SELECTING HYPERPARAMETERS FOR THE S&P 500

This section aims to find tree models that accurately represent the distributions of returns for the S&P 500 index. These models help us understand how the distribution changes across different macroeconomic scenarios. To do this, we'll use two metrics: the range score and the variance of the model's leaves. The range score shows if there are enough out-of-sample points within the distribution limits, while the standard deviation of the tree's leaves indicates if there's sufficient diversity among the distributions.

HYPERPARAMETER SELECTION

CCP_ALPHA AND MIN_IMPURITY DECREASE

Firstly, we will build several models using different hyperparameter values. For each model, we will determine the out-of-sample model evaluation measure using cross-validation and select the models with the best results. Some of the best models and the hyperparameters that were used to build them can be seen below.

```
{'ccp_alpha': 0.004, 'criterion': 'gini', 'max_depth': 4, 'max_features': 0.01, 'max_leaf_nodes': None, 'min_impurity_d
ecrease': 0, 'min_samples_leaf': 0.05, 'min_samples_split': 2, 'random_state': 244, 'splitter': 'best'}
```

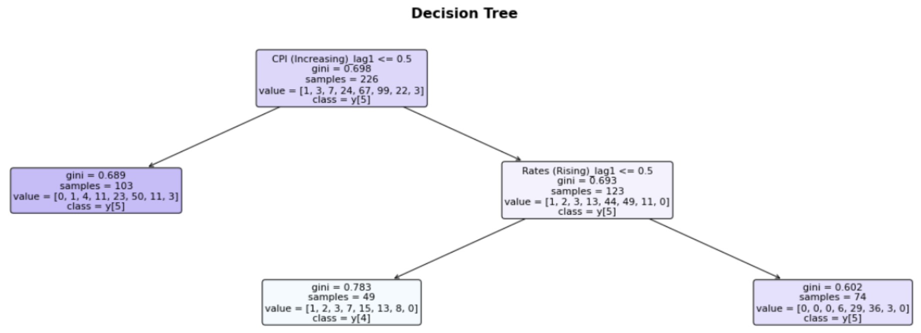

Decision Tree
CPI (Increasing)_lag1 <= 0.5
gini = 0.698
samples = 226
value = [1, 3, 7, 24, 67, 99, 22, 3]
class = y[5]
gini = 0.689
samples = 103
value = [0, 1, 4, 11, 23, 50, 11, 3]
class = y[5]
Rates (Rising)_lag1 <= 0.5
gini = 0.693
samples = 123
value = [1, 2, 3, 13, 44, 49, 11, 0]
class = y[5]
gini = 0.783
samples = 49
value = [1, 2, 3, 7, 15, 13, 8, 0]
class = y[4]
gini = 0.602
samples = 74
value = [0, 0, 0, 6, 29, 36, 3, 0]
class = y[5]

Image 1. Trivial Decision Tree # 1.

```
{'ccp_alpha': 0.004, 'criterion': 'gini', 'max_depth': 4, 'max_features': 0.01, 'max_leaf_nodes': None, 'min_impurity_d
ecrease': 0.001, 'min_samples_leaf': 0.05, 'min_samples_split': 10, 'random_state': 61, 'splitter': 'best'}
```

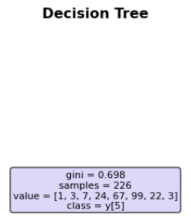


Image 2. Trivial Decision Tree # 2.

```
{'criterion': 'squared_error', 'max_depth': 4, 'max_features': 0.1, 'min_samples_leaf': 0.3, 'random_state': 365, 'spli
tter': 'best'}
```

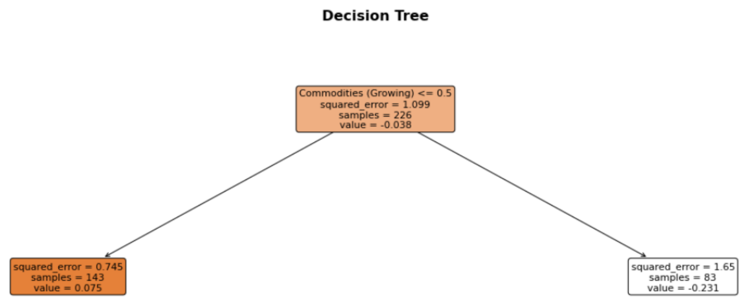


Image 3. Trivial Decision Tree # 3.

These models exhibit good accuracy and squared error measures, but unfortunately, they have a reduced number of nodes. Additionally, many of the models show little variation among the tree's leaves. Furthermore, many of them have range scores below 0.74.

The *ccp_alpha* hyperparameter (Cost Complexity Pruning alpha) in decision trees controls the pruning process to prevent overfitting. It determines how aggressively the tree is pruned after construction. A higher *ccp_alpha* value leads to more aggressive pruning, resulting in a smaller and less complex tree. Adjusting this hyperparameter helps control the complexity of the tree and enhances its ability to generalize to unseen data.

The *min_impurity_decrease* hyperparameter in decision trees controls the node splitting process based on impurity decrease. Impurity is a measure of the disorder or uncertainty in a set of data points within a node. When deciding whether to split a node further, the algorithm calculates the impurity decrease that would result from the split. If this decrease is greater than or equal to the *min_impurity_decrease* value, the split is made; otherwise, the node is kept as a leaf node.

From the results, it's evident that trivial trees are obtained when the values of *ccp_alpha* and *min_impurity_decrease* are greater than 0. To find trees with better range scores, higher leaf variance, and non-trivial structures, we will seek hyperparameters that favor these characteristics. Firstly, we will opt to set *ccp_alpha* and *min_impurity_decrease* values to 0.

DISTRIBUTION ESTIMATION

Empirical Distribution

To determine the return distribution in each leaf of the tree, we will employ three different methods. Initially, we will assume that the distribution is exactly the same as the distribution of the samples within each leaf of the tree. Consequently, the values of the 5th and 95th percentiles will correspond to the 5th and 95th percentiles of the samples. This distribution will be referred to as the empirical distribution.

Normal Distribution

Secondly, we will assume that the return distribution in the leaves follows a normal distribution, where the mean equals the mean of the samples and the standard deviation also corresponds to the standard deviation of the samples. The values of the 5th and 95th percentiles will be calculated in the same manner as for a normal distribution:

$$percentile_{0.05} = samples_{mean} - 1.645 * samples\_std$$

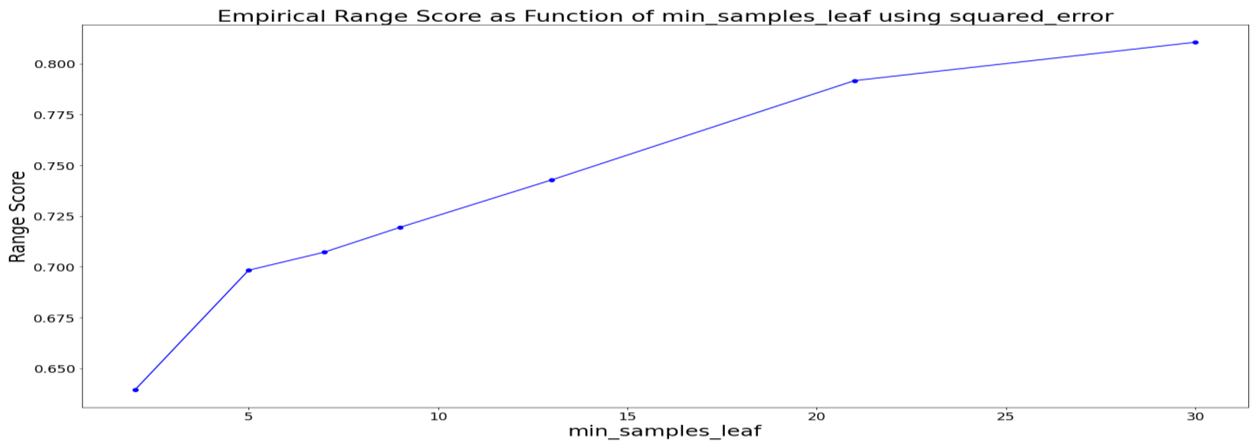
$$percentile_{0.95} = samples_{mean} + 1.645 * samples\_std$$

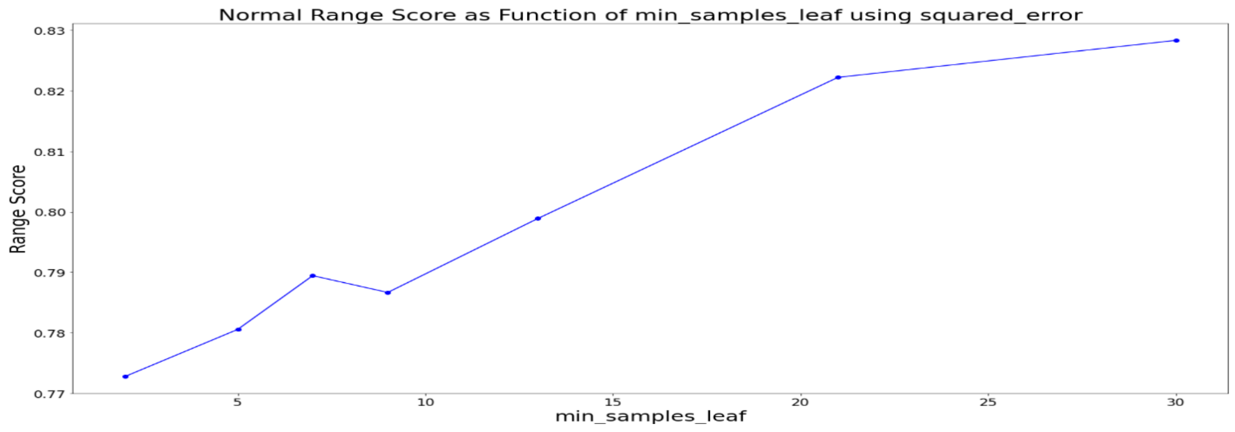
Interpolated Distribution

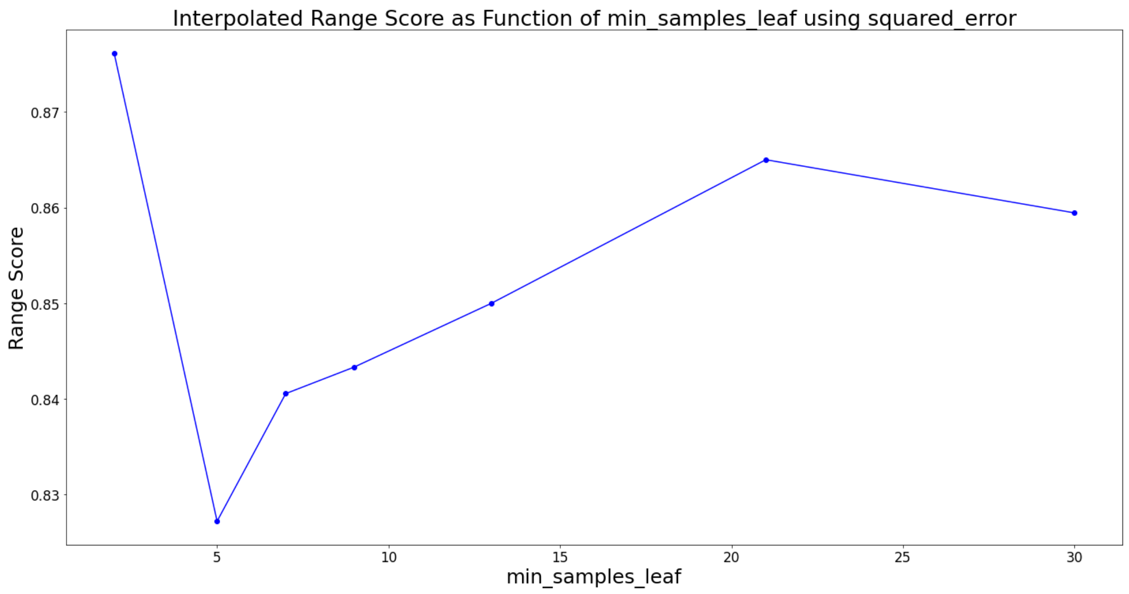
Finally, we will estimate a smoothed distribution based on the observations from the leaves. This distribution will not adhere to the same characteristics as a normal distribution; the skewness may be different from 0, and so may the kurtosis. This distribution will be calculated using the Gaussian_kde algorithm in Python. The percentiles will be computed based on the area under the curve of the estimated distribution.

$$percentile_{0.05} = x, s.t. Int(x * \ density(x)) = 0.05$$

$$percentile_{0.95} = x, s.t. Int(x * \ density(x)) = 0.95$$



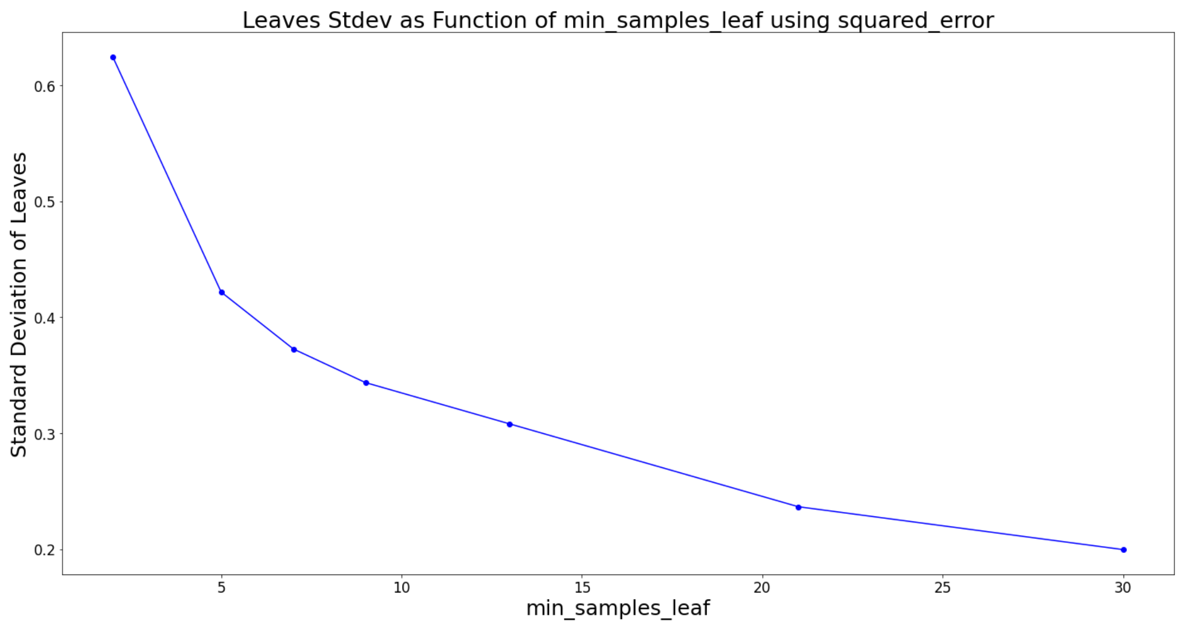




To pinpoint the distribution that best fits the out-of-sample data, we'll construct decision trees with varying selected variables and min_samples_leaf values. For each tree, we'll estimate the range score using these three types of distributions. Based on the results, it's evident that the interpolated distribution achieves the highest range score. Therefore, this is the distribution we'll employ for our analyses.

MIN_SAMPLES_LEAF

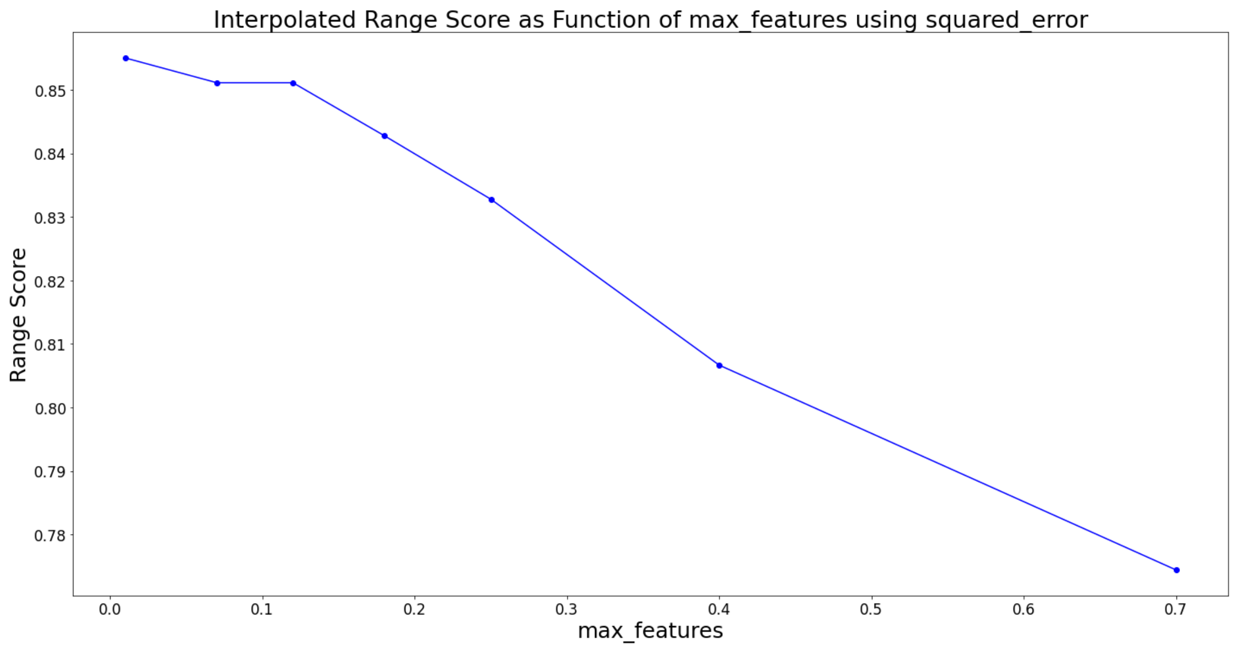
Secondly, it's apparent that the min_samples_leaf parameter significantly influences the range score. It's also notable that the variance of the leaves decreases as the min_samples_leaf increases. The relationship between this parameter and the variance of the leaves can be observed in the graph below. To obtain models with a sufficiently high range score yet also a large leaf variance, we will choose to set this parameter to an intermediate value of 7. It's worth mentioning that at this point on the graph, there is no substantial improvement when decreasing min_samples_leaf and at the same time there is no significant improvement in the range score when increasing min_samples_leaf.



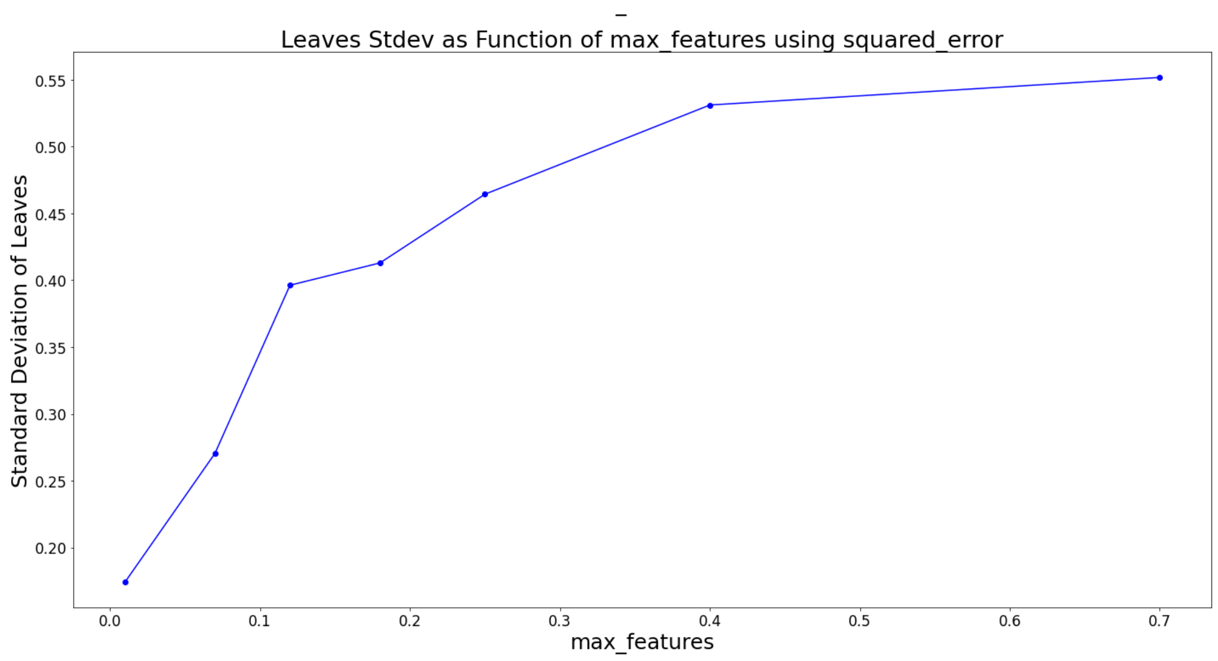
Graph 3. Leaves standard deviation vs ‘min_samples_leaf’ using ‘absolute_error’ criterion.

MAX_FEATURES

Additionally, it's clear that the best-performing models aren't just the ones the algorithm picks by default. Some models work well, but they might not be the ones that minimize absolute error or Gini impurity the best. To find these models, we need to set the 'max_features' hyperparameter to a value less than 1. This change makes the algorithm randomly choose variables for each split, giving us different models from the one that optimally reduces absolute_error. The graph below illustrates the relationship between max_features and range score, as well as max_features and the standard deviation of the leaves.



Graph 4. Range score vs 'max_features' using 'squared_error' criterion.



Graph 5. Leaves' standard deviation vs 'max_features' using 'squared_error' criterion.

As we decrease max_features, we see an increase in the range score but a decrease in the variance of the leaves. In the graph, it's evident that for values above 0.18, there's no significant enhancement in the variance. Additionally, at this threshold, the range score reaches a stable level. Hence, we'll set this parameter to 0.18.

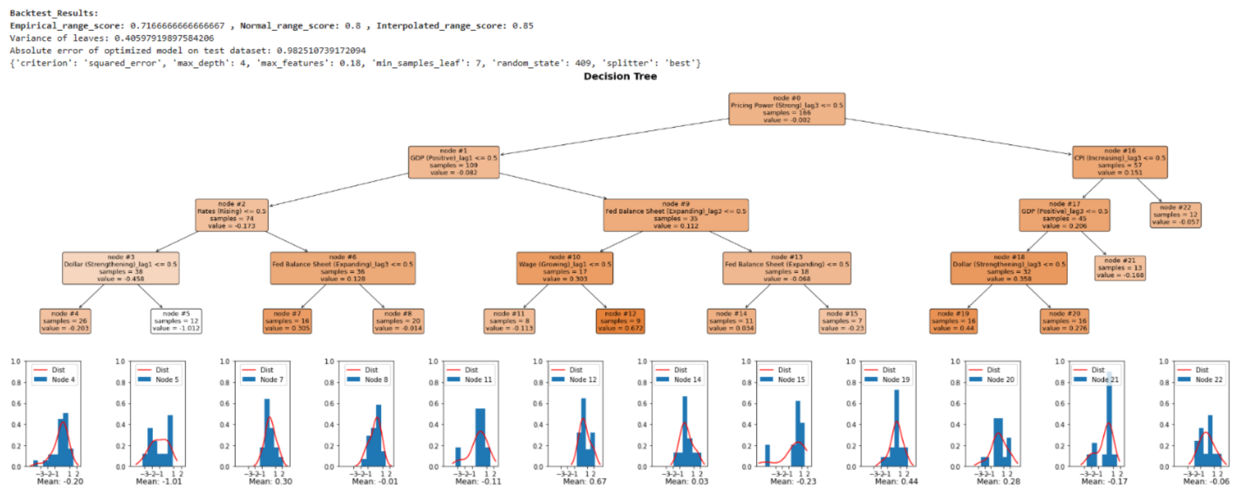
SCORING MECHANISM AND CRITERION

To identify the most promising models, we'll conduct extensive testing. This involves randomly selecting 600 states and constructing tree models using previously determined hyperparameters and three different criteria: Gini impurity, absolute error, and squared error. These models will be trained on the training dataset, followed by evaluation of their range score and standard deviation of their leaves on the validation set. We'll then select models with a leaf standard deviation greater than 0.4, and from this subset, we'll choose the top 50 models based on their range score. Subsequently, we'll assess the performance of these 50 models on the validation set. The table below displays the performance of these models.

- Gini impurity: Outlier events do not contribute to the tree's construction.
- Absolute error: Outlier events will have some relevance in the tree's construction.
- Squared error: Outlier events will have a substantial contribution to the model's construction.

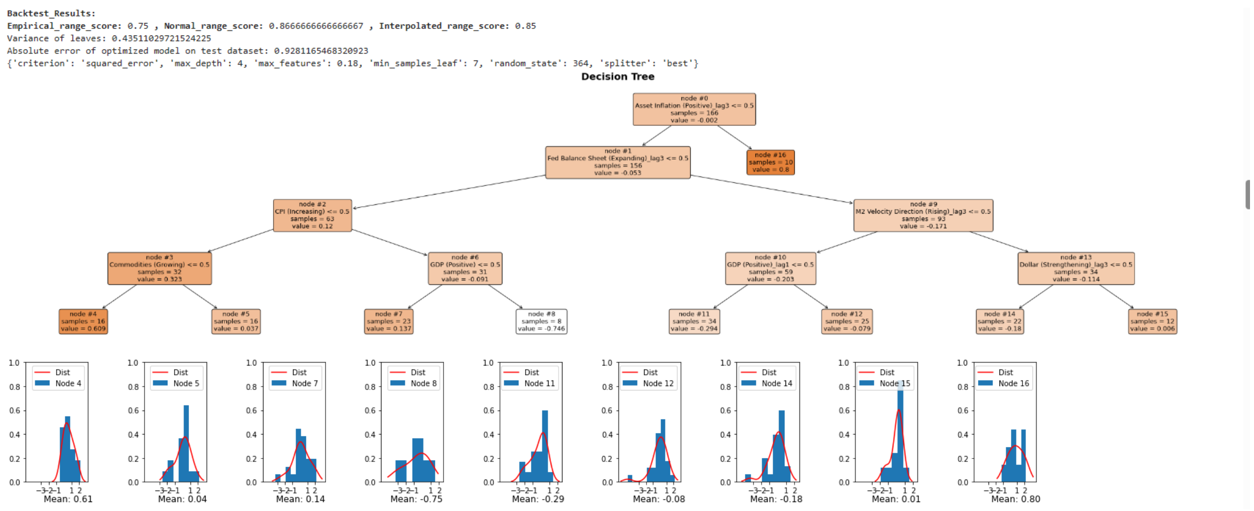
Criterion	Number of models with leaves' standard deviation > 0.4	Mean Out-of-sample range score of 50 best models	Mean Out-of-sample range score of 50 worst models
gini	48	80%	70%
absolute_error	84	82%	73%
squared_error	281	86%	71%

According to the results, it can be observed that with the squared error criterion, a higher number of models with leaf standard deviations greater than 0.4 are obtained. Additionally, it can be observed with this criterion that the range score on the validation set is higher among the 50 best models. For these reasons, we will select the squared error as the criterion for tree construction.



```
Backtest_Results:
Empirical_range_score: 0.7166666666666667 , Normal_range_score: 0.8 , Interpolated_range_score: 0.85
Variance of leaves: 0.40597919897584206
Absolute error of optimized model on test dataset: 0.982510739172094
{'criterion': 'squared_error', 'max_depth': 4, 'max_features': 0.18, 'min_samples_leaf': 7, 'random_state': 409, 'splitter': 'best'}
```

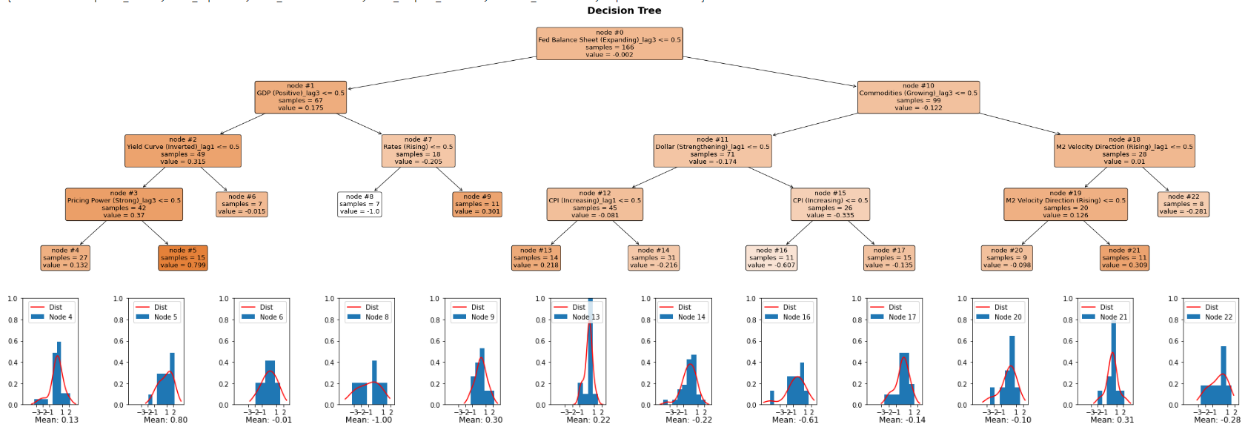
Image 4. Decision Tree # 1 using selected hyperparameters



```
Backtest_Results:
Empirical_range_score: 0.75 , Normal_range_score: 0.8666666666666667 , Interpolated_range_score: 0.85
Variance of leaves: 0.43511029721524225
Absolute error of optimized model on test dataset: 0.9281165468320923
{'criterion': 'squared_error', 'max_depth': 4, 'max_features': 0.18, 'min_samples_leaf': 7, 'random_state': 364, 'splitter': 'best'}
```

Image 5. Decision Tree # 2 using selected hyperparameters

```
Backtest_Results:
Empirical_range_score: 0.7666666666666667 , Normal_range_score: 0.8 , Interpolated_range_score: 0.8666666666666667
Variance of leaves: 0.445415992215159
Absolute error of optimized model on test dataset: 0.9663777144858271
{'criterion': 'squared_error', 'max_depth': 4, 'max_features': 0.18, 'min_samples_leaf': 7, 'random_state': 205, 'splitter': 'best'}
```



Decision Tree
node #0 Fed Balance Sheet (Expanding)_lag3 <= 0.5 samples = 166 value = -0.002
node #1 GDP (Positive)_lag3 <= 0.5 samples = 67 value = 0.175
node #2 Yield Curve (Inverted)_lag1 <= 0.5 samples = 49 value = 0.315
node #3 Pricing Power (Strong)_lag3 <= 0.5 samples = 42 value = 0.37
node #4 samples = 27 value = 0.132
node #5 samples = 15 value = 0.799
node #6 samples = 7 value = -0.015
node #7 Rates (Rising) <= 0.5 samples = 18 value = -0.205
node #8 samples = 7 value = -1.0
node #9 samples = 11 value = 0.301
node #10 Commodities (Growing)_lag3 <= 0.5 samples = 99 value = -0.122
node #11 Dollar (Strengthening)_lag1 <= 0.5 samples = 71 value = -0.174
node #12 CPI (Increasing)_lag1 <= 0.5 samples = 45 value = -0.081
node #13 samples = 14 value = 0.218
node #14 samples = 31 value = -0.216
node #15 CPI (Increasing) <= 0.5 samples = 26 value = -0.335
node #16 samples = 11 value = -0.607
node #17 samples = 15 value = -0.135
node #18 M2 Velocity Direction (Rising)_lag1 <= 0.5 samples = 28 value = 0.01
node #19 M2 Velocity Direction (Rising) <= 0.5 samples = 20 value = 0.126
node #20 samples = 9 value = -0.098
node #21 samples = 11 value = 0.309
node #22 samples = 8 value = -0.281
Dist
Node 4
Mean: 0.13
Node 5
Mean: 0.80
Node 6
Mean: -0.01
Node 8
Mean: -1.00
Node 9
Mean: 0.30
Node 13
Mean: 0.22
Node 14
Mean: -0.22
Node 16
Mean: -0.61
Node 17
Mean: -0.14
Node 20
Mean: -0.10
Node 21
Mean: 0.31
Node 22
Mean: -0.28

Image 6. Decision Tree # 3 using selected hyperparameters