9.2 Ensemble methods

Ensemble methods are a collection of methods that consists of combining "weak" learning methods into a larger more efficient one. Their primary purpose is to reduce model variance by combining individual models, rendering them particularly interesting in the context of regression trees.

9.2.1 Random Forest

Random Forest (RF) is an ensemble method applied on regression trees. Before delving into random forests, it's essential to grasp another widely used ensemble method upon which RFs are based: notably "bagging" - also called "Bootstrap Aggregation". This method consists of building different models and averaging them to obtain a more robust model. The different models are built by sampling, M times with replacement from the dataset, resulting in M datasets upon which M models are built. By training the model on different datasets, bootstrap aggregation is widely assumed to effectively diminish the model's variance. It is thus a suitable method for high-variance and low-bias models such as trees, and it typically performs bad for high-bias models, like for example linear models. In fact, as explained in the Generalized additive model chapter , linear models exhibit high bias asymptotically ; $MSE_{\text{linear}} = \underbrace{\sigma}_{\text{intrinsic error}} + \underbrace{a_{\text{linear}}}_{approximate error} + \underbrace{O\left(n^{-1}\right)}_{estimation error}$; while non parametric methods are unbiased asymptotically, $MSE_{\text{nonpara}} = \underbrace{\sigma^2}_{intrinsic error} + \underbrace{O\left(n^{-4/(p+4)}\right)}_{rate of convergence of estimation error}$, making them ideal for bagging. In addition, another important and more obvious aspect of bagging is that it mitigates the effect of outliers in the data (Grandvalet , 2002[7]). By bootstrapping with replacement from the dataset, outliers are weighted less in the final estimation (Asymptotically). Finally, from a Bayesian point of view, Tibshirani et al. (1997[28]) interpret bootstrap aggregation by describing the distribution resulting from bagging as an "approximate non-informative Bayesian posterior". This result holds asymptotically. In other words, this means that $P(\theta|X = \text{data})$ is approximately obtained by iteratively bootstrapping $P(X = \text{data}|\theta)$, without the need of any prior information.

Random Forest, is an ensemble method applied on regression trees that relies on bootstrap aggregation and a variant of the greedy algorithm. Specifically, M bagged trees are constructed through iterative sampling (with replacement) of subsets from the original data points. At each iteration, a tree is constructed [88] using a greedy algorithm, wherein, at each split, only a subset of features is considered.

Hence, for P total features in the dataset, Random forest method consists of picking randomly $D < P$ features at each split. This feature selection method is motivated by the consideration that

[88] on the bootstrapped subset of data points

bootstrap aggregation alone may not be sufficient: In fact, from a probabilistic point of view, bagging results in M independent and identically distributed models with each a variance σ^2 and a mean μ, the average of these models has a variance

$$\text{Var}\left(\frac{1}{M}\sum_{i=1}^{n} x_i\right) = \rho\sigma^2 + \frac{1-\rho}{M}v^2 \tag{4}$$

(Refer to footnotes for a detailed explanation [89]). This result is a crucial theoretic element in Random Forest's defence:

The result suggests that by increasing the number of bagged trees M one can reduce the variance of the bagged model; however, there will always remain some variance due to the correlation between the bagged trees ρ ;i.e. the model will always exhibit variance asymptotically due ρ.

Random Forest addresses this by randomly selecting a subset of features at each split, aiming to reduce the correlation between the different bagged trees. Asymptotically, using RF results in a zero variance model. However, in practice , the second term in 4 does not cancel out, and hence reducing the correlation would also have some upward effect on the total variance through the second term. Thus, instead of looking for the set of bagged trees with zero (or negative) correlation, the focus should be on finding the optimal balance of correlation that minimizes the total variance of the model. Achieving this balance involves tuning the model's feature selection parameters. And in fact, the number of selected factors is a hyperparameter of random forest method determined through Cross-Validation.

9.2.2 Boosted Trees

I also use boosted trees in this paper. Boosting, is also an ensemble method, where, like for bagging, different weak learners are combined to form a unique model that performs better. Adaptive boosting was first introduced by Freund and Schapire (1997)[4]; Their algorithm aimed to construct a robust model by adaptively combining weak learners. While delving into the intricacies of the algorithm is beyond the paper's scope, grasping its rationale proves beneficial. AdaBoost.M1 iterates M times [90] ; at each iteration, training points are reweighted, and a new model is fitted on the reweighed sample. The new model is scaled, then added to the one fitted in the previous iteration. The algorithm's output is thus the scaled sum of these models. Observations are reweighed based on the associated errors;

[89] That is because $\text{Var}\left(\frac{1}{M}\sum_{i=1}^{M} x_i\right) = \frac{1}{M^2}\text{Var}\left(\sum_{i=1}^{M} x_i\right) = \frac{1}{M^2}\left[\mathbb{E}\left[\left(\sum_{i=1}^{M} x_i\right)^2\right] - \mathbb{E}\left[\sum_{i=1}^{M} x_i\right]^2\right]$; with $\mathbb{E}\left[\sum_{i=1}^{M} x_i\right] = \sum_{i=1}^{M}\mathbb{E}(x_i) = M\mu$; and $\mathbb{E}\left(\left(\sum_{i=1}^{M} x_i\right)^2\right) = \sum_{i,j=1}^{M}\mathbb{E}(x_i x_j) = M\mathbb{E}(x_i^2) + (M^2 - M)\,\mathbb{E}(x_i x_j)$; This can be further reduced by noting that the correlation coefficient of two random variables x_i and x_j ,$\rho_{ij} = \frac{\mathbb{E}((x_i-\mu_i)(x_j-\mu_j))}{\sigma_i\sigma_j}$; is defined as $\rho_{ij} = \frac{E((x_i-\mu)(x_j-\mu))}{\sigma^2}$ for bagged (and hence i.i.d) models x_i and x_j . This correlation formula implies that $E[x_i x_j] = \rho\sigma^2 + \mu^2$. Utilizing this formula, $M\mathbb{E}(x_i^2) + (M^2 - M)\,\mathbb{E}(x_i x_j)$ reduces to $M\sigma^2 + M^2\rho\sigma^2 + M^2\mu^2 - M\rho\sigma^2$. Accordingly, $\text{Var}\left(\frac{1}{M}\sum_{i=1}^{M} x_i\right) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$.

[90] M is a tuning parameter of the algorithm

misclassified observations receive higher weights (They are thus more relevant in the subsequent fit) and model scaling is determined by the training error of the new model on the reweighed training set; higher errors result in lower weights.

In summary, Adaboost.m1 is a greedy adaptive algorithm to construct additive models using simple basis functions, taking into account previous errors at each iteration. This is equivalent to building an additive model $f(\mathbf{x}) = \sum_{m=1}^{M} b_m(\mathbf{x})$, defined as the expansion of some basis function $b_m(x)$; using a forward stage-wise additive algorithm with an exponential loss function. [91]. In essence, boosting methods are all forward stage-wise algorithms as such.

Accordingly, boosted trees are the additive expansion of simple trees (which can be interpreted as basis functions) $f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$, with Θ_m representing the tree parameters (R and c) where $\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$ is iteratively evaluated by forward stage wise algorithm. Using a Squared loss function, the problem reduces to a simple regression tree fitting problem applied on the residual from fitting the previous model rather than the dependent variable [92]. In this paper, I apply a "penalized" version of the boosted model, by scaling each of the added models by a shrinking penalty. The logic remains the same, I simply multiply every added model (at each iteration) by a scalar $v \in (0, 1)$. Tibshirani et al. (2001) suggest that by adding the v term, one could, by analogy with the functional gradient descent method, view the shrinkage parameter as the learning rate (step size) of the gradient descent.

Why Trees I use trees (and related ensemble methods) in this paper for many reasons. First, Generalized additive models, even though, as explained previously, are a good compromise with respect to other non linear models; they may be far fetched in the context of return forecasting as they are global models whose predictive function is the same across all its domain - which suggests that the underlying function is defined as a single function. Furthermore, GAMs pose an additional challenge. Despite their enhanced interpretability due to their additive composition, they abstract the dynamics of the function. Understanding the dynamics from a generalized additive model is nearly impossible [93].

On the contrary, trees offer a distinct advantage. By recursively partitioning the space into different regions, the tree defines different predictive dynamics (functions) across different paths that are easily understandable. Moreover, it is noteworthy that the search for similarities in trees represents a more robust approach compared to the K-Nearest Neighbors method. While K-Nearest Neighbors focuses on interpolating and smoothing based solely on the similarity of features, neglecting the dependent

[91] The forward stage wise algorithm consists of first initializing $f_0(\mathbf{x}) \leftarrow 0$ and then For $m = 1, \ldots, M$ finding the best model $f_m \leftarrow \arg\min_\theta \sum_i L\left(f_{m-1}(\mathbf{x}) + b_m(\mathbf{x}, \hat{\theta}), y_i\right)$ for some loss function $L(.)$ and updating by adding to the previous model $f_m(\mathbf{x}) \leftarrow f_{m-1}(\mathbf{x}) + f_m(\mathbf{x})$

[92] In fact $\hat{\Theta}_m = Argmin(y - f_{m-1}(x) - T_m(x))^2 = Argmin(resid - T_m(x))^2$

[93] And it is worse for Non-parametric smoothing methods where there is no clear distinction of the dynamics between the different features

variable; trees take into account the similarity in both dependent and independent variables. Terminal nodes in trees can be regarded as neighborhoods in the feature space containing datapoints with similar responses; consequently, Shalizi (2021) characterizes trees as "adaptive nearest-neighbor methods." In addition, there are many practical reasons that make tree popular forecasting methods: for example, it is straightforward to see which variables are relevant, it helps making predictions when variables are missing (If one wants to predict but does not have all features used in the tree, he can simply skip the missing information , without loss of generality) , it does not assume true smooth underlying function as the piece wise constant prediction can approximately represent both smooth (approximately), and non smooth true underlying functions , and finally there is no need for calculations to make predictions, one can just look at the tree.