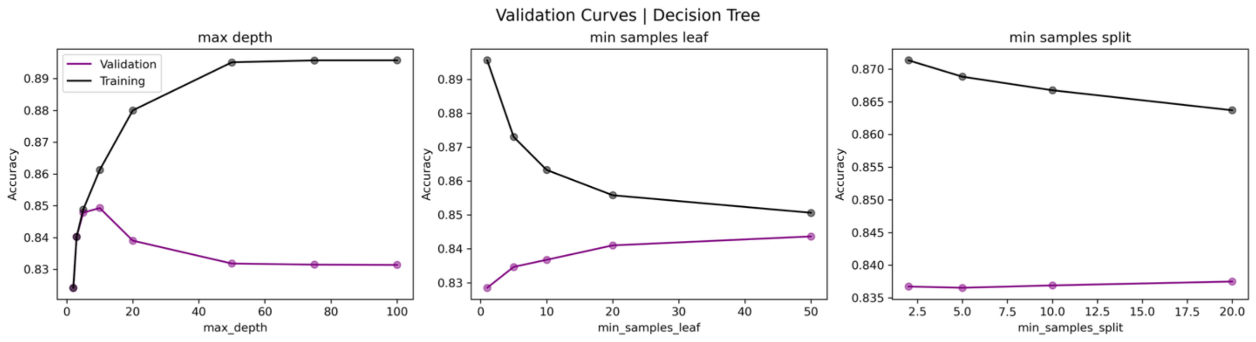
Dimitris Markopoulos
Statistical Machine Learning.

May 1, 2025
Columbia University

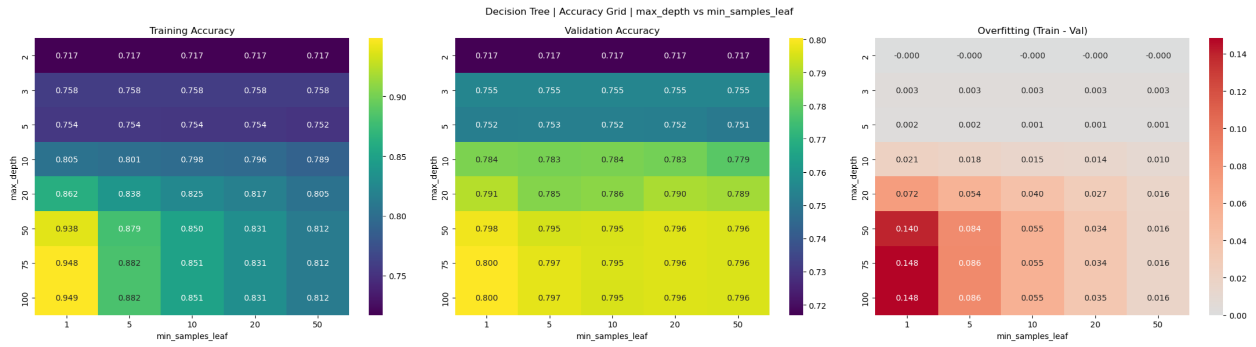
Note: I prioritized accuracy in the evaluation, therefore in training I did not assign weights to the minority dataset (earning over 50K) – except for Decision Tree notebook 02 I showed both routes of balancing and leaving unbalanced without weighing the smaller sized class. In the case of mislabeling someone with higher income as earning less income, I judged this not harmful and therefore valued overall accuracy.

Part (1Ai) – Overfitting.

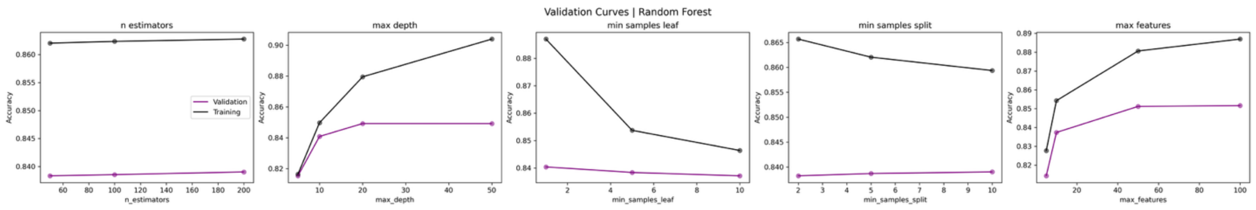
■ **Decision Tree**: I was able to overfit (and underfit) using a Decision Tree – evidenced by divergence in accuracy between training and validation set depending on some key hyperparameters. The hyperparameters: $[max\_depth, min\_sample\_leaf]$ were significant in determining if the model overfit to the training set, for example we can see by the visualization below that for $max\_depth > 10$ the training accuracy continues to grow while the validation accuracy levels off – this is a clear sign of overfitting! Likewise, when $min\_sample\_leaf < 20$ – the nodes become too pure and overfit to the training set.



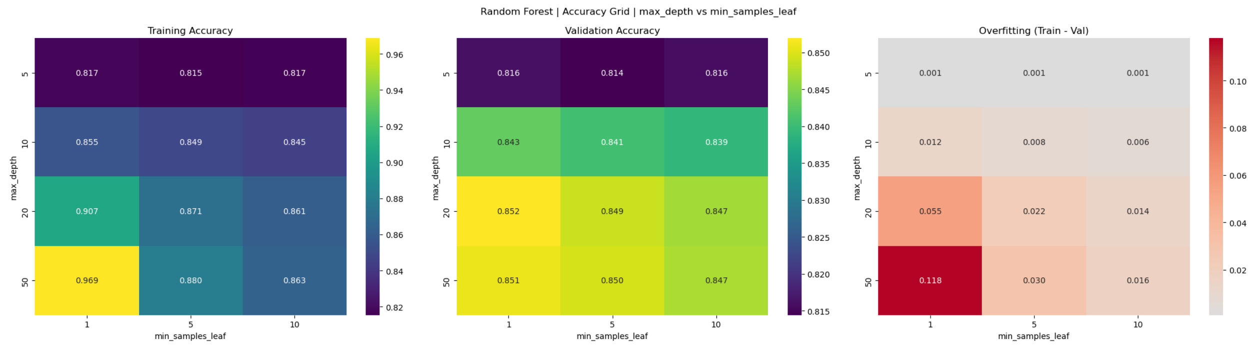
The plot below helps to verify this by taking both training and validation accuracy across a grid of hyperparameters and visualizing the difference (right) between the two. We can clearly see that as max_depth increases, and min_sample_leaf decreases we see a larger difference between the training and validation (and it is useful to note as max_depth decreases we see training = validation which supports the idea that we are underfitting and the model lacks complexity).



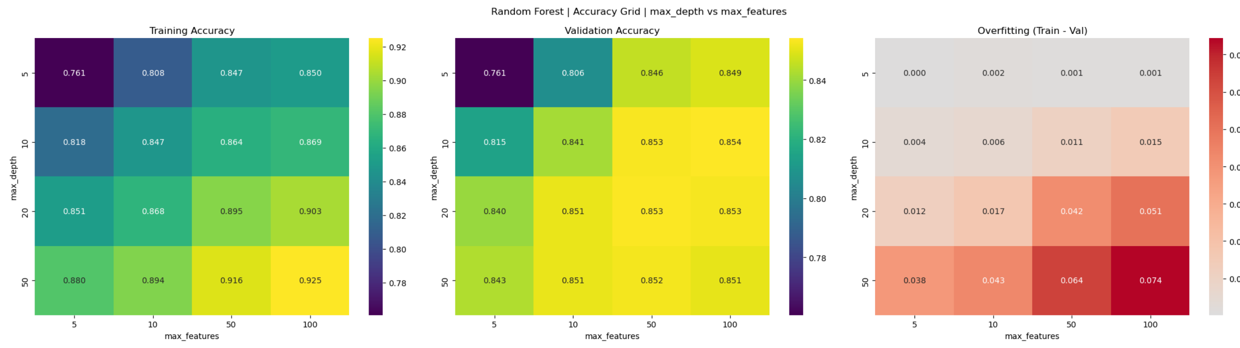
■ **Random Forest**: I observed overfitting in the Random Forest model, indicated by a widening gap between training and validation accuracy as certain hyperparameters were adjusted. Specifically, when $max\_depth > 10$, $min\_samples\_leaf < 5$, and $max\_features > 10$, the training accuracy increased while validation accuracy plateaued — a classic signal of overfitting. These hyperparameters control the tree complexity and generalization ability. Since a Random Forest is an ensemble of decision trees trained via bootstrapping and feature subsampling (bagging), its generalization benefits rely on the Law of Large Numbers (LLN) and the assumption of i.i.d. base learners. However, when individual trees overfit, this assumption weakens, and the ensemble can still overfit if not properly regularized.



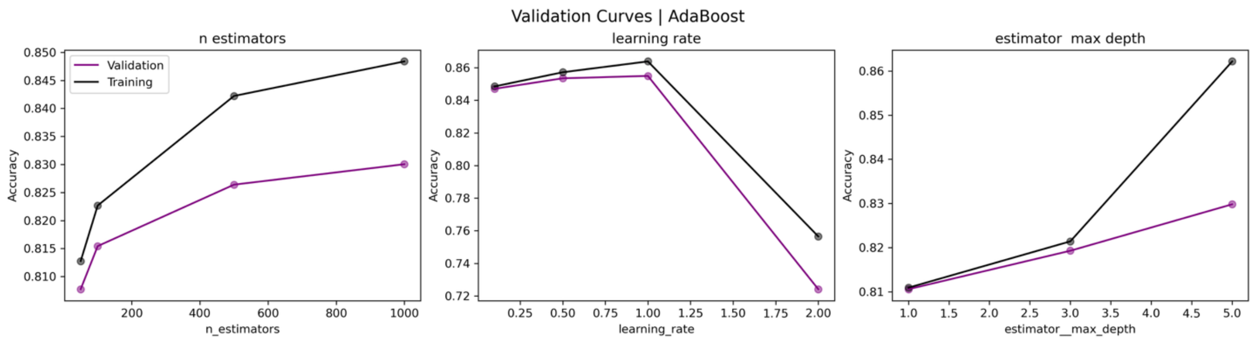
We can clearly see in the two plots below that as max_depth increases, and min_sample_leaf decreases we see a larger difference between the training and validation!



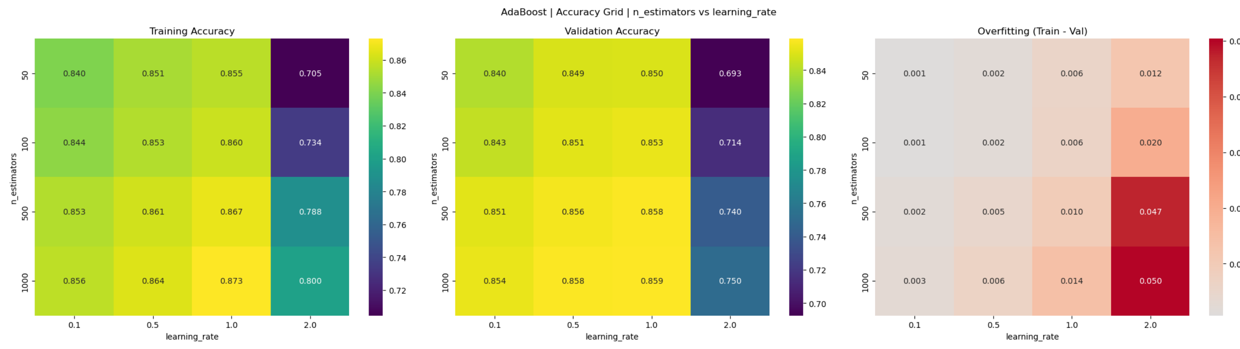
Furthermore, we can see that as $max\_features$ increases and max_depth increases we see the training and validation accuracy diverge supporting the claim the model is overfitting.



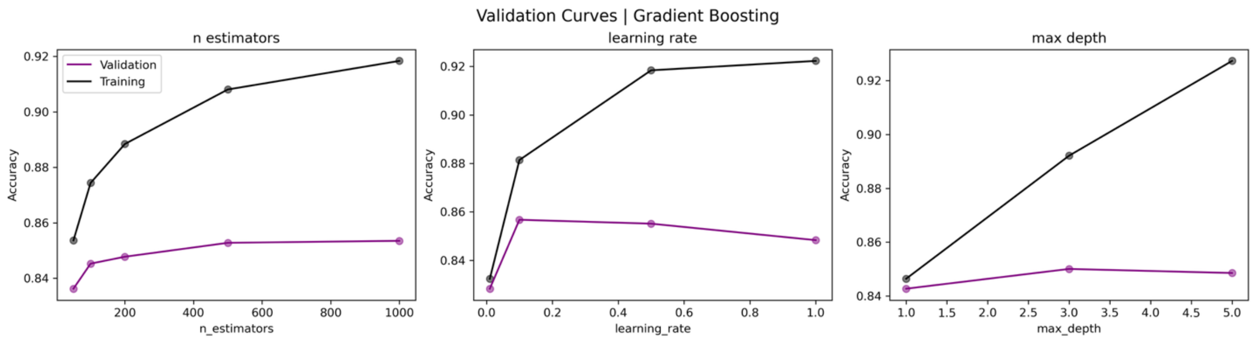
■ **AdaBoost**: With AdaBoost it was difficult to overfit the default model using decision trees with $max\_depth = 1$ as the base learner. Therefore, I also adjusted the hyperparameter max_depth of the base learner – Decision Tree. It is worth noting that the default model for AdaBoost was rather tricky to overfit! Below we can see that after adjusting the base learner as $n\_estimators > 50$, $learning\_rate > 1.00$ and $max_{depth} > 3$ we see overfitting, especially as we increase the max_depth of the base learner, the model learns far too fast and overfits early on in the training process defeating the purpose of gradually learning by fitting the residuals.



It is clear that the max_depth contrbibutes to overfitting but if we see the heatmap below it also is evident that as the $learning\_rate$ increases and $n\_estimators$ increase overfitting occurs!

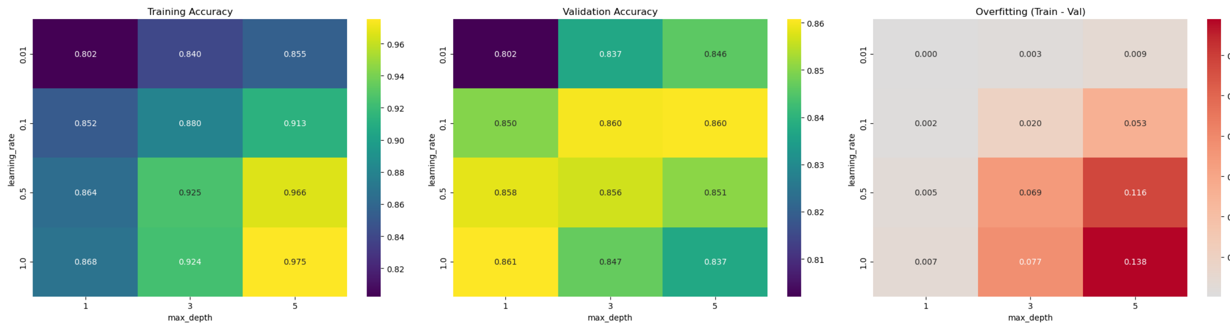


■ **Gradient Boosting**: With sklearns Gradient Boosting it was much easier to overfit the default model. Below we can see that as $n\_estimators > 50$, $learning\_rate > 0.05$ and $max_{depth} > 1$ we see overfitting, as the accuracy and validation accuracy significantly diverge – training curve continues climbing and the validation curve levels off or even declines.

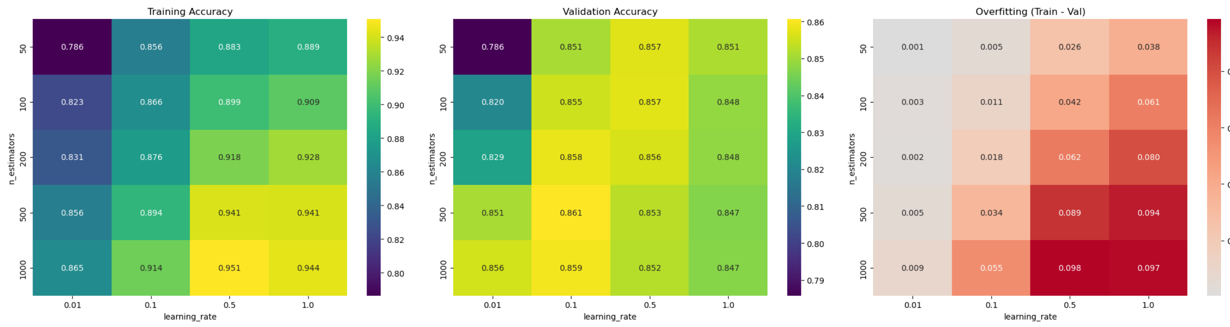


This is further emphasized by the heatmaps below where the darker the red becomes the larger the gap between the training and validation curves become providing a great visualization of the curves across two parameters!

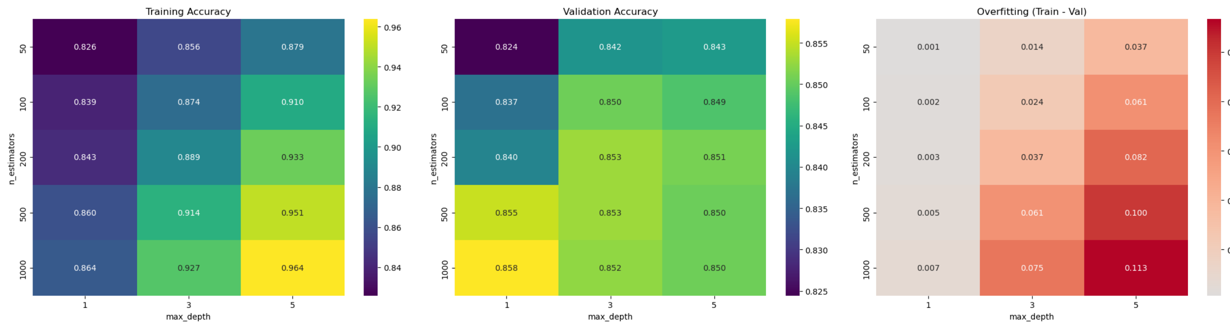
Gradient Boosting | Accuracy Grid | learning_rate vs max_depth



Gradient Boosting | Accuracy Grid | n_estimators vs learning_rate



Gradient Boosting | Accuracy Grid | n_estimators vs max_depth



Part (1Aii) – Report Accuracy & Runtime.

Accuracy Report
(if unspecified assume unbalanced weight for hyperparameter).

Decision Tree (balanced)	Decision Tree	Random Forest	AdaBoost	Gradient Boosting
DecisionTreeClassifier(class_weight='balanced') Grid - Hyperparameters: ['max_depth', 'min_samples_leaf', 'min_samples_split'] **Train Accuracy: 99.997%** **CV Validation Accuracy: 80.731%** **Test Accuracy: 25.903%**	DecisionTreeClassifier() Grid - Hyperparameters: ['max_depth', 'min_samples_leaf', 'min_samples_split'] **Train Accuracy: 85.893%** **CV Validation Accuracy: 85.021%** **Test Accuracy: 85.126%**	RandomForestClassifier() Grid - Hyperparameters: ['n_estimators', 'max_depth', 'min_samples_leaf', 'min_samples_split', 'max_features'] **Train Accuracy: 86.718%** **CV Validation Accuracy: 85.561%** **Test Accuracy: 85.691%**	Best Hyperparameters: {'estimator__max_depth': 5, 'learning_rate': 0.5, 'n_estimators': 1000} AdaBoostClassifier(algorithm='SAMME', estimator=DecisionTreeClassifier()) Grid - Hyperparameters: ['n_estimators', 'learning_rate', 'estimator__max_depth'] **Train Accuracy: 87.202%** **CV Validation Accuracy: 86.413%** **Test Accuracy: 86.786%**	GradientBoostingClassifier() Grid - Hyperparameters: ['n_estimators', 'learning_rate', 'max_depth'] Best Hyperparameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500} **Train Accuracy: 88.190%** **CV Validation Accuracy: 86.486%** **Test Accuracy: 86.932%**

Runtime Report

	# Grid Combinations	CV - k_fold	Models Fit	Runtime (Seconds)	Runtime (Minutes)
Decision Tree	160	5	800	27.4	0.46
Random Forest	432	2	864	269.45	4.49
AdaBoost	48	2	96	487.49	8.12
Gradient Boosting	60	2	120	420.62	7.01

We might expect Gradient Boosting to take longer than AdaBoost due to its gradient-based optimization, but in this case, AdaBoost had a longer runtime. This is because the base learners in AdaBoost were tuned during cross-validation, leading to deeper decision trees that required more computation. Although fewer models were fit, each model was more complex and contributed more significantly to runtime than the shallow stumps typically used in boosting.

All models were run with n_jobs=-1 to utilize all CPU cores, though this benefits methods like Random Forest more directly due to the independence of its base estimators. Boosting methods, being inherently sequential, see more limited speedup from parallelization.