**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.

A computer screen shot of a diagram

Description automatically generated

Image 1. Trivial Decision Tree # 1.

A computer screen shot of a computer code

Description automatically generated

Image 2. Trivial Decision Tree # 2.

A computer screen shot of a tree

Description automatically generated

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:

*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.

A graph with a line

Description automatically generated

A graph with a line

Description automatically generated

A graph with a line

Description automatically generated

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.

A graph with a line

Description automatically generated

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.

A graph with a line

Description automatically generated

Graph 4. Range score vs ‘max\_features’ using ‘squared\_error’ criterion.

A graph with a line

Description automatically generated

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. Top of FormA diagram of a tree

Description automatically generatedA computer screen shot of a diagram

Description automatically generatedA diagram of a tree

Description automatically generatedA diagram of a tree

Description automatically generatedA diagram of a tree

Description automatically generatedA diagram of a tree

Description automatically generated

* 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.



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.

A diagram of a graph

Description automatically generated with medium confidence

Image 4. Decision Tree # 1 using selected hyperparameters

A diagram of a graph

Description automatically generated with medium confidence

Image 5. Decision Tree # 2 using selected hyperparameters

A diagram of a graph

Description automatically generated

Image 6. Decision Tree # 3 using selected hyperparameters