**1. What is the difference between fit and fit\_transform method of scikit learn**

**Ans :** To center the data (make it have zero mean and unit standard error), you

subtract the mean and then divide the result by the standard deviation.

x′=(x−μ)/σ

You do that on the training set of data. But then you have to apply the same

transformation to your testing set (e.g. in cross-validation), or to newly

obtained examples before forecast. But you have to use the same two parameters

μ and σ (values) that you used for centering the training set.

Hence, every sklearn's transform's fit() just calculates the parameters

(e.g. μ and σ in case of StandardScaler) and saves them as an internal objects state.

Afterwards, you can call its transform() method to apply the transformation to a particular set of examples.

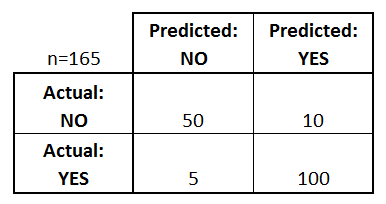
fit\_transform() joins these two steps and is used for the initial fitting of parameters on the training set x,

but it also returns a transformed x′. Internally, it just calls first fit() and then transform() on the same data.

**2.what is confusion matrix**

**Ans:** A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

Example: **confusion matrix for a binary classifier**



What can we learn from this matrix?

* There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
* The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
* Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
* In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates):

* **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
* **true negatives (TN):** We predicted no, and they don't have the disease.
* **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
* **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")



This is a list of rates that are often computed from a confusion matrix for a binary classifier:

* **Accuracy:** Overall, how often is the classifier correct?
  + (TP+TN)/total = (100+50)/165 = 0.91
* **Misclassification Rate:** Overall, how often is it wrong?
  + (FP+FN)/total = (10+5)/165 = 0.09
  + equivalent to 1 minus Accuracy
  + also known as "Error Rate"
* **True Positive Rate:** When it's actually yes, how often does it predict yes?
  + TP/actual yes = 100/105 = 0.95
  + also known as "Sensitivity" or "Recall"
* **False Positive Rate:** When it's actually no, how often does it predict yes?
  + FP/actual no = 10/60 = 0.17
* **Specificity:** When it's actually no, how often does it predict no?
  + TN/actual no = 50/60 = 0.83
  + equivalent to 1 minus False Positive Rate
* **Precision:** When it predicts yes, how often is it correct?
  + TP/predicted yes = 100/110 = 0.91
* **Prevalence:** How often does the yes condition actually occur in our sample?
  + actual yes/total = 105/165 = 0.64

A couple other terms are also worth mentioning:

* **Positive Predictive Value:** This is very similar to precision, except that it takes prevalence into account. In the case where the classes are perfectly balanced (meaning the prevalence is 50%), the positive predictive value (PPV) is equivalent to precision. (**More details about PPV.**
* **Null Error Rate:** This is how often you would be wrong if you always predicted the majority class. (In our example, the null error rate would be 60/165=0.36 because if you always predicted yes, you would only be wrong for the 60 "no" cases.) This can be a useful baseline metric to compare your classifier against. However, the best classifier for a particular application will sometimes have a higher error rate than the null error rate, as demonstrated by the [**Accuracy Paradox**](http://en.wikipedia.org/wiki/Accuracy_paradox).
* **Cohen's Kappa:** This is essentially a measure of how well the classifier performed as compared to how well it would have performed simply by chance. In other words, a model will have a high Kappa score if there is a big difference between the accuracy and the null error rate. ([**More details about Cohen's Kappa.**](http://en.wikipedia.org/wiki/Cohen's_kappa))
* **F Score:** This is a weighted average of the true positive rate (recall) and precision. ([**More details about the F Score.**](http://en.wikipedia.org/wiki/F1_score))
* **ROC Curve:** This is a commonly used graph that summarizes the performance of a classifier over all possible thresholds. It is generated by plotting the True Positive Rate (y-axis) against the False Positive Rate (x-axis) as you vary the threshold for assigning observations to a given class. ([**More details about ROC Curves.**](http://www.dataschool.io/roc-curves-and-auc-explained/))