Mayur Badole — Published On April 30, 2021 and Last Modified On April 26th, 2023

Beginner Libraries Programming Python

> "Consumer data will be the biggest differentiator in the next two to three years. Whoever unlocks the reams of data and uses it strategically will win"

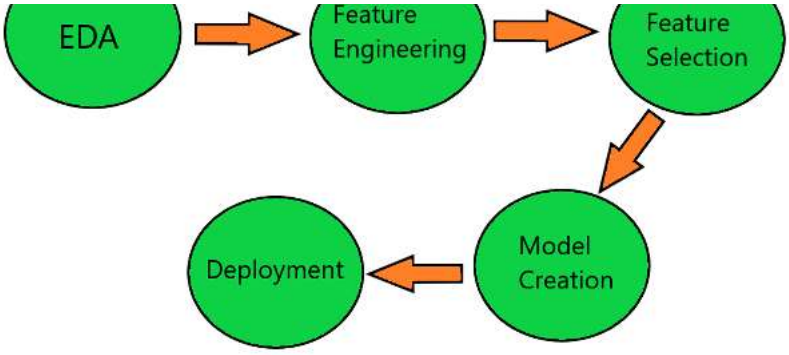
Table of Contents

Introduction

Before we start exploring the fit, transform, and fit_transform functions in Python, let's consider the life cycle of any data science project. This will give us a better idea of the steps involved in developing any data science project and the importance and usage of these functions. Let's discuss these steps in points:

1. **Exploratory Data Analysis (EDA)** is used to analyze the datasets using pandas, numpy, matplotlib, etc., and dealing with missing values. By doing EDA, we summarize their main importance.
2. **Feature Engineering** is the process of extracting features from raw data with some domain knowledge.
3. **Feature Selection** is where we select those features from the dataframe that will give a high impact on the estimator.
4. **Model creation** in this, we create a machine learning model using suitable algorithms, e.g., regressor or classifier.
5. **Deployment** where we deploy our ML model on the web.

Difference Between fit(), transform(), and fit_transform() Methods in Scikit-Learn



If we consider the first 3 steps, then it will probably be more towards Data Preprocessing, and Model Creation is more towards Model Training. So these are the two most important steps whenever we want to deploy any machine learning application.

Learning Objectives

- We will learn the main difference between functions in python's library sklearn, like fit(), transform(), and fit_transform().
- Recognize scenarios in which it may be necessary or beneficial to separate the fit() and transform() steps, such as when applying the same preprocessing to multiple datasets.
- This tutorial will also compare and contrast the behavior of fit(), transform(), and fit_transform() across different scikit-learn classes

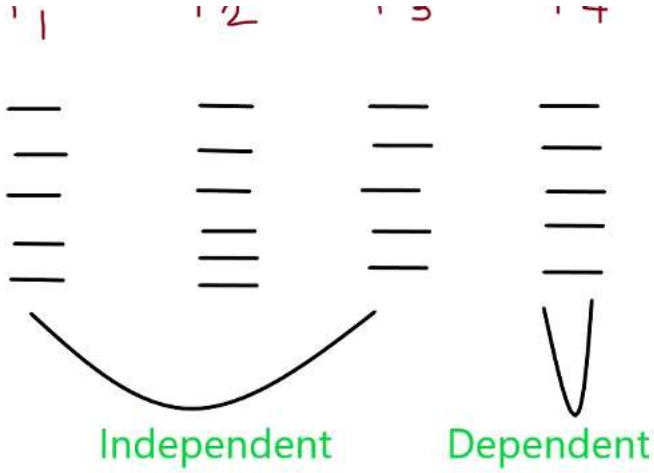
This article was published as a part of the Data Science Blogathon.

Transformer In Sklearn

Scikit-learn has an object, usually, something called a **Transformer.** The use of a transformer is that it will be performing data preprocessing and feature transformation, but in the case of model training, we have learning algorithms like linear regression, logistic regression, knn, etc., if we talk about the examples of Transformer-like **StandardScaler,** which helps us to do feature transformation where it converts the feature with mean =**0** and standard deviation =**1**, **PCA**, **Imputer**, **MinMaxScaler,** etc. then all these particular techniques have seen that we are doing some preprocessing on the input data will change the format of training dataset, and that data will be used for model training.

Suppose we take **f1, f2, f3,** and **f4** features where f1, f2, and f3 are independent features, and f4 is our dependent feature. We apply a standardization process in which it takes a feature **F** and converts it into **F'** by applying a formula of standardization. If you notice, at this stage, we take one input feature F and convert it into another input feature F' itself So, in this condition, we do three different operations:

1. **fit()**
2. **transform()**
3. **fit_transform()**



Now, we will discuss how the following operations are different from each other.

Difference Between fit and fit_transform

fit()

In the **fit()** method, where we use the required formula and perform the calculation on the feature values of input data and fit this calculation to the transformer. For applying the fit() method (fit transform in python), we have to use **fit()** in frontof the transformer object.

Suppose we initialize the StandardScaler object **O** and we do **.fit().** It takes the feature **F** and computes the **mean (μ)** and **standard deviation (σ)** of feature **F.** That is what happens in the fit method.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# split training and testing data
xtrain,xtest,ytrain,ytest= train_test_split(
                                            x,y,
                                            test_size=0.3,
                                            random_state=42
                                           )
# creating object
stand= StandardScaler()
# fit data
Fit= stand.fit(xtrain)
```

First, we have to split the dataset into training and testing subsets, and after that, we apply a transformer to that data.

In the next step, we basically perform a transform because it is the second operation on the transformer.

calculations.

We use the example that is used above section when we create an object of the fit method. We then put it in front of the .transform, and the transform method uses those calculations to transform the scale of the data points, and the output will we get is always in the form of a sparse matrix or array.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# split training and testing data
xtrain,xtest,ytrain,ytest= train_test_split(
                                            x,y,
                                            test_size=0.3,
                                            random_state=42
                                           )
# creating object
stand= StandardScaler()
# fit data
Fit= stand.fit(xtrain)
# transform data
x_scaled = Fit.transform(xtrain)
```

```
array([[-0.82955914, -0.84844726,  0.76004081, ...,  0.42452065,
         1.71511025, -0.28720569],
       [ 0.38933126, -0.84844726, -0.82003838, ...,  1.15770723,
         0.13623772, -0.64327699],
       [ 0.3313334 ,  1.64017602, -0.82003838, ..., -1.57425478,
         1.38751668, -0.71110009],
       ...,
       [-0.87071891, -0.84844726,  1.14837868, ...,  1.36496845,
         0.34050194,  0.93361021],
       [ 1.79250517,  0.54283955, -0.82003838, ..., -1.57425478,
         0.11696751, -0.28720569],
       [ 0.28643184, -0.84844726,  0.95656807, ..., -0.62992083,
         0.13623772, -0.28720569]])
```

As you can see that the output of the transform is in the form of an array in which data points vary from 0 to 1.

Note: It will only perform when we want to do some kind of transformation on the input data.

fit_transform() or fit transform sklearn

The fit_transform() method is basically the combination of the fit method and the transform method. This method simultaneously performs fit and transform operations on the input data and converts the data points.Using fit and transform separately when we need them both decreases the efficiency of the model. Instead, fit_transform() is used to get both works done.

Suppose we create the StandarScaler object, and then we perform **.fit_transform().** It will calculate the mean(**μ**)and standard deviation(**σ**) of the feature **F** at a time it will transform the data points of the feature F.

```
# split training and testing data
xtrain,xtest,ytrain,ytest= train_test_split(
                                            x,y,
                                            test_size=0.3,
                                            random_state=42
                                           )
stand= StandardScaler()
Fit_Transform = stand.fit_transform(xtrain)
Fit_Transform
```

```
array([[-0.82955914, -0.84844726,  0.76004081, ...,  0.42452065,
         1.71511025, -0.28720569],
       [ 0.38933126, -0.84844726, -0.82003838, ...,  1.15770723,
         0.13623772, -0.64327699],
       [ 0.3313334 ,  1.64017602, -0.82003838, ..., -1.57425478,
         1.38751668, -0.71110009],
       ...,
       [-0.87071891, -0.84844726,  1.14837868, ...,  1.36496845,
         0.34050194,  0.93361021],
       [ 1.79250517,  0.54283955, -0.82003838, ..., -1.57425478,
         0.11696751, -0.28720569],
       [ 0.28643184, -0.84844726,  0.95656807, ..., -0.62992083,
         0.13623772, -0.28720569]])
```

This method output is the same as the output we obtain after applying the separate fit() and transform() methods.

Conclusion

In conclusion, the scikit-learn library provides us with three important methods, namely fit(), transform(), and fit_transform(), that are used widely in machine learning. The fit() method helps in fitting the data into a model, transform() method helps in transforming the data into a form that is more suitable for the model. Fit_transform() method, on the other hand, combines the functionalities of both fit() and transform() methods in one step. Understanding the differences between these methods is very important to perform effective data preprocessing and feature engineering.

Key Takeaways

- The fit() method helps in fitting the training dataset into an estimator (ML algorithms).
- The transform() helps in transforming the data into a more suitable form for the model.
- The fit_transform() method combines the functionalities of both fit() and transform().

Frequently Asked Questions

Q1. Can we use transform() without using fit() in scikit-learn?

A. Yes, transform() method can be used without using fit() method in scikit-learn. This is useful when we want to transform new data using the same scaling or encoding applied to the training data.

Q2. What is the purpose of fit_transform() in scikit-learn?

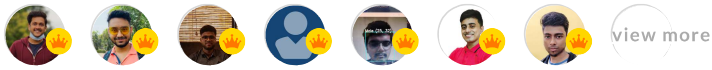
A. The fit_transform() method is used to fit the data into a model and transform it into a form that is more suitable for the model in a single step. This saves us the time and effort of calling both fit() and transform() separately.

Q3. Are there any limitations to using fit(), transform(), and fit_transform() methods in scikit-learn?

A. The main limitation of these methods is that they may not work well with certain types of data, such as data with null values or outliers, and we might need to perform additional preprocessing steps.

Difference Between fit(), transform(), and fit_transform() Methods in Scikit-Learn

Our Top Authors



Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Make Your Tableau Visuals More Effective – Tips And Tricks

Next Post

How to Download Kaggle Datasets using Jupyter Notebook

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

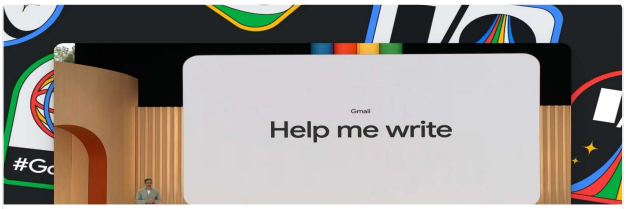
Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

Submit

Top Resources



Google Announced “Help Me Write” Feature in Gmail – How..

Aayush Tyagi - MAY 15, 2023

AI Pilots May Soon Fly Passenger Planes, Says Emirates Airline..

K.sabreena - MAY 15, 2023

Chatgpt-4 v/s Google Bard: A Head-to-Head Comparison

Gyan Prakash Tripathi - MAY 11, 2023

One-Stop Framework Building Applications with LLMs

Ajay Kumar Reddy - MAY 14, 2023