There are a few scoring metric that come into play when evaluating your machine learning models. Imagine having a set of five flashcards with different fruits pictured on them. If you're trying to learn the names of the fruits in a new language, you'll have to do some training. Once you've memorized them all, you might hand the cards to a friend and ask her to test you. You've already seen all the cards before, along with their correct labels in the language you're studying. Your ability to recall, or a predictor's ability to get the expected label back when tested against the training data, is the simplest metric considered when evaluating machine learning algorithms. We haven't really dealt with this metric yet during the course.

But what happens when you go to the market and you actually want to purchase some fruit. The fruit you see in the shops might have slightly different coloration, or shapes and sizes from the one's pictured on your flash card. Your ability to predict, or a predictor's ability to generalize and get the expected label back when tested against unseen testing data, is the next metric considered when evaluating machine learning algorithms. Depending on your purposes, you might need an algorithm with very high recall and low predictive abilities, or vice versa. This is the metric we've been focusing on so far in this course.

Let's take the analogy a bit further. It just so happens that you're allergic to grapefruit, but you absolutely love oranges, lemons, limes, and tangerines. For you, it would be detrimental to mistakenly consume a grapefruit, mistaking it for an irresistibly large orange. But for your friend who enjoys both grapefruits and oranges, having purchased the wrong fruit might be a bit of a surprise, but nothing more than that. To account for such situations, we need to talk about the ratio of true-positives, true-negatives, false-positives, and false-negatives.

- **True Positive** You predicted it was a desired orange, and it was indeed an orange. Good job!
- **True Negative** You predicted it was an undesired grapefruit, and it was indeed a grapefruit.
- **False Positive** You predicted it was a delicious orange, but behold! It was a actually a sour grapefruit.
- **False Negative** You predicted it was a detested grapefruit, but it actually ended up being the best orange of the season

SciKit-Learn's metrics model helps you calculate many of these metrics automatically. Given the following setup:

```
>>> import sklearn.metrics as metrics
>>> y_true = [1, 1, 2, 2, 3, 3]  # Actual, observed testing datset value
>>> y_pred = [1, 1, 1, 3, 2, 3]  # Predicted values from your model
```

You already know how to calculate the accuracy_score, which is the same value you get from using `model.score()` on your predictor:

```
>>> metrics.accuracy_score(y_true, y_pred)
0.5

>>> metrics.accuracy_score(y_true, y_pred, normalize=False)
3
```

Recall

To calculate the recall score, or the ratio of true_positives / (true_positives + false_negatives):

```
>>> metrics.recall_score(y_true, y_pred, average='weighted')
0.5

>>> metrics.recall_score(y_true, y_pred, average=None)
array([ 1. ,  0. ,  0.5])
```

Precision

You can also calculated the precision score. It is defined very similarly: true_positives / (true_positives + false_positives). The only difference is the very last term in the equation:

```
>>> metrics.precision_score(y_true, y_pred, average='weighted')
0.38888888888888884

>>> metrics.precision_score(y_true, y_pred, average=None)
array([ 0.66666667,  0.        ,  0.5       ])
```

F1

The F1 Score is a weighted average of the precision and recall. Defined as 2 * (precision * recall) / (precision + recall), the best possible result is 1 and the worst possible score is 0:

```
>>> metrics.f1_score(y_true, y_pred, average='weighted')
0.43333333333333335

>>> metrics.f1_score(y_true, y_pred, average=None)
array([ 0.8,  0. ,  0.5])
```

Full Report

As a convenience, SciKit-Learn also has a built-in method for producing a full report of the above scores for you simultaneously, on a per label basis:

```
>>> target_names = ['Fruit 1', 'Fruit 2', 'Fruit 3']
>>> metrics.classification_report(y_true, y_pred, target_names=target_na


             precision    recall  f1-score   support

    Fruit 1       0.67      1.00      0.80         2
    Fruit 2       0.00      0.00      0.00         2
    Fruit 3       0.50      0.50      0.50         2

avg / total       0.39      0.50      0.43         6
```