Assignment 4: Random Forest Analysis of Tree Size

Grant Booysen 25849646
Email: 25849646@sun.ac.za

***Abstract*—The report investigates how Random Forest performance varies with decision-tree depth, the number of randomly selected features per split, and ensemble size across three classification datasets of increasing complexity. The goal is to track the performance from underfitting with very shallow trees and few features to overfitting with deep trees and many features, identify points that maximize generalization, and characterize interactions among these hyperparameters. Experiments with (i) maximum depth from minimal to deep trees, (ii) number of features to split on with depth fixed at the best value from (i), and (iii) the balance number of trees with depth were conducted. A final experiment creates ensembles that mix under- and overfitting trees. Models are evaluated with train/test accuracy and 5-fold cross-validation. Findings are consistent: increased depth improves test accuracy up to a point, after which performance degrades or stagnates and the optimal depth increases with problem complexity. With depth fixed, increased number of features per split improves performance up to a plateau where medium complexity favors moderate feature subsets, while complex data benefits from larger subsets. The inclusion of more trees in the ensemble improves stability and accuracy up to a plateau, with gains moderated by tree depth. Bagging with only overfitted trees proved to have either the best stability, cross validation accuracy or test accuracy across all three datasets.**

Overall, the best configurations balance depth, feature subsampling, and ensemble size where larger values are generally preferred as task complexity rises.

***Index Terms*—Random Forest, Ensemble learning, Bagging, Decision Trees**

I. Introduction

Random forests are widely used for classification due to strong generalization, robustness, and modest tuning requirements. The random forest model's performance is controlled by key hyperparameters that trade off bias and variance such as: maximum decision-tree depth, the number of randomly selected features per split, and ensemble size. The report aims to provide understanding for how these controls move the model to better performance. The idea is to identify configurations that generalize across problem complexities.

The goal of this study is to characterize how tree depth and feature subsampling influence random forest performance, and to identify configurations that maximize test accuracy and cross-validated performance across datasets of increasing difficulty. The analysis is achieved by: (i) exploring maximum depth from minimal to deep trees at fixed ensemble size, number of random features per split, and bagging; (ii) fixing depth at the best value from (i) and exploring the optimal number of features to split on per node; (iii) exploring depth and ensemble-size interactions; and (iv) composing ensembles that intentionally mix underfitted and overfitted trees. Evaluation uses train/test accuracy and 5-fold cross-validation to better assess generalization and stability.

The report aims to provide a reproducible behaviour for the bias-variance trade-off for random forests, and to extract generalizations for hyperparameter selection that adapts to dataset complexity. The design follows controlled experiments to reveal how each hyperparameter influences performance, and how they interact. The analysis spans three synthetic datasets of increasing complexity to assess how optimal configurations shift with task difficulty.

The main observations are consistent with the bias-variance expectations. Increasing the depth of the trees in the ensemble improves test accuracy up to a point, after which gains stagnate or degrade; the optimal depth increases with complexity, for example, approximately 8, 15, and 20 on the simple, medium, and complex datasets respectively. With depth fixed, increasing the number of features to split on improves performance then plateaus; moderate subsets suit medium complexity, while larger subsets benefit complex data. Adding trees improves stability and accuracy up to a plateau, where the number of trees needed to reach the plateau is dependent on the depth of the trees. Mixed-depth and only overfit ensembles: on the simple dataset, the highest test accuracy was 0.960 with a tie between the balanced and only overfit compositions, while only overfit attained the best cross-validated score of 0.966; on the medium dataset, the only-overfit composition achieved both the highest test accuracy of 0.685 and the best cross-validated score of 0.691; on the complex dataset, only overfitted trees achieved the highest test accuracy of 0.610 and the most stable cross-validation, with majority overfitted exhibiting a marginally higher CV mean of 0.564 versus 0.563 for only overfit.

The remainder of the report is organized as follows: Section 2 provides background; Section 3 details the methodology and implementation; Section 4 describes the empirical procedure; Section 5 presents results and discussion; Section 6 concludes with implications and future directions.

II. Background

The background section discusses the fundamental concepts and algorithms used in the report, including decision trees, ensemble learning, and random forests.

A. Decision Trees

The decision tree is a supervised learning algorithm used for classification and regression tasks. The decision tree works by recursively splitting the data into subsets based on feature values, creating a tree-like model of decisions. Each internal node represents a condition on a feature, each branch represents the outcome of the test, and each leaf node represents a class label in classification or a continuous value in regression. Decision trees are easy to interpret and visualize, but they can be prone to overfitting, especially with deep trees.

B. Ensemble Learning: Bagging

Bagging, or Bootstrap Aggregating, is an ensemble learning technique that aims to improve the stability and accuracy of machine learning algorithms. The algorithm works by training multiple models on different subsets of the training data, which are created by randomly sampling with replacement known as bootstrapping. The final prediction is made by aggregating the predictions of all individual models, typically by averaging for regression or majority voting for classification. Bagging helps reduce variance and combat overfitting, making the approach particularly effective for high-variance models like decision trees.

C. Random Forests

Random forests are an ensemble learning method that combines multiple decision trees to improve predictive performance and control overfitting. The key idea is to leverage the diversity of individual trees by training them on different subsets of the data and features. Traditionally, random forests use bagging to create diverse training sets for each tree. In random forests, it is expected that each tree should be grown to the maximum depth and overfit. The high variance of each tree is then averaged out by the ensemble, leading to a final model that generalizes well.

Algorithm 1 Random Forest (high-level pseudocode)

```
Require: Training data $(X, y)$, n_estimators, max_depth,
    max_features, bootstrap size
1: for each tree $t = 1 \ldots$ n_estimators do
2:     Draw a bootstrap sample from $(X, y)$
3:     Grow a decision tree with depth limit max_depth
4:     At each node, consider max_features randomly selected
       features
5: end for
6: For prediction, average class probabilities (or majority
   vote) across trees
```

D. Statistical comparison of models

To compare multiple model configurations evaluated on the same data partitions, non-parametric, paired statistical tests are used:

- **Friedman test**: compares k related treatments across N blocks by ranking methods within each block and testing the null hypothesis that their median ranks are equal. It avoids normality assumptions and is appropriate for repeated-measures cross-validation.
- **Wilcoxon signed-rank test**: conducts a paired comparison between two configurations across the same blocks. A one-sided alternative tests whether a reference method such as “Only Overfit” tends to outperform a comparator. Holm correction controls the family-wise error rate over multiple pairwise tests.

These tests operate on cross-validated fold scores, respect pairing, and avoid using the held-out test split for model selection.

III. Implementation

The implementation section describes the datasets, data preparation, and the random forest model used in the experiments.

A. Data preparation

Three synthetic, tabular classification datasets of increasing complexity were programmatically generated using `make_classification` from `sklearn.datasets` [Pedregosa et al., 2011] to enable controlled bias-variance studies and avoid confounds from missing values or categorical encoding. A single `random_state` ensured reproducibility for generation and splitting.

Datasets:

- Simple: binary classification, low noise, no overlap between classes, `n_samples=500`, `n_features=10` where eight are informative and two are redundant, `class_sep=2.0`, `flip_y=0.0`. Designed to saturate quickly with shallow trees and show limited overfitting.
- Medium: three classes, moderate overlap, mild noise, `n_samples=2000`, `n_features=25` where 18 are informative and five are redundant, `class_sep=1.0`, `flip_y=0.2`. Targets visible bias-variance trade-offs at medium depths.
- Complex: four classes, many features, higher noise, `n_samples=6000`, `n_features=50` where 35 are informative and ten are redundant, `class_sep=0.7`, `flip_y=0.3`. Encourages deeper trees and progressive overfitting.

Train/test split: A stratified split with `test_size=0.30` and fixed `random_state` preserves class proportions across splits and guarantees identical partitions across runs.

Preprocessing: No scaling or normalization is applied as decision trees are insensitive to monotonic feature transforms. No imputation is required as the synthetic data contain no missing values. No label encoding is needed as the labels are already integers.

Depth ranges for experimental control: Per-dataset depth grids are predefined to bound sweeps and reflect expected capacity needs:

B. Random forest Model

A custom scikit-learn compatible estimator was implemented by extending `BaseEstimator` and `ClassifierMixin`. Each base learner is a `DecisionTreeClassifier`. The ensemble exposes the hyperparameters: `n_estimators`, `max_depth`, `max_features` (per-split random subspace), `bootstrap`, `criterion`, `min_samples_*`, `max_leaf_nodes`, and `ccp_alpha`. These parameters allow control over tree complexity and randomness. Each individual tree needed to be able to be individually controlled to allow for the mixed underfit/overfit experiments.

Training uses standard bagging: for each tree, a bootstrap sample is drawn. A distinct, reproducible random seed is generated per tree from a generator seeded by the estimator's random seed. Each tree `max_features` parameter enforces feature sub-sampling at each split, mirroring the random subspace mechanism. Prediction averages class probabilities across trees uses majority vote.

The model has the following characteristics:

- Equal-weight averaging across trees with no out-of-bag estimates or class weighting by default.
- No post-hoc pruning as overfitting is intentionally permitted to study depth effects.
- Deterministic behavior controlled by a random seed.

C. Cross-validation procedure

Across all experiments, performance for each hyperparameter setting was estimated with K-fold stratified cross-validation on the training split only using `StratifiedKFold` from `sklearn.model_selection` [Pedregosa et al., 2011], reporting the mean and standard deviation of accuracy. Concretely, for the tree-depth, feature-selection, and depth by trees grids, stratified folds for classification were used to compute cross-validated accuracy scores, and record CV mean and CV std. The held-out test split is evaluated separately to provide a single final generalization number, but not to select hyperparameters. Where an optimal depth is required to center configurations, the depth is chosen by maximizing the CV mean from the prior depth experiment.

D. Statistical testing protocol

For each dataset, repeated stratified K-fold cross-validation with 5 folds and 7 repeats provides fold-wise accuracies for each of the five ensemble compositions. The Friedman test serves as an omnibus check for any differences among configurations. When significant, one-sided Wilcoxon signed-rank tests compare the "Only Overfit" configuration against each alternative, with Holm-adjusted p-values to control family-wise error. Unless stated otherwise, the significance level is set to $\alpha = 0.05$, and post-hoc rejections are determined by comparing Holm-adjusted p-values against α.

IV. Empirical Procedure

The emperical procedure section details the datasets, performance measures, control parameters, and the four experiments executed to answer the study goals. All experiments are fully reproducible given the seeds and grids below.

A. Benchmarks and Splitting

The three datasets Simple, Medium, and Complex were generated as described in the Data Preparation section. Each dataset was split into a training set at a 70% split and a held-out test set at a 30% split using a stratified split to maintain class proportions.

B. Performance Measures and Validation

The primary metric used was accuracy and was reported for train, test, and 5-fold stratified cross-validation on the training split. For each configuration, the following are recorded: train accuracy, test accuracy and CV mean and standard deviation.

C. Implementation and Controls

A global seed of 42 was used. Per-tree seeds are drawn from a generator seeded by this value to ensure determinism. Unless varied explicitly, defaults for the decision trees are:

- Criterion for split quality is the `gini` impurity.
- Minimum samples to split a node is two.
- Minimum samples per leaf is one.
- No pruning.
- Bootstrap sampling.
- Equal weighting across trees.

Dataset-specific depth grids: Each dataset has a predefined grid of maximum depths to explore, reflecting expected capacity needs:

- Simple: $\{1, 2, 3, 4, 5, 6, 8, 10, 12\}$
- Medium: $\{1, 3, 5, 7, 10, 12, 15, 18, 20, 25, 30\}$
- Complex: $\{1, 3, 5, 7, 10, 12, 15, 20, 25, 30, 35, 40\}$

D. Cross-validation for Experiments

All experiments use 5-fold stratified cross-validation on the training split (StratifiedKFold). For each hyperparameter configuration we report the cross-validation mean and standard deviation, and rebuild models or ensembles within each fold when applicable. The held-out test split is evaluated separately to report train and test accuracy. The same reproducible controls (global random seed, per-tree seeds, stratified splits) apply across all experiments.

E. Experiment 1: Maximum Tree Depth

The objective of Experiment 1 is to isolate the effect of maximum tree depth on generalization and overfitting. To this end, we sweep the maximum depth from very shallow to very deep trees while holding other factors constant: a forest of decision trees is trained and all features are used at each split so that each tree is as strong as possible. The variable is the maximum depth, swept over the dataset-specific grids defined above.

F. Experiment 2: Number of Features per Split

The objective is to quantify the effect of per-split feature subsampling while holding depth near the optimal values found in Experiment 1. The experiment fixes the maximum depth to the CV optimal values (Simple: `max_depth=8`, Medium: `max_depth=15`, Complex: `max_depth=20`), enables bagging with 30, 60, and 90 trees for the Simple, Medium, and Complex datasets respectively to stabilise results, and sweeps the maximum number of features per split over $\{1,2,3,5,7,10,15,20,25,30,35,40,45,50\}$, skipping values that exceed each dataset's dimensionality.

G. Experiment 3: Depth vs Number of Trees

The objective is to study the interaction between ensemble size and tree depth. In this experiment the number of features considered per split is fixed to `max_features='sqrt'` to provide a moderate level of randomness and bagging is enabled. Tree depth is swept over the dataset-specific depth grids while the number of trees is varied per dataset. Results are visualized as heatmaps of test accuracy on the depth by number of trees grid to quantify how increasing the number of trees reduces variance up to a plateau and how the benefit of additional trees depends on tree depth. Cross-validation mean and standard deviation are reported as described above.

H. Experiment 4: Mixed-Depth Ensembles

This experiment evaluates ensembles that mix underfitting and overfitting trees to probe bias-variance blending. Controls are fixed where the number of features per split is set to the optimal values from Experiment 2, ensemble sizes and per-dataset maximum tree depths use the optima from Experiment 3, and bagging is enabled. Five ensemble compositions are compared:

- all underfitted;
- majority underfitted (75% underfitted, 25% overfitted);
- balanced (50/50);
- majority overfitted (25% underfitted, 75% overfitted);
- all overfitted.

Underfitted depths are sampled from depths below the optimal and overfitted depths from just above the optimal up to the maximum. The goal is to determine whether mixing tree capacities yields improved generalization or stability compared to homogeneous ensembles of the same size.

V. Research Results

This section reports results for all four experiments. For each configuration the cross-validated accuracy mean and standard deviation over 5 folds on the training split and the held-out test accuracy are reported.

A. Experiment 1: Maximum Tree Depth

Across datasets, increasing maximum depth initially raises both CV and test accuracy as shallow ensembles underfit. Beyond dataset-specific optima, test/CV curves plateau. The figures highlight this bias-variance trade-off and localize the depth that maximizes generalization for each dataset.

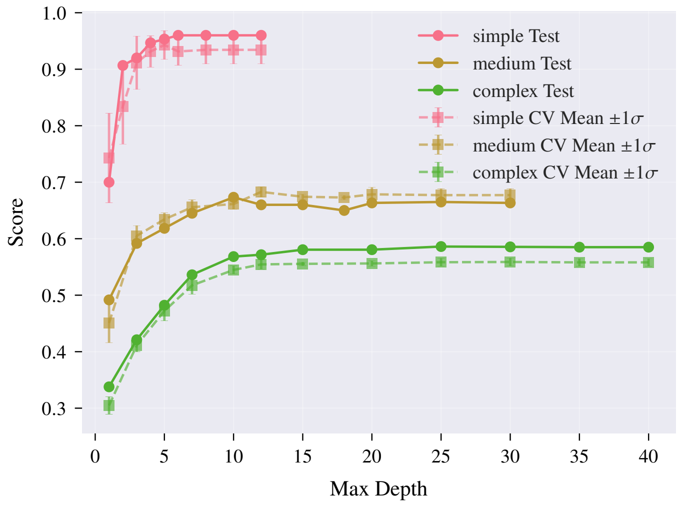
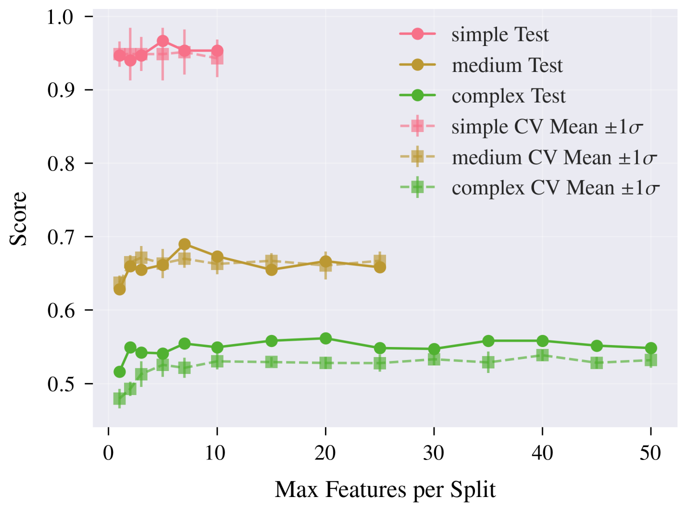


Fig. 1: Maximum tree depth vs. test and cross-validated accuracy for the three datasets.

Discussion: The results in Figures 1 show the relationship between maximum tree depth and both test and cross-validated accuracy. As expected, very shallow trees in the ensemble underfit the data, which results in low accuracy on both training and test sets. As depth increases, accuracy improves significantly, indicating that the model is capturing more complex patterns in the data. However, beyond a certain depth for example, depth 8 for Simple, 15 for Medium, and 20 for Complex, the test accuracy plateaus or slightly decreases while training accuracy continues to rise. This indicates overfitting, where the model learns noise in the training data rather than generalizable patterns. The optimal depths identified align with the complexity of each dataset, with more complex datasets requiring deeper trees to capture their structure. These results occur due to the bias-variance trade-off inherent in decision trees. Shallow trees have high bias and low variance, leading to underfitting, while deep trees have low bias and high variance, leading to overfitting. The ensemble averages out some the variance of the overfitted individual trees. However, if there are not enough trees in the ensemble of overfitted trees, the variance does not average out sufficiently, leading to the seen degradation in test performance. With more overfitted trees in the ensemble, the variance would average out more effectively, potentially allowing for deeper trees to be used without overfitting.

B. Experiment 2: Number of Features per Split

Varying the number of features per split yields modest gains that plateau: allowing more features strengthens individual trees but increases inter-tree correlation, limiting bagging's variance reduction. Accuracy typically improves up to a mid-range (often near $\sqrt{d}$) and then remains flat or slightly declines. CV and test trends align, indicating limited sensitivity once depth is near-optimal.



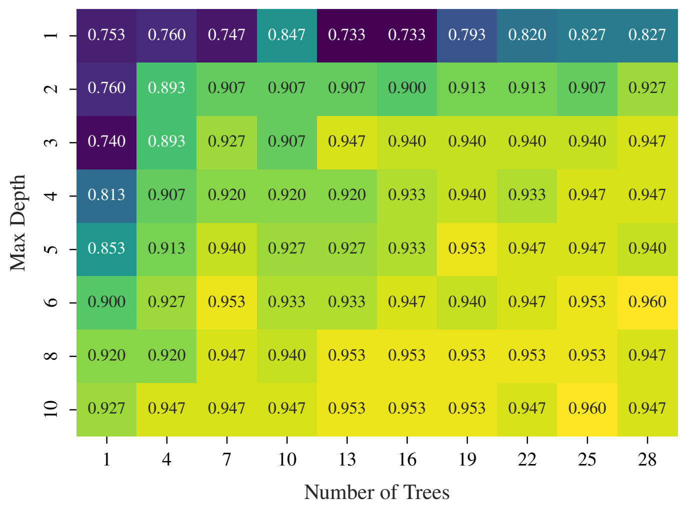
(a) Medium: Test and CV accuracy vs. max_features.

Fig. 2: Number of features per split vs. accuracy (example).

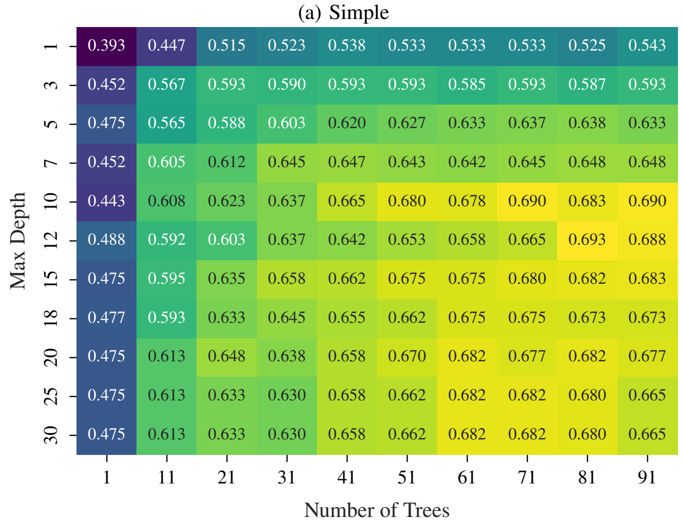
Discussion: The results in Figure 2a illustrate how varying the number of features considered at each split affects test and cross-validated accuracy. As the number of features increase, there is only a slight improvement in accuracy up to a point, after which the performance does not show any clear signs of improvement. Since each individual tree was allowed to grow deep, the benefit of considering more features at each split gets mitigated by the increased correlation between trees. The correlation reduces the effectiveness of bagging, as the ensemble relies on diverse predictions from its members to reduce variance. Therefore, while increasing the number of features per split can enhance the strength of individual trees, it also increases their similarity, which can limit the ensemble's overall performance. Overall, the findings highlight the nuanced role of feature subsampling in random forests, where both too few and too many features can hinder performance due to underfitting or over-correlation among trees.

C. Experiment 3: Depth vs Number of Trees

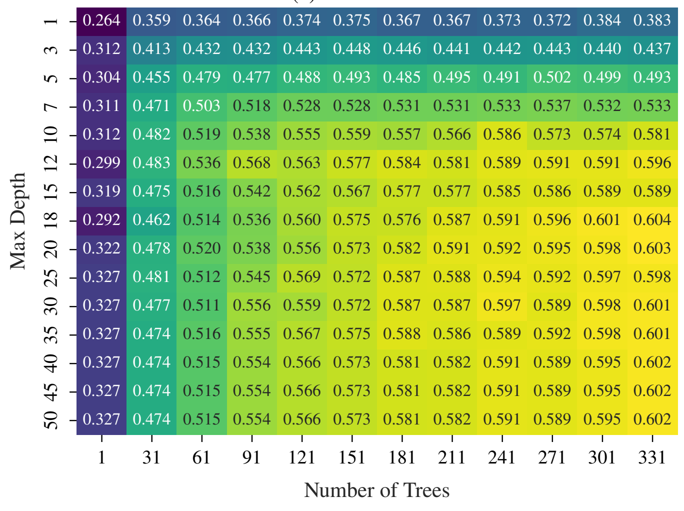
The heatmaps show a clear interaction between depth and ensemble size: deeper trees require more trees to stabilize variance, while extra trees deliver diminishing returns once variance is averaged out. Accuracy concentrates along a ridge where both parameters are sufficient; increasing one without the other beyond this region offers little benefit, marking natural plateaus.



(a) Simple



(b) Medium



(c) Complex

Fig. 3: Heatmaps of test accuracy over maximum tree depth vs. number of trees for each dataset.

Discussion:
Each heatmap in Figure 3 has the same pattern - there is a

clear relationship between the number of trees in the ensemble and the maximum depth of the trees in the forest. Each heatmap shows a convergence to an optimal configuration. The forest created is dependent on the max depth as seen in Experiment 1, but the performance is also dependent on the number of trees. The results show that even when increasing the max depth, the best results come from increasing the number of trees in conjunction. The general trend shows that more trees in the ensemble and deeper trees both contribute to higher accuracy, which is expected in the case of bagging. The improvement in accuracy with more and deeper trees is due to the overall reduction in variance that bagging provides where, although individual trees may overfit, the ensemble averages out their errors.

a) Fixed optimal configurations after Experiment 1-3.: To standardize subsequent comparisons, the following dataset-specific settings were fixed based on the first two experiments and the Experiment 3 ridge regions:

- `optimal_estimators` = {`simple`: 25, `medium`: 100, `complex`: 360}
- `optimal_depths` = {`simple`: 8, `medium`: 10, `complex`: 20}
- `optimal_num_split` = {`simple`: 5, `medium`: 7, `complex`: 15}

These settings are used in Experiment 4 to define the underfit/overfit depth ranges and to hold constant the ensemble size and per-split feature counts.

D. Experiment 4: Mixed-Depth Ensembles

The tables compare ensembles mixing underfitted and overfitted trees. Across Simple, Medium, and Complex, the overfit-only composition attains the best test and CV accuracy, indicating that under strong bagging, averaging many high-variance trees outperforms mixing capacities.

TABLE I: Experiment 4 (Simple): Composition vs. accuracy.

Composition	Train Acc	Test Acc	CV Acc
Only Underfit	0.991	0.940	0.943 ± 0.016
Majority Underfit	0.991	0.933	0.954 ± 0.028
Balanced	1.000	0.960	0.946 ± 0.031
Majority Overfit	1.000	0.940	0.963 ± 0.021
Only Overfit	**1.000**	**0.960**	$\mathbf{0.966 \pm 0.025}$

TABLE II: Experiment 4 (Medium): Composition vs. accuracy.

Composition	Train Acc	Test Acc	CV Acc
Only Underfit	0.969	0.655	0.669 ± 0.025
Majority Underfit	0.984	0.677	0.661 ± 0.028
Balanced	1.000	0.660	0.677 ± 0.019
Majority Overfit	1.000	0.652	0.672 ± 0.026
Only Overfit	**1.000**	**0.685**	$\mathbf{0.691 \pm 0.021}$

Discussion: The results in Tables I, II, and III show the performance of ensembles with different compositions of underfitted and overfitted trees across the three datasets. A clear trend emerges where ensembles composed entirely of overfitted trees consistently achieve the highest test accuracy and cross-validated accuracy. The trend suggests that, in the context of bagging, the high variance of overfitted trees is effectively averaged out by the ensemble, leading to better generalization. Mixed compositions do not outperform the all-overfit configuration, indicating that the benefits of averaging high-variance learners outweigh the potential advantages of bias-variance blending through mixed depths. The results are somewhat surprising, as one might expect that a mix of underfitted and overfitted trees would provide a better balance of bias and variance, leading to improved generalization. However, the effectiveness of bagging in reducing variance appears to be sufficient to allow overfitted trees to perform best. With weaker bagging or fewer trees, mixtures may outperform overfit-only. Therefore, it is essential to ensure that there are enough trees within the ensemble to ensure robustness. In another approach, weaker learners could be used exclusively, such as shallow trees, and follow the boosting approach of sequentially correcting errors. However, this would be a different ensemble method and not bagging.

TABLE III: Experiment 4 (Complex): Composition vs. accuracy.

Composition	Train Acc	Test Acc	CV Acc
Only Underfit	1.000	0.576	0.541 ± 0.013
Majority Underfit	1.000	0.583	0.550 ± 0.015
Balanced	1.000	0.595	0.550 ± 0.026
Majority Overfit	1.000	0.597	0.564 ± 0.021
Only Overfit	**1.000**	**0.610**	$\mathbf{0.563 \pm 0.008}$

E. Omnibus and post-hoc statistical tests

To test for differences among the five ensemble compositions on each dataset, a Friedman test was applied to 35 paired cross-validation blocks, followed by Wilcoxon signed-rank tests which were one-sided and Holm-corrected, comparing the "Only Overfit" configuration against each alternative. The omnibus Friedman results were: *simple*: $\chi^2 = 21.122$, $p = 2.995 \times 10^{-4}$; *medium*: $\chi^2 = 69.402$, $p = 3.035 \times 10^{-14}$; *complex*: $\chi^2 = 82.083$, $p = 6.302 \times 10^{-17}$. These indicate statistically significant differences among configurations. Detailed pairwise outcomes are reported in the accompanying notebook.

TABLE IV: Post-hoc Wilcoxon vs "Only Overfit" — *simple* (35 blocks). One-sided alternative tests whether the reference outperforms the comparator; Holm-adjusted p-values control FWER.

Comparator	W	p	Holm p	Reject	Mean Δ Acc
Only Underfit	304.00	0.00050	0.00199	True	0.01470
Majority Underfit	153.50	0.03428	0.10280	False	0.00730
Balanced	243.50	0.04017	0.10280	False	0.00570
Majority Overfit	179.00	0.46420	0.46420	False	0.00120

a) Discussion of post-hoc results.: On the *simple* dataset, "Only Overfit" is significantly better than "Only

TABLE V: Post-hoc Wilcoxon vs "Only Overfit" — *medium* (35 blocks). One-sided alternative; Holm-adjusted p-values.

Comparator	W	p	Holm p	Reject	Mean Δ Acc
Only Underfit	627.00	0.00000	0.00000	True	0.03520
Majority Underfit	541.50	0.00000	0.00000	True	0.02130
Balanced	562.50	0.00000	0.00001	True	0.01710
Majority Overfit	363.50	0.00349	0.00349	True	0.00950

TABLE VI: Post-hoc Wilcoxon vs "Only Overfit" — *complex* (35 blocks). One-sided alternative; Holm-adjusted p-values.

Comparator	W	p	Holm p	Reject	Mean Δ Acc
Only Underfit	595.00	0.00000	0.00000	True	0.02860
Majority Underfit	610.50	0.00000	0.00000	True	0.01950
Balanced	581.00	0.00001	0.00001	True	0.01150
Majority Overfit	519.50	0.00040	0.00040	True	0.00770

Underfit" after Holm correction, with Holm $p = 0.001989$, while gains over the other three compositions are not statistically significant where mean accuracy deltas are small ($\approx 0.1\%$–1.5%). The other configurations are not significantly different from each other because the problem is small and relatively easy. Trees saturate quickly, and their structures become highly similar across compositions. Strong bagging further reduces variance, so fold-wise paired differences fall within sampling noise. Refer to Table IV for details.

On the *medium* dataset, "Only Overfit" significantly outperforms all four alternatives under the one-sided Wilcoxon with Holm adjustment, with Holm $p < 0.004$, with effect sizes from $\approx 0.9\%$ to 3.5%, indicating a robust advantage as task complexity increases. Refer to Table V for the medium dataset details.

On the *complex* dataset, "Only Overfit" significantly outperforms all four alternatives with p-values less than 0.05 for all comparisons, corroborating the Friedman omnibus and aligning with the Experiment V-D test/CV summaries. Refer to Table VI for details.

VI. Conclusion

The assignment investigated how random forest performance is controlled by maximum tree depth, the number of features considered per split, and ensemble size across three datasets of increasing complexity. A reproducible pipeline was implemented with stratified splits, per-tree seeding, and 5-fold cross-validation, and four experiments were conducted: (i) depth sweeps, (ii) feature-subsampling per split sweeps at near-optimal depth, (iii) depth-by-trees interaction grids, and (iv) ensembles mixing under- and overfitting trees.

The main findings are consistent with bias-variance theory and answer the following:

- Maximum depth: Test and cross-validated accuracy increase from shallow to moderate depths, then plateau or decline as overfitting emerges. The depth that maximizes generalization increases with task complexity. With sufficiently many trees, deeper models become more viable because bagging reduces variance.
- Features per split: Allowing more features strengthens individual trees but raises inter-tree correlation. Accuracy improves to a plateau, after which gains are marginal or flat.
- Depth vs. ensemble size: There is a trend for better performance where both depth and number of trees are sufficiently large enough and tuned to the dataset complexity. Deeper trees require more trees to stabilize variance. The parameters are limited by each other, where increasing one without the other beyond a certain point yields little benefit.
- Mixed-depth ensembles: Under strong bagging, ensembles composed solely of overfitted trees matched or outperformed mixed-capacity ensembles on test and CV accuracy, with comparable or smaller gaps. When ensembles are small or bagging is weak, mixing capacities may be more beneficial.

The following practical suggestions emerges from these results: (i) pick depth by maximizing CV performance; (ii) grow the number of trees until CV mean stabilizes and its variance shrinks, noting that deeper forests require more trees; and (iii) when using sufficiently many trees with bagging, homogeneous deep ensembles are a strong baseline.

Limitations and future work: the study used synthetic, balanced datasets and focused on accuracy where if extending to real-world datasets, additional metrics, and class imbalance would broaden conclusions. Secondly, exploring other ensemble methods like boosting or stacking could reveal different dynamics. Finally, investigating interactions with other hyperparameters such as minimum samples per leaf or pruning would provide a more comprehensive understanding of random forest tuning. Overall, the best configurations balance depth, feature subsampling, and ensemble size, with preferred values growing with task complexity. Under strong bagging, larger ensembles of deeper trees generally deliver the most stable and accurate performance.

The associated code for all the calculations. [1].

Appendix

List of Acronyms

Acronym	**Definition**
CV	Cross-validation

References

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

[1]The code for this report is available on GitHub: https://github.com/grantxxcoder/25849646rw441Assignment4.