Topic – 05
Part-2
Classification & more

Multiclass Classification

Like the name suggests, Multiclass Classification are problems where can be more than two classes for a solution and the instances are classified into one of the Classes.

Multiclass classification should not be confused with Multi-label classification where Multiple Labels are to be predicted for each instance.

For example:

- Random Forest
- Naïve Bayes

One vs All (OvA) Strategy / One vs the Rest Strategy

There are different ways to perform Multi-class Classification operations using Multiple Binary Classifiers. One of the widely used way is One vs All approach.

For example: We will make a solution which can identify digits, starting from 0 to 9.

Normally, this can be done by applying Binary Classification for each of the Digits (10 Binary Classification for 10 Digits; a 0-detector, a 1-detector, a 2-detector, and so on) and based on that, when we will be classifying an image, we get the decision score from each classifier for that image and you select the class whose classifier outputs the highest score.

One vs One (OvO) Strategy

Another approach to turn a Binary Classifier to a Multi-class Classifier is called One vs One Strategy.

In this case, we train a binary classifier for every pair of digits: one to distinguish 0s and 1s, another to distinguish 0s and 2s, another for 1s and 2s, and so on.

If there are N classes, we need to train $N \times (N - 1) / 2$ classifiers.

For our Handwritten Digit Recognition, this means training 45 binary classifiers! When we want to classify an image, we have to run the image through all 45 classifiers and see which class wins the most duels.

The main advantage of OvO is that each classifier only needs to be trained on the part of the training set for the two classes that it must distinguish.

Confusion Matrix

A better way to evaluate the performance of a classifier is to look at the **Confusion Matrix**. The idea is to count the number of times instances of class A are classified as class B.

For example, to know the number of times the classifier confused images of 5s with 3s, you would look in the 5th row and 3rd column of the confusion matrix. To compute the confusion matrix, you first need to have a set of predictions, so they can be compared to the actual targets. Each row in a confusion matrix represents an actual class, while each column represents a predicted class.

Scikit Learn has a function for Confusion Matrix. Just pass it the target classes (y_train_target) and the predicted classes (y_train_pred):

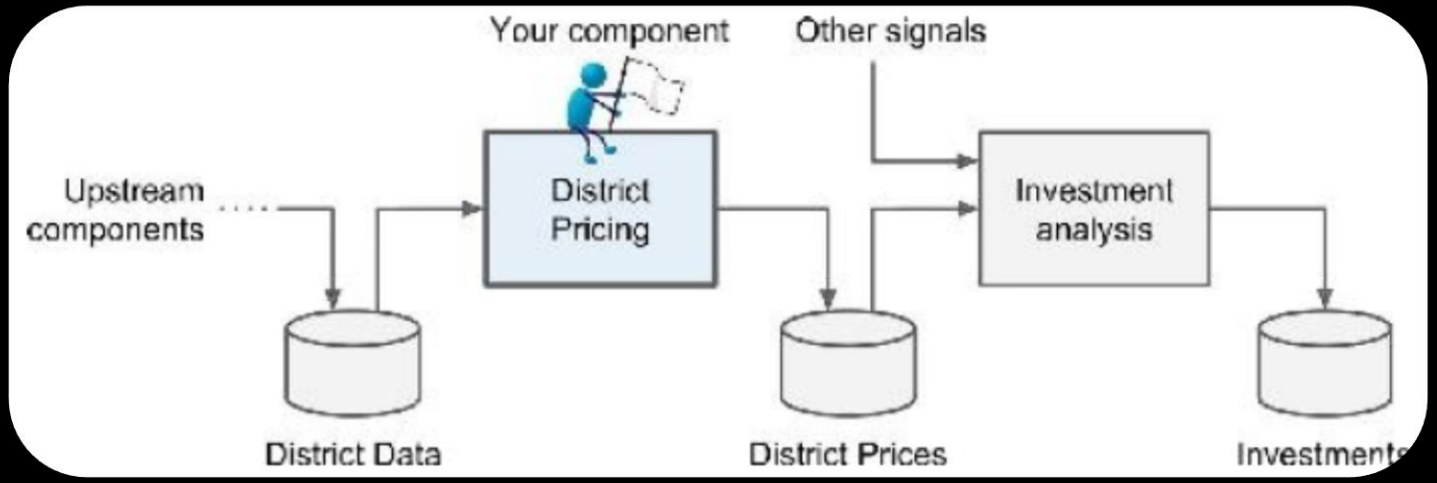
```
>>> from sklearn.metrics import confusion_matrix
>>> confusion_matrix(y_train_target, y_train_pred)
Output: array([[TN, FP], [ FN, TP]])
```

Framing the Problem

The first question to ask your boss is what exactly is the business objective; building a model is probably not the end goal. How does the company expect to use and benefit from this model? This is important because it will determine –

- How you frame the problem
- What algorithms you will select
- What performance measure you will use to evaluate
- How much effort you should spend tweaking it.

Your boss answers that your model's output (a prediction of a district's median housing price) will be fed to another Machine Learning system (see Figure 2-2), along with many other signals. This downstream system will determine whether it is worth investing in a given area or not. Getting this right is critical, as it directly affects revenue.



Data Pipeline

A sequence of data processing components is called a Data Pipeline. Pipelines are very common in Machine Learning systems, since there is a lot of data to manipulate and many data transformations to apply.

Components typically run asynchronously. Each component pulls in a large amount of data, processes it, and spits out the result in another data store, and then some time later the next component in the pipeline pulls this data and spits out its own output, and so on. Each component is fairly self-contained: the interface between components is simply the data store. This makes the system quite simple to grasp (with the help of a data flow graph), and different teams can focus on different components.

Moreover, if a component breaks down, the downstream components can often continue to run normally (at least for a while) by just using the last output from the broken component. This makes the architecture quite robust.

On the other hand, a broken component can go unnoticed for some time if proper monitoring is not implemented. The data gets stale and the overall system’s performance drops.

Performance Measure – Root Mean Square Error (RMSE)

A typical performance measure for regression problems is the Root Mean Square Error (RMSE). It gives an idea of how much error the system typically makes in its predictions, with a higher weight for large errors. Equation 2-1 shows the mathematical formula to compute the RMSE.

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

Here, m is the number of instances in the dataset you are measuring the RMSE on.
For example, if you are evaluating the RMSE on a validation set of 2,000 districts, then m = 2,000.

x(i) is a vector of all the feature values (excluding the label) of the ith instance in the dataset, and y(i) is its label (the desired output value for that instance).

Computing the root of a sum of squares (RMSE) corresponds to the Euclidian norm: it is the notion of distance you are familiar with. It is also called the ℓ2 norm,

Performance Measure – Mean Absolute Error (MAE) / Avg. Absolute Deviation

Another Performance measure is Mean Absolute Error.

For example: In the house Price problem, there are many outlier districts. In that case, you may consider using the Mean Absolute Error (also called the Average Absolute Deviation)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m} \left| h(\mathbf{x}^{(i)}) - y^{(i)} \right|$$

Computing the sum of absolutes (MAE) corresponds to the ℓ1 norm, noted $\cdot 1$.

It is sometimes called the Manhattan norm because it measures the distance between two points in a city if you can only travel along orthogonal city blocks.

RMSE vs MAE

The higher the norm index, the more it focuses on large values and neglects small ones.

This is why the RMSE is more sensitive to outliers than the MAE.

But when outliers are exponentially rare (like in a bell-shaped curve), the RMSE performs very well and is generally preferred.