📚 **Tutorial: baseline RS methods in `BaseEstimator` type**

Step-by-Step Conversion

1. **Inherit from `BaseEstimator`**:
   - We will inherit from `BaseEstimator` to get basic functionality like hyperparameter setting and getting.
2. **Define the `fit` and `predict` methods**:
   - The `fit` method will compute the global mean of the training ratings.
   - The `predict` method will return the global mean for each test pair.

Here's the example code for global mean RS methods:

```
import numpy as np
from sklearn.base import BaseEstimator

class GlobalMeanRS(BaseEstimator):
    def __init__(self):
        # model parameters
        self.glb_mean_ = 0

    def fit(self, rating):
        # fit parameter
        self.glb_mean_ = np.mean(rating)

    def predict(self, pair):
        # Ensure the estimator is fitted
        check_is_fitted(self, 'glb_mean_')
        # Return the global mean for each test pair
        r_pred = np.ones(len(pair))
        return r_pred*self.glb_mean_
```

Explanation

1. **Initialization (`__init__` method)**:
   - Initialize `glb_mean_` to 0. The trailing underscore indicates that this is an attribute set during the `fit` method.
2. **Fitting the Model (`fit` method)**:
   - Compute the global mean of the training ratings and store it in `glb_mean_`.
3. **Making Predictions (`predict` method)**:
   - Ensure the estimator has been fitted using `check_is_fitted`.
   - Return the global mean for each test pair.

Usage

The example usage demonstrates how to use this custom estimator with Scikit-learn's familiar `fit`, `predict` methods. This allows you to leverage Scikit-learn's powerful tools for model selection, evaluation, and preprocessing while using your tailored algorithms.

By following this approach, you can create custom machine learning estimators that integrate seamlessly with Scikit-learn's ecosystem.