- **Data Preprocessing in Machine learning**

Dr. Jagendra Singh

Machine Learning

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.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data.

And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

WHY DO WE NEED DATA PREPROCESSING?

- A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models.

- Data preprocessing 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 following steps:

  o **Getting the dataset**

  o **Importing libraries**

  o **Importing datasets**

  o **Finding Missing Data**

  o **Encoding Categorical Data**

  o **Splitting dataset into training and test set**

  o **Feature scaling**

1. GET THE DATASET

To create a machine learning model, 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.

Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient.

So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV file.

WHAT IS A CSV 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.

Here we will use a demo dataset for data preprocessing, and for practice, it can be downloaded from here, "https://www.superdatascience.com/pages/machine-learning

For real-world problems, we can download datasets online from various sources such as https://www.kaggle.com/uciml/datasets

- , https://archive.ics.uci.edu/ml/index.php

We can also create our dataset by gathering data using various API with Python and put that data into a .csv file.

2. IMPORTING LIBRARIES

There are three specific libraries that we will use for data preprocessing, which are:

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

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

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.

3. IMPORTING THE DATASETS

Now we need to import the datasets which we have collected for our machine learning project.

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.

IMPORTING THE DATASETS

We can use read_csv function as below:

data_set= pd.read_csv('Dataset.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.

- Once we execute the above line of code, it will successfully import the dataset in our code. We can also check the imported dataset by clicking on the section **variable explorer**, and then double click on **data_set**. Consider the below image:

IMPORTING THE DATASETS

4. HANDLING MISSING DATA

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

- There are mainly two ways to handle missing data, which are:

- By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

- 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. Here, we will use this approach.

HANDLING MISSING DATA

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. Below is the code for it:

```
#handling missing data (Replacing missing data with the mean value)

#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])
```

HANDLING MISSING DATA

- As we can see in the below output, the missing values have been replaced with the means of rest column values.

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
```

5. 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

- **For Country variable:**

- Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **preprocessing** library.

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

Output:

```
Out[15]:
  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],
        [2, 35.0, 58000.0],
        [1, 41.111111111111114, 53000.0],
        [0, 49.0, 79000.0],
        [2, 50.0, 88000.0],
       [0, 37.0, 77000.0]], dtype=object)
```

ENCODING CATEGORICAL DATA

Explanation:

- In above code, we have imported LabelEncoder class of sklearn library. This class has successfully encoded the variables into digits.

- 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.

ENCODING CATEGORICAL DATA

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.

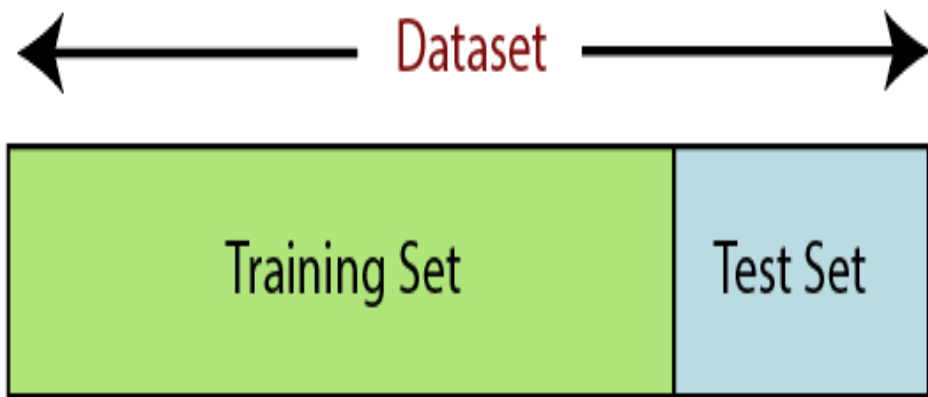# 6. SPLITTING THE DATASET INTO THE TRAINING SET AND TEST SET

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.

Suppose, if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.
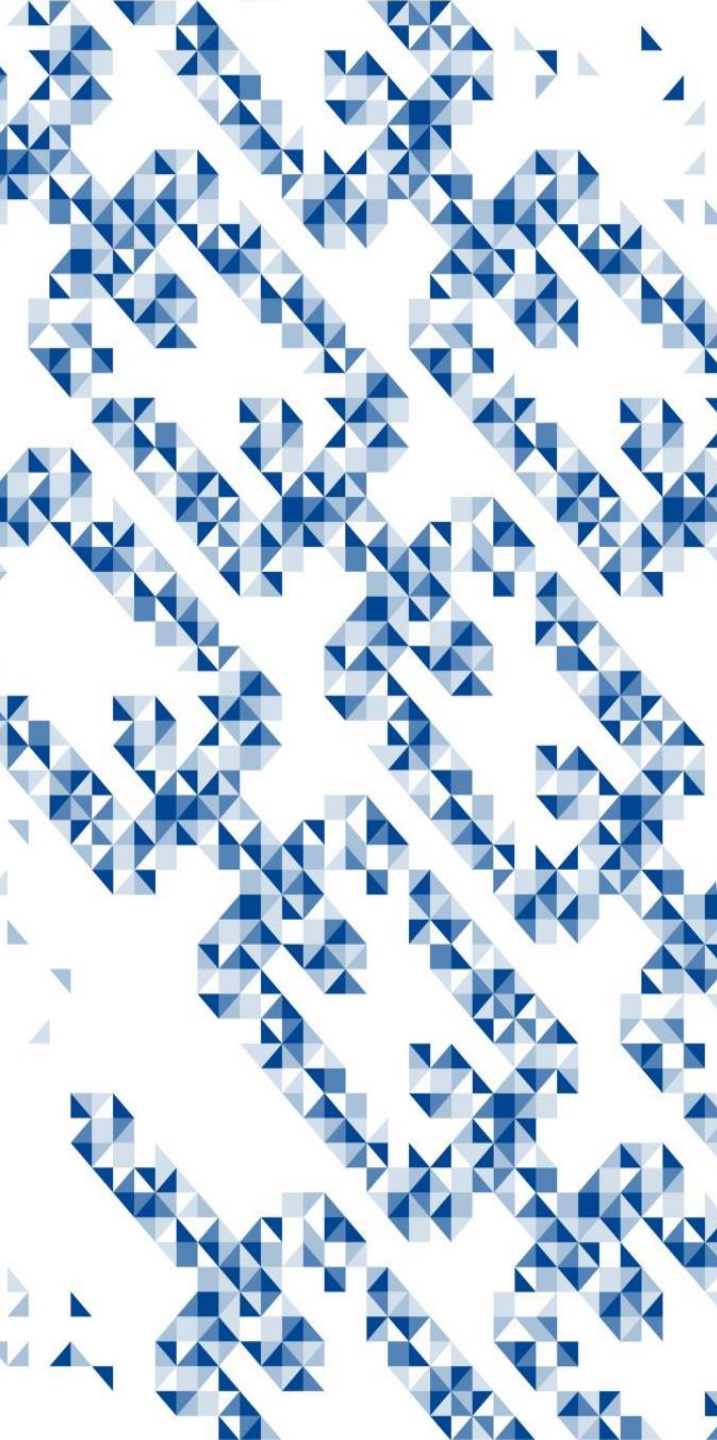
If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance.

So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

TRAINING SET AND TEST SET

Dataset

Training Set | 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.

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

1. from sklearn.model_selection import train_test_split

2. x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)

TRAINING SET AND TEST SET

- **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:** Independent variable for testing data

○ In **train_test_split() function**, we have passed four parameters in which first two are for arrays of data, and **test_size** is for specifying the size of the test set.

○ The test_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.

○ The last parameter **random_state** is used to set a seed for a random generator so that you always get the same result.

7. 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.

Consider the below dataset:

FEATURE SCALING

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

data_set - DataFrame

Format Resize ☑ Background color ☑ Column min/max Save and Close Close

FEATURE SCALING
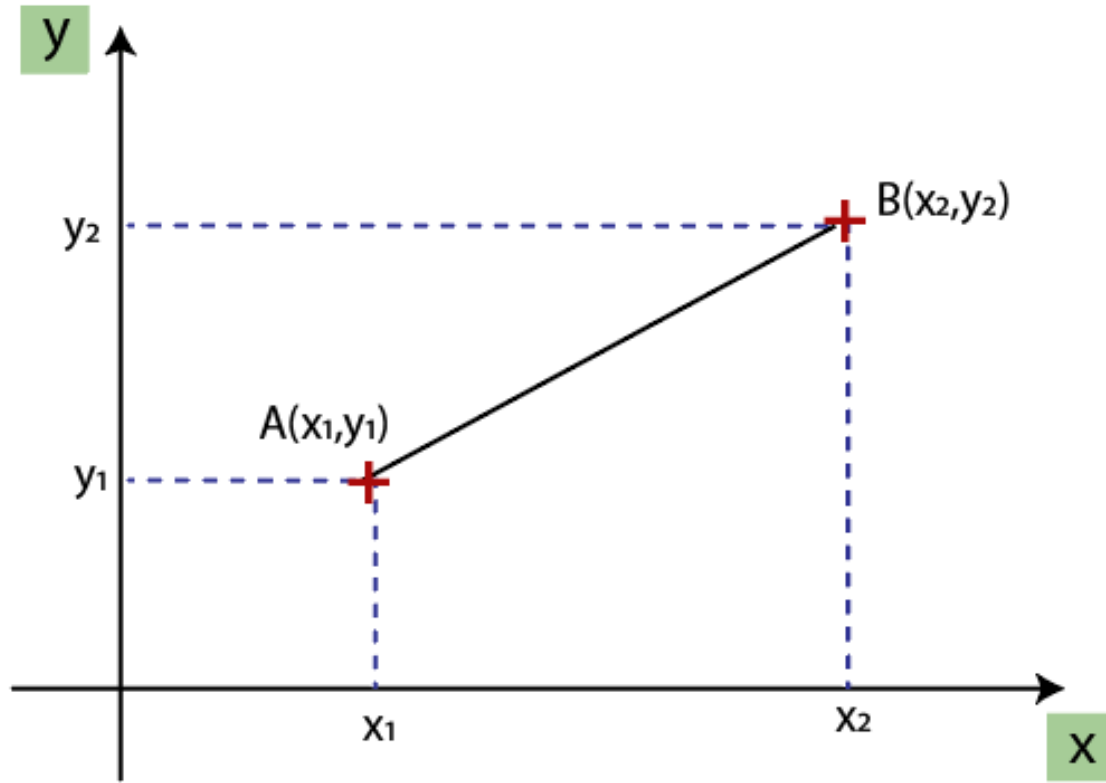
Euclidean Distance Between A and B=$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

- 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.

- Euclidean distance is given as:

- 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.

FEATURE SCALING

There are two ways to perform feature scaling in machine learning:

Standardization

Normalization

$$X' = \frac{x - mean(x)}{a}$$

new value → X'
original value → x - mean(x)
mean → mean
Standard deviation → a

$$X' = \frac{x - min(x)}{max(x) - min(x)}$$

new value → X'
original value → x - min(x)

THANK YOU