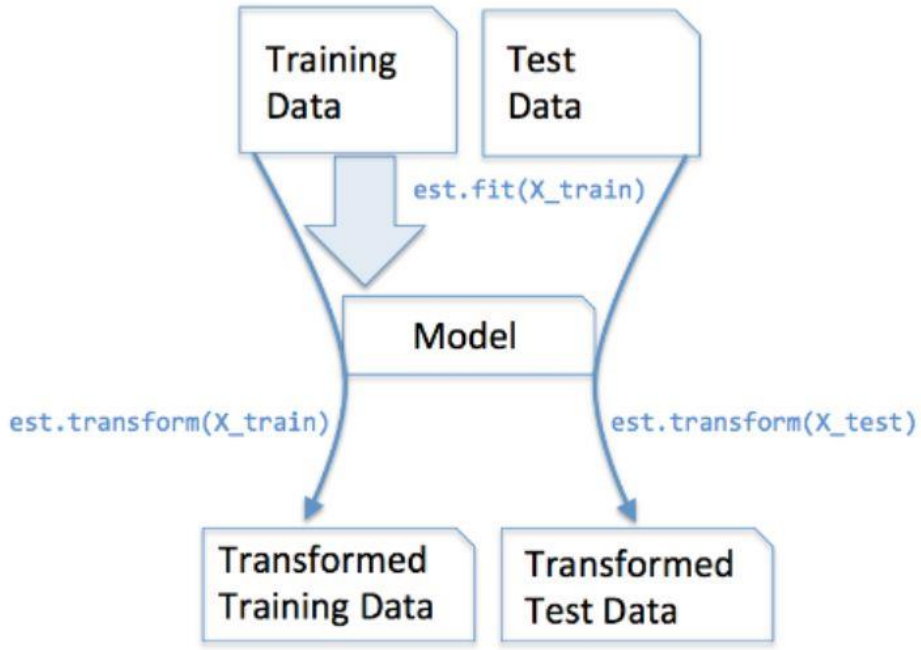
In **scikit-learn estimator api**,

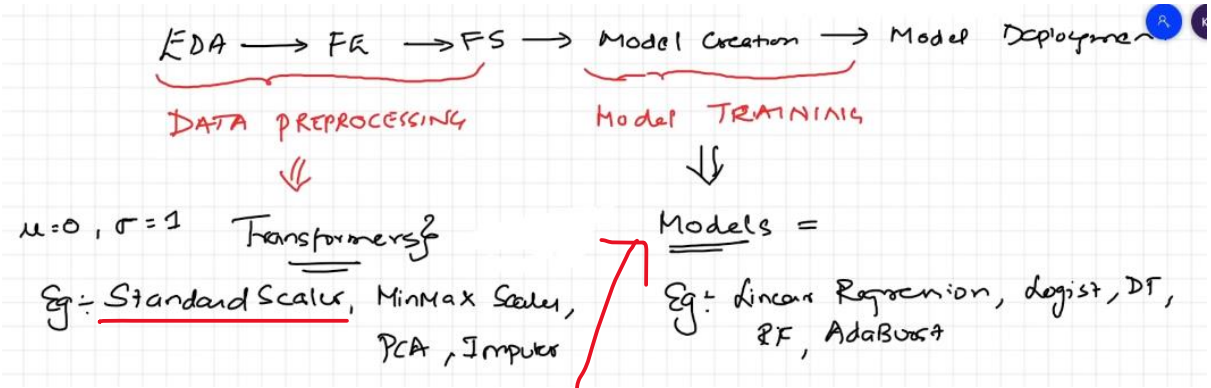
`fit()` : used for generating learning model parameters from training data

`transform()` : parameters generated from `fit()` method,applied upon model to generate transformed data set.

`fit_transform()` : combination of `fit()` and `transform()` api on same data set



We will be doing the pre-processing on the input data to change into some-other format. And these format is used for model training.



Will perform transform only when we want to do some transformation on the input data.

output	feature - 1	feature - 2	feature - 3
	-	-	-
	-	-	-
	-	-	-

X → Z

Standard Scalar

i) fit() ⇒ Takes the feature 1 and computes μ and σ

ii) transform() ⇒ Will apply $z \Rightarrow \frac{x_i - \mu}{\sigma}$ for every observation in feature - 1.

to both the job

iii) fit_transform() ⇒ In feature - 1, μ and σ are calculated then $z = \frac{x_i - \mu}{\sigma}$ for every observation

(feature - 1)' [after applying Standard Scalar]

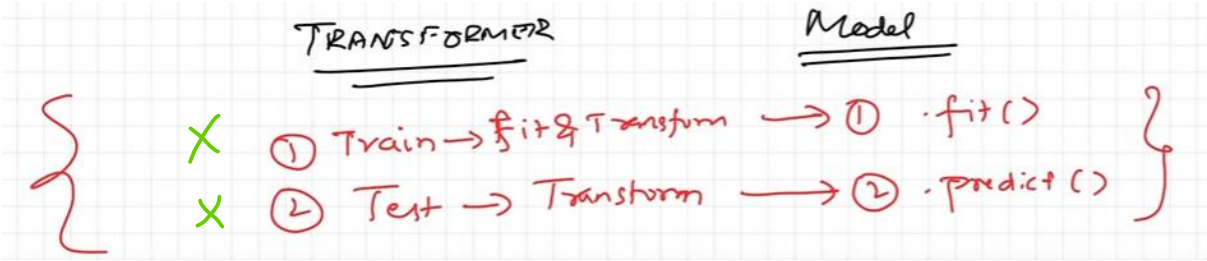
Similary we have,

output	feature 1	feature 2	feature 3	(feature 1)'	(feature 2)'	(feature 3)'

fit() → Model ← fit()

Model . fit (x_train_scaled , y_train)

Model . predict (x_test_scaled)




```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x,y,test_size=0.33,random_state=45)
```

```
## Applying Standard Scaler to the independent features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled  =  scaler.fit_transform(x_train)
x_test_scaled   =  scaler.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
classification = LogisticRegression()

classification.fit(x_train_scaled , y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for
example using ravel().
  return f(*args, **kwargs)
```

```
LogisticRegression()
```

```
classification.predict(x_test_scaled)
```

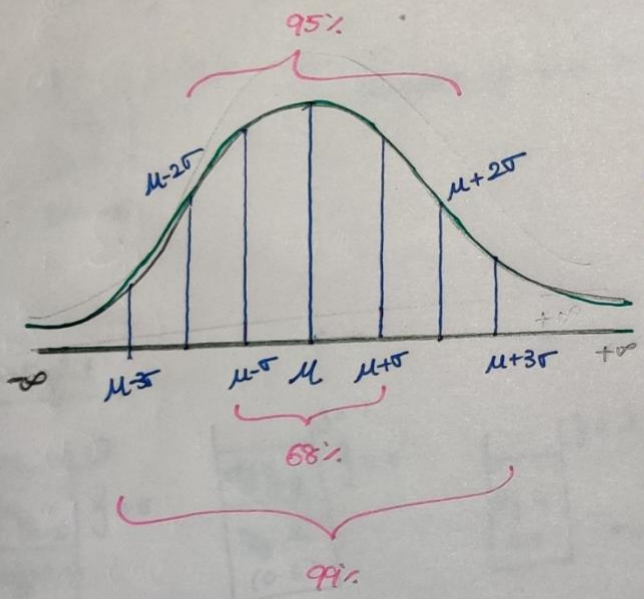
A random variable x follows Normal Distribution with mean μ and standard deviation σ

$$X \sim N(\mu, \sigma)$$

(or)

$$X \sim N(\mu, \sigma^2)$$

Variance = SD^2



Normal / Gaussian Distribution

Probability density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Min - Max Scalar

Standard Normal Distribution

$$\phi(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(z)^2}$$

where $\left(z = \frac{x-\mu}{\sigma}\right)$

changing from x-scale to z-scale.

Standard Scalar

fit_transform()

fit_transform() is used on the training data so that we can scale the training data and also learn the scaling parameters of that data. Here, the model built by us will learn the mean and variance of the features of the training set. These learned parameters are then used to scale our test data.

So what actually is happening here! 😉

The fit method is calculating the mean and variance of each of the features present in our data. The transform method is transforming all the features using the respective mean and variance.

Now, we want scaling to be applied to our test data too and at the same time do not want to be biased with our model. We want our test data to be a completely new and a surprise set for our model. The transform method helps us in this case.

transform()

Using the transform method we can use the same mean and variance as it is calculated from our training data to transform our test data. Thus, the parameters learned by our model using the training data will help us to transform our test data.

Now the question is why we did this? 🙃

Here is the simple logic behind it!

If we will use the fit method on our test data too, we will compute a new mean and variance that is a new scale for each feature and will let our model learn about our test data too. Thus, what we want to keep as a surprise is no longer unknown to our model and we will not get a good estimate of how our model is performing on the test (unseen) data which is the ultimate goal of building a model using machine learning algorithm.

This is the standard procedure to scale our data while building a machine learning model so that our model is not biased towards a particular feature of the dataset and at the same time prevents our model to learn the features/values/trends of our test data.

I hope this explanation will help you understand the simple logic behind these methods.

..\audio_recordings\fit()_transform()_fit_transform().mp4

References

https://www.youtube.com/watch?v=BotYLBQfd5M&t=527s

https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe