Topic – 06
Preprocessing and Hyperparameter Configuration

+88 01831-661534

jehadfeni@gmail.com

Feature Scaling

Before you start training, in some of the cases, you have to transform your data.

One of the most important transformations you need to apply to your data is FEATURE SCALING. With few exceptions, Machine Learning algorithms don't perform well when the input numerical attributes have very different scales.

For example, in the case of House Pricing problem, if we look into the Housing Data:

The ***TOTAL NUMBER OF ROOMS*** ranges from about 6 to 39,320, while the ***MEDIAN INCOMES*** only range from 0 to 15.

Scaling the ***TARGET VALUES*** is generally not required.

There are two common ways to get all attributes to have the same scale:

- Min-Max Scaling (Used in most cases)
- Standardization

Min-Max Scaling (aka Normalization)

Min-Max Scaling is quite simple. In Min-Max Scaling, values are shifted and rescaled so that they end up ranging from ***0 to 1***.

We do this by subtracting the ***MIN VALUE*** from the ***CURRENT VALUE*** and dividing by the ***MAX VALUE*** minus the ***MIN VALUE***.

$$\text{Min Max Scaling} = \frac{\text{CURRENT VALUE} - \text{MIN VALUE}}{\text{MAX VALUE} - \text{MIN VALUE}}$$

Scikit-Learn provides a transformer called ***MinMaxScaler*** for this operation.

It has a ***feature_range hyperparameter*** that lets you change the range if you don't want 0–1 for some reason.

Better Evaluation using Cross-Validation

One way to evaluate the any model would be to use the ***train_test_split*** function to split the training set into a smaller training set and a validation set, then train your models against the smaller training set and evaluate them against the validation set. It's a bit of work, but nothing too difficult and it would work fairly well.

A great alternative is to use Scikit-Learn's ***cross-validation*** feature. The following code performs K-fold cross-validation: it randomly splits the training set into 10 distinct subsets called folds, then it trains and evaluates the Model 10 times, picking a different fold for evaluation every time and training on the other 9 folds.

Grid Search

When you will be finding the most suitable value of Hyperparameter, one way is manually inputting different values for Testing, which is not very helpful. You may feel tired after some time.

Instead of doing that you can use the Scikit-Learn’s GridSearchCV to search for you.

All you need to do is tell it which hyperparameters you want it to experiment with, and what values to try out, and it will evaluate all the possible combinations of hyperparameter values, using cross-validation.

Randomized Search

When the hyperparameter search space is large, it is often preferable to use RandomizedSearchCV instead of GridSearchCV. This class can be used in much the same way as the GridSearchCV class.

But instead of trying out all possible combinations, it evaluates a given number of random combinations by selecting a random value for each hyperparameter at every iteration.

This approach has two main benefits: If you let the randomized search run for, say, 1,000 iterations, this approach will explore 1,000 different values for each hyperparameter (instead of just a few values per hyperparameter with the grid search approach). You have more control over the computing budget you want to allocate to hyperparameter search, simply by setting the number of iterations.

Correlation Coefficient / Pearson's R

If the data you are working with is not too large, you can easily compute the standard Correlation Coefficient (also called Pearson's R) between every pair of attributes using the ***corr()*** method.

```
corr_matrix = housing.corr()
```

The correlation coefficient ranges from –1 to 1. When it is close to 1, it means that there is a strong positive correlation. For example, the ***MEDIAN HOUSE VALUE*** tends to go up when the ***Median Income*** goes up.

When the coefficient is close to –1, it means that there is a strong negative correlation. For example: There is a small negative correlation between the ***latitude*** and the ***Median House Value***.

Finally, coefficients close to zero mean that there is no linear correlation

Attribute Combinations

One final step before starting your Training is to try out various ***Attribute Combinations***.

For example, the ***Total Number of Rooms in a District*** is not very useful if you don't know how many ***Households*** there are. What you really want is the **Number of Rooms per Household**.

Similarly, the ***Total Number of Bedrooms*** by itself is not very useful: you probably want to compare it to the ***Number of Rooms***. And the **Population Per Household** also seems like an interesting attribute combination to look at.

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"]=housing["population"]/housing["households"]
```

Doing this can improve the Accuracy later on.