**WEEK-2**

**AIM: Apply Data Pre-processing Techniques.**

**DESCRIPTION:**

Data pre-processing 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 do we need Data Pre-processing?

A real-world data generally c**ontains noises, missing values, and maybe in an unusable format** which cannot be directly used for machine learning models. **Data pre-processing is required tasks for cleaning the data** and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

**It involves below steps:**

1. Getting the Dataset
2. Importing libraries
3. Importing datasets
4. Finding Missing Data
5. Encoding Categorical Data
6. Splitting dataset into training and test set
7. Feature scaling

**1. Getting the Dataset:** The first thing we required is a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the **dataset**. To use the dataset in our code, we usually put it into a **CSV** **file**. However, sometimes, we may also need to use an HTML or xlsx file.

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs.

**2. Importing libraries:** In order to perform data pre-processing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are **three specific libraries that we will use for data pre-processing, which are:**

**(a) Numpy:** Numpy Python library is used for including any type **of mathematical operation in the code**. It is the fundamental package for scientific calculation in Python. It also supports to **add large, multidimensional arrays and matrices**. **So, in Python, we can import it as:**

**import numpy as nm**

Here we have used **nm**, which is a short name for Numpy, and it will be used in the whole program.

**(b) Matplotlib:** The second library is matplotlib, which is a **Python 2D plotting library**, and with this library, we need to import a sub-library pyplot. This library is **used to plot any type of charts in Python** for the code. **It will be imported as below:**

**import matplotlib.pyplot as mpt**

Here we have used **mpt a**s a short name for this library.

**(c) Pandas:** The last library is the Pandas library, which is one of the most famous Python libraries and used for **importing and managing the datasets**. It is an **open-source data manipulation and analysis library**. It will be imported as below:

**import pandas as pd**

Here, we have used **pd** as a short name for this library.

**3. Importing datasets:**

**read\_csv() function:** Now to import the dataset, 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. **We can use read\_csv function as below:**

**For eg :** data\_set= pd.read\_csv('Student.csv')

Here, **data\_set** is a name of the variable to store our dataset, and inside the function, we have passed the name of our dataset.

**Extracting Independent and Dependent Variables:**

In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

**For Eg:** In our dataset, there are three independent variables that are **Country, Age** and **Salary**, and one is a dependent variable which is **purchased**.

**Extracting Independent Variables:** To extract an independent variable, we will use

**iloc [ ]**method of **Pandas library**. It is used to **extract the required rows and columns** from the dataset.

**For Eg: x= data\_set. iloc [:,:-1].values**

In the above code, the **first colon (:) is used to take all the rows**, and the **second colon (:) is for all the columns.** Here we have used **(:-1), because we don't want to take the last column** as it contains the **dependent variable**. So by doing this, we will get the matrix of features.

**Extracting dependent Variables:** To extract dependent variables, again, we will use **Pandas .iloc[]** method.

**For Eg: y= data\_set. iloc [:, 3].values**

Here we have taken **all the rows with the last column only**. It will give the array of dependent variables.

**4. Finding Missing Data:**

**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.

To handle missing values, we will use **Scikit-learn** library in our code, which contains **various libraries** for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library.

**For Eg:**

**#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])

**5. Encoding Categorical Data:** 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.

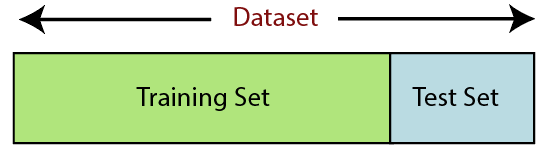
**For Country variable:** Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **pre-processing** library.

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

**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.

**For Eg :** 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 **pre-processing** library.

**6. Splitting dataset into training and test set:** we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.



**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.

For **splitting the dataset, we will use the below lines of code:**

from sklearn.model\_selection import train\_test\_split

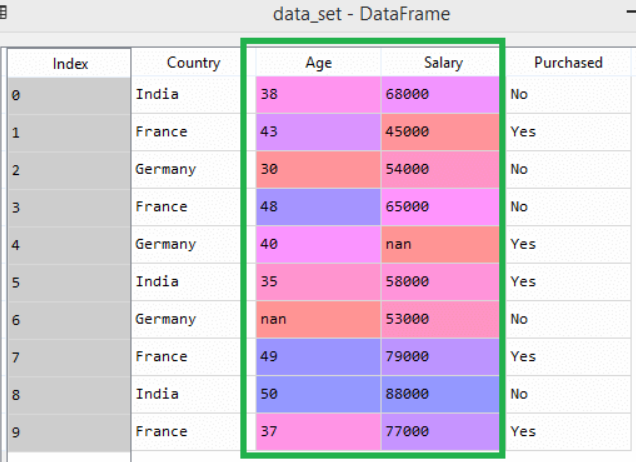
x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

**Explanation:**

* In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
* In the second line, we have used four variables for our output that are
  + x\_train: features for the training data
  + x\_test: features for testing data
  + y\_train: Dependent variables for training data
  + y\_test: dependent variable for testing data

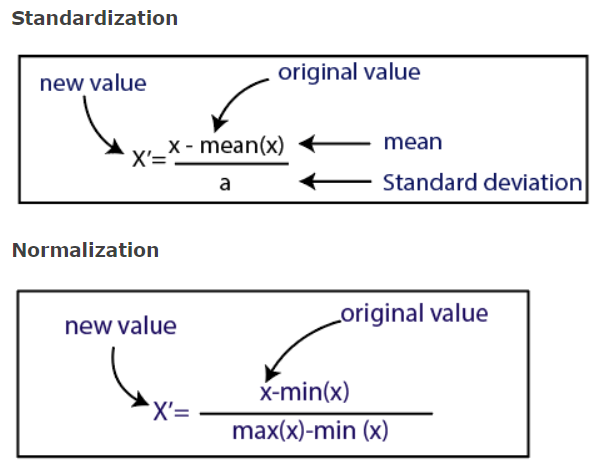
**7. Feature Scaling:** Feature scaling is the final step of data pre-processing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put in the same range and in the same scale so that no any variable dominate the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

**For Eg:**



As we can see, the **age and salary column values are not on the same scale**. A machine learning model is based on **Euclidean distance**, and if we do not scale the variable, then it will cause some issue in our machine learning model.

If we compute **any two values from age and salary**, then **salary values will dominate the age values,** and it will produce an **incorrect result**. So to remove this issue, we need to perform feature scaling for machine learning. **There are two ways to perform feature scaling in machine learning:**

****

Here, we will use the **standardization method for our dataset**.

For feature scaling, we will import ***StandardScaler* class** of ***sklearn.preprocessing*** library as:

**from sklearn.preprocessing import StandardScaler**

Now, we will create the object of **StandardScaler** **class** for **independent variables or** features. And then we will fit and **transform the training dataset.**

**st\_x= StandardScaler ()**

**x\_train= st\_x.fit\_transform(x\_train)**

For test dataset, we will directly apply **transform()** function instead of **fit\_transform()** . because it is already done in training set.

**x\_test= st\_x.transform (x\_test)**

**PROGRAM:**

## #importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

## # importing the dataset

dataset = pd.read\_csv('D:\AI TOOLS(PVP-19)\AI TOOLS LAB\LAB-1\Data.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 3].values

print(X)

print(y)

## #Taking care of missing data

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan, strategy='mean')

imputer.fit(X[:,1:3])

X[:, 1:3] = imputer.transform(X[:, 1:3])

print(X)

## # Encoding categorical data

### #Encoding the Independent Variable

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])],

remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

### #Encoding the Dependent Variable

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y = le.fit\_transform(y)

print(y)

## #splitting the dataset into the Training set and Test set

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 = 1)

print(X\_train)

print(X\_test)

print(y\_train)

print(y\_test)

## #Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

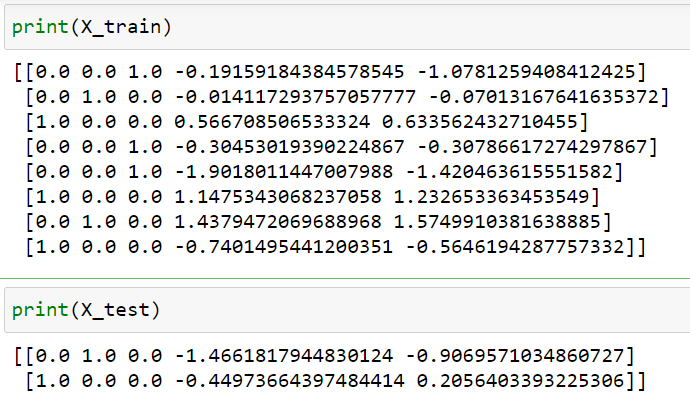
X\_train[:, 3:] = sc.fit\_transform(X\_train[:, 3:])

X\_test[:, 3:] = sc.transform(X\_test[:, 3:])

print(X\_train)

print(X\_test)

**INPUT/OUTPUT:**

****

**CONCLUSION: Program is executed successfully without any error.**