Explain the working of HoeffdingTreeRegressor present in river.tree

The **Hoeffding Tree Regressor (HTR)** in `river.tree` is an incremental decision tree algorithm adapted for regression tasks. It is based on the principles of the Hoeffding Tree Classifier but focuses on reducing variance in the target space to decide splits, making it suitable for predicting continuous values.

Key Components and Parameters

1. **Hoeffding Bound**: This statistical bound is used to determine when to split a node. It ensures that the decision to split is made with a certain level of confidence, based on the number of observations.
2. **Parameters**:
    - `grace_period`: The number of instances a leaf must observe before attempting to split. Default is 200.
    - `max_depth`: The maximum depth of the tree. If `None`, the tree grows until it reaches the system recursion limit.
    - `delta`: Significance level for the Hoeffding bound. Lower values imply longer delays in split decisions. Default is

      $$1e-07$$

      .
      - `tau`: Threshold to force splits when ties occur. Default is 0.05.
      - `leaf_prediction`: Mechanism used for predictions at leaf nodes. Options include `'mean'`, `'model'`, and `'adaptive'`. Default is `'adaptive'`.
      - `leaf_model`: The regression model used if `leaf_prediction='model'`. Defaults to `LinearRegression`.

How It Works

1. **Incremental Learning**: The HTR learns incrementally from a stream of data, updating its structure as new data arrives.
2. **Split Decisions**: The algorithm uses the Hoeffding bound to decide when to split a node. It calculates the reduction in variance at potential split points and chooses the one that results in the most homogeneous partitions.
3. **Leaf Node Predictions**: At each leaf node, the model can either use the mean of the target values (`'mean'`) or fit a linear model (`'model'`) to make predictions. The `'adaptive'` strategy dynamically chooses between these methods.

4. **Continuous Updates**: As new data arrives, the tree continuously updates its structure by adding new nodes or adjusting existing ones based on the observed variance reduction.

Example Usage

```
from river import tree
from river import datasets
from river import metrics

# Create a Hoeffding Tree Regressor model
model = tree.HoeffdingTreeRegressor(grace_period=100)

# Evaluate the model on a dataset
dataset = datasets.TrumpApproval()
metric = metrics.MAE()
evaluate.progressive_val_score(dataset, model, metric)
```

This example demonstrates how to create a `HoeffdingTreeRegressor` and evaluate its performance on a dataset using the Mean Absolute Error (MAE) metric.

⁂