Hyperparameter Tuning Methods

Consistently tuning hyperparameters for Gradient Boosting Machines and Random Forest models to improve their performance. The process includes testing various configurations using grid search for smaller ranges and random search for larger ones. Each approach is combined with cross-validation to make sure the results remain reliable across various data sets.

Gradient Boosting Machines:

- **Learning Rate:**
  Tested values of 0.01, 0.05, 0.1, and 0.2 to control the pace of learning. Lower values like 0.01 allow the model to learn slowly, reducing the risk of missing the best solutions, while higher values like 0.2 accelerate training but might cause the model to miss the desired outcome.
- **Number of Estimators:**
  Tested configurations with 100, 200, 500, and 1000 estimators. Increasing the number of estimators tends to improve model accuracy, but it also extends training time and can increase the chance of overfitting.
- **Max Depth:**
  Explored tree depths of 3, 5, 7, and 9 to find the right balance between capturing data complexity and avoiding overfitting. Smaller trees limit model complexity, while larger trees provide more detail but may result in overfitting.
- **Subsample:**
  Values of 0.5, 0.7, and 1.0 were tested to examine the effect of training on random subsets of data. Subsampling reduces overfitting by adding randomness, while using the full dataset guarantees complete data utilization.
- **Colsample_bytree:**
  Evaluated 0.5, 0.7, and 1.0 to control the number of features used when building each tree. Sampling fewer features can improve model stability by reducing the risk of overfitting to insignificant features.
- **Progress:**
  The combination of learning_rate = 0.1, max_depth = 5, and n_estimators = 200 has provided the best performance so far. This setup offers a good balance between accuracy, training time, and overfitting. Additionally, early stopping is used to end training when

validation performance stabilizes, preserving computational resources and reducing overfitting.

Random Forest:

- **Number of Trees:**
  Tested different configurations with 50, 100, 200, and 500 trees to find the ideal model size. Adding more trees usually improves accuracy by making predictions more precise, though it also increases computation time and model complexity.
- **Max Depth:**
  Evaluated tree depths of none, 10, 20, and 30 to find the best balance between handling data complexity and preventing overfitting. Smaller trees help avoid overfitting, while larger trees can reveal more details but might become too specific to the training data.
- **Min Samples Split:**
  Ttested values of 2, 5, and 10 to determine the fewest number of samples needed to split a node. A higher number makes sure the model only splits when there's enough data to make reliable predictions, reducing overfitting.
- **Max Features:**
  Explored sqrt, log2, and none to decide how many features should be used to split each node. Limiting the number of features at each split can reduce overfitting and improve the model's performance.
- **Progress:**
  A configuration of 100 trees, max_depth = 20, and sqrt for max features is providing stable validation scores, indicating that the model performs well on new data and is not overfitting.

Justifications for Tuning Choices:

- **GBMs:**
  These models are effective at identifying complex relationships between features. Tuning hyperparameters like the learning rate and max depth helps balance model complexity and training time, while also controlling overfitting.
- **Random Forests:**
  These models are effective with both categorical and continuous features. Important parameters like max features and max depth are vital for controlling the model’s complexity, preventing it from becoming too detailed and overfitting the data.

- **Cross-Validation:**
  This technique helps confirm that the model's performance is consistent and reliable by testing it on various sections of the dataset. It prevents results from being skewed by one specific data split.
- **Early Stopping:**
  This approach saves time and computational resources by stopping training once the model's performance on the validation set stops improving. It also helps reduce overfitting by preventing unnecessary training that does not improve results.

The tuning process has effectively identified the best ranges for key parameters. Currently, the Gradient Boosting Machine outperforms other models due to its ability to recognize complex relationships and non linear patterns in the data. In contrast, Random Forest provides strong generalization with less tuning, making it easier to work with. Further adjustments and testing will help improve these models to boost their performance even more.

Results & Brief Interpretations

Current Best Model:

The Gradient Boosting Machine is currently the best performing model. It outperforms other models, including Random Forest, because it effectively identifies complex, non linear relationships between features. GBM's ability to model interactions between features makes it highly effective for this dataset, where feature interactions play a significant role in predicting the target variable.

Why is GBM the best?

- **Accuracy:**
  GBM consistently delivers higher accuracy compared to Random Forest due to its ability to identify more complex patterns in the data.
- **Training Time:**
  The configuration of learning_rate = 0.1, max_depth = 5, and n_estimators = 200 achieves a good balance between model accuracy and training efficiency, allowing fast progress and reducing computational cost.
- **Overfitting Control:**
  GBM's tuning helps maintain a balance between model complexity and avoiding overfitting. The model's performance is stable across both training and validation datasets, and early stopping guarantees that it does not continue training once the validation score stabilizes, further reducing overfitting risk.

Interpretation of Model Metrics:

R2 Score:

The R2 score is a measure of how well the model's predictions match the actual data. A higher R2 score shows that the model explains more of the variance in the target variable.

- For GBM, the higher R2 shows its ability to identify complex patterns and interactions between the features, making it a better fit for predicting salary based on the features available in the dataset.
- Random Forest performs well but doesn't handle the interactions between features as effectively as GBM. It is a simpler model that works well for generalization.

Mean Squared Error:

MSE measures the average squared difference between predicted and actual values. A lower MSE means better performance.

- GBM has a lower MSE, indicating that its predictions are closer to the true values than those of Random Forest. This reinforces the fact that GBM is providing better predictions on the test data.
- Random Forest, while offering strong generalization, has a higher MSE, which implies it may not be accounting for all the complexities of the data as well as GBM.

Black Box Models & Overfitting Reassurance:

Both GBM and Random Forest are black box models, meaning they are complex and harder to understand in terms of feature importance or direct coefficient analysis. However, there are still ways to confirm the models are not overfitting.

Overfitting Reassurance:

- **Cross validation:**
  Both models were validated using cross validation, which guarantees that the models are not overfitting to a specific section of the data. By training on various parts of the data, can confirm that the models generalize well and are not overfitting the data.
- **Early Stopping for GBM:**
  The use of early stopping for GBM stops the training process when the validation

performance stops improving. This prevents the model from learning unnecessary patterns in the training data, helping reduce overfitting.

- **Hyperparameter Tuning:**
  Both models were carefully tuned to identify the best hyperparameters. For GBM, selected values for learning_rate, n_estimators, and max_depth that balance training time, model complexity, and performance. For Random Forest, tuning of max_depth, min_samples_split, and n_estimators guarantees the model handles enough complexity without overfitting.