Random Forests: Why bagging works + feature randomness

Tutorial Prepared By: Syed Sohail

Github: https://github.com/syedamirsohail467-jpg/ML-Project

1. What a random forest is actually doing

A random forest is a combination of decision trees, with its trees being trained on a bootstrap resample of the training data, and most importantly, every split in every tree is restricted to using a random subset of features. There is majority vote (classification) and averaging (regression) prediction. In the original framing by Breiman, the generalisation performance of the forest is determined by two antagonistic forces; the strength of individual trees and their correlation to each other. The random components of the algorithm are only necessary to ensure that correlation is low without playing havoc with individual-tree signal (Palimkar et al., 2021).

2. Why bagging reduces variance (the core idea)

The idea of bagging (bootstrap aggregating) involves the initial high-variance base learner (a deep decision tree), and averaging both many versions of it that are trained independently. The most important consequence is reduction of variance: averaging neutralizes the idiosyncratic variation of any particular tree which is due to noise in sampling. Assuming that the individual predictor trees have average variance (sigma 2) and that any two trees are correlated by (rho) then the average variance of (B) trees is.

$$\mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B} T_b\right) = \sigma^2\left(\rho + \frac{1-\rho}{B}\right).$$

This statement reveals the trade-off in the engineering. Incr A larger B (more trees) pushes the (1-rho)/B term towards zero generating diminishing returns when (B) is large. Nonetheless, it does not disappear in the term of (rho): in case trees are strongly correlated, averaging cannot eliminate such a common variation. There is therefore maximum strength of bagging in the case where the base learner is correct but unstable (high variance) and where the ensemble process is set to induce low correlation among its members. Bootstrap resampling is another practical method of producing those multiple versions of the predictor as the original bagging paper suggests (Wu and Chang, 2024).

3. Bootstrap sampling and the out-of-bag (OOB) idea

The tree of bagging or random forests is trained on bootstrap (n) of the (n) training cases with replacement. One of the usual consequences of this is that the probability of a certain training example being not picked in a single bootstrap sample is roughly (1-1/n)n) = -1 =

0.368. That is, any given tree has an out-of-bag proportion of the observations of the training of (36.8) percent.

The fact is exploited by out-of-bag estimation to obtain an implicit validation estimate without a validation split. At every training point (i), only the subspace of trees which did not train on (i) are used to predict (i). An average of those predictions gives an OOB prediction of each training point, and the comparison of such OOB predictions to the actual labels provides an OOB error estimate. Breiman defined OOB estimation as an ideal method to estimate both test error and other values utilizing the bootstrap structure.

OOB estimation is revealed in scikit-learn as oob score=True (bootstrapping is on). Another warning that the library gives is that when the forest is too small some of the points might never be missed, and OOB predictions will be undefined at those points. The intuitive sense of it is that OOB is reliable only after n estimators has gotten large.

4. Why random forests beat plain bagging: reducing correlation with feature randomness

The advantage of bagging is that it averages variance, but does not directly regulate correlation. Correlations in decision trees are especially vulnerable to the greediness of decision trees: when a single feature is highly predictive, many decision trees will use it towards the root, and their descendant structure will resemble even when perturbed by bootstrapping. Random forests accomplish this by itself by adding randomness to the features that can be used at a split. This is usually defined by the hyperparameter of max features (also referred to as mtry) (Palimkar et al., 2021).

The key idea, which was already in Tin Kam Ho with his random subspace concept - where trees are built in randomly chosen feature subspaces - is that instead of viewing all the feature space, we can just view a part of it, and this can enhance generalisation as an ensemble. (Random Forests in the ACM Digital Library, 2006) Random forests use the related notion on a narrower scale: at each split, a new random subset of features is selected, which further minimises the correlation.

This has been formalised in the random forest paper by Breiman as a strengthcorrelation trade-off: forests perform well with strong trees whose pairwise correlation is low. Reducing max features will normally decrease correlation (good in variance), however individual trees may be hindered (possibly augmenting prejudice). Increasing the value of max features makes trees stronger and more correlated so that the advantage of averaging is reduced.

5. Variance reduction in practice: what to expect as forests get larger

The performance of the test usually reaches its maximum at an early stage with an increase in the number of estimators, and thereafter, it levels off. The above variance formula predicts that such behaviour: when the ensemble average has cancelled the uncorrelated contribution to the tree noise, the subsequent set of trees tends to decrease a very small residual contribution. It is also the reason why random forests tend not to be very sensitive to the exact number of trees beyond a certain threshold, assuming that the compute cost is not excessive (Sanij, Babagoli and Elyasi, 2025).

The OOB error is particularly useful in this case: by visualizing OOB error versus n estimators, one can get a near real time picture of the convergence. The example provided by scikit-learn shows this application in fact: OOB is used to validate in real-time. In practice, the strategy that is often used is to keep adding more and more n est estimators until the OOB curve levels off, and then to cease this process, since additional trees contribute additional compute with negligible predictive value (Wu and Chang, 2024).

6. What `max_features` changes

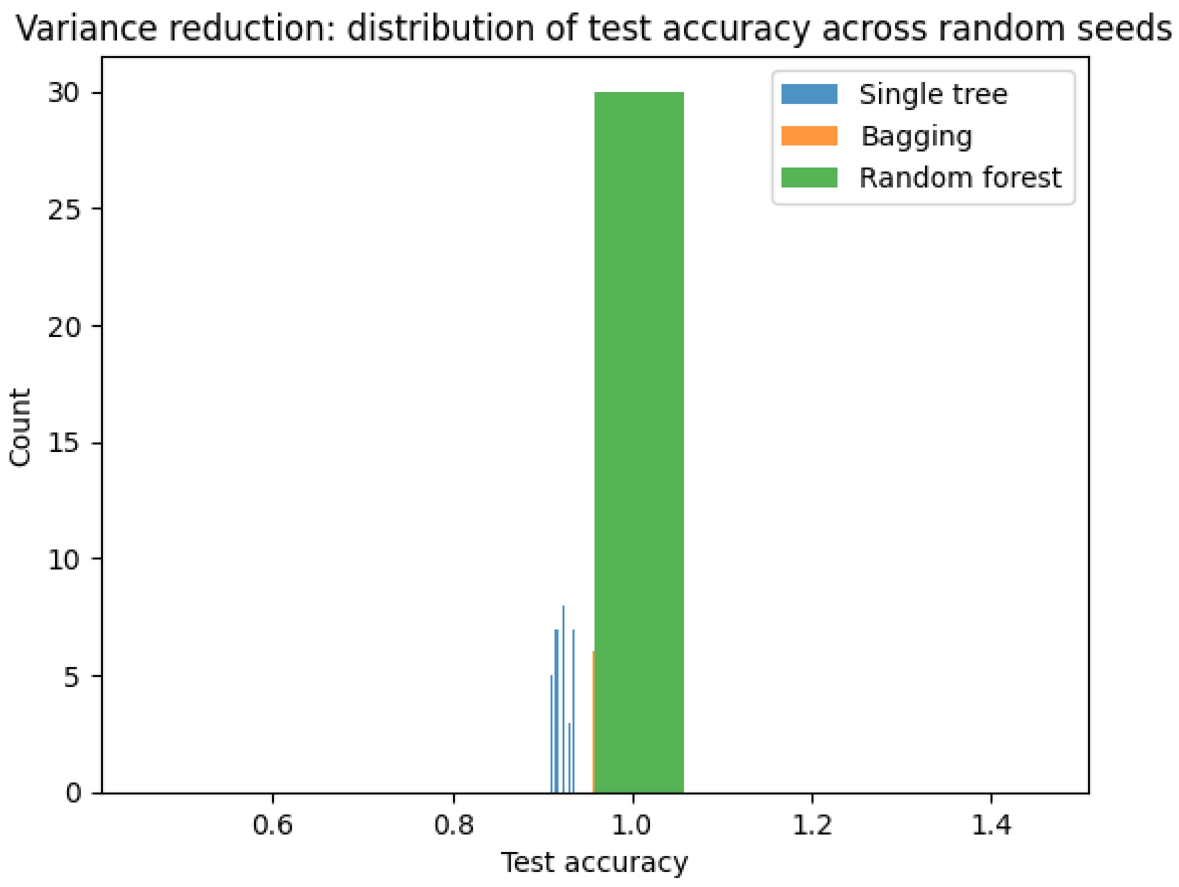
The ‘max features’ parameter regulates the randomness of features. Random Forests subclassifiers in scikit-learn make use of the value max features transparent, and the default values mirror common practice (in classification, it is usually something like (squared root p), in regression, it is often more features). The tune rationale is simply a variation of the strength-correlation narrative:

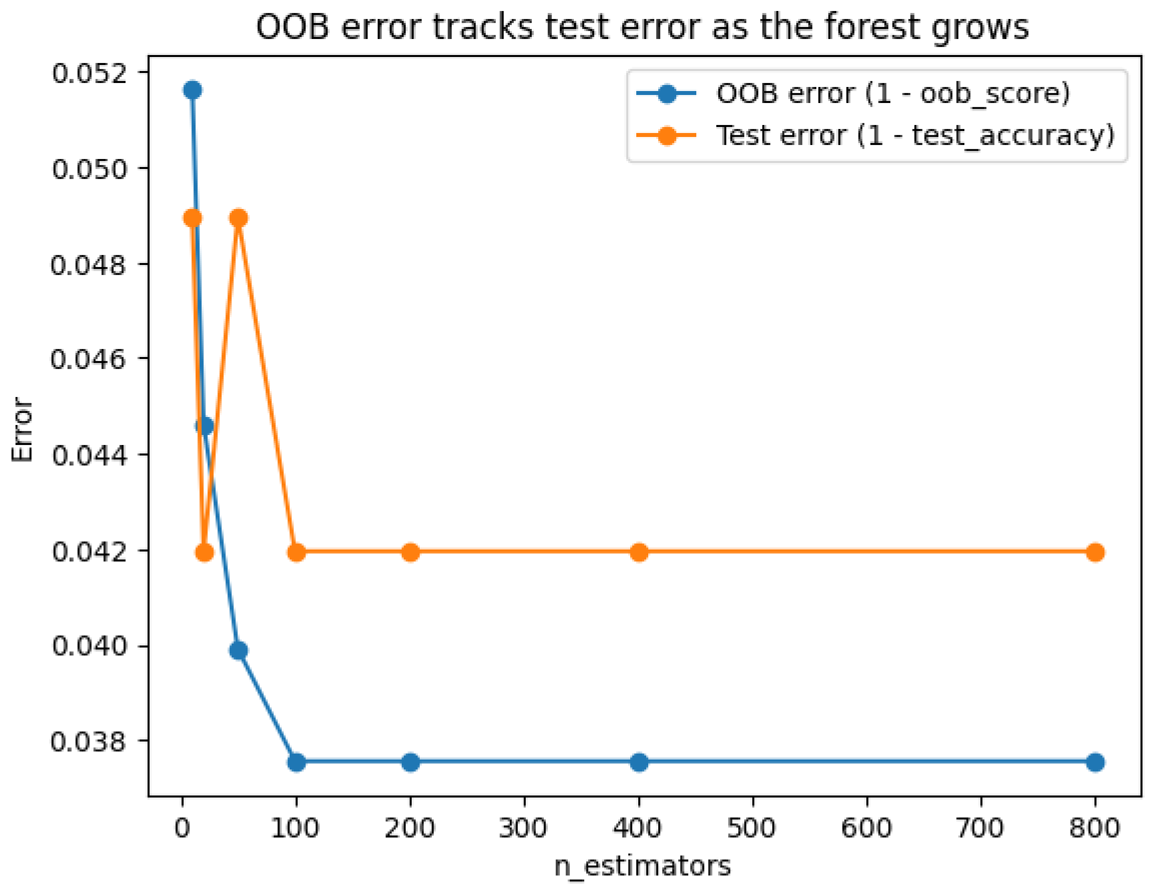
Higher values of max features enhance diversity since the trees will be forced to explore alternative options when making splits, which will decrease correlation and as such, decrease the value of variance in the ensemble (Amiri *et al.*, 2024). This can be useful particularly where redundant features are very many and highly predictive features that would otherwise dominate the premature splits across trees. Nonetheless, when small values are selected as max features, trees can very easily still fail to see the actually informative feature when it counts, so they make weaker splits, and can tend to be more biased. A lower max feature will on the other hand cause the bias to be less since trees can make a better choice more frequently but will cause trees to be more similar, thus the reduction in variance will level off shortly (Wu and Chang, 2024).

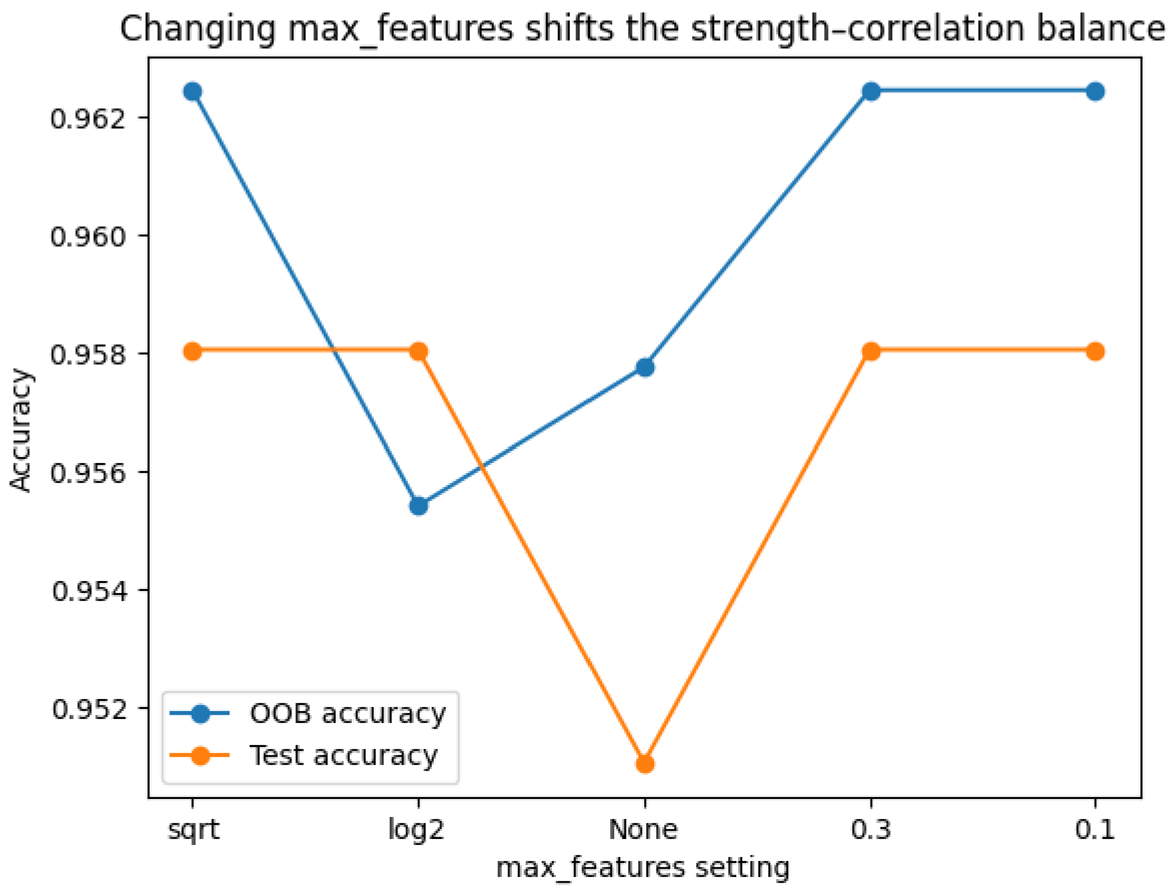
This randomness has also been explained by modern theory as a kind of implicit regularisation, which justifies why random forests can be made robust even when the trees used in them are grown deeply. As an example, Mentch and Zhou propose the complexity perspective of randomisation as regularisation, with the character of regularisation strength related to max feature/mtry (Disha and Waheed, 2022).

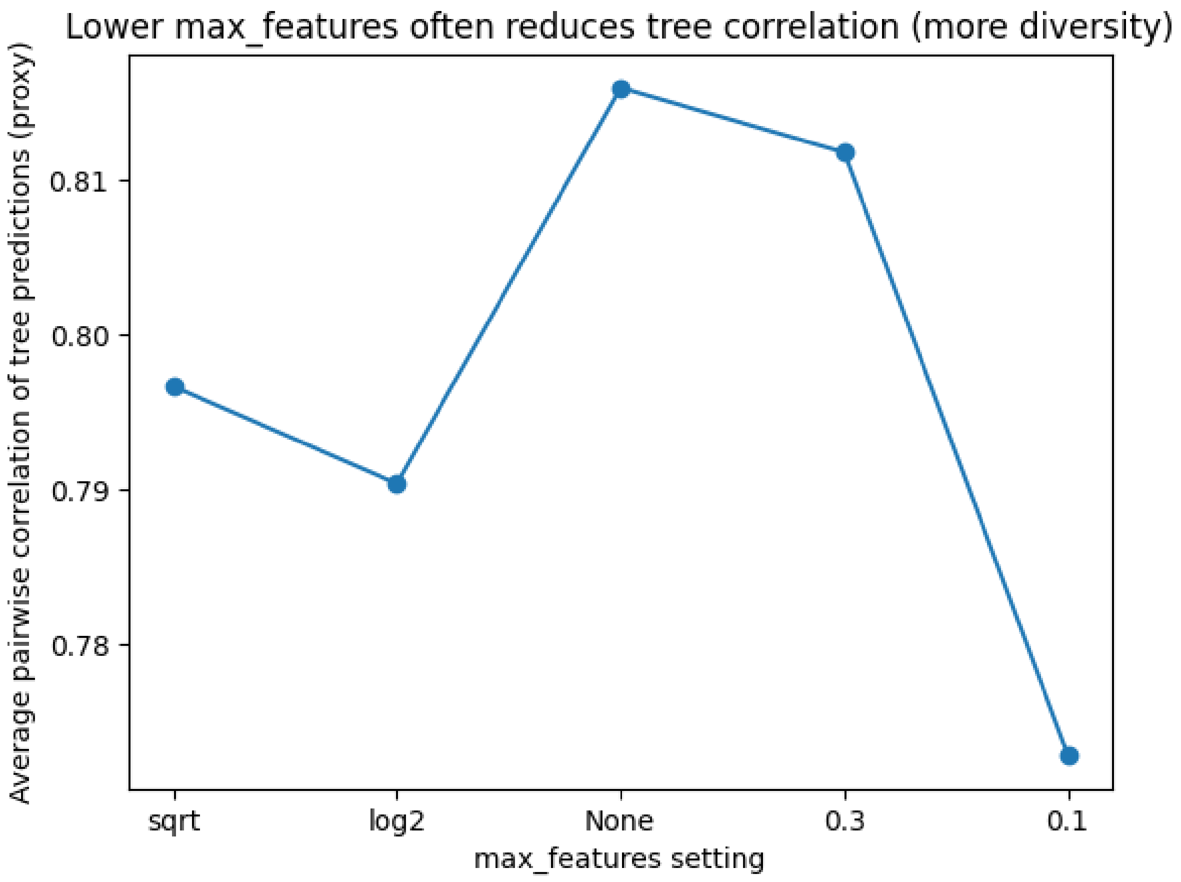
7. Practical tuning rules using OOB error (without a separate validation split)

One use-case that would work in practice is to take OOB error as an internal cross-validation surrogate on the choice of max features and to use an internal cross-validation surrogate on the choice of a large enough n estimators. The first step is to find a reasonably large n estimators (sufficiently large to make OOB sufficiently well-defined at all or most points). Then sweep max features over a small grid and pick the max features value that should minimise OOB error (or maximise OOB score). Lastly, the OOB curve should plateau after increasing the number of estimators with max features fixed (Pellegrino *et al.*, 2021). The trick does not lie in this, OOB remains an approximation and may be noisy on small datasets or when there is a keen class imbalance. However, it is computationally appealing due to its re-use of training and excludes repeated cross-validation fittings on a full basis. Breiman notes OOB and scikit-learn OOB example both show OOB as a key diagnostic of ensemble algorithms based on bootstrap samples (Salman et al., 2024).




OOB error tracks test error as the forest grows
OOB error (1 - oob_score)
Test error (1 - test_accuracy)
Error
n_estimators
0.052
0.050
0.048
0.046
0.044
0.042
0.040
0.038
0
100
200
300
400
500
600
700
800


Changing max_features shifts the strength–correlation balance
OOB accuracy
Test accuracy
Accuracy
max_features setting
0.962
0.960
0.958
0.956
0.954
0.952
sqrt
log2
None
0.3
0.1



8. Limitations and common pitfalls

Random forests may not be able to extrapolate well (trees are piecewise-constant or piecewise-linear based on implementation choices) and may not work with very small signal to noise ratios and some structure of interest that cannot be represented by axis-aligned splits. They too, are prone to misleading by variable importance when correlated attributes share information, since the importance may be divided up between correlated predictors. That is a subject that is out of the scope of this tutorial, although this is one reason that OOB-based importance measures should not be taken at face value (Elfanagely *et al.*, 2021). Implementation wise, OOB scoring needs bootstrapping and sufficient trees that most of the data is out of bag of a sufficient number of trees; failing which OOB predictions are undefined at some points in scikit-learn. Lastly, random forests are known to be computationally expensive; therefore, the increase in the number of estimators should be trade off with runtime and latency limits, particularly in production.

The reason why bagging is effective is that the averaging of a large group of unstable learners reduces variance, but it is not as effective as it would be in an uncorrelated sample. Random forests enhance plain bagging since it adds randomness of features to each node, reducing correlation, and thus, enhancing the reduction in variance that averaging can accomplish. OOB error transforms the bootstrap scheme into an implicit validation scheme, which allows practical tuning of the max-features parameter, and the choice of a large enough n-estimators parameter without a validation set. The combination of these concepts - variance reduction through averaging, correlation control via randomness and OOB-based diagnostics - all make random forests a useful approach to default in many supervised learning tasks (Aria, Cuccurullo and Gnasso, 2021).

References

Adetunji, A.B., Akande, O.N., Ajala, F.A., Oyewo, O., Akande, Y.F. and Oluwadara, G., 2022. House price prediction using random forest machine learning technique. *Procedia Computer Science*, *199*, pp.806-813.

Amiri, A.F., Oudira, H., Chouder, A. and Kichou, S. (2024) ‘Faults detection and diagnosis of PV systems based on machine learning approach using random forest classifier’, *Energy Conversion and Management*, 301, p. 118076.

Aria, M., Cuccurullo, C. and Gnasso, A. (2021) ‘A comparison among interpretative proposals for Random Forests’, *Machine Learning with Applications*, 6, p. 100094.

Disha, R.A. and Waheed, S. (2022) ‘Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique’, *Cybersecurity*, 5(1), p. 1. Available at: https://doi.org/10.1186/s42400-021-00103-8.

Elfanagely, O., Toyoda, Y., Othman, S., Mellia, J.A., Basta, M., Liu, T., Kording, K., Ungar, L. and Fischer, J.P. (2021) ‘Machine learning and surgical outcomes prediction: a systematic review’, *Journal of Surgical Research*, 264, pp. 346–361.

Palimkar, P., Shaw, R.N. and Ghosh, A., 2021. Machine learning technique to prognosis diabetes disease: Random forest classifier approach. In *Advanced computing and intelligent technologies: proceedings of ICACIT 2021* (pp. 219-244). Singapore: Springer Singapore.

Pellegrino, E., Jacques, C., Beaufils, N., Nanni, I., Carlioz, A., Metellus, P. and Ouafik, L. (2021) ‘Machine learning random forest for predicting oncosomatic variant NGS analysis’, *Scientific reports*, 11(1), p. 21820.

Salman, H.A., Kalakech, A. and Steiti, A., 2024. Random forest algorithm overview. *Babylonian Journal of Machine Learning*, *2024*, pp.69-79.

Sanij, H.K., Babagoli, R. and Elyasi, R.M. (2025) ‘Enhancing stone matrix asphalt performance with sugarcane bagasse ash: Mechanical properties and machine learning-based predictions using XGBoost and random forest’, *Case Studies in Construction Materials*, p. e05186.

Wu, Y.C. and Chang, Y.L., 2024. Ransomware detection on linux using machine learning with random forest algorithm. *Authorea Preprints*.