [0, 48.0, 65000.0],
       [1, 40.0, 65222.22222222222],.....
```

Encoding Categorical data: contd.

- Dummy encoding:
  - But in our case, there are three country variables, and as we can see in the above output, these variables are encoded into 0, 1, and 2. By these values, the machine learning model may assume that there is some correlation between these variables which will produce the wrong output. So to remove this issue, we will use **dummy encoding**.
- **Dummy Variables:**
  - Dummy variables are those variables which have values 0 or 1. The 1 value gives the presence of that variable in a particular column, and rest variables become 0. With dummy encoding, we will have a number of columns equal to the number of categories.
  - In our dataset, we have 3 categories so it will produce three columns having 0 and 1 values. For Dummy Encoding, we will use **OneHotEncoder** class of **preprocessing** library.

Encoding Categorical data: contd.

```
#for Country Variable
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
#Encoding for dummy variables
onehot_encoder= OneHotEncoder(categorical_features= [0])
x= onehot_encoder.fit_transform(x).toarray()
```

Output:

```
array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00,
   3.80000000e+01,       6.80000000e+04],       [1.00000000e+00,
   0.00000000e+00, 0.00000000e+00, 4.30000000e+01,
   4.50000000e+04],       [0.00000000e+00, 1.00000000e+00,
   0.00000000e+00, 3.00000000e+01,       5.40000000e+04],.....)
```

Encoding Categorical data: contd.

- **For Purchased Variable:**

```
labelencoder_y= LabelEncoder()
y= labelencoder_y.fit_transform(y)
```

Output:

array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1])

Splitting the Dataset into the Training set and Test set

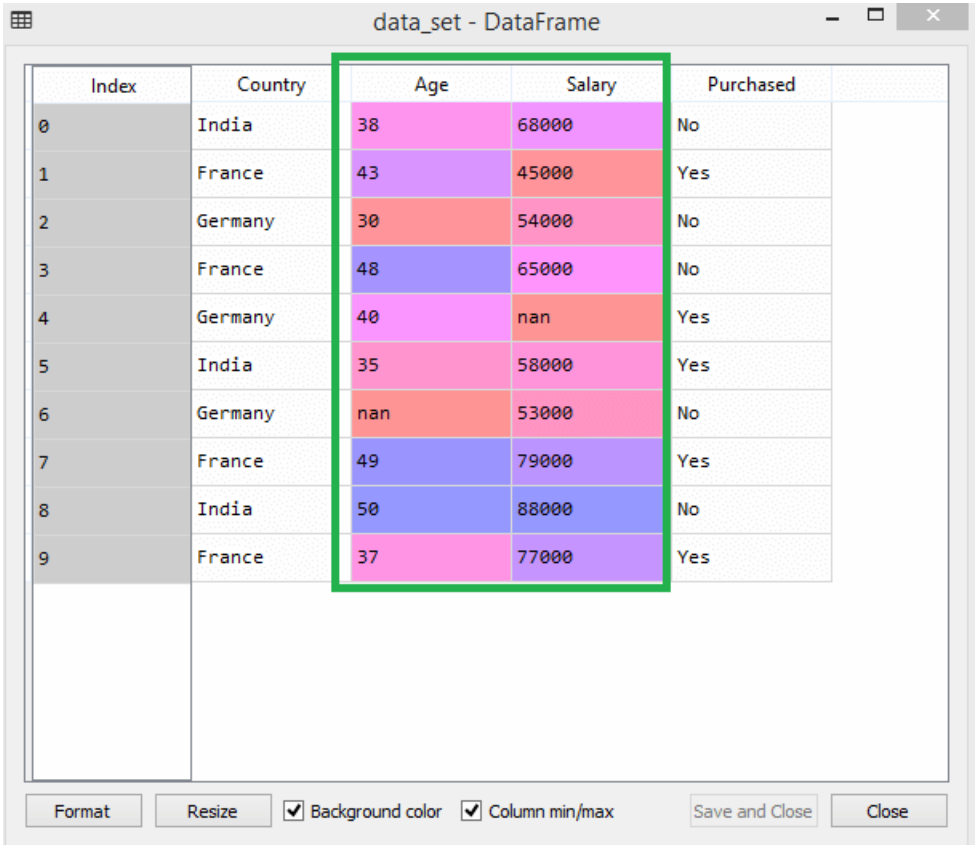
- **Training Set:** A subset of dataset to train the machine learning model, and we already know the output.
- **Test set:** A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_
  state=0)
```

Feature Scaling

- Feature scaling is the final step of data preprocessing in machine learning.
- It is a technique to standardize the independent variables of the dataset in a specific range.
- In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable.

Feature Scaling: contd.



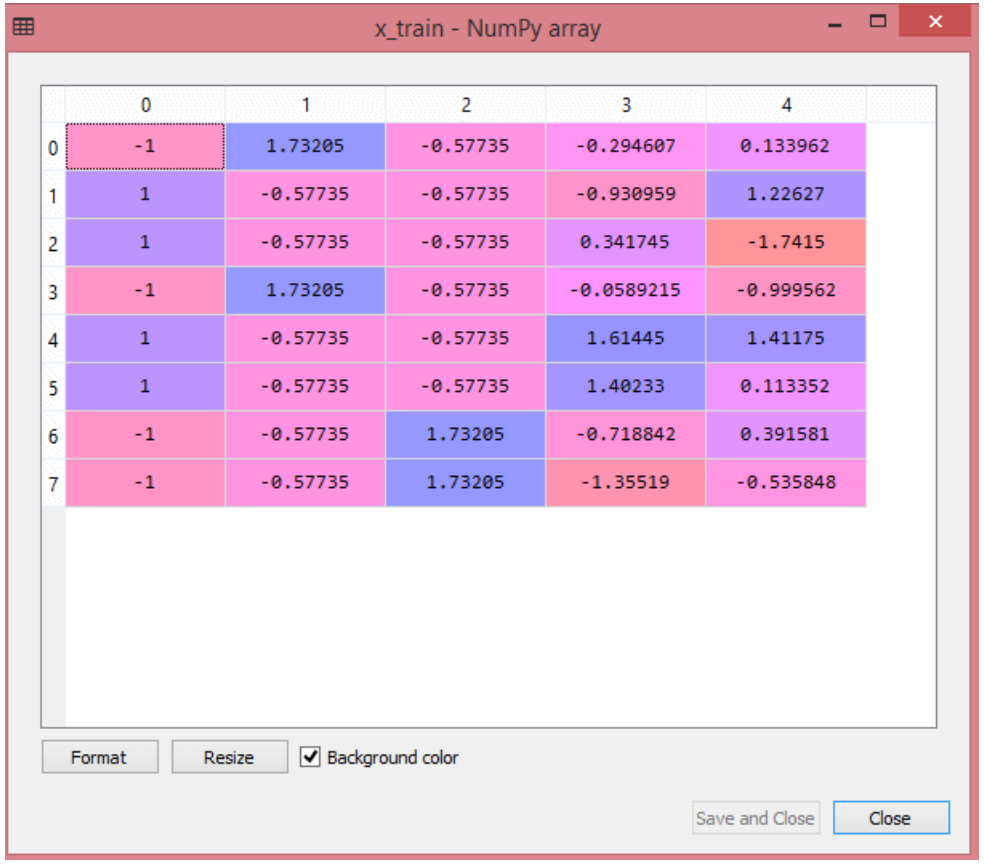
Index	Country	Age	Salary	Purchased
0	India	38	68000	No
1	France	43	45000	Yes
2	Germany	30	54000	No
3	France	48	65000	No
4	Germany	40	nan	Yes
5	India	35	58000	Yes
6	Germany	nan	53000	No
7	France	49	79000	Yes
8	India	50	88000	No
9	France	37	77000	Yes

Feature Scaling: contd.

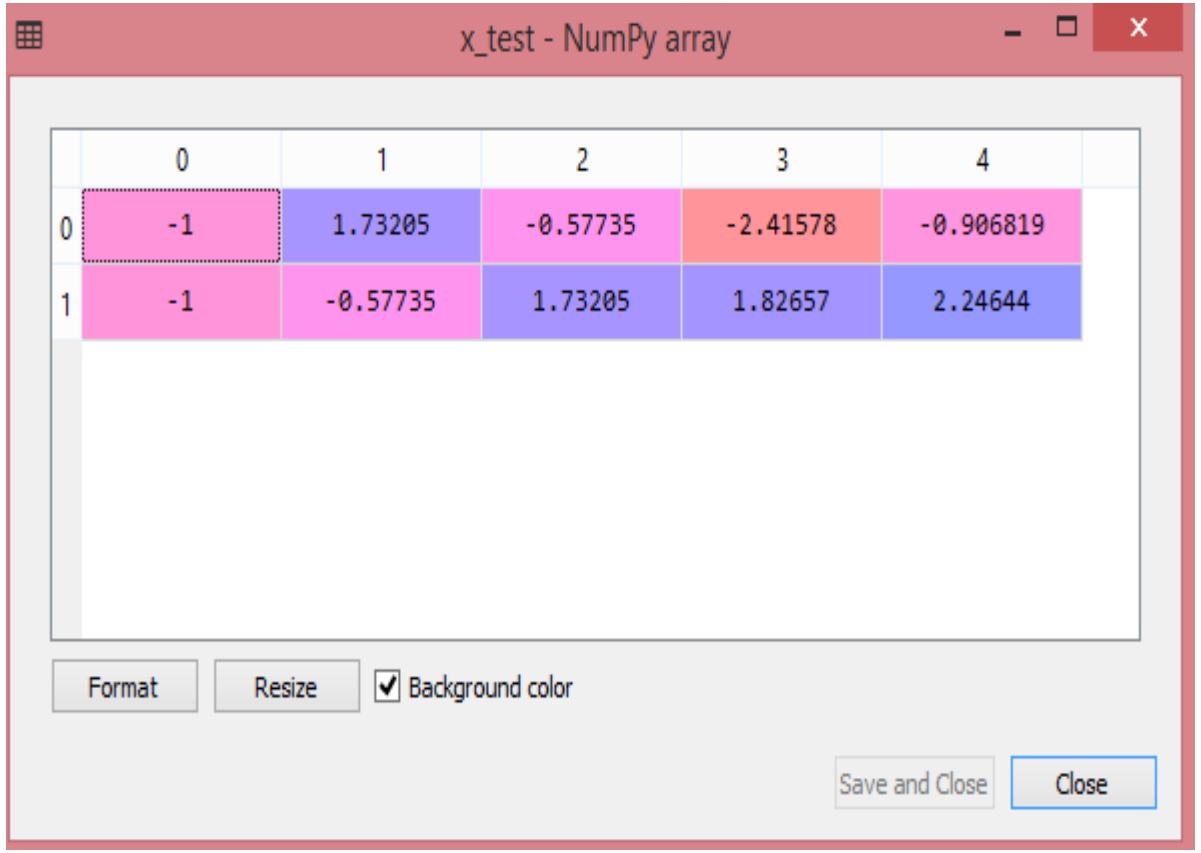
- There are two ways to perform feature scaling in machine learning:
  - **Standardization**
  - **Normalization**
- **Code:**

```
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

Output



	0	1	2	3	4
0	-1	1.73205	-0.57735	-0.294607	0.133962
1	1	-0.57735	-0.57735	-0.930959	1.22627
2	1	-0.57735	-0.57735	0.341745	-1.7415
3	-1	1.73205	-0.57735	-0.0589215	-0.999562
4	1	-0.57735	-0.57735	1.61445	1.41175
5	1	-0.57735	-0.57735	1.40233	0.113352
6	-1	-0.57735	1.73205	-0.718842	0.391581
7	-1	-0.57735	1.73205	-1.35519	-0.535848



	0	1	2	3	4
0	-1	1.73205	-0.57735	-2.41578	-0.906819
1	-1	-0.57735	1.73205	1.82657	2.24644

References

- Online resources
- Machine Learning tutorial, https://www.javatpoint.com/machine-learning